Open Geospatial Consortium Inc.

Date: 2016-12-05

Reference number of this OGC® document: OGC 07-036r1

Version: 3.2.2

Category: OpenGIS® Standard

Editor: Clemens Portele

OpenGIS® Geography Markup Language (GML) Encoding Standard

Warning

OGC official documents use a triple decimal-dot notation (i.e. MM.xx.ss). This document may be identified as MM.xx (Major.minor) and may include increments to the third dot series (schema changes) without any modification to this document, or the version displayed on the document. This means, for example, that a document labelled with versions 1.1.0 and 1.1.1 or even 1.1.9 are exactly the same except for modifications to the official schemas that are maintained and perpetually located at: http://schemas.opengis.net/. Note that corrections to the document are registered via corrigendums. A corrigendum will change the base document and notice will be given by appending a c# to the version (where # specifies the corrigendum number). In corrigendums that correct both the schemas and the base document, the third triplet of the document version will increment and the 'c1' or subsequent identifier will be appended, however the schemas will only increase the third triplet of the version.

This document is an OGC Standard. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporti

Document type: OpenGIS® Standard
Document subtype:
Document stage: Approved
Document language: English

Contents

Page

Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The OGC shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

The Geography Markup Language (GML) was originally developed within the Open Geospatial Consortium, Inc. (OGC). ISO 19136 was prepared by ISO/TC 211 jointly with the OGC.

This version is a corrigendum to GML 3.2.1 (ISO 19136:2007). It addresses the OGC Change Request 12-092 (gml:id attribute on LinearRing) by applying the following changes:

- the XML attribute gml:id in gml:AbstractGMLType has been made optional;
- the elements gml:AbstractRing and gml:Shell have been added to the substitutionGroups gml:AbstractCurve and gml:AbstractSurface respectively;
- the types gml:AbstractRingType and gml:ShellType are now extended from base types gml:AbstractCurveType and gml:AbstractSurfaceType respectively;

These changes correct inconsistencies with ISO 19107 without breaking the validity of instance documents created using the GML 3.2.1 schema. I.e., all GML 3.2 instance documents that are valid against the GML 3.2.1 schema are also valid against the GML 3.2.2 schema.

The corrected GML 3.2 schema is available at http://schemas.opengis.net/gml/3.2.1/. Note that the use of "3.2.1" in the URL is unchanged since this version (3.2.2) is a corrigendum and the corrected schema replaces the GML 3.2.1 schema. Previous versions of the GML 3.2.1 schema are available at http://schemas.opengis.net/gml/gml-3_2_1.zip.

The change to the gml:id attribute reverts a change that has been made between GML 3.1.1 and GML 3.2.1. Reverting this change also addresses comments raised by several communities since the release of GML 3.2.1 / ISO 19136:2007.

As the correction relaxes a constraint in the XML schema, not all instance documents created based on the GML 3.2.2 schema will be valid against the GML 3.2.1 schema:

- all GML 3.2 instance documents that include a gml:id attribute on a ring or shell element are not valid against the GML 3.2.1 schema;
- all GML 3.2 instance documents that include a feature, a spatial object or a temporal object without a gml:id attribute are not valid against the GML 3.2.1 schema.

Local copies of the GML 3.2.1 schema documents have to be replaced by the GML 3.2.2 schema documents – or be replaced by links to http://schemas.opengis.net/gml/3.2.1/gml.xsd.

This corrigendum also updates URIs – mainly in examples – where OGC policies have changed since the release of GML 3.2.1 (location of the Xlink schema document, use of OGC HTTP URIs for coordinate reference systems).

As the corrigendum is currently not published by ISO, the reference to the normative schema documents in Annex C now refers to the OGC schema repository.

Introduction

Geography Markup Language is an XML grammar written in XML Schema for the description of application schemas as well as the transport and storage of geographic information.

The key concepts used by Geography Markup Language (GML) to model the world are drawn from the ISO 19100 series of International Standards and the OpenGIS Abstract Specification.

A feature is an “abstraction of real world phenomena” (ISO 19101); it is a geographic feature if it is associated with a location relative to the Earth. So a digital representation of the real world may be thought of as a set of features. The state of a feature is defined by a set of properties, where each property may be thought of as a {name, type, value} triple.

The number of properties a feature may have, together with their names and types, is determined by its type definition. Geographic features with geometry are those with properties that may be geometry-valued. A feature collection is a collection of features that may itself be regarded as a feature; as a consequence a feature collection has a feature type and thus may have distinct properties of its own, in addition to the features it contains.

Following ISO 19109, the feature types of an application or application domain is usually captured in an application schema. A GML application schema is specified in XML Schema and can be constructed in two different and alternative ways:

— by adhering to the rules specified in ISO 19109 for application schemas in UML, and conforming to both the constraints on such schemas and the rules for mapping them to GML application schemas specified in this International Standard;

— by adhering to the rules for GML application schemas specified in this International Standard for creating a GML application schema directly in XML Schema.

Both ways are supported by this International Standard. To ensure proper use of the conceptual modelling framework of the ISO 19100 series of International Standards, all application schemas are expected to be modelled in accordance with the General Feature Model as specified in ISO 19109. Within the ISO 19100 series, UML is the preferred language by which to model conceptual schemas.

GML specifies XML encodings, conformant with ISO 19118, of several of the conceptual classes defined in the ISO 19100 series of International Standards and the OpenGIS Abstract Specification. These conceptual models include those defined in:

— ISO/TS 19103 — Conceptual schema language (units of measure, basic types);

— ISO 19107 — Spatial schema (geometry and topology objects);

— ISO 19108 — Temporal schema (temporal geometry and topology objects, temporal reference systems);

— ISO 19109 — Rules for application schemas (features);

— ISO 19111 — Spatial referencing by coordinates (coordinate reference systems);

— ISO 19123 — Schema for coverage geometry and functions.

The aim is to provide a standardized encoding (i.e. a standardized implementation in XML) of types specified in the conceptual models specified by the International Standards listed above. If every application schema were encoded independently and the encoding process included the types from, for example, ISO 19108, then, without unambiguous and completely fixed encoding rules, the XML encodings would be different. Also, since every implementation platform has specific strengths and weaknesses, it is helpful to standardize XML encodings for core geographic information concepts modelled in the ISO 19100 series of International Standards and commonly used in application schemas.

In many cases, the mapping from the conceptual classes is straightforward, while in some cases the mapping is more complex (a detailed description of the mapping is part of this International Standard).

In addition, GML provides XML encodings for additional concepts not yet modelled in the ISO 19100 series of International Standards or the OpenGIS Abstract Specification, for example, dynamic features, simple observations or value objects.

Predefined types of geographic feature in GML include coverages and simple observations.

A coverage is a subtype of feature that has a coverage function with a spatiotemporal domain and a value set range of homogeneous 1- to n-dimensional tuples. A coverage may represent one feature or a collection of features “to model and make visible spatial relationships between, and the spatial distribution of, Earth phenomena” (OGC Abstract Specification Topic 6 [20]) and a coverage “acts as a function to return values from its range for any direct position within its spatiotemporal domain” (ISO 19123).

An observation models the act of observing, often with a camera or some other procedure, a person or some form of instrument (Merriam-Webster Dictionary: “an act of recognizing and noting a fact or occurrence often involving measurement with instruments”). An observation is considered to be a GML feature with a time at which the observation took place, and with a value for the observation.

A reference system provides a scale of measurement for assigning values to a position, time or other descriptive quantity or quality.

A coordinate reference system consists of a set of coordinate system axes that is related to the Earth through a datum that defines the size and shape of the Earth.

A temporal reference system provides standard units for measuring time and describing temporal length or duration.

A reference system dictionary provides definitions of reference systems used in spatial or temporal geometries.

Spatial geometries are the values of spatial feature properties. They indicate the coordinate reference system in which their measurements have been made. The “parent” geometry element of a geometric complex or geometric aggregate makes this indication for its constituent geometries.

Temporal geometries are the values of temporal feature properties. Like their spatial counterparts, temporal geometries indicate the temporal reference system in which their measurements have been made.

Spatial or temporal topologies are used to express the different topological relationships between features.

A units of measure dictionary provides definitions of numerical measures of physical quantities, such as length, temperature and pressure, and of conversions between units.

Geographic information — Geography Markup Language (GML)

1 Scope

The Geography Markup Language (GML) is an XML encoding in compliance with ISO 19118 for the transport and storage of geographic information modelled in accordance with the conceptual modelling framework used in the ISO 19100 series of International Standards and including both the spatial and non-spatial properties of geographic features.

This International Standard defines the XML Schema syntax, mechanisms and conventions that:

— provide an open, vendor-neutral framework for the description of geospatial application schemas for the transport and storage of geographic information in XML;

— allow profiles that support proper subsets of GML framework descriptive capabilities;

— support the description of geospatial application schemas for specialized domains and information communities;

— enable the creation and maintenance of linked geographic application schemas and datasets;

— support the storage and transport of application schemas and datasets;

— increase the ability of organizations to share geographic application schemas and the information they describe.

Implementers may decide to store geographic application schemas and information in GML, or they may decide to convert from some other storage format on demand and use GML only for schema and data transport.

NOTE If an ISO 19109 conformant application schema described in UML is used as the basis for the storage and transportation of geographic information, this International Standard provides normative rules for the mapping of such an application schema to a GML application schema in XML Schema and, as such, to an XML encoding for data with a logical structure in accordance with the ISO 19109 conformant application schema.

2 Conformance

2.1 Conformance requirements

Clauses 7 to 19 of this International Standard specify XML Schema components, i.e. the GML schema, which shall be used in GML application schemas in accordance with Clause 21. Clause 20 specifies rules for the specification of a GML profile that may be defined for use in a GML application schema.

Few applications will require the full range of capabilities described by the GML schema. This clause, therefore, defines a set of conformance classes that will support applications whose requirements range from the minimum necessary to define simple feature types to full use of the GML schema.

Most of the schema components specified in this International Standard implement concepts defined in the ISO 19100 series of International Standards. In these cases, the conformance classes defined in this International Standard are based on the conformance classes defined in the corresponding standard.

Any GML application schema, GML profile or software implementation claiming conformance with one of the conformance classes shall pass all test cases of the corresponding abstract test suite.

Any software implementation claiming conformance to this International Standard shall document the GML profile supported by the implementation. The GML profile shall pass all mandatory test cases of the abstract test suite corresponding to GML profiles.

2.2 Conformance classes related to GML application schemas

GML application schemas claiming conformance to this International Standard shall conform to the rules specified in Clauses 7 to 21 and pass all relevant test cases of the abstract test suite in A.1.

Depending on the characteristics of a GML application schema, 12 conformance classes are distinguished. Table 1 lists these classes and the corresponding subclause of the abstract test suite.

Table 1 — Conformance classes related to GML application schemas

Conformance class	Subclause of the abstract test suite
All GML application schemas	A.1.1
GML application schemas converted from an ISO 19109 application schema in UML	A.1.2
GML application schemas to be converted to an ISO 19109 application schema in UML	A.1.3
GML application schemas defining features and feature collections	A.1.4
GML application schemas defining spatial geometries	A.1.5
GML application schemas defining spatial topologies	A.1.6
GML application schemas defining time	A.1.7
GML application schemas defining coordinate reference systems	A.1.8
GML application schemas defining coverages	A.1.9
GML application schemas defining observations	A.1.10
GML application schemas defining dictionaries and definitions	A.1.11
GML application schemas defining values	A.1.12

2.3 Conformance classes related to GML profiles

The requirements of an application schema determine the XML Schema components from the GML schema that shall be included in a GML profile. GML profiles claiming conformance to this International Standard shall satisfy the requirements of the abstract test suite in A.2.

Depending on the contents and requirements concerning a specific GML profile, 31 conformance classes are distinguished. Table 2 lists these classes and the corresponding subclause of the abstract test suite.

Table 2 — Conformance classes related to GML profiles

Conformance class	Subclause of the abstract test suite
All GML profiles	A.2.1
Geometric primitives (spatial) — 0-dimensional	A.2.2.1.1
Geometric primitives (spatial) — 0/1-dimensional	A.2.2.1.2
Geometric primitives (spatial) — 0/1/2-dimensional	A.2.2.1.3
Geometric primitives (spatial) — 0/1/2/3-dimensional	A.2.2.1.4
Geometric complexes (spatial) — 0/1-dimensional	A.2.3.1.1
Geometric complexes (spatial) — 0/1/2-dimensional	A.2.3.1.2
Geometric complexes (spatial) — 0/1/2/3-dimensional	A.2.3.1.3
Topologic complexes (spatial) — 0/1-dimensional	A.2.4.1.1
Topologic complexes (spatial) — 0/1/2-dimensional	A.2.4.1.2
Topologic complexes (spatial) — 0/1/2/3-dimensional	A.2.4.1.3
Topologic complexes with geometric realization (spatial) — 1-dimensional	A.2.5.1.1
Topologic complexes with geometric realization (spatial) — 2-dimensional	A.2.5.1.2
Topologic complexes with geometric realization (spatial) — 3-dimensional	A.2.5.1.3
Coordinate reference systems	A.2.6
Coordinate operations between two coordinate reference systems	A.2.7
Temporal geometry — 0-dimensional	A.2.8.1
Temporal geometry — 0/1-dimensional	A.2.8.2
Temporal topology	A.2.9
Temporal reference systems	A.2.10
Dynamic features	A.2.11
Dictionaries	A.2.12
Units dictionaries	A.2.13
Observations	A.2.14
Abstract coverage	A.2.15.1
Discrete point coverage	A.2.15.2
Discrete curve coverage	A.2.15.3
Discrete surface coverage	A.2.15.4
Discrete solid coverage	A.2.15.5
Grid coverage	A.2.15.6
Continuous coverage	A.2.15.7

Curve implementations, for those GML profiles including 1-dimensional spatial geometry objects, shall always include a “linear” interpolation technique. Surface implementations, for those GML profiles including 2-dimensional spatial geometry objects, shall always include a “planar” interpolation technique. Additional curve and surface interpolation mechanisms are optional but, if implemented, they shall follow the definition included in this International Standard.

NOTE 1 Compare these conformance classes with ISO 19107:2003, Clause 2, ISO 19108:2002, 2.2, and ISO 19123:2005, Clause 2.

NOTE 2 A GML profile conforming to the three conformance classes “Geometric primitives (spatial) — 0-dimensional”, “Geometric primitives (spatial) — 0/1-dimensional”, and “Geometric primitives (spatial) — 0/1/2-dimensional” (in addition to conformance class “All GML profiles”) conforms to the spatial profile defined in ISO 19137:2007 and the respective conformance tests in ISO 19137:2007, B.1, B.2 and B.3.

2.4 Conformance classes related to GML documents

GML documents claiming conformance to this International Standard shall conform to the rules specified in Clauses 7 to 21 and pass all relevant test cases of the abstract test suite in A.3.

2.5 Conformance classes related to software implementations

Software implementations reading or writing GML or GML application schemas claiming conformance to this International Standard shall pass all of the corresponding abstract test suites described in the abstract test suite in Annex B.

Depending on the capabilities of the implementation, 11 conformance classes are distinguished. Table 3 lists these classes and the corresponding subclause of the abstract test suite.

Table 3 — Conformance classes related to implementations

Conformance class	Subclause of the abstract test suite
All software implementations	B.1
Support for remote simple Xlinks	B.2.1
Support for extended Xlinks	B.2.2
Support for nillable properties	B.2.3
Support for units of measurement	B.2.4
Support for ownership semantics of properties	B.2.5
Metadata properties	B.2.6
Support for GML profiles in instance validation	B.2.7
Writing GML	B.3
Reading GML	B.4
Writing GML application schemas	B.5
Reading GML application schemas	B.6

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO/IEC 11404:1996, *Information technology — Programming languages, their environments and system software interfaces — Language-independent datatypes*

ISO/TS 19103:2005, *Geographic information — Conceptual schema language*

ISO 19107:2003, *Geographic information — Spatial schema*

ISO 19108:2002, *Geographic information — Temporal schema*

ISO 19109:2005, *Geographic information — Rules for application schema*

ISO 19111:2007, *Geographic information — Spatial referencing by coordinates*

ISO 19115:2003, *Geographic information — Metadata*

ISO 19118:2005, *Geographic information — Encoding*

ISO 19123:2005, *Geographic information — Schema for coverage geometry and functions*

ISO/TS 19139, *Geographic information — Metadata — XML schema implementation*

ISO/IEC 19757-3, *Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron*

ISO 80000-3, *Quantities and units — Part 3: Space and time*

IETF RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax* (August 1998)

W3C XLink, *XML Linking Language (XLink) Version 1.0*, W3C Recommendation (27 June 2001)

W3C XML, *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation (4 February 2004)

W3C XML Namespaces, *Namespaces in XML*, W3C Recommendation (14 January 1999)

W3C XML Schema Part 1, *XML Schema Part 1: Structures*, W3C Recommendation (2 May 2001)

W3C XML Schema Part 2, *XML Schema Part 2: Datatypes*, W3C Recommendation (2 May 2001)

4 Terms and symbols

4.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1.1
application schema
conceptual **schema** for data required by one or more applications

[ISO 19101:2002]

4.1.2
association
semantic relationship between two or more classifiers that specifies connections among their instances

[ISO/IEC 19501:2005:]

4.1.3
attribute <XML>
name-value pair contained in an **element**

NOTE In this document an attribute is an XML attribute unless otherwise specified. The syntax of an XML attribute is "Attribute::= Name = AttValue". An attribute typically acts as an XML element modifier (e.g. <Road gml:id = "r1" />; here gml:id is an attribute).

4.1.4
boundary
set that represents the limit of an entity

[ISO 19107:2003]

4.1.5
child element <XML>
immediate descendant **element** of an **element**

4.1.6
closure
union of the **interior** and **boundary** of a **topological** or **geometric object**

[ISO 19107:2003]

4.1.7
codelist
value domain including a code for each permissible value

4.1.8
codespace
rule or authority for a code, name, term or category

EXAMPLE Examples of codespaces include dictionaries, authorities, codelists, etc.

4.1.9
composite curve
sequence of **curves** such that each **curve** (except the first) starts at the end point of the previous **curve** in the **sequence**

[ISO 19107:2003]

NOTE A composite curve, as a set of direct positions, has all the properties of a curve.

4.1.10
composite solid
connected **set** of **solids** adjoining one another along shared **boundary surfaces**

[ISO 19107:2003]

NOTE A composite solid, as a set of direct positions, has all the properties of a solid.

4.1.11
composite surface
connected **set** of **surfaces** adjoining one another along shared **boundary curves**

[ISO 19107:2003]

NOTE A composite surface, as a set of direct positions, has all the properties of a surface.

4.1.12
coordinate
one of a **sequence** of n numbers designating the position of a **point** in n-dimensional space

[ISO 19111:2007]

NOTE In a coordinate reference system, the n numbers shall be qualified by units.

4.1.13
coordinate reference system
coordinate system that is related to an object by a **datum**

[ISO 19111:2007]

4.1.14
coordinate system
set of mathematical rules for specifying how **coordinates** are to be assigned to **points**

[ISO 19111:2007]

4.1.15
coordinate tuple
tuple composed of a sequence of **coordinates**

[ISO 19111:2007]

4.1.16
coverage
feature that acts as a function to return values from its range for any **direct position** within its spatial, temporal or spatiotemporal **domain**

[ISO 19123:2005]

4.1.17
curve
1-dimensional **geometric primitive**, representing the continuous image of a line

[ISO 19107:2003]

NOTE The boundary of a curve is the set of points at either end of the curve. If the curve is a cycle, the two ends are identical, and the curve (if topologically closed) is considered to not have a boundary. The first point is called the start point, and the last is the end point. Connectivity of the curve is guaranteed by the "continuous image of a line" clause. A topological theorem states that a continuous image of a connected set is connected.

4.1.18
data type
specification of a value **domain** with operations allowed on values in this **domain**

[ISO/TS 19103:2005]

EXAMPLE Integer, Real, Boolean, String, Date (conversion of data into a series of codes).

NOTE Data types include primitive predefined types and user-definable types. All instances of a data type lack identity.

4.1.19
datum
parameter or set of parameters that define the position of the origin, the scale, and the orientation of a coordinate system

[ISO 19111:2007]

NOTE A datum may be a geodetic datum, a vertical datum, an engineering datum, an image datum or a temporal datum.

4.1.20
direct position
position described by a single **set** of **coordinates** within a **coordinate reference system**

[ISO 19107:2003]

4.1.21
domain
well-defined **set**

[ISO/TS 19103:2005]

NOTE 1 A mathematical **function** may be defined on this set, i.e. in a function f:A→B, A is the domain of the function f.

NOTE 2 A domain as in domain of discourse refers to a subject or area of interest.

4.1.22
edge
1-dimensional topological primitive

[ISO 19107:2003]

4.1.23
element <XML>
basic information item of an XML document containing **child elements**, **attributes** and character data

NOTE From the XML Information Set: “Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag. Each element has a type, identified by name, sometimes called its ‘generic identifier’ (GI), and may have a set of attribute specifications. Each attribute specification has a name and a value.”

4.1.24
exterior
difference between the universe and the **closure**

[ISO 19107:2003]

4.1.25
face
2-dimensional topological primitive

[ISO 19107:2003]

NOTE The geometric realization of a face is a surface. The boundary of a face is the set of directed edges within the same topology complex that are associated to the face via the boundary relations. These may be organized as rings.

4.1.26
feature
abstraction of real world phenomena

[ISO 19101:2002]

NOTE A feature may occur as a type or an instance. The term “feature type” or “feature instance” should be used when only one is meant.

4.1.27
feature association
relationship that links instances of one **feature** type with instances of the same or a different **feature** type

[ISO 19110:2005]

4.1.28
function
rule that associates each element from a **domain** (source, or domain of the function) to a unique element in another **domain** (target, co-domain, or **range**)

[ISO 19107:2003]

4.1.29
geodetic datum
datum describing the relationship of a 2- or 3-dimensional **coordinate system** to the Earth

[ISO 19111:2007]

4.1.30
geometric object
spatial object representing a geometric set

[ISO 19107:2003]

4.1.31
geometric primitive
geometric object representing a single, connected, homogeneous element of space

[ISO 19107:2003]

4.1.32
geometric set
set of **direct positions**

[ISO 19107:2003]

4.1.33
geometry property <GML>
property of a GML **feature** that describes some aspect of the geometry of the feature

NOTE The geometry property name is the role of the geometry in relation to the feature.

4.1.34
GML application schema
application schema written in XML Schema in accordance with the rules specified in this International Standard

4.1.35
GML document
XML document with a root element that is one of the elements AbstractFeature, Dictionary or TopoComplex specified in the **GML schema** or any element of a substitution group of any of these elements

4.1.36
GML profile
subset of the **GML schema**

4.1.37
GML schema
schema components in the XML namespace “http://www.opengis.net/gml/3.2” as specified in this International Standard

4.1.38
grid
network composed of two or more **sets** of **curves** in which the members of each **set** intersect the members of the other **sets** in an algorithmic way

[ISO 19123:2005]

NOTE The curves partition a space into grid cells.

4.1.39
interior
set of all **direct positions** that are on a **geometric object** but which are not on its **boundary**

[ISO 19107:2003]

4.1.40
line string
curve composed of straight-line segments

4.1.41
measure <GML>
value described using a numeric amount with a scale or using a scalar reference system

NOTE When used as a noun, measure is a synonym for physical quantity.

4.1.42
measurand
particular quantity subject to measurement

4.1.43
namespace <XML>
collection of names, identified by a **URI** reference, which are used in XML documents as element names and attribute names (W3C XML Namespaces)

4.1.44
node
0-dimensional topological primitive

[ISO 19107:2003]

4.1.45
object
entity with a well defined **boundary** and identity that encapsulates state and behaviour

[ISO 19107:2003]

NOTE A GML object is an XML element of a type derived from AbstractGMLType.

4.1.46
observable type
data type to indicate the **physical quantity** as a result of an observation

4.1.47
point
0-dimensional **geometric primitive**, representing a position

[ISO 19107:2003]

NOTE The boundary of a point is the empty set.

4.1.48
polygon
planar **surface** defined by 1 **exterior boundary** and 0 or more **interior boundaries**

4.1.49
property <GML>
a **child element** of a GML object

NOTE It corresponds to feature attribute and feature association role in ISO 19109. If a GML property of a feature has an xlink:href attribute that references a feature, the property represents a feature association role.

4.1.50
physical quantity
quantity used for the quantitative description of physical phenomena

[ISO 31-0:1992]

NOTE In GML a physical quantity is always a value described using a numeric amount with a scale or using a scalar reference system. Physical quantity is a synonym for measure when the latter is used as a noun.

4.1.51
range
set of all values a function f can take as its arguments vary over its **domain**

4.1.52
rectified grid
grid for which there is an affine transformation between the grid coordinates and the **coordinates** of an external **coordinate reference system**

[ISO 19123:2005]

4.1.53
schema
formal description of a model

[ISO 19101:2002]

NOTE In general, a schema is an abstract representation of an object's characteristics and relationship to other objects. An XML schema represents the relationship between the attributes and elements of an XML object (for example, a document or a portion of a document)

4.1.54
schema <XML Schema>
collection of **schema** components within the same target **namespace**

EXAMPLE Schema components of W3C XML Schema are types, elements, attributes, groups, etc.

4.1.55
schema document <XML Schema>
XML document containing **schema** component definitions and declarations

NOTE The W3C XML Schema provides an XML interchange format for schema information. A single schema document provides descriptions of components associated with a single XML namespace, but several documents may describe components in the same schema, i.e. the same target namespace.

4.1.56
semantic type
category of objects that share some common characteristics and are thus given an identifying type name in a particular domain of discourse

4.1.57
sequence
finite, ordered collection of related items (objects or values) that may be repeated

[ISO 19107:2003]

4.1.58
set
unordered collection of related items (**objects** or values) with no repetition

[ISO 19107:2003]

4.1.59
spatial object
object used for representing a spatial characteristic of a **feature**

[ISO 19107:2003]

4.1.60
surface
2-dimensional geometric primitive, locally representing a continuous image of a region of a plane

NOTE The boundary of a surface is the set of oriented, closed curves that delineate the limits of the surface. Surfaces that are isomorphic to a sphere, or to an n-torus (a topological sphere with n "handles") have no boundary. Such surfaces are called cycles.

[ISO 19107:2003]

4.1.61
tag <XML>
markup in an XML document delimiting the content of an **element**

EXAMPLE <Road>

NOTE A tag with no forward slash (e.g. <Road>) is called a start-tag (also opening tag), and one with a forward slash (e.g. </Road> is called an end-tag (also closing tag).

4.1.62
topological object
spatial object representing spatial characteristics that are invariant under continuous transformations

[ISO 19107:2003]

4.1.63
tuple
ordered list of values

NOTE The number of values in a tuple is immutable.

4.1.64
UML application schema
application schema written in UML in accordance with ISO 19109

4.1.65
Uniform Resource Identifier (URI)
unique identifier for a resource, structured in conformance with IETF RFC 2396

NOTE The general syntax is <scheme>::<scheme-specific-part>. The hierarchical syntax with a namespace is <scheme>://<authority><path>?<query> — see RFC 2396.

4.2 Symbols and abbreviated terms

The following symbols and abbreviated terms are used in this document:

CRS	Coordinate Reference System
CS	Coordinate System
CSV	Comma Separated Values
CT	Coordinate Transformation
DTD	Document Type Definition
EPSG	European Petroleum Survey Group
GIS	Geographic Information System
GML	Geography Markup Language
	NOTE The acronym GML was previously used in ISO also as Generalized Markup Language (which led to SGML, Standard Generalized Markup Language, ISO 8879).
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
OGC	Open Geospatial Consortium

RDF	Resource Description Framework
RFC	Request for Comments
SMIL	Synchronized Multimedia Integration Language
SOAP	Simple Object Access Protocol
SVG	Scalable Vector Graphics
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WFS	Web Feature Service
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language — Transformations
0D	Zero Dimensional
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional

5 Conventions

5.1 XML namespaces

All components of the GML schema are defined in the namespace with the identifier "http://www.opengis.net/gml/3.2", for which the prefix **gml** or the default namespace is used within this International Standard.

All components described by the W3C Xlink Recommendation are defined in the namespace with the identifier "http://www.w3.org/1999/xlink", for which the prefix **xlink** is used within this International Standard.

NOTE The schema components in both namespaces are documented in XML Schema documents in Annex C.

5.2 Versioning

Each schema document specifying components of the GML schema shall carry a version attribute as defined in the XML Schema Recommendation. The format of the version attribute string is x.y.z where x denotes the major version number, y denotes a minor version number and z denotes a bug fix release for that document. The version described by this International Standard is 3.2.1.

NOTE This is the first version of GML as an International Standard. Previous versions of GML have been developed and published by the Open Geospatial Consortium (OGC).

5.3 Deprecated parts of previous versions of GML

The verb "**deprecate**" provides notice that the referenced portion of this International Standard is being retained for backwards compatibility with earlier versions but may be removed from a future version without further notice.

Sections of this International Standard that describe or refer to deprecated GML components are written in *italics*.

NOTE 1 This International Standard is published by the Open Geospatial Consortium as GML 3.2.1, the previous adopted version of GML in the Open Geospatial Consortium was 3.1.1.

NOTE 2 All schema components that were part of version 2.1 of GML but were deprecated in version 3.0 of GML have been removed and are not supported by this International Standard.

5.4 UML notation

Many diagrams that appear in this International Standard are presented using the Unified Modeling Language (UML) static structure diagram. The UML notations used in this International Standard are described in Figure 1.

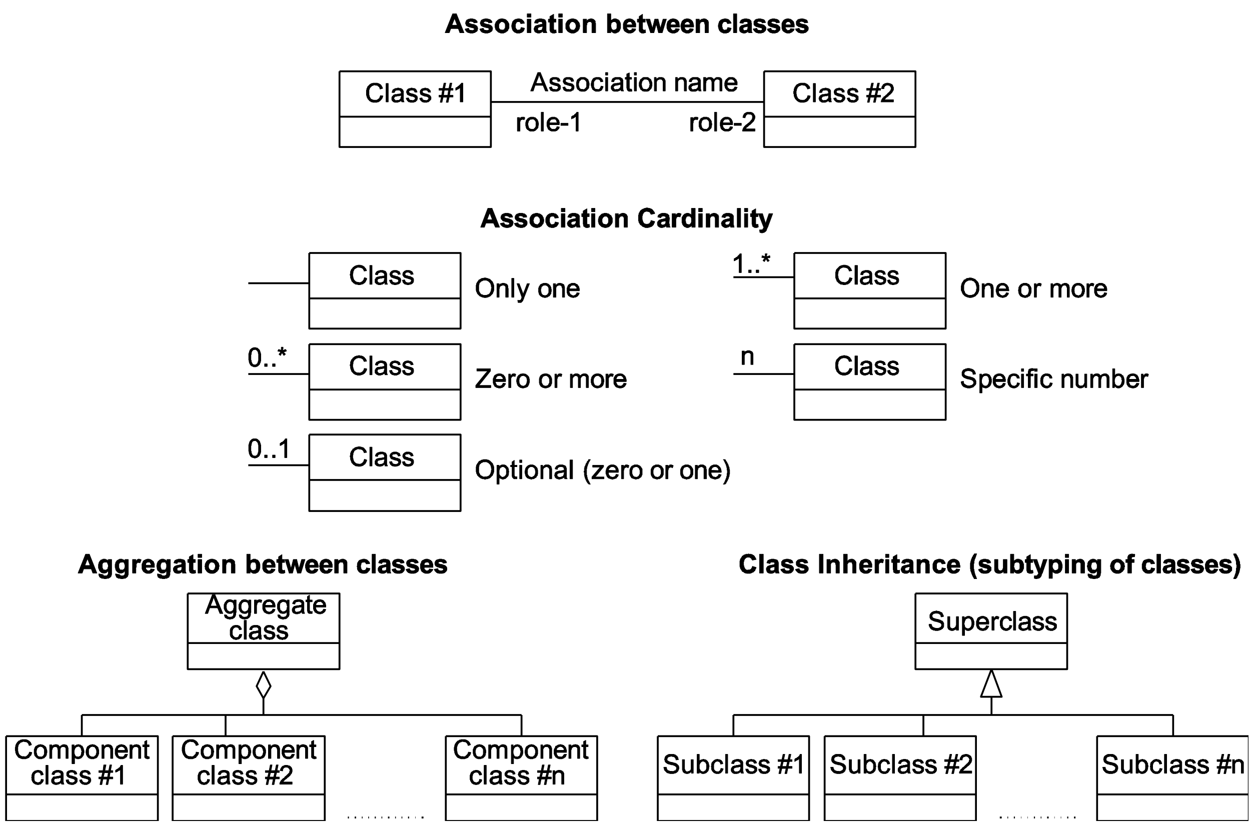


Figure 1 — UML notation

In this International Standard, the following stereotypes of UML class are used:

— <<DataType>> is a set of properties that lack identity (independent existence and the possibility of side effects). A DataType is a class with no operations whose primary purpose is to hold the information.

- <<Union>> is a set of properties. The semantics is that only one of the properties may be present at any time.

- <<FeatureType>> is a feature as defined in ISO 19109.

- <<CodeList>> is a flexible enumeration that uses string values for expressing a list of potential values.

- <<Enumeration>> is a fixed list of valid identifiers of named literal values. Attributes of an enumerated type may only take values from this list.

- <<Abstract>> is an abstract object type (the stereotype is used in addition to formatting the class name in italics).

- <<Type>> is a set of abstract attributes and associations. Abstract means that their specification does not imply that they have to be concretely implemented as instance variables.

In this International Standard, the following standard data types are used:

- CharacterString — A sequence of characters (in general this data type is mapped to “string” in XML Schema).

- Integer — An integer number (in general this data type is mapped to “integer” in XML Schema).

- Real — A floating point number (in general this data type is mapped to “double” in XML Schema).

- Boolean — A value specifying TRUE or FALSE (in general this data type is mapped to “boolean” in XML Schema).

5.5 XML Schema

The normative parts of this International Standard use the W3C XML Schema language to describe the grammar of conformant GML data instances. XML Schema is a rich language with many capabilities and subtleties. While a reader who is unfamiliar with XML Schema may be able to follow the description in a general fashion, this International Standard is not intended to serve as an introduction to XML Schema. In order to have a full understanding of this International Standard it is necessary for the reader to have a reasonable knowledge of XML Schema.

6 Overview of the GML schema

6.1 GML schema

GML specifies XML encodings of a number of the conceptual classes defined in the ISO 19100 series of International Standards and the OpenGIS Abstract Specification in conformance with these standards and specifications.

The relevant conceptual models include those defined in:

- ISO/TS 19103 — Conceptual schema language (units of measure, basic types);

- ISO 19107 — Spatial schema (spatial geometry and topology);

- ISO 19108 — Temporal schema (temporal geometry and topology, temporal reference systems);

- ISO 19109 — Rules for application schemas (features);

— ISO 19111 — Spatial referencing by coordinates (coordinate reference systems);

— ISO 19123 — Schema for coverage geometry and functions (coverages, grids).

In many cases, the mapping from the conceptual classes to XML is straightforward, while in some cases the mapping is more complex. For both cases, the mapping is documented in detail in Annex D.

In addition, GML provides XML encodings for additional concepts not yet modelled in the ISO 19100 series of International Standards or the OpenGIS Abstract Specification. Examples include moving objects, simple observations or value objects. Additional conceptual classes corresponding to these extensions are also specified in Annex D.

The GML schema comprises the components (XML elements, attributes, simple types, complex types, attribute groups, groups, etc.) that are described in this International Standard. The XML encoding conforms to ISO 19118.

6.2 GML application schemas

Designers of GML application schemas may extend or restrict the types defined in the GML schema to define appropriate types for an application domain. Non-abstract elements, attributes and types from the GML schema may be used directly in an application schema, if no changes are required.

Following ISO 19109, the feature types of an application or application domain are specified in an application schema. A GML application schema shall be specified in XML Schema and import the GML schema. It may be constructed in one of two different ways:

— By adhering to the rules for GML application schemas specified in Clause 21 for creating a GML application schema directly in XML Schema.

— By adhering to the rules specified in ISO 19109 for application schemas in UML, and conforming to both the constraints on such schemas and the rules for mapping them to GML application schemas specified in Annex E of this International Standard. The mapping from an ISO 19109 conformant Application Schema in UML to the corresponding GML application schema is based on a set of encoding rules. These encoding rules conform with the rules for GML application schemas and ISO 19118.

Both ways are valid approaches to construct GML application schemas. All application schemas shall be modelled in accordance with the General Feature Model specified in ISO 19109. Within the ISO 19100 series, UML is the preferred language to describe conceptual schemas.

The second approach is recommended in general to ensure proper use of the conceptual modelling framework of the ISO 19100 series of International Standards. However, the following reasons are examples where it may be justified to apply the first approach:

— Additional capabilities of the GML schema may be required in addition to the capabilities that are accessible by using the encoding rules specified in Annex E.

— Only an XML representation may be required and the application schema may be relatively simple, so the use of a conceptual schema language may be considered an unjustified overhead.

— The application may need a more optimized or compact XML encoding than the one that is the result of the encoding rules specified in Annex E.

NOTE Annex F provides rules for mapping a GML application schema to an ISO 19109 conformant Application Schema in UML.

In both cases, GML application schemas conformant with this International Standard shall use all of the applicable GML schema components, either directly or by specialization, and are valid in accordance with the rules for XML Schema. How the GML application schemas were produced is not relevant for conformance to the requirements of this International Standard.

6.3 Relationship between the ISO 19100 series of International Standards, the GML schema and GML application schemas

The approach taken by this International Standard is shown in Figure 2. The two main aspects are:

— Clear documentation of the conceptual model of GML: The profile of the ISO 19100 series of International Standards that is implemented by GML is documented as well as the extensions to this profile.

— Support for application schema development either in UML or XML Schema: In order to achieve this two-way mapping between UML (i.e. ISO 19109 conformant application schemas in UML) and XML Schema (i.e. GML application schemas in XML Schema) the constructs used in both representations have been limited. While this reduces the expressiveness of the schema descriptions to some extent, this also reduces their complexity and may make them easier to implement.

NOTE While the mapping from UML to XML Schema is discussed in ISO 19118, Annex A, the reverse mapping is not discussed in any other standard in the ISO 19100 series of International Standards.

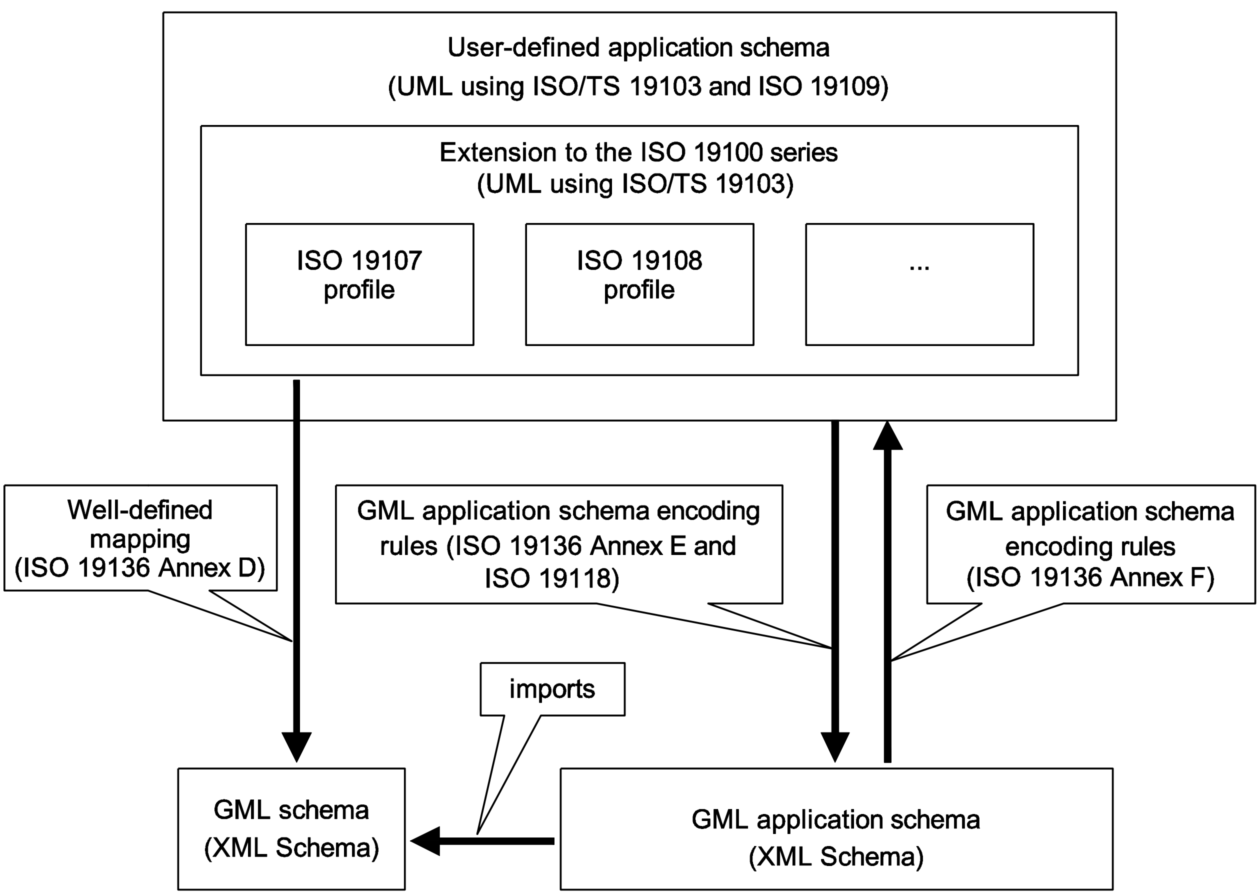


Figure 2 — Relationship between the ISO 19100 series of International Standards and ISO 19136/GML

6.4 Organization of this International Standard

GML defines the various entities such as features, geometries, topologies etc. through a hierarchy of GML object types. The mapping between GML object types and classes in the conceptual model of the ISO 19100 series of International Standards and the OGC Abstract Specification is shown in Table D.2. The normative GML schema is organized around these object types.

Subclause 7.2 describes basic schema components of GML. It defines the root object, `gml:AbstractObject`, and the root of the GML class hierarchy, `gml:AbstractGML`.

Subclause 8.1 describes the Xlink schema. This schema is an OGC implementation of the XLink specification using XML Schema. It may be replaced in some future release by an equivalent schema from the W3C.

NOTE 1 Within this International Standard an XML Schema description is provided for xlink components. This is provided for convenience in the context of an XML Schema-based environment. The normative definitions are given a non-XML Schema form in the XLink Recommendation.

Subclause 8.2 defines the GML representation of some basic data types that are used in the GML schema. Most of these types are simple types or simple content types.

Clause 9 describes the feature schema components which defines `gml:AbstractFeature` and some derived components.

Clause 10, 10.5.10 and Clause11 describe the geometry schema components that define `gml:AbstractGeometry`, `gml:AbstractGeometricPrimitive`, `gml:AbstractGeometricAggregate`, `gml:GeometricComplex` and some derived components.

Clause 12 describes the coordinate reference system schema components that define the subtypes of `gml:IdentifiedObject`, `gml:AbstractCRS`, `gml:AbstractCoordinateReferenceSystem`, and the elements and types required to construct specific coordinate reference systems.

Clause 13 describes the schema components for topology which define `gml:AbstractTopology`, `gml:AbstractTopoPrimitive`, `gml:TopoComplex` and some derived components.

Clause 14 describes the schema components for temporal constructs defining `gml:AbstractTimeObject`, `gml:AbstractTimePrimitive`, `gml:AbstractTimeGeometricPrimitive`, `gml:AbstractTimeTopologyPrimitive`, `gml:AbstractTimeComplex` and derived components as well as `gml:DynamicFeature` and derived components.

Clause 15 describes the schema components for definitions and dictionaries including `gml:Definition` and `gml:Dictionary`.

Clause 16 describes the schema components for the construction of units of measure (`gml:UnitDefinition` and derived components), measures and value objects (`gml:AbstractValue`, `gml:AbstractScalarValue`, `gml:AbstractScalarValueList` and derived components).

Clause 17 describes the schema components for the description of direction.

Clause 18 describes the schema components for simple observations (`gml:Observation` and derived components).

Clause 19 describes the schema components for grids and coverages. This describes `gml:Grid`, `gml:AbstractCoverage`, `gml:AbstractDiscreteCoverage`, `gml:AbstractContinuousCoverage` and derived components.

These clauses describe the normative GML schema and explain their contents, structure and dependencies.

The representation of the GML schema presented in this International Standard uses the XML interchange format provided by W3C XML Schema. The descriptions of the set of components are factored into schema documents, where each document gathers together components that correspond broadly to the classification shown in Figure 2. However, while the XML representation of each GML schema component in this International Standard is normative, the packaging into schema documents is not. Clause 20 (profiles) and Annex G (subsetting) describe principles and methods for alternative packaging of the XML representation of GML schema components.

All components defined or declared in this International Standard use the same target namespace of http://www.opengis.net/gml/3.2.

NOTE 2 XML namespaces provide a mechanism for avoiding ambiguity arising from name clashes within XML documents. All components described in a single schema document are in a single target namespace, but more than one schema document may describe components in a namespace. Within the XML development community there are precedents for assigning either one or several namespaces to a set of schema components for a single application. The use of a single namespace for GML schema components is consistent with the non-normative factoring of the XML representation of GML components between schema documents.

UML uses packages to collect related components. Furthermore, within the ISO 19100 series of International Standards, prefixes following the pattern "AA_" are used to distinguish classes from different packages in a way that resembles XML namespaces. However, for the reasons given above, packaging of GML components is non-normative and all GML components are in a single namespace, so no correspondence between the two letter prefixes in the ISO 19100 series of International Standards and XML namespaces in GML is possible.

6.5 Deprecated and experimental schema components

Experimental, informative schema components dealing with rules for a default styling of GML objects are described in Annex H.

Deprecated global schema components (elements, attributes, types) are included in Annex I.

7 GML schema — General rules and base schema components

7.1 GML model and syntax

7.1.1 GML instance documents

GML uses an explicit syntax to instantiate a GML application schema conformant with the General Feature Model defined in ISO 19109 in an XML document.

A feature is encoded as an XML element with the name of the feature type. Other identifiable objects are encoded as XML elements with the name of the object type.

Each feature attribute and feature association role is a property of a feature. Feature properties are encoded in an XML element.

NOTE 1 The term "attribute" in XML refers to a specific syntactic component in XML documents, so to avoid confusion when describing the XML encoding, GML follows RDF (W3C, 1999) terminology and uses the term property rather than attribute or association role. The General Feature Model (ISO 19109) also uses the term "property" as a generalization for "attribute", "association role" or "operation".

Furthermore, the property semantics, which is indicated by the name of the element representing the property, is distinguished from the property value, which is given by the content of the property element. A property element may contain its value as content encoded inline, or reference its value with a simple XLink. The value of a property may be simple, or it may be a feature or other complex object. When recorded inline, the value of a simple property is recorded as a literal value with no embedded markup (text), while if the value is complex it appears as a subtree using XML markup (i.e. an XML element with sub-structure).

NOTE 2 The GML model has a straightforward representation using the UML profile used in the ISO 19100 series of International Standards (defined in ISO/TS 19103). This is described in detail in Annex D and Annex E, but can be summarized approximately and briefly as follows.

Features are represented

— in UML by objects, where the name of the feature type is used as the name of the object class;

— in GML instances by XML elements, where the name of the feature type is used as the name of the element.

Feature properties are represented

— in UML by association roles with feature type classes, and attributes of feature type classes, where the property semantics are given by the association role name or attribute name;

— in GML instances by sub-elements (known as property elements) of feature elements, where the property semantics are given by the property element name.

The property value has a type indicated

— in UML by the class of the association target, or by the data type of the attribute;

— in GML, in the case of properties with complex values, by the name of the object element contained within the property element and in case of a property with simple value by the type of the literal value containing no embedded XML markup.

The result is a layered XML document, in which XML elements corresponding to features, objects or values occur interleaved with XML elements corresponding to the properties that relate them. The function of a feature, object or value in context can always be determined by inspecting the name of the property element which directly contains it, or which carries the reference to it.

NOTE 3 This encoding pattern is sometimes referred to as the “object-property model” and has been the basis of the GML encoding model since the first version was adopted by OGC. While in some cases this encoding pattern adds extra levels of elements in instance documents it also provides significant benefits: It helps to make a GML instance document understandable on its own, provides a predictable structure and avoids too heavy reliance on XML Schema as it is expected that GML instance documents may outlive the common use of W3C XML Schema language.

7.1.2 Lexical conventions

There are several lexical conventions used in the GML schema for the names of elements and complex types to assist in human comprehension of GML instances and schemas:

— objects are instantiated as XML elements with a conceptually meaningful name in UpperCamelCase;

— properties are instantiated as XML elements whose name is in lowerCamelCase;

— abstract elements have a prefix “Abstract” (objects) or “abstract” (properties) prepended to their name;

— the names of XML Schema complex types are in UpperCamelCase ending in the word “Type”;

— abstract XML Schema complex types have the word “Abstract” prepended.

It is strongly recommended to follow these conventions also in GML application schemas. The rules are only applicable in languages that distinguish between upper and lower case.

NOTE UpperCamelCase is a naming convention in which a name is formed of multiple words that are joined together as a single word with the first letter of each of the multiple words capitalized within the new word that forms the name. lowerCamelCase is a variation in which the first letter of the new word is lower case, allowing it to be easily distinguished from an UpperCamelCase name.

7.1.3 XML Schema definition of GML language

The GML schema consists of W3C XML Schema components that define types and declare

— XML elements to encode GML objects with identity,

— XML elements to encode GML properties of those objects, and

— XML attributes qualifying those properties.

A GML object is an XML element of a type derived directly or indirectly from `gml:AbstractGMLType`. From this derivation, a GML object may have a `gml:id` attribute.

A GML property shall not be derived from `gml:AbstractGMLType`, shall not have a `gml:id` attribute, or any other attribute of XML type ID.

An element is a GML property if and only if it is a child element of a GML object.

A GML object shall not appear as the immediate child of a GML object.

Consequently, no element may be both a GML object and a GML property.

All XML attributes declared in the GML schema are defined without namespace, the only exception is the `gml:id` XML attribute.

NOTE The use of additional XML attributes in a GML application schema is discouraged.

7.2 gmlBase schema components

7.2.1 Goals of base schema components

The gmlBase schema components establish the GML model and syntax, in particular

— a root XML type from which XML types for all GML objects should be derived,

— a pattern and components for GML properties,

— patterns for collections and arrays, and components for generic collections and arrays,

— components for associating metadata with GML objects,

— components for constructing definitions and dictionaries.

NOTE The corresponding schema document in Annex C is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:gmlBase:3.2.1

7.2.2 Base objects

7.2.2.1 AbstractObject

An abstract convenience element `gml:AbstractObject` is declared as follows:

```
<element name="AbstractObject" abstract="true"/>
```

This element has no type defined, and is therefore implicitly (in accordance with the rules of W3C XML Schema) an XML Schema `anyType`. It is used as the head of an XML Schema substitution group which unifies complex content and certain simple content elements used for datatypes in GML, including the `gml:AbstractGML` substitution group.

NOTE `gml:AbstractObject` is defined primarily to act as a variable in certain aggregate patterns where it is necessary to allow either elements in the `gml:AbstractGML` substitution group, or certain complex content or simple content elements to be valid in an instance.

A GML dataset (also called a data instance or data document) is represented by an object element. This object may in turn be a collection of GML objects.

7.2.2.2 AbstractGML, AbstractGMLType

The most basic components for representations of identifiable objects are described in the schema as follows:

```
<element name="AbstractGML" type="gml:AbstractGMLType" abstract="true" substitutionGroup="gml:AbstractObject"/>

<complexType name="AbstractGMLType" abstract="true">
	<sequence>
		<group ref="gml:StandardObjectProperties"/>
	</sequence>
	<attribute ref="gml:id"/>
</complexType>

<group name="StandardObjectProperties">
	<sequence>
		<element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
		<element ref="gml:description" minOccurs="0"/>
		<element ref="gml:descriptionReference" minOccurs="0"/>
		<element ref="gml:identifier" minOccurs="0"/>
		<element ref="gml:name" minOccurs="0" maxOccurs="unbounded"/>
	</sequence>
</group>
```

The abstract element `gml:AbstractGML` is “any GML object having identity”. It acts as the head of an XML Schema substitution group, which may include any element which is a GML feature, or other object, with identity. This is used as a variable in content models in GML core and application schemas. It is effectively an abstract superclass for all GML objects.

The pairing of `gml:AbstractGML` and `gml:AbstractGMLType` shows a basic pattern used in the GML schema, whereby each GML object type is represented by a global element declaration, which has an associated XML Schema type definition. The name of an element representing a GML object indicates the conceptual meaning of the object. Generic element names in GML include `gml:AbstractObject`, `gml:AbstractGML`, `gml:AbstractFeature`, `gml:AbstractValue`, `gml:AbstractCoverage`, `gml:AbstractTopology` and `gml:AbstractCRS`. These other generic elements representing objects are defined elsewhere in this International Standard.

The child XML elements and XML attributes of a GML object are properties of that object. Thus an object represented by an `gml:AbstractGML` element has five non-deprecated properties: `gml:identifier`, `gml:description`, `gml:descriptionReference`, `gml:name` and `gml:id`. These are described in 7.2.4.

NOTE The group `gml:StandardObjectProperties` is provided for convenience in the construction of application schema, particularly when it is desired to define types derived by restriction from `gml:AbstractGMLType` and `gml:AbstractFeatureType`. Derivation by restriction requires that all components that are used unchanged are copied down into the new type definition. As an alternative to including element declarations for all the standard object properties, a one line reference to `gml:StandardObjectProperties` may be used instead:

```
<group ref="gml:StandardObjectProperties"/>
```

7.2.3 GML properties

7.2.3.1 Introduction

The term "property" is used to refer to a GML property, which is any characteristic of a GML object. An element in a GML document or data stream is a GML property if and only if it is a child element of a GML object element. The meaning of each property shall be indicated by the name of the element that instantiates it.

GML objects may have an unlimited number of properties, in addition to those inherited from `gml:AbstractGMLType`. A property may be defined to have either simple or complex content. A property with simple content has an XML Schema simple content type, as illustrated by the case of the standard property elements `gml:description` and `gml:name`. A property with complex content has an XML Schema complex content type.

Property elements may use two modes:

— inline: the value of the property is represented directly, as the content of the property element. This method is used by the standard property `gml:name` and may be used for `gml:description` (see 7.2.4.2).

— by reference: the value of the property is available elsewhere, and is identified by the value of an `xlink:href` attribute on the property element. This alternative method shall be used for the standard property `gml:descriptionReference` (see 7.2.4.3).

EXAMPLE See 8.1 for examples on the use of xlink references.

NOTE The roles of feature associations as defined in ISO 19109 (General Feature Model) and OpenGIS Abstract Specification Topic 8 can be represented in several ways in a GML application schema:

— By implementing only one role of the association as navigable, i.e. representing it in the XML encoding. This is the usual representation in the GML schema itself with some exceptions, for example, the boundary and co-boundary association roles between the topology objects.

— By specifying individual properties in the feature types participating in the association. However in this case, the consistency constraints implied by the association cannot be enforced by XML Schema validation. This encoding style is, for example, used for the boundary and co-boundary association roles between the topology objects and in Annex E. See also 7.2.3.9.

— By creating an association object as a GML object. This also allows n-ary associations and associations with properties to be modelled.

— By using extended Xlinks. This encoding is similar to the "association object" representation.

7.2.3.2 AssociationAttributeGroup

XLink components are the standard method to support hypertext referencing in XML. An XML Schema attribute group, `gml:AssociationAttributeGroup`, is provided to support the use of Xlinks as the method for indicating the value of a property by reference in a uniform manner in GML. This attribute group is defined as follows:

```
<attributeGroup name="AssociationAttributeGroup">
    <attributeGroup ref="xlink:simpleLink"/>
    <attribute name="nilReason" type="gml:NilReasonType"/>
    <attribute ref="gml:remoteSchema"/>
</attributeGroup>
```

with the following definitions from Xlink (see 8.1):

```
<attributeGroup name="simpleLink">
    <attribute name="type" type="string" fixed="simple" form="qualified"/>
    <attribute ref="xlink:href"/>
    <attribute ref="xlink:role"/>
    <attribute ref="xlink:arcrole"/>
    <attribute ref="xlink:title"/>
    <attribute ref="xlink:show"/>
    <attribute ref="xlink:actuate"/>
</attributeGroup>
```

The value of a GML property that carries an `xlink:href` attribute is the resource returned by traversing the link.

The `nilReason` attribute may be used in a property element that is nillable to indicate a reason for a nil value.

NOTE All components in the attribute group are optional.

7.2.3.3 abstractAssociationRole, AssociationRoleType

To support the encoding of properties that may have complex content, a basic pattern for property elements is provided in the GML schema as follows:

```
<element name="abstractAssociationRole" type="gml:AssociationRoleType" abstract="true"/>

<complexType name="AssociationRoleType">
    <sequence minOccurs="0">
        <any namespace="##any"/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

Applying this pattern shall restrict the multiplicity of objects in a property element using this content model to exactly one. An instance of this type shall contain an element representing an object, or serve as a pointer to a remote object.

Applying the pattern to define an application schema specific property type allows to restrict

— the inline object to specified object types,

— the encoding to "by reference only“ (see 7.2.3.7),

— the encoding to "inline only“ (see 7.2.3.8).

NOTE 1 The declaration of `gml:abstractAssociationRole` and its accompanying type definition is provided for convenience, to act as a template or pattern for the construction of property elements in application schemas. There is no requirement for specific properties to use XML Schema type derivation from gml:AssociationType to create properties in a conformant GML application schema. This contrasts with the requirement that the content model for all identifiable objects shall derive from gml:AbstractGMLType, and for all features from gml:AbstractFeatureType.

NOTE 2 While `gml:abstractAssociationRole` is abstract, its type `gml:AssociationRoleType` is not, because the same type is used by the instantiable `gml:member` property (see 7.2.3.10). Note also that this property has been deprecated.

7.2.3.4 Inline or by reference?

The `any` element in the content model for properties is optional. In combination with the component cardinalities in `gml:AssociationAttributeGroup` this means that an element of this type may have a content element or xlink attributes. GML property elements which follow this pattern may be used to attach values either inline or by-reference.

EXAMPLE A utility property provided for features is “centerOf”. This may be used to indicate a spatial location inline as follows:

```
<gml:centerOf>
  <gml:Point gml:id="point96" srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
    <gml:pos>-31.936 15.834</gml:pos>
  </gml:Point>
</gml:centerOf>
```

which uses the `gml:Point` object as defined in the GML geometry schemas (described in 10.2). The same property element may be used to indicate a location by reference as follows:

```
<gml:centerOf xlink:href="http://my.big.org/locations/point53"/>
```

where “http://my.big.org/location/point53” identifies a point (a `gml:Point` element) supplied by the service indicated.

However, a property element following this pattern may have no content or attributes, or it may have both content and attributes, and still be XML Schema-valid. It is not possible to constrain the co-occurrence of content or attributes, so it is not possible to use W3C XML Schema to restrict a property to be either inline or by-reference only.

If both a link and content are present in an instance of a property element, then the object found by traversing the `xlink:href` link shall be the normative value of the property. The object included as content shall be used by the data recipient only if the remote instance cannot be resolved; this may be considered to be a "cached" version of the object.

NOTE Most GML-Object-valued properties in the GML schema can be encoded either inline or by-reference. However, using a GML profile (see Clause 20) it is possible to restrict the usage to “inline only” or “by-reference only”.

7.2.3.5 Ownership of property values

Encoding a GML property inline vs. by-reference shall not imply anything about the “ownership” of the contained or referenced GML object, i.e. the encoding style shall not imply any “deep-copy” or “deep-delete” semantics. To express ownership over the contained or referenced GML object, the `gml:OwnershipAttributeGroup` attribute group may be added to object-valued property elements. If the attribute group is not part of the content model of such a property element, then the value may not be “owned”.

The attribute group is defined as follows:

```
<attributeGroup name="OwnershipAttributeGroup">
    <attribute name="owns" type="boolean" default="false"/>
```



```
</attributeGroup>
```

When the value of the `owns` attribute is “true”, the existence of inline or referenced object(s) depends upon the existence of the parent object.

EXAMPLE If a property “hasOwner” is represented in an instance document as

```
<Parcel gml:id=”p123”>
  <hasOwner xlink:href=”urn:x-abc:id:o123”/>
</Parcel>
```

then the referenced object, e.g. a person, is not “owned” by the parcel feature, i.e. the person feature will not be deleted, if the parcel is deleted. However, if a property is encoded with an attribute owns=”true”, for example

```
<Car gml:id=”c123”>
  <hasParts owns=”true” xlink:href=”urn:x-abc:id:x123”/>
  <!-- … -->
</Car>
```

then the referenced object is “owned” by the car feature, i.e. the part will be deleted, if the car is deleted.

7.2.3.6 abstractStrictAssociationRole

The constraint that the value of a property may be either embedded inline or specified by an xlink reference may be described precisely using the auxiliary constraint language Schematron (see ISO/IEC 19757-3). The abstract, global elements `gml:abstractAssociationRole` and `gml:abstractStrictAssociationRole` both use `gml:AssociationRoleType`, but the following schema fragments shows how an element declaration may accompanied by a Schematron constraint to limit the property to act in either inline or by-reference mode, but not both.

```
<element name="abstractAssociationRole" type="gml:AssociationRoleType" abstract="true"/>

<element name="abstractStrictAssociationRole" type="gml:AssociationRoleType" abstract="true"/>

<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xlink="http://www.w3.org/1999/xlink" xml:lang="en">
	<sch:title>Schematron constraints for GML / ISO 19136</sch:title>
	<sch:ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron"/>
	<sch:ns prefix="gml" uri="http://www.opengis.net/gml/3.2"/>
	<sch:ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
	<sch:pattern>
		<sch:rule context="gml:abstractStrictAssociationRole">
			<sch:assert test="not(@xlink:href and (*|text()))">Property element may not carry both a reference to an object
and contain an object.</sch:assert>
			<sch:assert test="@xlink:href | (*|text())">Property element shall either carry a reference to an object or contain
an object.</sch:assert>
		</sch:rule>
	</sch:pattern>
</sch:schema>
```

NOTE Some XML validators will process the Schematron constraints automatically. Otherwise, the Schematron code may be treated merely as a formal description of the required constraint. It is included here primarily as an illustration of how this might be used for specific purposes by application schema developers.

7.2.3.7 abstractReference, ReferenceType

In order to support the encoding of properties whose value is provided remotely by-reference, the following components are provided:


```
<element name="abstractReference" type="gml:ReferenceType" abstract="true"/>

<complexType name="ReferenceType">
    <sequence/>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

The element `gml:abstractReference` is abstract, and thus may be used as the head of a substitution group of more specific elements providing a value by-reference.

NOTE While `gml:abstractReference` is abstract, its type `gml:ReferenceType` is not, because the type is intended to be used in application schemas directly, if a property element shall use a “by-reference only” encoding.

7.2.3.8 abstractInlineProperty, InlinePropertyType

In order to support the encoding of properties whose value is provided inline, the following components are provided:

```
<element name="abstractInlineProperty" type="gml:InlinePropertyType" abstract="true"/>

<complexType name="InlinePropertyType">
    <sequence>
        <any namespace=”##any”/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The element `gml:abstractInlineProperty` is abstract, and thus may be used as the head of a substitution group of more specific elements providing a value inline.

7.2.3.9 Properties representing the same relationship

If the value of an object property is another object and that object contains also a property for the association between the two objects, then this name of the reverse property may be encoded in a `gml:reversePropertyName` element in an appinfo annotation of the property element to document the constraint between the two properties. The value of the element shall contain the qualified name of the property element.

```
<element name="reversePropertyName" type="string"/>
```

EXAMPLE

```
<element name="owner" type="ex:PersonPropertyType" minOccurs=”0”>
    <annotation>
        <appinfo>
            <gml:reversePropertyName>ex:owns</gml:reversePropertyName>
        </appinfo>
    </annotation>
</element>
…
<complexType name="PersonPropertyType">
    <sequence minOccurs="0">
        <element ref="ex:Person"/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```


7.2.3.10 Properties of value objects

Value objects, see 16.4, are special objects in the sense that in the case of a single property that can be represented by a single literal value, the value appears as the direct content of object element without an extra element for the property.

EXAMPLE <gml:Integer>5</gml:Integer> is used instead of, for example, <gml:Integer> <gml:value>5</gml:value> </gml:Integer>.

7.2.4 Standard properties of GML objects

7.2.4.1 Derivation from AbstractGMLType

XML Schema types for all GML objects derive directly or indirectly from `gml:AbstractGMLType`. This means that all GML objects inherit certain standard properties that are included in the content model of `gml:AbstractGMLType`.

7.2.4.2 description

The value of this property is a text description of the object. `gml:description` uses `gml:StringOrRefType` (see 0) as its content model, i.e. it should contain a simple text string content.

```
<element name="description" type="gml:StringOrRefType"/>
```

NOTE The use of gml:description to reference an external description has been deprecated and replaced by the gml:descriptionReference property (see 7.2.4.3).

7.2.4.3 descriptionReference

The value of this property is a remote text description of the object. The xlink:href attribute of the `gml:descriptionReference` property references the external description.

```
<element name="descriptionReference" type="gml:ReferenceType"/>
```

7.2.4.4 name, identifier

The `gml:name` property provides a label or identifier for the object, commonly a descriptive name.

An object may have several names, typically assigned by different authorities. `gml:name` uses the `gml:CodeType` content model. The authority for a name is indicated by the value of its (optional) `codeSpace` attribute. The name may or may not be unique, as determined by the rules of the organization responsible for the `codeSpace`. In common usage there will be one name per authority, so a processing application may select the name from the `codeSpace` that it prefers.

```
<element name="name" type="gml:CodeType"/>
```

Often, a special identifier is assigned to an object by the authority that maintains the feature with the intention that it is used in references to the object. For such cases, the `codeSpace` shall be provided. That identifier is usually unique either globally or within an application domain. `gml:identifier` is a predefined property for such identifiers.

EXAMPLE UUIDs and URNs are commonly used globally unique identifiers.

```
<element name="identifier" type="gml:CodeWithAuthorityType"/>
```


7.2.4.5 id

The attribute `gml:id` supports provision of a handle for the XML element representing a GML object. Its use is recommended for all GML objects. In particular, all GML objects that are intended to be referenced should carry an attribute `gml:id`. For some GML object types, the attribute `gml:id` is mandatory.

```
<attribute name="id" type="ID"/>
```

It is of XML type **ID**, so is constrained to be unique in the XML document within which it occurs. An external identifier for the XML element representing the GML object in the form of a URI may be constructed using standard methods (IETF RFC 2396). This is done by concatenating the URI for the document, a fragment separator “#”, and the value of the attribute of XML type ID.

7.2.5 Collections of GML objects

7.2.5.1 AbstractMemberType and derived property types

To create a collection of GML objects that are not all features, a property type shall be derived by extension from `gml:AbstractMemberType`.

```
<complexType name="AbstractMemberType" abstract="true">
	<sequence/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The derived property type shall follow one of the patterns specified in 7.2.3 and may set the multiplicity of the objects in the collection as required for its intended use.

This abstract property type is intended to be used only in object types where software shall be able to identify that an instance of such an object type is to be interpreted as a collection of objects.

EXAMPLE See `gml:DictionaryEntryType` in 15.2.3 for such a property type.

By default, this abstract property type does not imply any ownership of the objects in the collection. The `owns` attribute of `gml:OwnershipAttributeGroup` may be used on a property element instance to assert ownership of an object in the collection. A collection shall not own an object already owned by another object.

7.2.5.2 GML object collections, AggregationAttributeGroup

A GML object collection is any `gml:AbstractObject` with a property element in its content model whose content model is derived by extension from `gml:AbstractMemberType`.

EXAMPLE `gml:Dictionary` is a GML object collection, because the content model of property `gml:dictionaryEntry` specified in 15.2.3 is derived by extension from `gml:AbstractMemberType`.

In addition, the complex type describing the content model of the GML object collection may also include a reference to the attribute group `gml:AggregationAttributeGroup` to provide additional information about the semantics of the object collection. This information may be used by applications to group GML objects, and optionally to order and index them.

```
<attributeGroup name="AggregationAttributeGroup">
	<attribute name="aggregationType" type="gml:AggregationType"/>
</attributeGroup>
```

The allowed values for the `aggregationType` attribute are defined by `gml:AggregationType`. See 8.4 of ISO/IEC 11404:1996 for the meaning of the values in the enumeration.


```
<simpleType name="AggregationType" final="#all">
    <restriction base="string">
        <enumeration value="set"/>
        <enumeration value="bag"/>
        <enumeration value="sequence"/>
        <enumeration value="array"/>
        <enumeration value="record"/>
        <enumeration value="table"/>
    </restriction>
</simpleType>
```

NOTE 1 If a collection of aggregation type “array” is implemented in an application schema, then the array type in the application schema needs to model the additional information to cope with indexing.

NOTE 2 If a collection of aggregation type “table” is implemented in an application schema, then the table type in the application schema needs to model the additional information to add the required information about the fields and their structure.

7.2.6 Metadata

To associate metadata described by any XML Schema with a GML object, a property element shall be defined whose content model is derived by extension from `gml:AbstractMetadataPropertyType`.

The value of such a property shall be metadata. The content model of such a property type, i.e. the metadata application schema shall be specified by the GML application schema.

```
<complexType name="AbstractMetadataPropertyType" abstract="true">
    <sequence/>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The property type derived from `gml:AbstractMetadataPropertyType` shall follow one of the patterns specified for GML property types in 7.2.3.

By default, this abstract property type does not imply any ownership of the metadata. The `owns` attribute of `gml:OwnershipAttributeGroup` may be used on a metadata property element instance to assert ownership of the metadata.

If metadata following the conceptual model of ISO 19115 is to be encoded in a GML document, the corresponding Implementation Specification specified in ISO/TS 19139 shall be used to encode the metadata information.

EXAMPLE 1 Assume that a feature type "Road" can be associated with two metadata elements, a data quality property "horizontalAbsolutAccuracy" and a generic ISO/TS 19139 "metadata" property.

This may be mapped in the application schema as follows by bundling the metadata properties in a complex property:

```
<complexType name="RoadType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <!-- ... -->
        <element name="roadMetadata" type="ex:RoadMetadataPropertyType"/>
        <!-- ... -->
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```
<complexType name="RoadMetadataPropertyType">
    <complexContent>
        <extension base="gml:AbstractMetadataPropertyType">
            <sequence>
                <element ref="ex:RoadMetadata"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="RoadMetadataType">
    <complexContent>
        <extension base="gmx:AbstractObjectMetadata_Type">
            <sequence>
                <element name="horizontalAbsoluteAccuracy" minOccurs="0"
                  type="gmd:DQ_AbsoluteExternalPositionalAccuracy_PropertyType"/>
                <element name="metadata" minOccurs="0" type="gmd:MD_Metadata_PropertyType"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<element name="RoadMetadata" type="myAs:RoadMetadataType" substitutionGroup="gmx:AbstractObjectMetadata"/>
```

Then, an instance of a Road feature could look like:

```
<ex:Road>
  <!-- … -->
  <ex:roadMetadata>
    <ex:RoadMetadata>
      <ex:horizontalAbsoluteAccuracy>
        <gmd:DQ_AbsoluteExternalPositionalAccuracy>
          <!-- The DQ_Element subelements are not detailed -->
        </gmd:DQ_AbsoluteExternalPositionalAccuracy>
      </ex:horizontalAbsoluteAccuracy>
      <ex:metadata>
        <gmd:MD_Metadata>
          <!-a full set of ISO/TS 19139 metadata elements -->
        </gmd:MD_Metadata>
      </ex:metadata>
    </ex:RoadMetadata>
  </ex:roadMetadata>
  <!-- … -->
</myAs:Road>
```

An alternative encoding representing the metadata properties as separate properties of the feature would be:

```
<complexType name="RoadType">
    <complexContent>
        <extension base="gml:AbstractFeatureType">
            <sequence>
               <!-- ... -->
               <element name="horizontalAbsoluteAccuracy" minOccurs="0">
                <complexType>
                  <complexContent>
                      <extension base="gml:AbstractMetadataPropertyType">
                          <sequence>
                            <element ref="gmd:DQ_AbsoluteExternalPositionalAccuracy"/>
                          </sequence>
                      </extension>
                  </complexContent>
                </complexType>
```

```
          </element>
          <!-- ... -->
          <element name="metadata" minOccurs="0">
           <complexType>
             <complexContent>
                <extension base="gml:AbstractMetadataPropertyType">
                    <sequence>
                      <element ref="gmd:MD_Metadata"/>
                    </sequence>
                </extension>
             </complexContent>
           </complexType>
          </element>
          <!-- ... -->
        </sequence>
      </extension>
   </complexContent>
</complexType>
```

The instance example would then look like this:

```
<ex:Road>
  <!-- ... -->
  <ex:horizontalAbsoluteAccuracy>
    <gmd:DQ_AbsoluteExternalPositionalAccuracy>
      <!-- The DQ_Element subelements are not detailed here -->
    </gmd:DQ_AbsoluteExternalPositionalAccuracy>
  </ex:horizontalAbsoluteAccuracy>
  <!-- ... -->
  <ex:metadata>
    <gmd:MD_Metadata>
      <!-a full set of ISO/TS 19139 metadata elements -->
    </gmd:MD_Metadata>
  </ex:metadata>
  <!-- ... -->
</ex:Road>
```

EXAMPLE 2 Assume that a dataset shall be enabled to contain Dublin Core metadata elements. This may be mapped in the application schema as follows:

```
<import namespace="http://www.purl.org/dc/terms/" schemaLocation="http://schemas.opengis.net/csw/2.0.0/rec-
dcterms.xsd"/>

<complexType name="DatasetType">
   <complexContent>
      <extension base="gml:AbstractFeatureType">
         <sequence>
            <!-- ... -->
            <element name="generalMetadata" type="ex:DublinCoreMetadataPropertyType"/>
            <!-- ... -->
         </sequence>
      </extension>
   </complexContent>
</complexType>

<complexType name="GeneralMetadataPropertyType">
   <complexContent>
      <extension base="gml:AbstractMetadataPropertyType">
         <sequence>
            <element ref="ex:DublinCoreMetadata"/>
         </sequence>
      </extension>
```



```
    </complexContent>
</complexType>

<element name="DublinCoreMetadata">
    <complexType name="DublinCoreMetadataType">
        <sequence>
            <group ref="dct:DCMI-terms" xmlns:dct="http://www.purl.org/dc/terms/"/>
        </sequence>
    </complexType>
</element>
```

An instance example could look like this:

```
<ex:Dataset>
  <!-- ... -->
  <ex:generalMetadata>
    <ex:DublinCoreMetadata>
        <dc:title>Vector Smart Map Level 0</dc:title>
        <dct:abstract>Vector Map: a general purpose database design to support GIS applications</dct:abstract>
        <dc:publisher>US National Geospatial-Intelligence Agency</dc:publisher>
        <dc:format>VPF</dc:format>
        <dc:coverage>world</dc:coverage>
        <dc:language>en</dc:language>
        <!-- … -->
    </ex:DublinCoreMetadata>
  </ex:generalMetadata>
  <!-- ... -->
</ex:Dataset>
```

8 GML schema — Xlinks and basic types

8.1 Xlinks — Object associations and remote properties

The normative Xlink specification is available from W3C.

NOTE A schema document xlinks.xsd is provided as part of the GML schema documents in Annex C.

Xlink components are used in GML to implement associations between objects by reference. GML property elements (see 7.2.3) may carry Xlink attributes, which support the encoding of an association relationship by reference, the name of the property element denoting the target role in the association. The most important Xlink component is:

`xlink:href` identifier of the resource which is the target of the association, given as a URI

The appearance of an `xlink:href` on a GML property indicates that the value of the property shall be found by traversing the link, that is the value is pointed to by the value of the `xlink:href` attribute. Following the terminology of Xlink, GML properties with `xlink:href` attributes are sometimes referred to as remote properties.

The other Xlink components are used to indicate additional semantics of the relationship. The most useful of these are

`xlink:role` description of the nature of the target resource, given as a URI

`xlink:arcrole` description of the role or purpose of the target resource in relation to the present resource, given as a URI

`xlink:title` description of the association or the target resource, given as text

For complete definitions of these and other Xlink components, including their use in extended Xlink association maps, refer to the Xlink specification.

A URI reference [URI] is defined as an optional choice between an absolute or relative URI, followed by fragment identifier that consists of a crosshatch ("#") and additional reference information. For GML object properties and remote associations, this additional reference information shall be one of the following:

— a shorthand (formerly called "barename") XPointer [XPointer Framework] consisting of the value of the gml:id attribute of a GML object, or

— an element() scheme based XPointer [XPointer element()], or

— an xpointer() scheme based XPointer [XPointer xpointer()] containing an XPath [XPath] expression that selects a GML object, optionally preceded by one or more xmlns() scheme based XPointer(s) [XPointer xmlns()] that define the namespace prefixes used in the XPath expression.

A URI that does not contain an absolute or relative URI, but that consists entirely of a fragment identifier, refers to a GML object elsewhere in the same GML document.

Absolute and relative URIs may include a query component that consists of a question mark ("?") followed by a query to be interpreted by the resource. For GML object properties and remote associations, any such query shall be a request to a service that returns a GML object. The URI containing such a query may or may not make use of a fragment identifier, depending on the request syntax defined by the service.

In the GML schema, simple Xlinks are used exclusively to denote association roles of GML objects and to denote remotely referenced property values.

EXAMPLE 1 A reference to an object element in the same GML document may be encoded as:

```
<myProperty xlink:href="#o1"/>
```

EXAMPLE 2 A reference to an object element in a remote XML document using the gml:id value of that object may be encoded as:

```
<myProperty xlink:href="http://my.big.org/test.xml#o1"/>
```

EXAMPLE 3 A reference to an object element in a remote XML document (or GML object repository) using the gml:identifier property value of that object may be encoded as:

```
<myProperty xlink:href="http://my.big.org/test.xml#element
    (//gml:GeodeticCRS[./gml:identifier[@codeSpace="http://www.opengis.net/def/crs/EPSG/0/"]="4326"])"/>
```

EXAMPLE 4 A reference to an object element with a uniform resource name may be encoded as follows (note that a URN resolver is required to resolve the URN and access the referenced object):

```
<myProperty xlink:href="urn:ogc:def:crs:EPSG::4326"/>
```

The IDREF data type and the unique, key, and keyref elements defined in the XML and XML Schema specifications provide alternative identification and linking mechanisms to the ID data type and Xlink reference for use within a single XML document. Although these XML components may be used in XML Schemas, they have no normative role in GML, and shall not be used to denote association roles of GML objects or remotely referenced property values.

8.2 Basic types

8.2.1 Overview

W3C XML Schema provides a set of built-in “simple” types which define methods for representing values as literals without internal markup. These are described in W3C XML Schema Part 2:2001. Because GML is an XML encoding in which instances are described using XML Schema, these simple types shall be used as far as possible and practical for the representation of data types. W3C XML Schema also provides methods for defining

— new simple types by restriction and combination of the built-in types, and

— complex types, with simple content, but which also have XML attributes.

In many places where a suitable built-in simple type is not available, simple content types derived using the XML Schema mechanisms are used for the representation of data types in GML.

A set of these simple content types that are required by several GML components are defined in the basicTypes schema, as well as some elements based on them. These are primarily based around components needed to record amounts, counts, flags and terms, together with support for exceptions or null values.

NOTE The basic types and elements are described in the basicTypes schema document in Annex C. The schema is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:basicTypes:3.2.1

8.2.2 Relationship with ISO/TS 19103

ISO/TS 19103 defines basic types for the conceptual schemas in the ISO 19100 series of International Standards. GML implements a subset of these basic types as described in D.2.2.

NOTE Some of the ISO/TS 19103 basic types are specified in other schema documents of the GML schema: units of measure are specified in Clause 16 and vector in 10.1.4.5.

8.2.3 Simple types

8.2.3.1 NilReasonType

`gml:NilReasonType` defines a content model that allows recording of an explanation for a void value or other exception.

```
<simpleType name="NilReasonEnumeration">
  <union>
    <simpleType>
      <restriction base="string">
        <enumeration value="inapplicable"/>
        <enumeration value="missing"/>
        <enumeration value="template"/>
        <enumeration value="unknown"/>
        <enumeration value="withheld"/>
      </restriction>
    </simpleType>
    <simpleType>
      <restriction base="string">
        <pattern value="other:\w{2,}"/>
      </restriction>
    </simpleType>
  </union>
</simpleType>
```

```
<simpleType name="NilReasonType">
  <union memberTypes="gml:NilReasonEnumeration anyURI"/>
</simpleType>
```

`gml:NilReasonType` is a union of the following enumerated values:

— **"inapplicable"**: there is no value

— **"missing"**: the correct value is not readily available to the sender of this data. Furthermore, a correct value may not exist

— **"template"**: the value will be available later

— **"unknown"**: the correct value is not known to, and not computable by, the sender of this data. However, a correct value probably exists

— **"withheld"**: the value is not divulged

— **"other:"+text**: other brief explanation, where **text** is a string of two or more characters with no included spaces

and

— **anyURI** which should refer to a resource which describes the reason for the exception

A particular community may choose to assign more detailed semantics to the standard values provided. Alternatively, the URI method enables a specific or more complete explanation for the absence of a value to be provided elsewhere and indicated by-reference in an instance document.

`gml:NilReasonType` is used as a member of a union in a number of simple content types defined below (see 8.2.3.4, 8.2.4.1, 8.2.4.2, 8.2.4.3) where it is necessary to permit a value from the NilReasonType union as an alternative to the primary type.

8.2.3.2 Elements declared to be “nillable”

The XML Schema attribute `nillable` may be included in any element declaration within a schema.

NOTE By default the schema attribute `nillable` has a value of “false”.

EXAMPLE 1 The following element declarations illustrate the use of the `nillable` attribute:

```
<element name=”amount” type=”double” nillable=”true”/>
<element ref=”my:amount” nillable=”true”/>
```

By declaring an element as nillable (`nillable="true"`), an instance of that element may omit its content in cases where an empty value would normally not be schema valid by supplying an attribute `nil` from the XML Schema Instance namespace with the value “true”.

EXAMPLE 2 Elements that have been declared with this `nillable="true"` in the schema may appear in instance documents as follows:

```
<my:amount>34.567</my:amount>
<my:amount xsi:nil=”true”/>
```


Declaring an element to be nil is an implementation of the “Void” data type of ISO/IEC 11404, i.e. represents “an object whose presence is syntactically or semantically required, but carries no information in a given instance” [ISO/IEC 11404].

NOTE This is different to an element declaration with the cardinality attribute set to make the element optional, such as:

```
<element name=”amount” type=”double” minOccurs=”0”/>
```

which allows the element to be omitted in the instance entirely.

In some situations where it is required to declare an element in an application schema nillable, it may be convenient to also add an attribute of type gml:NilReasonType.

EXAMPLE 3 The application schema components

```
<element name="amount" nillable=”true”>
 <complexType>
  <simpleContent>
   <extension base="double">
    <attribute name="nilReason" type="gml:NilReasonType"/>
   </extension>
  </simpleContent>
 </complexType>
</element>

<element name=”money” type=my:NRMeasureType” nillable=”true”/>

<complexType name=”NRMeasureType”>
 <simpleContent>
  <extension base="decimal">
   <attribute name=”uom” type=”token” use=”required”/>
   <attribute name="nilReason" type="gml:NilReasonType"/>
  </extension>
 </simpleContent>
</complexType>
```

would allow the instances to be augmented with an additional attribute explaining the absence of a value, such as

```
<my:amount xsi:nil=”true” nilReason=”unknown”/>
<my:money xsi:nil=”true” nilReason=”other:myDaughterSpentIt” uom=”AUD”/>
```

In the GML schema and in GML application schemas, the “nillable” and “nilReason” construction may be used on elements representing GML properties (see 7.2.3). This allows properties that are part of the content of objects and features in GML and GML application languages to be declared to be mandatory, while still permitting them to appear in an instance document with no value.

NOTE Both simple content and complex content elements may be declared as nillable, so this construction allows a uniform syntax for properties with void values.

8.2.3.3 SignType

`gml:SignType` is a convenience type with values “+” (plus) and “-” (minus).

```
<simpleType name="SignType">
 <restriction base="string">
```

```
    <enumeration value="-"/>
    <enumeration value="+"/>
  </restriction>
</simpleType>
```

NOTE Elements or attributes of this type are used in various places, e.g. to indicate the direction of topological objects with "+" for forwards, or "-" for backwards.

8.2.3.4 booleanOrNilReason, doubleOrNilReason, integerOrNilReason, NameOrNilReason, stringOrNilReason

The types `gml:booleanOrNilReason`, `gml:doubleOrNilReason`, `gml:integerOrNilReason`, `gml:NameOrNilReason`, `gml:stringOrNilReason` provide extensions to the respective XML Schema built-in simple types to allow a choice of either a value of the built-in simple type or a reason for a nil value. They are constructed as follows:

```
<simpleType name="booleanOrNilReason">
  <union memberTypes="gml:NilReasonEnumeration boolean anyURI"/>
</simpleType>

<simpleType name="doubleOrNilReason">
  <union memberTypes="gml:NilReasonEnumeration double anyURI"/>
</simpleType>

<simpleType name="integerOrNilReason">
  <union memberTypes="gml:NilReasonEnumeration integer anyURI"/>
</simpleType>

<simpleType name="NameOrNilReason">
  <union memberTypes="gml:NilReasonEnumeration Name anyURI"/>
</simpleType>

<simpleType name="stringOrNilReason">
  <union memberTypes="gml:NilReasonEnumeration string anyURI"/>
</simpleType>
```

8.2.3.5 CodeType, CodeWithAuthorityType

`gml:CodeType` is a generalized type to be used for a term, keyword or name.

```
<complexType name="CodeType">
  <simpleContent>
    <extension base="string">
      <attribute name="codeSpace" type="anyURI"/>
    </extension>
  </simpleContent>
</complexType>
```

It adds an XML attribute `codeSpace` to a term, where the value of the `codeSpace` attribute (if present) shall indicate a dictionary, thesaurus, classification scheme, authority, or pattern for the term.

EXAMPLE The gmlBase schema contains an element declaration using this type (see 8.2.3.5):

```
<element name="name" type="gml:CodeType"/>
```

so a corresponding element might appear in an instance document as follows :

```
<gml:name codeSpace = “http://www.ukusa.gov/placenames”>St Paul</gml:name>
```


In this example “St Paul” is asserted to be a meaningful name in accordance with http://www.ukusa.gov/placenames. Note that in all cases the rules for the values, including such things as uniqueness constraints, are set by the authority responsible for the codeSpace.

The derived type `gml:CodeWithAuthorityType` requires that the `codeSpace` attribute is provided in an instance.

```
<complexType name="CodeWithAuthorityType">
	<simpleContent>
		<restriction base="gml:CodeType">
			<attribute name="codeSpace" type="anyURI" use="required"/>
		</restriction>
	</simpleContent>
</complexType>
```

8.2.3.6 MeasureType, UomIdentifier

`gml:MeasureType` supports recording an amount encoded as a value of XML Schema double, together with a units of measure indicated by an attribute `uom`, short for “units of measure”. The value of the `uom` attribute identifies a reference system for the amount, usually a ratio or interval scale.

`gml:MeasureType` is defined as follows:

```
<complexType name="MeasureType">
  <simpleContent>
    <extension base="double">
      <attribute name="uom" type="gml:UomIdentifier" use="required"/>
    </extension>
  </simpleContent>
</complexType>
```

EXAMPLE An application schema may contain an element declaration using this type

```
<element name = “height” type = “gml:MeasureType”/>
```

Elements corresponding to this might appear in an data instance document as follows:

```
<height uom=“m">1.4224</height>

<height uom=“http://www.equestrian.org/units/hands">14</height>
```

where the value of the uom attribute identifies the unit of measure or a resource that defines the unit of measure.

The simple type `gml:UomIdentifer` defines the syntax and value space of the unit of measure identifier. This is a union type defined as follows:

```
<simpleType name="UomIdentifier">
	<union memberTypes="gml:UomSymbol gml:UomURI"/>
</simpleType>
```

The first member of the union type, `gml:UomSymbol`, is defined as follows:

```
<simpleType name="UomSymbol">
	<restriction base="string">
		<pattern value="[^: \n\r\t]+"/>
	</restriction>
</simpleType>
```

This type specifies a character string of length at least one, and restricted such that it must not contain any of the following characters: ":" (colon), " " (space), (new line), (carriage return), (tab). This allows values corresponding to familiar abbreviations, such as "kg", "m/s", etc.

NOTE It is recommended that the symbol be an identifier for a unit of measure as specified in the "Unified Code of Units of Measure" (UCUM) (http://aurora.regenstrief.org/UCUM). This provides a set of symbols and a grammar for constructing identifiers for units of measure that are unique, and may be easily entered with a keyboard supporting the limited character set known as 7-bit ASCII. ISO 2955 formerly provided a specification with this scope, but was withdrawn in 2001. UCUM largely follows ISO 2955 with modifications to remove ambiguities and other problems.

The second member of the union type, `gml:UomURI`, is defined as follows:

```
<simpleType name="UomURI">
    <restriction base="anyURI">
        <pattern value="([a-zA-Z][a-zA-Z0-9\-\+\.]*:|\.\./|\./|#).*"/>
    </restriction>
</simpleType>
```

This type specifies a URI, restricted such that it must start with one of the following sequences: "#", "./", "../", or a string of characters followed by a ":". These patterns ensure that the most common URI forms are supported, including absolute and relative URIs and URIs that are simple fragment identifiers, but prohibits certain forms of relative URI that could be mistaken for unit of measure symbol[1)].

NOTE It is possible to re-write such a relative URI to conform to the restriction (e.g. "./m/s").

In an instance document, on elements of type `gml:MeasureType` the mandatory `uom` attribute shall carry a value corresponding to either

— a conventional unit of measure symbol,

— a link to a definition of a unit of measure that does not have a conventional symbol, or when it is desired to indicate a precise or variant definition.

GML components for the latter purpose are defined in 16.2.

8.2.3.7 CoordinatesType

```
<complexType name="CoordinatesType">
  <simpleContent>
    <extension base="string">
      <attribute name="decimal" type="string" default="."/>
      <attribute name="cs" type="string" default=","/>
      <attribute name="ts" type="string" default="&#x20;"/>
    </extension>
  </simpleContent>
</complexType>
```

This type is deprecated for tuples with ordinate values that are numbers.

`gml:CoordinatesType` is a text string, intended to be used to record an array of tuples or coordinates.

1) e.g. "m/s".

While it is not possible to enforce the internal structure of the string through schema validation, some optional attributes have been provided in previous versions of GML to support a description of the internal structure. These attributes are deprecated. The attributes were intended to be used as follows:

Decimal *symbol used for a decimal point (default="." a stop or period)*

cs *symbol used to separate components within a tuple or coordinate string (default="," a comma)*

ts *symbol used to separate tuples or coordinate strings (default=" " a space)*

Since it is based on the XML Schema string type, `gml:CoordinatesType` may be used in the construction of tables of tuples or arrays of tuples, including ones that contain mixed text and numeric values.

EXAMPLE <my:tupleList>bettong,357.,2.3 skink,140.,0.75 wombat,770.,17.5</my:tupleList>

8.2.4 Lists

8.2.4.1 booleanList, doubleList, integerList, NameList, NCNameList, QNameList, booleanOrNilReasonList, NameOrNilReasonList, doubleOrNilReasonList, integerOrNilReasonList

A set of types for lists of simple values are constructed in accordance with the following patterns as follows:

```
<simpleType name="booleanList">
  <list itemType="boolean"/>
</simpleType>

<simpleType name="doubleList">
  <list itemType="double"/>
</simpleType>

<simpleType name="integerList">
  <list itemType="integer"/>
</simpleType>

<simpleType name="NameList">
  <list itemType="Name"/>
</simpleType>

<simpleType name="NCNameList">
  <list itemType="NCName"/>
</simpleType>

<simpleType name="QNameList">
  <list itemType="QName"/>
</simpleType>

<simpleType name="booleanOrNilReasonList">
  <list itemType="gml:booleanOrNilReason"/>
</simpleType>

<simpleType name="NameOrNilReasonList">
  <list itemType="gml:NameOrNilReason"/>
</simpleType>

<simpleType name="doubleOrNilReasonList">
  <list itemType="gml:doubleOrNilReason"/>
```



```
</simpleType>

<simpleType name="integerOrNilReasonList">
  <list itemType="gml:integerOrNilReason"/>
</simpleType>
```

These types are defined as a list of values of the respective XML Schema built-in simple types, or of the union types specified in previous subclauses. The `…OrNilReasonList` types support reasons for nil values interspersed within a list.

NOTE 1 These types are provided as convenience types. They may be helpful in cases where a simple content type is to be defined that is a union of such a list and another simple content type.

NOTE 2 Some of the types start with an upper case letter, some with a lower case letter. The reason is that the case of the XML Schema base type has been preserved in the GML types for clarity.

NOTE 3 An element which uses one of these types will contain a whitespace-separated list of members of the relevant type (see http://www.w3.org/TR/xmlschema-2/#atomic-vs-list for more details of the XML list structure).

NOTE 4 None of the list types defined here use an XML Schema string as an item. The reason for this is that a string may include embedded spaces, linefeeds, etc (http://www.w3.org/TR/xmlschema-2/#string). Since whitespace acts as the item separator in a list instance, there would be ambiguity in identifying items that potentially contain whitespace. On the other hand, an instance of the XML Schema Name type may not contain whitespace (http://www.w3.org/TR/2000/WD-xml-2e-20000814#NT-Name), so this may be used safely in a list context. The corollary of this is that if a term may contain whitespace, then such a term may not occur in a list instance.

8.2.4.2 CodeListType, CodeOrNilReasonListType

The two types `gml:CodeListType` and `gml:CodeOrNilReasonListType` provide for lists of terms. The schema definitions are as follows:

```
<complexType name="CodeListType">
  <simpleContent>
    <extension base="gml:NameList">
      <attribute name="codeSpace" type="anyURI"/>
    </extension>
  </simpleContent>
</complexType>

<complexType name="CodeOrNilReasonListType">
  <simpleContent>
    <extension base="gml:NameOrNilReasonList">
      <attribute name="codeSpace" type="anyURI"/>
    </extension>
  </simpleContent>
</complexType>
```

The values in an instance element of `gml:CodeListType` shall all be valid in accordance with the rules of the dictionary, classification scheme, or authority identified by the value of its `codeSpace` attribute.

EXAMPLE An application schema may contain an element declaration using this type

```
<element name = “species”  type = “gml:CodeListType”/>
```

so a corresponding element might appear in an instance document as follows :

```
<species codeSpace=”http://my.big.org/florelegium”>dryandra banksia hardenbergia lavender</species>
```

where the listed items are from “http://my.big.org/florelegium” which is a (hypothetical) list of flowers.

An instance element of `gml:CodeOrNilReasonListType` may also include embedded values from `gml:NilReasonType`. It is intended to be used in situations where a term or classification is expected, but the value may be absent for some reason.

8.2.4.3 MeasureListType, MeasureOrNilReasonListType

The two types `gml:MeasureListType` and `gml:MeasureOrNilReasonListType` provide for lists of quantities. The schema definitions are as follows:

```
<complexType name="MeasureListType">
  <simpleContent>
    <extension base="gml:doubleList">
      <attribute name="uom" type="gml:UomIdentifier" use="required"/>
    </extension>
  </simpleContent>
</complexType>

<complexType name="MeasureOrNilReasonListType">
  <simpleContent>
    <extension base="gml:doubleOrNilReasonList">
      <attribute name="uom" type="gml:UomIdentifier" use="required"/>
    </extension>
  </simpleContent>
</complexType>
```

EXAMPLE An application schema may contain element declarations using these types

```
<element name = “heights” type = “gml:MeasureListType”/>

<element name = “weights” type = “gml:MeasureOrNilReasonListType”/>
```

so corresponding elements might appear in an instance document as follows:

```
<heights uom=”m”>1.76 1.85 1.56 1.98</heights>

<weights uom=”kg”>67.0 73.4 withheld 85.1</weights>
```

In both examples all of the values in the list are described using the same scale.

In the second example a value describing the reason for a nil value appears where a measure is normally expected, but the value may be absent for some reason.

9 GML schema — Features

9.1 General concepts

A GML feature is a feature encoded using GML.

EXAMPLE A road, a river, a person, a vehicle, an administrative area, an event, etc.

The feature schema provides a framework for the creation of GML features and feature collections.

NOTE The feature schema document feature.xsd (see Annex C) is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:feature:3.2.1

9.2 Relationship with ISO 19109

The GML feature model follows the principles specified in ISO 19109:2005, Clause 7. It provides a conformant, partial implementation of the ISO 19109 General Feature Model. The relationship is discussed in detail in D.2.6.

NOTE The GML feature model also draws the feature collection concept from OGC Abstract Specification Topics 5 and 10.

9.3 Features

9.3.1 AbstractFeatureType

The basic feature model is given by the `gml:AbstractFeatureType`, defined in the schema as follows:

```
<complexType name="AbstractFeatureType" abstract="true">
  <complexContent>
    <extension base="gml:AbstractGMLType">
      <sequence>
        <element ref="gml:boundedBy" minOccurs="0"/>
        <element ref="gml:location" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The content model for `gml:AbstractFeatureType` adds two specific properties suitable for geographic features to the content model defined in `gml:AbstractGMLType`.

The value of the `gml:boundedBy` property describes an envelope that encloses the entire feature instance, and is primarily useful for supporting rapid searching for features that occur in a particular location.

The value of the `gml:location` property describes the extent, position or relative location of the feature. gml:location is deprecated as part of the standard content model of gml:AbstractFeatureType.

9.3.2 AbstractFeature

The element `gml:AbstractFeature` is declared as follows:

```
<element name="AbstractFeature" type="gml:AbstractFeatureType" abstract="true"
substitutionGroup="gml:AbstractGML"/>
```

This abstract element serves as the head of a substitution group which may contain any elements whose content model is derived from `gml:AbstractFeatureType`. This may be used as a variable in the construction of content models.

`gml:AbstractFeature` may be thought of as “anything that is a GML feature” and may be used to define variables or templates in which the value of a GML property is “any feature”. This occurs in particular in a GML feature collection (see 9.9) where the feature member properties contain one or multiple copies of `gml:AbstractFeature` respectively.

9.4 Standard feature properties

9.4.1 boundedBy, BoundingShapeType, EnvelopeWithTimePeriod, EnvelopeWithTimePeriodType

This property describes the minimum bounding box or rectangle that encloses the entire feature. Its content model is as follows:

```
<element name="boundedBy" type="gml:BoundingShapeType" nillable="true"/>

<complexType name="BoundingShapeType">
	<sequence>
		<choice>
			<element ref="gml:Envelope"/>
			<element ref="gml:Null"/>
		</choice>
	</sequence>
	<attribute name="nilReason" type="gml:NilReasonType"/>
</complexType>
```

The `gml:Envelope` element is defined in 10.1.4.6.

A nil value shall be encoded as described in 8.2.3.2. The attribute nilReason may be used in such cases to specify the reason for the nil value.

The value of `gml:Null`, used in previous versions of GML to encode that an extent is not applicable or not available for some reason, has been deprecated.

NOTE 1 Since an envelope is defined simply by the positions of two diagonally opposing corners, the exact footprint of an envelope depends on the coordinate reference system used. If the feature being described has zero extent, then the two corners will coincide and the envelope has zero size. The `gml:boundedBy` property is provided by a data supplier for convenience. The value of the envelope is usually computable by the data consumer from the spatio-temporal properties of a feature. As for all properties, it is the responsibility of the data provider to ensure that the value is correct.

For envelopes that include a temporal extent, `gml:EnvelopeWithTimePeriod` is provided, defined as follows:

```
<element name="EnvelopeWithTimePeriod" type="gml:EnvelopeWithTimePeriodType"
substitutionGroup="gml:Envelope"/>

<complexType name="EnvelopeWithTimePeriodType">
	<complexContent>
		<extension base="gml:EnvelopeType">
			<sequence>
				<element name="beginPosition" type="gml:TimePositionType"/>
				<element name="endPosition" type="gml:TimePositionType"/>
			</sequence>
			<attribute name="frame" type="anyURI" default="#ISO-8601"/>
		</extension>
	</complexContent>
</complexType>
```

This adds two time position properties, `gml:beginPosition` and `gml:endPosition`, which describe the extent of a time-envelope.

Since `gml:EnvelopeWithTimePeriod` is assigned to the substitution group headed by `gml:Envelope`, it may be used whenever `gml:Envelope` is valid.

NOTE 2 In common with all geometry elements derived from `gml:AbstractGeometryType` (see 10.1.3.1), the coordinate reference system used for the positions defining the `gml:Envelope` may be indicated using the optional XML attribute `srsName`. If the coordinate reference system being used includes a time axis, then `gml:Envelope` may be used directly to describe a spatio-temporal extent.

9.4.2 locationName, locationReference

The `gml:locationName` property element is a convenience property where the text value describes the location of the feature. It is defined as follows:

```
<element name="locationName" type="gml:CodeType"/>
```

If the location names are selected from a controlled list, then the list shall be identified in the `codeSpace` attribute.

The `gml:locationReference` property element is a convenience property where the text value referenced by the `xlink:href` attribute describes the location of the feature. It is defined as follows:

```
<element name="locationReference" type="gml:ReferenceType"/>
```

EXAMPLE The following instances illustrate the different ways that a `gml:locationName` or `gml:locationReference` may appear in a data instance.

Location given using a name from a controlled source:

```
<Feature>
  <gml:locationName codeSpace="http://www.icsm.gov.au/icsm/cgna/index.html">Leederville</gml:locationName>
</Feature>
```

Location given using a text string:

```
<Feature>
  <gml:locationName>Nigel Foster's town of residence</gml:locationName>
</Feature>
```

Location given by another service:

```
<Feature>
  <gml:locationReference
      xlink:href="http://www.ga.gov.au/bin/gazm01?placename=leederville&placetype=R&state=WA+"/>
</Feature>
```

9.4.3 FeaturePropertyType, FeatureArrayPropertyType

A particular class of properties defines associations between features. These use the `gml:AssociationRoleType` pattern as follows:

```
<complexType name="FeaturePropertyType">
    <sequence minOccurs="0">
        <element ref="gml:AbstractFeature"/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

At times it is useful to define a property containing an array of other features. This is done using a feature array property type as defined by the following content model:

```
<complexType name="FeatureArrayPropertyType">
  <sequence minOccurs="0" maxOccurs="unbounded">
```

```
    <element ref="gml:AbstractFeature" />
  </sequence>
</complexType>
```

9.5 Geometry properties

Application-specific names shall be chosen for geometry properties in GML application schemas. The names of the properties should be chosen to express the semantics of the value. Using application specific names is the preferred method for names of properties including geometry properties.

There are no inherent restrictions in the type of geometry property a feature type may have as long as the property value is a geometry object substitutable for `gml:AbstractGeometry`.

EXAMPLE 1 A `RadioTower` feature type could have a *location* that returns a point geometry to identify its location through a representative point, and have another geometry property called *floorSpace* that returns a surface geometry describing its physical structure, and have yet a third geometry property called *serviceArea* that returns a surface geometry describing the area in which its transmissions can be received reliably.

```
<complexType name="RadioTowerType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element name="location" type="gml:PointPropertyType"/>
        <element name="floorSpace" type="gml:SurfacePropertyType"/>
        <element name="serviceArea" type="gml:SurfacePropertyType"/>
        <!-- … -->
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The GML schema includes predefined property types that may be used as types of geometry property element.

Table 4 — Predefined geometry property types

XML Schema property type	Associated geometry object types (element names)
PointPropertyType	Point
CurvePropertyType	AbstractCurve LineString Curve OrientableCurve CompositeCurve
SurfacePropertyType	AbstractSurface Polygon Surface OrientableSurface CompositeSurface
SolidPropertyType	AbstractSolid Solid CompositeSolid
MultiPointPropertyType	MultiPoint
MultiCurvePropertyType	MultiCurve
MultiSurfacePropertyType	MultiSurface
MultiSolidPropertyType	MultiSolid
MultiGeometryPropertyType	MultiGeometry
PointArrayPropertyType	Point(s)
CurveArrayPropertyType	AbstractCurve(s) LineString(s) Curve(s) OrientableCurve(s) CompositeCurve(s)
SurfaceArrayPropertyType	AbstractSurface(s) Polygon(s) Surface(s) OrientableSurface(s) CompositeSurface(s)
SolidArrayPropertyType	AbstractSolid(s) Solid(s) CompositeSolid(s)

9.6 Topology properties

Like with geometry properties, application-specific names shall be chosen for topology properties in GML application schemas. The names of the properties should be chosen to express the semantics of the value.

EXAMPLE A `StatisticalArea` feature type could have one or more *boundary* properties that return a TopoCurve to represent the boundary of the Statistical Area, and one or more *surface* properties that return a TopoSurface to represent the Statistical Area itself.

```
<complexType name="StatisticalAreaType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element name="boundary" type="gml:TopoCurvePropertyType" maxOccurs="unbounded"/>
        <element name="surface" type="gml:TopoSurfacePropertyType" maxOccurs="unbounded"/>
        <!-- … -->
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The GML schema includes predefined property types that may be used as types of topology property element. The first four of these properties express direction, whereas the others do not.

Table 5 — Predefined formal topology property types

XML Schema property type	Associated topology object types (element names)
DirectedNodePropertyType	Node
DirectedEdgePropertyType	Edge
DirectedFacePropertyType	Face
DirectedTopoSolidPropertyType	TopoSolid
TopoPointPropertyType	TopoPoint
TopoCurvePropertyType	TopoCurve
TopoSurfacePropertyType	TopoSurface
TopoVolumePropertyType	TopoVolume
TopoComplexPropertyType	TopoComplex

9.7 Temporal properties

Like for geometry and topology properties, the definition of temporal property elements is in the responsibility of the application schema designer.

EXAMPLE A feature type *Building* may have a *constructionTime* property whose XML type is "gml:TimePeriodPropertyType", a *completionTime* property whose XML type is "gml:TimeInstantPropertyType", and an *age* whose XML type is "duration" or "gml:TimeIntervalLengthType".

```
<complexType name="BuildingType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
```

```
        <element name="constructionTime" type="gml:TimePeriodPropertyType"/>
        <element name="completionTime" type="gml:TimeInstantPropertyType"/>
        <element name="age" type="gml:TimeIntervalLengthType"/>
        <!-- … -->
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The types shown in Table 6 are provided for direct use in declaring property elements.

Table 6 — Predefined formal temporal property types

XML Schema property type	Associated temporal object types (element names)
TimePrimitivePropertyType	AbstractTimePrimitive AbstractTimeGeometricPrimitive TimeInstant TimePeriod AbstractTimeTopologyPrimitive TimeEdge TimeNode
TimeGeometricPrimitivePropertyType	AbstractTimeGeometricPrimitive TimeInstant TimePeriod
TimeInstantPropertyType	TimeInstant
TimePeriodPropertyType	TimePeriod
TimeTopologyPrimitivePropertyType	AbstractTimeTopologyPrimitive TimeEdge TimeNode
TimeEdgePropertyType	TimeEdge
TimeNodePropertyType	TimeNode
TimeTopologyComplexPropertyType	TimeTopologyComplex
TimeOrdinalEraPropertyType	TimeOrdinalEra
TimeCalendarPropertyType	TimeCalendar
TimeCalendarEraPropertyType	TimeCalendarEra
TimeClockPropertyType	TimeClock
TimePositionType	- (simple type)
xsd:duration	- (simple type)
TimeIntervalLengthType	- (simple type)

The temporal property types listed above provide a relatively comprehensive set of components for associating temporal information with features and other objects.

9.8 Defining application-specific feature types

All specific feature types defined in application schemas shall be implemented as global XML elements whose content model (XML Schema types) are derived from `gml:AbstractFeatureType`, and thus all GML features inherit the optional `gml:boundedBy` property, as well as the standard `gml:identifier`, `gml:description`, `gml:descriptionReference` and `gml:name` properties inherited in turn from `gml:AbstractGMLType`, unless any property is suppressed in a derivation by restriction. `gml:AbstractFeatureType` also inherits `gml:id` from `gml:AbstractGMLType` and this is the preferred means of supporting database identifiers in GML. Features should carry a `gml:id` attribute.

NOTE 1 The deprecated properties have been omitted in this list of inherited properties.

NOTE 2 Every feature accessible via an OGC Web Feature Service will always carry a persistent `gml:id` attribute.

This type derivation requirement means that general purpose software designed to process arbitrary GML data shall be able to traverse the XML Schema derivation tree in order to determine whether or not a given element in the data stream is a GML feature.

A GML feature has a set of properties, where the specific set of properties defines the feature type. Properties have simple values, using XML Schema simple content types, or properties may have complex values, in which case they should be declared using the patterns described in 7.2.3.

In the application schema defining a feature there shall be a global element declared whose name is the semantic type of the feature in the domain of discourse. The global element shall be made a member of the `gml:AbstractFeature` substitution group (directly or indirectly).

```
<element name="<<featureName>>"  type = “<<contentModel >>” substitutionGroup=”gml:AbstractFeature” />
```

The content model of the feature may be a named or anonymous complex type.

9.9 Feature collections

9.9.1 GML feature collections

A GML feature collection is a collection of GML feature instances.

A GML feature collection is any GML feature with a property element in its content model whose content model is derived by extension from `gml:AbstractFeatureMemberType` (see 9.9.2).

In addition, the complex type describing the content model of the GML feature collection may also include a reference to the attribute group `gml:AggregationAttributeGroup` to provide additional information about the semantics of the object collection as specified in 7.2.5.1.

EXAMPLE The following schema components model a simple collection of arbitrary features; the collection is called MyFeatures:

```
<element name=“MyFeatures” type=“ex:MyFeaturesType” substitutionGroup="gml:AbstractFeature"/>

<complexType name="MyFeaturesType">
	<complexContent>
		<extension base="gml:AbstractFeatureType">
			<sequence>
				<element name="myMember" type="ex:MyFeaturesMemberType"
					minOccurs="0" maxOccurs="unbounded"/>
			</sequence>
			<attributeGroup ref="gml:AggregationAttributeGroup"/>
		</extension>
```



```
        </complexContent>
</complexType>


<complexType name="MyFeaturesMemberType">
        <complexContent>
                <extension base="gml:AbstractFeatureMemberType">
                        <sequence minOccurs="0">
                                <element ref="gml:AbstractFeature"/>
                        </sequence>
                        <attributeGroup ref="gml:AssociationAttributeGroup"/>
                </extension>
        </complexContent>
</complexType>
```

An instance example encoding a collection with set semantics where the bounding envelope is provided, too:

```
<MyFeatures aggregationType=”set”>
        <gml:boundedBy>
                <gml:Envelope srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
                        <gml:lowerCorner>50.23 9.23</gml:lowerCorner>
                        <gml:upperCorner>50.31 9.27</gml:upperCorner>
                </gml:Envelope>
        </gml:boundedBy>
        <myMember>
                <MyFeature gml:id=”f1”/>
        </myMember>
        <myMember>
                <MyFeature gml:id=”f2”/>
        </myMember>
        <myMember xlink:href=”#f3”/>
</MyFeatures>
```

EXAMPLE 2 Often, the feature collection will contain instances of a specific type. In the example below, the feature collection is a road that consists of road segments.

```
<element name=“Road” type=“ex:RoadType” substitutionGroup="gml:AbstractFeature"/>

<complexType name="RoadType">
        <complexContent>
                <extension base="gml:AbstractFeatureType">
                        <sequence>
                                <element name="segment" type="ex:RoadMemberType" minOccurs="0" maxOccurs="unbounded"/>
                        </sequence>
                        <attributeGroup ref="gml:AggregationAttributeGroup"/>
                </extension>
        </complexContent>
</complexType>

<complexType name="RoadMemberType">
        <complexContent>
                <extension base="gml:AbstractFeatureMemberType">
                        <sequence minOccurs="0">
                                <element ref="ex:RoadSegments"/>
                        </sequence>
                        <attributeGroup ref="gml:AssociationAttributeGroup"/>
                </extension>
        </complexContent>
</complexType>
```

An example instance fragment encoding a ordered collection of road segments is shown below:

<Road gml:id="r1" aggregationType=”sequence”>
    <segment>
        <RoadSegment gml:id=”s1”/>
    </segment>
    <segment xlink:href="#s8"/>
    <segment>
        <RoadSegment gml:id=”s4”/>
    </segment>
</Road>

9.9.2 AbstractFeatureMemberType and derived property types

To create a collection of GML features, a property type shall be derived by extension from `gml:AbstractFeatureMemberType`.

```
	<complexType name="AbstractFeatureMemberType" abstract="true">
		<sequence/>
		<attributeGroup ref="gml:OwnershipAttributeGroup"/>
	</complexType>
```

The derived property type shall follow one of the patterns specified in 7.2.3 and may set the multiplicity of the objects in the collection as required for its intended use.

By default, this abstract property type does not imply any ownership of the features in the collection. The `owns` attribute of `gml:OwnershipAttributeGroup` may be used on a property element instance to assert ownership of a feature in the collection. A collection shall not own a feature already owned by another object.

9.10 Spatial reference system used in a feature or feature collection

The value of the `gml:boundedBy` property for a feature or feature collection is usually a `gml:Envelope`. In common with all geometry elements derived from `gml:AbstractGeometryType` (see 10.1.3.1), the coordinate reference system used for the positions defining the `gml:Envelope` may be indicated using the optional XML attribute `srsName`.

For convenience in constructing feature and feature collection instances, the value of the `srsName` attribute on the `gml:Envelope` which is the value of the `gml:boundedBy` property of the feature shall be inherited by all directly expressed geometries in all properties of the feature or members of the collection, unless overruled by the presence of a local `srsName`. Thus it is not necessary for a geometry to carry a `srsName` attribute, if it uses the same coordinate reference system as given on the `gml:boundedBy` property of its parent feature. Inheritance of the coordinate reference system continues to any depth of nesting, but if overruled by a local srsName declaration, then the new coordinate reference system is inherited by all its children in turn.

Notwithstanding this rule, all the geometries used in a feature or feature collection may carry srsName attributes, in order to indicate the reference system used locally, even if they are the same as the parent.

10 GML schema — Geometric primitives

10.1 General concepts

10.1.1 Overview

NOTE 1 The geometry model of GML complies with ISO 19107. The underlying concepts of the types and elements of the GML geometry model are discussed in this document in more detail.

This clause describes the schema components for geometric primitives as specified by GML.

NOTE 2 The corresponding geometry schema documents, geometryBasic0d1d.xsd, geometryBasic2d.xsd and geometryPrimitives.xsd (see Annex C), are identified by the following location-independent names (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:geometryBasic0d1d:3.2.1

urn:x-ogc:specification:gml:schema-xsd:geometryBasic2d:3.2.1

urn:x-ogc:specification:gml:schema-xsd:geometryPrimitives:3.2.1

Any geometry element that inherits the semantics of `gml:AbstractGeometryType` may be viewed as a set of direct positions.

All of the classes derived from `gml:AbstractGeometryType` inherit an optional association to a coordinate reference system. All direct positions shall directly or indirectly be associated with a coordinate reference system. When geometry elements are aggregated in another geometry element (such as a `gml:MultiGeometry` or `gml:GeometricComplex`), which already has a coordinate reference system specified, then these elements are assumed to be in that same coordinate reference system unless otherwise specified.

The geometry model distinguishes geometric primitives, aggregates and complexes.

Geometric primitives, i.e. instances of a subtype of `gml:AbstractGeometricPrimitiveType`, will be open, that is, they will not contain their boundary points; curves will not contain their end points, surfaces will not contain their boundary curves, and solids will not contain their bounding surfaces.

10.1.2 Relationship with ISO 19107

The spatial geometry components of the GML schema specified in Clauses 10 and 11 provide a conformant, partial implementation of the ISO 19107 spatial schema (geometry). The relationship is discussed in detail in D.2.3.

The ISO 19107 geometry types implemented in GML are specified in ISO 19107; some additional constraints are specified in ISO 19107 for these types, which are also constraints on the spatial geometry components of the GML schema.

In addition, GML specifies complementary spatial geometry schema components as described in D.3.5 to D.3.8.

10.1.3 Abstract geometry

10.1.3.1 AbstractGeometryType

```
<complexType name="AbstractGeometryType" abstract="true">
    <complexContent>
        <extension base="gml:AbstractGMLType">
            <attributeGroup ref="gml:SRSReferenceGroup"/>
        </extension>
    </complexContent>
</complexType>
```

All geometry elements are derived directly or indirectly from this abstract supertype. A geometry element may have an identifying attribute (`gml:id`), may have one or more names (elements `gml:identifier` and gml:name) and a description (elements gml:description and gml:descriptionReference)[2]. It may be associated with a spatial reference system (attribute group gml:SRSReferenceGroup).

The following rules shall be adhered to:

- Every geometry type shall derive from this abstract type.
- Every geometry element (i.e. an element of a geometry type) shall be directly or indirectly in the substitution group of AbstractGeometry.

10.1.3.2 SRSReferenceGroup

```
<attributeGroup name="SRSReferenceGroup">
	<attribute name="srsName" type="anyURI" />
	<attribute name="srsDimension" type="positiveInteger" />
	<attributeGroup ref="gml:SRSInformationGroup"/>
</attributeGroup>
```

The attribute group gml:SRSReferenceGroup is an optional reference to the CRS used by this geometry, with optional additional information to simplify the processing of the coordinates when a more complete definition of the CRS is not needed.

In general the attribute srsName points to a CRS instance of gml:AbstractCoordinateReferenceSystem (see 12.2.3). For well-known references it is not required that the CRS description exists at the location the URI points to.

If no srsName attribute is given, the CRS shall be specified as part of the larger context this geometry element is part of.

EXAMPLE A geometric aggregate or a feature collection are typical “larger contexts”.

NOTE The name “srsName” has been chosen deliberately. In the current version of GML “crsName” would be more appropriate, however, in future versions other types of spatial reference system, i.e. those using geographic identifiers, may be supported by GML, too.

The optional attribute srsDimension is the number of coordinate values in a position. This dimension is derived from the coordinate reference system. When the srsName attribute is omitted, this attribute shall be omitted.

10.1.3.3 SRSInformationGroup

```
<attributeGroup name="SRSInformationGroup">
	<attribute name="axisLabels" type="gml:NCNameList" />
	<attribute name="uomLabels" type="gml:NCNameList" />
</attributeGroup>
```

The attributes uomLabels and axisLabels, defined in the gml:SRSInformationGroup attribute group, are optional additional and redundant information for a CRS to simplify the processing of the coordinate values when a more complete definition of the CRS is not needed. This information shall be the same as included in the complete definition of the CRS, referenced by the srsName attribute. When the srsName attribute is included, either both or neither of the axisLabels and uomLabels attributes shall be included. When the srsName attribute is omitted, both of these attributes shall be omitted.

---

2) Deprecated properties have been omitted from this list. Nevertheless, they are still valid content.

The attribute axisLabels is an ordered list of labels for all the axes of this CRS. The gml:axisAbbrev value should be used for these axis labels, after spaces and forbidden characters are removed. When the srsName attribute is included, this attribute is optional. When the srsName attribute is omitted, this attribute shall also be omitted.

The attribute uomLabels is an ordered list of unit of measure (uom) labels for all the axes of this CRS. The value of the string in the gml:catalogSymbol should be used for this uom labels, after spaces and forbidden characters are removed. When the axisLabels attribute is included, this attribute shall also be included. When the axisLabels attribute is omitted, this attribute shall also be omitted.

10.1.3.4 AbstractGeometry

```
<element name="AbstractGeometry" type="gml:AbstractGeometryType" abstract="true"
 substitutionGroup="gml:AbstractGML" />
```

The gml:AbstractGeometry element is the abstract head of the substitution group for all geometry elements of GML. This includes predefined and user-defined geometry elements. Any geometry element shall be a direct or indirect extension/restriction of gml:AbstractGeometryType and shall be directly or indirectly in the substitution group of gml:AbstractGeometry.

D.2.3.2 specifies the implementation of ISO 19107 GM_Object by this GML object.

10.1.3.5 GeometryPropertyType

```
<complexType name="GeometryPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractGeometry"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

A geometric property may either be any geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same or another document). Note that either the reference or the contained element shall be given, but not both or none, see 7.2.3.

If a feature has a property that takes a geometry element as its value, this is called a geometry property. A generic type for such a geometry property is gml:GeometryPropertyType which follows the general rules described in 7.2.3.

10.1.3.6 GeometryArrayPropertyType

```
<complexType name="GeometryArrayPropertyType">
	<sequence minOccurs="0" maxOccurs="unbounded">
		<element ref="gml:AbstractGeometry" />
	</sequence>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

If a feature has a property which takes an array of geometry elements as its value, this is called a geometry array property. A generic type for such a geometry property is gml:GeometryArrayPropertyType which follows the general rules described in 7.2.3.

The elements are always contained inline in the array property. Referencing geometry elements or arrays of geometry elements via XLinks is not supported.

EXAMPLE All elements in a `gml:GeometryArrayPropertyType` are of the type `gml:AbstractGeometryType` (including types derived from this abstract base type) as long as the element is directly or indirectly substitutable for `gml:AbstractGeometry`.

10.1.4 Coordinate geometry, vectors and envelopes

10.1.4.1 DirectPositionType, pos

```
<complexType name="DirectPositionType">
	<simpleContent>
		<extension base="gml:doubleList">
			<attributeGroup ref="gml:SRSReferenceGroup"/>
		</extension>
	</simpleContent>
</complexType>

<element name="pos" type="gml:DirectPositionType"/>

<sch:pattern>
	<sch:rule context="gml:pos">
		<sch:assert test="not(@srsDimension) or @srsName">The presence of a dimension attribute implies the
presence of the srsName attribute.</sch:assert>
		<sch:assert test="not(@axisLabels) or @srsName">The presence of an axisLabels attribute implies the
presence of the srsName attribute.</sch:assert>
		<sch:assert test="not(@uomLabels) or @srsName">The presence of an uomLabels attribute implies the
presence of the srsName attribute.</sch:assert>
		<sch:assert test="(not(@uomLabels) and not(@axisLabels)) or (@uomLabels and @axisLabels)">The
presence of an uomLabels attribute implies the presence of the axisLabels attribute and vice versa.</sch:assert>
	</sch:rule>
</sch:pattern>
```

Direct position instances hold the coordinates for a position within some coordinate reference system (CRS). Since direct positions, as data types, will often be included in larger objects (such as geometry elements) that have references to CRS, the `srsName` attribute will in general be missing, if this particular direct position is included in a larger element with such a reference to a CRS. In this case, the CRS is implicitly assumed to take on the value of the containing object's CRS.

The attribute group `gml:SRSReferenceGroup` is described in 10.1.3.2. If no `srsName` attribute is given, the CRS shall be specified as part of the larger context this geometry element is part of, typically a geometric object like a point, curve, etc.

NOTE It is expected that the attribute will be specified at the direct position level only in rare cases.

D.2.3.4 specifies the implementation of ISO 19107 DirectPosition by these schema components.

10.1.4.2 DirectPositionListType, posList

```
<complexType name="DirectPositionListType">
	<simpleContent>
		<extension base="gml:doubleList">
			<attributeGroup ref="gml:SRSReferenceGroup"/>
			<attribute name="count" type="positiveInteger" />
		</extension>
	</simpleContent>
</complexType>

<element name="posList" type="gml:DirectPositionListType" />
```


gml:posList instances (and other instances with the content model specified by DirectPositionListType) hold the coordinates for a sequence of direct positions within the same coordinate reference system (CRS).

The attribute group “SRSReferenceGroup” is described in 10.1.3.2. If no srsName attribute is given, the CRS shall be specified as part of the larger context this geometry element is part of, typically a geometric object like a point, curve, etc.

NOTE It is expected that the attribute srsName will be specified at the direct position level only in rare cases.

The optional attribute count specifies the number of direct positions in the list. If the attribute count is present then the attribute srsDimension shall be present, too.

The number of entries in the list is equal to the product of the dimensionality of the coordinate reference system (i.e. it is a derived value of the coordinate reference system definition) and the number of direct positions.

D.2.3.4 specifies the implementation of ISO 19107 GM_PointArray using direct positions only by these schema components.

10.1.4.3 geometricPositionGroup

```
<group name="geometricPositionGroup">
	<choice>
		<element ref="gml:pos"/>
		<element ref="gml:pointProperty"/>
	</choice>
</group>
```

GML supports two different ways to specify a geometric position: either by a direct position (a data type) or a point (a geometric object).

gml:pos elements are positions that are “owned” by the geometric primitive encapsulating this geometric position.

gml:pointProperty elements contain a point that may be referenced from other geometry elements or reference another point defined elsewhere (reuse of existing points).

D.2.3.4 specifies the implementation of ISO 19107 GM_Position by this choice group.

10.1.4.4 geometricPositionListGroup

```
<group name="geometricPositionListGroup">
	<choice>
		<element ref="gml:posList"/>
		<group ref="gml:geometricPositionGroup" maxOccurs="unbounded"/>
	</choice>
</group>
```

GML supports two different ways to specify a list of geometric positions: either by a sequence of geometric positions (by reusing the group definition) or a sequence of direct positions (element gml:posList).

The gml:posList element allows for a compact way to specify the coordinates of the positions, if all positions are represented in the same coordinate reference system.

D.2.3.4 specifies the implementation of ISO 19107 GM_PointArray by this choice group.

NOTE The definition of this group may be used as a pattern in the definition of geometric primitives instead of using this group definition directly. The main change will typically be a change in the multiplicity of the referenced group. A LineString, for example, requires at least two positions.

Also, to support deprecated elements, i.e. `gml:coordinates` (superceded by `gml:posList`) and `gml:pointRep` (superseded by `gml:pointProperty`), the current encodings of point arrays in GML, e.g. in curve segments, uses this group as a pattern and adds the deprecated elements.

10.1.4.5 VectorType, Vector

```
<complexType name="VectorType">
  <simpleContent>
    <restriction base="gml:DirectPositionType"/>
  </simpleContent>
</complexType>

<element name="vector" type="gml:VectorType" />
```

`gml:vector` implements ISO/TS 19103 Vector (see D.2.3.2 and ISO/TS 19103:2005, 6.5.2.6).

For some applications the components of the position may be adjusted to yield a unit vector.

NOTE This definition allows VectorType to be used elsewhere when appropriate — e.g. for offsetVector in grids.xsd, and vector to be used directly when appropriate — e.g. in DirectionVector in direction.xsd.

10.1.4.6 EnvelopeType, Envelope

```
<complexType name="EnvelopeType">
    <choice>
        <sequence>
            <element name="lowerCorner" type="gml:DirectPositionType"/>
            <element name="upperCorner" type="gml:DirectPositionType"/>
        </sequence>
        <element ref="gml:pos" minOccurs="2" maxOccurs="2"/>
        <element ref="gml:coordinates"/>
    </choice>
    <attributeGroup ref="gml:SRSReferenceGroup"/>
</complexType>

<element name="Envelope" type="gml:EnvelopeType" substitutionGroup="gml:AbstractObject"/>
```

`gml:Envelope` implements ISO 19107 GM_Envelope (see D.2.3.4 and ISO 19107:2003, 6.4.3).

Envelope defines an extent using a pair of positions defining opposite corners in arbitrary dimensions. The first direct position is the "lower corner" (a coordinate position consisting of all the minimal ordinates for each dimension for all points within the envelope), the second one the "upper corner" (a coordinate position consisting of all the maximal ordinates for each dimension for all points within the envelope).

The use of the properties "coordinates" and "pos" in Envelope has been deprecated. The explicitly named properties "lowerCorner" and "upperCorner" shall be used instead.

NOTE Regardless of dimension, an envelope can be represented without ambiguity as two direct positions (coordinate points) provided the ordering of those points adheres to the specified rule. Envelope is often referred to as a minimum bounding box or rectangle. However, this Envelope will not always specify the MINIMUM rectangular bounding region, if the referenced CRS is a Geodetic CRS, or uses an Ellipsoidal, Spherical, Polar, or Cylindrical coordinate system, as those terms are specified in 12.4. Specifically, this Envelope will not specify the MINIMUM rectangular bounding region of a geometry whose set of points span the value discontinuity in an angular coordinate axis. Such axes include the Longitude and Latitude of Ellipsoidal and Spherical coordinate systems. That geometry could lie within a small region on the surface of the ellipsoid or sphere, or could extend completely around the ellipsoid or sphere.

10.2 Abstract geometric primitives

10.2.1 AbstractGeometricPrimitiveType, AbstractGeometricPrimitive

```
<complexType name="AbstractGeometricPrimitiveType" abstract="true">
	<complexContent>
		<extension base="gml:AbstractGeometryType" />
	</complexContent>
</complexType>

<element name="AbstractGeometricPrimitive" type="gml:AbstractGeometricPrimitiveType" abstract="true"
	substitutionGroup="gml:AbstractGeometry" />
```

`gml:AbstractGeometricPrimitiveType` is the abstract root type of the geometric primitives. A geometric primitive is a geometric object that is not decomposed further into other primitives in the system. All primitives are oriented in the direction implied by the sequence of their coordinate tuples.

The `gml:AbstractGeometricPrimitive` element is the abstract head of the substitution group for all (pre- and user-defined) geometric primitives.

`gml:AbstractGeometricPrimitive` implements ISO 19107 GM_Primitive (see D.2.3.3 and ISO 19107:2003, 6.3.10).

10.2.2 GeometricPrimitivePropertyType

```
<complexType name="GeometricPrimitivePropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractGeometricPrimitive" />
	</sequence>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
	<attributeGroup ref="gml:AssociationAttributeGroup" />
</complexType>
```

A property that has a geometric primitive as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.

10.3 Geometric primitives (0-dimensional)

10.3.1 PointType, Point

```
<complexType name="PointType">
	<complexContent>
		<extension base="gml:AbstractGeometricPrimitiveType">
			<sequence>
				<choice>
					<element ref="gml:pos" />
					<element ref="gml:coordinates" />
				</choice>
			</sequence>
		</extension>
	</complexContent>
</complexType>

<element name="Point" type="gml:PointType" substitutionGroup="gml:AbstractGeometricPrimitive" />
```

A `gml:Point` is defined by a single coordinate tuple. The direct position of a point is specified by the `gml:pos` element which is of type `gml:DirectPositionType`.

`gml:Point` implements ISO 19107 GM_Point (see D.2.3.3 and ISO 19107:2003, 6.3.11).

The use of the element "coordinates" is deprecated. Use "pos" instead.

10.3.2 PointPropertyType, pointProperty

```
<complexType name="PointPropertyType">
      <sequence minOccurs="0">
            <element ref="gml:Point"/>
      </sequence>
      <attributeGroup ref="gml:AssociationAttributeGroup"/>
      <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>

<element name="pointProperty" type="gml:PointPropertyType" />
```

A property that has a point as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.

This property element either references a point via the XLink-attributes or contains the point element. pointProperty is the predefined property which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for `gml:Point`.

10.3.3 PointArrayPropertyType, pointArrayProperty

```
<complexType name="PointArrayPropertyType">
      <sequence minOccurs="0" maxOccurs="unbounded">
            <element ref="gml:Point" />
      </sequence>
      <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>

<element name="pointArrayProperty" type="gml:PointArrayPropertyType" />
```

`gml:PointArrayPropertyType` is a container for an array of points. The elements are always contained inline in the array property. Referencing geometry elements or arrays of geometry elements via XLinks is not supported.

This property element contains a list of point elements. pointArrayProperty is the predefined property which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for a list of points.

10.4 Geometric primitives (1-dimensional)

10.4.1 AbstractCurveType, AbstractCurve

```
<complexType name="AbstractCurveType" abstract="true">
      <complexContent>
            <extension base="gml:AbstractGeometricPrimitiveType"/>
      </complexContent>
</complexType>

<element name="AbstractCurve" type="gml:AbstractCurveType" abstract="true"
```

```
        substitutionGroup="gml:AbstractGeometricPrimitive" />
```

`gml:AbstractCurveType` is an abstraction of a curve to support the different levels of complexity. The curve may always be viewed as a geometric primitive, i.e. is continuous.

The `gml:AbstractCurve` element is the abstract head of the substitution group for all (continuous) curve elements.

10.4.2 CurvePropertyType, curveProperty

```
<complexType name="CurvePropertyType">
    <sequence minOccurs="0">
        <element ref="gml:AbstractCurve"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>

<element name="curveProperty" type="gml:CurvePropertyType" />
```

A property that has a curve as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.

This property element either references a curve via the XLink-attributes or contains the curve element. curveProperty is the predefined property which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for `gml:AbstractCurve`.

10.4.3 CurveArrayPropertyType, curveArrayProperty

```
<complexType name="CurveArrayPropertyType">
    <sequence minOccurs="0" maxOccurs="unbounded">
        <element ref="gml:AbstractCurve" />
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>

<element name="curveArrayProperty" type="gml:CurveArrayPropertyType" />
```

A container for an array of curves. The elements are always contained inline in the array property. Referencing geometry elements or arrays of geometry elements via XLinks is not supported.

This property element contains a list of curve elements. curveArrayProperty is the predefined property which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for a list of curves.

10.4.4 LineStringType, LineString

```
<complexType name="LineStringType">
    <complexContent>
        <extension base="gml:AbstractCurveType">
            <sequence>
                <choice>
                    <choice minOccurs="2" maxOccurs="unbounded">
                        <element ref="gml:pos"/>
                        <element ref="gml:pointProperty"/>
                        <element ref="gml:pointRep"/>
```

```
                    </choice>
                    <element ref="gml:posList"/>
                    <element ref="gml:coordinates"/>
                </choice>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<element name="LineString" type="gml:LineStringType" substitutionGroup="gml:AbstractCurve" />
```

A `gml:LineString` is a special curve that consists of a single segment with linear interpolation (see D.3.5). It is defined by two or more coordinate tuples, with linear interpolation between them.

The encoding of the control points follows the pattern described in 10.1.4.4. The number of direct positions in the list shall be at least two.

NOTE ISO 19107 GM_LineString is implemented by `gml:LineStringSegment`.

10.4.5 CurveType, Curve

```
<complexType name="CurveType">
    <complexContent>
        <extension base="gml:AbstractCurveType">
            <sequence>
                <element ref="gml:segments" />
            </sequence>
        </extension>
    </complexContent>
</complexType>

<element name="Curve" type="gml:CurveType" substitutionGroup="gml:AbstractCurve" />
```

`gml:Curve` implements ISO 19107 GM_Curve (see D.2.3.3 and ISO 19107:2003, 6.3.16).

A curve is a 1-dimensional primitive. Curves are continuous, connected, and have a measurable length in terms of the coordinate system.

A curve is composed of one or more curve segments. Each curve segment within a curve may be defined using a different interpolation method. The curve segments are connected to one another, with the end point of each segment except the last being the start point of the next segment in the segment list.

The orientation of the curve is positive.

The element `gml:segments` encapsulates the segments of the curve.

10.4.6 OrientableCurveType, OrientableCurve, baseCurve

```
<complexType name="OrientableCurveType">
    <complexContent>
        <extension base="gml:AbstractCurveType">
            <sequence>
                <element ref="gml:baseCurve" />
            </sequence>
            <attribute name="orientation" type="gml:SignType" default="+" />
        </extension>
    </complexContent>
</complexType>
```

```
<element name="baseCurve" type="gml:CurvePropertyType" />

<element name="OrientableCurve" type="gml:OrientableCurveType" substitutionGroup="gml:AbstractCurve" />
```

`gml:OrientableCurve` implements ISO 19107 GM_OrientableCurve (see D.2.3.3 and ISO 19107:2003, 6.3.14).

`gml:OrientableCurve` consists of a curve and an `orientation`. If the `orientation` is "+", then the `gml:OrientableCurve` is identical to the `gml:baseCurve`. If the `orientation` is "-", then the `gml:OrientableCurve` is related to another `gml:AbstractCurve` with a parameterization that reverses the sense of the curve traversal.

The property `gml:baseCurve` references or contains the base curve, i.e. it either references the base curve via the XLink-attributes or contains the curve element. A curve element is any element which is substitutable for `gml:AbstractCurve`. The base curve has positive orientation.

NOTE This definition allows for a nested structure, i.e. an `gml:OrientableCurve` may use another `gml:OrientableCurve` as its base curve.

10.4.7 Curve segments

10.4.7.1 AbstractCurveSegmentType, AbstractCurveSegment

```
<complexType name="AbstractCurveSegmentType" abstract="true">
    <attribute name="numDerivativesAtStart" type="integer" default="0" />
    <attribute name="numDerivativesAtEnd" type="integer" default="0" />
    <attribute name="numDerivativeInterior" type="integer" default="0" />
</complexType>

<element name="AbstractCurveSegment" type="gml:AbstractCurveSegmentType" abstract="true"
substitutionGroup="gml:AbstractObject" />
```

`gml:AbstractCurveSegment` implements ISO 19107 GM_CurveSegment (see D.2.3.3 and ISO 19107:2003, 6.4.9).

A curve segment defines a homogeneous segment of a curve.

The attributes `numDerivativesAtStart`, `numDerivativesAtEnd` and `numDerivativesInterior` specify the type of continuity as specified in ISO 19107:2003, 6.4.9.3.

The `gml:AbstractCurveSegment` element is the abstract head of the substitution group for all curve segment elements, i.e. continuous segments of the same interpolation mechanism.

The encoding of the control points in a curve segment shall follow the pattern described in 10.1.4.4.

All curve segments shall have an attribute `interpolation` with type `gml:CurveInterpolationType` specifying the curve interpolation mechanism used for this segment. This mechanism uses the control points and control parameters to determine the position of this curve segment.

10.4.7.2 CurveSegmentArrayPropertyType, segments

```
<complexType name="CurveSegmentArrayPropertyType">
    <sequence minOccurs="0" maxOccurs="unbounded">
        <element ref="gml:AbstractCurveSegment" />
    </sequence>
```

```
</complexType>
```

`gml:CurveSegmentArrayPropertyType` is a container for an array of curve segments.

```
<element name="segments" type="gml:CurveSegmentArrayPropertyType" />
```

This property element contains a list of curve segments. The order of the elements is significant and shall be preserved when processing the array.

10.4.7.3 CurveInterpolationType

```
<simpleType name="CurveInterpolationType">
    <restriction base="string">
        <enumeration value="linear" />
        <enumeration value="geodesic" />
        <enumeration value="circularArc3Points" />
        <enumeration value="circularArc2PointWithBulge" />
        <enumeration value="circularArcCenterPointWithRadius" />
        <enumeration value="elliptical" />
        <enumeration value="clothoid" />
        <enumeration value="conic" />
        <enumeration value="polynomialSpline" />
        <enumeration value="cubicSpline" />
        <enumeration value="rationalSpline" />
    </restriction>
</simpleType>
```

`gml:CurveInterpolationType` is a list of codes that may be used to identify the interpolation mechanisms specified by an application schema.

This type implements ISO 19107 GM_CurveInterpolation (see D.2.3.4 and ISO 19107:2003, 6.4.8).

10.4.7.4 LineStringSegmentType, LineStringSegment

```
<complexType name="LineStringSegmentType">
    <complexContent>
        <extension base="gml:AbstractCurveSegmentType">
            <sequence>
                <choice>
                    <choice minOccurs="2" maxOccurs="unbounded">
                        <element ref="gml:pos" />
                        <element ref="gml:pointProperty" />
                        <element ref="gml:pointRep"/>
                    </choice>
                    <element ref="gml:posList" />
                    <element ref="gml:coordinates" />
                </choice>
            </sequence>
            <attribute name="interpolation" type="gml:CurveInterpolationType" fixed="linear" />
        </extension>
    </complexContent>
</complexType>

<element name="LineStringSegment" type="gml:LineStringSegmentType"
    substitutionGroup="gml:AbstractCurveSegment" />
```

`gml:LineStringSegment` implements ISO 19107 GM_LineString (see D.2.3.4 and ISO 19107:2003, 6.4.10).

A `gml:LineStringSegment` is a curve segment that is defined by two or more control points including the start and end point, with linear interpolation between them.

The content model follows the general pattern for the encoding of curve segments (see 10.4.7).

10.4.7.5 ArcStringType, ArcString

```
<complexType name="ArcStringType">
    <complexContent>
        <extension base="gml:AbstractCurveSegmentType">
            <sequence>
                <choice>
                    <choice minOccurs="3" maxOccurs="unbounded">
                        <element ref="gml:pos" />
                        <element ref="gml:pointProperty" />
                        <element ref="gml:pointRep" />
                    </choice>
                    <element ref="gml:posList" />
                    <element ref="gml:coordinates" />
                </choice>
            </sequence>
            <attribute name="interpolation" type="gml:CurveInterpolationType" fixed="circularArc3Points" />
            <attribute name="numArc" type="integer" />
        </extension>
    </complexContent>
</complexType>

<element name="ArcString" type="gml:ArcStringType" substitutionGroup="gml:AbstractCurveSegment" />
```

`gml:ArcString` implements ISO 19107 GM_ArcString (see D.2.3.4 and ISO 19107:2003, 6.4.14).

A `gml:ArcString` is a curve segment that uses three-point circular arc interpolation ("circularArc3Points"). The number of arcs in the arc string may be explicitly stated in the attribute `numArc`. The number of control points in the arc string shall be 2 * numArc + 1.

The content model follows the general pattern for the encoding of curve segments (see 10.4.7).

10.4.7.6 ArcType, Arc

```
<complexType name="ArcType">
    <complexContent>
        <restriction base="gml:ArcStringType">
            <sequence>
                <choice>
                    <choice minOccurs="3" maxOccurs="3">
                        <element ref="gml:pos" />
                        <element ref="gml:pointProperty" />
                        <element ref="gml:pointRep" />
                    </choice>
                    <element ref="gml:posList" />
                    <element ref="gml:coordinates" />
                </choice>
            </sequence>
            <attribute name="numArc" type="integer" fixed="1" />
        </restriction>
    </complexContent>
</complexType>

<element name="Arc" type="gml:ArcType" substitutionGroup="gml:ArcString" />
```

`gml:Arc` implements ISO 19107 GM_Arc (see D.2.3.4 and ISO 19107:2003, 6.4.15).

An Arc is an arc string with only one arc unit, i.e. three control points including the start and end point. As arc is an arc string consisting of a single arc, the attribute “numArc” is fixed to "1".

10.4.7.7 CircleType, Circle

```
<complexType name="CircleType">
	<complexContent>
		<extension base="gml:ArcType" />
	</complexContent>
</complexType>

<element name="Circle" type="gml:CircleType" substitutionGroup="gml:Arc" />
```

`gml:Circle` implements ISO 19107 GM_Circle (see D.2.3.4 and ISO 19107:2003, 6.4.16).

A Circle is an arc whose ends coincide to form a simple closed loop. The three control points shall be distinct non-co-linear points for the circle to be unambiguously defined. The arc is simply extended past the third control point until the first control point is encountered.

10.4.7.8 ArcStringByBulgeType, ArcStringByBulge

```
<complexType name="ArcStringByBulgeType">
	<complexContent>
		<extension base="gml:AbstractCurveSegmentType">
			<sequence>
				<choice>
					<choice minOccurs="2" maxOccurs="unbounded">
						<element ref="gml:pos" />
						<element ref="gml:pointProperty" />
						<element ref="gml:pointRep" />
					</choice>
					<element ref="gml:posList" />
					<element ref="gml:coordinates" />
				</choice>
				<element name="bulge" type="double" maxOccurs="unbounded"/>
				<element name="normal" type="gml:VectorType" maxOccurs="unbounded" />
			</sequence>
			<attribute name="interpolation" type="gml:CurveInterpolationType"
				fixed="circularArc2PointWithBulge" />
			<attribute name="numArc" type="integer" />
		</extension>
	</complexContent>
</complexType>

<element name="ArcStringByBulge" type="gml:ArcStringByBulgeType"
	substitutionGroup="gml:AbstractCurveSegment" />
```

`gml:ArcStringByBuldge` implements ISO 19107 GM_ArcStringByBuldge (see D.2.3.4 and ISO 19107:2003, 6.4.17).

This variant of the arc computes the mid points of the arcs instead of storing the coordinates directly. The control point sequence consists of the start and end points of each arc plus the `gml:bulge` (see ISO 19107:2003, 6.4.17.2). The `gml:normal` is a vector normal (perpendicular) to the chord of the arc (see ISO 19107:2003, 6.4.17.4).

The `interpolation` is fixed as "circularArc2PointWithBulge".

The number of arcs in the arc string may be explicitly stated in the attribute `numArc`. The number of control points in the arc string shall be numArc + 1.

The content model follows the general pattern for the encoding of curve segments (see 10.4.7).

10.4.7.9 ArcByBulgeType, ArcByBulge

```
<complexType name="ArcByBulgeType">
    <complexContent>
        <restriction base="gml:ArcStringByBulgeType">
            <sequence>
                <choice>
                    <choice minOccurs="2" maxOccurs="2">
                        <element ref="gml:pos" />
                        <element ref="gml:pointProperty" />
                        <element ref="gml:pointRep" />
                    </choice>
                    <element ref="gml:posList" />
                    <element ref="gml:coordinates" />
                </choice>
                <element name="bulge" type="double" />
                <element name="normal" type="gml:VectorType"/>
            </sequence>
            <attribute name="numArc" type="integer" fixed="1" />
        </restriction>
    </complexContent>
</complexType>

<element name="ArcByBulge" type="gml:ArcByBulgeType" substitutionGroup="gml:ArcStringByBulge" />
```

`gml:ArcByBuldge` implements ISO 19107 GM_ArcByBuldge (see D.2.3.4 and ISO 19107:2003, 6.4.18).

An ArcByBulge is an arc string with only one arc unit, i.e. two control points, one bulge and one normal vector.

As arc is an arc string consisting of a single arc, the attribute “numArc” is fixed to "1".

10.4.7.10 ArcByCenterPointType, ArcByCenterPoint

```
<complexType name="ArcByCenterPointType">
    <complexContent>
        <extension base="gml:AbstractCurveSegmentType">
            <sequence>
                <choice>
                    <choice>
                        <element ref="gml:pos" />
                        <element ref="gml:pointProperty" />
                        <element ref="gml:pointRep" />
                    </choice>
                    <element ref="gml:posList" />
                    <element ref="gml:coordinates" />
                </choice>
                <element name="radius" type="gml:LengthType" />
                <element name="startAngle" type="gml:AngleType" minOccurs="0" />
                <element name="endAngle" type="gml:AngleType" minOccurs="0" />
            </sequence>
            <attribute name="interpolation" type="gml:CurveInterpolationType"
                fixed="circularArcCenterPointWithRadius" />
```



```
                <attribute name="numArc" type="integer" use="required" fixed="1" />
            </extension>
        </complexContent>
    </complexType>

    <element name="ArcByCenterPoint" type="gml:ArcByCenterPointType"
        substitutionGroup="gml:AbstractCurveSegment" />
```

This variant of the arc requires that the points on the arc shall be computed instead of storing the coordinates directly. The single control point is the center point of the arc plus the radius and the bearing at start and end. This representation can be used only in 2D.

The element `gml:radius` specifies the radius of the arc.

The element `gml:startAngle` specifies the bearing of the arc at the start.

The element `gml:endAngle` specifies the bearing of the arc at the end.

The `interpolation` is fixed as "circularArcCenterPointWithRadius".

Since this type describes always a single arc, the attribute “numArc” is fixed to "1".

The content model follows the general pattern for the encoding of curve segments (see 10.4.7).

10.4.7.11 CircleByCenterPointType, CircleByCenterPoint

```
    <complexType name="CircleByCenterPointType">
        <complexContent>
            <restriction base="gml:ArcByCenterPointType">
                <sequence>
                    <choice>
                        <choice>
                            <element ref="gml:pos" />
                            <element ref="gml:pointProperty" />
                            <element ref="gml:pointRep" />
                        </choice>
                        <element ref="gml:posList" />
                        <element ref="gml:coordinates" />
                    </choice>
                    <element name="radius" type="gml:LengthType" />
                </sequence>
            </restriction>
        </complexContent>
    </complexType>

    <element name="CircleByCenterPoint" type="gml:CircleByCenterPointType"
        substitutionGroup="gml:ArcByCenterPoint" />
```

A `gml:CircleByCenterPoint` is an `gml:ArcByCenterPoint` with identical start and end angle to form a full circle. Again, this representation can be used only in 2D.

10.4.7.12 CubicSplineType, CubicSpline

```
    <complexType name="CubicSplineType">
        <complexContent>
            <extension base="gml:AbstractCurveSegmentType">
                <sequence>
```



```
				<choice>
					<choice minOccurs="2" maxOccurs="unbounded">
						<element ref="gml:pos" />
						<element ref="gml:pointProperty" />
						<element ref="gml:pointRep" />
					</choice>
					<element ref="gml:posList" />
					<element ref="gml:coordinates" />
				</choice>
				<element name="vectorAtStart" type="gml:VectorType" />
				<element name="vectorAtEnd" type="gml:VectorType" />
			</sequence>
			<attribute name="interpolation" type="gml:CurveInterpolationType" fixed="cubicSpline" />
			<attribute name="degree" type="integer" fixed="3" />
		</extension>
	</complexContent>
</complexType>

<element name="CubicSpline" type="gml:CubicSplineType" substitutionGroup="gml:AbstractCurveSegment" />
```

`gml:CubicSpline` implements ISO 19107 GM_CubicSpline (see D.2.3.4 and ISO 19107:2003, 6.4.28).

The number of control points shall be at least three.

`gml:vectorAtStart` is the unit tangent vector at the start point of the spline. `gml:vectorAtEnd` is the unit tangent vector at the end point of the spline. Only the direction of the vectors shall be used to determine the shape of the cubic spline, not their length.

`interpolation` is fixed as "cubicSpline".

`degree` shall be the degree of the polynomial used for the interpolation in this spline. Therefore the degree for a cubic spline is fixed to "3".

The content model follows the general pattern for the encoding of curve segments (see 10.4.7).

10.4.7.13 BSplineType, BSpline

```
<complexType name="BSplineType">
	<complexContent>
		<extension base="gml:AbstractCurveSegmentType">
			<sequence>
				<choice>
					<choice minOccurs="0" maxOccurs="unbounded">
						<element ref="gml:pos" />
						<element ref="gml:pointProperty" />
						<element ref="gml:pointRep" />
					</choice>
					<element ref="gml:posList" />
					<element ref="gml:coordinates" />
				</choice>
				<element name="degree" type="nonNegativeInteger" />
				<element name="knot" type="gml:KnotPropertyType" minOccurs="2"
					maxOccurs="unbounded" />
			</sequence>
			<attribute name="interpolation" type="gml:CurveInterpolationType" default="polynomialSpline" />
			<attribute name="isPolynomial" type="boolean" />
			<attribute name="knotType" type="gml:KnotTypesType" />
		</extension>
	</complexContent>
```

```
</complexType>

<element name="BSpline" type="gml:BSplineType" substitutionGroup="gml:AbstractCurveSegment" />
```

`gml:BSpline` implements ISO 19107 GM_BSplineCurve (see D.2.3.4 and ISO 19107:2003, 6.4.30).

A B-Spline is a piecewise parametric polynomial or rational curve described in terms of control points and basis functions as specified in ISO 19107:2003, 6.4.30. Therefore, `interpolation` may be either "polynomialSpline" or "rationalSpline" depending on the interpolation type; default is "polynomialSpline".

`degree` shall be the degree of the polynomial used for interpolation in this spline.

`gml:knot` shall be the sequence of distinct knots used to define the spline basis functions (see ISO 19107:2003, 6.4.26.2).

The attribute `isPolynomial` shall be set to “true” if this is a polynomial spline (see ISO 19107:2003, 6.4.30.5).

The attribute `knotType` shall provide the type of knot distribution used in defining this spline (see ISO 19107:2003, 6.4.30.4).

The content model follows the general pattern for the encoding of curve segments (see 10.4.7).

10.4.7.14 KnotType, KnotPropertyType

```
<complexType name="KnotType">
    <sequence>
        <element name="value" type="double" />
        <element name="multiplicity" type="nonNegativeInteger" />
        <element name="weight" type="double" />
    </sequence>
</complexType>
```

`gml:Knot` implements ISO 19107 GM_Knot (see D.2.3.4 and ISO 19107:2003, 6.4.24).

A knot is a breakpoint on a piecewise spline curve.

`gml:value` is the value of the parameter at the knot of the spline (see ISO 19107:2003, 6.4.24.2).

`gml:multiplicity` is the multiplicity of this knot used in the definition of the spline (with the same weight).

`gml:weight` is the value of the averaging weight used for this knot of the spline.

```
<complexType name="KnotPropertyType">
    <sequence>
        <element name="Knot" type="gml:KnotType" />
    </sequence>
</complexType>
```

`gml:KnotPropertyType` encapsulates a knot to use it in a geometric type.

10.4.7.15 KnotTypesType

```
<simpleType name="KnotTypesType">
    <restriction base="string">
        <enumeration value="uniform" />
        <enumeration value="quasiUniform" />
```

```
        <enumeration value="piecewiseBezier" />
    </restriction>
</simpleType>
```

`gml:KnotTypesType` implements ISO 19107 GM_KnotType (see D.2.3.4 and ISO 19107:2003, 6.4.25).

This enumeration type specifies values for the knots' type (see ISO 19107:2003, 6.4.25).

10.4.7.16 BezierType, Bezier

```
<complexType name="BezierType">
    <complexContent>
        <restriction base="gml:BSplineType">
            <sequence>
                <choice>
                    <choice minOccurs="0" maxOccurs="unbounded">
                        <element ref="gml:pos" />
                        <element ref="gml:pointProperty" />
                        <element ref="gml:pointRep" />
                    </choice>
                    <element ref="gml:posList" />
                    <element ref="gml:coordinates" />
                </choice>
                <element name="degree" type="nonNegativeInteger" />
                <element name="knot" type="gml:KnotPropertyType" minOccurs="2" maxOccurs="2" />
            </sequence>
            <attribute name="interpolation" type="gml:CurveInterpolationType" fixed="polynomialSpline" />
            <attribute name="isPolynomial" type="boolean" fixed="true" />
            <attribute name="knotType" type="gml:KnotTypesType" use="prohibited" />
        </restriction>
    </complexContent>
</complexType>

<element name="Bezier" type="gml:BezierType" substitutionGroup="gml:BSpline" />
```

`gml:Bezier` implements ISO 19107 GM_Bezier (see D.2.3.4 and ISO 19107:2003, 6.4.31).

Bezier curves are polynomial splines that use Bezier or Bernstein polynomials for interpolation purposes. It is a special case of the B-Spline curve with two knots.

`gml:degree` shall be the degree of the polynomial used for interpolation in this spline.

`gml:knot` shall be the sequence of distinct knots used to define the spline basis functions.

`interpolation` is fixed as "polynomialSpline".

`isPolynomial` is fixed as “true”.

`knotType` is not relevant for Bezier curve segments.

10.4.7.17 OffsetCurveType, OffsetCurve

```
<complexType name="OffsetCurveType">
    <complexContent>
        <extension base="gml:AbstractCurveSegmentType">
            <sequence>
                <element name="offsetBase" type="gml:CurvePropertyType"/>
                <element name="distance" type="gml:LengthType"/>
```

```
                <element name="refDirection" type="gml:VectorType" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<element name="OffsetCurve" type="gml:OffsetCurveType" substitutionGroup="gml:AbstractCurveSegment"/>
```

An offset curve is a curve at a constant distance from the basis curve.

`gml:OffsetCurve` implements ISO 19107 GM_OffsetCurve (see D.2.3.4 and ISO 19107:2003, 6.4.23). `gml:offsetBase` is the base curve from which this curve is defined as an offset. `gml:distance` and `gml:refDirection` have the same meaning as specified in ISO 19107:2003, 6.4.23.

The content model follows the general pattern for the encoding of curve segments (see 10.4.7).

10.4.7.18 AffinePlacementType, AffinePlacement

```
<complexType name="AffinePlacementType">
    <sequence>
        <element name="location" type="gml:DirectPositionType"/>
        <element name="refDirection" type="gml:VectorType" maxOccurs="unbounded"/>
        <element name="inDimension" type="positiveInteger"/>
        <element name="outDimension" type="positiveInteger"/>
    </sequence>
</complexType>

<element name="AffinePlacement" type="gml:AffinePlacementType" substitutionGroup="gml:AbstractObject"/>
```

`gml:AffinePlacement` implements ISO 19107 GM_AffinePlacement (see D.2.3.4 and ISO 19107:2003, 6.4.21 and 6.4.20.1). `gml:location`, `gml:refDirection`, `gml:inDimension` and `gml:outDimension` have the same meaning as specified in ISO 19107:2003, 6.4.21.

10.4.7.19 ClothoidType, Clothoid

```
<complexType name="ClothoidType">
    <complexContent>
        <extension base="gml:AbstractCurveSegmentType">
            <sequence>
                <element name="refLocation">
                    <complexType>
                        <sequence>
                            <element ref="gml:AffinePlacement"/>
                        </sequence>
                    </complexType>
                </element>
                <element name="scaleFactor" type="decimal"/>
                <element name="startParameter" type="double"/>
                <element name="endParameter" type="double"/>
            </sequence>
            <attribute name="interpolation" type="gml:CurveInterpolationType" fixed="clothoid"/>
        </extension>
    </complexContent>
</complexType>

<element name="Clothoid" type="gml:ClothoidType" substitutionGroup="gml:AbstractCurveSegment"/>
```

A clothoid, or Cornu's spiral, is plane curve whose curvature is a fixed function of its length.

gml:Clothoid implements ISO 19107 GM_Clothoid (see D.2.3.4 and ISO 19107:2003, 6.4.22). gml:refLocation, gml:startParameter, gml:endParameter and gml:scaleFactor have the same meaning as specified in ISO 19107:2003, 6.4.22.

interpolation is fixed as "clothoid".

The content model follows the general pattern for the encoding of curve segments (see 10.4.7).

10.4.7.20 GeodesicStringType, GeodesicString

```
<complexType name="GeodesicStringType">
	<complexContent>
		<extension base="gml:AbstractCurveSegmentType">
			<choice>
				<element ref="gml:posList"/>
				<group ref="gml:geometricPositionGroup" minOccurs="2" maxOccurs="unbounded"/>
			</choice>
			<attribute name="interpolation" type="gml:CurveInterpolationType" fixed="geodesic"/>
		</extension>
	</complexContent>
</complexType>

<element name="GeodesicString" type="gml:GeodesicStringType"
	substitutionGroup="gml:AbstractCurveSegment"/>
```

gml:GeodesicString implements ISO 19107 GM_GeodesicString (see D.2.3.4 and ISO 19107:2003, 6.4.12), a sequence of geodesic segments.

The number of control points shall be at least two.

interpolation is fixed as "geodesic".

The content model follows the general pattern for the encoding of curve segments (see 10.4.7).

10.4.7.21 GeodesicType, Geodesic

```
<complexType name="GeodesicType">
	<complexContent>
		<extension base="gml:GeodesicStringType"/>
	</complexContent>
</complexType>

<element name="Geodesic" type="gml:GeodesicType" substitutionGroup="gml:GeodesicString"/>
```

gml:Geodesic implements ISO 19107 GM_Geodesic (see D.2.3.4 and ISO 19107:2003, 6.4.13).

10.5 Geometric primitives (2-dimensional)

10.5.1 AbstractSurfaceType, AbstractSurface

```
<complexType name="AbstractSurfaceType" abstract="true">
	<complexContent>
		<extension base="gml:AbstractGeometricPrimitiveType"/>
	</complexContent>
</complexType>

<element name="AbstractSurface" type="gml:AbstractSurfaceType" abstract="true"
	substitutionGroup="gml:AbstractGeometricPrimitive" />
```

gml:AbstractSurfaceType is an abstraction of a surface to support the different levels of complexity. A surface is always a continuous region of a plane.

The gml:AbstractSurface element is the abstract head of the substitution group for all (continuous) surface elements.

10.5.2 SurfacePropertyType, surfaceProperty

```
<complexType name="SurfacePropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractSurface"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>

<element name="surfaceProperty" type="gml:SurfacePropertyType" />
```

A property that has a surface as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.

This property element either references a surface via the XLink-attributes or contains the surface element. surfaceProperty is the predefined property which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for gml:AbstractSurface.

10.5.3 SurfaceArrayPropertyType, surfaceArrayProperty

```
<complexType name="SurfaceArrayPropertyType">
	<sequence minOccurs="0" maxOccurs="unbounded">
		<element ref="gml:AbstractSurface" />
	</sequence>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

gml:SurfaceArrayPropertyType is a container for an array of surfaces. The elements are always contained in the array property, referencing geometry elements or arrays of geometry elements via XLinks is not supported.

```
<element name="surfaceArrayProperty" type="gml:SurfaceArrayPropertyType" />
```

This property element contains a list of surface elements. surfaceArrayProperty is the predefined property which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for a list of AbstractSurfaces.

10.5.4 PolygonType, Polygon

```
<complexType name="PolygonType">
	<complexContent>
		<extension base="gml:AbstractSurfaceType">
			<sequence>
				<element ref="gml:exterior" minOccurs="0" />
				<element ref="gml:interior" minOccurs="0" maxOccurs="unbounded" />
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

```
<element name="Polygon" type="gml:PolygonType" substitutionGroup="gml:AbstractSurface" />
```

A `gml:Polygon` is a special surface that is defined by a single surface patch (see D.3.6). The boundary of this patch is coplanar and the polygon uses planar interpolation in its interior.

NOTE ISO 19107 GM_Polygon is implemented by `gml:PolygonPatch`.

The elements `gml:exterior` and `gml:interior` describe the surface boundary of the polygon and are specified below.

10.5.5 exterior, interior

```
<element name="exterior" type="gml:AbstractRingPropertyType" />
```

A boundary of a surface consists of a number of rings. In the normal 2D case, one of these rings is distinguished as being the exterior boundary. In a general manifold this is not always possible, in which case all boundaries shall be listed as interior boundaries, and the exterior will be empty.

```
<element name="interior" type="gml:AbstractRingPropertyType" />
```

A boundary of a surface consists of a number of rings. The "interior" rings separate the surface/surface patch from the area enclosed by the rings.

10.5.6 AbstractRingType, AbstractRing

```
<complexType name="AbstractRingType" abstract="true">
    <complexContent>
        <extension base="gml:AbstractCurveType">
            <sequence/>
        </extension>
    </complexContent>
</complexType>

<element name="AbstractRing" type="gml:AbstractRingType" abstract="true"
    substitutionGroup="gml:AbstractCurve"/>
```

An abstraction of a ring to support surface boundaries of different complexity.

The `gml:AbstractRing` element is the abstract head of the substitution group for all closed boundaries of a surface patch.

10.5.7 AbstractRingPropertyType

```
<complexType name="AbstractRingPropertyType">
    <sequence>
        <element ref="gml:AbstractRing"/>
    </sequence>
</complexType>
```

A property with the content model of `gml:AbstractRingPropertyType` encapsulates a ring to represent the surface boundary property of a surface.

10.5.8 LinearRingType, LinearRing

```
<complexType name="LinearRingType">
    <complexContent>
        <extension base="gml:AbstractRingType">
```



```
                <sequence>
                    <choice>
                        <choice minOccurs="4" maxOccurs="unbounded">
                            <element ref="gml:pos" />
                            <element ref="gml:pointProperty" />
                            <element ref="gml:pointRep"/>
                        </choice>
                        <element ref="gml:posList" />
                        <element ref="gml:coordinates" />
                    </choice>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <element name="LinearRing" type="gml:LinearRingType" substitutionGroup="gml:AbstractRing" />
```

A `gml:LinearRing` is defined by four or more coordinate tuples, with linear interpolation between them; the first and last coordinates shall be coincident.

The encoding of the control points follows the pattern described in 10.1.4.4. The number of direct positions in the list shall be at least four.

10.5.9 LinearRingPropertyType

```
    <complexType name="LinearRingPropertyType">
        <sequence>
            <element ref="gml:LinearRing"/>
        </sequence>
    </complexType>
```

A property with the content model of `gml:LinearRingPropertyType` encapsulates a linear ring to represent a component of a surface boundary.

10.5.10 SurfaceType, Surface

```
    <complexType name="SurfaceType">
        <complexContent>
            <extension base="gml:AbstractSurfaceType">
                <sequence>
                    <element ref="gml:patches" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <element name="Surface" type="gml:SurfaceType" substitutionGroup="gml:AbstractSurface" />
```

A Surface is a 2-dimensional primitive and is composed of one or more surface patches as specified in ISO 19107:2003, 6.3.17.1. The surface patches are connected to one another.

`gml:Surface` implements ISO 19107 GM_Surface (see D.2.3.4 and ISO 19107:2003, 6.3.17).

`gml:patches` encapsulates the patches of the surface.

10.5.11 OrientableSurfaceType, OrientableSurface, baseSurface

```
    <complexType name="OrientableSurfaceType">
        <complexContent>
```

```
            <extension base="gml:AbstractSurfaceType">
                <sequence>
                    <element ref="gml:baseSurface" />
                </sequence>
                <attribute name="orientation" type="gml:SignType" default="+" />
            </extension>
        </complexContent>
    </complexType>

    <element name="baseSurface" type="gml:SurfacePropertyType" />

    <element name="OrientableSurface" type="gml:OrientableSurfaceType" substitutionGroup="gml:AbstractSurface" />
```

`gml:OrientableSurface` implements ISO 19107 GM_OrientableSurface (see D.2.3.4 and ISO 19107:2003, 6.3.15).

`gml:OrientableSurface` consists of a surface and an orientation. If the `orientation` is "+", then the `gml:OrientableSurface` is identical to the `gml:baseSurface`. If the `orientation` is "-", then the `gml:OrientableSurface` is a reference to a `gml:AbstractSurface` with an up-normal that reverses the direction for this `gml:OrientableSurface`, the sense of "the top of the surface".

The property `gml:baseSurface` references or contains the base surface. The property `gml:baseSurface` either references the base surface via the XLink-attributes or contains the surface element. A surface element is any element which is substitutable for `gml:AbstractSurface`. The base surface has positive orientation.

NOTE This definition allows for a nested structure, i.e. a `gml:OrientableSurface` may use another `gml:OrientableSurface` as its base surface.

10.5.11.1 Ring, RingType, curveMember

```
    <complexType name="RingType">
        <complexContent>
            <extension base="gml:AbstractRingType">
                <sequence>
                    <element ref="gml:curveMember" maxOccurs="unbounded" />
                </sequence>
                <attributeGroup ref="gml:AggregationAttributeGroup"/>
            </extension>
        </complexContent>
    </complexType>

    <element name="Ring" type="gml:RingType" substitutionGroup="gml:AbstractRing" />

    <element name="curveMember" type="gml:CurvePropertyType" />
```

`gml:Ring` implements ISO 19107 GM_Ring (see D.2.3.4 and ISO 19107:2003, 6.3.6).

A ring is used to represent a single connected component of a surface boundary as specified in ISO 19107:2003, 6.3.6.

Every `gml:curveMember` references or contains one curve, i.e. any element which is substitutable for `gml:AbstractCurve`. In the context of a ring, the curves describe the boundary of the surface. The sequence of curves shall be contiguous and connected in a cycle.

If provided, the `aggregationType` attribute shall have the value “sequence”.

NOTE This definition allows for a nested structure, i.e. a gml:curveMember may be a gml:CompositeCurve which in turn may be constructed from other gml:CompositeCurves as a curve members.

10.5.11.2 RingPropertyType

```
<complexType name="RingPropertyType">
    <sequence>
        <element ref="gml:Ring"/>
    </sequence>
</complexType>
```

A property with the content model of gml:RingPropertyType encapsulates a ring to represent a component of a surface boundary.

10.5.11.3 PolyhedralSurface

```
<element name="PolyhedralSurface" type="gml:SurfaceType" substitutionGroup="gml:Surface"/>
```

gml:PolyhedralSurface implements ISO 19107 GM_PolyhedralSurface (see D.2.3.4 and ISO 19107:2003, 6.4.35).

A polyhedral surface is a surface composed of polygon patches connected along their common boundary curves.

gml:patches encapsulates the polygon patches of the polyhedral surface. All patches shall be polygon patches.

10.5.11.4 TriangulatedSurface

```
<element name="TriangulatedSurface" type="gml:SurfaceType" substitutionGroup="gml:Surface"/>
```

gml:TriangultedSurface implements ISO 19107 GM_TriangulatedSurface (see D.2.3.4 and ISO 19107:2003, 6.4.37).

A triangulated surface is a polyhedral surface that is composed only of triangles. There is no restriction on how the triangulation is derived.

gml:patches encapsulates the triangles of the triangulated surface. All patches shall be triangle patches.

10.5.11.5 TinType, Tin

```
<complexType name="TinType">
    <complexContent>
        <extension base="gml:TriangulatedSurfaceType">
            <sequence>
                <element name="stopLines" type="gml:LineStringSegmentArrayPropertyType" minOccurs="0"
                 maxOccurs="unbounded"/>
                <element name="breakLines" type="gml:LineStringSegmentArrayPropertyType" minOccurs="0"
                 maxOccurs="unbounded"/>
                <element name="maxLength" type="gml:LengthType"/>
                <element name="controlPoint">
                    <complexType>
                        <choice>
                            <element ref="gml:posList"/>
                            <group ref="gml:geometricPositionGroup" minOccurs="3"
                             maxOccurs="unbounded"/>
                        </choice>
                    </complexType>
                </element>
            </sequence>
```



```
            </extension>
        </complexContent>
    </complexType>

    <element name="Tin" type="gml:TinType" substitutionGroup="gml:TriangulatedSurface"/>
```

`gml:Tin` implements ISO 19107 GM_Tin (see D.2.3.4 and ISO 19107:2003, 6.4.39).

A tin is a triangulated surface that uses the Delauny algorithm or a similar algorithm complemented with consideration of stoplines (`gml:stopLines`), breaklines (`gml:breakLines`), and maximum length of triangle sides (`gml:maxLength`). `gml:controlPoint` shall contain a set of the positions (three or more) used as posts for this TIN (corners of the triangles in the TIN). See ISO 19107:2003, 6.4.39 for details.

10.5.11.6 LineStringSegmentArrayPropertyType

```
<complexType name="LineStringSegmentArrayPropertyType">
    <sequence minOccurs="0" maxOccurs="unbounded">
        <element ref="gml:LineStringSegment" />
    </sequence>
</complexType>
```

`gml:LineStringSegmentArrayPropertyType` provides a container for line strings.

10.5.12 Surface patches

10.5.12.1 AbstractSurfacePatchType, gml : AbstractSurfacePatch

```
<complexType name="AbstractSurfacePatchType" abstract="true" />

<element name="AbstractSurfacePatch" type="gml:AbstractSurfacePatchType" abstract="true" />
```

A surface patch defines a homogenous portion of a surface.

`gml:AbstractSurfacePatch` implements ISO 19107 GM_SurfacePatch (see D.2.3.4 and ISO 19107:2003, 6.4.34).

The `gml:AbstractSurfacePatch` element is the abstract head of the substitution group for all surface patch elements describing a continuous portion of a surface.

All surface patches shall have an attribute `interpolation` (declared in the types derived from `gml:AbstractSurfacePatchType`) specifying the interpolation mechanism used for the patch using `gml:SurfaceInterpolationType` (see 10.5.12.3).

10.5.12.2 SurfacePatchArrayPropertyType, patches

```
<complexType name="SurfacePatchArrayPropertyType">
    <sequence minOccurs="0" maxOccurs="unbounded">
        <element ref="gml:AbstractSurfacePatch" />
    </sequence>
</complexType>
```

`gml:SurfacePatchArrayPropertyType` is a container for a sequence of surface patches.

```
<element name="patches" type="gml:SurfacePatchArrayPropertyType" />
```

The `gml:patches` property element contains the sequence of surface patches. The order of the elements is significant and shall be preserved when processing the array.

10.5.12.3 SurfaceInterpolationType

```
<simpleType name="SurfaceInterpolationType">
	<restriction base="string">
		<enumeration value="none" />
		<enumeration value="planar" />
		<enumeration value="spherical" />
		<enumeration value="elliptical" />
		<enumeration value="conic" />
		<enumeration value="tin" />
		<enumeration value="parametricCurve" />
		<enumeration value="polynomialSpline" />
		<enumeration value="rationalSpline" />
		<enumeration value="triangulatedSpline" />
	</restriction>
</simpleType>
```

`gml:SurfaceInterpolationType` is a list of codes that may be used to identify the interpolation mechanisms specified by an application schema.

This type implements ISO 19107 GM_SurfaceInterpolation (see D.2.3.4 and ISO 19107:2003, 6.4.32).

10.5.12.4 PolygonPatchType, PolygonPatch

```
<complexType name="PolygonPatchType">
	<complexContent>
		<extension base="gml:AbstractSurfacePatchType">
			<sequence>
				<element ref="gml:exterior" minOccurs="0" />
				<element ref="gml:interior" minOccurs="0" maxOccurs="unbounded" />
			</sequence>
			<attribute name="interpolation" type="gml:SurfaceInterpolationType" fixed="planar" />
		</extension>
	</complexContent>
</complexType>

<element name="PolygonPatch" type="gml:PolygonPatchType" substitutionGroup="gml:AbstractSurfacePatch" />
```

`gml:PolygonPatch` implements ISO 19107 GM_Polygon (see D.2.3.4 and ISO 19107:2003, 6.4.36).

A `gml:PolygonPatch` is a surface patch that is defined by a set of boundary curves and an underlying surface to which these curves adhere. The curves shall be coplanar and the polygon uses planar interpolation in its interior.

`interpolation` is fixed to "planar", i.e. an interpolation shall return points on a single plane. The boundary of the patch shall be contained within that plane.

10.5.12.5 TriangleType, Triangle

```
<complexType name="TriangleType">
	<complexContent>
		<extension base="gml:AbstractSurfacePatchType">
			<sequence>
				<element ref="gml:exterior" />
			</sequence>
			<attribute name="interpolation" type="gml:SurfaceInterpolationType" fixed="planar" />
		</extension>
	</complexContent>
</complexType>
```



```
<element name="Triangle" type="gml:TriangleType" substitutionGroup="gml:AbstractSurfacePatch" />
```

`gml:Triangle` represents a triangle as a surface patch with an outer boundary consisting of a linear ring. Note that this is a polygon (subtype) with no inner boundaries. The number of points in the linear ring shall be four.

The ring (element `gml:exterior`) shall be a `gml:LinearRing` and shall form a triangle, the first and the last position shall be coincident.

`interpolation` is fixed to "planar", i.e. an interpolation shall return points on a single plane. The boundary of the patch shall be contained within that plane.

10.5.12.6 RectangleType, Rectangle

```
<complexType name="RectangleType">
	<complexContent>
		<extension base="gml:AbstractSurfacePatchType">
			<sequence>
				<element ref="gml:exterior" />
			</sequence>
			<attribute name="interpolation" type="gml:SurfaceInterpolationType" fixed="planar" />
		</extension>
	</complexContent>
</complexType>
```

```
<element name="Rectangle" type="gml:RectangleType" substitutionGroup="gml:AbstractSurfacePatch" />
```

`gml:Rectangle` represents a rectangle as a surface patch with an outer boundary consisting of a linear ring. Note that this is a polygon (subtype) with no inner boundaries. The number of points in the linear ring shall be five.

NOTE While conceptually a rectangle is a subtype of a polygon, defining gml:RectangleType as a type derived by restriction from gml:PolygonType is problematic due to the way the restriction construct is defined in XML Schema and has thus been avoided in this case.

The ring (element `gml:exterior`) shall be a `gml:LinearRing` and shall form a rectangle; the first and the last position shall be coincident.

`interpolation` is fixed to "planar", i.e. an interpolation shall return points on a single plane. The boundary of the patch shall be contained within that plane.

10.5.12.7 PointGrid

```
<group name="PointGrid">
	<sequence>
		<element name="rows">
			<complexType>
				<sequence>
					<element name="Row" maxOccurs="unbounded">
						<complexType>
							<group ref="gml:geometricPositionListGroup"/>
						</complexType>
					</element>
				</sequence>
			</complexType>
		</element>
	</sequence>
</group>
```

`gml:PointGrid` implements ISO 19107 GM_PointGrid (see D.2.3.4 and ISO 19107:2003, 6.4.6).

A `gml:PointGrid` group contains or references points or positions which are organized into sequences or grids. All `gml:row`s shall have the same number of positions (columns).

10.5.12.8 AbstractParametricCurveSurfaceType, AbstractParametricCurveSurface

```
<complexType name="AbstractParametricCurveSurfaceType" abstract="true">
	<complexContent>
		<extension base="gml:AbstractSurfacePatchType">
			<attributeGroup ref="gml:AggregationAttributeGroup"/>
		</extension>
	</complexContent>
</complexType>

<element name="AbstractParametricCurveSurface" type="gml:AbstractParametricCurveSurfaceType" abstract="true"
substitutionGroup="gml:AbstractSurfacePatch"/>
```

`gml:AbstractParametricCurveSurface` implements ISO 19107 GM_ParametricCurveSurface (see D.2.3.4 and ISO 19107:2003, 6.4.40).

The element provides a substitution group head for the surface patches based on parametric curves. All properties are specified in the derived subtypes. All derived subtypes shall conform to the constraints specified in ISO 19107:2003, 6.4.40.

If provided, the `aggregationType` attribute shall have the value “set”.

10.5.12.9 AbstractGriddedSurfaceType, AbstractGriddedSurface

```
<complexType name="AbstractGriddedSurfaceType" abstract="true">
	<complexContent>
		<extension base="gml:AbstractParametricCurveSurfaceType">
			<sequence>
				<group ref="gml:PointGrid"/>
			</sequence>
			<attribute name="rows" type="integer"/>
			<attribute name="columns" type="integer"/>
		</extension>
	</complexContent>
</complexType>

<element name="AbstractGriddedSurface" type="gml:AbstractGriddedSurfaceType" abstract="true"
 substitutionGroup="gml:AbstractParametricCurveSurface"/>
```

`gml:AbstractGriddedSurface` implements ISO 19107 GM_GriddedSurface (see D.2.3.4 and ISO 19107:2003, 6.4.41). If provided, `rows` gives the number of rows, `columns` gives the number of columns in the parameter grid. The parameter grid is represented by an instance of the `gml:PointGrid` group.

The element provides a substitution group head for the surface patches based on a grid. All derived subtypes shall conform to the constraints specified in ISO 19107:2003, 6.4.41.

10.5.12.10 ConeType, Cone

```
<complexType name="ConeType">
	<complexContent>
		<extension base="gml:AbstractGriddedSurfaceType">
			<attribute name="horizontalCurveType" type="gml:CurveInterpolationType"
			fixed="circularArc3Points"/>
			<attribute name="verticalCurveType" type="gml:CurveInterpolationType"
			fixed="linear"/>
```

```
            </extension>
        </complexContent>
    </complexType>

    <element name="Cone" type="gml:ConeType" substitutionGroup="gml:AbstractGriddedSurface"/>
```

`gml:Cone` implements ISO 19107 GM_Cone (see D.2.3.4 and ISO 19107:2003, 6.4.42).

10.5.12.11 CylinderType, gmlCylinder

```
    <complexType name="CylinderType">
        <complexContent>
            <extension base="gml:AbstractGriddedSurfaceType">
                <attribute name="horizontalCurveType" type="gml:CurveInterpolationType"
                fixed="circularArc3Points"/>
                <attribute name="verticalCurveType" type="gml:CurveInterpolationType"
                fixed="linear"/>
            </extension>
        </complexContent>
    </complexType>

    <element name="Cylinder" type="gml:CylinderType" substitutionGroup="gml:AbstractGriddedSurface"/>
```

`gml:Cylinder` implements ISO 19107 GM_Cylinder (see D.2.3.4 and ISO 19107:2003, 6.4.43).

10.5.12.12 SphereType, Sphere

```
    <complexType name="SphereType">
        <complexContent>
            <extension base="gml:AbstractGriddedSurfaceType">
                <attribute name="horizontalCurveType" type="gml:CurveInterpolationType"
                 fixed="circularArc3Points"/>
                <attribe name="verticalCurveType" type="gml:CurveInterpolationType"
                fixed="circularArc3Points"/>
            </extension>
        </complexContent>
    </complexType>

    <element name="Sphere" type="gml:SphereType" substitutionGroup="gml:AbstractGriddedSurface"/>
```

`gml:Sphere` implements ISO 19107 GM_Sphere (see D.2.3.4 and ISO 19107:2003, 6.4.44).

10.6 Geometric primitives (3-dimensional)

10.6.1 AbstractSolidType, AbstractSolid

```
    <complexType name="AbstractSolidType">
        <complexContent>
            <extension base="gml:AbstractGeometricPrimitiveType"/>
        </complexContent>
    </complexType>

    <element name="AbstractSolid" type="gml:AbstractSolidType" abstract="true"
    substitutionGroup="gml:AbstractGeometricPrimitive" />
```

`gml:AbstractSolidType` is an abstraction of a solid to support the different levels of complexity. The solid may always be viewed as a geometric primitive, i.e. is contiguous.

The gml:AbstractSolid element is the abstract head of the substitution group for all (continuous) solid elements.

10.6.2 SolidPropertyType, solidProperty

```
<complexType name="SolidPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractSolid"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

A property that has a solid as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.

```
<element name="solidProperty" type="gml:SolidPropertyType"/>
```

This property element either references a solid via the XLink-attributes or contains the solid element. gml:solidProperty is the predefined property which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for gml:AbstractSolid.

10.6.3 SolidArrayPropertyType, solidArrayProperty

```
<complexType name="SolidArrayPropertyType">
	<sequence minOccurs="0" maxOccurs="unbounded">
		<element ref="gml:AbstractSolid" />
	</sequence>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

gml:SolidArrayPropertyType is a container for an array of solids. The elements are always contained in the array property, referencing geometry elements or arrays of geometry elements is not supported.

```
<element name="solidArrayProperty" type="gml:SolidArrayPropertyType"/>
```

This property element contains a list of solid elements. solidArrayProperty is the predefined property which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for a list of gml:AbstractSolid.

10.6.4 SolidType, Solid

```
<complexType name="SolidType">
	<complexContent>
		<extension base="gml:AbstractSolidType">
			<sequence>
				<element name="exterior" type="gml:ShellPropertyType" minOccurs="0"/>
				<element name="interior" type="gml:ShellPropertyType" minOccurs="0"
				 maxOccurs="unbounded"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>

<element name="Solid" type="gml:SolidType" substitutionGroup="gml:AbstractSolid" />
```


`gml:Solid` implements ISO 19107 GM_Solid (see D.2.3.3 and ISO 19107:2003, 6.3.18).

A solid is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces as specified in ISO 19107:2003, 6.3.18. `gml:exterior` specifies the outer boundary, `gml:interior` the inner boundary of the solid.

10.6.5 ShellType, Shell

```
<complexType name="ShellType">
    <complexContent>
        <extension base="gml:AbstractSurfaceType">
            <sequence>
                <element ref="gml:surfaceMember" maxOccurs="unbounded"/>
            </sequence>
            <attributeGroup ref="gml:AggregationAttributeGroup"/>
        </extension>
    </complexContent>
</complexType>

<element name="Shell" type="gml:ShellType" substitutionGroup="gml:AbstractSurface"/>

<element name="surfaceMember" type="gml:SurfacePropertyType"/>
```

`gml:Shell` implements ISO 19107 GM_Shell (see D.2.3.3 and ISO 19107:2003, 6.3.8).

A shell is used to represent a single connected component of a solid boundary as specified in ISO 19107:2003, 6.3.8.

Every `gml:surfaceMember` references or contains one surface, i.e. any element which is substitutable for `gml:AbstractSurface`. In the context of a shell, the surfaces describe the boundary of the solid.

If provided, the `aggregationType` attribute shall have the value “set”.

NOTE This definition allows for a nested structure, i.e. a `gml:surfaceMember` may be a `gml:CompositeSurface` which in turn may be constructed from other `gml:CompositeSurface`s as a surface members.

10.6.6 ShellPropertyType

```
<complexType name="ShellPropertyType">
    <sequence>
        <element ref="gml:Shell"/>
    </sequence>
</complexType>
```

A property with the content model of `gml:ShellPropertyType` encapsulates a shell to represent a component of a solid boundary.

11 GML schema — Geometric complex, geometric composites and geometric aggregates

11.1 Overview

This clause describes the geometry schema components for geometric complexes and aggregates.

NOTE The geometry schema documents, geometryAggregates.xsd and geometryComplexes.xsd (see Annex C), are identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:geometryAggregates:3.2.1

urn:x-ogc:specification:gml:schema-xsd:geometryComplexes:3.2.1

Geometric aggregates (i.e. instances of a subtype of `gml:AbstractGeometricAggregateType`) are arbitrary aggregations of geometry elements. They are not assumed to have any additional internal structure and are used to "collect" pieces of geometry of a specified type. Application schemas may use aggregates for features that use multiple geometric objects in their representations.

Geometric complexes (i.e. instances of `gml:GeometricComplexType`) are closed collections of geometric primitives, i.e. they will contain their boundaries.

A geometric complex (`gml:GeometricComplex`) is defined by ISO 19107:2003, 6.6.1 as "a set of primitive geometric objects (in a common coordinate system) whose interiors are disjoint. Further, if a primitive is in a geometric complex, then there exists a set of primitives in that complex whose point-wise union is the boundary of this first primitive."

A geometric composite (`gml:CompositeCurve`, `gml:CompositeSurface` and `gml:CompositeSolid`) represents a geometric complex with an underlying core geometry that is isomorphic to a primitive, i.e. it can be viewed as a primitive and as a complex. See ISO 19107:2003, 6.1 and 6.6.3 for more details on the nature of composite geometries.

Geometric complexes and composites are intended to be used in application schemas where the sharing of geometry is important.

11.2 Geometric complex and geometric composites

11.2.1 Geometric complex

11.2.1.1 GeometricComplexType, GeometricComplex

```
<complexType name="GeometricComplexType">
    <complexContent>
        <extension base="gml:AbstractGeometryType">
            <sequence>
                <element name="element" type="gml:GeometricPrimitivePropertyType"
                maxOccurs="unbounded" />
            </sequence>
            <attributeGroup ref="gml:AggregationAttributeGroup"/>
        </extension>
    </complexContent>
</complexType>

<element name="GeometricComplex" type="gml:GeometricComplexType"
substitutionGroup="gml:AbstractGeometry" />
```

`gml:GeometricComplex` implements ISO 19107 GM_Complex (see ISO 19107:2003, 6.6.2 and 6.6.1) as specified in D.2.3.6.

`gml:element` references or contains inline one geometric primitive (this includes composite geometries).

11.2.1.2 GeometricComplexPropertyType

```
<complexType name="GeometricComplexPropertyType">
    <sequence minOccurs="0">
        <choice>
            <element ref="gml:GeometricComplex"/>
```



```
                <element ref="gml:CompositeCurve"/>
                <element ref="gml:CompositeSurface"/>
                <element ref="gml:CompositeSolid"/>
            </choice>
        </sequence>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

A property that has a geometric complex as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.

NOTE The allowed geometry elements contained in such a property (or referenced by it) are modelled by an XML Schema choice element since the composites (conceptually) inherit both from geometric complex *and* geometric primitive and are already part of the `gml:AbstractGeometricPrimitive` substitution group.

11.2.2 Composite geometries

11.2.2.1 General representation of composites in GML

The members of a geometric composite shall represent a homogeneous collection of geometric primitives whose union would be the core geometry of the composite. The complex would include all member primitives *and* all primitives on the boundary of these primitives, and so forth until `gml:Point`s are included. Thus the "member" properties in `gml:CompositeCurve`, `gml:CompositeSurface` and `gml:CompositeSolid` represent a subset of the `gml:element` property of `gml:GeometricComplex`.

As XML Schema does not support the concept of "multiple inheritance" which is used in ISO 19107 to express the duality of the geometric composites (as an open primitive and as a closed complex) in the GML schema, the composites derive from `gml:AbstractGeometricPrimitiveType` only. However, by using a `<choice>` element in the property type `gml:GeometricComplexPropertyType`, a composite can be used in any property which expects a `gml:GeometricComplex` as its value.

11.2.2.2 CompositeCurveType, CompositeCurve

```
<complexType name="CompositeCurveType">
    <complexContent>
        <extension base="gml:AbstractCurveType">
            <sequence>
                <element ref="gml:curveMember" maxOccurs="unbounded" />
            </sequence>
            <attributeGroup ref="gml:AggregationAttributeGroup"/>
        </extension>
    </complexContent>
</complexType>

<element name="CompositeCurve" type="gml:CompositeCurveType" substitutionGroup="gml:AbstractCurve" />
```

`gml:CompositeCurve` implements ISO 19107 GM_CompositeCurve (see ISO 19107:2003, 6.6.5) as specified in D.2.3.6.

A `gml:CompositeCurve` is represented by a sequence of (orientable) curves such that each curve in the sequence terminates at the start point of the subsequent curve in the list.

`gml:curveMember` references or contains inline one curve in the composite curve.

The curves are contiguous, the collection of curves is ordered. Therefore, if provided, the `aggregationType` attribute shall have the value “sequence”.

NOTE This definition allows for a nested structure, i.e. a `gml:CompositeCurve` may use, for example, another `gml:CompositeCurve` as a curve member.

11.2.2.3 CompositeSurfaceType, CompositeSurface

```
<complexType name="CompositeSurfaceType">
	<complexContent>
		<extension base="gml:AbstractSurfaceType">
			<sequence>
				<element ref="gml:surfaceMember" maxOccurs="unbounded" />
			</sequence>
			<attributeGroup ref="gml:AggregationAttributeGroup"/>
		</extension>
	</complexContent>
</complexType>

<element name="CompositeSurface" type="gml:CompositeSurfaceType" substitutionGroup="gml:AbstractSurface" />
```

`gml:CompositeSurface` implements ISO 19107 GM_CompositeSurface (see ISO 19107:2003, 6.6.6) as specified in D.2.3.6.

A `gml:CompositeSurface` is represented by a set of orientable surfaces. It is a geometry type with all the geometric properties of a (primitive) surface. Essentially, a composite surface is a collection of surfaces that join in pairs on common boundary curves and which, when considered as a whole, form a single surface.

`gml:surfaceMember` references or contains inline one surface in the composite surface.

The surfaces are contiguous.

NOTE This definition allows for a nested structure, i.e. a `gml:CompositeSurface` may use, for example, another `gml:CompositeSurface` as a surface member.

11.2.2.4 CompositeSolidType, CompositeSolid

```
<complexType name="CompositeSolidType">
	<complexContent>
		<extension base="gml:AbstractSolidType">
			<sequence>
				<element ref="gml:solidMember" maxOccurs="unbounded" />
			</sequence>
			<attributeGroup ref="gml:AggregationAttributeGroup"/>
		</extension>
	</complexContent>
</complexType>

<element name="CompositeSolid" type="gml:CompositeSolidType" substitutionGroup="gml:AbstractSolid" />
```

`gml:CompositeSolid` implements ISO 19107 GM_CompositeSolid (see ISO 19107:2003, 6.6.7) as specified in D.2.3.6.

A `gml:CompositeSolid` is represented by a set of orientable surfaces. It is a geometry type with all the geometric properties of a (primitive) solid. Essentially, a composite solid is a collection of solids that join in pairs on common boundary surfaces and which, when considered as a whole, form a single solid.

`gml:solidMember` references or contains one solid in the composite solid. The solids are contiguous.

NOTE This definition allows for a nested structure, i.e. a `gml:CompositeSolid` may use, for example, another `gml:CompositeSolid` as a member.

11.3 Geometric aggregates

11.3.1 Aggregates of unspecified dimensionality

11.3.1.1 AbstractGeometricAggregateType, AbstractGeometricAggregate

```
<complexType name="AbstractGeometricAggregateType" abstract="true">
    <complexContent>
        <extension base="gml:AbstractGeometryType">
            <attributeGroup ref="gml:AggregationAttributeGroup"/>
        </extension>
    </complexContent>
</complexType>

<element name="AbstractGeometricAggregate" type="gml:AbstractGeometricAggregateType" abstract="true"
substitutionGroup="gml:AbstractGeometry" />
```

`gml:AbstractGeometricAggregate` implements ISO 19107 GM_Aggregate (see ISO 19107:2003, 6.5.2) as specified in D.2.3.5. It is the abstract head of the substitution group for all geometric aggregates.

11.3.1.2 MultiGeometryType, MultiGeometry, geometryMember, geometryMembers

```
<complexType name="MultiGeometryType">
    <complexContent>
        <extension base="gml:AbstractGeometricAggregateType">
            <sequence>
                <element ref="gml:geometryMember" minOccurs="0" maxOccurs="unbounded" />
                <element ref="gml:geometryMembers" minOccurs="0" />
            </sequence>
        </extension>
    </complexContent>
</complexType>

<element name="MultiGeometry" type="gml:MultiGeometryType"
substitutionGroup="gml:AbstractGeometricAggregate" />
```

`gml:MultiGeometry` is a collection of one or more GML geometry objects of arbitrary type (see D.3.8).

The members of the geometric aggregate may be specified either using the "standard" property (`gml:geometryMember`) or the array property (`gml:geometryMembers`). It is also valid to use both the "standard" and the array properties in the same collection.

NOTE Array properties cannot reference remote geometry elements via XLinks.

```
<element name="geometryMember" type="gml:GeometryPropertyType" />
```

This property element either references a geometry element via the XLink-attributes or contains the geometry element.

```
<element name="geometryMembers" type="gml:GeometryArrayPropertyType" />
```

This property element contains a list of geometry elements. The order of the elements is significant and shall be preserved when processing the array.

11.3.1.3 MultiGeometryPropertyType, multiGeometryProperty

```
<complexType name="MultiGeometryPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractGeometricAggregate"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

A property that has a geometric aggregate as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.

```
<element name="multiGeometryProperty" type="gml:MultiGeometryPropertyType" />
```

This property element either references a geometric aggregate via the XLink-attributes or contains the "multi geometry" element. `gml:multiGeometryProperty` is the predefined property, which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for `gml:AbstractGeometricAggregate`.

11.3.2 0-Dimensional aggregates

11.3.2.1 MultiPointType, MultiPoint, pointMember, pointMembers

```
<complexType name="MultiPointType">
	<complexContent>
		<extension base="gml:AbstractGeometricAggregateType">
			<sequence>
				<element ref="gml:pointMember" minOccurs="0" maxOccurs="unbounded" />
				<element ref="gml:pointMembers" minOccurs="0" />
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

```
<element name="MultiPoint" type="gml:MultiPointType" substitutionGroup="gml:AbstractGeometricAggregate" />
```

`gml:MultiPoint` implements ISO 19107 GM_MultiPoint (see ISO 19107:2003, 6.5.4) as specified in D.2.3.5. A `gml:MultiPoint` consists of one or more `gml:Points`.

The members of the geometric aggregate may be specified either using the "standard" property (`gml:pointMember`) or the array property (`gml:pointMembers`). It is also valid to use both the "standard" and the array properties in the same collection.

NOTE Array properties cannot reference remote geometry elements via XLinks.

```
<element name="pointMember" type="gml:PointPropertyType" />
```

This property element either references a Point via the XLink-attributes or contains the Point element.

```
<element name="pointMembers" type="gml:PointArrayPropertyType" />
```

This property element contains a list of points. The order of the elements is significant and shall be preserved when processing the array.

11.3.2.2 MultiPointPropertyType, multiPointProperty

```
<complexType name="MultiPointPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:MultiPoint"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

A property that has a collection of points as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.

```
<element name="multiPointProperty" type="gml:MultiPointPropertyType" />
```

This property element either references a point aggregate via the XLink-attributes or contains the "multi point" element. `gml:multiPointProperty` is the predefined property, which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for `gml:MultiPoint`.

11.3.3 1-Dimensional aggregates

11.3.3.1 MultiCurveType, multiCurve, curveMembers

```
<complexType name="MultiCurveType">
	<complexContent>
		<extension base="gml:AbstractGeometricAggregateType">
			<sequence>
				<element ref="gml:curveMember" minOccurs="0" maxOccurs="unbounded" />
				<element ref="gml:curveMembers" minOccurs="0" />
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

```
<element name="MultiCurve" type="gml:MultiCurveType" substitutionGroup="gml:AbstractGeometricAggregate" />
```

`gml:MultiCurve` implements ISO 19107 GM_MultiCurve (see ISO 19107:2003, 6.5.5) as specified in D.2.3.5. A `gml:MultiCurve` is defined by one or more `gml:AbstractCurve`s.

The members of the geometric aggregate may be specified either using the "standard" property (`gml:curveMember`) or the array property (`gml:curveMembers`). It is also valid to use both the "standard" and the array properties in the same collection.

NOTE 1 Array properties cannot reference remote geometry elements via XLinks.

```
<element name="curveMembers" type="gml:CurveArrayPropertyType" />
```

This property element contains a list of curves. The order of the elements is significant and shall be preserved when processing the array.

NOTE 2 `gml:curveMember` is declared in 10.5.11.1.

11.3.3.2 MultiCurvePropertyType, multiCurveProperty

```
<complexType name="MultiCurvePropertyType">
	<sequence minOccurs="0">
```

```
            <element ref="gml:MultiCurve"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </complexType>
```

A property that has a collection of curves as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.

```
    <element name="multiCurveProperty" type="gml:MultiCurvePropertyType" />
```

This property element either references a curve aggregate via the XLink-attributes or contains the "multi curve" element. `gml:multiCurveProperty` is the predefined property, which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for `gml:MultiCurve`.

11.3.4 2-Dimensional aggregates

11.3.4.1 MultiSurfaceType, MultiSurface, surfaceMember, surfaceMembers

```
    <complexType name="MultiSurfaceType">
        <complexContent>
            <extension base="gml:AbstractGeometricAggregateType">
                <sequence>
                    <element ref="gml:surfaceMember" minOccurs="0" maxOccurs="unbounded" />
                    <element ref="gml:surfaceMembers" minOccurs="0" />
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <element name="MultiSurface" type="gml:MultiSurfaceType"
    substitutionGroup="gml:AbstractGeometricAggregate" />
```

`gml:MultiSurface` implements ISO 19107 GM_MultiSurface (see ISO 19107:2003, 6.5.6) as specified in D.2.3.5. A `gml:MultiSurface` is defined by one or more `gml:AbstractSurfaces`.

The members of the geometric aggregate may be specified either using the "standard" property (`gml:surfaceMember`) or the array property (`gml:surfaceMembers`). It is also valid to use both the "standard" and the array properties in the same collection.

NOTE Array properties cannot reference remote geometry elements via XLinks.

```
    <element name="surfaceMember" type="gml:SurfacePropertyType" />
```

This property element either references a surface via the XLink-attributes or contains the surface element. A surface element is any element, which is substitutable for `gml:AbstractSurface`.

```
    <element name="surfaceMembers" type="gml:SurfaceArrayPropertyType" />
```

This property element contains a list of surfaces. The order of the elements is significant and shall be preserved when processing the array.

11.3.4.2 MultiSurfacePropertyType, multiSurfaceProperty

```
    <complexType name="MultiSurfacePropertyType">
        <sequence minOccurs="0">
```

```
            <element ref="gml:MultiSurface"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

A property that has a collection of surfaces as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.

```
<element name="multiSurfaceProperty" type="gml:MultiSurfacePropertyType" />
```

This property element either references a surface aggregate via the XLink-attributes or contains the "multi surface" element. `gml:multiSurfaceProperty` is the predefined property, which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for `gml:MultiSurface`.

11.3.5 3-Dimensional aggregates

11.3.5.1 MultiSolidType, MultiSolid, solidMember, solidMembers

```
<complexType name="MultiSolidType">
    <complexContent>
        <extension base="gml:AbstractGeometricAggregateType">
            <sequence>
                <element ref="gml:solidMember" minOccurs="0" maxOccurs="unbounded" />
                <element ref="gml:solidMembers" minOccurs="0" />
            </sequence>
        </extension>
    </complexContent>
</complexType>

<element name="MultiSolid" type="gml:MultiSolidType" substitutionGroup="gml:AbstractGeometricAggregate" />
```

`gml:MultiSolid` implements ISO 19107 GM_MultiSolid (see ISO 19107:2003, 6.5.7) as specified in D.2.3.5. A `gml:MultiSolid` is defined by one or more `gml:AbstractSolids`.

The members of the geometric aggregate may be specified either using the "standard" property (`gml:solidMember`) or the array property (`gml:solidMembers`). It is also valid to use both the "standard" and the array properties in the same collection.

NOTE Array properties cannot reference remote geometry elements via XLinks.

```
<element name="solidMember" type="gml:SolidPropertyType" />
```

This property element either references a solid via the XLink-attributes or contains the solid element. A solid element is any element, which is substitutable for `gml:AbstractSolid`.

```
<element name="solidMembers" type="gml:SolidArrayPropertyType" />
```

This property element contains a list of solids. The order of the elements is significant and shall be preserved when processing the array.

11.3.5.2 MultiSolidPropertyType, multiSolidProperty

```
<complexType name="MultiSolidPropertyType">
    <sequence minOccurs="0">
```

```
            <element ref="gml:MultiSolid"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </complexType>
```

A property that has a collection of solids as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.

```
    <element name="multiSolidProperty" type="gml:MultiSolidPropertyType" />
```

This property element either references a solid aggregate via the XLink-attributes or contains the "multi solid" element. `gml:multiSolidProperty` is the predefined property, which may be used by GML application schemas whenever a GML feature has a property with a value that is substitutable for `gml:MultiSolid`.

12 GML schema — Coordinate reference systems schemas

12.1 Overview

12.1.1 Introduction

This clause describes the GML schema components for encoding the definitions of coordinate reference systems and coordinate operations, explaining their contents, structure, and dependencies.

12.1.2 Relationship with ISO 19111

The schema components of the GML schema specified in this clause provide a conformant, full implementation of the conceptual schema specified in ISO 19111. The relationship is discussed in detail in D.2.7. Additional components for temporal reference systems are specified in D.3.9.

The ISO 19111 types implemented in GML are specified in ISO 19111; some additional constraints are specified in ISO 19111 for these types, which are also constraints on the schema components of the GML schema.

NOTE The corresponding five schema documents are referenceSystems.xsd, coordinateReferenceSystems.xsd, datums.xsd, coordinateSystems.xsd, and coordinateOperations.xsd. These schema documents implement the UML package with a similar name in the conceptual model.

12.1.3 Important XML elements

These XML Schema components encode definition data for both Coordinate Reference Systems (CRSs) and Coordinate Operations (including coordinate Transformations and Conversions). This definition data includes identification and specification data, both included as needed. See ISO 19111 for the semantics of the schema components.

The specified XML encoding includes multiple alternative top-level XML elements that can be used where needed. (That is, there is not a single top-level element that may be the basis for all XML documents.) Most of these top-level XML elements are GML objects that include identification information, allowing it to be referenced. The alternative top-level XML elements include:

— All concrete XML elements in the substitution group headed by the abstract SingleCRS XML element. These elements may each be used to transfer the definition of one coordinate reference system of that type. These eight concrete XML elements are named:

    — CompoundCRS

  - GeodeticCRS
  - ProjectedCRS
  - EngineeringCRS
  - ImageCRS
  - VerticalCRS
  - TemporalCRS
  - DerivedCRS
- All concrete XML elements that are substitutable for the abstract CoordinateOperation XML element, namely:
  - ConcatenatedOperation
  - PassThroughOperation
  - Transformation
  - Conversion

The concrete XML elements that are substitutable for SingleCRS use multiple lower-level XML elements containing data structures. These lower-level elements include all five concrete elements that are substitutable for the abstract Datum XML element, named:

- GeodeticDatum
- VerticalDatum
- TemporalDatum
- EngineeringDatum
- ImageDatum

These lower-level XML elements also include all ten concrete elements that are substitutable for the abstract CoordinateSystem XML element, named:

- EllipsoidalCS
- VerticalCS
- CartesianCS
- AffineCS
- LinearCS
- PolarCS

— SphericalCS

— CylindricalCS

— TimeCS

— UserDefinedCS

The concrete XML elements that are substitutable for the CoordinateOperation element use multiple lower-level elements containing data structures, including the elements named:

— OperationMethod

— OperationParameter

— OperationParameterGroup

— ParameterValue

— ParameterValueGroup

12.2 Reference systems

12.2.1 Overview

The reference systems schema components have two logical parts, which define elements and types for XML encoding of the definitions of:

— Identified Object, inherited by the ten types of GML object used for coordinate reference systems and coordinate operations

— High-level part of the definitions of coordinate reference systems

This schema encodes the Identified Object and Reference System packages of the UML Model for ISO 19111.

NOTE The referenceSystems schema includes the dictionary.xsd GML schema document, and imports the metadataEntitySet.xsd schema document from ISO 19139. This schema document is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:referenceSystems:3.2.1

12.2.2 IdentifiedObjectType

```
<complexType name="IdentifiedObjectType" abstract="true">
    <complexContent>
        <extension base="gml:DefinitionType"/>
    </complexContent>
</complexType>
```

`gml:IdentifiedObjectType` provides identification properties of a CRS-related object. In `gml:DefinitionType`, the `gml:identifier` element shall be the primary name by which this object is identified, encoding the "name" attribute in the UML model.

Zero or more of the gml:name elements can be an unordered set of "identifiers", encoding the "identifier" attribute in the UML model. Each of these gml:name elements can reference elsewhere the object's defining information or be an identifier by which this object can be referenced.

Zero or more other gml:name elements can be an unordered set of "alias" alternative names by which this CRS related object is identified, encoding the "alias" attributes in the UML model. An object may have several aliases, typically used in different contexts. The context for an alias is indicated by the value of its (optional) codeSpace attribute.

Any needed version information shall be included in the codeSpace attribute of a gml:identifier and gml:name elements. In this use, the gml:remarks element in the gml:DefinitionType shall contain comments on or information about this object, including data source information.

12.2.3 Abstract coordinate reference system

12.2.3.1 AbstractCRS

```
<element name="AbstractCRS" type="gml:AbstractCRSType" abstract="true" substitutionGroup="gml:Definition"/>

<complexType name="AbstractCRSType" abstract="true">
	<complexContent>
		<extension base="gml:IdentifiedObjectType">
			<sequence>
				<element ref="gml:domainOfValidity" minOccurs="0" maxOccurs="unbounded"/>
				<element ref="gml:scope" maxOccurs="unbounded"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

gml:AbstractCRS specifies a coordinate reference system which is usually single but may be compound. This abstract complex type shall not be used, extended, or restricted, in a GML application schema, to define a concrete subtype with a meaning equivalent to a concrete subtype specified in this document.

12.2.3.2 domainOfValidity

```
<element name="domainOfValidity">
	<complexType>
		<sequence minOccurs="0">
			<element ref="gmd:EX_Extent"/>
		</sequence>
		<attributeGroup ref="gml:AssociationAttributeGroup"/>
	</complexType>
</element>
```

The gml:domainOfValidity property implements an association role to an EX_Extent object as encoded in ISO/TS 19139, either referencing or containing the definition of that extent.

12.2.3.3 scope

```
<element name="scope" type="string"/>
```

The gml:scope property provides a description of the usage, or limitations of usage, for which this CRS-related object is valid. If unknown, enter "not known".

12.2.3.4 CRSPropertyType

```
<complexType name="CRSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractCRS"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:CRSPropertyType` is a property type for association roles to a CRS abstract coordinate reference system, either referencing or containing the definition of that CRS.

12.3 Coordinate reference systems

12.3.1 Overview

The spatial-temporal coordinate reference systems schema components are divided into two logical parts. One part defines elements and types for XML encoding of abstract coordinate reference systems definitions. The larger part defines specialized constructs for XML encoding of definitions of the multiple concrete types of spatial-temporal coordinate reference system.

These schema components encode the Coordinate Reference System packages of the UML Models of ISO 19111:2007, Clause 8, and D.3.9 of this International Standard, with the exception of the abstract "SC_CRS" class.

NOTE The coordinateReferenceSystems schema document includes the coordinateSystems.xsd, datums.xsd, and coordinateOperations.xsd GML schema documents. This schema document is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:coordinateReferenceSystems:3.2.1

12.3.2 Abstract coordinate reference systems

12.3.2.1 AbstractSingleCRS

```
<element name="AbstractSingleCRS" type="gml:AbstractCRSType" abstract="true"
 substitutionGroup="gml:AbstractCRS"/>
```

`gml:AbstractSingleCRS` implements a coordinate reference system consisting of one coordinate system and one datum (as opposed to a Compound CRS).

12.3.2.2 SingleCRSPropertyType

```
<complexType name="SingleCRSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractSingleCRS"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:SingleCRSPropertyType` is a property type for association roles to a single coordinate reference system, either referencing or containing the definition of that coordinate reference system.

12.3.2.3 AbstractGeneralDerivedCRS

```
<element name="AbstractGeneralDerivedCRS" type="gml:AbstractGeneralDerivedCRSType" abstract="true"
substitutionGroup="gml:AbstractSingleCRS"/>
```

```
<complexType name="AbstractGeneralDerivedCRSType" abstract="true">
	<complexContent>
		<extension base="gml:AbstractCRSType">
			<sequence>
				<element ref="gml:conversion"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:AbstractGeneralDerivedCRS` is a coordinate reference system that is defined by its coordinate conversion from another coordinate reference system. This abstract complex type shall not be used, extended, or restricted, in a GML application schema, to define a concrete subtype with a meaning equivalent to a concrete subtype specified in this document.

12.3.2.4 Conversion

```
<element name="conversion" type="gml:GeneralConversionPropertyType"/>
```

`gml:conversion` is an association role to the coordinate conversion used to define the derived CRS.

12.3.3 Concrete coordinate reference systems

12.3.3.1 CompoundCRS

```
<element name="CompoundCRS" type="gml:CompoundCRSType" substitutionGroup="gml:AbstractCRS"/>

<complexType name="CompoundCRSType">
	<complexContent>
		<extension base="gml:AbstractCRSType">
			<sequence>
				<element ref="gml:componentReferenceSystem" minOccurs="2" maxOccurs="unbounded"/>
			</sequence>
			<attributeGroup ref="gml:AggregationAttributeGroup"/>
		</extension>
	</complexContent>
</complexType>
```

`gml:CompundCRS` is a coordinate reference system describing the position of points through two or more independent coordinate reference systems. It is associated with a non-repeating sequence of two or more instances of SingleCRS.

12.3.3.2 ComponentReferenceSystem

```
<element name="componentReferenceSystem" type="gml:SingleCRSPropertyType"/>
```

The `gml:componentReferenceSystem` elements are an ordered sequence of associations to all the component coordinate reference systems included in this compound coordinate reference system. The `gml:AggregationAttributeGroup` should be used to specify that the `gml:componentReferenceSystem` properties are ordered.

12.3.3.3 CompoundCRSPropertyType

```
<complexType name="CompoundCRSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:CompoundCRS"/>
	</sequence>
```

```
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:CompoundCRSPropertyType` is a property type for association roles to a compound coordinate reference system, either referencing or containing the definition of that reference system.

12.3.3.4 GeodeticCRS

```
<element name="GeodeticCRS" type="gml:GeodeticCRSType" substitutionGroup="gml:AbstractSingleCRS"/>

<complexType name="GeodeticCRSType">
        <complexContent>
                <extension base="gml:AbstractCRSType">
                        <sequence>
                                <choice>
                                        <element ref="gml:ellipsoidalCS"/>
                                        <element ref="gml:cartesianCS"/>
                                        <element ref="gml:sphericalCS"/>
                                </choice>
                                <element ref="gml:geodeticDatum"/>
                        </sequence>
                </extension>
        </complexContent>
</complexType>
```

`gml:GeodeticCRS` is a coordinate reference system based on a geodetic datum.

12.3.3.5 EllipsoidalCS

```
<element name="ellipsoidalCS" type="gml:EllipsoidalCSPropertyType"/>
```

`gml:ellipsoidalCS` is an association role to the ellipsoidal coordinate system used by this CRS.

12.3.3.6 cartesianCS

```
<element name="cartesianCS" type="gml:CartesianCSPropertyType"/>
```

`gml:cartesianCS` is an association role to the Cartesian coordinate system used by this CRS.

12.3.3.7 sphericalCS

```
<element name="sphericalCS" type="gml:SphericalCSPropertyType"/>
```

`gml:sphericalCS` is an association role to the spherical coordinate system used by this CRS.

12.3.3.8 geodeticDatum

```
<element name="geodeticDatum" type="gml:GeodeticDatumPropertyType"/>
```

`gml:geodeticDatum` is an association role to the geodetic datum used by this CRS.

12.3.3.9 GeodeticCRSPropertyType

```
<complexType name="GeodeticCRSPropertyType">
        <sequence minOccurs="0">
                <element ref="gml:GeodeticCRS"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
```

```
</complexType>
```

`gml:GeodeticCRSPropertyType` is a property type for association roles to a geodetic coordinate reference system, either referencing or containing the definition of that reference system.

12.3.3.10 VerticalCRS

```
<element name="VerticalCRS" type="gml:VerticalCRSType" substitutionGroup="gml:AbstractSingleCRS"/>

<complexType name="VerticalCRSType">
	<complexContent>
		<extension base="gml:AbstractCRSType">
			<sequence>
				<element ref="gml:verticalCS"/>
				<element ref="gml:verticalDatum"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:VerticalCRS` is a 1D coordinate reference system used for recording heights or depths. Vertical CRSs make use of the direction of gravity to define the concept of height or depth, but the relationship with gravity may not be straightforward. By implication, ellipsoidal heights (h) cannot be captured in a vertical coordinate reference system. Ellipsoidal heights cannot exist independently, but only as an inseparable part of a 3D coordinate tuple defined in a geographic 3D coordinate reference system.

12.3.3.11 verticalCS

```
<element name="verticalCS" type="gml:VerticalCSPropertyType"/>
```

`gml:verticalCS` is an association role to the vertical coordinate system used by this CRS.

12.3.3.12 verticalDatum

```
<element name="verticalDatum" type="gml:VerticalDatumPropertyType"/>
```

`gml:verticalDatum` is an association role to the vertical datum used by this CRS.

12.3.3.13 VerticalCRSPropertyType

```
<complexType name="VerticalCRSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:VerticalCRS"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:VerticalCRSPropertyType` is a property type for association roles to a vertical coordinate reference system, either referencing or containing the definition of that reference system.

12.3.3.14 ProjectedCRS

```
<element name="ProjectedCRS" type="gml:ProjectedCRSType"
substitutionGroup="gml:AbstractGeneralDerivedCRS"/>

<complexType name="ProjectedCRSType">
	<complexContent>
		<extension base="gml:AbstractGeneralDerivedCRSType">
```

```
                <sequence>
                    <choice>
                        <element ref="gml:baseGeodeticCRS"/>
                        <element ref="gml:baseGeographicCRS"/>
                    </choice>
                    <element ref="gml:cartesianCS"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

`gml:ProjectedCRS` is a 2D coordinate reference system used to approximate the shape of the Earth on a planar surface, but in such a way that the distortion that is inherent to the approximation is carefully controlled and known. Distortion correction is commonly applied to calculated bearings and distances to produce values that are a close match to actual field values.

12.3.3.15 baseGeodeticCRS

```
<element name="baseGeodeticCRS" type="gml:GeodeticCRSPropertyType"/>
```

`gml:baseGeodeticCRS` is an association role to the geodetic coordinate reference system used by this projected CRS.

12.3.3.16 ProjectedCRSPropertyType

```
<complexType name="ProjectedCRSPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:ProjectedCRS"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:ProjectedCRSPropertyType` is a property type for association roles to a projected coordinate reference system, either referencing or containing the definition of that reference system.

12.3.3.17 DerivedCRS

```
<element name="DerivedCRS" type="gml:DerivedCRSType" substitutionGroup="gml:AbstractGeneralDerivedCRS"/>

<complexType name="DerivedCRSType">
    <complexContent>
        <extension base="gml:AbstractGeneralDerivedCRSType">
            <sequence>
                <element ref="gml:baseCRS"/>
                <element ref="gml:derivedCRSType"/>
                <element ref="gml:coordinateSystem"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

`gml:DerivedCRS` is a single coordinate reference system that is defined by its coordinate conversion from another single coordinate reference system known as the base CRS. The base CRS can be a projected coordinate reference system, if this DerivedCRS is used for a georectified grid coverage as described in ISO 19123:2005, Clause 8.

12.3.3.18 baseCRS

```
<element name="baseCRS" type="gml:SingleCRSPropertyType"/>
```

`gml:baseCRS` is an association role to the coordinate reference system used by this derived CRS.

12.3.3.19 derivedCRSType

```
<element name="derivedCRSType" type="gml:CodeWithAuthorityType"/>
```

The `gml:derivedCRSType` property describes the type of a derived coordinate reference system. The required `codeSpace` attribute shall reference a source of information specifying the values and meanings of all the allowed string values for this property.

12.3.3.20 coordinateSystem

```
<element name="coordinateSystem" type="gml:CoordinateSystemPropertyType"/>
```

`gml:usesCS` is an association role to the coordinate system used by this CRS.

12.3.3.21 DerivedCRSPropertyType

```
<complexType name="DerivedCRSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:DerivedCRS"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:DerivedCRSPropertyType` is a property type for association roles to a non-projected derived coordinate reference system, either referencing or containing the definition of that reference system.

12.3.3.22 EngineeringCRS

```
<element name="EngineeringCRS" type="gml:EngineeringCRSType" substitutionGroup="gml:AbstractSingleCRS"/>

<complexType name="EngineeringCRSType">
	<complexContent>
		<extension base="gml:AbstractCRSType">
			<sequence>
				<choice>
					<element ref="gml:affineCS"/>
					<element ref="gml:cartesianCS"/>
					<element ref="gml:cylindricalCS"/>
					<element ref="gml:linearCS"/>
					<element ref="gml:polarCS"/>
					<element ref="gml:sphericalCS"/>
					<element ref="gml:userDefinedCS"/>
					<element ref="gml:coordinateSystem"/>
				</choice>
				<element ref="gml:engineeringDatum"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:EngineeringCRS` is a contextually local coordinate reference system which can be divided into two broad categories:

- Earth-fixed systems applied to engineering activities on or near the surface of the Earth;
- CRSs on moving platforms such as road vehicles, vessels, aircraft, or spacecraft, see ISO 19111:2007, 8.3.

12.3.3.23 cylindricalCS

```
<element name="cylindricalCS" type="gml:CylindricalCSPropertyType"/>
```

`gml:cylindricalCS` is an association role to the cylindrical coordinate system used by this CRS.

12.3.3.24 linearCS

```
<element name="linearCS" type="gml:LinearCSPropertyType"/>
```

`gml:linearCS` is an association role to the linear coordinate system used by this CRS.

12.3.3.25 polarCS

```
<element name="polarCS" type="gml:PolarCSPropertyType"/>
```

`gml:polarCS` is an association role to the polar coordinate system used by this CRS.

12.3.3.26 userDefinedCS

```
<element name="userDefinedCS" type="gml:UserDefinedCSPropertyType"/>
```

`gml:userDefinedCS` is an association role to the user defined coordinate system used by this CRS.

12.3.3.27 engineeringDatum

```
<element name="engineeringDatum" type="gml:EngineeringDatumPropertyType"/>
```

`gml:engineeringDatum` is an association role to the engineering datum used by this CRS.

12.3.3.28 EngineeringCRSPropertyType

```
<complexType name="EngineeringCRSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:EngineeringCRS"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:EngineeringCRSPropertyType` is a property type for association roles to an engineering coordinate reference system, either referencing or containing the definition of that reference system.

12.3.3.29 ImageCRS

```
<element name="ImageCRS" type="gml:ImageCRSType" substitutionGroup="gml:AbstractSingleCRS"/>

<complexType name="ImageCRSType">
	<complexContent>
		<extension base="gml:AbstractCRSType">
			<sequence>
				<choice>
					<element ref="gml:cartesianCS"/>
					<element ref="gml:affineCS"/>
```

```
                    <element ref="gml:usesObliqueCartesianCS"/>
                </choice>
                <element ref="gml:imageDatum"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

gml:ImageCRS is an engineering coordinate reference system applied to locations in images. Image coordinate reference systems are treated as a separate subtype because the definition of the associated image datum contains two attributes not relevant to other engineering datums.

12.3.3.30 affineCS

```
<element name="affineCS" type="gml:AffineCSPropertyType"/>
```

gml:affineCS is an association role to the affine coordinate system used by this CRS.

12.3.3.31 imageDatum

```
<element name="imageDatum" type="gml:ImageDatumPropertyType"/>
```

gml:imageDatum is an association role to the image datum used by this CRS.

12.3.3.32 ImageCRSPropertyType

```
<complexType name="ImageCRSPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:ImageCRS"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

gml:ImageCRSPropertyType is a property type for association roles to an image coordinate reference system, either referencing or containing the definition of that reference system.

12.3.3.33 TemporalCRS

```
<element name="TemporalCRS" type="gml:TemporalCRSType" substitutionGroup="gml:AbstractSingleCRS"/>

<complexType name="TemporalCRSType">
    <complexContent>
        <extension base="gml:AbstractCRSType">
            <sequence>
                <choice>
                    <element ref="gml:timeCS"/>
                    <element ref="gml:usesTemporalCS"/>
                </choice>
                <element ref="gml:temporalDatum"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

gml:TemporalCRS is a 1D coordinate reference system used for the recording of time.

12.3.3.34 timeCS

```
<element name="timeCS" type="gml:TimeCSPropertyType"/>
```

`gml:timeCS` is an association role to the time coordinate system used by this CRS.

12.3.3.35 temporalDatum

```
<element name="temporalDatum" type="gml:TemporalDatumPropertyType"/>
```

`gml:temporalDatum` is an association role to the temporal datum used by this CRS.

12.3.3.36 TemporalCRSPropertyType

```
<complexType name="TemporalCRSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:TemporalCRS"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:TemporalCRSPropertyType` is a property type for association roles to a temporal coordinate reference system, either referencing or containing the definition of that reference system.

12.4 Coordinate systems

12.4.1 Overview

The coordinate systems schema components can be divided into three logical parts, which define elements and types for XML encoding of the definitions of:

— coordinate system axes;

— abstract coordinate system;

— multiple concrete types of spatial-temporal coordinate system.

These schema components encode the Coordinate System packages of the UML Models of ISO 19111:2007, Clause 9, and D.3.9 of this International Standard.

NOTE The coordinateSystems schema document includes the referenceSystems.xsd GML schema document. This schema is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:coordinateSystems:3.2.1

12.4.2 Coordinate system axes

12.4.2.1 CoordinateSystemAxis

```
<element name="CoordinateSystemAxis" type="gml:CoordinateSystemAxisType"
substitutionGroup="gml:Definition"/>

<complexType name="CoordinateSystemAxisType">
	<complexContent>
		<extension base="gml:IdentifiedObjectType">
			<sequence>
				<element ref="gml:axisAbbrev"/>
```

```
                <element ref="gml:axisDirection"/>
                <element ref="gml:minimumValue" minOccurs="0"/>
                <element ref="gml:maximumValue" minOccurs="0"/>
                <element ref="gml:rangeMeaning" minOccurs="0"/>
            </sequence>
            <attribute name="uom" type="gml:UomIdentifier" use="required"/>
        </extension>
    </complexContent>
</complexType>
```

`gml:CoordinateSystemAxis` is a definition of a coordinate system axis.

12.4.2.2 axisAbbrev

```
<element name="axisAbbrev" type="gml:CodeType"/>
```

`gml:axisAbbrev` is the abbreviation used for this coordinate system axis; this abbreviation is also used to identify the coordinates in the coordinate tuple. The `codeSpace` attribute may reference a source of more information on a set of standardized abbreviations, or on this abbreviation.

EXAMPLE Typical abbreviations are “X” and “Y”.

12.4.2.3 axisDirection

```
<element name="axisDirection" type="gml:CodeWithAuthorityType"/>
```

`gml:axisDirection` is the direction of this coordinate system axis (or in the case of Cartesian projected coordinates, the direction of this coordinate system axis at the origin).

EXAMPLE Typical directions may be "north" or "south", "east" or "west", "up" or "down".

Within any set of coordinate system axes, only one of each pair of terms may be used. For Earth-fixed CRSs, this direction is often approximate and intended to provide a human interpretable meaning to the axis. When a geodetic datum is used, the precise directions of the axes may therefore vary slightly from this approximate direction.

NOTE A `gml:EngineeringCRS` often requires specific descriptions of the directions of its coordinate system axes.

The `codeSpace` attribute shall reference a source of information specifying the values and meanings of all the allowed string values for this property.

12.4.2.4 minimumValue, maximumValue, rangeMeaning

```
<element name="minimumValue" type="double"/>
<element name="maximumValue" type="double"/>
```

The `gml:minimumValue` and `gml:maximumValue` properties allow the specification of minimum and maximum value normally allowed for this axis, in the unit of measure for the axis. For a continuous angular axis such as longitude, the values wrap-around at this value. Also, values beyond this minimum/maximum can be used for specified purposes, such as in a bounding box. A value of minus infinity shall be allowed for the `gml:minimumValue` element, a value of plus infinity for the `gml:maximumValue` element. If these elements are omitted, the value is unspecified.

```
<element name="rangeMeaning" type="gml:CodeWithAuthorityType"/>
```

`gml:rangeMeaning` describes the meaning of axis value range specified by `gml:minimumValue` and `gml:maximumValue`. This element shall be omitted when both `gml:minimumValue` and `gml:maximumValue` are omitted. This element should be included when gml:minimumValue and/or gml:maximumValue are included. If this element is omitted when the gml:minimumValue and/or gml:maximumValue are included, the meaning is unspecified. The codeSpace attribute shall reference a source of information specifying the values and meanings of all the allowed string values for this property.

12.4.2.5 uom

The uom attribute provides an identifier of the unit of measure used for this coordinate system axis. The value of this coordinate in a coordinate tuple shall be recorded using this unit of measure, whenever those coordinates use a coordinate reference system that uses a coordinate system that uses this axis.

12.4.2.6 CoordinateSystemAxisPropertyType

```
<complexType name="CoordinateSystemAxisPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:CoordinateSystemAxis"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

gml:CoordinateSystemAxisPropertyType is a property type for association roles to a coordinate system axis, either referencing or containing the definition of that axis.

12.4.3 Abstract coordinate system

12.4.3.1 AbstractCoordinateSystem

```
<element name="AbstractCoordinateSystem" type="gml:AbstractCoordinateSystemType" abstract="true"
substitutionGroup="gml:Definition"/>

<complexType name="AbstractCoordinateSystemType" abstract="true">
	<complexContent>
		<extension base="gml:IdentifiedObjectType">
			<sequence>
				<element ref="gml:axis" maxOccurs="unbounded"/>
			</sequence>
			<attributeGroup ref="gml:AggregationAttributeGroup"/>
		</extension>
	</complexContent>
</complexType>
```

gml:AbstractCoordinateSystem is the non-repeating sequence of coordinate system axes that spans a given coordinate space. A CS is derived from a set of mathematical rules for specifying how coordinates in a given space are to be assigned to points. The coordinate values in a coordinate tuple shall be recorded in the order in which the coordinate system axes associations are recorded. This abstract complex type shall not be used, extended, or restricted, in an Application Schema, to define a concrete subtype with a meaning equivalent to a concrete subtype specified in this document.

12.4.3.2 axis

```
<element name="axis" type="gml:CoordinateSystemAxisPropertyType"/>
```

The gml:axis property is an association role (ordered sequence) to the coordinate system axes included in this coordinate system. The coordinate values in a coordinate tuple shall be recorded in the order in which the coordinate system axes associations are recorded, whenever those coordinates use a coordinate reference system that uses this coordinate system. The gml:AggregationAttributeGroup should be used to specify that the axis objects are ordered.

12.4.3.3 CoordinateSystemPropertyType

```
<complexType name="CoordinateSystemPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:AbstractCoordinateSystem"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:CoordinateSystemPropertyType` is a property type for association roles to a coordinate system, either referencing or containing the definition of that coordinate system.

12.4.4 Concrete coordinate systems

12.4.4.1 EllipsoidalCS

```
<element name="EllipsoidalCS" type="gml:EllipsoidalCSType" substitutionGroup="gml:AbstractCoordinateSystem"/>

<complexType name="EllipsoidalCSType">
    <complexContent>
        <extension base="gml:AbstractCoordinateSystemType"/>
    </complexContent>
</complexType>
```

`gml:EllipsoidalCS` is a two- or three-dimensional coordinate system in which position is specified by geodetic latitude, geodetic longitude, and (in the three-dimensional case) ellipsoidal height. An EllipsoidalCS shall have two or three `gml:axis` property elements; the number of associations shall equal the dimension of the CS.

12.4.4.2 EllipsoidalCSPropertyType

```
<complexType name="EllipsoidalCSPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:EllipsoidalCS"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:EllipsoidalCSPropertyType` is a property type for association roles to an ellipsoidal coordinate system, either referencing or containing the definition of that coordinate system.

12.4.4.3 CartesianCS

```
<element name="CartesianCS" type="gml:CartesianCSType" substitutionGroup="gml:AbstractCoordinateSystem"/>

<complexType name="CartesianCSType">
    <complexContent>
        <extension base="gml:AbstractCoordinateSystemType"/>
    </complexContent>
</complexType>
```

`gml:CartesianCS` is a 1-, 2-, or 3-dimensional coordinate system. In the 1-dimensional case, it contains a single straight coordinate axis. In the 2- and 3-dimensional cases gives the position of points relative to orthogonal straight axes. In the multi-dimensional case, all axes shall have the same length unit of measure. A CartesianCS shall have one, two, or three `gml:axis` property elements.

12.4.4.4 CartesianCSPropertyType

```
<complexType name="CartesianCSPropertyType">
```

```
        <sequence minOccurs="0">
            <element ref="gml:CartesianCS"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
```

`gml:CartesianCSPropertyType` is a property type for association roles to a Cartesian coordinate system, either referencing or containing the definition of that coordinate system.

12.4.4.5 VerticalCS

```
    <element name="VerticalCS" type="gml:VerticalCSType" substitutionGroup="gml:AbstractCoordinateSystem"/>

    <complexType name="VerticalCSType">
        <complexContent>
            <extension base="gml:AbstractCoordinateSystemType"/>
        </complexContent>
    </complexType>
```

`gml:VerticalCS` is a one-dimensional coordinate system used to record the heights or depths of points. Such a coordinate system is usually dependent on the Earth's gravitational field, perhaps loosely as when atmospheric pressure is the basis for the vertical coordinate system axis. A VerticalCS shall have one `gml:axis` property element.

12.4.4.6 VerticalCSPropertyType

```
    <complexType name="VerticalCSPropertyType">
        <sequence minOccurs="0">
            <element ref="gml:VerticalCS"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
```

`gml:VerticalCSPropertyType` is a property type for association roles to a vertical coordinate system, either referencing or containing the definition of that coordinate system.

12.4.4.7 TimeCS

```
    <element name="TimeCS" type="gml:TimeCSType" substitutionGroup="gml:AbstractCoordinateSystem"/>

    <complexType name="TimeCSType">
        <complexContent>
            <extension base="gml:AbstractCoordinateSystemType"/>
        </complexContent>
    </complexType>
```

`gml:TimeCS` is a one-dimensional coordinate system containing a time axis, used to describe the temporal position of a point in the specified time units from a specified time origin. A TimeCS shall have one `gml:axis` property element.

12.4.4.8 TimeCSPropertyType

```
    <complexType name="TimeCSPropertyType">
        <sequence minOccurs="0">
            <element ref="gml:TimeCS"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
```

gml:TimeCSPropertyType is a property type for association roles to a time coordinate system, either referencing or containing the definition of that coordinate system.

12.4.4.9 LinearCS

```
<element name="LinearCS" type="gml:LinearCSType" substitutionGroup="gml:AbstractCoordinateSystem"/>

<complexType name="LinearCSType">
	<complexContent>
		<extension base="gml:AbstractCoordinateSystemType"/>
	</complexContent>
</complexType>
```

gml:LinearCS is a one-dimensional coordinate system that consists of the points that lie on the single axis described. The associated coordinate is the distance — with or without offset — from the specified datum to the point along the axis. A LinearCS shall have one gml:axis property element.

EXAMPLE Usage of the line feature representing a pipeline to describe points on or along that pipeline.

NOTE gml:LinearCS can only be used for simple continuous linear systems. Linear Reference Systems (LRS), particularly as applied to the transportation industry, are specified in ISO 19133 and are not implemented by this International Standard.

12.4.4.10 LinearCSPropertyType

```
<complexType name="LinearCSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:LinearCS"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

gml:LinearCSPropertyType is a property type for association roles to a linear coordinate system, either referencing or containing the definition of that coordinate system.

12.4.4.11 UserDefinedCS

```
<element name="UserDefinedCS" type="gml:UserDefinedCSType"
substitutionGroup="gml:AbstractCoordinateSystem"/>

<complexType name="UserDefinedCSType">
	<complexContent>
		<extension base="gml:AbstractCoordinateSystemType"/>
	</complexContent>
</complexType>
```

gml:UserDefinedCS is a two- or three-dimensional coordinate system that consists of any combination of coordinate axes not covered by any other coordinate system type. A UserDefinedCS shall have two or three gml:axis property elements; the number of property elements shall equal the dimension of the CS.

EXAMPLE A multilinear coordinate system which contains one coordinate axis that may have any 1D shape which has no intersections with itself. This non-straight axis is supplemented by one or two straight axes to complete a 2 or 3 dimensional coordinate system. The non-straight axis is typically incrementally straight or curved.

12.4.4.12 UserDefinedCSPropertyType

```
<complexType name="UserDefinedCSPropertyType">
```

```
        <sequence minOccurs="0">
            <element ref="gml:UserDefinedCS"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
```

`gml:UserDefinedCSPropertyType` is a property type for association roles to a user-defined coordinate system, either referencing or containing the definition of that coordinate system.

12.4.4.13 SphericalCS

```
<element name="SphericalCS" type="gml:SphericalCSType" substitutionGroup="gml:AbstractCoordinateSystem"/>

<complexType name="SphericalCSType">
    <complexContent>
        <extension base="gml:AbstractCoordinateSystemType"/>
    </complexContent>
</complexType>
```

`gml:SphericalCS` is a three-dimensional coordinate system with one distance measured from the origin and two angular coordinates.

NOTE It should not be confused with an ellipsoidal coordinate system based on an ellipsoid "degenerated" into a sphere.

A SphericalCS shall have three `gml:axis` property elements.

12.4.4.14 SphericalCSPropertyType

```
<complexType name="SphericalCSPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:SphericalCS"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:SphericalCSPropertyType` is property type for association roles to a spherical coordinate system, either referencing or containing the definition of that coordinate system.

12.4.4.15 PolarCS

```
<element name="PolarCS" type="gml:PolarCSType" substitutionGroup="gml:AbstractCoordinateSystem"/>

<complexType name="PolarCSType">
    <complexContent>
        <extension base="gml:AbstractCoordinateSystemType"/>
    </complexContent>
</complexType>
```

`gml:PolarCS` is a two-dimensional coordinate system in which position is specified by the distance from the origin and the angle between the line from the origin to a point and a reference direction. A PolarCS shall have two `gml:axis` property elements.

12.4.4.16 PolarCSPropertyType

```
<complexType name="PolarCSPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:PolarCS"/>
    </sequence>
```

```
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

gml:PolarCSPropertyType is a property type for association roles to a polar coordinate system, either referencing or containing the definition of that coordinate system.

12.4.4.17 CylindricalCS

```
<element name="CylindricalCS" type="gml:CylindricalCSType" substitutionGroup="gml:AbstractCoordinateSystem"/>

<complexType name="CylindricalCSType">
    <complexContent>
        <extension base="gml:AbstractCoordinateSystemType"/>
    </complexContent>
</complexType>
```

gml:CylindricalCS is a three-dimensional coordinate system consisting of a polar coordinate system extended by a straight coordinate axis perpendicular to the plane spanned by the polar coordinate system. A CylindricalCS shall have three gml:axis property elements.

12.4.4.18 CylindricalCSPropertyType

```
<complexType name="CylindricalCSPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:CylindricalCS"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

gml:CylindricalCSPropertyType is a property type for association roles to a cylindrical coordinate system, either referencing or containing the definition of that coordinate system.

12.4.4.19 AffineCS

```
<element name="AffineCS" type="gml:AffineCSType" substitutionGroup="gml:AbstractCoordinateSystem"/>

<complexType name="AffineCSType">
    <complexContent>
        <extension base="gml:AbstractCoordinateSystemType"/>
    </complexContent>
</complexType>
```

gml:AffineCS is a two- or three-dimensional coordinate system with straight axes that are not necessarily orthogonal. An AffineCS shall have two or three gml:axis property elements; the number of property elements shall equal the dimension of the CS.

12.4.4.20 AffineCSPropertyType

```
<complexType name="AffineCSPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:AffineCS"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

gml:AffineCSPropertyType is a property type for association roles to an affine coordinate system, either referencing or containing the definition of that coordinate system.

12.5 Datums

12.5.1 Overview

The datums schema components can be divided into three logical parts, which define elements and types for XML encoding of the definitions of:

— abstract datum;

— geodetic datums, including ellipsoid and prime meridian;

— multiple other concrete types of spatial or temporal datum.

These schema components encode the Datum packages of the UML Models of ISO 19111:2007, Clause 10, and D.3.9 of this International Standard.

NOTE The datums schema document includes the referenceSystems.xsd GML schema. This schema is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:datums:3.2.1

12.5.2 Abstract datum

12.5.2.1 AbstractDatum

```
<element name="AbstractDatum" type="gml:AbstractDatumType" abstract="true"
substitutionGroup="gml:Definition"/>

<complexType name="AbstractDatumType" abstract="true">
	<complexContent>
		<extension base="gml:IdentifiedObjectType">
			<sequence>
				<element ref="gml:domainOfValidity" minOccurs="0"/>
				<element ref="gml:scope" maxOccurs="unbounded"/>
				<element ref="gml:anchorDefinition" minOccurs="0"/>
				<element ref="gml:realizationEpoch" minOccurs="0"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

A `gml:AbstractDatum` specifies the relationship of a coordinate system to the Earth, thus creating a coordinate reference system. A datum uses a parameter or set of parameters that determine the location of the origin of the coordinate reference system. Each datum subtype may be associated with only specific types of coordinate system. This abstract complex type shall not be used, extended, or restricted, in a GML application schema, to define a concrete subtype with a meaning equivalent to a concrete subtype specified in this document.

12.5.2.2 anchorDefinition

```
<element name="anchorDefinition" type="gml:CodeType"/>
```

`gml:anchorDefinition` is a description, possibly including coordinates, of the definition used to anchor the datum to the Earth, also known as the “origin”, especially for engineering and image datums. The codeSpace attribute may be used to reference a source of more detailed on this point or surface, or on a set of such descriptions.

— For a geodetic datum, this point is also known as the fundamental point, which is traditionally the point where the relationship between geoid and ellipsoid is defined. In some cases, the "fundamental point" may consist of a number of points. In those cases, the parameters defining the geoid/ellipsoid relationship have been averaged for these points, and the averages adopted as the datum definition.

— For an engineering datum, the anchor definition may be a physical point, or it may be a point with defined coordinates in another CRS.

— For an image datum, the anchor definition is usually either the centre of the image or the corner of the image.

— For a temporal datum, this attribute is not defined. Instead of the anchor definition, a temporal datum carries a separate time origin of type DateTime.

12.5.2.3 realizationEpoch

```
<element name="realizationEpoch" type="date"/>
```

`gml:realizationEpoch` is the time after which this datum definition is valid. See ISO 19111:2007, Table 33, for details.

12.5.2.4 DatumPropertyType

```
<complexType name="DatumPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractDatum"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:DatumPropertyType` is a property type for association roles to a datum, either referencing or containing the definition of that datum.

12.5.3 Geodetic datum

12.5.3.1 GeodeticDatum

```
<element name="GeodeticDatum" type="gml:GeodeticDatumType" substitutionGroup="gml:AbstractDatum"/>

<complexType name="GeodeticDatumType">
	<complexContent>
		<extension base="gml:AbstractDatumType">
			<sequence>
				<element ref="gml:primeMeridian"/>
				<element ref="gml:ellipsoid"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:GeodeticDatum` is a geodetic datum defines the precise location and orientation in 3-dimensional space of a defined ellipsoid (or sphere), or of a Cartesian coordinate system centred in this ellipsoid (or sphere).

12.5.3.2 primeMeridian

```
<element name="primeMeridian" type="gml:PrimeMeridianPropertyType"/>
```

`gml:primeMeridian` is an association role to the prime meridian used by this geodetic datum.

12.5.3.3 ellipsoid

```
<element name="ellipsoid" type="gml:EllipsoidPropertyType"/>
```

`gml:ellipsoid` is an association role to the ellipsoid used by this geodetic datum.

12.5.3.4 GeodeticDatumPropertyType

```
<complexType name="GeodeticDatumPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:GeodeticDatum"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:GeodeticDatumPropertyType` is a property type for association roles to a geodetic datum, either referencing or containing the definition of that datum.

12.5.3.5 Ellipsoid, semiMajorAxis, secondDefiningParameter

```
<element name="Ellipsoid" type="gml:EllipsoidType" substitutionGroup="gml:Definition"/>

<complexType name="EllipsoidType">
	<complexContent>
		<extension base="gml:IdentifiedObjectType">
			<sequence>
				<element ref="gml:semiMajorAxis"/>
				<element ref="gml:secondDefiningParameter"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

A `gml:Ellipsoid` is a geometric figure that may be used to describe the approximate shape of the Earth. In mathematical terms, it is a surface formed by the rotation of an ellipse about its minor axis.

```
<element name="semiMajorAxis" type="gml:MeasureType"/>
```

`gml:semiMajorAxis` specifies the length of the semi-major axis of the ellipsoid, with its units. Uses the MeasureType with the restriction that the unit of measure referenced by uom must be suitable for a length, such as metres or feet.

```
<element name="secondDefiningParameter">
	<complexType>
		<sequence>
			<element ref="gml:SecondDefiningParameter"/>
		</sequence>
	</complexType>
</element>

<element name="SecondDefiningParameter">
	<complexType>
		<choice>
			<element name="inverseFlattening" type="gml:MeasureType"/>
			<element name="semiMinorAxis" type="gml:LengthType"/>
			<element name="isSphere" type="boolean" default="true"/>
		</choice>
	</complexType>
</element>
```

gml:secondDefiningParameter is a property containing the definition of the second parameter that defines the shape of an ellipsoid. An ellipsoid requires two defining parameters: semi-major axis and inverse flattening or semi-major axis and semi-minor axis. When the reference body is a sphere rather than an ellipsoid, only a single defining parameter is required, namely the radius of the sphere; in that case, the semi-major axis "degenerates" into the radius of the sphere.

The gml:inverseFlattening element contains the inverse flattening value of the ellipsoid. This value is a scale factor (or ratio). It uses gml:LengthType with the restriction that the unit of measure referenced by the uom attribute must be suitable for a scale factor, such as percent, permil, or parts-per-million.

The gml:semiMinorAxis element contains the length of the semi-minor axis of the ellipsoid. When the gml:isSphere element is included, the ellipsoid is degenerate and is actually a sphere. The sphere is completely defined by the semi-major axis, which is the radius of the sphere.

12.5.3.6 EllipsoidPropertyType

```
<complexType name="EllipsoidPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:Ellipsoid"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

gml:EllipsoidPropertyType is a property type for association roles to an ellipsoid, either referencing or containing the definition of that ellipsoid.

12.5.3.7 PrimeMeridian, greenwichLongitude

```
<element name="PrimeMeridian" type="gml:PrimeMeridianType" substitutionGroup="gml:Definition"/>

<complexType name="PrimeMeridianType">
	<complexContent>
		<extension base="gml:IdentifiedObjectType">
			<sequence>
				<element ref="gml:greenwichLongitude"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

A gml:PrimeMeridian defines the origin from which longitude values are determined. The default value for the prime meridian gml:identifier value is "Greenwich".

```
<element name="greenwichLongitude" type="gml:AngleType"/>
```

gml:greenwichLongitude is the longitude of the prime meridian measured from the Greenwich meridian, positive eastward. If the value of the prime meridian “name” is "Greenwich" then the value of greenwichLongitude shall be 0 degrees. The property uses gml:AngleType.

12.5.3.8 PrimeMeridianPropertyType

```
<complexType name="PrimeMeridianPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:PrimeMeridian"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
```

```
</complexType>
```

`gml:PrimeMeridianPropertyType` is a property type for association roles to a prime meridian, either referencing or containing the definition of that meridian.

12.5.4 Other concrete datums

12.5.4.1 EngineeringDatum

```
<element name="EngineeringDatum" type="gml:EngineeringDatumType" substitutionGroup="gml:AbstractDatum"/>

<complexType name="EngineeringDatumType">
	<complexContent>
		<extension base="gml:AbstractDatumType"/>
	</complexContent>
</complexType>
```

`gml:EngineeringDatum` defines the origin of an engineering coordinate reference system, and is used in a region around that origin. This origin may be fixed with respect to the Earth (such as a defined point at a construction site), or be a defined point on a moving vehicle (such as on a ship or satellite).

12.5.4.2 EngineeringDatumPropertyType

```
<complexType name="EngineeringDatumPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:EngineeringDatum"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:EngineeringDatumPropertyType` is a property type for association roles to an engineering datum, either referencing or containing the definition of that datum.

12.5.4.3 ImageDatum

```
<element name="ImageDatum" type="gml:ImageDatumType" substitutionGroup="gml:AbstractDatum"/>

<complexType name="ImageDatumType">
	<complexContent>
		<extension base="gml:AbstractDatumType">
			<sequence>
				<element ref="gml:pixelInCell"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:ImageDatum` defines the origin of an image coordinate reference system, and is used in a local context only. For an image datum, the anchor definition is usually either the centre of the image or the corner of the image. For more information, see ISO 19111:2007, B.3.5.

12.5.4.4 pixelInCell

```
<element name="pixelInCell" type="gml:CodeWithAuthorityType"/>
```

`gml:pixelInCell` is a specification of the way an image grid is associated with the image data attributes. The required `codeSpace` attribute shall reference a source of information specifying the values and meanings of all the allowed string values for this property.

12.5.4.5 ImageDatumPropertyType

```
<complexType name="ImageDatumPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:ImageDatum"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:ImageDatumPropertyType` is a property type for association roles to an image datum, either referencing or containing the definition of that datum.

12.5.4.6 VerticalDatum

```
<element name="VerticalDatum" type="gml:VerticalDatumType" substitutionGroup="gml:AbstractDatum"/>

<complexType name="VerticalDatumType">
    <complexContent>
        <extension base="gml:AbstractDatumType"/>
    </complexContent>
</complexType>
```

`gml:VerticalDatum` is a textual description and/or a set of parameters identifying a particular reference level surface used as a zero-height surface, including its position with respect to the Earth for any of the height types recognized by this International Standard.

12.5.4.7 VerticalDatumPropertyType

```
<complexType name="VerticalDatumPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:VerticalDatum"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:VerticalDatumPropertyType` is property type for association roles to a vertical datum, either referencing or containing the definition of that datum.

12.5.4.8 TemporalDatum, origin

```
<element name="TemporalDatum" type="gml:TemporalDatumType" substitutionGroup="gml:AbstractDatum"/>

<complexType name="TemporalDatumType">
    <complexContent>
        <extension base="gml:TemporalDatumBaseType">
            <sequence>
                <element ref="gml:origin"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="TemporalDatumBaseType" abstract="true">
    <complexContent>
        <restriction base="gml:AbstractDatumType">
            <sequence>
                <element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="gml:description" minOccurs="0"/>
                <element ref="gml:descriptionReference" minOccurs="0"/>
```

```
                <element ref="gml:identifier"/>
                <element ref="gml:name" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="gml:remarks" minOccurs="0"/>
                <element ref="gml:domainOfValidity" minOccurs="0"/>
                <element ref="gml:scope" maxOccurs="unbounded"/>
            </sequence>
            <attribute ref="gml:id" use="required"/>
        </restriction>
    </complexContent>
</complexType>
```

A `gml:TemporalDatum` defines the origin of a Temporal Reference System. This type omits the "anchorDefinition" and "realizationEpoch" elements and adds the "origin" element with the dateTime type.

The TemporalDatumBaseType partially defines the origin of a temporal coordinate reference system. This type restricts the AbstractDatumType to remove the "anchorDefinition" and "realizationEpoch" elements.

```
<element name="origin" type="dateTime"/>
```

`gml:origin` is the date and time origin of this temporal datum.

The metaDataProperty element has been deprecated, and the gml:description element has been partially deprecated.

12.5.4.9 TemporalDatumPropertyType

```
<complexType name="TemporalDatumPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:TemporalDatum"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:TemporalDatumPropertyType` is a property type for association roles to a temporal datum, either referencing or containing the definition of that datum.

12.6 Coordinate operations

12.6.1 Overview

The spatial or temporal coordinate operations schema components can be divided into five logical parts, which define elements and types for XML encoding of the definitions of:

— Multiple abstract coordinate operations

— Multiple concrete types of coordinate operation, including Transformations and Conversions

— Abstract and concrete parameter values and groups

— Operation methods

— Abstract and concrete operation parameters and groups

These schema components encode the Coordinate Operation package of the UML Model for ISO 19111:2007, Clause 11.

NOTE The coordinateOperations schema includes the coordinateOperations.xsd GML schema document. This schema document is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:coordinateOperations:3.2.1

12.6.2 Abstract coordinate operations

12.6.2.1 AbstractCoordinateOperation

```
<element name="AbstractCoordinateOperation" type="gml:AbstractCoordinateOperationType" abstract="true"
substitutionGroup="gml:Definition"/>

<complexType name="AbstractCoordinateOperationType" abstract="true">
	<complexContent>
		<extension base="gml:IdentifiedObjectType">
			<sequence>
				<element ref="gml:domainOfValidity" minOccurs="0"/>
				<element ref="gml:scope" maxOccurs="unbounded" />
				<element ref="gml:operationVersion" minOccurs="0"/>
				<element ref="gml:coordinateOperationAccuracy" minOccurs="0"
					maxOccurs="unbounded"/>
				<element ref="gml:sourceCRS" minOccurs="0"/>
				<element ref="gml:targetCRS" minOccurs="0"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:AbstractCoordinateOperation` is a mathematical operation on coordinates that transforms or converts coordinates to another coordinate reference system. Many but not all coordinate operations (from CRS A to CRS B) also uniquely define the inverse operation (from CRS B to CRS A). In some cases, the operation method algorithm for the inverse operation is the same as for the forward algorithm, but the signs of some operation parameter values shall be reversed. In other cases, different algorithms are required for the forward and inverse operations, but the same operation parameter values are used. If (some) entirely different parameter values are needed, a different coordinate operation shall be defined.

The optional `gml:coordinateOperationAccuracy` property elements provide estimates of the impact of this coordinate operation on point position accuracy.

12.6.2.2 operationVersion

```
<element name="operationVersion" type="string"/>
```

`gml:operationVersion` is the version of the coordinate transformation (i.e., instantiation due to the stochastic nature of the parameters). Mandatory when describing a transformation, and should not be supplied for a conversion.

12.6.2.3 coordinateOperationAccuracy

```
<element name="coordinateOperationAccuracy">
	<complexType>
		<sequence minOccurs="0">
			<element ref="gmd:AbstractDQ_PositionalAccuracy"/>
		</sequence>
		<attributeGroup ref="gml:AssociationAttributeGroup"/>
	</complexType>
</element>
```

`gml:coordinateOperationAccuracy` is an association role to a DQ_PositionalAccuracy object as encoded in ISO/TS 19139, either referencing or containing the definition of that positional accuracy. That object contains an estimate of the impact of this coordinate operation on point positional accuracy. That is, it gives position error estimates for the target coordinates of this coordinate operation, assuming no errors in the source coordinates.

12.6.2.4 sourceCRS

```
<element name="sourceCRS" type="gml:CRSPropertyType"/>
```

`gml:sourceCRS` is an association role to the source CRS (coordinate reference system) of this coordinate operation.

12.6.2.5 targetCRS

```
<element name="targetCRS" type="gml:CRSPropertyType"/>
```

`gml:targetCRS` is an association role to the target CRS (coordinate reference system) of this coordinate operation.

12.6.2.6 CoordinateOperationPropertyType

```
<complexType name="CoordinateOperationPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractCoordinateOperation"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:CoordinateOperationPropertyType` is a property type for association roles to a coordinate operation, either referencing or containing the definition of that coordinate operation.

12.6.2.7 AbstractSingleOperation

```
<element name="AbstractSingleOperation" type="gml:AbstractCoordinateOperationType" abstract="true"
substitutionGroup="gml:AbstractCoordinateOperation"/>
```

`gml:AbstractSingleOperation` is a single (not concatenated) coordinate operation.

12.6.2.8 SingleOperationPropertyType

```
<complexType name="SingleOperationPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractSingleOperation"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:SingleOperationPropertyType` is a property type for association roles to a single operation, either referencing or containing the definition of that single operation.

12.6.2.9 AbstractGeneralConversion

```
<element name="AbstractGeneralConversion" type="gml:AbstractGeneralConversionType" abstract="true"
 substitutionGroup="gml:AbstractOperation"/>

<complexType name="AbstractGeneralConversionType" abstract="true">
	<complexContent>
```

```
            <restriction base="gml:AbstractCoordinateOperationType">
                <sequence>
                    <element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
                    <element ref="gml:description" minOccurs="0"/>
                    <element ref="gml:descriptionReference" minOccurs="0"/>
                    <element ref="gml:identifier"/>
                    <element ref="gml:name" minOccurs="0" maxOccurs="unbounded"/>
                    <element ref="gml:remarks" minOccurs="0"/>
                    <element ref="gml:domainOfValidity" minOccurs="0"/>
                    <element ref="gml:scope" maxOccurs="unbounded"/>
                    <element ref="gml:coordinateOperationAccuracy" minOccurs="0"
                        maxOccurs="unbounded"/>
                </sequence>
                <attribute ref="gml:id" use="required"/>
            </restriction>
        </complexContent>
    </complexType>
```

`gm:AbstractGeneralConversion` is an abstract operation on coordinates that does not include any change of datum. The best-known example of a coordinate conversion is a map projection. The parameters describing coordinate conversions are defined rather than empirically derived. Note that some conversions have no parameters. The `gml:operationVersion`, `gml:sourceCRS`, and `gml:targetCRS` elements are omitted in a coordinate conversion.

This abstract complex type is expected to be extended for well-known operation methods with many Conversion instances, in GML application schemas that define operation-method-specialized element names and contents. This conversion uses an operation method, usually with associated parameter values. However, operation methods and parameter values are directly associated with concrete subtypes, not with this abstract type. All concrete types derived from this type shall extend this type to include a "usesMethod" element that references the "OperationMethod" element. Similarly, all concrete types derived from this type shall extend this type to include zero or more elements each named "uses...Value" that each use the type of an element substitutable for the "AbstractGeneralParameterValue" element.

The metaDataProperty element has been deprecated, and the gml:description element has been partially deprecated.

12.6.2.10 GeneralConversionPropertyType

```
    <complexType name="GeneralConversionPropertyType">
        <sequence minOccurs="0">
            <element ref="gml:AbstractGeneralConversion"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
```

`gml:GeneralConversionPropertyType` is a property type for association roles to a general conversion, either referencing or containing the definition of that conversion.

12.6.2.11 AbstractGeneralTransformation

```
    <element name="AbstractGeneralTransformation" type="gml:AbstractGeneralTransformationType" abstract="true"
     substitutionGroup="gml:AbstractOperation"/>

    <complexType name="AbstractGeneralTransformationType" abstract="true">
        <complexContent>
            <restriction base="gml:AbstractCoordinateOperationType">
                <sequence>
                    <element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
                    <element ref="gml:description" minOccurs="0"/>
```



```
                <element ref="gml:descriptionReference" minOccurs="0"/>
                <element ref="gml:identifier"/>
                <element ref="gml:name" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="gml:remarks" minOccurs="0"/>
                <element ref="gml:domainOfValidity" minOccurs="0"/>
                <element ref="gml:scope" maxOccurs="unbounded"/>
                <element ref="gml:operationVersion"/>
                <element ref="gml:coordinateOperationAccuracy" minOccurs="0"
                    maxOccurs="unbounded"/>
                <element ref="gml:sourceCRS"/>
                <element ref="gml:targetCRS"/>
            </sequence>
            <attribute ref="gml:id" use="required"/>
        </restriction>
    </complexContent>
</complexType>
```

`gml:AbstractGeneralTransformation` is an abstract operation on coordinates that usually includes a change of Datum. The parameters of a coordinate transformation are empirically derived from data containing the coordinates of a series of points in both coordinate reference systems. This computational process is usually "over-determined", allowing derivation of error (or accuracy) estimates for the transformation. Also, the stochastic nature of the parameters may result in multiple (different) versions of the same coordinate transformation. The `gml:operationVersion`, `gml:sourceCRS`, and `gml:targetCRS` property elements are mandatory in a coordinate transformation.

This abstract complex type is expected to be extended for well-known operation methods with many Transformation instances, in Application Schemas that define operation-method-specialized value element names and contents. This transformation uses an operation method with associated parameter values. However, operation methods and parameter values are directly associated with concrete subtypes, not with this abstract type. All concrete types derived from this type shall extend this type to include a "usesMethod" element that references one "OperationMethod" element. Similarly, all concrete types derived from this type shall extend this type to include one or more elements each named "uses...Value" that each use the type of an element substitutable for the "AbstractGeneralParameterValue" element.

The metaDataProperty element has been deprecated, and the gml:description element has been partially deprecated.

12.6.2.12 GeneralTransformationPropertyType

```
<complexType name="GeneralTransformationPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:AbstractGeneralTransformation"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:GeneralTransformationPropertyType` is a property type for association roles to a general transformation, either referencing or containing the definition of that transformation.

12.6.3 Concrete coordinate operations

12.6.3.1 ConcatenatedOperation

```
<element name="ConcatenatedOperation" type="gml:ConcatenatedOperationType"
 substitutionGroup="gml:AbstractCoordinateOperation"/>

<complexType name="ConcatenatedOperationType">
    <complexContent>
```

```
            <extension base="gml:AbstractCoordinateOperationType">
                <sequence>
                    <element ref="gml:coordOperation" minOccurs="2" maxOccurs="unbounded"/>
                </sequence>
                <attributeGroup ref="gml:AggregationAttributeGroup"/>
            </extension>
        </complexContent>
    </complexType>
```

`gml:ConcatenatedOperation` is an ordered sequence of two or more coordinate operations. This sequence of operations is constrained by the requirement that the source coordinate reference system of step (n+1) must be the same as the target coordinate reference system of step (n). The source coordinate reference system of the first step and the target coordinate reference system of the last step are the source and target coordinate reference system associated with the concatenated operation. Instead of a forward operation, an inverse operation may be used for one or more of the operation steps mentioned above, if the inverse operation is uniquely defined by the forward operation.

The `gml:coordOperation` property elements are an ordered sequence of associations to the two or more operations used by this concatenated operation. The `gml:AggregationAttributeGroup` should be used to specify that the `gml:coordOperation` associations are ordered.

12.6.3.2 CoordOperation

```
    <element name="coordOperation" type="gml:CoordinateOperationPropertyType"/>
```

`gml:coordOperation` is an association role to a coordinate operation.

12.6.3.3 ConcatenatedOperationPropertyType

```
    <complexType name="ConcatenatedOperationPropertyType">
        <sequence minOccurs="0">
            <element ref="gml:ConcatenatedOperation"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
```

`gml:ConcatenatedOperationPropertyType` is a property type for association roles to a concatenated operation, either referencing or containing the definition of that concatenated operation.

12.6.3.4 PassThroughOperation

```
    <element name="PassThroughOperation" type="gml:PassThroughOperationType"
    substitutionGroup="gml:AbstractSingleOperation"/>

    <complexType name="PassThroughOperationType">
        <complexContent>
            <extension base="gml:AbstractCoordinateOperationType">
                <sequence>
                    <element ref="gml:modifiedCoordinate" maxOccurs="unbounded"/>
                    <element ref="gml:coordOperation"/>
                </sequence>
                <attributeGroup ref="gml:AggregationAttributeGroup"/>
            </extension>
        </complexContent>
    </complexType>
```

`gml:PassThroughOperation` specifies that a subset of a coordinate tuple is subject to a specific coordinate operation.

The `gml:modifiedCoordinate` property elements are an ordered sequence of positive integers defining the positions in a coordinate tuple of the coordinates affected by this pass-through operation. The `gml:AggregationAttributeGroup` should be used to specify that the `gml:modifiedCoordinate` elements are ordered.

12.6.3.5 modifiedCoordinate

```
<element name="modifiedCoordinate" type="positiveInteger"/>
```

`gml:modifiedCoordinate` is a positive integer defining a position in a coordinate tuple.

12.6.3.6 PassThroughOperationPropertyType

```
<complexType name="PassThroughOperationPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:PassThroughOperation"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:PassThroughOperationPropertyType` is a property type for association roles to a pass through operation, either referencing or containing the definition of that pass through operation.

12.6.3.7 Conversion

```
<element name="Conversion" type="gml:ConversionType" substitutionGroup="gml:AbstractGeneralConversion"/>

<complexType name="ConversionType">
	<complexContent>
		<extension base="gml:AbstractGeneralConversionType">
			<sequence>
				<element ref="gml:method"/>
				<element ref="gml:parameterValue" minOccurs="0" maxOccurs="unbounded"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:Conversion` is a concrete operation on coordinates that does not include any change of Datum. The best-known example of a coordinate conversion is a map projection. The parameters describing coordinate conversions are defined rather than empirically derived. Note that some conversions have no parameters.

This concrete complex type can be used without using a GML application schema that defines operation-method-specialized element names and contents, especially for methods with only one Conversion instance.

12.6.3.8 method

```
<element name="method" type="gml:OperationMethodPropertyType"/>
```

`gml:method` is an association role to the operation method used by a coordinate operation.

12.6.3.9 parameterValue

```
<element name="parameterValue" type="gml:AbstractGeneralParameterValuePropertyType"/>
```

`gml:parameterValue` is a composition association to a parameter value or group of parameter values used by a coordinate operation.

12.6.3.10 ConversionPropertyType

```
<complexType name="ConversionPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:Conversion"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:ConversionPropertyType` is a property type for association roles to a concrete general-purpose conversion, either referencing or containing the definition of that conversion.

12.6.3.11 Transformation

```
<element name="Transformation" type="gml:TransformationType"
substitutionGroup="gml:AbstractGeneralTransformation"/>

<complexType name="TransformationType">
	<complexContent>
		<extension base="gml:AbstractGeneralTransformationType">
			<sequence>
				<element ref="gml:method"/>
				<element ref="gml:parameterValue" minOccurs="0" maxOccurs="unbounded"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:Transformation` is a concrete object element derived from `gml:AbstractGeneralTransformation` (12.6.2.11).

This concrete object can be used for all operation methods, without using a GML application schema that defines operation-method-specialized element names and contents, especially for methods with only one Transformation instance.

The `gml:parameterValue` elements are an unordered list of composition associations to the set of parameter values used by this conversion operation.

12.6.3.12 TransformationPropertyType

```
<complexType name="TransformationPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:Transformation"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:TransformationPropertyType` is a property type for association roles to a transformation, either referencing or containing the definition of that transformation.

12.6.4 Parameter values and groups

12.6.4.1 AbstractGeneralParameterValue

```
<element name="AbstractGeneralParameterValue" type="gml:AbstractGeneralParameterValueType" abstract="true"
substitutionGroup="gml:AbstractObject"/>


<complexType name="AbstractGeneralParameterValueType" abstract="true">
```



```
        <sequence/>
    </complexType>
```

`gml:AbstractGeneralParameterValue` is an abstract parameter value or group of parameter values.

This abstract complexType is expected to be extended and restricted for well-known operation methods with many instances, in Application Schemas that define operation-method-specialized element names and contents. Specific parameter value elements are directly contained in concrete subtypes, not in this abstract type. All concrete types derived from this type shall extend this type to include one "...Value" element with an appropriate type, which should be one of the element types allowed in the ParameterValueType. In addition, all derived concrete types shall extend this type to include a "operationParameter" property element that references one element substitutable for the "OperationParameter" object element.

12.6.4.2 AbstractGeneralParameterValuePropertyType

```
<complexType name="AbstractGeneralParameterValuePropertyType">
    <sequence>
        <element ref="gml:AbstractGeneralParameterValue" />
    </sequence>
</complexType>
```

`gml:AbstractGeneralParameterValuePropertyType` is a property type for inline association roles to a parameter value or group of parameter values, always containing the values.

12.6.4.3 ParameterValue

```
<element name="ParameterValue" type="gml:ParameterValueType"
substitutionGroup="gml:AbstractGeneralParameterValue"/>

<complexType name="ParameterValueType">
    <complexContent>
        <extension base="gml:AbstractGeneralParameterValueType">
            <sequence>
                <choice>
                    <element ref="gml:value"/>
                    <element ref="gml:dmsAngleValue"/>
                    <element ref="gml:stringValue"/>
                    <element ref="gml:integerValue"/>
                    <element ref="gml:booleanValue"/>
                    <element ref="gml:valueList"/>
                    <element ref="gml:integerValueList"/>
                    <element ref="gml:valueFile"/>
                </choice>
                <element ref="gml:operationParameter"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

`gml:ParameterValue` is a parameter value, an ordered sequence of values, or a reference to a file of parameter values. This concrete complex type may be used for operation methods without using an Application Schema that defines operation-method-specialized element names and contents, especially for methods with only one instance. This complex type may be used, extended, or restricted for well-known operation methods, especially for methods with many instances.

The dmsAngleValue element is deprecated.

12.6.4.4 value

```
<element name="value" type="gml:MeasureType"/>
```

gml:value is a numeric value of an operation parameter, with its associated unit of measure.

12.6.4.5 stringValue

```
<element name="stringValue" type="string"/>
```

gml:stringValue is a character string value of an operation parameter. A string value does not have an associated unit of measure.

12.6.4.6 integerValue

```
<element name="integerValue" type="positiveInteger"/>
```

gml:integerValue is a positive integer value of an operation parameter, usually used for a count. An integer value does not have an associated unit of measure.

12.6.4.7 booleanValue

```
<element name="booleanValue" type="boolean"/>
```

gml:booleanValue is a boolean value of an operation parameter. A Boolean value does not have an associated unit of measure.

12.6.4.8 valueList

```
<element name="valueList" type="gml:MeasureListType"/>
```

gml:valueList is an ordered sequence of two or more numeric values of an operation parameter list, where each value has the same associated unit of measure. An element of this type contains a space-separated sequence of double values.

12.6.4.9 integerValueList

```
<element name="integerValueList" type="gml:integerList"/>
```

gml:integerValueList is an ordered sequence of two or more integer values of an operation parameter list, usually used for counts. These integer values do not have an associated unit of measure. An element of this type contains a space-separated sequence of integer values.

12.6.4.10 valueFile

```
<element name="valueFile" type="anyURI"/>
```

gml:valueFile is a reference to a file or a part of a file containing one or more parameter values, each numeric value with its associated unit of measure. When referencing a part of a file, that file shall contain multiple identified parts, such as an XML encoded document. Furthermore, the referenced file or part of a file may reference another part of the same or different files, as allowed in XML documents.

12.6.4.11 operationParameter

```
<element name="operationParameter" type="gml:OperationParameterPropertyType"/>
```

gml:operationParameter is an association role to the operation parameter of which this is a value.

12.6.4.12 ParameterValueGroup

```
<element name="ParameterValueGroup" type="gml:ParameterValueGroupType"
substitutionGroup="gml:AbstractGeneralParameterValue"/>

<complexType name="ParameterValueGroupType">
	<complexContent>
		<extension base="gml:AbstractGeneralParameterValueType">
			<sequence>
				<element ref="gml:parameterValue" minOccurs="2" maxOccurs="unbounded"/>
				<element ref="gml:group"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:ParameterValueGroup` is a group of related parameter values. The same group can be repeated more than once in a Conversion, Transformation, or higher-level ParameterValueGroup, if those instances contain different values of one or more parameterValues which suitably distinguish among those groups. This concrete complex type can be used for operation methods without using an Application Schema that defines operation-method-specialized element names and contents. This complex type may be used, extended, or restricted for well-known operation methods, especially for methods with only one instance.

The `gml:parameterValue` elements are an unordered set of composition association roles to the parameter values and groups of values included in this group.

12.6.4.13 group

```
<element name="group" type="gml:OperationParameterGroupPropertyType"/>
```

`gml:group` is an association role to the operation parameter group for which this element provides parameter values.

12.6.5 Operation method

12.6.5.1 OperationMethod

```
<element name="OperationMethod" type="gml:OperationMethodType" substitutionGroup="gml:Definition"/>

<complexType name="OperationMethodType">
	<complexContent>
		<extension base="gml:IdentifiedObjectType">
			<sequence>
				<choice>
					<element ref="gml:formulaCitation"/>
					<element ref="gml:formula"/>
				</choice>
				<element ref="gml:sourceDimensions" minOccurs="0"/>
				<element ref="gml:targetDimensions" minOccurs="0"/>
				<element ref="gml:parameter" minOccurs="0" maxOccurs="unbounded"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:OperationMethod` is a method (algorithm or procedure) used to perform a coordinate operation. Most operation methods use a number of operation parameters, although some coordinate conversions use none. Each coordinate operation using the method assigns values to these parameters.

The `gml:parameter` elements are an unordered list of associations to the set of operation parameters and parameter groups used by this operation method.

12.6.5.2 formula, formulaCitation

```
<element name="formula" type="gml:CodeType"/>
```

`gml:formula` Formula(s) or procedure used by an operation method.

The use of the codespace attribute has been deprecated. The property value shall be a character string.

```
<element name="formulaCitation">
    <complexType>
        <sequence minOccurs="0">
            <element ref="gmd:CI_Citation"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
</element>
```

`gml:formulaCitation` provides a reference to a publication giving the formula(s) or procedure used by an coordinate operation method.

12.6.5.3 sourceDimensions

```
<element name="sourceDimensions" type="positiveInteger"/>
```

`gml:sourceDimensions` is the number of dimensions in the source CRS of this operation method.

12.6.5.4 targetDimensions

```
<element name="targetDimensions" type="positiveInteger"/>
```

`gml:targetDimensions` is the number of dimensions in the target CRS of this operation method

12.6.5.5 parameter

```
<element name="parameter " type="gml:AbstractGeneralOperationParameterPropertyType"/>
```

`gml:parameter` is an association to an operation parameter or parameter group.

12.6.5.6 OperationMethodPropertyType

```
<complexType name="OperationMethodPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:OperationMethod"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:OperationMethodPropertyType` is a property type for association roles to a concrete general-purpose operation method, either referencing or containing the definition of that method.

12.6.6 Operation parameters and groups

12.6.6.1 GeneralOperationParameter

```
<element name="AbstractGeneralOperationParameter" type="gml:AbstractGeneralOperationParameterType"
abstract="true" substitutionGroup="gml:Definition"/>

<complexType name="AbstractGeneralOperationParameterType" abstract="true">
	<complexContent>
		<extension base="gml:IdentifiedObjectType">
			<sequence>
				<element ref="gml:minimumOccurs" minOccurs="0"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:GeneralOperationParameter` is the abstract definition of a parameter or group of parameters used by an operation method.

12.6.6.2 minimumOccurs

```
<element name="minimumOccurs" type="nonNegativeInteger"/>
```

`gml:minimumOccurs` is the minimum number of times that values for this parameter group or parameter are required. If this attribute is omitted, the minimum number shall be one.

12.6.6.3 AbstractGeneralOperationParameterPropertyType

```
<complexType name="AbstractGeneralOperationParameterPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractGeneralOperationParameter"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:AbstractGeneralOperationParameterPropertyType` is a property type for association roles to an operation parameter or group, either referencing or containing the definition of that parameter or group.

12.6.6.4 OperationParameter

```
<element name="OperationParameter" type="gml:OperationParameterType"
substitutionGroup="gml:AbstractGeneralOperationParameter"/>

<complexType name="OperationParameterType">
	<complexContent>
		<extension base="gml:AbstractGeneralOperationParameterType">
			<sequence/>
		</extension>
	</complexContent>
</complexType>
```

`gml:OperationParameter` is the definition of a parameter used by an operation method. Most parameter values are numeric, but other types of parameter value are possible. This complex type is expected to be used or extended for all operation methods, without defining operation-method-specialized element names.

12.6.6.5 OperationParameterPropertyType

```
<complexType name="OperationParameterPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:OperationParameter"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:OperationParameterPropertyType` is a property type for association roles to an operation parameter, either referencing or containing the definition of that parameter.

12.6.6.6 OperationParameterGroup

```
<element name="OperationParameterGroup" type="gml:OperationParameterGroupType"
substitutionGroup="gml:AbstractGeneralOperationParameter"/>

<complexType name="OperationParameterGroupType">
    <complexContent>
        <extension base="gml:AbstractGeneralOperationParameterType">
            <sequence>
                <element ref="gml:maximumOccurs" minOccurs="0"/>
                <element ref="gml:parameter" minOccurs="2" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

`gml:OperationParameterGroup` is the definition of a group of parameters used by an operation method. This complex type is expected to be used or extended for all applicable operation methods, without defining operation-method-specialized element names.

The `gml:parameter` elements are an unordered list of associations to the set of operation parameters that are members of this group.

12.6.6.7 maximumOccurs

```
<element name="maximumOccurs" type="positiveInteger"/>
```

`gml:maximumOccurs` is the maximum number of times that values for this parameter group may be included. If this attribute is omitted, the maximum number shall be one.

12.6.6.8 OperationParameterPropertyType

```
<complexType name="OperationParameterGroupPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:OperationParameterGroup"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:OperationParameterPropertyType` is a property type for association roles to an operation parameter group, either referencing or containing the definition of that parameter group.

13 GML schema — Topology

13.1 General concepts

13.1.1 Overview

Topology is the branch of mathematics describing the properties of objects which are invariant under continuous deformation. For example, a circle is topologically equivalent to an ellipse because one can be transformed into the other by stretching. In geographic modelling, the foremost use of topology is in accelerating computational geometry. The constructs of topology allow characterization of the spatial relationships between objects using simple combinatorial or algebraic algorithms. Topology, realized by the appropriate geometry, also allows a compact and unambiguous mechanism for expressing shared geometry among geographic features.

NOTE 1 The topology model of GML complies with ISO 19107. The underlying concepts of the types and elements of the GML topology model are discussed in this document in more detail.

This clause describes the topology schema components as specified by GML.

NOTE 2 The corresponding topology schema document, topology.xsd (see Annex C), is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:topology:3.2.1

There are four instantiable classes of primitive topology objects, one for each dimension up to 3D. In addition, topology complexes are supported.

There is strong symmetry in the (topological boundary and coboundary) relationships between topology primitives of adjacent dimensions. Topology primitives are bounded by directed primitives of one lower dimension. The coboundary of each topology primitive is formed from directed topology primitives of one higher dimension.

13.1.2 Relationship with ISO 19107

The spatial topology components of the GML schema specified in this clause provide a conformant, partial implementation of the ISO 19107 spatial schema (topology). The relationship is discussed in detail in D.2.3.

The ISO 19107 topology types implemented in GML are specified in ISO 19107; some additional constraints are specified in ISO 19107 for these types, which are also constraints on the spatial topology components of the GML schema.

In addition, GML specifies complementary spatial topology schema components as described in D.3.10.

13.2 Abstract topology

```
<complexType name="AbstractTopologyType" abstract="true">
  <complexContent>
    <extension base="gml:AbstractGMLType"/>
  </complexContent>
</complexType>

<element name="AbstractTopology" type="gml:AbstractTopologyType" abstract="true"
substitutionGroup="gml:AbstractGML"/>
```

This abstract type supplies the root or base type for all topological elements including primitives and complexes. It inherits AbstractGMLType and hence can be identified using the `gml:id` attribute.

`gml:AbstractTopology` implements ISO 19107 TP_Object (see D.2.4.2 and ISO 19107:2003, 7.2.2).

13.3 Topological primitives

13.3.1 Abstract topological primitives

13.3.1.1 AbstractTopoPrimitive, AbstractTopoPrimtive

```
<complexType name="AbstractTopoPrimitiveType" abstract="true">
  <complexContent>
    <extension base="gml:AbstractTopologyType">
      <sequence/>
    </extension>
  </complexContent>
</complexType>

<element name="AbstractTopoPrimitive" type="gml:AbstractTopoPrimitiveType" abstract="true"
 substitutionGroup="gml:AbstractTopology"/>
```

`gml:AbstractTopoPrimitive` implements ISO 19107 TP_Primitive (see D.2.4.3 and ISO 19107:2003, 7.3.10). This abstract type acts as the base type for all topological primitives. Topological primitives are the atomic (smallest possible) units of a topology complex.

Each topological primitive may contain references to other topology primitives of codimension 2 or more (`gml:isolated`, implemented in subtypes).

EXAMPLE Faces may contain isolated nodes and solids may contain isolated nodes and edges.

Conversely, nodes may have faces as containers and nodes and edges may have `solids` as containers (`gml:container`, implemented in subtypes).

13.3.2 Topological primitives (0-dimensional)

13.3.2.1 NodeType, Node

```
<complexType name="NodeType">
  <complexContent>
    <extension base="gml:AbstractTopoPrimitiveType">
      <sequence>
        <element name="container" type="gml:FaceOrTopoSolidPropertyType" minOccurs="0"/>
        <element ref="gml:directedEdge" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="gml:pointProperty" minOccurs="0"/>
      </sequence>
      <attributeGroup ref="gml:AggregationAttributeGroup"/>
    </extension>
  </complexContent>
</complexType>

<element name="Node" type="gml:NodeType" substitutionGroup="gml:AbstractTopoPrimitive"/>
```

`gml:Node` represents the 0-dimensional primitive and implements ISO 19107 TP_Node (see D.2.4.3 and ISO 19107:2003, 7.3.12).

The `gml:container` property element implements the role of the same name of the ISO 19107 “Isolated In” association (see ISO 19107:2003, 7.3.10.4 and D.2.4.3).

```
<complexType name="FaceOrTopoSolidPropertyType">
    <choice minOccurs="0">
        <element ref="gml:Face"/>
        <element ref="gml:TopoSolid"/>
```

```
        </choice>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </complexType>
```

The optional coboundary of a node (`gml:directedEdge`) is a sequence of directed edges which are incident on this node. Edges emanating from this node appear in the node coboundary with a negative orientation.

In the case of planar topology, the sequence of `gml:directedEdge`s shall be clockwise to ensure a lossless topology representation.

If provided, the `aggregationType` attribute shall have the value “sequence”.

A node may optionally be realized by a 0-dimensional geometric primitive (`gml:pointProperty`).

13.3.2.2 DirectedNodePropertyType, directedNode

```
    <element name="directedNode" type="gml:DirectedNodePropertyType"/>

    <complexType name="DirectedNodePropertyType">
        <sequence minOccurs="0">
            <element ref="gml:Node"/>
        </sequence>
        <attribute name="orientation" type="gml:SignType" default="+"/>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </complexType>
```

A `gml:directedNode` property element describes the boundary of topology edges and is used in the support of topological point features via the `gml:TopoPoint` expression, see below. The orientation attribute of type `gml:SignType` expresses the sense in which the included node is used: start (“-”) or end (“+”) node.

13.3.3 Topological primitives (1-dimensional)

13.3.3.1 EdgeType, Edge

```
  <complexType name="EdgeType">
    <complexContent>
      <extension base="gml:AbstractTopoPrimitiveType">
        <sequence>
          <element name="container" type="gml:TopoSolidPropertyType" minOccurs="0"/>
          <element ref="gml:directedNode" minOccurs="2" maxOccurs="2"/>
          <element ref="gml:directedFace" minOccurs="0" maxOccurs="unbounded"/>
          <element ref="gml:curveProperty" minOccurs="0"/>
        </sequence>
        <attributeGroup ref="gml:AggregationAttributeGroup"/>
      </extension>
    </complexContent>
  </complexType>

  <element name="Edge" type="gml:EdgeType" substitutionGroup="gml:AbstractTopoPrimitive"/>
```

`gml:Edge` represents the 1-dimensional primitive and implements ISO 19107 TP_Edge (see D.2.4.3 and ISO 19107:2003, 7.3.14).

The `gml:container` property element implements the role of the same name of the ISO 19107 “Isolated In” association (see ISO 19107:2003, 7.3.10.4 and D.2.4.3).

The topological boundary of an Edge (gml:directedNode) consists of a negatively directed start Node and a positively directed end Node.

The optional coboundary of an edge (gml:directedFace) is a circular sequence of directed faces which are incident on this edge in document order. In the 2D case, the orientation of the face on the left of the edge is "+"; the orientation of the face on the right on its right is "-".

If provided, the aggregationType attribute shall have the value “sequence”.

An edge may optionally be realized by a 1-dimensional geometric primitive (gml:curveProperty).

13.3.3.2 DirectedEdgePropertyType, directedEdge

```
<element name="directedEdge" type="gml:DirectedEdgePropertyType"/>

<complexType name="DirectedEdgePropertyType">
    <sequence minOccurs="0">
        <element ref="gml:Edge"/>
    </sequence>
    <attribute name="orientation" type="gml:SignType" default="+"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

A gml:directedEdge property element describes the boundary of topology faces, the coBoundary of topology nodes and is used in the support of topological line features via the gml:TopoCurve expression, see below. The orientation attribute of type gml:SignType expresses the sense in which the included edge is used, i.e. forward or reverse.

13.3.4 Topological primitives (2-dimensional)

13.3.4.1 FaceType, Face

```
<complexType name="FaceType">
  <complexContent>
    <extension base="gml:AbstractTopoPrimitiveType">
      <sequence>
        <element name="isolated" type="gml:NodePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="gml:directedEdge" maxOccurs="unbounded"/>
        <element ref="gml:directedTopoSolid" minOccurs="0" maxOccurs="2"/>
        <element ref="gml:surfaceProperty" minOccurs="0"/>
      </sequence>
      <attributeGroup ref="gml:AggregationAttributeGroup"/>
      <attribute name="universal" type="boolean" use="optional" default="false"/>
    </extension>
  </complexContent>
</complexType>

<element name="Face" type="gml:FaceType" substitutionGroup="gml:AbstractTopoPrimitive"/>
```

gml:Face represents the 2-dimensional topology primitive and implements ISO 19107 TP_Face (see D.2.4.3 and ISO 19107:2003, 7.3.16).

The gml:isolated property element implements the role of the same name of the ISO 19107 “Isolated In” association (see ISO 19107:2003, 7.3.10.4 and D.2.4.3).

The topological boundary of a face (gml:directedEdge) consists of a sequence of directed edges. If provided, the aggregationType attribute shall have the value “sequence”.

NOTE 1 All edges associated with the face, including dangling edges, appear in the boundary. A dangling edge has the same face on both sides. Consequently, a dangling edge has two different nodes in its boundary. A dangling edge may share zero, one or two bounding nodes with other edges in the boundary of a face. Two directedEdge elements with opposite orientations reference each dangling edge in the boundary of a face. The non-dangling edges in the boundary of a face comprise one or more topological rings. Each such ring consists of directedEdges connected in a cycle, and is oriented with the face on its left.

The optional coboundary of a face (`gml:directedTopoSolid`) is a pair of directed solids which are bounded by this face. A positively directed solid corresponds to a solid which lies in the direction of the negatively directed normal to the face in any geometric realization.

A face may optionally be realized by a 2-dimensional geometric primitive (`gml:surfaceProperty`).

If the topological representation exists an unbounded manifold (e.g. Euclidean plane), a `gml:Face` shall indicate whether it is a universal face or not, to ensure a lossless topology representation. The optional universal attribute of type boolean is used to indicate this.

NOTE 2 The universal face is normally not part of any feature, and is used to represent the unbounded portion of the data set. Its interior boundary (it has no exterior boundary) would normally be considered the exterior boundary of the map represented by the data set.

13.3.4.2 DirectedFacePropertyType, directedFace

```
<element name="directedFace" type="gml:DirectedFacePropertyType"/>

<complexType name="DirectedFacePropertyType">
    <sequence minOccurs="0">
        <element ref="gml:Face"/>
    </sequence>
    <attribute name="orientation" type="gml:SignType" default="+"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The `gml:directedFace` property element describes the boundary of topology solids, in the coBoundary of topology edges and is used in the support of surface features via the `gml:TopoSurface` expression, see below. The orientation attribute of type `gml:SignType` expresses the sense in which the included face is used i.e. inward or outward with respect to the surface normal in any geometric realization.

13.3.5 Topological primitives (3-dimensional)

13.3.5.1 TopoSolidType, TopoSolid

```
<complexType name="TopoSolidType">
 <complexContent>
  <extension base="gml:AbstractTopoPrimitiveType">
   <sequence>
    <element name="isolated" type="gml:NodeOrEdgePropertyType" minOccurs="0" maxOccurs="unbounded"/>
    <element ref="gml:directedFace" maxOccurs="unbounded"/>
    <element ref="gml:solidProperty" minOccurs="0"/>
   </sequence>
   <attributeGroup ref="gml:AggregationAttributeGroup"/>
   <attribute name="universal" type="boolean" use="optional" default="false"/>
  </extension>
 </complexContent>
</complexType>

<element name="TopoSolid" type="gml:TopoSolidType" substitutionGroup="gml:AbstractTopoPrimitive"/>
```


`gml:TopoSolid` represents the 3-dimensional topology primitive and implements ISO 19107 TP_Solid (see D.2.4.3 and ISO 19107:2003, 7.3.18).

The `gml:isolated` property element implements the role of the same name of the ISO 19107 “Isolated In” association (see ISO 19107:2003, 7.3.10.4 and D.2.4.3).

```
<complexType name="NodeOrEdgePropertyType">
	<choice minOccurs="0">
		<element ref="gml:Node"/>
		<element ref="gml:Edge"/>
	</choice>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The topological boundary of a solid (`gml:directedFace`) consists of a set of directed faces.

NOTE 1 All faces associated with the solid, including dangling faces, appear in the boundary. A dangling face has the same solid on both sides. Two directedFace elements with opposite orientations reference each dangling face in the boundary of a topological solid.

A solid may optionally be realized by a 3-dimensional geometric primitive (`gml:solidProperty`).

A `gml:TopoSolid` shall indicate whether it is a universal topo solid or not, to ensure a lossless topology representation. The optional universal attribute of type boolean is used to indicate this and the default is false.

NOTE 2 The universal topo solid is normally not part of any feature, and is used to represent the unbounded portion of the data set. Its interior boundary (it has no exterior boundary) would normally be considered the exterior boundary of the data set.

13.3.5.2 DirectedTopoSolidPropertyType, directedTopoSolid

```
<element name="directedTopoSolid" type="gml:DirectedTopoSolidPropertyType"/>

<complexType name="DirectedTopoSolidPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:TopoSolid"/>
	</sequence>
	<attribute name="orientation" type="gml:SignType" default="+"/>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The `gml:directedSolid` property element describes the coBoundary of topology faces and is used in the support of volume features via the `gml:TopoVolume` expression, see below. The orientation attribute of type `gml:SignType` expresses the sense in which the included solid appears in the face coboundary. In the context of a `gml:TopoVolume` the orientation attribute has no meaning.

13.4 Topological collections

13.4.1 Topological collection (0-dimensional)

13.4.1.1 TopoPointType, TopoPoint

```
<complexType name="TopoPointType">
  <complexContent>
     <extension base="gml:AbstractTopologyType">
        <sequence>
```



```
                <element ref="gml:directedNode"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<element name="TopoPoint" type="gml:TopoPointType"/>
```

The intended use of `gml:TopoPoint` is to appear within a point feature to express the structural and possibly geometric relationships of this feature to other features via shared node definitions.

NOTE The `orientation` assigned to the `gml:directedNode` has no meaning in this context. It is preserved for symmetry with the corresponding elements of other dimensions, see below.

13.4.1.2 TopoPointPropertyType, topoPointProperty

```
<complexType name="TopoPointPropertyType">
    <sequence>
        <element ref="gml:TopoPoint"/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>

<element name="topoPointProperty" type="gml:TopoPointPropertyType"/>
```

The `gml:topoPointProperty` property element may be used in features to express their relationship to the referenced topology node.

13.4.2 Topological collection (1-dimensional)

13.4.2.1 TopoCurveType, TopoCurve

```
<complexType name="TopoCurveType">
    <complexContent>
        <extension base="gml:AbstractTopologyType">
            <sequence>
                <element ref="gml:directedEdge" maxOccurs="unbounded"/>
            </sequence>
            <attributeGroup ref="gml:AggregationAttributeGroup"/>
        </extension>
    </complexContent>
</complexType>

<element name="TopoCurve" type="gml:TopoCurveType"/>
```

`gml:TopoCurve` represents a homogeneous topological expression, a sequence of directed edges, which if realized are isomorphic to a geometric curve primitive. The intended use of `gml:TopoCurve` is to appear within a line feature to express the structural and geometric relationships of this feature to other features via the shared edge definitions.

If provided, the `aggregationType` attribute shall have the value “sequence”.

13.4.2.2 TopoCurvePropertyType, topoCurveProperty

```
<complexType name="TopoCurvePropertyType">
    <sequence>
        <element ref="gml:TopoCurve"/>
    </sequence>
```



```
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>

<element name="topoCurveProperty" type="gml:TopoCurvePropertyType"/>
```

The `gml:topoCurveProperty` property element may be used in features to express their relationship to the referenced topology edges.

13.4.3 Topological collection (2-dimensional)

13.4.3.1 TopoSurfaceType, TopoSurface

```
<complexType name="TopoSurfaceType">
  <complexContent>
    <extension base="gml:AbstractTopologyType">
      <sequence>
        <element ref="gml:directedFace" maxOccurs="unbounded"/>
      </sequence>
      <attributeGroup ref="gml:AggregationAttributeGroup"/>
    </extension>
  </complexContent>
</complexType>

<element name="TopoSurface" type="gml:TopoSurfaceType"/>
```

`gml:TopoSurface` represents a homogeneous topological expression, a set of directed faces, which if realized are isomorphic to a geometric surface primitive. The intended use of `gml:TopoSurface` is to appear within a surface feature to express the structural and possibly geometric relationships of this surface feature to other features via the shared face definitions.

13.4.3.2 TopoSurfacePropertyType, topoSurfaceProperty

```
<complexType name="TopoSurfacePropertyType">
    <sequence>
        <element ref="gml:TopoSurface"/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>

<element name="topoSurfaceProperty" type="gml:TopoSurfacePropertyType"/>
```

The `gml:topoSurfaceProperty` property element may be used in features to express their relationship to the referenced topology faces.

13.4.4 Topological collection (3-dimensional)

13.4.4.1 TopoVolumeType, TopoVolume

```
<complexType name="TopoVolumeType">
  <complexContent>
    <extension base="gml:AbstractTopologyType">
      <sequence>
        <element ref="gml:directedTopoSolid" maxOccurs="unbounded"/>
      </sequence>
      <attributeGroup ref="gml:AggregationAttributeGroup"/>
    </extension>
  </complexContent>
</complexType>
```



```
<element name="TopoVolume" type="gml:TopoVolumeType"/>
```

`gml:TopoVolume` represents a homogeneous topological expression, a set of directed topologic solids, which if realized are isomorphic to a geometric solid primitive. The intended use of `gml:TopoVolume` is to appear within a solid feature to express the structural and geometric relationships of this solid feature to other features via the shared solid definitions.

NOTE The `orientation` assigned to the `gml:directedSolid` has no meaning in three dimensions. It is preserved for symmetry with the corresponding elements of other dimensions, see above.

13.4.4.2 TopoVolumePropertyType, topoVolumeProperty

```
<complexType name="TopoVolumePropertyType">
	<sequence>
		<element ref="gml:TopoVolume"/>
	</sequence>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>

<element name="topoVolumeProperty" type="gml:TopoVolumePropertyType"/>
```

The `gml:topoVolumeProperty` element may be used in features to express their relationship to the referenced topology volume.

13.5 Topology complex

13.5.1 TopoComplexType, TopoComplex

```
<complexType name="TopoComplexType">
	<complexContent>
		<extension base="gml:AbstractTopologyType">
			<sequence>
				<element ref="gml:maximalComplex"/>
				<element ref="gml:superComplex" minOccurs="0" maxOccurs="unbounded"/>
				<element ref="gml:subComplex" minOccurs="0" maxOccurs="unbounded"/>
				<element ref="gml:topoPrimitiveMember" minOccurs="0" maxOccurs="unbounded"/>
				<element ref="gml:topoPrimitiveMembers" minOccurs="0"/>
			</sequence>
			<attribute name="isMaximal" type="boolean" default="false"/>
			<attributeGroup ref="gml:AggregationAttributeGroup"/>
		</extension>
	</complexContent>
</complexType>

<element name="TopoComplex" type="gml:TopoComplexType" substitutionGroup="gml:AbstractTopology"/>
```

`gml:TopoComplex` is a collection of topological primitives and implements ISO 19107 TP_Complex (see D.2.4.4 and ISO 19107:2003, 7.4.2).

Each complex holds a reference to its maximal complex (`gml:maximalComplex`) and optionally to sub- or super-complexes (`gml:subComplex`, `gml:superComplex`).

A topology complex contains its primitive and sub-complex members.

NOTE The maximal complex is the complex which has no super-complex. There is one and only one maximal complex per topological manifold.

13.5.2 Maximal, sub- and super-complexes

```
<element name="subComplex" type="gml:TopoComplexPropertyType"/>
<element name="superComplex" type="gml:TopoComplexPropertyType"/>
<element name="maximalComplex" type="gml:TopoComplexPropertyType"/>
```

The property elements `gml:subComplex`, `gml:superComplex` and `gml:maximalComplex` provide an encoding for relationships between topology complexes as described for `gml:TopoComplex` above.

13.5.3 topoPrimitiveMember

```
<element name="topoPrimitiveMember" type="gml:TopoPrimitivePropertyType"/>

<complexType name="TopoPrimitivePropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractTopoPrimitive" />
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The `gml:topoPrimitiveMember` property element encodes for the relationship between a topology complex and a single topology primitive.

13.5.4 topoPrimitiveMembers

```
<element name="topoPrimitiveMembers" type="gml:TopoPrimitiveArrayAssociationType"/>

<complexType name="TopoPrimitiveArrayAssociationType">
	<sequence minOccurs="0" maxOccurs="unbounded">
		<element ref="gml:AbstractTopoPrimitive" />
	</sequence>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The `gml:topoPrimitiveMembers` property element encodes the relationship between a topology complex and an arbitrary number of topology primitives.

NOTE Because the property value can be multiple topological primitives, the elements representing the topology primitives are always encoded inline.

13.5.5 TopoComplexPropertyType, topoComplexProperty

```
<complexType name="TopoComplexPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:TopoComplex" />
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>

<element name="topoComplexProperty" type="gml:TopoComplexPropertyType"/>
```

The `gml:topoComplexProperty` property element encodes the relationship between a GML object and a topology complex.

EXAMPLE This allows a feature collection to contain or reference a topology complex that contains topology objects referenced by members of the feature collection.

14 GML schema — Temporal information and dynamic features

14.1 General concepts

14.1.1 Overview

The GML temporal schemas include components for describing temporal geometry and topology, temporal reference systems, and the temporal characteristics of geographic data. The model underlying the representation constitutes a profile of the conceptual schema described in ISO 19108. The underlying spatiotemporal model strives to accommodate both feature-level and attribute-level time stamping; basic support for tracking moving objects is also included.

Time is measured on two types of scale: interval and ordinal. An interval scale offers a basis for measuring duration, an ordinal scale provides information only about relative position in time.

EXAMPLE A stratigraphic sequence or the geological time scale are examples of ordinal scales.

Two other ISO standards are relevant to describing temporal objects: ISO 8601 describes encodings for time instants and time periods, as text strings with particular structure and punctuation; ISO/IEC 11404 provides a detailed description of time intervals as part of a general discussion of language independent datatypes.

The temporal schemas cover two interrelated topics and provide basic schema components for representing temporal instants and periods, temporal topology, and reference systems; more specialized schema components defines components used for dynamic features. Instances of temporal geometric types are used as values for the temporal properties of geographic features.

NOTE The main temporal schema document is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:temporal:3.2.1

The temporal topology schema document is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:temporalTopology:3.2.1

The schema document for temporal reference systems is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:temporalReferenceSystems:3.2.1

The dynamic feature schema document for representing time-varying properties of geographic features is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:dynamicFeature:3.2.1

All schema documents are listed in Annex C.

14.1.2 Relationship with ISO 19108

The temporal geometry and topology components of the GML schema specified in this clause provide a conformant, partial implementation of the ISO 19108 temporal schema. The relationship is discussed in detail in D.2.5.

The ISO 19108 geometry and topology types implemented in GML are specified in ISO 19108; some additional constraints are specified in ISO 19108 for these types, which are also constraints on the temporal geometry and topology components of the GML schema.

In addition, GML specifies complementary temporal schema components as described in D.3.11.

14.2 Temporal schema

14.2.1 Abstract temporal objects

14.2.1.1 AbstractTimeObject

`gml:AbstractTimeObject` implements ISO 19108 TM_Object (see D.2.5.2 and ISO 19108:2002, 5.2.2) and acts as the head of a substitution group for all temporal primitives and complexes. It is declared as follows:

```
<element name="AbstractTimeObject" type="gml:AbstractTimeObjectType" abstract="true"
 substitutionGroup="gml:AbstractGML"/>
```

A `gml:AbstractTimeObject` may be used in any position that a `gml:AbstractGML` is valid. Its content model is defined as follows:

```
<complexType name="AbstractTimeObjectType" abstract="true">
  <complexContent>
    <extension base="gml:AbstractGMLType"/>
  </complexContent>
</complexType>
```

NOTE The content model of gml:AbstractTimeObject is a vacuous extension of AbstractGMLType. Types derived from this have the standard GML object properties available: abstractMetadataProperty, description, descriptionReference, name.

14.2.1.2 AbstractTimePrimitive

`gml:AbstractTimePrimitive` implements ISO 19108 TM_Primitive (see D.2.5.2 and ISO 19108:2002, 5.2.3) and acts as the head of a substitution group for geometric and topological temporal primitives. It is declared as follows:

```
<element name="AbstractTimePrimitive" type="gml:AbstractTimePrimitiveType" abstract="true"
 substitutionGroup="gml:AbstractTimeObject"/>
```

A `gml:AbstractTimePrimitive` may be used in any position that a `gml:AbstractTimeObject` is valid. Its content model is defined as follows:

```
<complexType name="AbstractTimePrimitiveType" abstract="true">
  <complexContent>
    <extension base="gml:AbstractTimeObjectType">
      <sequence>
        <element name="relatedTime" type="gml:RelatedTimeType" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

This extends the model for generic temporal objects with properties indicating relationships between this temporal primitive and other temporal primitives. The definition of `gml:RelatedTimeType` is provided in 14.2.1.4.

14.2.1.3 TimePrimitivePropertyType, validTime

`gml:TimePrimitivePropertyType` provides a standard content model for associations between an arbitrary member of the substitution group whose head is `gml:AbstractTimePrimitive` and another object:

```
<complexType name="TimePrimitivePropertyType">
```



```
        <sequence minOccurs="0">
            <element ref="gml:AbstractTimePrimitive"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </complexType>
```

`gml:validTime` is a convenience property element declared as follows:

```
<element name="validTime" type="gml:TimePrimitivePropertyType"/>
```

14.2.1.4 RelatedTimeType

`gml:RelatedTimeType` provides a content model for indicating the relative position of an arbitrary member of the substitution group whose head is `gml:AbstractTimePrimitive`. It extends the generic `gml:TimePrimitivePropertyType` with an XML attribute `relativePosition`, whose value is selected from the set of 13 temporal relationships identified by Allen (1983):

```
<complexType name="RelatedTimeType">
  <complexContent>
    <extension base="gml:TimePrimitivePropertyType">
      <attribute name="relativePosition">
        <simpleType>
          <restriction base="string">
            <enumeration value="Before"/>
            <enumeration value="After"/>
            <enumeration value="Begins"/>
            <enumeration value="Ends"/>
            <enumeration value="During"/>
            <enumeration value="Equals"/>
            <enumeration value="Contains"/>
            <enumeration value="Overlaps"/>
            <enumeration value="Meets"/>
            <enumeration value="OverlappedBy"/>
            <enumeration value="MetBy"/>
            <enumeration value="BegunBy"/>
            <enumeration value="EndedBy"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
```

14.2.1.5 AbstractTimeComplex

`gml:AbstractTimeComplex` is a collection of temporal primitives and implements ISO 19108 TM_Complex (see D.2.5.2 and ISO 19108:2002, 5.2.2) and acts as the head of a substitution group for temporal complexes. It is declared as follows:

```
<element name="AbstractTimeComplex" type="gml:AbstractTimeComplexType" abstract="true"
substitutionGroup="gml:AbstractTimeObject"/>
```

A `gml:AbstractTimeComplex` may be used in any position that a `gml:AbstractTimeObject` is valid. Its content model is defined as follows:

```
<complexType name="AbstractTimeComplexType" abstract="true">
  <complexContent>
    <extension base="gml:AbstractTimeObjectType"/>
```



```
  </complexContent>
</complexType>
```

NOTE 1 This International Standard only specifies a temporal topology complex. A temporal geometric complex is not specified.

NOTE 2 This International Standard does not distinguish a temporal linear graph from a temporal non-linear graph.

14.2.2 Temporal geometry

14.2.2.1 Introduction

Temporal geometry is described in terms of time instants, periods, positions and lengths.

14.2.2.2 AbstractTimeGeometricPrimitive

`gml:TimeGeometricPrimitive` implements ISO 19108 TM_GeometricPrimitive (see D.2.5.2 and ISO 19108:2002, 5.2.3) and acts as the head of a substitution group for geometric temporal primitives. It is declared as follows:

```
<element name="AbstractTimeGeometricPrimitive" type="gml:AbstractTimeGeometricPrimitiveType" abstract="true"
 substitutionGroup="gml:AbstractTimePrimitive"/>
```

A `gml:AbstractTimeGeometricPrimitive` may be used in any position that a `gml:AbstractTimePrimitive` is valid. Its content model is defined as follows:

```
<complexType name="AbstractTimeGeometricPrimitiveType" abstract="true">
  <complexContent>
    <extension base="gml:AbstractTimePrimitiveType">
      <attribute name="frame" type="anyURI" default="#ISO-8601"/>
    </extension>
  </complexContent>
</complexType>
```

A temporal geometry shall be associated with a temporal reference system through the `frame` attribute that provides a URI reference that identifies a description of the reference system. Following ISO 19108, the Gregorian calendar with UTC is the default reference system, but others may also be used.

The two geometric primitives in the temporal dimension are the instant and the period. GML components are defined to support these as follows.

14.2.2.3 TimeInstant

`gml:TimeInstant` implements ISO 19108 TM_Instant (see D.2.5.2 and ISO 19108:2002, 5.2.3.2) and acts as a zero-dimensional geometric primitive that represents an identifiable position in time. It is declared as follows:

```
<element name="TimeInstant" type="gml:TimeInstantType" substitutionGroup="gml:AbstractTimeGeometricPrimitive"/>
```

A `gml:TimeInstant` may be used in any position that a `gml:AbstractTimeGeometricPrimitive` is valid. Its content model is defined as follows:

```
<complexType name="TimeInstantType" final="#all">
  <complexContent>
    <extension base="gml:AbstractTimeGeometricPrimitiveType ">
      <sequence>
        <element ref="gml:timePosition"/>
      </sequence>
```



```
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE In an instance document, a `gml:TimeInstant` contains a `gml:timePosition` as follows:

```
<gml:TimeInstant gml:id="t11">
  <gml:description>Abby's birthday</gml:description>
  <gml:timePosition>2001-05-23</gml:timePosition>
</gml:TimeInstant>
```

14.2.2.4 TimeInstantPropertyType

`gml:TimeInstantPropertyType` is a specialization of `gml:TimePrimitivePropertyType` that provides for associating a `gml:TimeInstant` with an object:

```
<complexType name="TimeInstantPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:TimeInstant"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

14.2.2.5 TimePeriod

`gml:TimePeriod` implements ISO 19108 TM_Period (see D.2.5.2 and ISO 19108:2002, 5.2.3.3) and acts as a one-dimensional geometric primitive that represents an identifiable extent in time. It is declared as follows:

```
<element name="TimePeriod" type="gml:TimePeriodType" substitutionGroup="gml:AbstractTimeGeometricPrimitive"/>
```

`gml:TimePeriod` may be used in any position that a `gml:AbstractTimeGeometricPrimitive` is valid. Its content model is defined as follows:

```
<complexType name="TimePeriodType">
  <complexContent>
    <extension base="gml:AbstractTimeGeometricPrimitiveType">
      <sequence>
        <choice>
          <element name="beginPosition" type="gml:TimePositionType"/>
          <element name="begin" type="gml:TimeInstantPropertyType"/>
        </choice>
        <choice>
          <element name="endPosition" type="gml:TimePositionType"/>
          <element name="end" type="gml:TimeInstantPropertyType"/>
        </choice>
        <group ref="gml:timeLength" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The location in time of a `gml:TimePeriod` is described by the temporal positions of the instants at which it begins and ends. The length of the period is equal to the temporal distance between the two bounding temporal positions.

Both beginning and end may be described in terms of their direct position using `gml:TimePositionType` (see 14.2.2.7) which is an XML Schema simple content type, or by reference to an identifiable time instant using `gml:TimeInstantPropertyType` (see 14.2.2.4).

EXAMPLE 1 Within a `gml:TimePeriod`, a `gml:TimeInstant` may appear directly as the value of the begin and end as follows:

```
<gml:TimePeriod gml:id="p22">
  <gml:begin>
    <gml:TimeInstant gml:id="t11">
      <gml:timePosition>2001-05-23</gml:timePosition>
    </gml:TimeInstant>
  </gml:begin>
  <gml:end>
    <gml:TimeInstant gml:id="t12">
      <gml:timePosition>2001-06-23</gml:timePosition>
    </gml:TimeInstant>
  </gml:end>
</gml:TimePeriod>
```

Alternatively a limit of a `gml:TimePeriod` may use the conventional GML property model to make a reference to a time instant described elsewhere, or a limit may be indicated as a direct position.

EXAMPLE 2 The following mixed example shows both of these, as well as including the optional gml:duration property:

```
<gml:TimePeriod gml:id="p22">
  <gml:begin xlink:href="#t11"/>
  <gml:endPosition>2002-05-23</gml:endPosition>
  <gml:duration>P1Y</gml:duration>
</gml:TimePeriod>
```

14.2.2.6 TimePeriodPropertyType

`gml:TimePeriodPropertyType` is a specialization of `gml:TimePrimitivePropertyType` that provides for associating a `gml:TimePeriod` with an object:

```
<complexType name="TimePeriodPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:TimePeriod"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

14.2.2.7 TimePositionType, timePosition

The method for identifying a temporal position is specific to each temporal reference system. `gml:TimePositionType` supports the description of temporal position in accordance with the subtypes described in ISO 19108. It implements ISO 19108 TM_Position (see D.2.5.5 and ISO 19108:2002, 5.4.2).

Values based on calendars and clocks use lexical formats that are based on ISO 8601, as described in XML Schema Part 2:2001. A decimal value may be used with coordinate systems such as GPS time or UNIX time. A URI may be used to provide a reference to some era in an ordinal reference system[3].

3) e.g. a geological epoch.

In common with many of the components modelled as data types in the ISO 19100 series of International Standards, the corresponding GML component has simple content. However, the content model `gml:TimePositionType` is defined in several steps (the details of the mapping to ISO 19108 TM_Position are described in D.2.5.5):

```
<complexType name="TimePositionType" final="#all">
  <simpleContent>
    <extension base="gml:TimePositionUnion">
      <attribute name="frame" type="anyURI" default="#ISO-8601"/>
      <attribute name="calendarEraName" type="string" />
      <attribute name="indeterminatePosition" type="gml:TimeIndeterminateValueType" />
    </extension>
  </simpleContent>
</complexType>
```

Three XML attributes appear on `gml:TimePositionType`:

A time value shall be associated with a temporal reference system through the `frame` attribute that provides a URI reference that identifies a description of the reference system. Following ISO 19108, the Gregorian calendar with UTC is the default reference system, but others may also be used. Components for describing temporal reference systems are described in 14.4, but it is not required that the reference system be described in this manner, as the reference may refer to any resource that may be identified with a URI.

For time values using a calendar containing more than one era, the (optional) `calendarEraName` attribute provides the name of the calendar era.

Inexact temporal positions may be expressed using the optional `indeterminatePosition` attribute. This takes a value from an enumeration defined as follows:

```
<simpleType name="TimeIndeterminateValueType">
  <restriction base="string">
    <enumeration value="after"/>
    <enumeration value="before"/>
    <enumeration value="now"/>
    <enumeration value="unknown"/>
  </restriction>
</simpleType>
```

These values are interpreted as follows:

— “unknown” indicates that no specific value for temporal position is provided;

— “now” indicates that the specified value shall be replaced with the current temporal position whenever the value is accessed;

— “before” indicates that the actual temporal position is unknown, but it is known to be before the specified value;

— “after” indicates that the actual temporal position is unknown, but it is known to be after the specified value.

A value for indeterminatePosition may

— be used either alone, or

— qualify a specific value for temporal position[4].

4) e.g. before 2002-12, after 1019624400.

The simple type gml:TimePositionUnion is a union of XML Schema simple types which instantiate the subtypes for temporal position described in ISO 19108.

```
<simpleType name="TimePositionUnion">
   <union memberTypes="gml:CalDate time dateTime anyURI decimal"/>
 </simpleType>
```

An ordinal era may be referenced via URI. A decimal value may be used to indicate the distance from the scale origin[5]. time is used for a position that recurs daily (see ISO 19108:2002 5.4.4.2).

Finally, calendar and clock forms that support the representation of time in systems based on years, months, days, hours, minutes and seconds, in a notation following ISO 8601, are assembled as follows:

```
 <simpleType name="CalDate">
   <union memberTypes="date gYearMonth gYear"/>
 </simpleType>
```

NOTE 1 The XML Schema simpleType dateTime does not permit right-truncation, except for fractions of seconds, which is why date, gYear and gYearMonth are required.

NOTE 2 Following ISO 19108, when used with non-Gregorian calendars based on years, months, days, the same lexical representation should still be used. Following XML Schema Part 2, leading zeros should be added if the year value would otherwise have fewer than four digits.

The element gml:timePosition is declared as follows:

```
 <element name="timePosition" type="gml:TimePositionType"/>
```

This element is used directly as a property of gml:TimeInstant (see 14.2.2.3), and may also be used in application schemas.

EXAMPLE The following examples illustrate how gml:timePosition or other elements of this type may appear in a data instance:

```
<gml:timePosition>2002-11-25T13:20:20Z</gml:timePosition>

<gml:timePosition indeterminatePosition="after">1994</gml:timePosition>

<gml:timePosition indeterminatePosition="now">1994-07-10</gml:timePosition>

<gml:timePosition frame="http://my.big.org/TRS/GPS">25876321.01</gml:timePosition>

<gml:timePosition frame="http://my.big.org/TRS/archaeology"> http://my.history.org/eras/bronzeAge </gml:timePosition>

<gml:timePosition frame="http://my.big.org/TRS/calendars/japanese" calendarEraName="Meiji">0025-03</gml:timePosition>
```

14.2.2.8 timeLength, duration, timeInterval, TimeUnitType

The length of a time period is described using the group gml:timeLength, which is declared in the schema as follows:

```
      <group name="timeLength">
            <choice>
```

5) e.g. UNIX time, GPS calendar.

```
            <element ref="gml:duration"/>
            <element ref="gml:timeInterval"/>
        </choice>
    </group>
```

Its content model is a choice of two property elements:

```
    <element name="duration" type="duration"/>
```

`gml:duration` conforms to the ISO 8601 syntax for temporal length as implemented by the XML Schema duration type. The other alternative is `gml:timeInterval` which conforms to ISO/IEC 11404 which is based on floating point values for temporal length.

```
    <element name="timeInterval" type="gml:TimeIntervalLengthType"/>

    <complexType name="TimeIntervalLengthType" final="#all">
        <simpleContent>
            <extension base="decimal">
                <attribute name="unit" type="gml:TimeUnitType" use="required"/>
                <attribute name="radix" type="positiveInteger"/>
                <attribute name="factor" type="integer"/>
            </extension>
        </simpleContent>
    </complexType>
```

ISO/IEC 11404 syntax specifies the use of a positiveInteger together with appropriate values for radix and factor. The resolution of the time interval is to one radix ^(-factor) of the specified time unit.

EXAMPLE unit="second", radix="10", factor="3" specifies a resolution of milliseconds

The value of the unit is either selected from the units for time intervals from ISO 80000-3, or is another suitable unit. The encoding is defined for GML in `gml:TimeUnitType`:

```
  <simpleType name="TimeUnitType">
    <union>
      <simpleType>
        <restriction base="string">
          <enumeration value="year"/>
          <enumeration value="month"/>
          <enumeration value="day"/>
          <enumeration value="hour"/>
          <enumeration value="minute"/>
          <enumeration value="second"/>
        </restriction>
      </simpleType>
      <simpleType>
        <restriction base="string">
          <pattern value="other:\w{2,}"/>
        </restriction>
      </simpleType>
    </union>
  </simpleType>
```

The second component of this union type provides a method for indicating time units other than the six standard units given in the enumeration.

EXAMPLE To express a period length of 5 days, 14 hours and 30 minutes, any of the following instances are acceptable:

```
<duration>P5DT14H30M</duration>
```



```
<timeInterval unit="hour" radix="10" factor="0">134.5</timeInterval>
<timeInterval unit="other:week" radix="10" factor="0"> 0.800595</timeInterval>
```

14.3 Temporal topology schema

14.3.1 Introduction

Temporal topology is described in terms of time complexes, nodes, and edges, and the connectivity between these. Temporal topology does not directly provide information about temporal position. It is used in the case of describing a lineage or a history (e.g. a family tree expressing evolution of species, an ecological cycle, a lineage of lands or buildings, or a history of separation and merger of administrative boundaries). The following subclause specifies the temporal topology as temporal characteristics of features in compliance with ISO 19108.

14.3.2 Temporal topology objects

14.3.2.1 Overview

A temporal topology object shall be a temporal element that describes the order of features or feature properties as temporal characteristics of features. The two temporal topology objects are primitive and complex.

As time is a one dimensional topological space, temporal topology primitives shall be a time node corresponding to an instant, and a time edge corresponding to a period. A time node is an abstraction of an event that happened at a certain instant as a start or an end of one or more states. A state is a condition — a characteristic of a feature or data set that persists for a period. A "static feature" in this International Standard means a feature that holds a consistent identifier during its life span. Time edge is an abstraction of a state, and associates with time nodes representing its start and end. However, temporal topology primitives do not directly indicate "when" or "how long." A time node need not be a start or an end of a time edge in the case of describing the event not associating with states. Such a node is called an isolated node.

A topology complex is a collection of topological primitives that is closed under the boundary operation. A temporal topology complex shall be a connected acyclic directed graph composed of time edges and time nodes. A minimum temporal topology complex is a time edge with two time nodes at its both ends.

EXAMPLE A lifecycle of a building can be described as a sequence of stages: plan, designing, construction, utilization, disposal and demolition. Each stage can be represented as a time edge. The boundary of each stage describing as a time node represents an event of decision-making, which terminates the stage and also originates the next stage. Thus, a lifecycle of a building is described as a temporal topology complex composed of a sequence of time edges connected with time nodes.

14.3.2.2 AbstractTimeTopologyPrimitive

Temporal topology primitives shall imply the ordering information between features or feature properties. The temporal connection of features can be examined if they have temporal topology primitives as values of their properties. Usually, an instantaneous feature associates with a time node, and a static feature associates with a time edge. A feature with both modes associates with the temporal topology primitive: a supertype of time nodes and time edges.

`gml:TimeTopologyPrimitive` implements ISO 19108 TM_TopologicalPrimitive (see D.2.5.6 and ISO 19108:2002, 5.2.4.2) and acts as the head of a substitution group for temporal topology primitives. It is defined in the schema as follows:

```
<element name="AbstractTimeTopologyPrimitive" type="gml:AbstractTimeTopologyPrimitiveType" abstract="true"
substitutionGroup="gml:AbstractTimePrimitive"/>
```

`gml:AbstractTimeTopologyPrimitive` may be used in any position that a `gml:AbstractTimePrimitive` is valid. Its content model is defined as follows:

```
<complexType name="AbstractTimeTopologyPrimitiveType" abstract="true">
```

```
  <complexContent>
    <extension base="gml:AbstractTimePrimitiveType">
      <sequence>
        <element name="complex" type="gml:ReferenceType" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

A topological primitive is always connected to one or more other topological primitives, and is, therefore, always a member of a topology complex. In a GML instance, this will often be indicated by the primitives being described by elements that are descendents of an element describing a complex. However, in order to support the case where a temporal topology primitive is described in another context, the optional `gml:complex` property is provided, which carries a reference to the parent temporal topology complex.

14.3.2.3 TimeTopologyPrimitivePropertyType

`gml:TimeTopologyPrimitivePropertyType` provides for associating a `gml:AbstractTimeTopologyPrimitive` with an object:

```
<complexType name="TimeTopologyPrimitivePropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractTimeTopologyPrimitive"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

14.3.2.4 TimeTopologyComplex

A temporal topology complex shall be the connected acyclic directed graph composed of temporal topology primitives, i.e. time nodes and time edges. Because a time edge may not exist without two time nodes on its boundaries, static features have time edges from a temporal topology complex as the values of their temporal properties, regardless of explicit declarations.

A temporal topology complex expresses a linear or a non-linear graph. A temporal linear graph, composed of a sequence of time edges, provides a lineage described only by “substitution” of feature instances or feature element values. A time node as the start or the end of the graph connects with at least one time edge. A time node other than the start and the end shall connect to at least two time edges: one of starting from the node, and another ending at the node.

`gml:TimeTopologyComplex` implements ISO 19108 TM_TopologicalComplex (see D.2.5.6 and ISO 19108:2002, 5.2.4.5) and is declared as follows:

```
<element name="TimeTopologyComplex" type="gml:TimeTopologyComplexType"
 substitutionGroup="gml:AbstractTimeComplex"/>
```

`gml:TimeTopologyComplex` may be used in any position that a `gml:AbstractTimeComplex` is valid. Its content model is defined as follows:

```
<complexType name="TimeTopologyComplexType" abstract="true">
  <complexContent>
    <extension base="gml:AbstractTimeComplexType">
      <sequence>
        <element name="primitive" type="gml:TimeTopologyPrimitivePropertyType" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
```

```
  </complexType>
```

A temporal topology complex is a set of connected temporal topology primitives. The member primitives are indicated, either by reference or by value, using the `gml:primitive` property.

14.3.2.5 TimeTopologyComplexPropertyType

`gml:TimeTopologyComplexPropertyType` provides for associating a `gml:TimeTopologyComplex` with an object:

```
<complexType name="TimeTopologyComplexPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:TimeTopologyComplex"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

14.3.2.6 TimeNode

A time node is a zero-dimensional topology primitive that represents an identifiable node in time (it is equivalent to a point in space). A node may act as the termination or initiation of any number of time edges. A time node may be realized as a geometry, its position, whose value is a time instant.

`gml:TimeNode` implements ISO 19108 TM_Node (see D.2.5.6 and ISO 19108:2002, 5.2.4.3) and is declared as follows:

```
  <element name="TimeNode" type="gml:TimeNodeType" substitutionGroup="gml:AbstractTimeTopologyPrimitive"/>
```

`gml:TimeNode` may be used in any position that a `gml:AbstractTimeTopologyPrimitive` is valid. Its content model is defined as follows:

```
  <complexType name="TimeNodeType">
    <complexContent>
      <extension base="gml:AbstractTimeTopologyPrimitiveType">
        <sequence>
          <element name="previousEdge" type="gml:TimeEdgePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <element name="nextEdge" type="gml:TimeEdgePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <element name="position" type="gml:TimeInstantPropertyType" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
```

14.3.2.7 TimeNodePropertyType

`gml:TimeNodePropertyType` provides for associating a `gml:TimeNode` with an object:

```
<complexType name="TimeNodePropertyType">
	<sequence minOccurs="0">
		<element ref="gml:TimeNode"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```


14.3.2.8 TimeEdge

A time edge is a one-dimensional topology primitive. It is an open interval that starts and ends at a node. The edge may be realized as a geometry whose value is a time period.

`gml:TimeEdge` implements ISO 19108 TM_Edge (see D.2.5.6 and ISO 19108:2002, 5.2.4.4) and is declared as follows:

```
<element name="TimeEdge" type="gml:TimeEdgeType" substitutionGroup="gml:AbstractTimeTopologyPrimitive"/>
```

`gml:TimeEdge` may be used in any position that a `gml:AbstractTimeTopologyPrimitive` is valid. Its content model is defined as follows:

```
<complexType name="TimeEdgeType">
  <complexContent>
    <extension base="gml:AbstractTimeTopologyPrimitiveType">
      <sequence>
        <element name="start" type="gml:TimeNodePropertyType"/>
        <element name="end" type="gml:TimeNodePropertyType"/>
        <element name="extent" type="gml:TimePeriodPropertyType" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

14.3.2.9 TimeEdgePropertyType

`gml:TimeEdgePropertyType` provides for associating a `gml:TimeEdge` with an object:

```
<complexType name="TimeEdgePropertyType">
    <sequence minOccurs="0">
        <element ref="gml:TimeEdge"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

14.4 Temporal reference systems

14.4.1 Overview

A value in the time domain is measured relative to a temporal reference system. Common types of reference system include calendars, ordinal temporal reference systems, and temporal coordinate systems (time elapsed since some epoch). The primary temporal reference system for use with geographic information is the Gregorian Calendar and 24 hour local or Coordinated Universal Time (UTC), but special applications may entail the use of alternative reference systems. The Julian day numbering system is a temporal coordinate system that has an origin earlier than any known calendar, at noon on 1 January 4713 BC in the Julian proleptic calendar, and is useful in transformations between dates in different calendars.

In GML seven concrete elements are used to describe temporal reference systems: `gml:TimeReferenceSystem`, `gml:TimeCoordinateSystem`, `gml:TimeCalendar`, `gml:TimeCalendarEra`, `gml:TimeClock`, `gml:TimeOrdinalReferenceSystem`, and `gml:TimeOrdinalEra`.

14.4.2 Basic temporal reference system, TimeReferenceSystem

A reference system is characterized in terms of its domain of validity: the spatial and temporal extent over which it is applicable. The basic GML element for temporal reference systems is `gml:TimeReferenceSystem`. Its content model extends `gml:DefinitionType` (see 15.2.1) with one additional property, `gml:domainOfValidity`. It is implemented as follows:

```
<element name="TimeReferenceSystem" type="gml:TimeReferenceSystemType"
 substitutionGroup="gml:Definition"/>
```

`gml:TimeReferenceSystem` may be used in any position that a `gml:Definition` is valid. Its content model is defined as follows:

```
<complexType name="TimeReferenceSystemType">
  <complexContent>
    <extension base="gml:DefinitionType">
      <sequence>
        <element name="domainOfValidity" type="string"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE This element might appear in an instance document as follows:

```
<gml:TimeReferenceSystem gml:id="JulianCalendar">
  <gml:description xlink:href="http://aa.usno.navy.mil/data/docs/JulianDate.html"/>
  <gml:name>Julian Calendar</gml:name>
  <gml:domainOfValidity>Western Europe</gml:domainOfValidity>
</gml:TimeReferenceSystem>
```

14.4.3 TimeCoordinateSystem

A temporal coordinate system shall be based on a continuous interval scale defined in terms of a single time interval.

`gml:TimeCoordinateSystem` implements ISO 19108 TM_CoordinateSystem (see D.2.5.9 and ISO 19108:2002, 5.3.3) with the exceptions specified below and is declared as follows:

```
<element name="TimeCoordinateSystem" type="gml:TimeCoordinateSystemType"
 substitutionGroup="gml:TimeReferenceSystem"/>
```

`gml:TimeCoordinateSystem` may be used in any position that a `gml:TimeReferenceSystem` is valid. Its content model is defined as follows:

```
<complexType name="TimeCoordinateSystemType">
  <complexContent>
    <extension base="gml:TimeReferenceSystemType">
      <sequence>
        <choice>
          <element name="originPosition" type="gml:TimePositionType"/>
          <element name="origin" type="gml:TimeInstantPropertyType"/>
        </choice>
        <element name="interval" type="gml:TimeIntervalLengthType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```


The differences to ISO 19108 TM_CoordinateSystem are:

— the origin is specified either using the property `gml:originPosition` whose value is a direct time position (see 14.2.2.7), or using the property `gml:origin` whose model is `gml:TimeInstantPropertyType` (see 14.2.2.4); this permits more flexibility in representation and also supports referring to a value fixed elsewhere;

— the interval uses `gml:TimeIntervalLengthType`, defined in 14.2.2.8.

EXAMPLE Coordinate systems might be described in data instances as follows:

```
<gml:TimeCoordinateSystem gml:id="Laser36">
  <gml:description>Time scale used during a laser experiment</gml:description>
  <gml:name>Laser timescale 36</gml:name>
  <gml:domainOfValidity>Laser laboratory</gml:domainOfValidity>
  <gml:origin>
    <gml:TimeInstant>
      <gml:timePosition>2002-11-28T12:50:00+08:00</gml:timePosition>
    </gml:TimeInstant>
  </gml:origin>
  <gml:interval unit="second" radix="10" factor="12">1.0</gml:interval>
</gml:TimeCoordinateSystem>

<gml:TimeCoordinateSystem gml:id="geologyMa">
  <gml:name>Geological time system</gml:name>
  <gml:domainOfValidity>Earth</gml:domainOfValidity>
  <gml:origin>
    <gml:TimeInstant>
      <gml:description xlink:href="http://www.c14dating.com/agecalc.html">Conventional origin used for carbon dating.
Equivalent to "present" for other radiometric dating techniques which have much lower precision.</gml:description>
      <gml:timePosition>1950</gml:timePosition>
    </gml:TimeInstant>
  </gml:origin>
  <gml:interval unit="year" radix="10" factor="-6">1.0</gml:interval>
</gml:TimeCoordinateSystem>
```

14.4.4 Calendars and clocks

14.4.4.1 Overview

Calendars and clocks are both based on interval scales. A calendar is a discrete temporal reference system that provides a basis for defining temporal position to a resolution of one day. A clock provides a basis for defining temporal position within a day. A clock shall be used with a calendar in order to provide a complete description of a temporal position within a specific day.

Calendars have a variety of complex internal structures. This schema defines a simple external calendar interface. Every calendar provides a set of rules for composing a calendar date from a set of elements such as year, month, and day. In every calendar, years are numbered relative to the date of a reference event that defines a calendar era. A single calendar may reference more than one calendar era.

14.4.4.2 TimeCalendar, TimeCalendarEra

A calendar is a discrete temporal reference system that provides a basis for defining temporal position to a resolution of one day. `gml:TimeCalendar` implements ISO 19108 TM_Calender (see D.2.5.8 and ISO 19108:2002, 5.3.2.3) and is declared as follows:

```
<element name="TimeCalendar" type="gml:TimeCalendarType" substitutionGroup="gml:TimeReferenceSystem"/>
```

`gml:TimeCalendar` may be used in any position that a `gml:TimeReferenceSystem` is valid. Its content model is defined as follows:

```
<complexType name="TimeCalendarType">
  <complexContent>
    <extension base="gml:TimeReferenceSystemType">
      <sequence>
        <element name="referenceFrame" type="gml:TimeCalendarEraPropertyType" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

`gml:TimeCalendar` adds one property to those inherited from `gml:TimeReferenceSystem`. A `gml:referenceFrame` provides a link to a `gml:TimeCalendarEra` that it uses. A `gml:TimeCalendar` may reference more than one calendar era.

The referenceFrame element follows the standard GML property model, allowing the association to be instantiated either using an inline description using the `gml:TimeCalendarEra` element, or a link to a `gml:TimeCalendarEra` which is explicit elsewhere.

`gml:TimeCalendarEra` implements ISO 19108 TM_CalenderEra (see D.2.5.8 and ISO 19108:2002, 5.3.2.1) and is declared as follows:

```
<element name="TimeCalendarEra" type="gml:TimeCalendarEraType" />
```

Its content model is defined as follows:

```
<complexType name="TimeCalendarEraType">
  <complexContent>
    <extension base="gml:DefinitionType">
      <sequence>
        <element name="referenceEvent" type="gml:StringOrRefType"/>
        <element name="referenceDate" type="gml:CalDate"/>
        <element name="julianReference" type="decimal"/>
        <element name="epochOfUse" type="gml:TimePeriodPropertyType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

`gml:TimeCalendarEra` inherits basic properties from `gml:DefinitionType` (see 15.2.1) and has the following additional properties:

— `gml:referenceEvent` is the name or description of a mythical or historic event which fixes the position of the base scale of the calendar era. This is given as text or using a link to description held elsewhere.

— `gml:referenceDate` specifies the date of the referenceEvent expressed as a date in the given calendar. In most calendars, this date is the origin (i.e., the first day) of the scale, but this is not always true.

— `gml:julianReference` specifies the Julian date that corresponds to the reference date. The Julian day number is an integer value; the Julian date is a decimal value that allows greater resolution. Transforming calendar dates to and from Julian dates provides a relatively simple basis for transforming dates from one calendar to another.

— `gml:epochOfUse` is the period for which the calendar era was used as a basis for dating.

14.4.4.3 TimeCalendarPropertyType, TimeCalendarEraPropertyType

`gml:TimeCalendarPropertyType` provides for associating a `gml:TimeCalendar` with an object:

```
<complexType name="TimeCalendarPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:TimeCalendar"/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:TimeCalendarEraPropertyType` provides for associating a `gml:TimeCalendarEra` with an object:

```
<complexType name="TimeCalendarEraPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:TimeCalendarEra"/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

14.4.4.4 TimeClock

A clock provides a basis for defining temporal position within a day. A clock shall be used with a calendar in order to provide a complete description of a temporal position within a specific day. `gml:TimeClock` implements ISO 19108 TM_Clock (see D.2.5.8 and ISO 19108:2002, 5.3.2.4) and is declared as follows:

```
<element name="TimeClock" type="gml:TimeClockType" substitutionGroup="gml:TimeReferenceSystem"/>
```

`gml:TimeClock` may be used in any position that a `gml:TimeReferenceSystem` is valid. Its content model is defined as follows:

```
<complexType name="TimeClockType" final="#all">
  <complexContent>
    <extension base="gml:TimeReferenceSystemType">
      <sequence>
        <element name="referenceEvent" type="gml:StringOrRefType"/>
        <element name="referenceTime" type="time"/>
        <element name="utcReference" type="time"/>
        <element name="dateBasis" type="gml:TimeCalendarPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

`gml:TimeClock` adds the following properties to those inherited from `gml:TimeReferenceSystemType`:

— `gml:referenceEvent` is the name or description of an event, such as solar noon or sunrise, which fixes the position of the base scale of the clock.

— `gml:referenceTime` specifies the time of day associated with the reference event expressed as a time of day in the given clock. The reference time is usually the origin of the clock scale.

— `gml:utcReference` specifies the 24 hour local or UTC time that corresponds to the reference time.

— `gml:dateBasis` contains or references the calendars that use this clock.

14.4.4.5 TimeClockPropertyType

`gml:TimeClockPropertyType` provides for associating a `gml:TimeClock` with an object:

```
<complexType name="TimeClockPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:TimeClock"/>
	</sequence>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

14.4.5 Ordinal temporal reference systems

14.4.5.1 Overview

In some applications of geographic information — such as geology and archaeology — relative position in time is known more precisely than absolute time or duration. The order of events in time can be well established, but the magnitude of the intervals between them cannot be accurately determined; in such cases, the use of an ordinal temporal reference system is appropriate. An ordinal temporal reference system is composed of a sequence of named coterminous eras, which may in turn be composed of sequences of member eras at a finer scale, giving the whole a hierarchical structure of eras of varying resolution.

An ordinal temporal reference system whose component eras are not further subdivided is effectively a temporal topology complex constrained to be a linear graph. An ordinal temporal reference system some or all of whose component eras are subdivided is effectively a temporal topology complex with the constraint that parallel branches may only be constructed in pairs where one is a single temporal ordinal era and the other is a sequence of temporal ordinal eras that are called "members" of the "group". This constraint means that within a single temporal ordinal reference system, the relative position of all temporal ordinal eras is unambiguous.

The positions of the beginning and end of a given era may calibrate the relative time scale.

14.4.5.2 TimeOrdinalReferenceSystem, TimeOrdinalEra

`gml:TimeOrdinalReferenceSystem` implements ISO 19108 TM_OrdinalReferenceSystem (see D.2.5.10 and ISO 19108:2002, 5.3.4) by adding one or more `gml:component` properties to the generic temporal reference system model. It is declared as follows:

```
<element name="TimeOrdinalReferenceSystem" type="gml:TimeOrdinalReferenceSystemType"
substitutionGroup="gml:TimeReferenceSystem"/>
```

`gml:TimeOrdinalReferenceSystem` may be used in any position that a `gml:TimeReferenceSystem` is valid. Its content model is defined as follows:

```
<complexType name="TimeOrdinalReferenceSystemType">
  <complexContent>
    <extension base="gml:TimeReferenceSystemType">
      <sequence>
        <element name="component" type="gml:TimeOrdinalEraPropertyType" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

`gml:TimeOrdinalEra` implements ISO 19108 TM_OrdinalEra (see D.2.5.10 and ISO 19108:2002, 5.3.4). Its content model follows the pattern of `gml:TimeEdge` (see 14.3.2.8), inheriting standard properties from `gml:DefinitionType` (see 15.2.1), and adding `gml:start`, `gml:end` and `gml:extent` properties, a set of `gml:member` properties which indicate ordered `gml:TimeOrdinalEra` elements, and a `gml:group` property which points to the parent era. This is declared as follows:

```
<element name="TimeOrdinalEra" type="gml:TimeOrdinalEraType"/>

<complexType name="TimeOrdinalEraType">
  <complexContent>
   <extension base="gml:DefinitionType">
    <sequence>
     <element name="relatedTime" type="gml:RelatedTimeType" minOccurs="0" maxOccurs="unbounded"/>
     <element name="start" minOccurs="0" type="gml:TimeNodePropertyType"/>
     <element name="end" minOccurs="0" type="gml:TimeNodePropertyType"/>
     <element name="extent" type="gml:TimePeriodPropertyType" minOccurs="0"/>
     <element name="member" type="gml:TimeOrdinalEraPropertyType" minOccurs="0" maxOccurs="unbounded"/>
     <element name="group" type="gml:ReferenceType" minOccurs="0"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
```

The recursive inclusion of `gml:TimeOrdinalEra` elements allow the construction of an arbitrary depth hierarchical ordinal reference schema, such that an ordinal era at a given level of the hierarchy includes a sequence of shorter, coterminous ordinal eras.

EXAMPLE The example below shows a portion of the geological time scale depicted as an ordinal reference system:

```
<gml:TimeOrdinalReferenceSystem gml:id="GeologicalTimeScale">
 <gml:description xlink:href="ftp://ftp.iugs.org/pub/iugs/iugs_intstratchart.pdf"/>
 <gml:name>Geological time scale</gml:name>
 <gml:domainOfValidity>Earth</gml:domainOfValidity>
 <!-- Earlier eras omitted -->
 <gml:component>
  <gml:TimeOrdinalEra gml:id="Cenozoic">
   <gml:name>Cenozoic Era</gml:name>
   <gml:start xlink:href="#basePaleocene"/>
   <gml:end xlink:href="#now"/>
   <gml:member>
    <gml:TimeOrdinalEra gml:id="Tertiary">
     <gml:name>Tertiary Period</gml:name>
     <gml:start xlink:href="#baseTertiary"/>
     <gml:end xlink:href="#basePleistocene"/>
     <gml:member>
      <gml:TimeOrdinalEra gml:id="Paleogene">
       <gml:name>Paleogene sub-period</gml:name>
       <gml:start>
        <gml:TimeInstant gml:id="basePaleogene">
         <gml:timePosition frame="#geologyMa">65.0</gml:timePosition>
        </gml:TimeInstant>
       </gml:start>
       <gml:end xlink:href="#baseNeogene"/>
       <gml:member>
        <gml:TimeOrdinalEra gml:id="Paleocene">
         <gml:name>Paleocene Epoch</gml:name>
         <gml:start xlink:href="#basePaleogene"/>
         <gml:end xlink:href="#baseEocene"/>
        </gml:TimeOrdinalEra>
       </gml:member>
       <gml:member>
        <gml:TimeOrdinalEra gml:id="Eocene">
         <gml:name>Paleocene Epoch</gml:name>
         <gml:start >
```



```
              <gml:TimeInstant gml:id="baseEocene">
                <gml:timePosition frame="#geologyMa">57.8</gml:timePosition>
              </gml:TimeInstant>
            </gml:start >
            <gml:end xlink:href="#baseOligocene"/>
          </gml:TimeOrdinalEra>
        </gml:member>
        <gml:member>
          <gml:TimeOrdinalEra gml:id="Oligocene">
            <gml:name>Oligocene Epoch</gml:name>
            <gml:start >
              <gml:TimeInstant gml:id="baseOligocene">
                <gml:timePosition frame="#geologyMa">33.7</gml:timePosition>
              </gml:TimeInstant>
            </gml:start >
            <gml:end xlink:href="#baseNeogene"/>
          </gml:TimeOrdinalEra>
        </gml:member>
      </gml:TimeOrdinalEra>
    </gml:member>
 <!-- Neogene sub-period and Quaternerary period omitted -->
      </gml:TimeOrdinalEra>
    </gml:member>
   </gml:TimeOrdinalEra>
  </gml:component>
</gml:TimeOrdinalReferenceSystem>
```

Note that the use of references on various begin and end elements allows the position of the boundaries between eras to be recorded once and then re-used many times as appropriate, corresponding to a non-linear graph when appropriate. All positions refer to a frame “geologyMa” which would be defined as a temporal coordinate system (e.g see Clause 12).

14.4.5.3 TimeOrdinalEraPropertyType

`gml:TimeOrdinalEraPropertyType` provides for associating a `gml:TimeOrdinalEra` with an object:

```
<complexType name="TimeOrdinalEraPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:TimeOrdinalEra"/>
	</sequence>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

14.5 Representing dynamic features

14.5.1 Overview

A number of types and relationships are defined to represent the time-varying properties of geographic features.

In a comprehensive treatment of spatiotemporal modelling, Langran (see Bibliography) distinguished three principal temporal entities: *states*, *events*, and *evidence*; the schema specified in the following subclauses incorporates elements for each. The conceptual model is shown in D.3.11.

14.5.2 dataSource

In GML, *evidence* is represented by a simple `gml:dataSource` or `gml:dataSourceReference` property that indicates the source of the temporal data.

```
<element name="dataSource" type="gml:StringOrRefType"/>
```



```
<element name="dataSourceReference" type="gml:ReferenceType"/>
```

The remote link attributes of the `gml:dataSource` element have been deprecated along with its current type. To refer to a remote data source, use the remote link attributes of `gml:dataSourceReference` instead.

EXAMPLE A human observer or an *in situ* sensor.

14.5.3 Dynamic properties

A convenience group `gml:dynamicProperties` is defined as follows:

```
<group name="dynamicProperties">
  <sequence>
    <element ref="gml:validTime" minOccurs="0"/>
    <element ref="gml:history" minOccurs="0"/>
    <element ref="gml:dataSource" minOccurs="0"/>
    <element ref="gml:dataSourceReference" minOccurs="0"/>
  </sequence>
</group>
```

This allows an application schema developer to include dynamic properties in a content model in a standard fashion. The `gml:validTime` property is specified in 14.2.1.3. The other properties are specified elsewhere in 14.5.

14.5.4 DynamicFeature

States are captured by time-stamped instances of a feature. `gml:DynamicFeature` implements DynamicFeature as shown in D.3.11 and is declared as follows:

```
<element name="DynamicFeature" type="gml:DynamicFeatureType" substitutionGroup="gml:AbstractFeature"/>
```

The content model extends the standard `gml:AbstractFeatureType` with the `gml:dynamicProperties` model group:

```
<complexType name="DynamicFeatureType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <group ref="gml:dynamicProperties"/>
    </extension>
  </complexContent>
</complexType>
```

Each time-stamped instance represents a ‘snapshot’ of a feature. The dynamic feature classes will normally be extended to suit particular applications. A dynamic feature bears either a time stamp or a history.

NOTE A history consists of a set of `gml:AbstractTimeSlices` and such time slices may contain any time varying properties. We might for example use such a mechanism to describe a feature with one property that varies in time.

14.5.5 DynamicFeatureCollection

`gml:DynamicFeatureCollection` implements DynamicFeatureCollection as shown in D.3.11 and is declared as follows:

```
<element name="DynamicFeatureCollection" type="gml:DynamicFeatureCollectionType"
substitutionGroup="gml:DynamicFeature"/>
```

The content model extends `gml:DynamicFeatureType` with the `gml:dynamicMembers` property:

```
<complexType name="DynamicFeatureCollectionType">
  <complexContent>
    <extension base="gml:DynamicFeatureType">
      <sequence>
        <element ref="gml:dynamicMembers"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="dynamicMembers" type="gml:DynamicFeatureMemberType"/>

<complexType name="DynamicFeatureMemberType">
  <complexContent>
    <extension base="gml:AbstractFeatureMemberType">
      <sequence>
        <element ref="gml:DynamicFeature" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </extension>
  </complexContent>
</complexType>
```

A `gml:DynamicFeatureCollection` is a feature collection that has a `gml:validTime` property (i.e. is a snapshot of the feature collection) or which has a `gml:history` property that contains one or more `gml:AbstractTimeSlices` each of which contain values of the time varying properties of the feature collection. Note that the `gml:DynamicFeatureCollection` may be one of the following:

1. A feature collection which consists of static feature members (members do not change in time) but which has properties of the collection object as a whole that do change in time[6)].

EXAMPLE 1 A Train: The Train is a feature collection. The position and speed of the train are time varying and could be captured in the history of the Train. The featureMembers of the Train are the individual cars including the locomotive. The properties of the cars are static such as the position of the car in the train (we ignore any re-organization of the train in this example), the cargo, the make of the car and its type (e.g. grain car, oil car etc.).

2. A feature collection which consists of dynamic feature members (the members are `gml:DynamicFeatures`) but which also has properties of the collection as a whole that vary in time.

EXAMPLE 2 A collection of sail boats in a yachting race. The sail boats may disappear from the race or reappear. The area encompassing the boats in the race (think of a race like the Vendée Globe) would be time variant.

NOTE One may also have a feature collection with dynamic feature members but such that the properties of the collection as a whole are static. This might also be applied to the sail boat race where we only have properties like the organization committee, and the location of the starting point and finish line.

14.5.6 AbstractTimeSlice

To describe an *event* — an action that occurs at an instant or over an interval of time — GML provides the `gml:AbstractTimeSlice` element, which is declared as follows:

```
<element name="AbstractTimeSlice" type="gml:AbstractTimeSliceType" abstract="true"
substitutionGroup="gml:AbstractGML"/>

<complexType name="AbstractTimeSliceType" abstract="true">
```

6) e.g. described by a history.


```
  <complexContent>
    <extension base="gml:AbstractGMLType">
      <sequence>
        <element ref="gml:validTime"/>
        <element ref="gml:dataSource" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

A timeslice encapsulates the time-varying properties of a dynamic feature – it shall be extended to represent a time stamped projection of a specific feature. The `gml:dataSource` property describes how the temporal data was acquired.

A `gml:AbstractTimeSlice` instance is a GML object that encapsulates updates of the dynamic—or volatile—properties that reflect some change event; it thus includes only those feature properties that have actually changed due to some process.

EXAMPLE 1 Suppose that ownership of a building changes and it is renamed. If no other building properties have changed, then the event will only include the updated name.

`gml:AbstractTimeSlice` basically provides a facility for *attribute-level* time stamping, in contrast to the *object-level* time stamping of dynamic feature instances.

The time slice can thus be viewed as event or process-oriented, whereas a snapshot is more state or structure-oriented. A timeslice has richer causality, whereas a snapshot merely portrays the status of the whole.

EXAMPLE 2 A feature collection might have a 'life cycle' represented by a sequence of snapshots, see Figure 3.

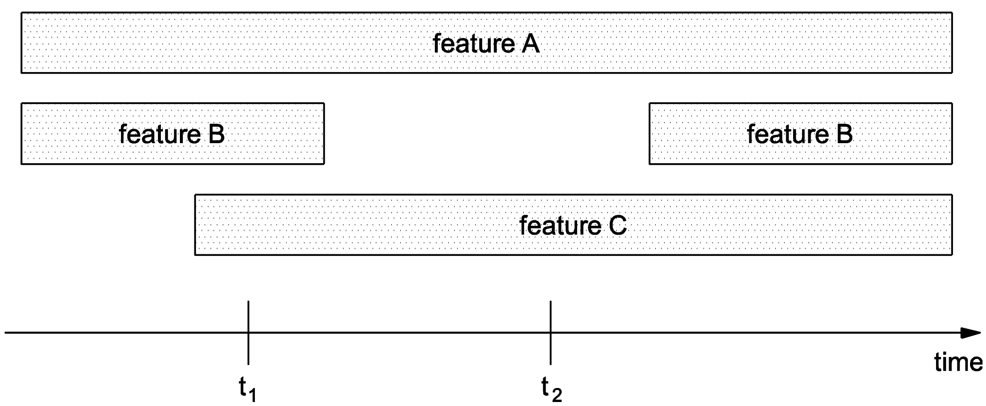


Figure 3 — The life cycle of a feature collection

At instant t_1, feature A, feature B, and feature C are all members of the collection. However, at instant t_2 only feature A and feature C are members. Closer examination of the history of feature B will reveal its ephemeral nature (e.g. a building is dismantled and reconstructed on a seasonal basis).

14.5.7 history

A generic sequence of events constitute a `gml:history` of an object. This property element is declared as follows:

```
<element name="history" type="gml:HistoryPropertyType"/>
```

```
<complexType name="HistoryPropertyType">
  <sequence>
    <element ref="gml:AbstractTimeSlice" maxOccurs="unbounded"/>
  </sequence>
  <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The `gml:history` element contains a set of elements in the substitution group headed by the abstract element `gml:AbstractTimeSlice`, representing the time-varying properties of interest. The history property of a dynamic feature associates a feature instance with a sequence of time slices (i.e. change events) that encapsulate the evolution of the feature.

NOTE The `gml:history` property is intended to capture time varying properties of a feature whose identity is invariant over the lifetime of the temporal model. In this way the detailed evolution of a feature can be described such as its motion in space or changes in its shape. The `gml:history` property may be related to temporal topology objects specified in 14.3. Every `gml:AbstractTimeSlice` in the `gml:history` of a dynamic feature may correspond to a `gml:TimeEdge` in the temporal topology model, if the topology of the valid times of different time slices shall be expressed explicitly. In temporal topology one constructs a temporal topology complex that provides a framework to which one can attach the lineage of a feature or temporal collection of features including dynamic features.

EXAMPLE `gml:MovingObjectStatus` (see I.7.2) is one example of how `gml:AbstractTimeSlice` may be extended to capture the status of a moving object at certain times. The type has been deprecated pending a harmonization with the future ISO 19141 standard (Schema for moving features).

If a feature represents a moving object such as a ground vehicle or a ship, then the `gml:history` property comprises a sequence of `gml:MovingObjectStatus` elements. For example, a dynamic feature such as a cyclone may have a `gml:history` property such as shown in the following fragment:

```
<app:Cyclone gml:id="c1">
  <gml:history>
    <gml:MovingObjectStatus>
      <gml:validTime><gml:TimeInstant>
        <gml:timePosition>2005-11-28T13:00:00Z</gml:timePosition>
      </gml:TimeInstant></gml:validTime>
      <gml:location><gml:Point gml:id="p1" srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
        <gml:pos>-35 140</gml:pos>
      </gml:Point></gml:location>
      <gml:speed uom="#kph">12</gml:speed>
      <gml:bearing>
        <gml:CompassPoint>SE</gml:CompassPoint>
      </gml:bearing>
    </gml:MovingObjectStatus>
    <gml:MovingObjectStatus>
      <gml:validTime><gml:TimeInstant>
        <gml:timePosition>2005-11-28T14:00:00Z</gml:timePosition>
      </gml:TimeInstant></gml:validTIme>
      <gml:location><gml:Point>
        <gml:pos>-34.9 140.1</gml:pos gml:id="p1" srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
      </gml:Point></gml:location>
      <gml:speed uom="#kph">23</gml:speed>
      <gml:bearing>
        <gml:CompassPoint>ESE</gml:CompassPoint>
      </gml:bearing>
    </gml:MovingObjectStatus>
  </gml:history>
</app:Cyclone>
```


15 GML schema — Definitions and dictionaries

15.1 Overview

Many applications require definitions of terms which are used within instance documents as the values of certain properties or as reference information to tie properties to standard information values in some way. Units of measure and descriptions of measurable phenomena are two particular examples.

It will often be convenient to use definitions provided by external authorities. These may already be packaged for delivery in various ways, both online and offline. In order that they may be referred to from GML documents it is generally necessary that a URI be available for each definition. Where this is the case then it is usually preferable to refer to these directly.

Alternatively, it may be convenient or necessary to capture definitions in XML, either embedded within an instance document containing features or as a separate document. The definitions may be transcriptions from an external source, or may be new definitions for a local purpose. In order to support this case, some simple components are provided in GML in the form of

— a generic `gml:Definition`, which may serve as the basis for more specialized definitions

— a generic `gml:Dictionary`, which allows a set of definitions or references to definitions to be collected

These components may be used directly, but also serve as the basis for more specialized definition elements in GML, in particular: coordinate operations (Clause 12), coordinate reference systems (Clause 12), datums (Clause 12), temporal reference systems (Clause 14), and units of measure (Clause 16).

Note that the GML definition and dictionary components implement a simple nested hierarchy of definitions with identifiers. The latter provide handles which may be used in the description of more complex relationships between terms. However, the GML dictionary components are not intended to provide direct support for complex taxonomies, ontologies or thesauri. Specialized XML tools are available to satisfy the more sophisticated requirements.

NOTE The dictionary schema document is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:dictionary:3.2.1

15.2 Dictionary schema

15.2.1 Definition, DefinitionType, remarks

The basic `gml:Definition` element specifies a definition, which can be included in or referenced by a dictionary. It is declared as follows:

```
<element name="Definition" type="gml:DefinitionType" substitutionGroup="gml:AbstractGML"/>

<complexType name="DefinitionBaseType">
	<complexContent>
		<restriction base="gml:AbstractGMLType">
			<sequence>
				<element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
				<element ref="gml:description" minOccurs="0"/>
				<element ref="gml:descriptionReference" minOccurs="0"/>
				<element ref="gml:identifier"/>
				<element ref="gml:name" minOccurs="0" maxOccurs="unbounded"/>
			</sequence>
			<attribute ref="gml:id" use="required"/>
```

```
            </restriction>
        </complexContent>
    </complexType>

    <complexType name="DefinitionType">
        <complexContent>
            <extension base="gml:DefinitionBaseType">
                <sequence>
                    <element ref="gml:remarks" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <element name="remarks" type="string"/>
```

The content model for a generic definition is a derivation from `gml:AbstractGMLType`. The `gml:id` attribute is mandatory for all definitions.

The `gml:description` property element shall hold the definition if this can be captured in a simple text string, or the `gml:descriptionReference` property element may carry a link to a description elsewhere.

The `gml:identifier` element shall provide one identifier identifying this definition. The identifier shall be unique within the dictionaries using this definition.

The `gml:name` elements shall provide zero or more terms and synonyms for which this is the definition.

The `gml:remarks` element shall be used to hold additional textual information that is not conceptually part of the definition but is useful in understanding the definition.

15.2.2 Dictionary, DictionaryType

Sets of definitions may be collected into dictionaries or collections. These are declared in the schema as follows:

```
<element name="Dictionary" type="gml:DictionaryType" substitutionGroup="gml:Definition"/>

<complexType name="DictionaryType">
  <complexContent>
    <extension base="gml:DefinitionType">
      <choice minOccurs="0" maxOccurs="unbounded">
        <element ref="gml:dictionaryEntry"/>
        <element ref="gml:indirectEntry"/>
      </choice>
      <attributeGroup ref="gml:AggregationAttributeGroup"/>
    </extension>
  </complexContent>
</complexType>
```

A `gml:Dictionary` is a non-abstract collection of definitions.

The `gml:Dictionary` content model adds a list of `gml:dictionaryEntry` and *`gml:indirectEntry` (deprecated)* properties that contain *or reference* `gml:Definition` objects. A database handle (`gml:id` attribute) is required, in order that this collection may be referred to. The standard `gml:identifier`, `gml:description`, `gml:descriptionReference` and `gml:name` properties are available to reference or contain more information about this dictionary. The `gml:description` and `gml:descriptionReference` property elements may be used for a description of this dictionary. The derived `gml:name` element may be used for the name(s) of this dictionary.

15.2.3 dictionaryEntry, DictionaryEntryType

These elements contain or refer to the definitions which are members of a dictionary. The element `gml:dictionaryEntry` is declared as follows:

```
<element name="dictionaryEntry" type="gml:DictionaryEntryType"/>

<complexType name="DictionaryEntryType">
	<complexContent>
		<extension base="gml:AbstractMemberType">
			<sequence minOccurs="0">
				<element ref="gml:Definition"/>
			</sequence>
			<attributeGroup ref="gml:AssociationAttributeGroup"/>
		</extension>
	</complexContent>
</complexType>
```

The content model follows the standard GML property pattern, so a `gml:dictionaryEntry` may either contain or refer to a single `gml:Definition`. Since `gml:Dictionary` is substitutable for `gml:Definition`, the content of an entry may itself be a lower-level dictionary.

Note that if the value is provided by reference, this definition does *not* carry a handle (`gml:id`) in this context, so does *not* allow external references to this specific definition in this context. When used in this way the referenced definition will usually be in a dictionary in the same XML document.

15.2.4 Using definitions and dictionaries

Dictionaries and definitions are GML objects, so may be found in independent GML data instance documents.

In application schemas it might be useful to attach a `gml:Dictionary` or `gml:Definition`s to a feature collection in order to record definitions used in properties of members of the collection.

EXAMPLE	The following example shows two instances of dictionaries:

```
 <gml:Dictionary gml:id="rockTypes">
  <gml:description>A simple dictionary of rock types using components from gmlBase</gml:description>
  <gml:identifier codeSpace="http://www.abc.org/terms">Rock Types</gml:identifier>
  <gml:dictionaryEntry>
   <gml:Definition gml:id="granite">
    <gml:description>A igneous rock normally composed of quartz, two feldspars and optional mica</gml:description>
    <gml:identifier codeSpace="http://www.abc.org/terms">Granite</gml:identifier>
   </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
   <gml:Definition gml:id="sst">
    <gml:description>A detrital sedimentary rock normally composed of siliceous grains</gml:description>
    <gml:identifier codeSpace="http://www.abc.org/terms">Sandstone</gml:identifier>
   </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry xlink:href="http://my.big.org/definitions/geology/limestone"/>
</gml:Dictionary>

<gml:Dictionary gml:id="AbridgedGMLdictionary">
  <gml:description>Abridged GML dictionary.</gml:description>
  <gml:identifier codeSpace="http://www.opengis.net/gml/3.2">GML Dictionary</gml:identifier>
  <gml:dictionaryEntry>
   <gml:Definition gml:id="term4.1">
    <gml:description>conceptual schema for data required by one or more applications</gml:description>
```



```
    <gml:identifier codeSpace="http://www.isotc211.org/19101">application schema</gml:identifier>
   </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
   <gml:Definition gml:id="term4.2">
    <gml:description>application schema written in XML Schema in accordance with the rules specified in
ISO 19136</gml:description>
    <gml:identifier codeSpace="http://www.opengis.net/gml/3.2">GML application schema</gml:identifier>
   </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
   <gml:Definition gml:id="term4.3">
    <gml:description>semantic relationship between two or more classifiers that specifies connections among their instances
</gml:description>
    <gml:identifer codeSpace="http://www.uml.org/1.3">association</gml:identifier>
   </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
   <gml:Definition gml:id="term4.4">
    <gml:description>name-value pair contained in an element</gml:description>
    <gml:identifer codeSpace="http://www.w3.org/XML/1998/namespace">attribute</gml:identifier>
   </gml:Definition>
  </gml:dictionaryEntry>
  <!-- … -->
</gml:Dictionary>
```

16 GML schema — Units, measures and values

16.1 Introduction

Several GML schema components concern or require quantitative values which use a reference scale or units of measure. In 8.2 the types `gml:MeasureType`, `gml:MeasureListType` and `gml:MeasureOrNilReasonListType` are defined to enable GML properties and objects to carry units of measure, in accordance with the following pattern:

```
<abc:length uom="m">100</abc:length>
```

The attribute `uom` means "unit of measure" and holds a `gml:UomIdentifier` (see 8.2.3.6).

This clause describes schema components concerning three topics:

— a set of components for defining units of measure,

— a set of typed measures,

— structures for aggregates and lists of measures.

16.2 Units schema

16.2.1 Overview

Several GML schema components concern or require a reference scale or units of measure. Units are required for quantities that may occur as values of properties of feature types, as the results of observations, in the range parameters of a coverage, and for measures used in Coordinate Reference System definitions.

NOTE The schema document units.xsd defines components to support the definition of units of measure. The units schema is listed in Annex C; it is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:units:3.2.1

The basic unit definition is an extension of the general `gml:Definition` element defined in 15.2.1. Three specialized elements for unit definition are further derived from this.

This model is based on the SI system of units (see ISO 1000), which distinguishes between base units and derived units.

— **Base units** are the preferred units for a set of orthogonal fundamental quantities which define the particular system of units, which may not be derived by combination of other base units.

— **Derived units** are the preferred units for other quantities in the system, which may be defined by algebraic combination of the base units.

In some application areas, **conventional units** are used, which may be converted to the preferred units using a scaling factor or a formula which defines a re-scaling and offset. The set of preferred units for all physical quantity types in a particular system of units is composed of the union of its base units and derived units.

16.2.2 Using unit definitions

Unit definitions are substitutable for the `gml:Definition` element declared as part of the dictionary model. A dictionary that contains only unit definitions and references to unit definitions is a units dictionary.

16.2.3 unitOfMeasure, UnitOfMeasureType

The element `gml:unitOfMeasure` is a property element to refer to a unit of measure. It is declared in the schema as follows:

```
<element name="unitOfMeasure" type="gml:UnitOfMeasureType"/>

<complexType name="UnitOfMeasureType">
  <sequence/>
  <attribute name="uom" type="gml:UomIdentifier" use="required"/>
</complexType>
```

This is an empty element which carries a reference to a unit of measure definition (see 8.2.3.6).

EXAMPLE This element may appear in a data instance as follows:

```
<unitOfMeasure uom="m"/>

<unitOfMeasure uom="http://my.standards.org/units/length/metre"/>
```

16.2.4 UnitDefinition, UnitDefinitionType

A `gml:UnitDefinition` is a general definition of a unit of measure. This generic element is used only for units for which no relationship with other units or units systems is known. It is declared in the schema as follows:

```
<element name="UnitDefinition" type="gml:UnitDefinitionType" substitutionGroup="gml:Definition"/>

<complexType name="UnitDefinitionType">
    <complexContent>
        <extension base="gml:DefinitionType">
```



```
            <sequence>
                <element ref="gml:quantityType" minOccurs="0"/>
                <element ref="gml:quantityTypeReference" minOccurs="0"/>
                <element ref="gml:catalogSymbol" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

The content model of `gml:UnitDefinition` adds three additional properties to `gml:Definition` (described in 15.2.1), `gml:quantityType`, `gml:quantityTypeReference` and `gml:catalogSymbol`.

The `gml:catalogSymbol` property optionally gives the short symbol used for this unit. This element is usually used when the relationship of this unit to other units or units systems is unknown.

16.2.5 quantityType, quantityTypeReference

The `gml:quantityType` and `gml:quantityTypeReference` properties indicate the phenomenon to which the units apply. They are declared as follows:

```
<element name="quantityType" type="gml:StringOrRefType"/>

<element name="quantityTypeReference" type="gml:ReferenceType"/>
```

This element contains an informal description of the phenomenon or type of physical quantity that is measured or observed.

EXAMPLE "length", "angle", "time", "pressure", or "temperature".

When the physical quantity is the result of an observation or measurement, this term is known as observable type or measurand.

The use of `gml:quantityType` for references to remote values is deprecated. `gml:quantityTypeReference` shall be used instead.

16.2.6 catalogSymbol

The catalogSymbol is the preferred lexical symbol used for this unit of measure. It is declared as follows:

```
<element name="catalogSymbol" type="gml:CodeType"/>
```

The `codeSpace` attribute in `gml:CodeType` identifies a namespace for the catalog symbol value, and might reference the external catalog. The `string` value in `gml:CodeType` contains the value of a symbol that should be unique within this catalog namespace. This symbol often appears explicitly in the catalog, but it could be a combination of symbols using a specified algebra of units.

EXAMPLE The symbol "cm" might indicate that it is the "m" symbol combined with the "c" prefix.

16.2.7 BaseUnit, BaseUnitType, unitsSystem

A base unit is a unit of measure that cannot be derived by combination of other base units within a particular system of units. For example, in the SI system of units, the base units are metre, kilogram, second, Ampere, Kelvin, mole, and candela, for the physical quantity types length, mass, time interval, electric current, thermodynamic temperature, amount of substance and luminous intensity, respectively.

This is supported using the `gml:BaseUnit` element which is declared as follows:


```
<element name="BaseUnit" type="gml:BaseUnitType" substitutionGroup="gml:UnitDefinition"/>

<complexType name="BaseUnitType">
  <complexContent>
    <extension base="gml:UnitDefinitionType">
      <sequence>
        <element name="unitsSystem" type="gml:ReferenceType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

`gml:BaseUnit` extends generic `gml:UnitDefinition` with the property `gml:unitsSystem`, which carries a reference to the units system to which this base unit is asserted to belong.

16.2.8 DerivedUnit, DerivedUnitType

Derived units are defined by combination of other units. Derived units are used for quantities other than those corresponding to the base units, such as hertz (s^{-1}) for frequency, Newton ($kg.m/s^2$) for force. Derived units based directly on base units are usually preferred for quantities other than the fundamental quantities within a system. If a derived unit is not the preferred unit, the `gml:ConventionalUnit` element (see 16.2.10) should be used instead. The `gml:DerivedUnit` element is declared as follows:

```
<element name="DerivedUnit" type="gml:DerivedUnitType" substitutionGroup="gml:UnitDefinition"/>

<complexType name="DerivedUnitType">
  <complexContent>
    <extension base="gml:UnitDefinitionType">
      <sequence>
        <element ref="gml:derivationUnitTerm" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The `gml:DerivedUnit` extends `gml:UnitDefinition` with the property `gml:derivationUnitTerms`.

16.2.9 derivationUnitTerms, DerivationUnitTermType

A set of `gml:derivationUnitTerm` elements describes a derived unit of measure. Each element carries an integer exponent. The terms are combined by raising each referenced unit to the power of its exponent and forming the product. The element `gml:derivationUnitTerm` is declared as follows:

```
<element name="derivationUnitTerm" type="gml:DerivationUnitTermType"/>

<complexType name="DerivationUnitTermType">
  <complexContent>
    <extension base="gml:UnitOfMeasureType">
      <attribute name="exponent" type="integer"/>
    </extension>
  </complexContent>
</complexType>
```

This unit term references another unit of measure (`uom`) and provides an integer exponent applied to that unit in defining the compound unit. The exponent may be positive or negative, but not zero.

16.2.10 ConventionalUnit, ConventionalUnitType

Conventional units that are neither base units nor defined by direct combination of base units are used in many application domains. For example electronVolt for energy, feet and nautical miles for length. In most cases there is a known, usually linear, conversion to a preferred unit which is either a base unit or derived by direct combination of base units. The `gml:ConventionalUnit` element is declared as follows:

```
<element name="ConventionalUnit" type="gml:ConventionalUnitType" substitutionGroup="gml:UnitDefinition"/>

<complexType name="ConventionalUnitType">
  <complexContent>
    <extension base="gml:UnitDefinitionType">
      <sequence>
        <choice>
          <element ref="gml:conversionToPreferredUnit"/>
          <element ref="gml:roughConversionToPreferredUnit"/>
        </choice>
        <element ref="gml:derivationUnitTerm" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The `gml:ConventionalUnit` extends `gml:UnitDefinition` with a property that describes a conversion to a preferred unit for this physical quantity. When the conversion is exact, the element `gml:conversionToPreferredUnit` should be used, or when the conversion is not exact the element `gml:roughConversionToPreferredUnit` is available. Both of these elements have the same content model. The `gml:derivationUnitTerm` property defined above is included to allow a user to optionally record how this unit may be derived from other ("more primitive") units.

16.2.11 conversionToPreferredUnit, roughConversionToPreferredUnit, ConversionToPreferredUnitType, FormulaType

The elements `gml:conversionToPreferredUnit` and `gml:roughConversionToPreferredUnit` represent parameters used to convert conventional units to preferred units for this physical quantity type. A preferred unit is either a Base Unit or a Derived Unit that is selected for all values of one physical quantity type. These conversions are declared in the schema as follows:

```
<element name="conversionToPreferredUnit" type="gml:ConversionToPreferredUnitType"/>

<element name="roughConversionToPreferredUnit" type="gml:ConversionToPreferredUnitType"/>

<complexType name="ConversionToPreferredUnitType">
  <complexContent>
    <extension base="gml:UnitOfMeasureType">
      <choice>
        <element name="factor" type="double"/>
        <element name="formula" type="gml:FormulaType"/>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

The inherited attribute `uom` references the preferred unit that this conversion applies to. The conversion of a unit to the preferred unit for this physical quantity type is specified by an arithmetic conversion (scaling and/or offset). The content model extends `gml:UnitOfMeasureType`, which has a mandatory attribute `uom` which identifies the preferred unit for the physical quantity type that this conversion applies to. The conversion is specified by a choice of

— `gml:factor`, which defines the scale factor, or

— `gml:formula`, which defines a formula

by which a value using the conventional unit of measure can be converted to obtain the corresponding value using the preferred unit of measure. The model for the formula is given as follows:

```
<complexType name="FormulaType">
  <sequence>
    <element name="a" type="double" minOccurs="0"/>
    <element name="b" type="double"/>
    <element name="c" type="double"/>
    <element name="d" type="double" minOccurs="0"/>
  </sequence>
</complexType>
```

This formula defines the parameters of a simple formula by which a value using the conventional unit of measure can be converted to the corresponding value using the preferred unit of measure. The formula element contains elements a, b, c and d, whose values use the XML Schema type `double`. These values are used in the formula y = (a + bx) / (c + dx), where x is a value using this unit, and y is the corresponding value using the base unit. The elements a and d are optional, and if values are not provided, those parameters are considered to be zero. If values are not provided for both a and d, the formula is equivalent to a fraction with numerator and denominator parameters.

16.2.12 Example of units dictionary <informative>

This dictionary contains definitions corresponding to all the base and derived units defined by in the SI system [SI], plus a selection of conventional units to illustrate the usage of these components.

```
<gml:Dictionary gml:id="unitsDictionary">
  <gml:description>A dictionary of units of measure</gml:description>
  <gml:identifier codeSpace="http://www.opengeospatial.org/initiatives/?iid=79">OWS-1.2 Units</gml:identifier>
  <gml:dictionaryEntry>
    <gml:Dictionary gml:id="SIBaseUnits">
      <gml:description>The Base Units from the SI units system.</gml:description>
      <gml:identifier codeSpace="http://www.opengeospatial.org/initiatives/?iid=79">OWS-1.2 SI Base Units</gml:identifier>
      <gml:dictionaryEntry>
        <gml:BaseUnit gml:id="m">
          <gml:description>The metre is the length of the path travelled by light in vacuum during a time interval of 1/299 792 458
of a second.</gml:description>
          <gml:identifier codeSpace="http://www.bipm.fr/en/3_SI/base_units.html">metre</gml:identifier>
          <gml:name xml:lang="en/US">meter</gml:name>
          <gml:quantityType>length</gml:quantityType>
          <gml:catalogSymbol codeSpace="http://www.bipm.fr/en/3_SI/base_units.html">m</gml:catalogSymbol>
          <gml:unitsSystem xlink:href="http://www.bipm.fr/en/3_SI"/>
        </gml:BaseUnit>
      </gml:dictionaryEntry>
      <!-- … -->
    </gml:Dictionary>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Dictionary gml:id="SIDerivedUnits">
      <gml:description>The Derived Units from the SI units system. These are all derived as a product of SI Base Units, except
degrees Celsius in which the conversion formaula to the SI Base Unit (kelvin) involves an offset. </gml:description>
      <gml:identifier codeSpace="http://www.opengeospatial.org/initiatives/?iid=79">OWS-1.2 SI Derived Units</gml:identifier>
      <gml:dictionaryEntry>
        <gml:DerivedUnit gml:id="rad">
          <gml:identifier codeSpace="http://www.bipm.fr/en/3_SI">radian</gml:identifier>
```



```
        <gml:quantityType>plane angle</gml:quantityType>
        <gml:catalogSymbol codeSpace="http://www.bipm.fr/en/3_SI">rad</gml:catalogSymbol>
        <gml:derivationUnitTerm  uom="#m" exponent="1"/>
        <gml:derivationUnitTerm uom="#m" exponent="-1"/>
      </gml:DerivedUnit>
    </gml:dictionaryEntry>
    <!-- … -->
  </gml:Dictionary>
 </gml:dictionaryEntry>
 <gml:dictionaryEntry>
  <gml:Dictionary gml:id="ConventionalUnitsDictionary">
    <gml:description>A collection of Conventional Units. These are units of measure which are either widely used or important
within a specific community. For most of these there is
1. a known derivation from more primitive units, which may or may not be SI Base Units, or
2. a known conversion to a preferred unit, which may or may not be an SI Base or Derived unit, through rescaling and offset,
or both.</gml:description>
    <gml:identifier codeSpace="http://www.opengeospatial.org/initiatives/?iid=79">OWS-1.2 Conventional
units.</gml:identifier>
    <gml:dictionaryEntry>
      <gml:DerivedUnit gml:id="m3">
        <gml:identifier codeSpace="http://www.opengeospatial.org/initiatives/?iid=79">cubic metre</gml:identifier>
        <gml:quantityType>Volume</gml:quantityType>
        <gml:derivationUnitTerm uom="#m" exponent="3"/>
      </gml:DerivedUnit>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
      <gml:ConventionalUnit gml:id="l">
        <gml:identifier codeSpace="http://www.opengeospatial.org/initiatives/?iid=79">litre</gml:identifier>
        <gml:quantityType>Volume</gml:quantityType>
        <gml:conversionToPreferredUnit uom="#m3">
          <gml:factor>0.001</gml:factor>
        </gml:conversionToPreferredUnit>
      </gml:ConventionalUnit>
    </gml:dictionaryEntry>
    <!-- … -->
  </gml:Dictionary>
 </gml:dictionaryEntry>
</gml:Dictionary>
```

16.3 Measures schema

16.3.1 Overview

`gml:MeasureType` is defined in the basicTypes schema. The measure types defined here correspond with a set of convenience measure types described in ISO/TS 19103. The XML implementation is based on the XML Schema simple type “double” which supports both decimal and scientific notation, and includes an XML attribute “uom” which refers to the units of measure for the value. Note that, there is no requirement to *store* values using any particular format, and applications receiving elements of this type may choose to coerce the data to any other type as convenient.

NOTE The schema document for specific measure types is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:measures:3.2.1

16.3.2 measure

This is the value of a physical quantity, together with its units. It is declared as follows:


```
<element name="measure" type="gml:MeasureType"/>
```

See 8.2.3.6 for the definition of `gml:MeasureType`.

16.3.3 Scalar measure types

A set of specific measure types are defined as vacuous extensions (i.e. aliases) of `gml:MeasureType`. A prototypical definition is as follows:

```
<complexType name="LengthType">
  <simpleContent>
    <extension base="gml:MeasureType"/>
  </simpleContent>
</complexType>
```

This content model supports the description of a length (or distance) quantity, with its units. The unit of measure referenced by uom shall be suitable for a length, such as metres or feet.

The other measure types that are defined following this pattern are: `gml:ScaleType`, `gml:GridLengthType`, `gml:AreaType`, `gml:VolumeType`, `gml:SpeedType`, `gml:TimeType`, and `gml:AngleType` .

EXAMPLE Elements using these content models might appear in a data instance as follows:

```
<my:length uom="m">1.76</my:length>

<my:scale uom="#percent">20.</my:scale>

<my:gridLength uom="#pixelSpacing">480</my:gridLength>
<my:gridLength uom="#imageHeight">0.00208333333333</my:gridLength>

<my:area uom="#ha">1.76</my:area>

<my:volume uom="l">0.45</my:volume>

<my:speed uom="#kmph">73.0</my:speed>

<gml:angle uom="#gradians">95.</gml:angle>

<my:time uom="#minutes">30.</my:time>
```

NOTE Note that the last element in the example addresses the same functional requirements as the elements in the `gml:AbstractTimeLength` substitution group, defined in Clause 14.

16.3.4 angle

The `gml:angle` property element is used to record the value of an angle quantity as a single number, with its units. It is declared as follows:

```
<element name="angle" type="gml:AngleType"/>
```

16.4 Value objects schema

16.4.1 Introduction

The elements declared in this clause build on other GML schema components, in particular `gml:AbstractTimeObject`, `gml:AbstractGeometry`, and the following types: `gml:MeasureType`, gml:MeasureListType, gml:CodeType, gml:CodeOrNilReasonListType, gml:BooleanOrNilReasonListType, gml:IntegerOrNilReasonList.

Of particular interest are elements that are the heads of substitution groups, and one named choice group. These are the primary reasons for the value objects schema, since they may act as variables in the definition of content models, such as Observations, when it is desired to permit alternative value types to occur some of which may have complex content such as arrays, geometry and time objects, and where it is useful not to prescribe the actual value type in advance. The members of the groups include quantities, category classifications, boolean, count, temporal and spatial values, and aggregates of these.

NOTE 1 The schema document valueObjects.xsd describing the components for generic values is listed in Annex C. It is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:valueObjects:3.2.1

NOTE 2 The elements declared in this schema are used for the direct representation of values. Their content models are in general not derived from gml:AbstractGMLType and they do not carry an identifier.

16.4.2 Value element hierarchy

The value objects are defined in a hierarchy. The conceptual model is shown in D.3.15.

The following relationships are defined:

— Concrete elements gml:Quantity, gml:Category, gml:Count and gml:Boolean are substitutable for the abstract element gml:AbstractScalarValue.

— Concrete elements gml:QuantityList, gml:CategoryList, gml:CountList and gml:BooleanList are substitutable for the abstract element gml:AbstractScalarValueList.

— Concrete element gml:ValueArray is substitutable for the concrete element gml:CompositeValue.

— Abstract elements gml:AbstractScalarValue and gml:AbstractScalarValueList, and concrete elements gml:CompositeValue, gml:ValueExtent, gml:CategoryExtent, gml:CountExtent and gml:QuantityExtent are substitutable for abstract element gml:AbstractValue.

— Abstract elements gml:AbstractValue, gml:AbstractTimeObject and gml:AbstractGeometry, and *concrete element gml:Null (deprecated)* are all in a choice group named gml:Value, which is used for compositing in gml:CompositeValue and gml:ValueExtent.

— Schemas which need values may use the abstract element gml:AbstractValue in a content model in order to permit any of the gml:AbstractScalarValues, gml:AbstractScalarValueLists, gml:CompositeValue or gml:ValueExtent to occur in an instance, or the named group gml:Value to also permit gml:AbstractTimeObjects, gml:AbstractGeometrys, and *gml:Nulls (deprecated)*.

16.4.3 Boolean, BooleanList

For recording a value or list of values from two-valued logic, using the XML Schema boolean type; these elements use the following schema declarations:

```
<element name="Boolean" substitutionGroup="gml:AbstractScalarValue" nillable="true">
    <complexType>
        <simpleContent>
            <extension base="boolean">
```



```
                <attribute name="nilReason" type="gml:NilReasonType"/>
            </extension>
        </simpleContent>
    </complexType>
</element>
<element name="BooleanList" type="gml:booleanOrNilReasonList"
substitutionGroup="gml:AbstractScalarValueList"/>
```

`gml:booleanOrNilReasonList` is described in 8.2.4.1.

EXAMPLE In an instance the following examples may be found:

```
<gml:Boolean>1</gml:Boolean>

<gml:Boolean>false</gml:Boolean>

<gml:Boolean xsi:nil="true" nilReason="missing"/>

<gml:BooleanList>1 missing 0 1 1 http://my.big.org/explanations/theDogAteIt 0 1</gml:BooleanList>
```

NOTE These examples illustrate the use of the various Boolean values {1, 0, true, false} and also the fact that nilReason values such as “missing” or a URI may be embedded within a list.

16.4.4 Category, CategoryList

For recording terms representing a classification. These elements use the following schema declarations:

```
<element name="Category" substitutionGroup="gml:AbstractScalarValue" nillable="true">
    <complexType>
        <simpleContent>
            <extension base="gml:CodeType">
                <attribute name="nilReason" type="gml:NilReasonType"/>
            </extension>
        </simpleContent>
    </complexType>
</element>

<element name="CategoryList" type="gml:CodeOrNilReasonListType"
substitutionGroup="gml:AbstractScalarValueList"/>
```

A `gml:Category` has an optional XML attribute `codeSpace`, whose value is a URI which identifies a dictionary, codelist or authority for the term.

EXAMPLE In an instance the following examples may be found:

```
<gml:Category>good</gml:Category>

<gml:Category xsi:nil="true" nilReason="missing"/>

<gml:Category codeSpace="http://my.big.org/dictionaries/rocktypes">Syenite</gml:Category>

<gml:CategoryList codeSpace="http://my.big.org/dictionaries/rocktypes">Syenite Granite missing Tuff</gml:CategoryList>

<gml:CategoryList codeSpace="http://my.big.org/species">bettong numbat phasogale wallaby possum</ gml:CategoryList>
```

16.4.5 Count, CountList

For recording integers representing a rate of occurrence. These elements use the following schema declarations:

```
<element name="Count" substitutionGroup="gml:AbstractScalarValue" nillable="true">
	<complexType>
		<simpleContent>
			<extension base="integer">
				<attribute name="nilReason" type="gml:NilReasonType"/>
			</extension>
		</simpleContent>
	</complexType>
</element>
<element name="CountList" type="gml:integerOrNilReasonList" substitutionGroup="gml:AbstractScalarValueList"/>
```

EXAMPLE In an instance the following examples may be found:

```
<gml:Count>513</gml:Count>

<gml:Count xsi:nil="true" nilReason="missing"/>

<gml:CountList>34 56 2 inapplicable 153</gml:CountList>
```

16.4.6 Quantity, QuantityList

For recording numeric values with a scale. The content of the element is an amount using the XML Schema type double which permits decimal or scientific notation. These elements use the following schema declarations:

```
<element name="Quantity" substitutionGroup="gml:AbstractScalarValue" nillable="true">
	<complexType>
		<simpleContent>
			<extension base="gml:MeasureType">
				<attribute name="nilReason" type="gml:NilReasonType"/>
			</extension>
		</simpleContent>
	</complexType>
</element>
<element name="QuantityList" type="gml:MeasureOrNilReasonListType"
 substitutionGroup="gml:AbstractScalarValueList"/>
```

An XML attribute `uom` ("unit of measure") is required, whose value is a URI which identifies the definition of a ratio scale or units by which the numeric value shall be multiplied, or an interval or position scale on which the value occurs.

EXAMPLE In an instance the following examples may be found:

```
<gml:Quantity uom="m">4.32e-4</gml:Quantity>

<gml:Quantity xsi:nil="true" nilReason="withheld"/>

<gml:QuantityList uom="#C">21. 37. withheld 25.</gml:QuantityList>
```

16.4.7 AbstractValue, AbstractScalarValue, AbstractScalarValueList

`gml:AbstractValue` is an abstract element which acts as the head of a substitution group which contains `gml:AbstractScalarValue`, `gml:AbstractScalarValueList`, `gml:CompositeValue` and `gml:ValueExtent`, and (transitively) the elements in their substitution groups.

`gml:AbstractScalarValue` is an abstract element which acts as the head of a substitution group which contains `gml:Boolean`, `gml:Category`, `gml:Count` and `gml:Quantity`, and (transitively) the elements in their substitution groups.

`gml:AbstractScalarValueList` is an abstract element which acts as the head of a substitution group which contains `gml:BooleanList`, `gml:CategoryList`, `gml:CountList` and `gml:QuantityList`, and (transitively) the elements in their substitution groups.

These elements use the following schema declarations:

```
<element name="AbstractValue" type="anyType" abstract="true" substitutionGroup="gml:AbstractObject"/>

<element name="AbstractScalarValue" type="anyType" abstract="true" substitutionGroup="gml:AbstractValue"/>

<element name="AbstractScalarValueList" type="anyType" abstract="true" substitutionGroup="gml:AbstractValue"/>
```

These elements may be used in an application schema as variables, so that in an XML instance document any member of its substitution group may occur.

16.4.8 Value

This is a convenience choice group which unifies generic values defined in this clause with spatial and temporal objects and the measures described above, so that any of these may be used within aggregate values. This element uses the following schema declaration:

```
<group name="Value">
  <choice>
    <element ref="gml:AbstractValue"/>
    <element ref="gml:AbstractGeometry"/>
    <element ref="gml:AbstractTimeObject"/>
    <element ref="gml:Null"/>
  </choice>
</group>
```

16.4.9 valueProperty, valueComponent, valueComponents

Elements that instantiates a GML property which refers to, or contains, a Value or Values; these elements use the following schema declarations:

```
<element name="valueProperty" type="gml:ValuePropertyType"/>

<element name="valueComponent" type="gml:ValuePropertyType"/>

<complexType name="ValuePropertyType">
	<sequence minOccurs="0">
		<group ref="gml:Value"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>

  <element name="valueComponents" type="gml:ValueArrayPropertyType"/>

<complexType name="ValueArrayPropertyType">
	<sequence maxOccurs="unbounded">
		<group ref="gml:Value" />
	</sequence>
	<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

Note that both `gml:ValuePropertyType` and `gml:ValueArrayPropertyType` have the group named `gml:Value` as their content. This means that any of the elements in the `gml:Value` choice group, or in the substitution groups of the members of the choice group may occur as the content of a value property.

The `gml:valueProperty` element is a convenience element for general use. The `gml:valueComponent` and `gml:valueComponents` elements are specifically used in compositing.

16.4.10 CompositeValue

`gml:CompositeValue` is an aggregate value built from other values . It contains zero or an arbitrary number of `gml:valueComponent` elements, and zero or one `gml:valueComponents` property elements. It may be used for strongly coupled aggregates (vectors, tensors) or for arbitrary collections of values. This element uses the following schema declarations:

```
<element name="CompositeValue" type="gml:CompositeValueType" substitutionGroup="gml:AbstractValue"/>

<complexType name="CompositeValueType">
  <complexContent>
    <extension base="gml:AbstractGMLType">
      <sequence>
        <element ref="gml:valueComponent" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="gml:valueComponents" minOccurs="0"/>
      </sequence>
      <attributeGroup ref="gml:AggregationAttributeGroup"/>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE In an instance a `gml:CompositeValue` may appear as in the following examples:

```
<gml:CompositeValue>
  <gml:valueComponent>
    <gml:QuantityList uom="#C">21. 37. withheld 25.</gml:QuantityList>
  </gml:valueComponent>
  <gml:valueComponent>
    <gml:Category>good</gml:Category>
  </gml:valueComponent>
  <gml:valueComponent>
    <gml:Count xsi:nil="true" nilReason="missing"/>
  </gml:valueComponent>
  <gml:valueComponents>
    <gml:Point srsName="http://www.opengis.net/def/crs/EPSG/0/4326"><gml:pos>71. -32.</gml:pos></gml:Point>
    <gml:Point srsName="http://www.opengis.net/def/crs/EPSG/0/4326"><gml:pos>70. -35.</gml:pos></gml:Point>
    <gml:Point srsName="http://www.opengis.net/def/crs/EPSG/0/4326"><gml:pos>74. -37.</gml:pos></gml:Point>
  </gml:valueComponents>
</gml:CompositeValue>

<gml:CompositeValue>
  <gml:valueComponents>
    <gml:Point srsName="http://www.opengis.net/def/crs/EPSG/0/4326"><gml:pos>-67.563 -13.834</gml:pos></gml:Point>
    <gml:Quantity uom="#km">632.</gml:Quantity>
    <gml:TimeInstant><gml:timePosition>1994-06-09T00:33:16.4</gml:timePosition></gml:TimeInstant>
    <gml:Quantity uom="#mom">-1.00</gml:Quantity>
    <gml:Quantity uom="#mom">0.92</gml:Quantity>
    <gml:Quantity uom="#mom">0.09</gml:Quantity>
    <gml:Quantity uom="#mom">-1.69</gml:Quantity
    <gml:Quantity uom="#mom">-0.09</gml:Quantity>
    <gml:Quantity uom="#mom">-0.37</gml:Quantity>
  </gml:valueComponents>
```

```
</gml:CompositeValue>
```

16.4.11 ValueArray

A Value Array is used for homogeneous arrays of primitive and aggregate values.

The member values may be scalars, composites, arrays or lists. This element uses the following schema declarations:

```
<element name="ValueArray" type="gml:ValueArrayType" substitutionGroup="gml:CompositeValue"/>

<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xlink="http://www.w3.org/1999/xlink" xml:lang="en">
        <sch:title>Schematron constraints for GML / ISO 19136</sch:title>
        <sch:ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron"/>
        <sch:ns prefix="gml" uri="http://www.opengis.net/gml/3.2"/>
        <sch:pattern>
                <sch:rule context="gml:ValueArray">
                        <sch:assert test="not(@codeSpace and @uom)">ValueArray may not carry both a reference to a
codeSpace and a uom</sch:assert>
                </sch:rule>
        </sch:pattern>
        <sch:pattern>
                <sch:rule context="gml:ValueArray">
                        <sch:assert test="count(gml:valueComponent/*) = count(gml:valueComponent/*[name() =
name(../../gml:valueComponent[1]/*[1])])">All components shall be of the same type</sch:assert>
                        <sch:assert test="count(gml:valueComponents/*) = count(gml:valueComponents/*[name() =
name(../*[1])])">All components shall be of the same type</sch:assert>
                </sch:rule>
        </sch:pattern>
</sch:schema>

<complexType name="ValueArrayType">
   <complexContent>
     <extension base="gml:CompositeValueType">
       <attributeGroup ref="gml:referenceSystem"/>
     </extension>
   </complexContent>
</complexType>

<attributeGroup name="referenceSystem">
   <attribute name="codeSpace" type="anyURI" />
   <attribute name="uom" type="gml:UomIdentifier" />
</attributeGroup>
```

ValueArray has the same content model as CompositeValue, but the member values shall be homogeneous. The element declaration contains a Schematron constraint which expresses this restriction precisely. Since the members are homogeneous, the `gml:referenceSystem` (`uom`, `codeSpace`) may be specified on the `gml:ValueArray` itself and inherited by all the members if desired.

EXAMPLE 1 The `gml:ValueArray` element may appear in instances as follows. In the first example a set of points are each the value of a `gml:valueComponent` property. One of the values is provided by-reference, using the standard xlink:href syntax:

```
<gml:ValueArray>
  <gml:valueComponent>
    <gml:Point srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
      <gml:pos>-32. 71.</gml:pos>
    </gml:Point>
  </gml:valueComponent>
```



```
  <gml:valueComponent>
    <gml:Point srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
      <gml:pos>-35. 70.</gml:pos>
    </gml:Point>
  </gml:valueComponent>
  <gml:valueComponent xlink:href="http://my.big.org/locations/points/point456"/>
</gml:ValueArray>
```

EXAMPLE 2 In the second example a set of quantities are contained within a `gml:valueComponents` property. One of the values is not available, indicated by a nil value:

```
<gml:ValueArray>
  <gml:valueComponents>
    <gml:Quantity uom="#C">21.</gml:Quantity>
    <gml:Quantity uom="#C">37.</gml:Quantity>
    <gml:Quantity xsi:nil="true" nilReason="missing"/>
  </gml:valueComponents>
</gml:ValueArray>
```

EXAMPLE 3 Note that a `gml:AbstractScalarValueList` is usually preferred for arrays of scalar values since this is a more efficient encoding. The information in the previous example may be expressed:

```
<gml:QuantityList uom="#C">21. 37. missing</gml:QuantityList>
```

However, if the values of the components are not scalars, then the explicit form is required.

16.4.12 Typed ValueExtents: CategoryExtent, CountExtent, QuantityExtent

Three elements are provided for typed value extents, for categories, counts and quantities. Their content models are defined by restricting the relevant scalar list types to contain exactly two items as follows:

```
<element name="CategoryExtent" type="gml:CategoryExtentType" substitutionGroup="gml:AbstractValue"/>

  <complexType name="CategoryExtentType">
    <simpleContent>
      <restriction base="gml:CodeOrNilReasonListType">
        <length value="2"/>
      </restriction>
    </simpleContent>
  </complexType>

  <element name="CountExtent" type="gml:CountExtentType" substitutionGroup="gml:AbstractValue"/>

  <simpleType name="CountExtentType">
    <restriction base="gml:integerOrNilReasonList">
      <length value="2"/>
    </restriction>
  </simpleType>

  <element name="QuantityExtent" type="gml:QuantityExtentType" substitutionGroup="gml:AbstractValue"/>

  <complexType name="QuantityExtentType">
    <simpleContent>
      <restriction base="gml:MeasureOrNilReasonListType">
        <length value="2"/>
      </restriction>
    </simpleContent>
  </complexType>
```

A `gml:QuantityExtent` element or another element using this type will contain two values and a scale.

EXAMPLE 1 <gml:QuantityExtent uom=”#mm”>0. 9.5</gml:QuantityExtent>

An element of `gml:CategoryExtentType` is useful if the codeSpace defines a set of ordered terms.

EXAMPLE 2 <my:AgeRange codeSpace=”http://iugg.org/geologicalPeriods”>Cambrian Devonian</my:AgeRange>

Any value extent may describe a single-ended interval by using a NilReason value for one of the limits.

EXAMPLE 3 <gml:CountExtent>53 inapplicable</gml:CountExtent> describes the integers starting with 53.

16.4.13 BooleanPropertyType, CategoryPropertyType, CountPropertyType, QuantityPropertyType

A set of convenience types (`gml:BooleanPropertyType`, `gml:CategoryPropertyType`, `gml:CountPropertyType`, `gml:QuantityPropertyType`) are provided for properties whose content is a specific member of the `gml:AbstractScalarValue` substitution group. Their definitions follow the same pattern, as exemplified by the definition of `gml:BooleanPropertyType:`

```
<complexType name="BooleanPropertyType">
   <sequence minOccurs="0">
      <element ref="gml:Boolean"/>
   </sequence>
   <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

17 GML schema — Directions

17.1 Direction schema

The direction schema components provide the GML application schema developer with a standard property element to describe direction, and associated objects that may be used to express orientation, direction, heading, bearing or other directional aspects of geographic features.

NOTE The corresponding schema document is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:direction:3.2.1

17.2 direction, DirectionPropertyType

The property `gml:direction` is intended as a predefined property expressing a direction to be assigned to features defined in a GML application schema. It is declared as follows:

```
<element name="direction" type="gml:DirectionPropertyType"/>

<complexType name="DirectionPropertyType">
    <sequence minOccurs="0">
        <choice>
            <element name="DirectionVector" type="gml:DirectionVectorType"/>
            <element name="DirectionDescription" type="gml:DirectionDescriptionType"/>
            <element name="CompassPoint" type="gml:CompassPointEnumeration"/>
            <element name="DirectionKeyword" type="gml:CodeType"/>
            <element name="DirectionString" type="gml:StringOrRefType"/>
        </choice>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```


The two alternative kinds of direction specifications, a vector or a description, are specified in the following subclauses.

17.3 DirectionVectorType

Direction vectors are specified by providing components of a vector as follows:

```
<complexType name="DirectionVectorType">
  <choice>
    <element ref="gml:vector"/>
    <sequence>
      <element name="horizontalAngle" type="gml:AngleType"/>
      <element name="verticalAngle" type="gml:AngleType"/>
    </sequence>
  </choice>
</complexType>
```

The `gml:vector` element is described in 10.1.4.5.

EXAMPLE This form may appear in a data instance as follows:

```
<gml:direction>
  <gml:DirectionVector>
    <gml:vector srsName="#wgs84">0.0 45.0</gml:vector>
  </gml:DirectionVector>
</gml:direction>
```

The use of the alternative representation via angles has been deprecated, gml:vector shall be used instead.

17.4 DirectionDescriptionType

Direction descriptions are specified by a compass point code, a keyword, a textual description or a reference to a description. The `gml:DirectionDescriptionType` element is declared as follows:

```
<complexType name="DirectionDescriptionType">
  <choice>
    <element name="compassPoint" type="gml:CompassPointEnumeration"/>
    <element name="keyword" type="gml:CodeType"/>
    <element name="description" type="string"/>
    <element name="reference" type="gml:ReferenceType"/>
  </choice>
</complexType>
```

A `gml:compassPoint` is specified by a simple enumeration string type that is declared as follows:

```
<simpleType name="CompassPointEnumeration">
  <restriction base="string">
    <enumeration value="N"/>
    <enumeration value="NNE"/>
    <enumeration value="NE"/>
    <enumeration value="ENE"/>
    <enumeration value="E"/>
    <enumeration value="ESE"/>
    <enumeration value="SE"/>
    <enumeration value="SSE"/>
    <enumeration value="S"/>
    <enumeration value="SSW"/>
    <enumeration value="SW"/>
    <enumeration value="WSW"/>
```

```
      <enumeration value="W"/>
      <enumeration value="WNW"/>
      <enumeration value="NW"/>
      <enumeration value="NNW"/>
    </restriction>
  </simpleType>
```

These directions are necessarily approximate, giving direction with a precision of 22.5°. It is thus generally unnecessary to specify the reference frame, though this may be detailed in the definition of a GML application language.

EXAMPLE 1 This form may appear in a data instance as follows:

```
<gml:direction>
  <gml:DirectionDescription>
    <gml:compassPoint>WNW</gml:compassPoint>
  </gml:DirectionDescription>
</gml:direction>
```

In addition, the elements to contain text-based descriptions of direction are provided.

If the direction is specified using a term from a list, `gml:keyword` should be used, and the list indicated using the value of the codeSpace attribute.

EXAMPLE 2 This form may appear in a data instance as follows:

```
<gml:direction>
  <gml:DirectionDescription>
    <gml:keyword codeSpace="http://my.big.org/terms/direction">onshore</gml:keyword>
  </gml:DirectionDescription>
</gml:direction>
```

If the direction is described in prose, `gml:direction` or `gml:reference` should be used, allowing the value to be included inline or by reference.

EXAMPLE 3 This form may appear in a data instance as follows:

```
<gml:direction>
  <gml:DirectionDescription>
    <gml:direction>Towards the lighthouse</gml:direction>
  </gml:DirectionDescription>
</gml:direction>

<gml:direction>
  <gml:DirectionDescription>
    <gml:reference xlink:href="http://my.big.org/logbook/20021127/paragraph6"/>
  </gml:DirectionDescription>
</gml:direction>
```

18 GML schema — Observations

18.1 Observations

A GML observation models the act of observing, often with a camera, a person or some form of instrument. An observation feature describes the “metadata” associated with an information capture event, together with a value for the result of the observation. This covers a broad range of cases, including tourist photos (not the photo but the act of taking the photo).

NOTE This schema is primarily intended to serve for "simple" observations. Schemas for scientific, technical and engineering observations and measurements will typically require the development of a GML application schema for such observations. See, for example, the Observations and Measurements specification from the Open Geospatial Consortium.

18.2 Observation schema

18.2.1 Overview

This clause describes two kinds of observations, `gml:Observation` and `gml:DirectedObservation`.

NOTE Observations are described in the schema document observations.xsd. The schema is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:observation:3.2.1

18.2.2 Observation

The `gml:Observation` element is declared in the schema as follows:

```
<element name="Observation" type="gml:ObservationType" substitutionGroup="gml:AbstractFeature"/>

<complexType name="ObservationType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element ref="gml:validTime"/>
        <element ref="gml:using" minOccurs="0"/>
        <element ref="gml:target" minOccurs="0"/>
        <element ref="gml:resultOf"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The content model is a straightforward extension of `gml:AbstractFeatureType`; it automatically has the `gml:identifier`, `gml:description`, `gml:descriptionReference`, `gml:name`, and `gml:boundedBy` properties.

The `gml:validTime` element is declared in 14.2.1.3. In this context it describes the time of the observation. Note that this may be a time instant or a time period.

EXAMPLE Some examples of simple observations are as follows:

```
<gml:Observation>
  <gml:validTime>
    <gml:TimeInstant>
      <gml:timePosition>2002-11-12T09:12:00</gml:timePosition>
    </gml:TimeInstant>
  </gml:validTime>
  <gml:using xlink:href="http://www.my.org/sensors/thermometer4"/>
  <gml:target xlink:href="http://www.environment.org/stations/l456"/>
  <gml:resultOf>
    <gml:Quantity uom="#C">18.4</gml:Quantity>
  </gml:resultOf>
</gml:Observation>

<gml:Observation>
```

```
  <gml:validTime>
    <gml:TimeInstant>
      <gml:timePosition>2002-11-12T09:12:00</gml:timePosition>
    </gml:TimeInstant>
  </gml:validTime>
  <gml:using xlink:href="http://www.my.org/cameras/leica2"/>
  <gml:subject xlink:href="http://www.tourist.org/sights/mountain3"/>
  <gml:resultOf xlink:href="http://www.my.org/photos/landscape1.jpg"/>
</gml:Observation>

<gml:Observation>
  <gml:validTime>
    <gml:TimeInstant>
      <gml:timePosition>2002-10-25T11:37:25</gml:timePosition>
    </gml:TimeInstant>
  </gml:validTime>
  <gml:subject xlink:href="http://www.people.org/kids/abby"/>
  <gml:resultOf xlink:href="myDaughtersPortrait.jpg"/>
</gml:Observation>
```

18.2.3 using

The `gml:using` property contains or references a description of a procedure (such as a camera) used for the observation. It is declared as follows:

```
  <element name="using" type="gml:ProcedurePropertyType"/>

  <complexType name="ProcedurePropertyType">
    <sequence minOccurs="0">
      <element ref="gml:AbstractFeature"/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
```

18.2.4 target

The `gml:target` property contains or references the specimen, region or station which is the object of the observation. This property element is declared in the schema as follows:

```
  <element name="target" type="gml:TargetPropertyType"/>

  <element name="subject" type="gml:TargetPropertyType" substitutionGroup="gml:target"/>

  <complexType name="TargetPropertyType">
    <choice minOccurs="0">
      <element ref="gml:AbstractFeature"/>
      <element ref="gml:AbstractGeometry"/>
    </choice>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
```

This property is particularly useful for remote observations, such as photographs, where a generic location property might apply to the location of the camera or the location of the field of view, and thus may be ambiguous.

The `gml:subject` element is provided as a convenient synonym for `gml:target`. This is the term commonly used in photography.

NOTE gml:Observation does not contain a predefined location property. If the schema developer wishes to specify a concrete location for the observation point (location of the sensor) would do so through a location property, e.g. with a point as a value. In the case where the target has a known direction but unknown distance to the observation point (remote sensing) gml:DirectedObservation should be used. Where the relative direction and distance are known, gml:DirectedObservationAtDistance should be used.

EXAMPLE An application defined observation feature type with a location of the observation point could be specified as

```
<element name="ObservationWithSensorLocation" type="app:ObservationWithSensorLocationType"
substitutionGroup="gml:Observation"/>
  <complexType name="ObservationWithSensorLocationType">
    <complexContent>
      <extension base="gml:ObservationType">
        <sequence>
          <element name="positionOfSensor" type="gml:PointPropertyType"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
```

18.2.5 resultOf

The gml:resultOf property indicates the result of the observation. The value may be inline, or a reference to a value elsewhere. It is declared in the schema as follows:

```
  <element name="resultOf" type="gml:ResultType"/>

  <complexType name="ResultType">
    <sequence minOccurs="0">
      <any namespace="##any"/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
```

EXAMPLE A result property with a gml:DataBlock recording the observation of a temperature and a pressure measurement.

```
<gml:DataBlock>
	<gml:rangeParameters>
		<gml:CompositeValue>
			<gml:valueComponents>
				<Temperature uom="Cel">template</Temperature>
				<Pressure uom= kPa">template</Pressure>
			</gml:valueComponents>
		</gml:CompositeValue>
	</gml:rangeParameters>
	<gml:tupleList>3,101.2</gml:tupleList>
</gml:DataBlock>
```

18.2.6 DirectedObservation

A gml:DirectedObservation is the same as an observation except that it adds an additional gml:direction property. This is the direction in which the observation was acquired. Clearly this applies only to certain types of observation such as visual observations by people, or observations obtained from terrestrial cameras.

```
  <element name="DirectedObservation" type="gml:DirectedObservationType"
```



```
    substitutionGroup="gml:Observation"/>

<complexType name="DirectedObservationType">
  <complexContent>
    <extension base="gml:ObservationType">
      <sequence>
        <element ref="gml:direction"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

EXAMPLE:

```
<gml:DirectedObservation>
  <gml:validTime>
    <gml:TimeInstant>
      <gml:timePosition>2002-11-12T09:12:00</gml:timePosition>
    </gml:TimeInstant>
  </gml:validTime>
  <gml:using xlink:href="http://www.my.org/cameras/leica2"/>
  <gml:target xlink:href="http://www.tourist.org/sights/mountain3"/>
  <gml:resultOf xlink:href="http://www.my.org/photos/landscape1.jpg"/>
  <gml:direction>
    <gml:DirectionVector>
      <gml:vector srsName="http://www.opengis.net/def/crs/EPSG/0/4326">0.0 45.0</gml:vector>
    </gml:DirectionVector>
  </gml:direction>
</gml:DirectedObservation>
```

18.2.7 DirectedObservationAtDistance

`gml:DirectedObservationAtDistance` adds an additional distance property. This is the distance from the observer to the subject of the observation. Clearly this applies only to certain types of observation such as visual observations by people, or observations obtained from terrestrial cameras.

```
  <element name="DirectedObservationAtDistance" type="gml:DirectedObservationAtDistanceType"
      substitutionGroup="gml:DirectedObservation "/>

  <complexType name="DirectedObservationAtDistanceType">
    <complexContent>
      <extension base="gml:DirectedObservationType">
        <sequence>
          <element name="distance" type="gml:MeasureType"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
```

EXAMPLE:

```
<gml:DirectedObservationAtDistance>
  <gml:validTime>
    <gml:TimeInstant>
      <gml:timePosition>2002-11-12T09:12:00</gml:timePosition>
    </gml:TimeInstant>
  </gml:validTime>
  <gml:using xlink:href="http://www.my.org/cameras/leica2"/>
  <gml:subject xlink:href="http://www.tourist.org/sights/mountain3"/>
  <gml:resultOf xlink:href="http://www.my.org/photos/landscape1.jpg"/>
  <gml:direction>
    <gml:DirectionVector>
      <gml:vector srsName="http://www.opengis.net/def/crs/EPSG/0/4326">0.0 45.0</gml:vector>
    </gml:DirectionVector>
  </gml:direction>
  <gml:distance uom="m">16500.</gml:distance>
</gml:DirectedObservationAtDistance>
```

19 GML schema — Coverages

19.1 The coverage model and representations

19.1.1 General remarks

This clause defines the GML encoding for coverages and is in agreement with the conceptual model outlined in ISO 19123.

ISO 19123 provides a definition:

> *Coverages support mapping from a spatiotemporal domain to attribute values where attribute types are common to all geographic positions within the spatiotemporal domain. A spatiotemporal domain consists of a collection of direct positions in a coordinate space. Examples of coverages include rasters, triangulated irregular networks, point coverages, and polygon coverages. Coverages are the prevailing data structures in a number of application areas, such as remote sensing, meteorology, and bathymetric, elevation, soil, and vegetation mapping.*

The information describing a coverage is conventionally represented in one of two ways:

a) As a set of discrete location-value pairs.

b) As a description of the spatio-temporal domain (multi-geometry, grid) and a description of the set of values from the range, together with a method or rule (which may be implicit) that assigns a value from the range set to each position within the domain.

The first method only applies to domains that are partitioned into discrete components. This representation may be realized in GML as a **homogeneous feature collection** (i.e. all the features have the same set of properties), where the set of locations from the features compose the domain and the set of property values compose the range. The mapping from domain to range is trivial: the properties on each feature are assigned to the location of that feature. For coverages whose domain is composed of a large set of locations this explicit representation may, however, be bulky.

The second method is more flexible in a number of ways.

— Since the domain and range are homogeneous sets, there may be efficiencies in the representation of either or both domain and range.

— The values in the range may be represented in an analytic form rather than as discrete explicit values, which is also related to the fact that as discrete explicit values.

— When the attribute values vary continuously across the domain, a functional form covering the complete domain is required to be able to provide values of the range at arbitrary locations. The function typically involves interpolation, possibly using a process model.

The first representation is typically used during data collection where a set or properties relating to a single location are managed together, or update of a datastore where only a small number of features are manipulated at one time. The second representation is more suitable for analysis, where spatio-temporal patterns and anomalies within a specific property are of interest.

It is the second method, using a functional map over the whole domain, which is the subject of the GML coverage encoding.

19.1.2 Formal description of a coverage

A coverage incorporates a mapping from a spatiotemporal domain to a range set, the latter providing the set in which the attribute values live. The range set may be an arbitrary set including discrete lists, integer or floating point ranges, and multi-dimensional vector spaces. This conceptual model of a coverage is described in Figure 4.

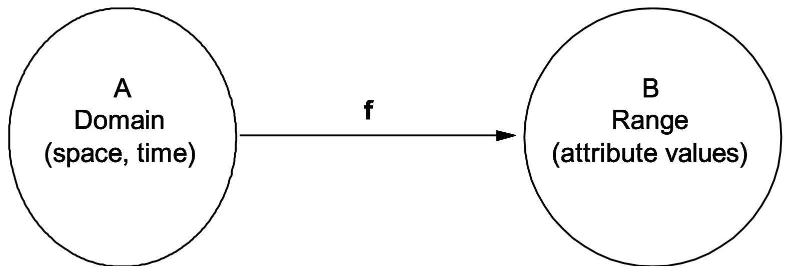


Figure 4 — Conceptual model of a coverage

A coverage can be viewed as the graph of the coverage function $f{:}A \rightarrow B$, that is as the set of ordered pairs

$$\{(x, f(x)) \mid \text{where } x \text{ is in } A\}$$

This view is especially applicable to the GML encoding of a coverage. In the case of a discrete coverage, the domain set A is partitioned into a collection of subsets (typically a disjoint collection) $A = UA_i$ and the function f is constant on each A_i. For a spatial domain, the A_i are geometry elements, hence the coverage can be viewed as a collection of (geometry,value) pairs, where the value is an element of the range set. If the spatial domain A is a topological space then the coverage can be viewed as a collection of (topology,value) pairs, where the topology element in the pair is a topological n-chain (in GML terms this is a `gml:TopoPoint`, `gml:TopoCurve`, `gml:TopoSurface` or `gml:TopoSolid`).

19.1.3 Coverage in GML

A coverage is implemented as a GML feature. We can thus speak of a “temperature distribution feature”, or a “remotely sensed image feature”, or a “soil distribution feature”.

As is the case for any GML object, a coverage object may also be the value of a property of a feature.

EXAMPLE The temperature distribution might be a property of a city feature (`abc:City`), so a description of the city of Ottawa might be represented in GML as follows (here, `abc:TempratureCoverag`e is a coverage feature that is a property of the city feature):

```
<abc:City gml:id = “Ottawa”>
        <abc:population>500000</abc:population>
        <abc:temperatureDistribution>
            <abc:TemperatureCoverage> … </abc:TemperatureCoverage>
        </abc:temperatureDistribution>
</abc:City>
```

NOTE Coverages in GML are supported by two schemas documents, coverage.xsd and grids.xsd. Coverages.xsd provides the basic GML coverage model. Grids.xsd provides grid geometry structures that are used in the description of gridded coverages but which could be employed for other applications.

The schema document grids.xsd is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:grids:3.2.1

The coverage.xsd schema document is identified by the following location-independent name (using URN syntax):

urn:x-ogc:specification:gml:schema-xsd:coverage:3.2.1

All schema documents are listed in Annex C.

19.1.4 Relationship with ISO 19123

The coverage components of the GML schema specified in this clause provide a conformant, partial implementation of the ISO 19123 coverage schema. The relationship is discussed in detail in D.2.11.

The ISO 19123 coverage types implemented in GML are specified in ISO 19123; additional constraints specified in ISO 19123 for these types are also constraints on the coverage components of the GML schema.

19.2 Grids schema

19.2.1 Overview

An implicit description of geometry is one in which the items of the geometry do not explicitly appear in the encoding. Instead, a compact notation records a set of parameters, and a set of objects may be generated using a rule with these parameters. This clause provides grid geometries that are used in the description of gridded coverages and other applications.

In GML two grid structures are defined, namely `gml:Grid` and `gml:RectifiedGrid`.

19.2.2 Grid

`gml:Grid` implements ISO 19123 CV_Grid (see D.2.11 and ISO 19123:2005, 8.3) and is defined as follows:

```
<element name="Grid" type="gml:GridType" substitutionGroup="gml:AbstractImplicitGeometry"/>

<element name="AbstractImplicitGeometry" type="gml:AbstractGeometryType" abstract="true"
substitutionGroup="gml:AbstractGeometry"/>

 <complexType name="GridType">
  <complexContent>
   <extension base="gml:AbstractGeometryType">
    <sequence>
     <element name="limits" type="gml:GridLimitsType"/>
     <choice>
        <element name="axisLabels" type="gml:NCNameList"/>
        <element name="axisName" type="string" maxOccurs="unbounded"/>
     </choice>
    </sequence>
    <attribute name="dimension" type="positiveInteger" use="required"/>
   </extension>
  </complexContent>
 </complexType>
```

The `gml:Grid` implicitly defines an unrectified grid, which is a network composed of two or more sets of curves in which the members of each set intersect the members of the other sets in an algorithmic way. The region of interest within the grid is given in terms of its `gml:limits`, being the grid coordinates of diagonally opposed corners of a rectangular region. `gml:axisLabels` is provided with a list of labels of the axes of the grid *(`gml:axisName` has been deprecated)*. `gml:dimension` specifies the dimension of the grid.

In GML the `gml:limits` element contains a single `gml:GridEnvelope`, in accordance with the following schema definitions:

```
 <complexType name="GridLimitsType">
  <sequence>
```

```
    <element name="GridEnvelope" type="gml:GridEnvelopeType"/>
  </sequence>
</complexType>

<complexType name="GridEnvelopeType">
  <sequence>
    <element name="low" type="gml:integerList"/>
    <element name="high" type="gml:integerList"/>
  </sequence>
</complexType>
```

The `gml:low` and `gml:high` elements are each `gml:integerLists`, which are coordinate tuples, the coordinates being measured as offsets from the origin of the grid along each axis, of the diagonally opposing corners of a “rectangular” region of interest.

EXAMPLE The following example illustrates a simple Grid.

```
  <gml:Grid dimension="2">
    <gml:limits>
      <gml:GridEnvelope>
        <gml:low>0 0</gml:low>
        <gml:high>3 3</gml:high>
      </gml:GridEnvelope>
    </gml:limits>
    <gml:axisLabels>x y</gml:axisLabels>
  </gml:Grid>
```

In this example the Grid has posts (points) at locations (0,0), (0,1),(1,0),(1,1) through to (3,3).

When a grid point is used to represent a sample space (e.g. image pixel), the grid point represents the center of the sample space (see ISO 19123:2005, 8.2.2).

19.2.3 RectifiedGrid

A rectified grid is a grid for which there is an affine transformation between the grid coordinates and the coordinates of an external coordinate reference system. It is defined by specifying the position (in some geometric space) of the grid “origin” and of the vectors that specify the post locations.

`gml:RectifiedGrid` implements ISO 19123 CV_RectifiedGrid (see D.2.11 and ISO 19123:2005, 8.9) and is declared as follows:

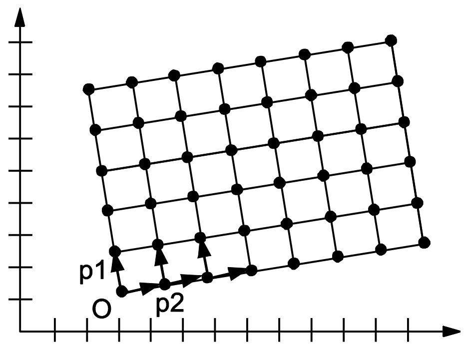
```
<element name="RectifiedGrid" type="gml:RectifiedGridType" substitutionGroup="gml:Grid"/>

<complexType name="RectifiedGridType">
  <complexContent>
    <extension base="gml:GridType">
      <sequence>
        <element name="origin" type="gml:PointPropertyType"/>
        <element name="offsetVector" type="gml:VectorType" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Note that the grid limits (post indexes) and axis name properties are inherited from `gml:GridType` and that `gml:RectifiedGrid` adds a `gml:origin` property (contains or references a `gml:Point`) and a list of `gml:offsetVector` properties (specified using `gml:VectorType` as its data type as described in 10.1.4.5).

NOTE gml:origin and the list of gml:offsetVector properties tie the grid to a position in geographic space and indicate the offset of cells along each axis. See ISO 19123:2005, 8.9.6, for a list of constraints on these properties.

EXAMPLE 1 Figure 5 shows the geometry of a rectified grid.



Key

O origin

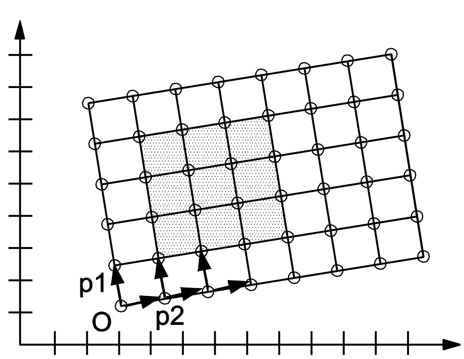
p1, p2 offset vectors

Figure 5 — RectifiedGrid Geometry

EXAMPLE 2 An example instance of a gml:RectifiedGrid is as follows:

```
<gml:RectifiedGrid dimension="2">
  <gml:limits>
   <gml:GridEnvelope>
    <gml:low>1 1</gml:low>
    <gml:high>4 4</gml:high>
   </gml:GridEnvelope>
  </gml:limits>
  <gml:axisLabels>u v</gml:axisLabels>
   <gml:origin>
   <gml:Point gml:id="palindrome" srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
    <gml:pos>3 1.1</gml:pos>
   </gml:Point>
  </gml:origin>
  <gml:offsetVector srsName=" http://www.opengis.net/def/crs/EPSG/0/4326">-0.2 1.25</gml:offsetVector>
  <gml:offsetVector srsName=" http://www.opengis.net/def/crs/EPSG/0/4326">1.3 0.2</gml:offsetVector>
 </gml:RectifiedGrid>
```

Note that in this example the rectified grid starts at integer offset 1 1 (value of low property) relative to the origin as shown in Figure 6.



Key

O origin

p1, p2 offset vectors

Figure 6 — RectifiedGrid with non-zero low limit

19.3 Coverage schema

19.3.1 AbstractCoverageType, AbstractCoverage

The base type for coverages is `gml:AbstractCoverageType`, defined in the schema as follows:

```
<complexType name="AbstractCoverageType" abstract="true">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element ref="gml:domainSet"/>
        <element ref="gml:rangeSet"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The basic elements of a coverage can be seen in this content model: the coverage contains `gml:domainSet` and `gml:rangeSet` properties. The `gml:domainSet` property describes the domain of the coverage and the `gml:rangeSet` property describes the range of the coverage.

The abstract element `gml:AbstractCoverage` implements ISO 19123 CV_Coverage (see D.2.11 and ISO 19123:2005, 5.3) and is declared as follows:

```
<element name="AbstractCoverage" type="gml:AbstractCoverageType" abstract="true"
substitutionGroup="gml:AbstractFeature"/>
```

This element serves as the head of a substitution group which may contain any coverage whose type is derived from `gml:AbstractCoverageType`. It may act as a variable in the definition of content models where it is required to permit any coverage to be valid.

19.3.2 DiscreteCoverageType, AbstractDiscreteCoverage

A discrete coverage consists of a domain set, range set and optionally a coverage function. The domain set consists of either spatial or temporal geometry objects, finite in number. The range set is comprised of a finite number of attribute values each of which is associated to every direct position within any single spatiotemporal object in the domain. In other words, the range values are constant on each spatiotemporal object in the domain. This coverage function maps each element from the coverage domain to an element in its range. This definition conforms to ISO 19123. The base type for discrete coverages is DiscreteCoverageType, defined in the schema as follows:

```
<complexType name="DiscreteCoverageType">
    <complexContent>
        <extension base="gml:AbstractCoverageType">
            <sequence>
                <element ref="gml:coverageFunction" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

The coverageFunction element describes the mapping “f” as shown in Figure 4.

The abstract element gml:AbstractDiscreteCoverage implements ISO 19123 CV_DiscreteCoverage (see D.2.11 and ISO 19123:2005, 5.7) and is declared as follows:

```
<element name="AbstractDiscreteCoverage" type="gml:DiscreteCoverageType" abstract="true"
 substitutionGroup="gml:AbstractCoverage"/>
```

This element serves as the head of a substitution group which may contain any discrete coverage.

19.3.3 AbstractContinuousCoverageType, AbstractContinuousCoverage

A continuous coverage as defined in ISO 19123 is a coverage that can return different values for the same feature attribute at different direct positions within a single spatiotemporal object in its spatiotemporal domain. The base type for continuous coverages is AbstractContinuousCoverageType, defined in the schema as follows:

```
<complexType name="AbstractContinuousCoverageType" abstract="true">
	<complexContent>
		<extension base="gml:AbstractCoverageType">
			<sequence>
				<element ref="gml:coverageFunction" minOccurs="0"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

The coverageFunction element describes the mapping “f” as shown in Figure 4.

The abstract element gml:AbstractContinuousCoverage is declared as follows:

```
<element name="AbstractContinuousCoverage" type="gml:AbstractContinuousCoverageType" abstract="true"
 substitutionGroup="gml:AbstractFeature"/>
```

This element serves as the head of a substitution group which may contain any continuous coverage whose type is derived from gml:AbstractContinuousCoverageType.

19.3.4 domainSet, DomainSetType

The gml:domainSet property element describes the spatio-temporal region of interest, within which the coverage is defined. Its content model is given by gml:DomainSetType which is defined as follows:

```
<element name="domainSet" type="gml:DomainSetType"/>

<complexType name="DomainSetType">
  <sequence minOccurs="0">
    <choice>
      <element ref="gml:AbstractGeometry"/>
      <element ref="gml:AbstractTimeObject"/>
    </choice>
  </sequence>
  <attributeGroup ref="gml:OwnershipAttributeGroup"/>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

The value of the domain is thus a choice between a gml:AbstractGeometry and a gml:AbstractTimeObject. In the instance these abstract elements will normally be substituted by a geometry complex or temporal complex, to represent spatial coverages and time-series, respectively.

NOTE Spatiotemporal domains are supported if the domain is described using a compound coordinate reference system, one of whose components is temporal. Otherwise, following the ISO 19100 series of International Standards, GML does not support combined spatial-temporal domains.

The presence of the `gml:AssociationAttributeGroup` means that domainSet follows the usual GML property model and may use the xlink:href attribute to point to the domain, as an alternative to describing the domain inline. Ownership semantics may be provided using the `gml:OwnershipAttributeGroup`.

19.3.5 rangeSet, RangeSetType

The `gml:rangeSet` property element contains the values of the coverage (sometimes called the attribute values). Its content model is given by `gml:RangeSetType` which is defined as follows::

```
<element name="rangeSet" type="gml:RangeSetType"/>

 <complexType name="RangeSetType">
  <choice>
   <element ref="gml:ValueArray" maxOccurs="unbounded"/>
   <element ref="gml:AbstractScalarValueList" maxOccurs="unbounded"/>
   <element ref="gml:DataBlock"/>
   <element ref="gml:File"/>
 </choice>
</complexType>
```

This content model supports a structural description of the range. The semantic information describing the range set is embedded using a uniform method, as part of the explicit values, or as a template value accompanying the representation using `gml:DataBlock` and `gml:File`.

The values from each component (or “band”) in the range may be encoded within a `gml:ValueArray` element or a concrete member of the `gml:AbstractScalarValueList` substitution group[7]. Use of these elements satisfies the value-type homogeneity requirement.

19.3.6 DataBlock

`gml:DataBlock` describes the Range as a block of text encoded values similar to a Common Separated Value (CSV) representation. The content model is as follows:

```
<element name="DataBlock" type="gml:DataBlockType" substitutionGroup="gml:AbstractObject"/>

 <complexType name="DataBlockType">
     <sequence>
          <element ref="gml:rangeParameters"/>
          <choice>
               <element ref="gml:tupleList"/>
               <element ref="gml:doubleOrNilReasonTupleList"/>
          </choice>
     </sequence>
 </complexType>
```

The range set parameterization is described by the property `gml:rangeParameters`.

19.3.7 rangeParameters

The `gml:rangeParameters` property is declared as follows:

7) e.g. gml:CategoryList, gml:QuantityList — see 16.4.


```
<element name="rangeParameters" type="gml:AssociationRoleType"/>
```

gml:rangeParameters provides a slot for the description of the range parameters. This may be a local description using a suitable record schema (see ISO/TS 19103), or may carry a link to an external range description that matches some standard. Specific range parameters for inline use may be defined through the creation of a GML application schema that may be based on the value objects schema, as described in 16.4.

19.3.8 tupleList

The gml:tupleList property is declared as follows:

```
<element name="tupleList" type="gml:CoordinatesType"/>
```

gml:CoordinatesType is described in 9.1.4.5. It consists of a list of coordinate tuples, with each coordinate tuple separated by the ts or tuple separator (whitespace), and each coordinate in the tuple by the cs or coordinate separator (comma).

The gml:tupleList encoding is effectively “band-interleaved”.

EXAMPLE A set of pairs of temperature and pressure observations might be recorded in a gml:DataBlock as follows:

```
<gml:DataBlock>
    <gml:rangeParameters>
        <gml:CompositeValue>
            <gml:valueComponents>
                <Temperature uom="Cel">template</Temperature>
                <Pressure uom="kPa">template</Pressure>
            </gml:valueComponents>
        </gml:CompositeValue>
    </gml:rangeParameters>
    <gml:tupleList>3,101.2 5,101.3 7,101.4 11,101.5  13,101.6  17,101.7  19,101.7  23,101.8  29,101.9  31,102.0
37,102.1  41,102.2  43,102.3  47,102.4  53,102.5  59,102.6</gml:tupleList>
</gml:DataBlock>
```

where Temperature and Pressure are elements defined in a local application schema, using gml:MeasureOrNilReasonListType.

19.3.9 doubleOrNilReasonTupleList

The gml:doubleOrNilReasonTupleList property is declared as follows:

```
<element name="doubleOrNilReasonTupleList" type="gml:doubleOrNilReasonList"/>
```

gml:doubleOrNilReasonList is described in 8.2.4.1. It consists of a list of gml:doubleOrNilReason values, each separated by a whitespace. The gml:doubleOrNilReason values are grouped into tuples where the dimension of each tuple in the list is equal to the number of range parameters.

EXAMPLE An example of the use of gml:doubleOrNilReasonTupleList to record the same set of pairs of temperature and pressure observations given in the gml:DataBlock example above is as follows:

```
<gml:DataBlock>
    <gml:rangeParameters>
        <gml:CompositeValue>
            <gml:valueComponents>
                <Temperature uom="Cel">template</Temperature>
                <Pressure uom="kPa">template</Pressure>
            </gml:valueComponents>
        </gml:CompositeValue>
```



```
			</gml:rangeParameters>
			<gml:doubleOrNilReasonTupleList>3 101.2 5 101.3 7 101.4 11 101.5  13 101.6  17 101.7  19 101.7  23 101.8  
29 101.9  31 102.0  37 102.1  41 102.2  43 102.3  47 102.4  53 102.5  59 102.6</gml:doubleOrNilReasonTupleList>
		</gml:DataBlock>
```

19.3.10 File, FileType

For efficiency reasons, GML also provides a means of encoding the range set in an arbitrary external encoding, such as a binary file. This encoding may be “well-known” but this is not required. This mode uses the `gml:File` element, which is declared as follows:

```
<element name="File" type="gml:FileType" substitutionGroup="gml:AbstractObject"/>

<complexType name="FileType">
  <sequence>
    <element ref="gml:rangeParameters"/>
    <choice>
      <element name="fileName" type="anyURI"/>
      <element name="fileReference" type="anyURI"/>
    </choice>
    <element name="fileStructure" type="gml:CodeType"/>
    <element name="mimeType" type="anyURI" minOccurs="0"/>
    <element name="compression" type="anyURI" minOccurs="0"/>
  </sequence>
</complexType>
```

In this version of the coverage encoding, the values of the coverage (attribute values in the range set) are transmitted in an external file that is referenced from the XML structure described by `gml:FileType`. The external file is referenced by the `gml:fileReference` property that is an `anyURI` *(the `gml:fileName` property has been deprecated)*. This means that the external file may be located remotely from the referencing GML instance.

EXAMPLE This can support, for example, both an http reference and a SOAP attachment.

The `gml:compression` property points to a definition of a compression algorithm through an `anyURI`. This may be a retrievable, computable definition or simply a reference to an unambiguous name for the compression method.

The `gml:mimeType` property points to a definition of the file mime type.

The `gml:fileStructure` property is defined by a codelist. An example of a values in the codelist could be “Record Interleaved”. Note further that all values shall be enclosed in a single file. Multi-file structures for values are not supported in GML.

The semantics of the range set is described as above using the `gml:rangeParameters` property.

The referenced file structure shall be as shown in Figure 7.

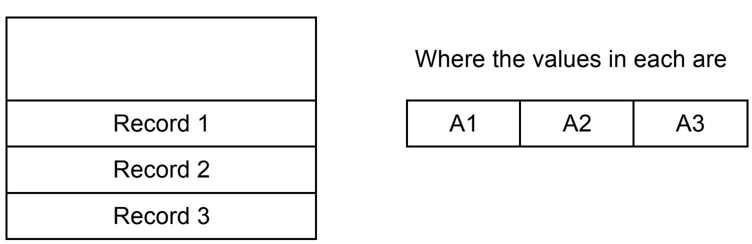


Figure 7 — File Record Structure or Coverage File

Note that if any compression algorithm is applied, the structure above applies only to the pre-compression or post-decompression structure of the file.

Note that the fields within a record match the `gml:valueComponents` of the `gml:CompositeValue` in document order.

EXAMPLE An encoding of a binary file may look as follows:

```
<gml:File>
	<gml:rangeParameters>
		<gml:CompositeValue>
			<gml:valueComponents>
				<Temperature uom="Cel">template</Temperature>
				<Pressure uom="kPa">template</Pressure>
			</gml:valueComponents>
		</gml:CompositeValue>
	</gml:rangeParameters>
	<gml:fileName>http://www.somedata.org/temp_pressure.dat</gml:fileName>
	<gml:fileStructure>Record Interleaved</gml:fileStructure>
</gml:File>
```

19.3.11 coverageFunction, CoverageFunctionType

This subclause describes the `gml:coverageFunction` property, that is, the mapping “f” (see Figure 4) from the domain to the range of the coverage. The content model for the coverage function is given by:

```
<element name="coverageFunction" type="gml:CoverageFunctionType" substitutionGroup="gml:AbstractObject"/>

<complexType name="CoverageFunctionType">
  <choice>
    <element ref="gml:MappingRule"/>
    <element ref="gml:CoverageMappingRule"/>
    <element ref="gml:GridFunction"/>
  </choice>
</complexType>
```

Note that the value of the CoverageFunction is one of *`gml:MappingRule`* *(deprecated)*, `gml:CoverageMappingRule` and `gml:GridFunction`.

If the `gml:coverageFunction` property is omitted for a gridded coverage (including rectified gridded coverages) the `gml:startPoint` is assumed to be the value of the `gml:low` property in the `gml:Grid` geometry, and the `gml:sequenceRule` is assumed to be linear and the `gml:axisOrder` property is assumed to be “+1 +2”.

EXAMPLE These defaults are best illustrated by a simple example as follows:

```
<AverageTempPressure
xmlns="http://www.opengis.net/app" xmlns:gml=”http://www.opengis.net/gml/3.2”
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/app ./CoverageExamples.xsd">
	<gml:domainSet>
```



```
            <gml:Grid dimension="2">
                <gml:limits>
                    <gml:GridEnvelope>
                        <gml:low>0 0</gml:low>
                        <gml:high>4 4</gml:high>
                    </gml:GridEnvelope>
                </gml:limits>
                <gml:axisLabels>x y</gml:axisLabels>
            </gml:Grid>
        </gml:domainSet>
        <gml:rangeSet>
            <gml:DataBlock>
                <gml:rangeParameters>
                    <gml:CompositeValue>
                        <gml:valueComponents>
                            <Temperature uom="Cel">template</Temperature>
                            <Pressure uom="kPa">template</Pressure>
                        </gml:valueComponents>
                    </gml:CompositeValue>
                </gml:rangeParameters>
                <gml:tupleList>3,101.2 5,101.3 7,101.4 11,101.5  13,101.6  17,101.7  19,101.7  23,101.8  29,101.9  31,102.0
37,102.1  41,102.2  43,102.3  47,102.4  53,102.5  59,102.6</gml:tupleList>
            </gml:DataBlock>
        </gml:rangeSet>
</AverageTempPressure>
```

Since no coverageFunction is specified the function is assumed to be that of linear scanning with “+1 +2” order starting at the location (0 0). If we look at the DataBlock, we see that we have the mapping shown in Table 4.

Table 7 — Data block example

Grid location	Data value
0 0	3,101.2
1 0	5,101.3
2 0	7,101.4
3 0	11,101.5
0 1	13,101.6
1 1	17,101.7
2 1	19,101.7
3 1	23,101.8
0 2	29,101.9
1 2	31,102.0
2 2	37,102.1
3 2	41,102.2
0 3	43,102.3
1 3	47,102.4
2 3	53,102.5
3 3	59,102.6

19.3.12 CoverageMappingRule

`gml:CoverageMappingRule` provides a formal or informal description of the coverage function, per:

```
<element name="CoverageMappingRule" type="gml:MappingRuleType" substitutionGroup="gml:AbstractObject"/>

<complexType name="MappingRuleType" final="#all">
	<choice>
		<element name="ruleDefinition" type="string"/>
		<element name="ruleReference" type="gml:ReferenceType"/>
	</choice>
</complexType>
```

EXAMPLE MathML can be used for formal descriptions, informal ones may be any free text.

The mapping rule may be defined as an inline string (`gml:ruleDefinition`) or via a remote reference through xlink:href (`gml:ruleReference`).

If no rule name is specified, the default is 'Linear' with respect to members of the domain in document order.

19.3.13 GridFunction, GridFunctionType

`gml:GridFunction` provides an explicit mapping rule for grid geometries, i.e. the domain shall be a geometry of type grid. It describes the mapping of grid posts (discrete point grid coverage) or grid cells (discrete surface coverage) to the values in the range set. The content model is as follows:

```
<element name="GridFunction" type="gml:GridFunctionType" substitutionGroup="gml:AbstractObject"/>

<complexType name="GridFunctionType">
  <sequence>
```

```
    <element name="sequenceRule" type="gml:SequenceRuleType" minOccurs="0"/>
    <element name="startPoint" type="gml:integerList" minOccurs="0"/>
  </sequence>
</complexType>
```

The `gml:startPoint` is the index position of a point in the grid that is mapped to the first point in the range set (this is also the index position of the first grid post*).* If the `gml:startPoint` property is omitted the `gml:startPoint` is assumed to be equal to the value of `gml:low` in the `gml:Grid` geometry. Subsequent points in the mapping are determined by the value of the `gml:sequenceRule`.

19.3.14 sequenceRule, SequenceRuleType, SequenceRuleEnumeration

The sequenceRule is described by the content model:

```
<complexType name="SequenceRuleType">
    <simpleContent>
        <extension base="gml:SequenceRuleEnumeration">
            <attribute name="order" type="gml:IncrementOrder"/>
            <attribute name="axisOrder" type="gml:AxisDirectionList"/>
        </extension>
    </simpleContent>
</complexType>
```

The `gml:SequenceRuleType` is derived from the `gml:SequenceRuleEnumeration` through the addition of an `axisOrder` attribute. The `gml:SequenceRuleEnumeration` is an enumerated type defined as:

```
<simpleType name="SequenceRuleEnumeration">
  <restriction base="string">
    <enumeration value="Linear"/>
    <enumeration value="Boustrophedonic"/>
    <enumeration value="Cantor-diagonal"/>
    <enumeration value="Spiral"/>
    <enumeration value="Morton"/>
    <enumeration value="Hilbert"/>
  </restriction>
</simpleType>
```

These rule names are defined in ISO 19123.

If no rule name is specified the default is “Linear”.

The `axisOrder` attribute has the following content model:

```
<simpleType name="AxisDirectionList">
    <list itemType="gml:AxisDirection"/>
</simpleType>

<simpleType name="AxisDirection">
    <restriction base="string">
        <pattern value="[\+\-][1-9][0-9]*"/>
    </restriction>
</simpleType>
```

The value of a `gml:AxisDirection` indicates the incrementation order to be used on an axis of the grid.

EXAMPLE 1 "+3" means that the points in the grid are to be traversed from lowest to highest on the 3rd axis.

The different values in a `gml:AxisDirectionList` indicate the incrementation order to be used on all axes of the grid. Each axis shall be mentioned once and only once.

EXAMPLE 2 "+1 -2 +3" means that the points are to be traversed from lowest to highest on the 1st axis, starting at the highest value on the 2nd axis and the lowest value on the 3rd axis points, incremented fastest on the 1st axis before incrementing on the 2nd axis and finally the 3rd.

19.3.15 Specific Coverage Types in GML

GML supports all of the discrete coverage types defined in ISO 19123.

The supported types are substitutable from `gml:AbstractDiscreteCoverage` and include:

— `gml:MultiPointCoverage` (CV_DiscretePointCoverage)

— `gml:MultiCurveCoverage` (CV_DiscreteCurveCoverage)

— `gml:MultiSurfaceCoverage` (CV_DiscreteSurfaceCoverage)

— `gml:MultiSolidCoverage` (CV_DiscreteSolidCoverage)

— `gml:GridCoverage` (CV_DiscreteGridPointCoverage)

— `gml:RectifiedGridCoverage` (CV_DiscreteGridPointCoverage)

NOTE Concrete continuous coverage types can be anticipated in future releases of this International Standard.

Users may also construct their own coverage types by using or deriving from `gml:DiscreteCoverageType`, `gml:AbstractContinuousCoverageType` or by using or derivation from the specific concrete coverage types above.

The same range set encodings apply for each of the different discrete coverage types as the latter are specified by the geometry type of the domain.

19.3.16 MultiPointCoverage

In a `gml:MultiPointCoverage` the domain set is a `gml:MultiPoint`, which is a collection of arbitrarily distributed geometric points. I.e., the value in `gml:domainSet` shall be a `gml:MultiPoint`.

```
<element name="MultiPointCoverage" type="gml:DiscreteCoverageType"
 substitutionGroup="gml:AbstractDiscreteCoverage"/>
```

In a `gml:MultiPointCoverage` the mapping from the domain to the range is straightforward.

— For `gml:DataBlock` encodings the points of the `gml:MultiPoint` are mapped in document order to the tuples of the data block.

— For `gml:CompositeValue` encodings the points of the `gml:MultiPoint` are mapped to the members of the composite value in document order.

— For `gml:File` encodings the points of the `gml:MultiPoint` are mapped to the records of the file in sequential order.

EXAMPLE A `gml:MultiPointCoverage` using value encoding:

```
<AverageTempPressure
    xmlns="http://www.opengis.net/app"
    xmlns:gml="http://www.opengis.net/gml/3.2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/app CoverageExamples.xsd">
    <gml:boundedBy>
```



```
        <gml:Envelope srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
            <gml:lowerCorner>1 1</gml:lowerCorner>
            <gml:upperCorner>4 4</gml:upperCorner>
        </gml:Envelope>
    </gml:boundedBy>
    <gml:domainSet>
        <gml:MultiPoint srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
            <gml:pointMember>
                <gml:Point>
                    <gml:pos>1 1</gml:pos>
                </gml:Point>
            </gml:pointMember>
            <gml:pointMember>
                <gml:Point>
                    <gml:pos>2 2</gml:pos>
                </gml:Point>
            </gml:pointMember>
            <gml:pointMember>
                <gml:Point>
                    <gml:pos>3 3</gml:pos>
                </gml:Point>
            </gml:pointMember>
            <gml:pointMember>
                <gml:Point>
                    <gml:pos>4 4</gml:pos>
                </gml:Point>
            </gml:pointMember>
        </gml:MultiPoint>
    </gml:domainSet>
    <gml:rangeSet>
        <gml:ValueArray>
            <gml:valueComponents>
                <Temperature uom="Cel">3</Temperature>
                <Temperature uom="Cel">5</Temperature>
                <Temperature uom="Cel">7</Temperature>
                <Temperature uom="Cel">11</Temperature>
            </gml:valueComponents>
        </gml:ValueArray>
    </gml:rangeSet>
</AverageTempPressure>
```

19.3.17 MultiCurveCoverage

In a `gml:MultiCurveCoverage` the domain is partitioned into a collection of curves comprising a `gml:MultiCurve`. The coverage function then maps each curve in the collection to a value in the range set.

```
<element name="MultiCurveCoverage" type="gml:DiscreteCoverageType "
 substitutionGroup="gml:AbstractDiscreteCoverage"/>
```

The value in `gml:domainSet` shall be a `gml:MultiCurve`.

In a `gml:MultiCurveCoverage` the mapping from the domain to the range is straightforward.

— For `gml:DataBlock` encodings the curves of the `gml:MultiCurve` are mapped in document order to the tuples of the data block.

— For `gml:CompositeValue` encodings the curves of the `gml:MultiCurve` are mapped to the members of the composite value in document order.

— For `gml:File` encodings the curves of the `gml:MultiCurve` are mapped to the records of the file in sequential order.

EXAMPLE A `gml:MultiCurveCoverage` using data block encoding:

```
<AverageTempPressure
    xmlns="http://www.opengis.net/app"
    xmlns:gml="http://www.opengis.net/gml/3.2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/app CoverageExamples.xsd">
    <gml:boundedBy>
        <gml:Envelope srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
            <gml:lowerCorner>1.1 1.1</gml:lowerCorner>
            <gml:upperCorner>5.5 5.5</gml:upperCorner>
        </gml:Envelope>
    </gml:boundedBy>
    <gml:domainSet>
        <gml:MultiCurve srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
            <gml:curveMember>
                <gml:LineString>
                    <gml:posList dimension="2">1.1 1.1 2.2 2.2</gml:posList>
                </gml:LineString>
            </gml:curveMember>
            <gml:curveMember>
                <gml:LineString>
                    <gml:posList dimension="2">2.2 2.2 3.3 3.3</gml:posList>
                </gml:LineString>
            </gml:curveMember>
            <gml:curveMember>
                <gml:LineString>
                    <gml:posList dimension="2">3.3 3.3 4.4 4.4</gml:posList>
                </gml:LineString>
            </gml:curveMember>
            <gml:curveMember>
                <gml:LineString>
                    <gml:posList dimension="2">4.4 4.4 5.5 5.5</gml:posList>
                </gml:LineString>
            </gml:curveMember>
        </gml:MultiCurve>
    </gml:domainSet>
    <gml:rangeSet>
        <gml:DataBlock>
            <gml:rangeParameters>
                <gml:CompositeValue>
                    <gml:valueComponents>
                        <Temperature uom="Cel">template</Temperature>
                        <Pressure uom="kPa">template</Pressure>
                    </gml:valueComponents>
                </gml:CompositeValue>
            </gml:rangeParameters>
            <gml:doubleOrNilReasonTupleList>3 101.2 5 101.3 7 101.4 11 101.5</gml:doubleOrNilReasonTupleList>
        </gml:DataBlock>
    </gml:rangeSet>
</AverageTempPressure>
```

19.3.18 MultiSurfaceCoverage

In a `gml:MultiSurfaceCoverage` the domain is partitioned into a collection of surfaces comprising a `gml:MultiSurface`. The coverage function than maps each surface in the collection to a value in the range set.


```
<element name="MultiSurfaceCoverage" type="gml:DiscreteCoverageType"
 substitutionGroup="gml:AbstractDiscreteCoverage"/>
```

The value in `gml:domainSet` shall be a `gml:MultiSurface`.

In a `gml:MultiSurfaceCoverage` the mapping from the domain to the range is straightforward.

— For `gml:DataBlock` encodings the surfaces of the `gml:MultiSurface` are mapped in document order to the tuples of the data block.

— For `gml:CompositeValue` encodings the surfaces of the `gml:MultiSurface` are mapped to the members of the composite value in document order.

— For `gml:File` encodings the surfaces of the `gml:MultiSurface` are mapped to the records of the file in sequential order.

EXAMPLE A `gml:MultiSurfaceCoverage` using file encoding:

```
<SoilData xmlns="http://www.opengis.net/app" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/app
./CoverageExamples.xsd">
	<gml:boundedBy>
		<gml:Envelope srsName="http://www.opengis.net/def/crs/EPSG/0/4329">
			<gml:lowerCorner>1 1 1</gml:lowerCorner>
			<gml:upperCorner>10 10 2</gml:upperCorner>
		</gml:Envelope>
	</gml:boundedBy>
	<gml:domainSet>
		<gml:MultiSurface srsName="http://www.opengis.net/def/crs/EPSG/0/4329">
			<gml:surfaceMember>
				<gml:Polygon gml:id="p1">
					<gml:exterior>
						<gml:LinearRing>
							<gml:posList dimension="3">1 1 1 1 5 1 5 5 1 5 1 1 1 1 1</gml:posList>
						</gml:LinearRing>
					</gml:exterior>
				</gml:Polygon>
			</gml:surfaceMember>
			<gml:surfaceMember>
				<gml:Polygon gml:id="p6">
					<gml:exterior>
						<gml:LinearRing>
							<gml:posList dimension="3">10 1 2 5 1 1 5 5 1 10 5 2 10 1 2</gml:posList>
						</gml:LinearRing>
					</gml:exterior>
				</gml:Polygon>
			</gml:surfaceMember>
			<gml:surfaceMember>
				<gml:Polygon gml:id="p11">
					<gml:exterior>
						<gml:LinearRing>
							<gml:posList dimension="3">5 5 1 1 5 1 1 10 1 5 10 1 5 5 1</gml:posList>
						</gml:LinearRing>
					</gml:exterior>
				</gml:Polygon>
			</gml:surfaceMember>
			<gml:surfaceMember>
				<gml:Polygon gml:id="p16">
					<gml:exterior>
```



```
                        <gml:LinearRing>
                            <gml:posList dimension="3">10 5 2 5 5 1 5 10 1 10 10 2 10 5 2</gml:posList>
                        </gml:LinearRing>
                    </gml:exterior>
                </gml:Polygon>
            </gml:surfaceMember>
        </gml:MultiSurface>
    </gml:domainSet>
    <gml:rangeSet>
        <gml:File>
            <gml:rangeParameters>
                <gml:CompositeValue>
                    <gml:valueComponent>
                        <SoilType codeSpace="http://my.big.org/classifications/soils">template</SoilType>
                    </gml:valueComponent>
                    <gml:valueComponent>
                        <SoilMoisture uom="http://my.big.org/units/percent">template</SoilMoisture>
                    </gml:valueComponent>
                </gml:CompositeValue>
            </gml:rangeParameters>
            <gml:fileReference>soil.dat</gml:fileReference>
            <gml:fileStructure>Record Interleaved</gml:fileStructure>
        </gml:File>
    </gml:rangeSet>
```

19.3.19 MultiSolidCoverage

In a `gml:MultiSolidCoverage` the domain is partitioned into a collection of solids comprising a `gml:MultiSolid`. The coverage function than maps each solid in the collection to a value in the range set.

```
<element name="MultiSolidCoverage" type="gml:DiscreteCoverageType"
 substitutionGroup="gml:AbstractDiscreteCoverage"/>
```

The value in `gml:domainSet` shall be a `gml:MultiSolid`.

In a `gml:MultiSolidCoverage` the mapping from the domain to the range is straightforward.

— For `gml:DataBlock` encodings the solids of the `gml:MultiSolid` are mapped in document order to the tuples of the data block.

— For `gml:CompositeValue` encodings the solids of the `gml:MultiSolid` are mapped to the members of the composite value in document order.

— For `gml:File` encodings the solids of the `gml:MultiSolid` are mapped to the records of the file in sequential order.

19.3.20 GridCoverage

A `gml:GridCoverage` is a discrete point coverage in which the domain set is a geometric grid of points as shown in Figure 8.

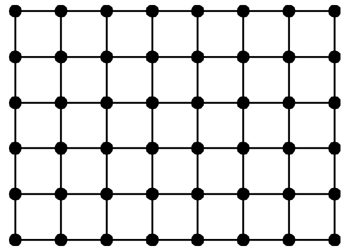


Figure 8 — Grid coverage domain is a grid of points

```
<element name="GridCoverage" type="gml:DiscreteCoverageType"
        substitutionGroup="gml:AbstractDiscreteCoverage"/>
```

Note that this is the same as the `gml:MultiPointCoverage` except that the value in `gml:domainSet` shall be a `gml:Grid`.

`gml:Grid` is defined in 19.2.2. Note that the simple grid coverage is not geometrically referenced and hence no geometric positions are assignable to the points in the grid. Such geometric positioning is introduced in the `gml:RectifiedGridCoverage` discussed in 19.3.21.

NOTE When a grid point is used to represent a sample space, the grid point represents the center of the sample space, see 19.2.2.

EXAMPLE A `gml:GridCoverage` using a file encoding for its values:

```
<AverageTempPressure>
        <gml:domainSet>
                <gml:Grid dimension="2">
                        <gml:limits>
                                <gml:GridEnvelope>
                                        <gml:low>0 0</gml:low>
                                        <gml:high>4 4</gml:high>
                                </gml:GridEnvelope>
                        </gml:limits>
                        <gml:axisLabels>x y</gml:axisLabels>
                </gml:Grid>
        </gml:domainSet>
        <gml:rangeSet>
                <gml:File>
                        <gml:rangeParameters>
                                <gml:CompositeValue>
                                        <gml:valueComponents>
                                                <Temperature uom="Cel">template</Temperature>
                                                <Pressure uom="kPa">template</Pressure>
                                        </gml:valueComponents>
                                </gml:CompositeValue>
                        </gml:rangeParameters>
                        <gml:fileReference>http://www.somedata.org/temp_pressure.dat</gml:fileReference>
                        <gml:fileStructure>Record Interleaved</gml:fileStructure>
                </gml:File>
        </gml:rangeSet>
</AverageTempPressure>
```

19.3.21 RectifiedGridCoverage

The gml:RectifiedGridCoverage is a discrete point coverage based on a rectified grid. It is similar to the grid coverage of 19.3.20 except that the points of the grid are geometrically referenced. The rectified grid coverage has a domain that is a `gml:RectifiedGrid` geometry as defined in 19.2.3.

```
<element name="RectifiedGridCoverage" type="gml:DiscreteCoverageType"
 substitutionGroup="gml:AbstractDiscreteCoverage"/>
```

The value in `gml:domainSet` shall be a `gml:RectifiedGrid`.

`gml:RectifiedGrid` is defined in 19.2.3.

EXAMPLE A `gml:RectifiedGridCoverage` (using a data block):

```
<AveragePressure xmlns="http://www.opengis.net/app" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/app
./CoverageExamples.xsd">
	<gml:boundedBy>
		<gml:Envelope srsName="http://www.opengis.net/def/crs/EPSG/0/4329">
			<gml:lowerCorner>1.2 3.3 2.1</gml:lowerCorner>
			<gml:upperCorner>13.6 12.1 15.3</gml:upperCorner>
		</gml:Envelope>
	</gml:boundedBy>
	<gml:domainSet>
		<gml:RectifiedGrid dimension="2">
			<gml:limits>
				<gml:GridEnvelope>
					<gml:low>1 1</gml:low>
					<gml:high>4 4</gml:high>
				</gml:GridEnvelope>
			</gml:limits>
			<gml:axisLabels>u v</gml:axisLabels>
			<gml:origin>
				<gml:Point gml:id="palindrome" srsName="http://www.opengis.net/def/crs/EPSG/0/4329">
					<gml:pos>1.2 3.3 2.1</gml:pos>
				</gml:Point>
			</gml:origin>
			<gml:offsetVector srsName="http://www.opengis.net/def/crs/EPSG/0/4329">1.1 2.2 3.3</gml:offsetVector>
			<gml:offsetVector srsName="http://www.opengis.net/def/crs/EPSG/0/4329">2.0 1.0 0.0</gml:offsetVector>
		</gml:RectifiedGrid>
	</gml:domainSet>
	<gml:rangeSet>
		<gml:DataBlock>
			<gml:rangeParameters>
				<Pressure uom="kPa">template</Pressure>
			</gml:rangeParameters>
			<gml:doubleOrNilReasonTupleList>101.2 101.3 101.4 101.5  101.6 101.7 101.7 101.8 101.9 102.0 102.1 102.2
102.3 102.4 102.5 102.6</gml:doubleOrNilReasonTupleList>
		</gml:DataBlock>
	</gml:rangeSet>
</AveragePressure>
```

20 Profiles

20.1 Profiles of GML and application schemas

GML is a complex standard that is richly expressive. In general, an application need not exploit the entire GML schema, but may employ a subset of constructs corresponding to specific relevant requirements.

We use this definition of a profile (ISO/IEC TR 10000-1:1998 and ISO 19106:2004):

***Profile**: A set of one or more base standards and/or [profiles], and, where applicable, the identification of chosen classes [(types, attributes and elements)], conforming subsets, options and parameters of those base standards, or [profiles] necessary to accomplish a particular function.*

This was defined for an OSI architecture model, so we translate 'class' to 'types, attributes and elements' to apply this definition to XML Schema. There are several ways to implement this, and GML profiles use a "copy and delete" approach. To create a profile, a developer might copy the applicable schema files from GML and simply delete any global types, elements and local optional particles that she does not need for her application schema.

20.2 Definition of profile

A profile of GML may be defined to enhance interoperability and to curtail ambiguity by allowing only a specific subset of GML. Application schemas may then conform to such a profile in order to take advantage of any interoperability or performance advantages that it offers in comparison with a complete GML. Such profiles may be defined for application schemas that are included in other specifications.

There are cases where reduced functionality is acceptable, or where processing requirements compel use of a logical subset of GML. For example, applications that do not need to handle XLink attributes in any form may adhere to a specific profile that excludes them; the constraint in this case would be to not use links. Other cases might include defining constraints on the level of nesting allowed inside tags (i.e. tree depth), or only allowing features with homogeneous properties as members of a feature collection. In many cases, such constraints may be enforced via new schemas; others may be enforced through procedural agreements within an information community.

20.3 Relation to application schema

A profile may be the beginning of an application schema.

EXAMPLE A location based service profile may limit the types of geometry to that used in LBS applications, and the LBS application schema may then add a "PointCircle," "PointEllipse" and "PointArc" elements to accommodate the LIF "CIRCLE," "ELLIPSE" and "ARC" elements, which are used to describe error estimates of mobile device location.

The building of such application schemas is thus a two-part process. The profile acts as a restriction of GML to produce types and elements consistent with the complete GML but potentially lacking in some optional particles. The application schema then uses these types as a common base, and uses them in new types and elements by extensions or inclusion.

$$\text{GML} \xrightarrow{\text{selection \& restriction}} \text{GML profile} \xrightarrow{\text{extension \& inclusion}} \text{application schema}$$

20.4 Rules for elements and types in a profile

Global profiled elements in a GML profile shall:

— share the same name (and namespace) of a parent element in GML.

— include all mandatory particles (subelements and attributes) of the parent element in GML.

— include no particle that is not in the parent element in GML.

— have the same default values for attributes as the parent element in GML.

— have a parallel substitution group hierarchy for named elements in both schemas.

Global types in a GML profile shall:

— share the same name (and namespace) of a parent type in GML.

— include all mandatory particles (subelements and attributes) of the parent type in GML.

— include no particle that is not in the parent type in GML.

— have the same default values for attributes as the parent type in GML.

— have a parallel derivation tree for named types in both schemas.

Instance documents of a profile shall be valid against the full GML schema.

Using the “copy and delete” metaphor described above, our mythic developer may:

— delete global elements and global types.

— delete optional subelements from any types or elements.

— make optional subelements or attributes mandatory in any type or element (if a default value exists, it shall be eliminated or the schema validation will report an error — default values are only valid for optional particles).

— restrict cardinality of any particle.

None of the above will affect the validity of a document that is designed against the profile, but tested against the full GML schema. Our mythic developer **may not**:

— delete mandatory subelements from any types or elements.

— make mandatory particles optional.

— relax cardinality restrictions of any particle.

— add or change a default or fixed value.

The last item is a bit subtler than the others are. Documents valid under the profile would still be valid under the full GML schema, but the interpretation of those documents would change. For example, if a profile specified a default coordinate reference system to be UTM, and the full schema specified a WGS 84 geodesic (latitude, longitude) as the default CRS, then the interpretation of the file would change when moving from the profile to the full schema.

20.5 Rules for referencing GML profiles from application schemas

A GML application schema shall reference the full GML schema in the `schemaLocation` attribute of the `<import>` element.

A GML application schema document conforming to one or more GML Profiles shall provide an appInfo annotation element `<gml:gmlProfileSchema>` for every profile in the root schema document `<schema>` element where the value is a schema location of the profile schema. Note that an application schema may conform to multiple profiles.

EXAMPLE


```
<schema ...>
    <annotation>
        <appInfo>
            <gml:gmlProfileSchema>http://schemas.opengis.net/gml/3.2.1/profiles/gmlSimpleFeatureProfile/1.1.0/gmlsf.xsd</gml:gmlProfileSchema>
            <gml:gmlProfileSchema>http://schemas.opengis.net/gml/3.2.1/profiles/gmlPointProfile/1.1.0/gmlPointProfile.xsd</gml:gmlProfileSchema>
        </appInfo>
    </annotation>
    ...
</schema>
```

The `<gml:gmlProfileSchema>` element is defined as

```
<element name="gmlProfileSchema" type="anyURI"/>
```

20.6 Recommendations for application schemas using GML profiles

In order that the profile within an application schema may be later extended to include other profiled GML elements, the following recommendations are made:

— Global elements that are not in a GML profile but are in an application schema using a GML profile should not have the same name as any element in the GML schema.

— Global types that are not in a GML profile but are in an application schema using a GML profile should not have the same name as any type in the GML schema.

If a type or element in an application schema is found to be of universal use, then the above conventions will aid the application schema from migrating that type or element from its own namespace to that of GML.

The following recommendations are made simply as a bookkeeping convenience to those trying to understand the role of the profile in the application schema:

— Profiled elements and types should be included either in a single file for smaller profile or in a file structure that parallels that of GML. The exact naming convention of the parallelism is left to the application schema author.

— A reference to the appropriate GML schema document should be made in a comment near the beginning of the file.

NOTE A method that has been found to be convenient is to package the required GML components into a “stub” schema document called, e.g., “gmlForApplicationDomain.xsd”. This document may comprise a copy of the necessary components assembled in a fine-grained manner (e.g. see Annex G), or may merely <include> a subset of the schema documents that comprise the standard GML distribution. The schema document gml.xsd is an exhaustive superset following the latter approach.

20.7 Summary of rules for GML profiles

In summary, the rules for a profile:

— A profile of GML is a logical restriction of a subset of GML.

— A profile shall not change the name, definition, or data type of mandatory GML elements or attributes.

— The relevant schema or schemas that define a profile shall use in the core ‘gml’ namespace http://www.opengis.net/gml/3.2.

— An application schema may extend and use types from the profile, but shall do so in its own namespace, and not use http://www.opengis.net/gml/3.2.

The functional test of these rules is:

Any instance document for an application schema using a GML profile will be valid against the same application schema if the GML profile is replaced by the complete GML schema. Further, the interpretation of that document would be the same regardless of which of the two schemas were used.

21 Rules for GML application schemas

21.1 Instances of GML objects

21.1.1 GML documents

An XML document contains a single XML element as its root. A GML document may be one of the following elements:

— A `gml:AbstractFeature` or any element directly or indirectly in its substitution group.

NOTE 1 This includes feature collections and coverages as both are features, too.

— A `gml:Dictionary` or any element directly or indirectly in its substitution group.

NOTE 2 This includes coordinate reference system and units dictionaries.

— A `gml:TopoComplex` or any element directly or indirectly in its substitution group.

The standard methods for XML documents based on W3C XML Schema provide that the XML namespaces used in a document are declared as attributes within the document, and the location of schema documents that provide the source components for each namespace may be indicated.

For a GML document, the source of the components describing the primary components within the document is a GML application schema. Both the document type and the associated GML application schema are described in this Clause.

Note that this does not imply that all elements and attributes in the GML document are defined by a single GML application schema. The schema components referenced from the GML document may be contained in any number of GML application schemas or other XML Schemas.

21.1.2 GML object elements in other XML documents

Elements of GML objects may occur in XML documents that are not GML documents, too. The XML document shall validate against an XML Schema document that imports directly or indirectly the GML schema or a GML profile and optionally one or more GML application schemas.

EXAMPLE GML object elements may be used in request and response messages of Web Services.

21.2 GML application schemas

21.2.1 Introduction

A GML application schema is an XML Schema, conforming to the rules outlined in this clause, which describes one or more types of geographic object, components of geographic objects or metadata, including dictionaries and definitions, used in the definition of geographic objects. A GML application schema defines a vocabulary for a particular domain of discourse by defining and describing the terms of that vocabulary (see ISO 19109) as follows:

An application schema may reference directly concrete, global GML elements (including groups) and attributes (including attributeGroups) whose names and content models accurately represent components of the vocabulary it defines.

EXAMPLE 1 This includes property elements like `gml:name` or `gml:description`, object elements like `gml:Observation`, `gml:Dictionary`, or `gml:Point`, and attributes like `gml:id`.

An application schema may declare new elements and attributes in its own namespace using GML types when the vocabulary it defines needs to include different names for the same content models to distinguish their semantic roles. The element declared in the application schema will be in a different namespace, and may be used in an instance document.

EXAMPLE 2 `gml:EnvelopeType` may be used unmodified as the content model for an element `xmml:Interval`.

EXAMPLE 3 `gml:LengthType` may be used unmodified as the content model for an element `ex:height`.

EXAMPLE 4 `gml:PointPropertyType` may be used directly as the content model for a property element `ex:representativePoint`.

An application schema may derive new types in its own namespace by extension of GML types when the vocabulary it defines needs to include components with additional, domain-specific properties.

NOTE The definition of application-specific feature types requires that the content model of the feature types is derived from `gml:AbstractFeatureType`, typically by extension.

EXAMPLE 5 The definition of new geometry types not specified in the GML schema, but required by an application, e.g. an ellipse.

An application schema may derive new types in its own namespace by restriction of GML types when the vocabulary it defines needs to include more specialized versions of GML types that restrict the cardinality or type of their properties.

EXAMPLE 6 An application wants to prohibit the use of multiple names in their feature types. This may be achieved by deriving an application-specific root feature type by restriction from `gml:AbstractFeatureType` that sets the maximum occurrence of the `gml:name` to “1”.

An application schema may declare new elements that are assigned to a substitution group whose head is an abstract or concrete GML element. The element declared in the application schema may then appear in instance documents in place of the substitution group head and be conformant to the content model that refers to the substitution group head. Note that in order to be a valid member of a substitution group, the type of the element shall be validly derived from the type of the element which is the head of the substitution group. All abstract elements in the GML schema are only useful acting as the heads of substitution groups.

EXAMPLE 7 `gml:AbstractGML`, `gml:AbstractFeature`, `gml:AbstractGeometry`, `gml:AbstractCoverage may all serve as the head of a substitution group for elements in an application schema.`

An application schema may declare new elements, attributes and types in its own namespace using types it has defined to give vocabulary-specific names to their content models.

EXAMPLE 8 Application-specific data types or enumerations.

All GML application schemas are constructed, using the general rules of this Clause, from one or more of the GML schema components defined in Clauses 6.5 to Clause 19.

GML allows the derivation of many other kinds of elements such as new units of measure, new geometry properties and new geometries. While these elements may be packaged into separate schemas they are viewed as subordinate to the schema categories of this Clause. Any GML application schema shall be of at least one of the schema types described in 21.3 through 21.11, and comply to the rules from the respective subclauses in addition to 21.2. It is thus permissible to create a GML application schema that defines Features, Coverages and Values, so long as this schema satisfies the rules of 21.2, 21.3, 21.8 and 21.9.

21.2.2 Target namespace

An application schema shall declare a target namespace. This is the namespace in which the terms for objects and properties of the vocabulary defined by the application schema live. This shall not be the GML namespace (http://www.opengis.net/gml/3.2). It is conventional for the namespace identifier to be a URL controlled by the application schema author's organization. A target namespace is declared in an application schema using the `targetNamespace` attribute of the schema element from XML Schema.

An application schema may be comprised of multiple schema documents that all declare the same target namespace.

NOTE It is recommended that a top-level schema document in such a modularized application schema should directly or indirectly include the other documents to avoid the XML application processing limitations discussed in Annex J.

21.2.3 Import GML schema

A GML application schema shall import the full GML schema. It may identify GML profiles that include all of the components from GML that it directly or indirectly uses to define its vocabulary as specified in 20.5.

The required import of the GML schema components may be provided indirectly via the import of another schema in the namespace of GML that includes the required GML schema documents.

EXAMPLE 1 The import of gml.xsd from Annex C would satisfy any of these schema import requirements.

```
<import namespace="http://www.opengis.net/gml/3.2" schemaLocation="../gml.xsd"/>
```

The `<import>` element specifies that the components described in the imported GML schema document are associated with the GML namespace http://www.opengis.net/gml/3.2. This namespace identifier shall match the target namespace specified in the schema being imported in order to ensure XML Schema validity.

The path (`schemaLocation`) to the imported GML schema document shall be provided and may be to a local copy of the document, or may be a URI reference to a copy of the schema document in some remote repository.

EXAMPLE 2 Examples for such repositories are `http://www.iso.org/ittf/ISO_19136_Schemas` on the ISO web site or `http://schemas.opengis.net/` on the OGC web site.

NOTE According to the W3C XML Schema specification, the schemaLocation attribute is only a hint to the physical location and may be disregarded by XML parsers.

21.2.4 Object type derivation

An object type declared by a GML application schema shall not violate any XML Schema derivation restrictions imposed by a `final` attribute on its base GML type nor any other XML Schema rules.

The content model of an object type defined by a GML application schema shall derive directly from the most specialized GML object type that may serve as the base for its content model while preserving semantic consistency and increasing type specialization.

21.2.5 Elements representing objects

A GML application schema shall declare a global element for any object type that is to serve as the root element in a GML document.

21.2.6 Property type derivation

A property type defined by a GML application schema to contain inline or reference a single GML object may be derived from `gml:AssociationRoleType` or may follow the pattern of this type.

A property type defined by a GML application schema to contain inline a single GML object may be derived from `gml:InlinePropertyType` or may follow the pattern of this type.

A property type defined by a GML application schema to reference a single GML object may be derived from `gml:ReferenceType` or may follow the pattern of this type.

A property type defined by an application schema to contain a homogeneous collection of GML objects shall follow the pattern of `gml:InlinePropertyType`, but may change the `minOccurs` and `maxOccurs` values in the <sequence> element.

NOTE As derivation-by-restriction of property types has created problems with commonly used XML parsers in the past, all instances of such derivations have been removed from the GML schema. It is recommended to avoid derivation-by-restriction in property types in application schemas, too.

21.2.7 Elements representing properties

Elements representing properties of GML objects may be declared as global elements in an application schema, or they may be declared locally within object content models (type definitions).

NOTE Elements in the content of complex types that are defined with local names in an application schema will prevent derivation by restriction in another namespace. Such complex types are appropriate for elements intended for use "as is" in their own namespace, and may be declared to be `final="restriction"`. Elements in the content of complex types defined by reference to global elements support derivation by restriction in another namespace, allowing restriction of cardinality, and/or replacement by a member of a substitution group. Such complex types designed for derivation by restriction are appropriate "library types" for elements in substitution groups that cross namespaces.

If the value of the property is expected to be available elsewhere, the type of the property element shall support referencing the GML object that is the value of the property (see 21.2.6).

If the value of the property is expected to be represented inline, the type of the property element shall support this, either by having XML Schema simple content of the appropriate simple type or by containing the GML object that is the value of the property inline (see 21.2.6).

If the value of the property is expected to be available either elsewhere, or represented inline, then the type of the property element shall support both methods. In this case the type for the property element shall have the `gml:AssociationAttributeGroup`, in which all members are optional, and the child element shall have `minOccurs="0"` so that in an instance document the property element may be empty if it carries an xlink (see 7.2.3). If it is desired to prohibit the possibility of both xlink attributes and content, or neither, then this constraint should be recorded as a normative directive in an <annotation> element on the element declaration in the GML application schema. The directive may be expressed as prose, or it may be expressed using a formal notation such as Schematron.

21.3 Schemas defining Features and Feature Collections

21.3.1 Introduction

Features and feature collections are the primary view of geospatial information supported by GML, and are particularly useful in modelling real world geography or in defining message types for geographic web services. A feature models a real world object or concept, see Clause 9.

Feature application schemas define geographic features and feature collections for a specific application domain or community. These GML application schemas shall obey the additional rules described in the following subclauses.

21.3.2 Import GML schema components

The application schema shall import the GML schema as described in 21.2.3. Any GML profile referenced from the application schema shall include at least the `gml:AbstractFeature` element and all schema components used by this element.

21.3.3 Elements representing features

All geographic features and feature collections in the application schema shall be declared as global elements in the schema, i.e. they shall be immediate child elements of the XML Schema <schema> element.

The name of an element that instantiates a GML feature shall be its feature type, in the sense described in ISO 19109.

An element representing a feature shall be either directly or indirectly in the substitution group of `gml:AbstractFeature`.

An element representing a feature collection type shall be either directly or indirectly in the substitution group of `gml:AbstractFeature` and have in its content model a property element whose content model is derived by extension from `gml:AbstractFeatureMemberType`.

In a GML application schema, an object that is an abstraction of a real world phenomenon shall be modelled as a feature. All other objects shall be modelled as a GML object that is not a feature, i.e. the element representing the object shall be either directly or indirectly in the substitution group of `gml:AbstractGML`, but neither directly nor indirectly in the substitution group of `gml:AbstractFeature`.

21.3.4 Application features are features

A feature defined in an application schema shall conform to the rules respecting GML features as described in Clause 9 and conform to the rules described in Clause 7.

NOTE 1 The name of a feature element is the semantic type of the feature.

NOTE 2 The children of a feature element are always property elements that describe the feature, and such properties are always encoded as child elements. Properties are not encoded as XML attributes.

21.4 Schemas defining spatial geometries

21.4.1 Import GML geometry schema components

The application schema shall import the GML schema as described in 21.2.3. Any GML profile referenced from the application schema shall include at least the `gml:AbstractGeometry` element and all schema components used by this element.

NOTE Typically additional geometry schema components are required besides those required by `gml:AbstractGeometry`. In practice, especially concrete elements and types like `gml:Point` and `gml:PointPropertyType` will typically be part of the profile.

21.4.2 User-defined geometry types and geometry property types

21.4.2.1 User-defined geometry types

Authors of application schemas may create their own geometry types if GML lacks the desired construct. To do this, authors shall ensure that the object elements of these concrete geometry and geometry collection types are in the substitution group (either directly or indirectly) of the corresponding GML object element: `gml:AbstractGeometry`.

EXAMPLE The following element and complex type definition in an application schema extends `gml:Point` and adds a bearing (e.g. for the orientation of a symbol in portrayal).

```
<element name="PointWithBearing" type="ex:PointWithBearingType" substitutionGroup="gml:Point">

<complexType name="PointWithBearingType">
	<complexContent>
		<extension base="gml:PointType">
			<sequence>
				<element name="bearing" type="gml:AngleType"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

Any user-defined geometry subtypes shall inherit the elements and attributes of the base GML geometry types without restriction, but may extend these base types to meet application requirements, such as providing a finer degree of interoperability with legacy systems and data sets.

All rules specified in Clause 7, Clause 10, 10.5.10 and Clause 11 shall be followed.

21.4.2.2 User-defined geometry property types

Furthermore, authors of application schemas may create their own geometry property types that encapsulate geometry types which are defined in Clause 10, 10.5.10 or Clause 11 or which they have defined in accordance with 21.4.2.1. They shall ensure that these properties follow the pattern used by `gml:GeometryPropertyType` for standard properties and `gml:GeometryArrayPropertyType` for array properties. The target type shall be a bona fide geometry object element.

A geometry property type may be a restriction of `gml:GeometryPropertyType`, but this is not a requirement. Nevertheless, every geometry property shall follow the pattern of this type. An application schema may support the choice between an inline or a by-reference semantic or it may restrict the use to either inline (prohibit the use of the Xlink attributes) or by-reference (prohibit the containment of the geometry in the feature).

A geometry array property type may be a restriction of `gml:GeometryArrayPropertyType`, but this is not a requirement. Nevertheless, every geometry property shall follow the pattern of this type. All geometry elements in the array are contained inline in the containing object, only inline semantics is supported by array properties.

EXAMPLE The following complex type definitions in an application schema define a “standard” property type for a user-defined geometry type and an array property type for the same geometry type.

```
<complexType name="MyGeometryPropertyType">
      <sequence>
            <element ref="foo:PointWithBearingType"  minOccurs="0"/>
      </sequence>
      <attributeGroup ref="gml:AssociationAttributeGroup" />
      <attributeGroup ref="gml:OwnershipAttributeGroup" />
</complexType>

<complexType name="MyGeometryArrayPropertyType">
      <sequence>
            <element ref="foo:PointWithBearingType" minOccurs="0" maxOccurs="unbounded" />
      </sequence>
</complexType>
```

21.5 Schemas defining spatial topologies

21.5.1 Import GML topology schema components

The application schema shall import the GML schema as described in 21.2.3. Any GML profile referenced from the application schema shall include at least the `gml:AbstractTopology` element and all schema components used by this element.

NOTE Typically additional topology schema components are required besides those required by `gml:AbstractTopology`. In practice, especially concrete elements and types like `gml:Edge` and `gml:DirectedEdgePropertyType` will typically be part of the profile.

EXAMPLE Import the GML schema for example as follows using a schema document of Annex C:

```
<import namespace="http://www.opengis.net/gml/3.2" schemaLocation="../gml.xsd"/>
```

21.5.2 User-defined topology types and topology property types

21.5.2.1 User-defined topology types

Authors of application schemas may create their own topology types if GML lacks the desired construct. To do this, authors shall ensure that the object elements of these concrete topology types are in the substitution group (either directly or indirectly) of the corresponding GML object element: `gml:AbstractTopology`.

Any user-defined topology subtypes shall inherit the elements and attributes of the base GML topology types without restriction, but may extend these base types to meet application requirements, such as providing a finer degree of interoperability with legacy systems and data sets.

All rules specified in Clauses 7 and 13 shall be followed.

21.5.2.2 User-defined topology property types

Furthermore, authors of application schemas may create their own (directed) topology property types that encapsulate topology types they have defined in accordance with Clause 13. They shall ensure that these properties follow the rules described in 21.2.6. In addition, the target type shall be a bona fide topology construct.

A topology property type may be a restriction of an existing topology property type.

A topology property type may support the choice between an inline or a by-reference semantic or it may restrict the use to either inline (prohibit the use of the Xlink attributes) or by-reference (prohibit the containment of the geometry in the feature).

21.6 Schemas defining time

21.6.1 Import GML temporal schema components

The application schema shall import the GML schema as described in 21.2.3. Any GML profile referenced from the application schema shall include at least the `gml:AbstractTimeObject` element and all schema components used by this element.

NOTE Typically additional temporal schema components are required besides those required by `gml:AbstractTimeObject`. In practice, especially concrete elements and types like `gml:TimeInstant` and `gml:TimeInstantPropertyType` will typically be part of the profile.

21.6.2 User-defined temporal types and temporal property types

21.6.2.1 User-defined temporal types

Authors of application schemas may create their own temporal types if GML lacks the desired construct. To do this, authors shall ensure that the object elements of these concrete temporal types are in the substitution group (either directly or indirectly) of the corresponding GML object element: `gml:AbstractTimeObject`.

Any user-defined temporal subtype shall inherit the elements and attributes of the base GML temporal types without restriction, but may extend these base types to meet application requirements, such as providing a finer degree of interoperability with legacy systems and data sets.

All rules specified in Clauses 7 and 14 shall be followed.

21.6.2.2 User-defined temporal property types

Furthermore, authors of application schemas may create their own temporal property types that encapsulate temporal types they have defined in accordance with Clause 14. They shall ensure that these properties follow the rules described in 21.2.6. In addition, the target type shall be a bona fide temporal construct.

A temporal property type may be a restriction of an existing temporal property type.

A temporal property type may support the choice between an inline or a by-reference semantic or it may restrict the use to either inline (prohibit the use of the Xlink attributes) or by-reference (prohibit the containment of the geometry in the feature).

21.7 Schemas defining coordinate reference systems

21.7.1 Introduction

Many of the concrete XML elements defined in the CRS Schemas may be used without Application Schemas, whenever no content extensions or restrictions are needed. An Application Schema shall be used whenever element contents extension is required, and should be used in most other cases to specify needed restrictions. That is, an Application Schema should be defined to extend and/or restrict elements as needed for a specific application, or a set of applications, to:

— Add elements to contents of existing elements, for recording additional data about that item needed for that application.

— Restrict the multiplicity of current contents elements, to eliminate flexibility not needed and perhaps confusing for that application.

— Use a different element name, to be more easily understood in that specific application, primarily for elements that will be instantiated many times.

— Specify standard contents and contents patterns for selected elements and attributes, as needed to improve interoperability.

— Specify standard XML and other documents to be referenced or otherwise used, as needed to improve interoperability.

Application Schemas may thus be used for XML document contents extensions, restrictions, or both. Contents extension is expected to be often used to record additional data needed for applications. Contents restriction is expected to be frequently used to restrict contents, in order to increase interoperability and reduce ambiguity when greater flexibility is not needed for applications. Extensions of existing concrete elements may be defined by extending that concrete element. In many cases, restrictions of existing concrete elements may be done by extending the abstract element from which that concrete element is derived, by adding somewhat different but corresponding extensions.

An Application Schema may specify a single top-level element for use in an XML document, with the XML elements and types that it uses. That single top-level XML element may be an object with identity, but this is not required. Such an Application Schema will import and build upon the XML Schema components specified in Clause 12.

Application Schemas could define additional concrete elements using or extending other abstract elements, if needed. However, an additional concrete element using or extending an abstract element should not be defined if that concrete element is largely similar to an existing element, and thus probably should extend or use an existing concrete element. In many cases, the existing concrete elements that use an abstract element are believed to be largely exhaustive. This is particularly true when the existing concrete elements include one element that is quite general, such as the elements EngineeringCRS, DerivedCRS, EngineeringDatum, UserDefinedCS, OperationParameter, and OperationParameterGroup.

The Conversion, Transformation, ParameterValue, and ParameterValueGroup elements may be used for well-known coordinate operation methods, especially when only one instance of that element is needed for that operation method. However, these elements should not be used for well-known coordinate operation methods when many instances of that element are needed for one operation method. Instead, an Application Schema that defines operation-method-specialized element names and contents should be prepared for each such operation method. For interoperability, a suitable geospatial information community should standardize each such Application Schema.

NOTE This use of Application Schemas follows the patterns used in Feature Application Schemas.

21.7.2 Import GML coordinate reference system schema components

The application schema shall import the GML schema as described in 21.2.3. Any GML profile referenced from the application schema shall include at least the `gml:AbstractCoordinateReferenceSystem` element and all schema components used by this element.

NOTE Typically additional coordinate reference system schema components are required besides those required by `gml:AbstractCoordinateReferenceSystem`. In practice, especially concrete elements and types will typically be part of the profile, too.

21.8 Schemas defining coverages

21.8.1 Introduction

The following subclauses define the rules for the construction of GML application schemas for coverages. Coverages are described in Clause 19. Note that coverages are features and hence the rules of 21.3 above apply also to coverages.

21.8.2 Import GML coverage schema components

The application schema shall import the GML schema as described in 21.2.3. Any GML profile referenced from the application schema shall include at least the `gml:AbstractCoverage` element and all schema components used by this element.

NOTE Typically additional coverage schema components are required besides those required by `gml:AbstractCoverage`. In practice, especially concrete elements and types like `gml:RectifiedGridCoverage` will typically be part of the profile, too.

21.8.3 User-defined coverage types

All geographic coverages in the GML application schema shall be declared as global elements in the schema, i.e. they shall be child elements of the XML Schema <schema> element.

Authors of application schemas may create their own coverage types if GML lacks the desired construct. To do this, authors shall ensure that the object elements of these coverage types are in the substitution group (either directly or indirectly) of the corresponding GML object element: either `gml:AbstractDiscreteCoverage` or `gml:AbstractContinuousCoverage`.

NOTE 1 These elements are indirectly in the substitution group of `gml:AbstractFeature` and hence the condition of the feature model is satisfied.

NOTE 2 This International Standard provides the concrete coverage types `gml:MultiPointCoverage`, `gml:MultiCurveCoverage`, `gml:MultiSurfaceCoverage`, `gml:MultiSolidCoverage`, `gml:GridCoverage`, and `gml:RectifiedGridCoverage`. Application coverages may derive from any of these as well.

Any user-defined coverage subtype shall inherit the elements and attributes of the base GML coverage types without restriction, but may extend these base types to meet application requirements, such as providing a finer degree of interoperability with legacy systems and data sets.

All rules specified in Clauses 7, 9 and 19 shall be followed.

21.8.4 Range parameters shall be substitutable for AbstractValue

The coverage application schema shall define or import the definitions for all range parameters. Each such range parameter shall be substitutable for `gml:AbstractValue` as defined in 16.4. Note that this allows the range parameter to take on a wide range of types. Note further that the schema components specified in 16.4 include several abstract subtypes that are substitutable for `gml:AbstractValue`, including `gml:AbstractScalarValue` and `gml:AbstractValueList`. Concrete scalar and value list types, and substitution group head elements, are also provided (substitutable for `gml:AbstractScalarValue` or `gml:AbstractValueList`) and include:

— `gml:Category` (the content model is specified by `gml:CodeType`)

— `gml:CategoryList` (the content model is specified by `gml:CodeOrNilReasonListType`)

— `gml:Quantity` (the content model is specified by `gml:MeasureType`)

— `gml:QuantityList` (the content model is specified by `gml:MeasureOrNilReasonListType`)

— `gml:Count` (the content model is specified by `gml:CountType`)

— `gml:Boolean` (the content model is specified by `gml:BooleanType`)

To define the range parameters in a Coverage Application Schema, refer to 16.4.

EXAMPLE Typical examples of the use of the value types in the development of a GML coverage can be found in 19.3, and are summarized in Table 8.

Table 8 — Range parameters for coverage schemas

Coverage	Range parameter	Definition in GML
Temperature Distribution (weather)	Temperature	Would be derived from `gml:MeasureOrNilReasonListType` and made substitutable for `gml:measure` defined in 16.3.2.
Soil type distribution (agronomy)	Soil type	Would be derived from `gml:CategoryType` and made substitutable for `gml:Category`. Weak reference to an enumeration of soil types.
Multi-spectral optical image (remote sensing)	Reflectance in each spectral band.	Would be derived from `gml:QuantityListType` and made substitutable for `gml:QuantityList`.
Distribution of West Nile Virus cases. (epidemiology)	CaseCount	Would be derived from `gml:integerOrNilReasonList`, and made substitutable for `gml:CountList`.

21.8.5 Coverage document

A coverage document is defined by a corresponding coverage schema. The root element of this document shall be a coverage defined in this schema or may be a feature collection as described in 9.9 whose members are coverages.

21.9 Schemas defining observations

21.9.1 Introduction

The following subclauses describe how to create an observation application schema. Observations are described in Clause 18. An observation application schema defines one or more types of observation in accordance with the following rules.

21.9.2 Import GML observation schema components

The application schema shall import the GML schema as described in 21.2.3. Any GML profile referenced from the application schema shall include at least the `gml:Observation` element and all schema components used by this element.

21.9.3 User-defined observation types

All observation types defined in the GML application schema shall be declared as global elements in the schema, i.e. they shall be child elements of the XML Schema <schema> element. The content model for such global elements shall derive by extension either directly or indirectly from `gml:ObservationType`.

Authors of application schemas may create their own observation types if GML lacks the desired construct. To do this, authors shall ensure that the object elements of these observation types are in the substitution group (either directly or indirectly) of the corresponding GML object element: `gml:Observation`.

NOTE 1 These elements are indirectly in the substitution group of `gml:AbstractFeature` and hence the condition of the feature model is satisfied.

NOTE 2 This International Standard provides the concrete simple observation types `gml:Observation`, `gml:DirectedObservation` and `gml:DirectedObservationAtDistance`. Application observation types may derive from any of these as well.

All rules specified in Clauses 7, 9 and 18 shall be followed.

21.9.4 Observation collections

All observation collections in the GML application schema shall be declared as global elements in the schema, i.e. they shall be child elements of the XML Schema <schema> element. An observation collection shall be a feature collection as described in 9.9 whose members are observations.

21.9.5 Observations are features

An observation defined in an application schema shall conform to the rules respecting GML features as described in Clause 9 and Clause 7. See also 21.3.4.

21.9.6 Observation collection document

Following the rules for GML documents (see 21.1), an Observation Collection document may reference observations that are defined in any number of GML application schemas and these may define observations only, observation collections or any combination of the same.

21.10 Schemas defining dictionaries and definitions

21.10.1 Introduction

The following subclauses describe how to create an application schema for definitions. Definitions and dictionaries are described in Clause 15. One set of specialized definitions is built in to GML, for units of measure, and serves as an example of how to derive specialized definition components.

An application schema for definitions defines one or more types of definition in accordance with the following rules.

21.10.2 Import GML dictionary schema components

The application schema shall import the GML schema as described in 21.2.3. Any GML profile referenced from the application schema shall include at least the `gml:Definition` element and all schema components used by this element.

21.10.3 User-defined definition types

All definitions in the application schema shall be declared as global elements in the schema, i.e. they shall be immediate child elements of the XML Schema <schema> element. The content model for such global elements shall derive either directly or indirectly from `gml:DefinitionType`.

Authors of application schemas may create their own definition types if GML lacks the desired construct. To do this, authors shall ensure that the object elements of these definition types are in the substitution group (either directly or indirectly) of the corresponding GML object element: `gml:Definition`.

All rules specified in Clauses 7 and 15 shall be followed.

21.10.4 User-defined dictionary types

A dictionary in the application schema shall be declared as a global element in the schema, that is it shall be a child element of the XML Schema <schema> element. The content model for such global elements shall derive either directly or indirectly from `gml:DictionaryType`.

Authors of application schemas may create their own dictionary types if GML lacks the desired construct. To do this, authors shall ensure that the object elements of these definition types are in the substitution group (either directly or indirectly) of the corresponding GML object element: `gml:Dictionary`.

All rules specified in Clauses 7 and 15 shall be followed.

21.11 Schemas defining values

21.11.1 Introduction

GML allows for user defined value types. Such value types may be used to express the property types of feature and other types of GML object. The basic root types for user-defined values are defined in 7.2.2.1. An alternative form for the expression of values is contained in 16.4. This is used mainly to provide values for the `gml:resultOf` parameter for an observation.

21.11.2 Import GML value objects schema components

The application schema shall import the GML schema as described in 21.2.3. Any GML profile referenced from the application schema shall include at least the `gml:Value` group and all schema components used by this element.

21.11.3 Construction of new value types

New value types may be created by derivation (typically by restriction) from any of the root types shown in Table 9.

Table 9 — Construction of new value types

Content model	Description
`gml:MeasureType`	A numerical quantity with a unit of measure (uom)
`gml:CategoryType`	A classification
`gml:CountType`	A count of occurrences, incidences etc.

Some standard value types can be found in the schema components specified in 16.3.

21.12 GML profiles of the GML schema

Typically a GML application schema will only require a limited subset of the schema components of the GML schema. It is recommended to identify and document the GML profile, see Clause 20, required by a GML application schema. Subclauses 21.3 to 21.11 contain some guidelines which schema components may be required depending on the type of the GML application schema.

NOTE 1 Annex G contains a method to automatically create a GML profile based on a list of schema components explicitly required by a GML application schema.

As a starting point, consider the following guidelines:

a) In an application schema modelling geographic features, `gml:AbstractFeature` and all schema components required by this element are required. See Clause 9.

b) In an application schema modelling also feature collections, `gml:AbstractFeatureMemberType` and `gml:AggregationAttributeGroup` are required, too, as well as all schema components required by them. See Clause 7.

c) If the features have properties which make use of units of measure, `gml:MeasureType` and all specific subtypes, e.g. `gml:LengthType`, are required as well as all schema components required by them (see Clause 8). `gml:BaseUnit`, `gml:DerivedUnit`, and/or `gml:ConventionalUnit` (and all schema components required by them, see Clause 16) are not required unless the application is defining units of measure such as appear in a units of measure dictionary.

d) If the application schema uses 0-dimensional spatial geometries, `gml:Point` is required (and all schema components required by it). See Clause 10.

e) If the application schema uses only simple 1-dimensional spatial geometries with linear interpolation, only `gml:LineString` is required (and all schema components required by it). See Clause 10.

f) If the application schema uses additional interpolation types, `gml:Curve` and any number of curve segments depending on the application (but at least one) are required. Again, this includes all schema components required by these elements. See 10.5.10.

g) If the application schema uses only simple 2-dimensional spatial geometries with linear interpolation along their boundaries without sharing boundary elements, only `gml:Polygon` and `gml:LinearRing` are required (and all schema components required by them). See Clause 10.

h) If the application schema uses additional interpolation types or surface patches, `gml:Surface` and any number of surface patches depending on the application (but at least one) are required. If surfaces shall share geometric primitives along their boundaries, `gml:Ring` is required, too. Again, this includes all schema components required by these elements. See 10.5.10.

i) If the application schema uses 3-dimensional spatial geometries, `gml:Solid` is required (and all schema components required by it). See 10.5.10.

j) The geometric aggregates schema components described in Clause 11 are required only, if the features use geometric objects that are collections of geometric primitives in their spatial properties.

k) The geometric complex and composites schema components described in Clause 11 are required only, if the features use geometric complexes in their spatial properties.

l) The topology schema components described in Clause 13 are required only, if the features have topology properties.

m) The Coordinate Reference System schema components described in Clause 12 are required only, if the application requires constructing or processing Coordinate Reference System dictionary entries (and their supporting components).

EXAMPLE 1 Prime Meridians, Geodetic Datums, etc. are supporting components.

n) The temporal schemas described in Clause 14 is required only, if the application schema is concerned with time dependent feature properties or dynamic features.

o) The coverage schema components described in Clause 19 are required only, if the application involves constructing or processing coverages.

EXAMPLE 2 Remotely sensed images, aerial photographs, soil distribution, digital elevation models are typical coverages.

p) The observation schema components described in Clause 18 are required only, if the application schema is concerned with modelling acts of observation such as taking photographs or making measurements. In the latter case, value objects and measure schema components are in most cases required, too.

q) The direction schema components described in Clause 17 are required only, if the application schema requires direction constructs such as compass bearings. The direction schema is used by `gml:DirectedObservation`.

r) Property elements whose content model is derived by extension from `gml:AbstractMetadataPropertyType`, see 7.2.6, is used to specify application or object specific metadata application schemas.

NOTE 2 In many applications, you will only need to import the feature.xsd of Annex C as this transitively imports the simple geometry schemas and gmlBase.xsd. For a thorough discussion of schema dependencies and modularity see Annex J.

Annex A
(normative)

Abstract test suites for GML application schemas, GML profiles and GML documents

A.1 Abstract test suite for GML application schemas

A.1.1 Test cases for mandatory conformance requirements

A.1.1.1 Use of XML namespaces

a) Test Purpose: Verify the correct use of XML namespaces in a GML application schema.

b) Test Method: Check that all schema components in the application schema are associated with an XML namespace and that this namespace is not "http://www.opengis.net/gml/3.2".

c) Reference: ISO 19136:2007, 21.2.2.

d) Test Type: Basic Test.

A.1.1.2 General rules

a) Test Purpose: Verify that the GML application schema obeys the general rules for constructing GML application schemas.

b) Test Method: Inspect the application schema and check that it satisfies the general rules described in 21.2.1.

c) Reference: ISO 19136:2007, 21.2.1.

d) Test Type: Capability Test.

A.1.1.3 Import of GML schema components

a) Test Purpose: Verify that the GML application schema imports the full GML schema and references GML profiles correctly.

b) Test Method: Inspect the import statements in the application schema (the full GML schema must be directly or indirectly imported). In addition, if one or more GML profiles are referenced, check that the XML Schema components specified in the `gml:gmlProfileSchema` elements satisfy all mandatory conformance requirements of the Abstract Test Suite in A.2.1.

c) Reference: ISO 19136:2007, 20.5, 21.2.3.

d) Test Type: Capability Test.

A.1.1.4 Valid XML Schema

a) Test Purpose: Verify the validity of the GML application schema XML document against the XML Schema specification.

b) Test Method: Validate the XML document of the GML application schema against the XML Schema specification. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the XML Schema specification.

c) Reference: ISO 19136:2007, 21.2; W3C XML Schema Part 1, W3C XML Schema Part 2 (see Clause 3).

d) Test Type: Capability Test.

A.1.1.5 Support for the GML model and syntax

a) Test Purpose: Verify that the GML application schema follows the rules for the encoding of objects and properties.

b) Test Method: Check the application schema.

c) Reference: ISO 19136:2007, 7.1, Clause 21.

d) Test Type: Capability Test.

A.1.1.6 Substitution group of object elements, type derivation

a) Test Purpose: Verify that all objects in the GML application schema are in the correct substitution group.

b) Test Method: Check the application schema that all object elements with identity are (directly or indirectly) in the substitution group of gml:AbstractGML. Check that the rules for derivation from base types stated in A.1.1.6 c) are followed.

c) Reference: ISO 19136:2007, 6.2, 7.2.2, 21.2.4, 21.2.5.

d) Test Type: Capability Test.

A.1.1.7 Property elements are not object elements

a) Test Purpose: Verify that all property elements in the GML application schema are not objects.

b) Test Method: Check the application schema that every child element of every object element is neither directly or indirectly in the substitution group of gml:AbstractObject.

c) Reference: ISO 19136:2007, 7.2.2, 21.2.5.

d) Test Type: Capability Test.

A.1.1.8 Content model of property elements

a) Test Purpose: Verify that all property elements in the GML application schema have a valid content model.

b) Test Method: Check every child element of every object element in the application schema.

c) Reference: ISO 19136:2007, 7.2.3, 21.2.6, 21.2.7.

d) Test Type: Capability Test.

A.1.1.9 Metadata and data quality properties

a) Test Purpose: Verify that all properties where the value is metadata about an object can be identified as a metadata property.

b) Test Method: Check the GML application schema that the content model of all metadata valued property elements is derived by extension from gml:AbstractMetadataPropertyType.

c) Reference: ISO 19136:2007, 7.2.6.

d) Test Type: Capability Test.

A.1.1.10 Spatial geometry properties

a) Test Purpose: Verify that all properties where the value is a spatial geometry object can be identified as such.

b) Test Method: Check the GML application schema that all properties with a geometric object or a collection of such objects are declared in accordance with 9.5.

c) Reference: ISO 19136:2007, 9.5.

d) Test Type: Capability Test.

A.1.1.11 Spatial topology properties

a) Test Purpose: Verify that all properties where the value is a spatial topology object can be identified as such.

b) Test Method: Check the GML application schema that all properties with a topological object or a collection of such objects are declared in accordance with 9.6.

c) Reference: ISO 19136:2007, 9.6.

d) Test Type: Capability Test.

A.1.1.12 Temporal properties

a) Test Purpose: Verify that all properties where the value is a temporal object can be identified as such.

b) Test Method: Check the GML application schema that all properties with a temporal object or a collection of such objects are declared in accordance with 9.7.

c) Reference: ISO 19136:2007, 9.7.

d) Test Type: Capability Test.

A.1.1.13 Location properties

a) Test Purpose: Verify that all properties where the value is a location description or reference can be identified as such.

b) Test Method: Check the GML application schema that all properties with spatial references by geographic identifiers use the property elements gml:locationName or gml:locationReference.

c) Reference: ISO 19136:2007, 9.4.2.

d) Test Type: Capability Test.

A.1.1.14 GML object collections

a) Test Purpose: Verify that all objects that are collections of GML objects can be identified as such.

b) Test Method: Check the GML application schema that such objects have one or more property elements with a content model that extend gml:AbstractMemberType. Check also that, if appropriate, the gml:aggregationType attribute is a child node of the object element.

c) Reference: ISO 19136:2007, 7.2.5.

d) Test Type: Capability Test.

A.1.1.15 Substitution group of feature elements

a) Test Purpose: Verify that all features in the GML application schema are in the correct substitution group.

b) Test Method: Check the application schema that all object elements representing features are (directly or indirectly) in the substitution group of gml:AbstractFeature.

c) Reference: ISO 19136:2007, 9.3.

d) Test Type: Capability Test.

A.1.1.16 GML feature collections

a) Test Purpose: Verify that all features that are collections of GML Features can be identified as such.

b) Test Method: Check the GML application schema that such features have one or more property elements with a content model that extends gml:AbstractFeatureMemberType. Check also that, if appropriate, the gml:aggregationType attribute is a child node of the object element.

c) Reference: ISO 19136:2007, 9.9.

d) Test Type: Capability Test.

A.1.2 Test cases for GML application schemas converted from an ISO 19109 Application Schema in UML

A.1.2.1 Valid ISO 19109 Application Schema

a) Test Purpose: If the GML application schema is mapped from an ISO 19109 Application Schema in UML, verify that the UML application schema satisfies the requirements of ISO 19109.

b) Test Method: Check the conformance of the UML application schema with ISO 19109 and check that the UML application schema has been constructed in accordance with E.2.1.

c) Reference: ISO 19109:2005, Clause 2, Annex A; ISO 19136:2007, E.2.1.

d) Test Type: Capability Test.

A.1.2.2 Mapping from an ISO 19109 Application Schema in UML

a) Test Purpose: If the ISO 19109 Application Schema in UML satisfies the requirements stated in A.1.2, verify that the GML application schema has been derived from the UML application schema correctly.

b) Test Method: Compare both descriptions of the application schema and check whether the conversion from UML to XML Schema is in accordance with the conversion rules in E.2.4.

c) Reference: ISO 19136:2007, E.2.4.

d) Test Type: Capability Test.

A.1.3 Test cases for ISO 19109 Application Schemas in UML converted from a GML application schema

A.1.3.1 Valid GML application schema

a) Test Purpose: If the GML application schema is mapped to an ISO 19109 Application Schema in UML, verify that the GML application schema has been constructed correctly.

b) Test Method: Check that the GML application schema has been constructed in accordance with F.2.1.

c) Reference: ISO 19136:2007, F.2.1.

d) Test Type: Capability Test.

A.1.3.2 Mapping to an ISO 19109 Application Schema in UML

a) Test Purpose: If the GML application schema satisfies the requirements stated in A.1.3, verify that the ISO 19109 Application Schema in UML has been derived from the GML application schema correctly.

b) Test Method: Compare both descriptions of the application schema and check whether the conversion from XML Schema to UML is in accordance with the conversion rules in F.2.3.

c) Reference: ISO 19136:2007, F.2.3.

d) Test Type: Capability Test.

A.1.4 GML application schemas defining features and feature collections

a) Test Purpose: If the GML application schema defines features, verify that the GML application schema has been constructed correctly.

b) Test Method: Check that the GML application schema has been constructed in accordance with 21.3.

c) Reference: ISO 19136:2007, 21.3.

d) Test Type: Capability Test.

A.1.5 GML application schemas defining spatial geometries

a) Test Purpose: If the GML application schema defines spatial geometric objects, verify that the GML application schema has been constructed correctly.

b) Test Method: Check that the GML application schema has been constructed in accordance with 21.4.

c) Reference: ISO 19136:2007, 21.4.

d) Test Type: Capability Test.

A.1.6 GML application schemas defining spatial topologies

a) Test Purpose: If the GML application schema defines spatial topology objects, verify that the GML application schema has been constructed correctly.

b) Test Method: Check that the GML application schema has been constructed in accordance with 21.5.

c) Reference: ISO 19136:2007, 21.5.

d) Test Type: Capability Test.

A.1.7 GML application schemas defining time

a) Test Purpose: If the GML application schema defines temporal objects, verify that the GML application schema has been constructed correctly.

b) Test Method: Check that the GML application schema has been constructed in accordance with 21.6.

c) Reference: ISO 19136:2007, 21.6.

d) Test Type: Capability Test.

A.1.8 GML application schemas defining coordinate reference systems

a) Test Purpose: If the GML application schema defines coordinate reference system objects, verify that the GML application schema has been constructed correctly.

b) Test Method: Check that the GML application schema has been constructed in accordance with 21.7.

c) Reference: ISO 19136:2007, 21.7.

d) Test Type: Capability Test.

A.1.9 GML application schemas defining coverages

a) Test Purpose: If the GML application schema defines coverage features, verify that the GML application schema has been constructed correctly.

b) Test Method: Check that the GML application schema has been constructed in accordance with 21.8.

c) Reference: ISO 19136:2007, 21.8.

d) Test Type: Capability Test.

A.1.10 GML application schemas defining observations

a) Test Purpose: If the GML application schema defines simple observation features, verify that the GML application schema has been constructed correctly.

b) Test Method: Check that the GML application schema has been constructed in accordance with 21.9.

c) Reference: ISO 19136:2007, 21.9.

d) Test Type: Capability Test.

A.1.11 GML application schemas defining dictionaries and definitions

a) Test Purpose: If the GML application schema defines dictionary and definition objects, verify that the GML application schema has been constructed correctly.

b) Test Method: Check that the GML application schema has been constructed in accordance with 21.10.

c) Reference: ISO 19136:2007, 21.10.

d) Test Type: Capability Test.

A.1.12 GML application schemas defining values

a) Test Purpose: If the GML application schema defines value objects, verify that the GML application schema has been constructed correctly.

b) Test Method: Check that the GML application schema has been constructed in accordance with 21.11.

c) Reference: ISO 19136:2007, 21.11.

d) Test Type: Capability Test.

A.2 Abstract test suite for GML profiles

A.2.1 Valid GML profile

a) Test Purpose: Verify that a profile is a GML profile in accordance with the rules and guidelines stated in Clause 20 and 21.12.

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, Clause 20, 21.12.

d) Test Type: Capability Test.

A.2.2 Geometric primitives (spatial)

A.2.2.1 Data types for geometric primitives

A.2.2.1.1 Data types for 0-dimensional geometry

a) Test Purpose: Verify that a GML profile includes gml:Point and gml:PointPropertyType. If the GML profile also includes gml:MultiPoint, verify that it includes gml:MultiPointPropertyType. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO, 19136 10.3, 11.3.2.

d) Test Type: Capability Test.

A.2.2.1.2 Data types for 1-dimensional geometry

a) Test Purpose: Verify that the GML profile satisfies all the requirements of A.2.2.1.1 and includes gml:Curve, gml:LineStringSegment, gml:LineString, and gml:CurvePropertyType. If the GML profile also includes gml:MultiCurve, verify that it includes gml:MultiCurvePropertyType. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.2.1.1, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 to 10.4.7.21, 11.3.3.

d) Test Type: Capability Test.

A.2.2.1.3 Data types for 2-dimensional geometry

a) Test Purpose: Verify that the GML profile satisfies all the requirements of A.2.2.1.2 and includes gml:Surface, gml:PolygonPatch, gml:Polygon, gml:SurfacePropertyType, gml:LinearRing, and gml:Ring. If the GML profile also includes gml:MultiSurface, verify that it includes gml:MultiSurfacePropertyType. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.2.1.2, 10.5.1, 10.5.2, 10.5.4 to 10.5.9, 10.5.10, 10.5.11.1, 10.5.12.4 to 10.5.11.6, 11.3.4.

d) Test Type: Capability Test.

A.2.2.1.4 Data types for 3-dimensional geometry

a) Test Purpose: Verify that the GML profile satisfies all the requirements of A.2.2.1.3 and includes gml:Solid, gml:SolidPropertyType, and gml:Shell. If the GML profile also includes gml:MultiSolid, verify that it includes gml:MultiSolidPropertyType. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.2.1.3, 10.6.1, 10.6.2, 10.6.4 to 10.6.6, 11.3.5.

d) Test Type: Capability Test.

A.2.3 Geometric complexes (spatial)

A.2.3.1 Data types for geometric complexes

A.2.3.1.1 Data types for 1-dimensional geometric complexes

a) Test Purpose: Verify that the GML profile satisfies all the requirements of A.2.2.1.2 and includes gml:CompositeCurve, gml:OrientableCurve, gml:GeometricComplex, and gml:GeometricComplexPropertyType. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.2.1.2, 10.4.6, 11.2.1.1, 11.2.1.2, 11.2.2.1, 11.2.2.2.

d) Test Type: Capability Test.

A.2.3.1.2 Data types for 2-dimensional geometric complexes

a) Test Purpose: Verify that the GML profile satisfies all the requirements of A.2.2.1.3 and A.2.3.1.1 and includes gml:CompositeSurface and gml:OrientableSurface. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.2.1.3, A.2.3.1.1, 10.5.11, 11.2.2.3.

d) Test Type: Capability Test.

A.2.3.1.3 Data types for 3-dimensional geometric complexes

a) Test Purpose: Verify that the GML profile satisfies all the requirements of A.2.2.1.4 and A.2.3.1.2 and includes gml:CompositeSolid. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.2.1.4, A.2.3.1.2, 11.2.2.4.

d) Test Type: Capability Test.

A.2.4 Topologic complexes (spatial)

A.2.4.1 Data types for topologic complexes

A.2.4.1.1 Data types for 1-dimensional topologic complexes

a) Test Purpose: Verify that the GML profile includes gml:TopoComplex, gml:TopoComplexPropertyType, gml:Node, gml:directedNode, gml:Edge, and gml:directedEdge. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1" except for gml:pointProperty in gml:Node and gml:curveProperty in gml:Edge.

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, 13.2, 13.3.1, 13.3.2, 13.3.3, 13.5.

d) Test Type: Capability Test.

A.2.4.1.2 Data types for 2-dimensional topologic complexes

a) Test Purpose: Verify that the GML profile satisfies all requirements of A.2.4.1.1 and includes gml:Face, and gml:directedFace. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1" except for gml:surfaceProperty in gml:Face.

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.4.1.1, 13.3.4.

d) Test Type: Capability Test.

A.2.4.1.3 Data types for 3-dimensional topologic complexes

a) Test Purpose: Verify that the GML profile satisfies all requirements of A.2.4.1.2 and includes gml:TopoSolid, and gml:directedTopoSolid. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1" except for gml:solidProperty in gml:TopoSolid.

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.4.1.2, 13.3.5.

d) Test Type: Capability Test.

A.2.5 Topologic complexes with geometric realization (spatial)

A.2.5.1 Data types for topologic complexes with geometric realization

A.2.5.1.1 Data types for 1-dimensional topologic complexes with geometric realization

a) Test Purpose: Verify that the GML profile satisfies all requirements of A.2.2.1.1, A.2.2.1.2 and A.2.4.1.1. Verify that it includes the properties gml:pointProperty in gml:Node and gml:curveProperty in gml:Edge with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.2.1.1, A.2.2.1.2, A.2.4.1.1, 13.3.2, 13.3.3.

d) Test Type: Capability Test.

A.2.5.1.2 Data types for 2-dimensional topologic complexes with geometric realization

a) Test Purpose: Verify that the GML profile satisfies all requirements of A.2.2.1.3 and A.2.4.1.2. Verify that it includes the property gml:surfaceProperty in gml:Face with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.2.1.3, A.2.4.1.2, 13.3.4.

d) Test Type: Capability Test.

A.2.5.1.3 Data types for 3-dimensional topologic complexes with geometric realization

a) Test Purpose: Verify that the GML profile satisfies all requirements of A.2.2.1.4 and A.2.4.1.3. Verify that it includes the property gml:solidProperty in gml:TopoSolid with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.2.1.4, A.2.4.1.3, 13.3.5.

d) Test Type: Capability Test.

A.2.6 Coordinate reference systems

a) Test Purpose: Verify that the GML profile contains all schema components defined in Clause 12 that are identified as mandatory or mandatory under the conditions in accordance with ISO 19111.

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, Clause 12, ISO 19111:2007, Tables 1 to 41 and, in the case of projected coordinate reference systems, Tables 42 to 56.

d) Test Type: Capability Test.

A.2.7 Coordinate operations

a) Test Purpose: Verify that the GML profile contains all schema components defined in 12.6 that are identified as mandatory or mandatory under the conditions in accordance with ISO 19111.

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, 12.6, ISO 19111:2007, Tables 42 to 56.

d) Test Type: Capability Test.

A.2.8 Temporal geometry

A.2.8.1 Data types for 0-dimensional geometry

a) Test Purpose: Verify that a GML profile includes gml:TimeInstant and gml:TimeInstantPropertyType. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, 14.2.1.1, 14.2.1.2, 14.2.1.4, 14.2.2.2, 14.2.2.3, 14.2.2.4, 14.2.2.7.

d) Test Type: Capability Test.

A.2.8.2 Data types for 1-dimensional geometry

a) Test Purpose: Verify that a GML profile satisfies the requirements of A.2.8.1 and includes gml:TimePeriod and gml:TimePeriodPropertyType. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.8.1, 14.2.2.5, 14.2.2.6.

d) Test Type: Capability Test.

A.2.9 Temporal topology

a) Test Purpose: Verify that a GML profile satisfies the requirements of A.2.8.2 and includes gml:TimeNode, gml:TimeNodePropertyType, gml:TimeEdge, gml:TimeEdgePropertyType, gml:TimeTopologyComplex, and gml:TimeTopologyComplexType. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, 14.3.2.

d) Test Type: Capability Test.

A.2.10 Temporal reference systems

a) Test Purpose: Verify that a GML profile includes gml:TimeReferenceSystem and at least one of gml:TimeCoordinateSystem, gml:TimeCalendar, gml:TimeClock, gml:TimeOrdinalReferenceSystem. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, 14.4.

d) Test Type: Capability Test.

A.2.11 Dynamic features

a) Test Purpose: Verify that a GML profile satisfies the requirements of A.2.8.2 and includes gml:DynamicFeature, gml:AbstractTimeSlice, and gml:history. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.8.2, 14.5.

d) Test Type: Capability Test.

A.2.12 Dictionaries

a) Test Purpose: Verify that a GML profile includes gml:Dictionary. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, Clause 15.

d) Test Type: Capability Test.

A.2.13 Units dictionaries

a) Test Purpose: Verify that a GML profile includes gml:UnitDictionary, gml:BaseUnit, gml:DerivedUnit, and gml:ConventionalUnit. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, 16.2.

d) Test Type: Capability Test.

A.2.14 Observations

a) Test Purpose: Verify that a GML profile includes gml:Observation and gml:DirectedObservation. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, 17.1, 18.2.

d) Test Type: Capability Test.

A.2.15 Coverages

A.2.15.1 Abstract coverage

a) Test Purpose: Verify that a GML profile includes gml:AbstractCoverage. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, 16.4, 19.3.1, 19.3.4 to 19.3.21.

d) Test Type: Capability Test.

A.2.15.2 Discrete point coverage

a) Test Purpose: Verify that a GML profile satisfies the requirements of A.2.15.1 and includes gml:MultiPointCoverage. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.15.1, 19.3.16.

d) Test Type: Capability Test.

A.2.15.3 Discrete curve coverage

a) Test Purpose: Verify that a GML profile satisfies the requirements of A.2.15.1 and includes gml:MultiCurveCoverage. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.15.1, 19.3.17.

d) Test Type: Capability Test.

A.2.15.4 Discrete surface coverage

a) Test Purpose: Verify that a GML profile satisfies the requirements of A.2.15.1 and includes gml:MultiSurfaceCoverage. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.15.1, 19.3.18.

d) Test Type: Capability Test.

A.2.15.5 Discrete solid coverage

a) Test Purpose: Verify that a GML profile satisfies the requirements of A.2.15.1 and includes gml:MultiSurfaceCoverage. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.15.1, 19.3.19.

d) Test Type: Capability Test.

A.2.15.6 Grid coverage

a) Test Purpose: Verify that a GML profile satisfies the requirements of A.2.15.1 and includes gml:GridCoverage and gml:RectifiedGridCoverage. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.15.1, 19.3.20, 19.3.21.

d) Test Type: Capability Test.

A.2.15.7 Continuous coverage

a) Test Purpose: Verify that a GML profile satisfies the requirements of A.2.15.1 and includes gml:AbstractContinuousCoverage. Verify that all non-deprecated properties of the object elements are part of the profile with a maxOccurs value of at least "1".

b) Test Method: Inspect the profile.

c) Reference: ISO 19136:2007, A.2.15.1, 19.3.3.

d) Test Type: Capability Test.

A.3 Abstract test suite for GML documents

A.3.1 Existence of a reference to an applicable GML application schema

a) Test Purpose: To verify the existence of a reference to a GML application schema applicable to the GML document.

b) Test Method: Check that an XML Schema file representing a GML application schema is referenced in the xsi:schemaLocation attribute of the root element of the GML document.

c) Reference: ISO 19136:2007, 21.1.

d) Test Type: Basic Test.

A.3.2 Existence of the referenced GML application schema

a) Test Purpose: To verify the existence of a GML application schema applicable to the GML document.

b) Test Method: Check that the XML Schema file representing the GML application schema referenced from the GML document can be accessed. Check that also all documents directly or indirectly accessed by the referenced file can be accessed.

c) Reference: ISO 19136:2007, 21.1.

d) Test Type: Basic Test.

A.3.3 Conformance of the referenced GML application schema

a) Test Purpose: Verify that the GML application schema referenced from the GML document is conformant to this International Standard.

b) Test Method: Verify that the application schema has passed all of the applicable tests specified in A.1.

c) Reference: ISO 19136:2007, A.1.

d) Test Type: Capability Test.

A.3.4 Valid XML

a) Test Purpose: Verify the validity of the GML document against the XML Schema components of the conformant GML application schema.

b) Test Method: Validate GML document against the referenced GML application schema. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the XML Schema specification.

c) Reference: ISO 19136:2007, 21.1.

d) Test Type: Capability Test.

A.3.5 Conformance of a GML document

a) Test Purpose: Verify that the GML document complies with all other constraints specified by this International Standard.

b) Test Method: Check that the requirements A.3.1 to A.3.3 are satisfied, that the GML document satisfies the requirements of A.3.4 and that it complies with all other constraints specified by this International Standard.

c) Reference: ISO 19136:2007, Clauses 7 to 21, in particular 7.2.3.4, 10.1.3.2, 10.1.3.3, 10.1.4.2, 16.4.11.

d) Test Type: Capability Test.

Annex B
(normative)

Abstract test suite for software implementations

B.1 Test cases for mandatory conformance requirements

B.1.1 GML profile

a) Test Purpose: Verify that a GML profile has been documented that is fully supported by the software implementation.

b) Test Method: Check the documentation of the software implementation to identify the profile. Check the profile that is satisfies the requirements of the Abstract Test Suite in A.1.12. Check further that the software implementation fully supports the profile and the semantics associated with all schema components in the profile.

c) Reference: ISO 19136:2007, A.1.12.

d) Test Type: Capability Test.

B.1.2 Support for local simple Xlinks

a) Test Purpose: If the software implementation has the capability to process GML object elements in XML format, verify that an implementation supports references to other objects within the same GML document.

b) Test Method: Check that the implementation can process property instances that use the xlink:href attribute with a content of a shorthand Xpointer pointing to a resource within the same XML document.

c) Reference: ISO 19136:2007, 8.1.

d) Test Type: Capability Test.

B.1.3 Coordinate reference systems used in features (software implementation)

a) Test Purpose: If the software implementation has the capability to process GML object elements in XML format and if the GML profile of the implementation includes features, verify that the mechanism for setting the default coordinate reference system for all geometric objects within a feature is followed.

b) Test Method: Check the Implementation that the srsName attribute of a gml:Envelope element that is the value of the gml:boundedBy property of a feature is used as the default coordinate reference system for all geometric objects encoded inline of the feature element.

c) Reference: ISO 19136:2007, 9.10.

d) Test Type: Capability Test.

B.2 Test cases for optional conformance requirements for software implementations with the capability to process GML object elements in XML format

B.2.1 Support for remote simple Xlinks

a) Test Purpose: Verify that an implementation supports references to other objects within or outside the same GML document.

b) Test Method: Check that the implementation satisfies the requirements of A.1.1.1 and can process property instances that use the xlink:href attribute with a content of an Xpointer pointing to a resource outside the same XML document.

c) Reference: ISO 19136:2007, 8.1.

d) Test Type: Capability Test.

B.2.2 Support for extended Xlinks

a) Test Purpose: Verify that an implementation supports extended Xlinks.

b) Test Method: Check that the implementation can process extended Xlink attributes.

c) Reference: ISO 19136:2007, 8.1.

d) Test Type: Capability Test.

B.2.3 Support for nillable properties

a) Test Purpose: Verify that an implementation supports nillable properties.

b) Test Method: Check that the implementation can process GML application schemas with property element declarations with the attribute xsi:nillable and that the implementation can process instances with the attributes xsi:nil and gml:nilReason in these elements.

c) Reference: ISO 19136:2007, 8.2.3.1 to 8.2.3.2.

d) Test Type: Capability Test.

B.2.4 Support for units of measurement

a) Test Purpose: Verify that an implementation can convert between two units of the same kind.

b) Test Method: Check that the implementation can process values in the uom attribute of gml:MeasureType as specified in 7.3.3.7 and convert measures to another unit of the same kind using a units dictionary as specified in 16.2.

c) Reference: ISO 19136:2007, 8.2.3.6, 16.2.

d) Test Type: Capability Test.

B.2.5 Support for ownership semantics of properties

a) Test Purpose: Verify that an implementation supports the "owns" attribute.

b) Test Method: If an implementation is capable of deleting objects from a GML document, check that the implementation deletes all objects that owned by another object as indicated by the owns attribute, if that object is deleted.

c) Reference: ISO 19136:2007, 7.2.3.5.

d) Test Type: Capability Test.

B.2.6 Metadata properties

a) Test Purpose: Verify that properties where the value is metadata about an object are identified as a metadata property.

b) Test Method: Check the implementation that property elements whose content model is derived from gml:AbstractMetadataPropertyType are identified as metadata properties.

c) Reference: ISO 19136:2007, 7.2.6.

d) Test Type: Capability Test.

B.2.7 Support for GML profiles in instance validation

a) Test Purpose: Verify that an implementation can use GML profiles for instance validation.

b) Test Method: Check that the implementation uses the GML profiles for instance validation if the profiles are referenced from an application schema using a `gml:gmlProfileSchema` annotation.

c) Reference: ISO 19136:2007, 20.5.

d) Test Type: Capability Test.

B.3 Test cases for writing GML

B.3.1 Serialization capability

a) Test Purpose: Verify the existence of the serialization operation of the implementation.

b) Test Method: Inspect the software implementation and its documentation to check that the implementation implements a serialization operation that writes valid instances of GML objects in XML format.

c) Reference: ISO 19136:2007, A.3.

d) Test Type: Basic Test.

B.3.2 Serialization validity

a) Test Purpose: Verify that the result of the serialization operation is conformant with this International Standard.

b) Test Method: Write typical GML documents using the serialization operation and check that the GML objects in XML format are valid.

c) Reference: ISO 19136:2007, A.3.

d) Test Type: Capability Test.

B.4 Test case for reading GML

a) Test Purpose: If the implementation has the capability to create implementation objects from GML object elements in XML and to serialize these implementation objects back to GML objects in XML format, verify that it does so validly.

b) Test Method: Check that successive actions of object creation and serialization (see B.1.2) produce the result that is without loss of information.

c) Reference: ISO 19136:2007, A.3.

d) Test Type: Capability Test.

B.5 Test cases for writing GML application schemas

B.5.1 Serialization capability

a) Test Purpose: Verify the existence of the serialization operation of the implementation.

b) Test Method: Inspect the software implementation and its documentation to check that the implementation implements a serialization operation that writes valid instances of GML application schemas in XML Schema format.

c) Reference: ISO 19136:2007, A.1.

d) Test Type: Basic Test.

B.5.2 Serialization validity

a) Test Purpose: Verify that the result of the serialization operation is conformant with this International Standard.

b) Test Method: Create typical GML application schemas using the serialization operation and check that the GML application schemas conform to the GML profile of the implementation and to this International Standard.

c) Reference: ISO 19136:2007, A.1, B.1.1.

d) Test Type: Capability Test.

B.6 Test cases for reading GML application schemas

a) Test Purpose: If the implementation has the capability to create implementation objects from GML application schema in XML and to serialize these implementation objects back to GML application schemas in XML format, verify that it does so validly.

b) Test Method: Check that successive actions of object creation and serialization (see B.5) produce the result that is without loss of information.

c) Reference: ISO 19136:2007, A.1, B.5.

d) Test Type: Capability Test.

Annex C
(informative)

GML schema

XML Schema documents with the GML schema are available online at:

http://schemas.opengis.net/gml/3.2.1/

NOTE The use of "3.2.1" in the URL is unchanged since this version 3.2.2 is a corrigendum to version 3.2.1 and the corrected schema replaces the GML 3.2.1 schema.

The schema components are modularized in the structure shown in Annex J, i.e. into the following schema documents:

- basicTypes.xsd
- coordinateOperations.xsd
- coordinateReferenceSystems.xsd
- coordinateSystems.xsd
- coverage.xsd
- datums.xsd
- dictionary.xsd
- direction.xsd
- dynamicFeature.xsd
- feature.xsd
- geometryAggregates.xsd
- geometryBasic0d1d.xsd
- geometryBasic2d.xsd
- geometryComplexes.xsd
- geometryPrimitives.xsd
- gml.xsd
- gmlBase.xsd
- grids.xsd

— measures.xsd

— observation.xsd

— referenceSystems.xsd

— temporal.xsd

— temporalReferenceSystems.xsd

— temporalTopology.xsd

— topology.xsd

— units.xsd

— valueObjects.xsd

The additional document defaultStyle.xsd contains informative schema components.

An Xlinks XML Schema document is located at http://www.w3.org/1999/xlink.xsd.

Annex D
(normative)

Implemented Profile of the ISO 19100 series of International Standards and Extensions

D.1 General remarks

The general relationship between the ISO 19100 series of International Standards and GML is discussed in Clause 6. This annex describes in detail the profile of the conceptual model defined in the ISO 19100 series of International Standards implemented by GML (see D.2) as well as the extensions to this profile (see D.3).

In this International Standard “profile” means a pure subset.

D.2 Profile of the ISO 19100 series of International Standards used by GML

D.2.1 Overview

The following subclauses describe the profile of the ISO 19100 series of International Standards that is used by GML. In the description of the class diagrams of the profile, the relationship and mapping to the GML schema are discussed.

NOTE 1 In general the encoding rules discussed in Annex E were used also in the encoding of the GML schema. However, since the GML schema was mostly handcrafted it exploits more of the specific capabilities of the implementation environment, i.e. XML and XML Schema. Examples are a number of predefined basic types (simple or complex types with simple content) or the use of global elements also for properties to be made substitutable (e.g. to define aliases for deprecated property names). These cases are documented in the following subclauses or are straightforward.

Only elements from International Standards discussed below are part of the profile. Elements from other International Standards are not part of the profile.

Due to the nature of GML no operation of any class is part of the profile.

In addition, interface classes (stereotype <<Interface>>) without data structures and “Realization” relationships to classes without data structures have been deleted.

Furthermore, the navigability of associations has been restricted to the directions in which GML represents explicit object properties (most associations in the GML schema are navigable only in a single direction).

NOTE 2 No deprecated types, elements and attributes of GML are considered in this annex.

NOTE 3 The general rules for the UML-to-XML-Schema mapping for GML application schemas are defined in Annex E.

NOTE 4 In this annex the namespace "xsd:" is used to refer to the namespace of XML Schema, which is "http://www.w3.org/2001/XMLSchema". The namespace “gml:” refers to the namespace of GML, which is “http://www.opengis.net/gml/3.2”.

Table D.1 provides a mapping between the high-level packages of the ISO 19100 series of International Standards and the subclauses of this International Standard defining GML schema components implementing types from these packages.

Table 10 — Overview of the implemented packages of the ISO 19100 series of International Standards

UML package	UML class prefix	ISO 19136 subclause	Annex D subclause
ISO/TS 19103:Basic Types:Units of Measure	—	8.2.3.6, 16.2	D.2.2
ISO 19107:Geometry:Geometric root	GM	10.1.3	D.2.3.2
ISO 19107:Geometry:Geometric primitive	GM	10.2, 10.3, 10.4, 10.5, 10.6	D.2.3.3
ISO 19107:Geometry:Geometric complex	GM	11.2	D.2.3.6
ISO 19107:Geometry:Geometric aggregates	GM	11.3	D.2.3.5
ISO 19107:Geometry:Coordinate geometry	GM	10.1.4	D.2.3.4
ISO 19107:Topology:Topology root	TP	13.2	D.2.4.2
ISO 19107:Topology:Topology primitive	TP	13.3	D.2.4.3
ISO 19107:Topology:Topology complex	TP	13.5	D.2.4.4
ISO 19108:Temporal Objects	TM	14.2, 14.3	D.2.5.2 to D.2.5.6
ISO 19108:Temporal Reference System	TM	14.4	D.2.5.7
ISO 19111:SC_CoordinateReferenceSystem	SC	12.2, 12.3	D.2.7.3
ISO 19111:SC_CoordinateSystem	CS	12.4	D.2.7.4
ISO 19111:SC_Datum	CD	12.5	D.2.7.5
ISO 19111:SC_CoordinateOperation	CC	12.6	D.2.7.6
ISO 19123	CV	19	D.2.11

Table D.2 provides a mapping between conceptual UML classes implemented by this International Standard and the associated GML object element, XML Schema type and GML property type.

The table consists of four columns. To provide a complete mapping from the UML type (first column) to XML Schema as used by GML, three different mappings are required which are shown in the three other columns. This is a result of the differences in mapping the General Feature Model to UML and to XML Schema, mainly because XML Schema separates XML elements and their content model.

The table has to be read as follows:

— The first column ("UML class") lists a class from the ISO 19100 series which is implemented in the GML schema.

  In some cases, this column is empty ("—") which indicates that the GML elements and types in the other columns implement a concept that is not specified in the ISO 19100 series but is introduced in D.3.

— The second column ("GML object element") specifies the GML object element that implements the type.

  This information is in particular used in two situations in the encoding rules in Annex E:

  — when no predefined property type for the object element is part of the GML schema (see the fourth column) and a property type has to be created in the application schema,

- when a subtype of the type is specified in an application schema in which case the object element representing the subtype is to be defined as part of the substitution group of the object element of the type.

Where no corresponding object element exists in the GML schema, this is indicated by an empty cell ("—").

- The third column ("GML type") specifies the XML Schema type that defines the content model of the GML object element in the second column.

This XML Schema type is in particular used in the encoding rules in Annex E when a subtype of the type in the first column is modelled in an application schema; in this case, the XML Schema implementation will specify a derived type of the XML Schema.

Where no corresponding XML Schema type exists in the GML schema, this is indicated by an empty cell ("—").

- The fourth column ("GML property type") specifies the type that is used as the XML Schema type, if the type is used as a value of a property in the application schema. In this case, this column provides the value of the XML Schema type that is the implementation of that type in the GML schema.

EXAMPLE If a feature type has a property with a type of "GM_Point", then in the XML Schema representation the corresponding GML property element declaration has gml:PointPropertyType as its type.

In case of a class with stereotype <<DataType>>, no XML Schema representation of the property is provided as the data types specified in the GML schema are typically not intended to be used as values of feature properties.

In some cases, the GML schema does not contain a predefined property type for that type and if required by an application schema, the property type needs to be constructed in accordance with the rules for GML property types (see 7.2.3) where the GML object element is given in the second column of the same row.

If the value is annotated with "(group)", then the property is implemented by a reference to the global group stated in the cell instead of a local property element.

Table 11 — Implementation of types from the ISO 19100 series of International Standards

UML class	GML object element	GML type	GML property type
GM_Object	gml:AbstractGeometry	gml:AbstractGeometryType	gml:GeometryPropertyType
GM_Primitive	gml:AbstractGeometricPrimtive	gml:AbstractGeometricPrimtiveType	gml:GeometricPrimtivePropertyType
DirectPosition	—	—	gml:DirectPositionType
GM_Position	—	—	gml:geometricPositionGroup (group)
GM_PointArray	—	—	gml:geometricPositionListGroup (group)
GM_Point	gml:Point	gml:PointType	gml:PointPropertyType
GM_Curve	gml:Curve	gml:CurveType	gml:CurvePropertyType
GM_Surface	gml:Surface	gml:SurfaceType	gml:SurfacePropertyType
GM_PolyhedralSurface	gml:PolyhedralSurface	gml:PolyhedralSurfaceType	*anonymous property type*[a]
GM_TriangulatedSurface	gml:TriangulatedSurface	gml:TriangulatedSurfaceType	*anonymous property type*
GM_Tin	gml:Tin	gml:TinType	*anonymous property type*
GM_Solid	gml:Solid	gml:SolidType	gml:SolidPropertyType
GM_OrientableCurve	gml:OrientableCurve	gml:OrientableCurveType	gml:CurvePropertyType
GM_OrientableSurface	gml:OrientableSurface	gml:OrientableSurfaceType	gml:SurfacePropertyType
GM_Ring	gml:Ring	gml:RingType	—
GM_Shell	gml:Shell	gml:ShellType	—
—	gml:LineString	gml:LineStringType	—
—	gml:Polygon	gml:PolygonType	—
—	gml:LinearRing	gml:LinearRingType	—
GM_CompositePoint	gml:Point	gml:PointType	gml:PointPropertyType
GM_CompositeCurve	gml:CompositeCurve	gml:CompositeCurveType	*anonymous property type*
GM_CompositeSurface	gml:CompositeSurface	gml:CompositeSurfaceType	*anonymous property type*
GM_CompositeSolid	gml:CompositeSolid	gml:CompositeSolidType	*anonymous property type*
GM_Complex	gml:GeometricComplex	gml:GeometricComplexType	gml:GeometricComplexPropertyType
GM_Aggregate	gml:MultiGeometry	gml:MultiGeometryType	gml:MultiGeometryPropertyType
GM_MultiPoint	gml:MultiPoint	gml:MultiPointType	gml:MultiPointPropertyType
GM_MultiCurve	gml:MultiCurve	gml:MultiCurveType	gml:MultiCurvePropertyType
GM_MultiSurface	gml:MultiSurface	gml:MultiSurfaceType	gml:MultiSurfacePropertyType
GM_MultiSolid	gml:MultiSolid	gml:MultiSolidType	gml:MultiSolidPropertyType
GM_MultiPrimitive	gml:MultiGeometry	gml:MultiGeometryType	gml:MultiGeometryPropertyType
GM_CurveSegment	gml:AbstractCurveSegment	gml:AbstractCurveSegmentType	—
GM_Arc	gml:Arc	gml:ArcType	—
GM_ArcByBulge	gml:ArcByBulge	gml:ArcByBulgeType	—
—	gml:ArcByCenterPoint	gml:ArcByCenterPointType	—
GM_ArcString	gml:ArcString	gml:ArcStringType	—
GM_ArcStringByBulge	gml:ArcStringByBulge	gml:ArcStringByBulgeType	—
GM_Bezier	gml:Bezier	gml:BezierType	—
GM_BsplineCurve	gml:BSpline	gml:BSplineType	—
GM_Circle	gml:Circle	gml:CircleType	—
—	gml:CircleByCenterPoint	gml:CircleByCenterPointType	—

Table D.2 *(continued)*

UML class	GML object element	GML type	GML property type
GM_Clothoid	gml:Clothoid	gml:ClothoidType	—
GM_CubicSpline	gml:CubicSpline	gml:CubicSplineType	—
GM_GeodesicString	gml:GeodesicString	gml:GeodesicStringType	—
GM_LineString	gml:LineStringSegment	gml:LineStringSegmentType	—
GM_OffsetCurve	gml:OffsetCurve	gml:OffsetCurveType	—
GM_SurfacePatch	gml:AbstractSurfacePatch	gml:AbstractSurfacePatchType	—
GM_GriddedSurface	gml:AbstractGriddedSurface	gml:AbstractGriddedSurfaceType	—
GM_ParametricCurveSurface	gml:AbstractParametricCurveSurface	gml:AbstractParametricCurveSurface Type	—
GM_Cone	gml:Cone	gml:ConeType	—
GM_Cylinder	gml:Cylinder	gml:CylinderType	—
GM_Geodesic	gml:Geodesic	gml:GeodesicType	—
GM_Polygon	gml:PolygonPatch	gml:PolygonPatchType	—
—	gml:Rectangle	gml:RectangleType	—
GM_Sphere	gml:Sphere	gml:SphereType	—
GM_Triangle	gml:Triangle	gml:TriangleType	—
TP_Object	gml:AbstractTopology	gml:AbstractTopologyType	*anonymous property type*
TP_Node	gml:Node	gml:NodeType	gml:DirectedNodePropertyType
TP_Edge	gml:Edge	gml:EdgeType	gml:DirectedEdgePropertyType
TP_Face	gml:Face	gml:FaceType	gml:DirectedFacePropertyType
TP_Solid	gml:TopoSolid	gml:TopoSolidType	gml:DirectedTopoSolidPropertyType
TP_DirectedNode	—	—	gml:DirectedNodePropertyType
TP_DirectedEdge	—	—	gml:DirectedEdgePropertyType
TP_DirectedFace	—	—	gml:DirectedFacePropertyType
TP_DirectedSolid	—	—	gml:DirectedTopoSolidPropertyType
TP_Complex	gml:TopoComplex	gml:TopoComplexType	gml:TopoComplexPropertyType
—	gml:TopoPoint	gml:TopoPointType	gml:TopoPointPropertyType
—	gml:TopoCurve	gml:TopoCurveType	gml:TopoCurvePropertyType
—	gml:TopoSurface	gml:TopoSurfaceType	gml:TopoSurfacePropertyType
—	gml:TopoVolume	gml:TopoVolumeType	gml:TopoVolumePropertyType
TM_Object	gml:AbstractTimeObject	gml:AbstractTimeObjectType	*anonymous property type*
TM_Complex	gml:AbstractTimeComplex	gml:AbstractTimeComplexType	*anonymous property type*
TM_GeometricPrimitive	gml:AbstractTimeGeometricPrimtive	gml:AbstractTimeGeometricPrimtiveT ype	gml:TimeGeometricPrimtiveProperty Type
TM_Instant	gml:TimeInstant	gml:TimeInstantType	gml:TimeInstantPropertyType
TM_Period	gml:TimePeriod	gml:TimePeriodType	gml:TimePeriodPropertyType
TM_TopologicalComplex	gml:TimeTopologyComplex	gml:TimeTopologyComplexType	gml:TimeTopologyComplexPropertyT ype
TM_TopologicalPrimitive	gml:AbstractTimeTopologyPrimtive	gml:AbstractTimeTopologyPrimtiveTy pe	gml:TimeTopologyPrimtivePropertyT ype
TM_Node	gml:TimeNode	gml:TimeNodeType	gml:TimeNodePropertyType
TM_Edge	gml:TimeEdge	gml:TimeEdgeType	gml:TimeEdgePropertyType

TM_PeriodDuration	—	—	gml:duration (property element), xsd:duration

Table D.2 *(continued)*

UML class	GML object element	GML type	GML property type
TM_IntervalLength	—	—	gml:timeInterval (group), gml:TimeIntervalLengthType
TM_Duration	—	—	gml:timeLength (group)
TM_Position	—	—	gml:TimePositionType
TM_IndeterminateValue	—	—	@TimeIndeterminateValue (attribute on TimePositionType)
TM_Coordinate	—	—	xsd:decimal
TM_CalDate	—	—	gml:CalDate
TM_ClockTime	—	—	xsd:time
TM_DateAndTime	—	—	xsd:dateTime
TM_Calendar	gml:TimeCalendar	gml:TimeCalendarType	gml:TimeCalendarPropertyType
TM_CalendarEra	gml:TimeCalendarEra	gml:TimeCalendarEraType	gml:TimeCalendarEraPropertyType
TM_Clock	gml:TimeClock	gml:TimeClockType	gml:TimeClockPropertyType
TM_CoordinateSystem	gml:TimeCoordinateSystem	gml:TimeCoordinateSystemType	*anonymous property type*
TM_OrdinalReferenceSystem	gml:TimeOrdinalReferenceSystem	gml:TimeOrdinalReferenceSystemType	*anonymous property type*
TM_OrdinalEra	gml:TimeOrdinalEra	gml:TimeOrdinalEraType	gml:TimeOrdinalEraPropertyType
SC_CRS	gml:AbstractCRS	gml:AbstractCRSType	gml:CRSPropertyType
SI_LocationInstance	—	—	gml:LocationName
CV_Coverage	gml:AbstractCoverage	gml:AbstractCoverageType	*anonymous property type*
CV_ContinuousCoverage	gml:AbstractContinuousCoverage	gml:AbstractContinuousCoverageType	*anonymous property type*
CV_DiscreteCoverage	gml:AbstractDiscreteCoverage	gml:DiscreteCoverageType	*anonymous property type*
CV_DiscretePointCoverage	gml:MultiPointCoverage	gml:MultiPointCoverageType	*anonymous property type*
CV_DiscreteCurveCoverage	gml:MultiCurveCoverage	gml:MultiCurveCoverageType	*anonymous property type*
CV_DiscreteSurfaceCoverage	gml:MultiSurfaceCoverage	gml:MultiSurfaceCoverageType	*anonymous property type*
CV_DiscreteSolidCoverage	gml:MultiSolidCoverage	gml:MultiSolidCoverageType	*anonymous property type*
CV_DiscreteGridPointCoverage	gml:GridCoverage	gml:GridCoverageType	*anonymous property type*
CharacterString	—	—	xsd:string
Boolean	—	—	xsd:boolean
Real, Number	—	—	xsd:double
Decimal	—	—	xsd:decimal
Date	—	—	xsd:date
Time	—	—	xsd:time
DateTime	—	—	xsd:dateTime
Integer	—	—	**xsd:integer**, xsd:nonPositiveInteger, xsd:negativeInteger, xsd:nonNegativeInteger, xsd:positiveInteger[b]
Vector	—	—	gml:VectorType
GenericName, LocalName or ScopeName	—	—	gml:CodeType
Length, Distance	—	—	gml:LengthType

Table D.2 *(continued)*

UML class	GML object element	GML type	GML property type
Angle	—	—	gml:AngleType
Speed	—	—	gml:SpeedType
Scale	—	—	gml:ScaleType
Area	—	—	gml:AreaType
Volume	—	—	gml:VolumeType
Measure	—	—	gml:MeasureType
Sign	—	—	gml:SignType
UnitOfMeasure	—	—	gml:UnitOfMeasureType

[a] An anonymous type following the pattern for GML property types. The object element referenced or embedded inline is the element in the fourth column in the same row.

[b] Multiple values in the second column are given to support the reverse mapping described in Annex F.

D.2.2 ISO/TS 19103 Conceptual schema language

In this subclause the basic types defined in ISO/TS 19103 that are directly available in GML are specified. In many cases simple types defined by XML Schema are used directly.

— "CharacterString" is implemented by xsd:string. The character encoding is defined in the processing instruction of the XML document (the default for XML documents is UTF-8).

— "Date" is implemented by xsd:date.

— "DateTime" is implemented by xsd:dateTime.

— "Time" is implemented by xsd:time.

— "Real" is implemented by xsd:double.

— "Decimal" is in general implemented by xsd:decimal. For practical reasons, often decimal values will also be represented in schemas by xsd:double.

— The generic basic type "Number" is in general implemented in GML schema by xsd:double.

— "Integer" is implemented by xsd:integer.

— "Boolean" is implemented by xsd:boolean.

— "Measure" is implemented by the simple type gml:MeasureType. The value is of type xsd:double, the uom-specifier is implemented by a URI which will normally resolve to a <gml:UnitDefinition> element or to a well-known unit string. See 8.2.3.6.

  "UnitOfMeasure" is implemented by gml:UnitDefinitionType.

  The following subtypes of "Measure" are implemented by GML, each with a uom attribute that points to a unit definition of a suitable type:

  - “Length” → gml:LengthType
  - “Scale” → gml:ScaleType
  - “Area” → gml:AreaType
  - “Volume” → gml:VolumeType
  - “Speed” → gml:SpeedType
  - “Time” → gml:TimeType
  - “Angle” → gml:AngleType

- “Vector” is implemented by gml:VectorType.

NOTE ISO/TS 19103 describes vector as “an ordered set of numbers called coordinates that represent a position in a coordinate system”. GML uses vector in this sense and provides a capability to explicitly state the coordinate system associated with the vector.

- “GenericName“ and "LocalName“ are implemented by gml:CodeType where the name space designator is a URI.

- "ScopedName“ is implemented by gml:CodeWithAuthorityType where the mandatory name space designator is a URI.

ISO/TS 19103 specifies that all “NULL” values are equivalent. This International Standard uses a more explicit approach by providing a mechanism to specify the reason for the “nil” value. Whether an application uses this added information or not is optional.

D.2.3 ISO 19107 Spatial schema (Geometry)

D.2.3.1 Overview

The UML model of the GML profile defined in this annex describes a conceptual model of the abstract types defined in ISO 19107. The same names for the classes and their properties as in ISO 19107 are used to document the GML profile for ease of comparison with that standard.

NOTE 1 See ISO 19107:2003, Clause 2, for more details.

The additional changes shown in Table D.3 have been applied to the geometry package of ISO 19107.

Table 12 — Description of the profile of ISO 19107 (geometry)

Change	Explanation
GM_Primitive: association “Interior to” deleted	Currently not supported by GML
GM_Polygon: attribute “spanningSurface” deleted	Currently not supported by GML
GM_Solid: converted the operation “boundary()” to an attribute	As the boundary of GM_Solid is accessible only via the “boundary()” operation, an attribute of the same name has been added. The attribute value is the result of the “boundary()” operation as defined in ISO 19107.
GM_Complex: association “Contains” deleted	Currently not supported by GML
Derived attributes deleted in GM_MultiPrimitive subtypes	These attributes may be derived from the digital representation of the objects, therefore the redundant information has been omitted.
GM_CompositePoint: deleted	GM_CompositePoint does not add any additional information. The type has been added in ISO 19107 for completeness only, but it is not expected that it would be used in instance documents. Therefore, it has been omitted in GML.
GM_PolynomialSpline has been made abstract	Currently not instantiable in GML, but the subtype GM_CubicSpline is.
GM_LineSegment: deleted	Not supported by GML, a GM_LineString with two control points shall be used instead.
GM_CurveBoundary: deleted	Only used in operations
GM_ComplexBoundary: deleted	Only used in operations

NOTE 2 GM_OrientableCurve and GM_OrientableSurface are “not abstract” (in accordance with ISO 19107).

D.2.3.2 Geometry root

The UML class diagrams in Figures D.1 and D.2 illustrate the profile of the “Geometry root” package (compare with ISO 19107:2003, Figures 5 and 6).

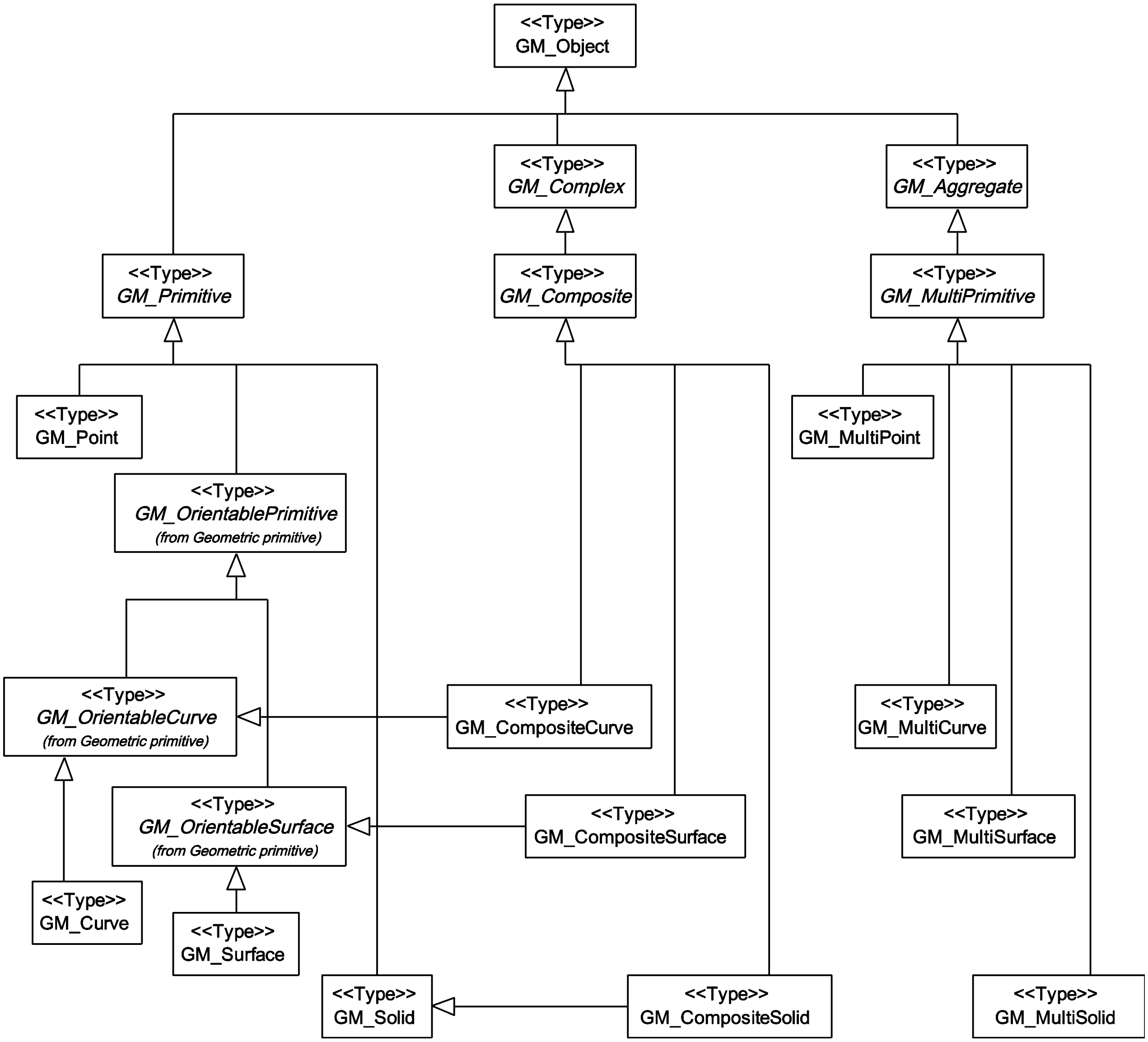


Figure 9 — Implemented subtypes of GM_Object

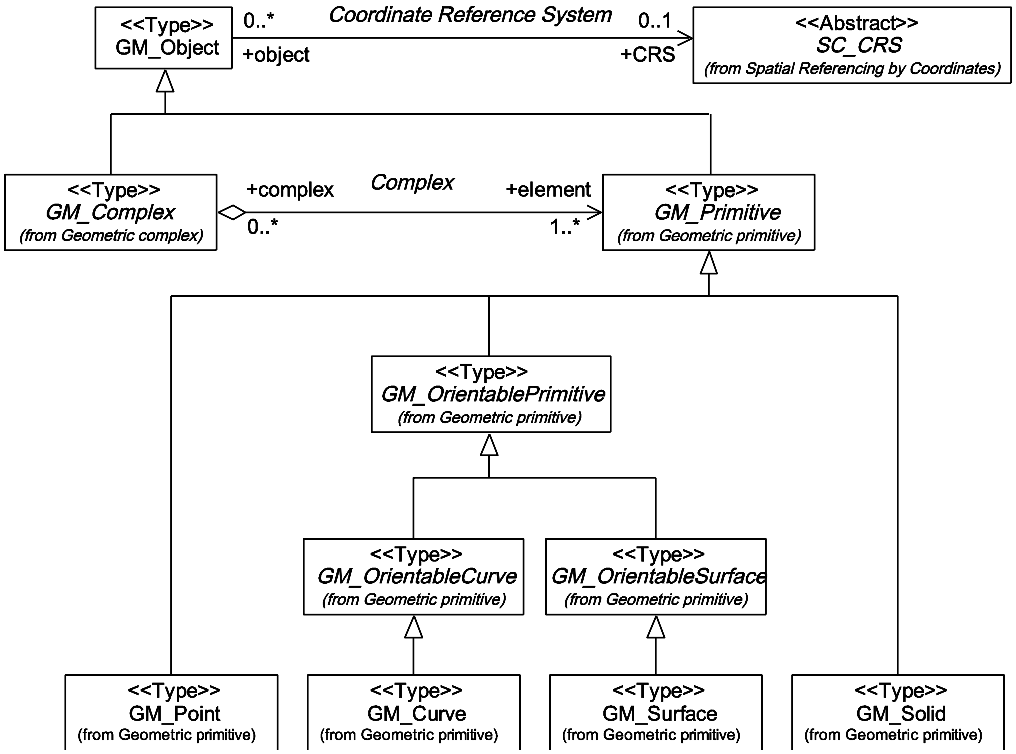


Figure 10 — Geometric primitives

The mapping of the different classes to the GML schema is explained in the subsequent subclauses showing details of the class hierarchy.

“GM_Object” is represented by the “AbstractGeometry” object element, the “CRS” role is represented by the “srsName” property.

The “AbstractGeometry” element may carry additional properties: an optional “description” element, zero or more “name” elements, an optional “identifier” element, and an optional “gml:id” attribute. The latter is particularly useful in supporting the re-use of geometry elements “by reference”, following the compact XPointer syntax.

D.2.3.3 Geometry primitive

The UML class diagrams in Figures D.3 to D. 9 illustrate the profile of the “Geometry primitive” package (compare with ISO 19107:2003, Figures 7 to 13).

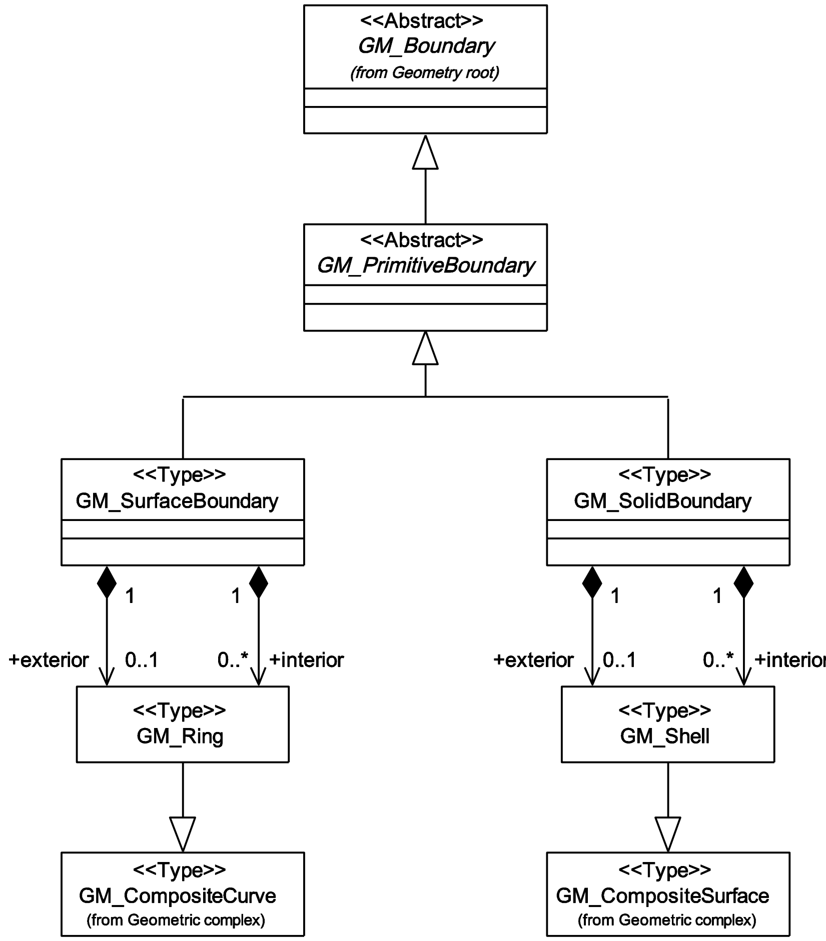


Figure 11 — Boundaries of geometric primitives

The boundary classes from ISO 19107 are not represented explicitly in GML. In ISO 19107 the boundary types are usually the return value of an operation “boundary()”. As the boundary of all surface (patches) or solids needs to be represented in GML explicitly as properties, the "exterior" and "interior" properties have been defined in GML directly as properties of the surface (patch) or solid.

“GM_Ring” is represented by the “Ring” object element. While a “Ring” is not substitutable for a “CompositeCurve” in GML it is structurally identical to a composite curve.

“GM_Shell” is represented by the “Shell” object element. While a “Shell” is not substitutable for a “CompositeSurface” in GML it is structurally identical to a composite surface.

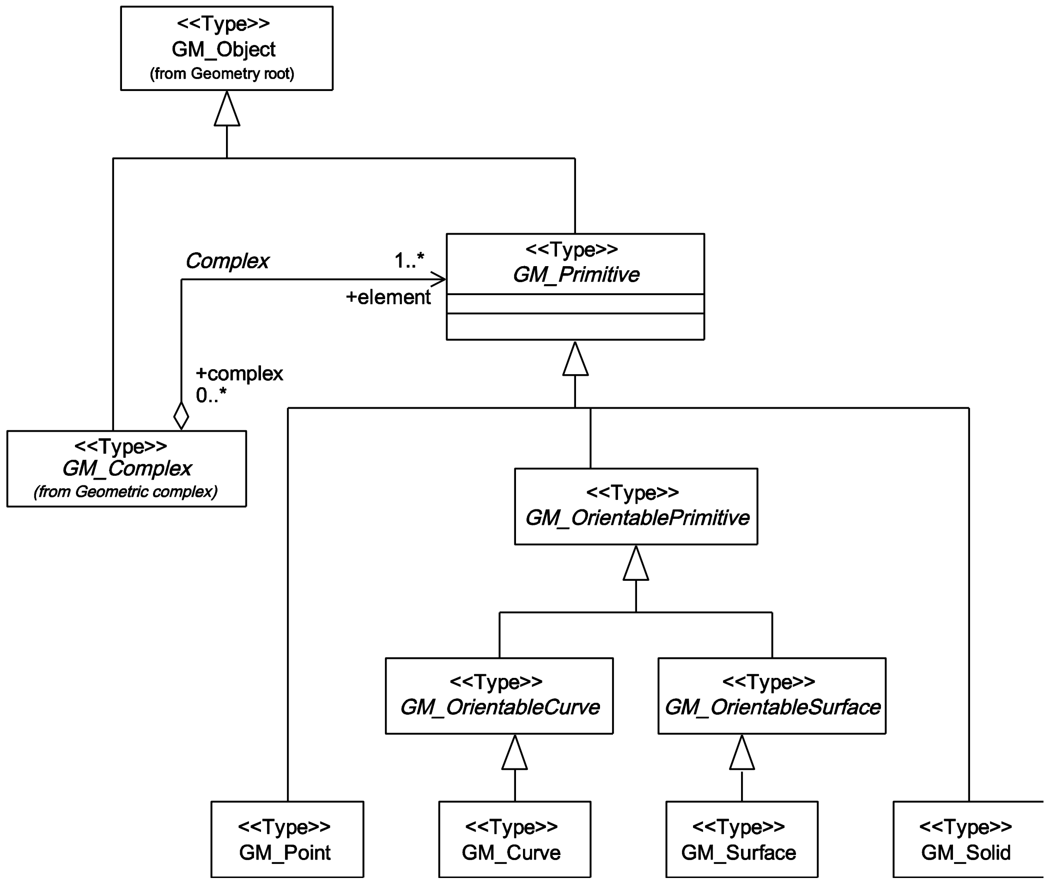


Figure 12 — Geometric primitives

"GM_Primitive" is represented by the "AbstractGeometricPrimitive" object element (both are abstract). The "complex" role is not navigable in GML.

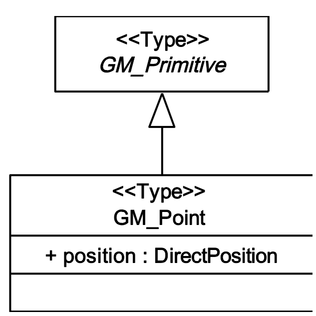


Figure 13 — Point

"GM_Point" is represented by a "Point" object element in GML. The "position" attribute is represented by a "pos" property (the type of the value is "DirectPosition").

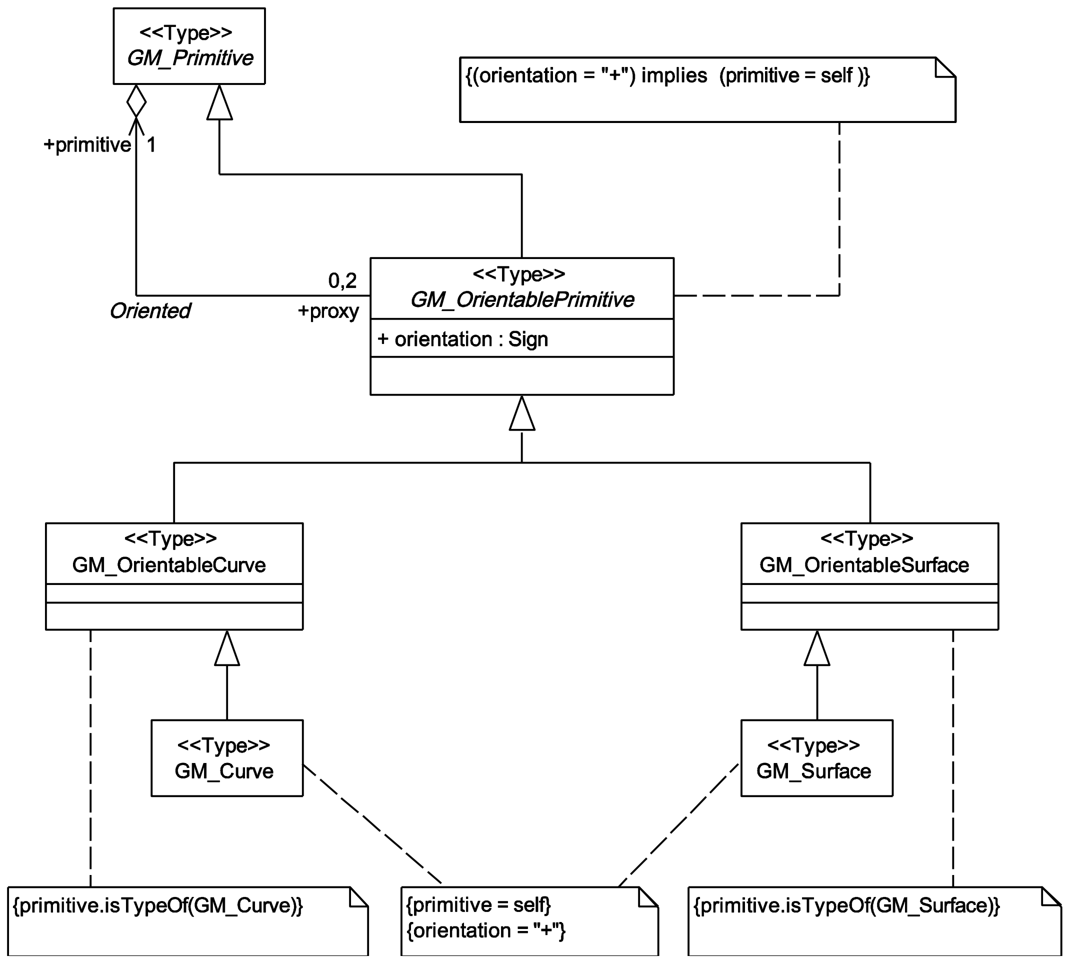


Figure 14 — Curve and Surface

“GM_Curve” is represented by the “Curve” object element in GML. The orientation is not an explicit property of a “Curve” and is implicitly fixed to “+”.

“GM_OrientableCurve” is represented by the “OrientableCurve” object element in GML. The “primitive” role is represented by the “baseCurve” property.

“GM_Surface” is represented by the “Surface” object element in GML. The orientation is not an explicit property of a “Surface” and is implicitly fixed to “+”.

“GM_Orientable Surface” is represented by the “OrientableSurface” object element in GML. The “primitive” role is represented by the “baseSurface” property.

To enable that “CompositeCurve” may be used in GML where in general a geometric primitive is expected, an abstract (and propertyless) object element “AbstractCurve” has been introduced and may be substituted by either “Curve”, “OrientableCurve” or “CompositeCurve”. The same mechanism is used with surfaces and solids. As a result, the “GM_OrientablePrimitive” class is not mapped to GML explicitly, however as this type is not instantiable, this does not impose any restrictions.

NOTE This mapping is a consequence of the fact that the spatial schema uses multiple inheritance to express that a composite geometry, which by definition is a complex geometry, may also represent a geometric primitive. Since XML Schema is not capable of multiple inheritance (or more precisely: derivation from multiple types), the abstract object elements

“AbstractCurve”, “AbstractSurface” and “AbstractSolid” have been introduced in GML to allow that both “true” geometric primitives (e.g. “Curve”) and composite geometries (e.g. “CompositeCurve”) can be in a common substitution group, although both are structurally different.

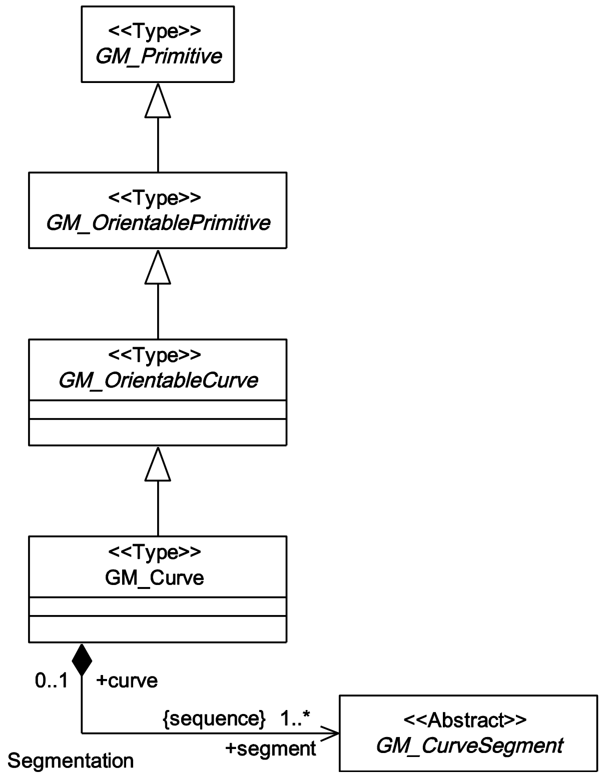


Figure 15 — Curve

As discussed above, “GM_Curve” is represented by the “Curve” object element in GML. The “segment” role is represented as an array property “segments” in GML.

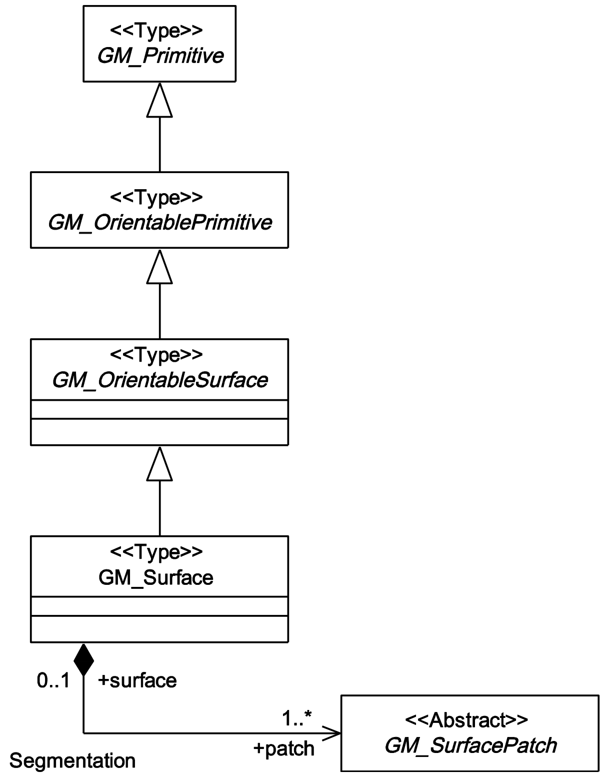


Figure 16 — Surface

As discussed above, “GM_Surface” is represented by a “Surface” object element in GML. The “patch” role is represented as an array property “patches” in GML.

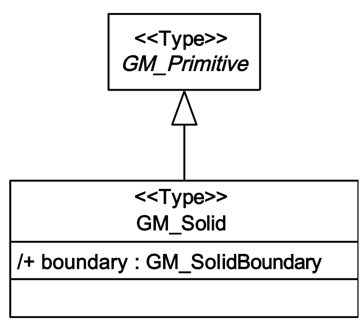


Figure 17 — Solid

“GM_Solid” is represented by a “Solid” object element in GML. The boundary of a Solid is directly expressed by “exterior” and “interior” properties of the solid as discussed above.

D.2.3.4 Coordinate Geometry

The UML class diagrams in Figures D.10 to D.19 illustrate the profile of the "Coordinate Geometry" package (compare with ISO 19107:2003, Figures 14 to 21).

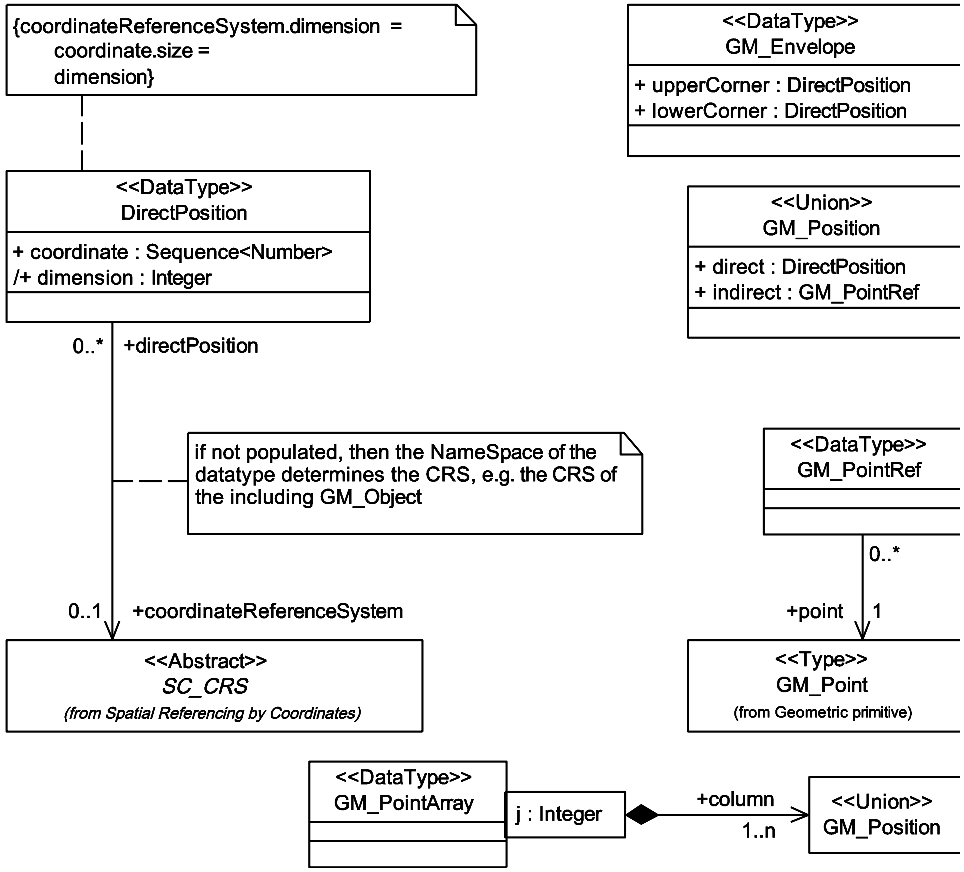


Figure 18 — Coordinate geometry

“DirectPosition” is represented in GML as a type with simple content where the “coordinate” attribute is mapped to a list of doubles. The “coordinateReferenceSystem” role is represented by a “srsName” attribute property and “dimension” is represented by an optional attribute property of the same name (type is positiveInteger).

“GM_Position” is mapped to a choice between a “pos” property (which is of type “DirectPosition”) and a “pointProperty” property (which is “Point”-valued). A “GM_PointArray” is represented as a similar choice element, but with appropriate settings for minimum and maximum occurrences.

A single “GM_Position” or a “GM_PointArray” can alternatively be represented by a “coordinates” property (the type of the value is “Coordinates” which is a type with simple content that represents a list of coordinates encoded as a string).

“GM_Envelope” is represented as the “Envelope” object element in GML. The two attributes “upperCorner” and “lowerCorner” are mapped to properties of the same name. The additional attribute “SRSReferenceGroup” in “gml:Envelope” has been added so that the coordinate reference system need only be specified once in the typical case of corners in the same coordinate reference system.

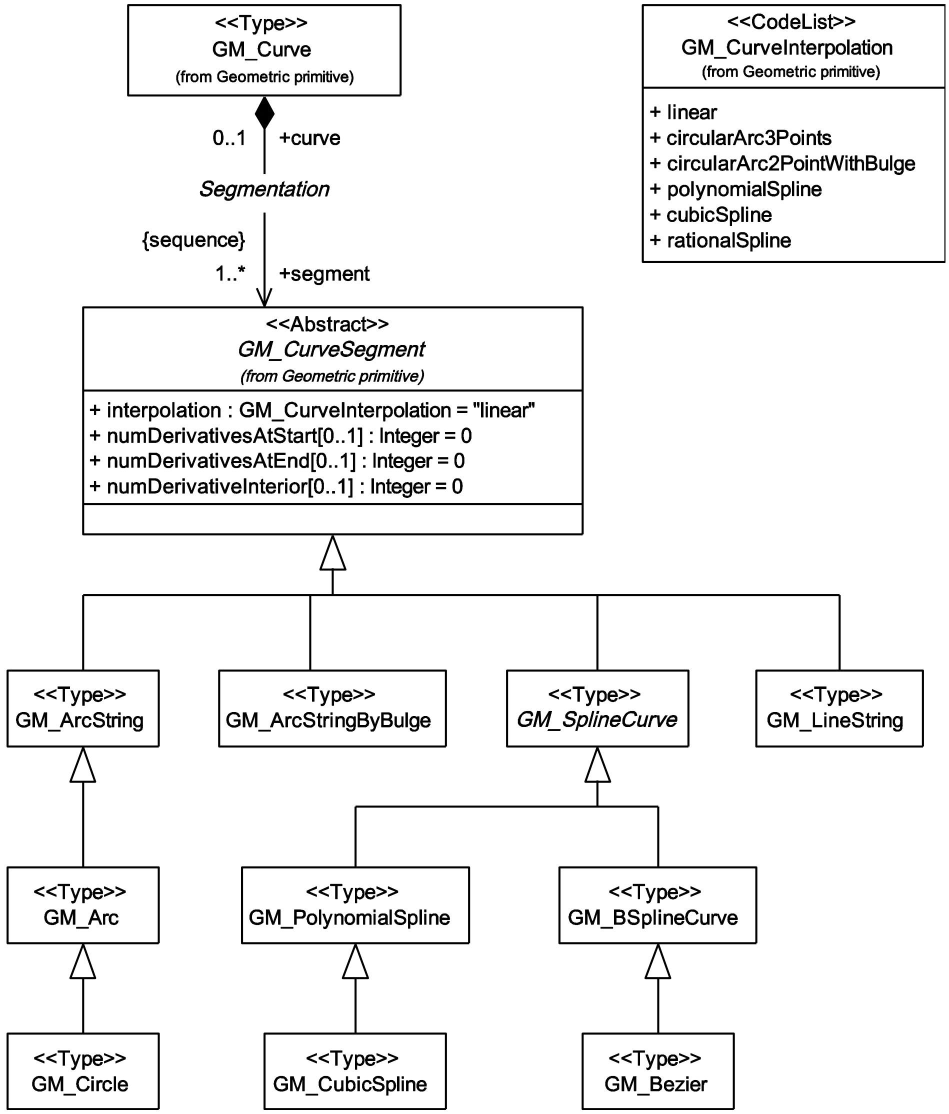


Figure 19 — Curve segments

"GM_CurveSegment" is represented in GML by the "AbstractCurveSegment" object element (both are abstract). The three "numDerivatives…" attributes are mapped to properties with the same definition. The "interpolation" attribute is not defined in "AbstractCurveSegment", but is defined (and set with appropriate initial values) in the instantiable subtypes.

GML currently supports a subset of all defined curve segments of ISO 19107.

Most subtypes of "GM_CurveSegment" carry a "controlPoint" attribute that is represented in GML by the choice element as described above (see discussion of the representation of a GM_PointArray).

The code list "GM_CurveInterpolation" has been mapped to GML as if it would be an enumeration, i.e. no additional values are allowed beside the predefined values in the GML schema.

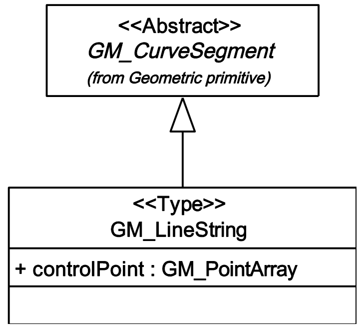


Figure 20 — Line string

"GM_LineString" is represented by the "LineStringSegment" object element. The "Segment" suffix is appended to the name in GML, because the name "LineString" is already reserved for another object element in GML (see D.3.5). To maintain backwards compatibility with previous versions of GML it was not possible to change the name of the existing element as even if the previous use of "LineString" would have been deprecated, the name would not have been available for the implementation of GM_LineString.

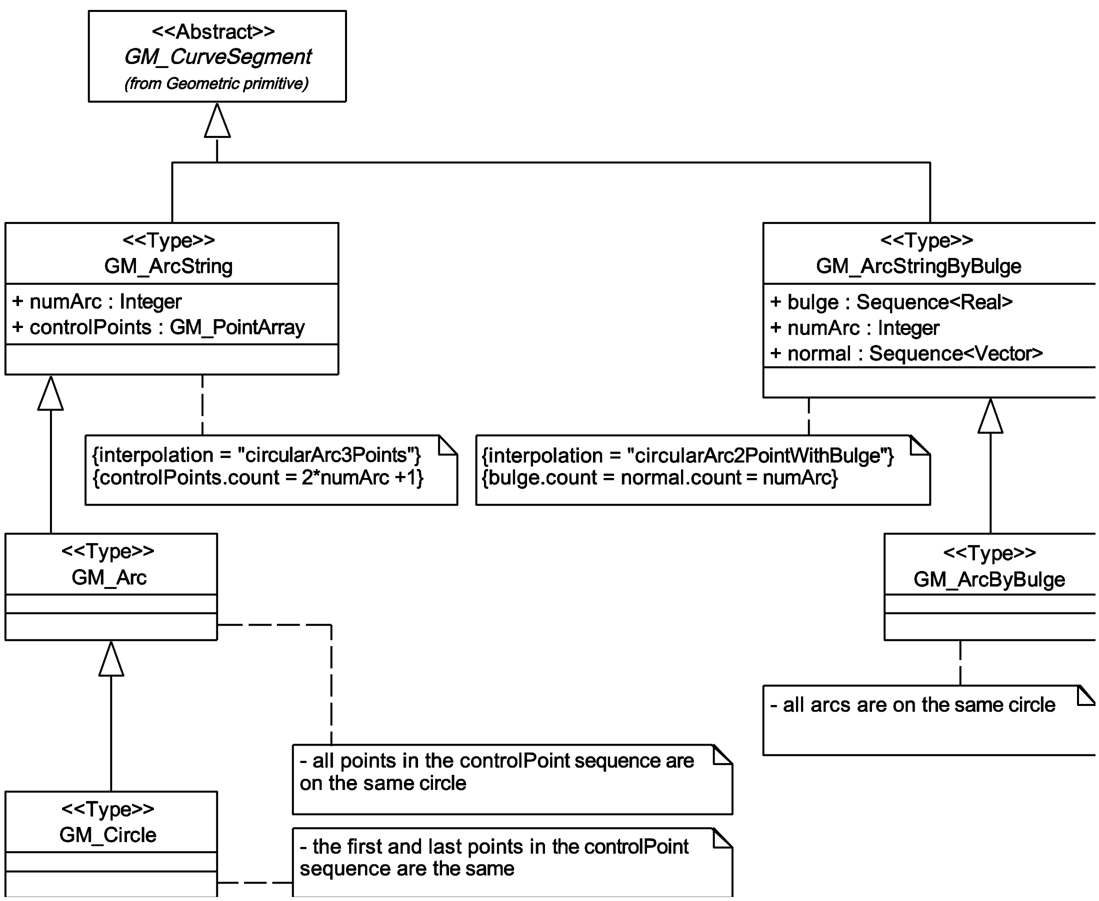


Figure 21 — Arcs and circles

The curve segment types are mapped to object elements in GML with the same name (but without the “GM_” prefix) and the same set of properties.

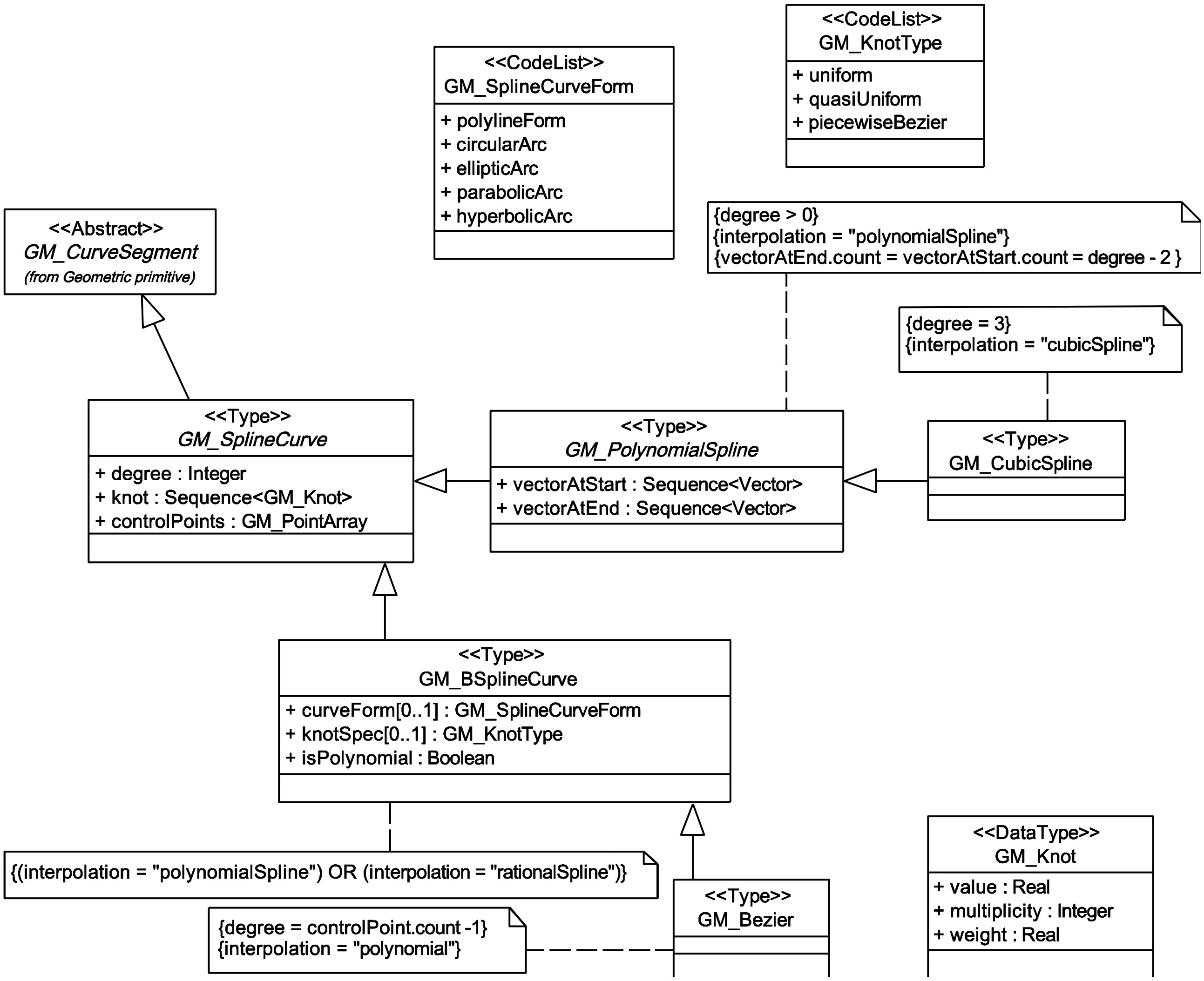


Figure 22 — Splines

Again, these curve segment types are mapped to object elements in GML with the same name (but without the “GM_” prefix[8]) and with the same properties[9]). The properties of the curve segment objects in GML have been specified taking the OCL constraints into account.

The code list “GM_KnotType” has been mapped to GML as if it would be an enumeration, i.e. no additional values are allowed beside the predefined values in the GML schema.

8) However, “GM_BSplineCurve” is represented by “BSpline”, i.e. without the “Curve” suffix. The name "BSpline" has been kept to maintain backwards compatibility with previous versions of GML.

9) The “knotSpec” attribute has been renamed to “knotType” in GML.

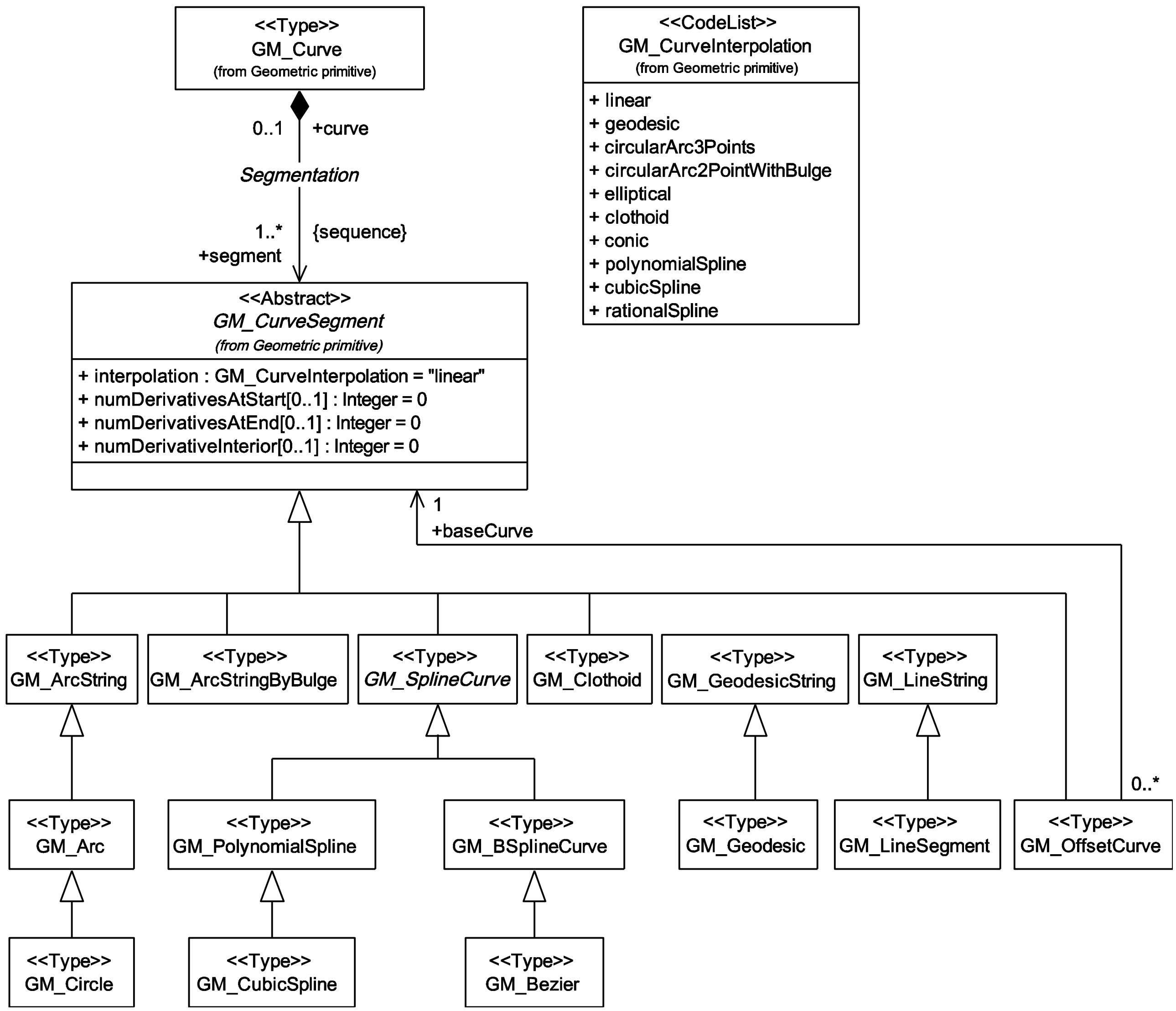


Figure 23 — Curve segments

“GM_OffsetCurve” class is represented in GML by the “OffsetCurve” object element. The object carries the same semantic interpretation as the class. The baseCurve property has been renamed to offsetBase.

“GM_AffinePlacement” is represented in GML by the “AffinePlacement” object element.

“GM_GeodesicString” is represented in GML by the “GeodesicString” object element.

“GM_Geodesic” is represented in GML by the “Geodesic” object element.

“GM_Clothoid” is represented in GML by the “Clothoid” object element.

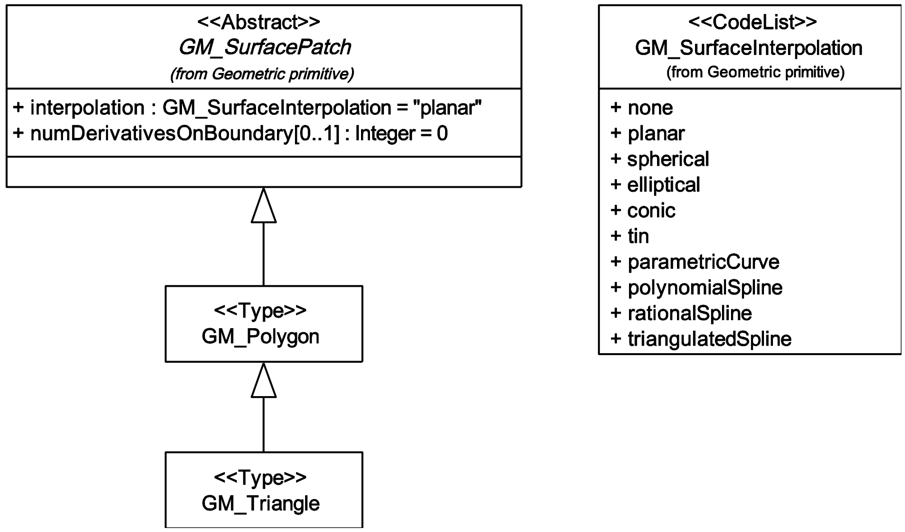


Figure 24 — Surface patches (first figure)

“GM_SurfacePatch” is represented in GML by the “AbstractSurfacePatch” object element (both are abstract). The “numDerivativesOnBoundary” attribute is currently not explicitly mapped in GML as only planar interpolation is currently supported in GML. The “interpolation” attribute is not defined in “AbstractSurfacePatch”, but it is defined (and set with appropriate initial values) in the instantiable subtypes.

GML currently supports a subset of all defined surface types and surface patch types of ISO 19107.

The code list “GM_SurfaceInterpolation” has been mapped to GML as if it would be an enumeration, i.e. no additional values are allowed beside the predefined values in the GML schema.

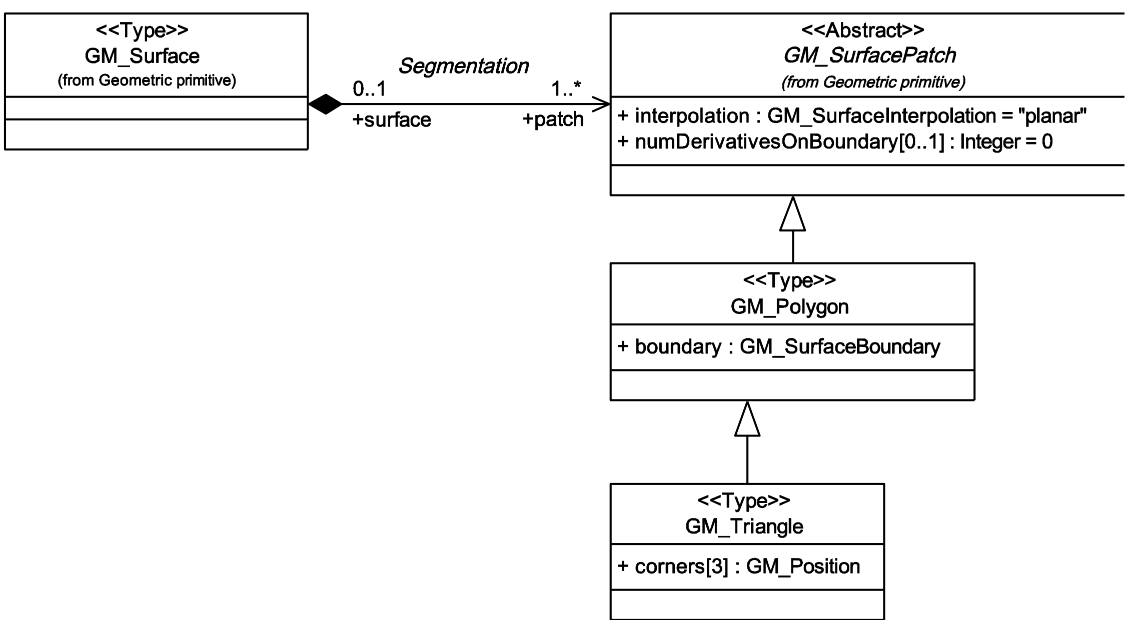


Figure 25 — Surface patches (second figure)

“GM_Polygon” is represented by the “PolygonPatch” object element. The “boundary” attribute is directly expressed by “exterior” and “interior” properties of the “PolygonPatch”.

The “Patch” suffix has been appended to the name in GML, because the name “Polygon” is already reserved for another object element in GML (see D.3.6). To maintain backwards compatibility with previous versions of GML it was not possible to change the name of the existing element as even if the previous use of "Polygon" would have been deprecated, the name would not have been available for the implementation of GM_Polygon.

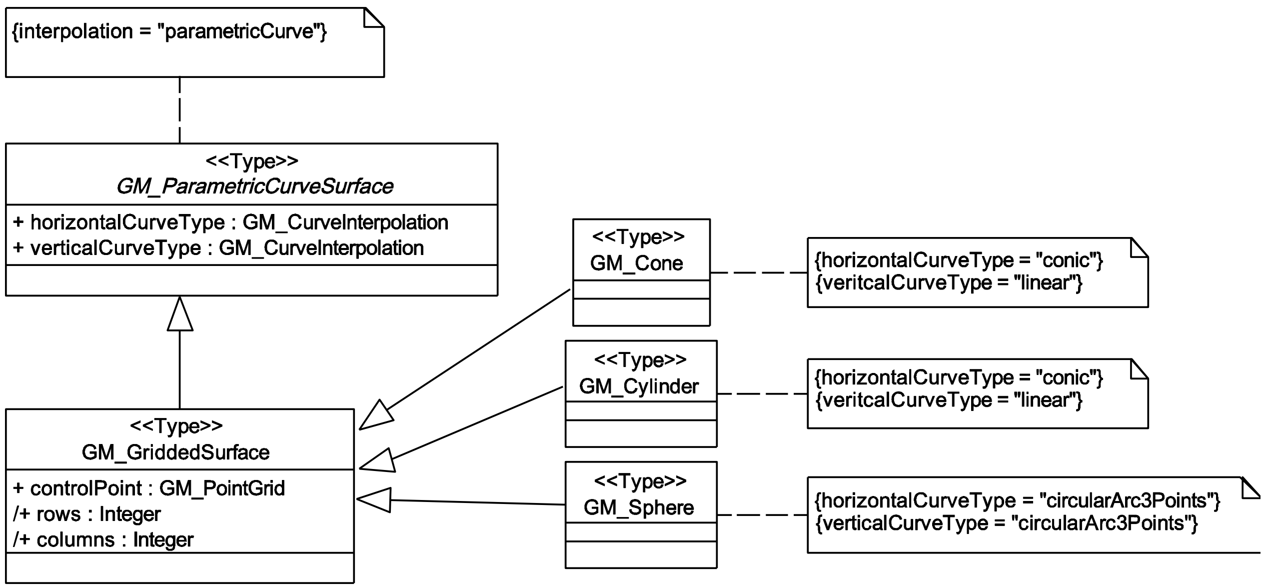


Figure 26 — Gridded surface patches

“GM_PointGrid” is represented in GML by the “PointGrid” group.

“GM_ParametricCurveSurface” is represented in GML by the “AbstractParametricCurveSurface” object element (both are abstract).

“GM_GriddedSurface” is represented in GML by the “AbstractGriddedSurface” object element (both are abstract).

“GM_Cone” is represented in GML by the “Cone” object element.

“GM_Cylinder” is represented in GML by the “Cylinder” object element.

“GM_Sphere” is represented in GML by the “Sphere” object element.

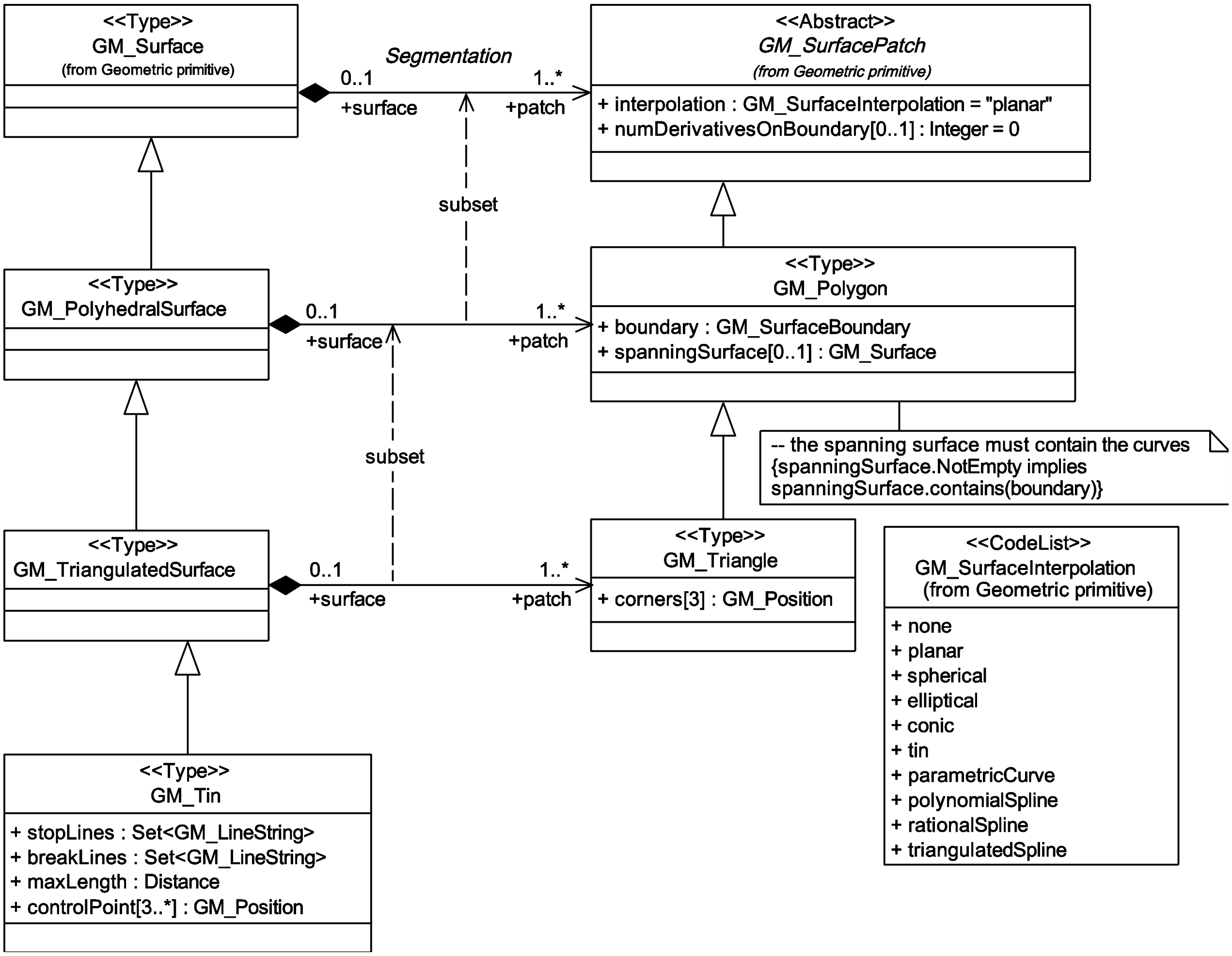


Figure 27 — Polyhedral and triangulated surfaces

“GM_PolyhedralSurface” is represented in GML by the “PolyhedralSurface” object element.

“GM_TriangulatedSurface” is represented in GML by the “TriangulatedSurface” object element.

“GM_Tin” is represented in GML by the “Tin” object element.

D.2.3.5 Geometry aggregates

The UML class diagram in Figure D.20 illustrates the profile of the “Geometry aggregates” package (compare with ISO 19107:2003, Figure 24).

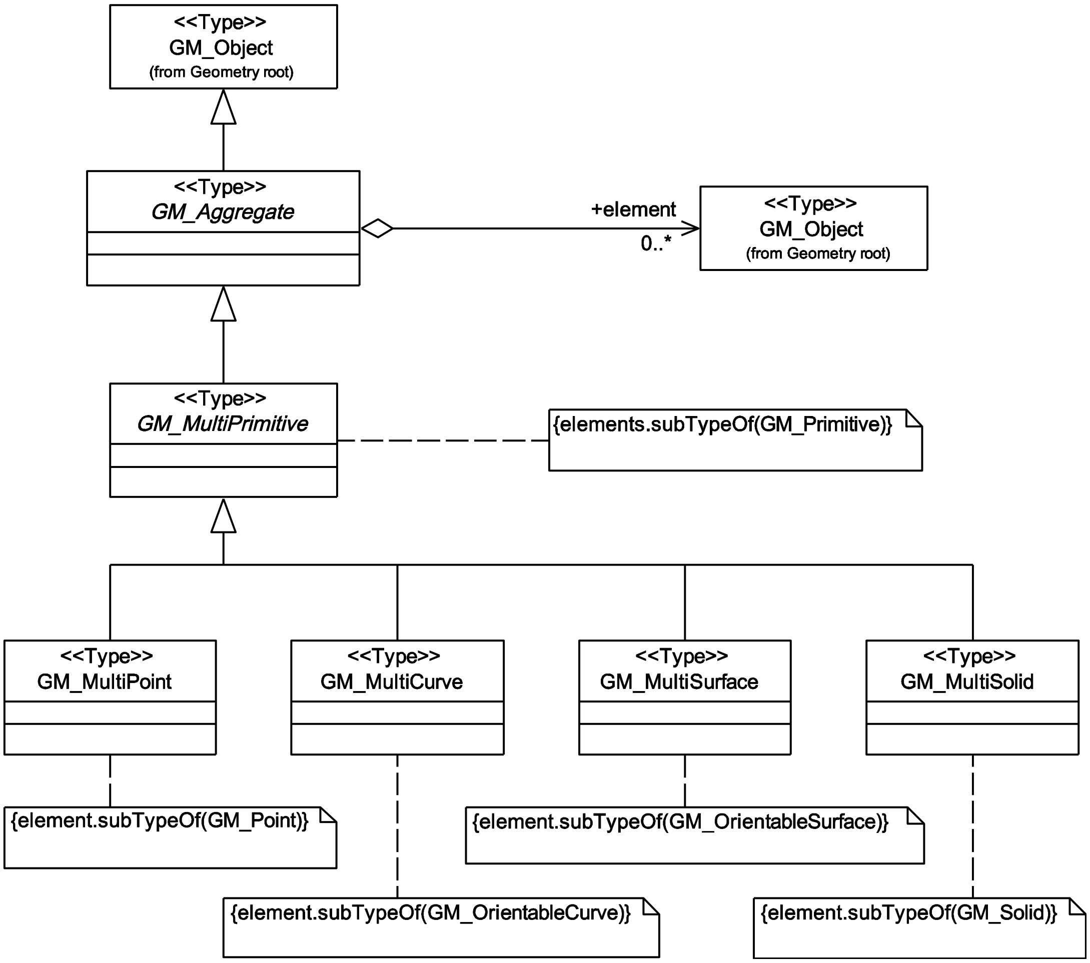


Figure 28 — Geometric aggregates

"GM_Aggregate" is represented by the "AbstractGeometricAggregate" object element (both are abstract). The "element" role is instantiated in GML in the instantiable subtypes. The general pattern is that two properties are defined, one is a regular association property and the other an array association property. The property names are "*x*Member" and "*x*Members" respectively where the "*x*" is replaced by "point", "curve", "surface" or "solid" depending on the elements of the collection. This represents the OCL-constraints for type safety.

"GM_MultiPoint" is represented by the "MultiPoint" object element in GML.

"GM_MultiCurve" is represented by the "MultiCurve" object element in GML.

"GM_MultiSurface" is represented by the "MultiSurface" object element in GML.

"GM_MultiSolid" is represented by the "MultiSolid" object element in GML.

"GM_MultiPrimitive" is not explicitly represented in GML.

D.2.3.6 Geometry complex

The UML class diagrams in Figures D.21 and D.22 illustrate the profile of the "Geometry complex" package (compare with ISO 19107:2003, Figures 25 to 30).

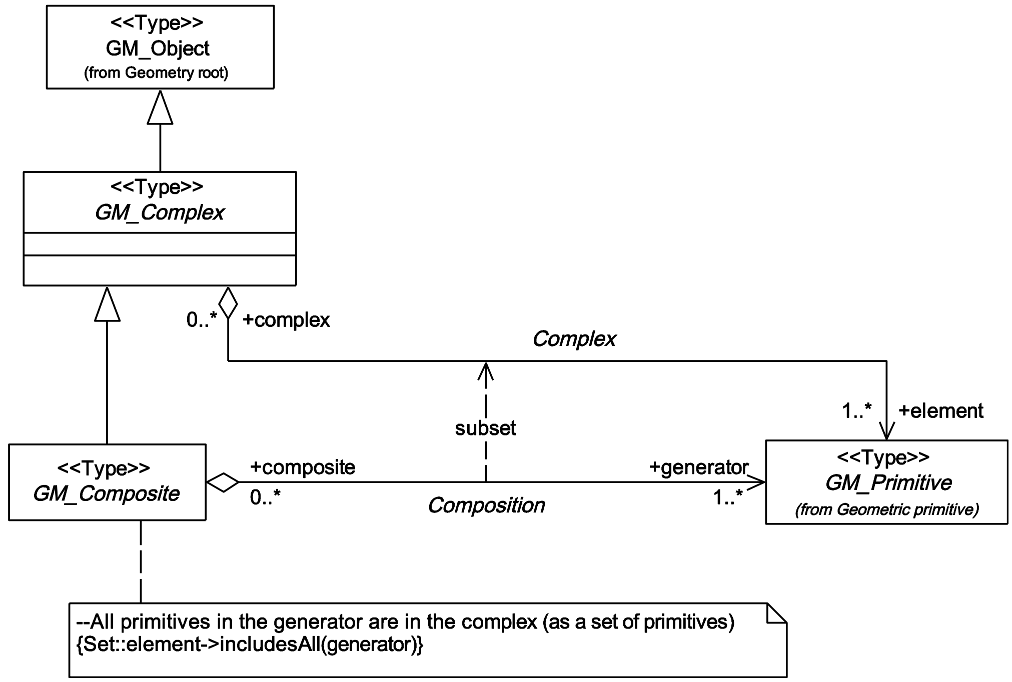


Figure 29 — Geometric composites (first figure)

“GM_Complex” is represented by the “GeometricComplex” object element. The “element” role is mapped to a property of the same name in GML.

The fact that the composite geometry types are subtypes of “GM_Complex” is represented in GML by the fact that every association property which takes a “GeometricComplex” accepts also one of the composites (due to a choice element in “GeometricComplexPropertyType”). This special mapping to XML Schema was necessary, because multiple inheritance, used in ISO 19107 to express the “dualism” of the composite geometries, is not supported by the derivation mechanism of XML Schema.

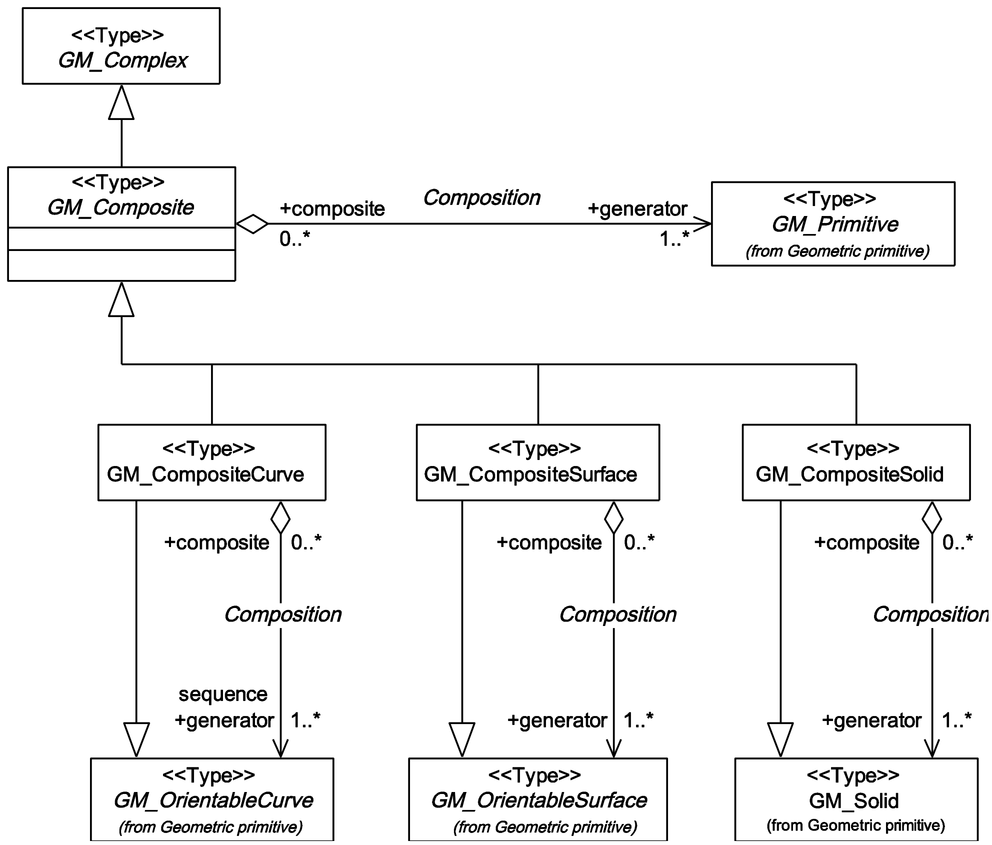


Figure 30 — Geometric composites (second figure)

"GM_Composite" is not explicitly represented by an object element in GML. However, the subtypes "GM_CompositeCurve", "GM_CompositeSurface" and "GM_CompositeSolid" are represented in GML by object elements of the same name (without the "GM_" prefix). The "generator" role is instantiated in GML in these subtypes by an association property with the name "*x*Member" where the "*x*" is replaced by "curve", "surface" or "solid" depending on the elements of the collection.

D.2.3.7 Conformance

The rules governing conformance of a profile of ISO 19107 are described in ISO 19107:2003, Clause 2 and Annex A. Concerning the three criteria defined in Clause 2, GML geometry covers the following levels:

Data Complexity:

— geometric primitives;

— geometric complexes.

Dimensionality:

— 0-, 1-, 2- and 3-dimensional objects.

Functional Complexity:

— data types only.

Thus, the relevant conformance clauses of ISO 19107 are:

— A.1.1.1 to A.1.1.4;

— A.2.1.1 to A.2.1.3.

The conditions of these conformance clauses are met.

Note that derived attributes are treated as operations and it is assumed that the derived attributes in the aggregate geometries will be derived from the data by the application handling the GML instances.

A GM_CompositePoint is represented by a “Point” object element in GML. The value of the “generator” association role is the same object, i.e. the “Point” object itself.

D.2.4 ISO 19107 spatial schema (topology)

D.2.4.1 Overview

The additional changes shown in Table D.4 have been applied to the topology package of ISO 19107.

Table 13 — Description of the profile of ISO 19107 (topology)

Change	Explanation
TP_Complex: association “isMaximal()” added as a derived attribute	The information was otherwise not accessible by means of predefined data structures of TP_Complex. The attribute is defined as a derived attribute representing the result of the “isMaximal()” operation as defined in ISO 19107.
TP_Object has been changed from an interface class to type class (however without any properties) and the Realization relationships from TP_Primitive and TP_Complex to TP_Object has been changed to Generalization relationships	Maintaining TP_Object as a root for the different topological subtypes makes mapping to GML clearer.
The optional association “Realization” between TP_Complex and GM_Complex has been deleted.	The realization can be derived from the realization of the primitives contained in the topology complex.
The “maximalComplex” role has been deleted from TP_Primitive.	Currently not supported in GML
TP_Boundary and subtypes as well as TP_Ring and TP_Shell have been deleted.	Only used in operations

D.2.4.2 Topology root

The UML class diagrams in Figures D.23 and D.24 illustrate the profile of the “Topology root” package (compare with ISO 19107:2003, Figures 32 and 33).

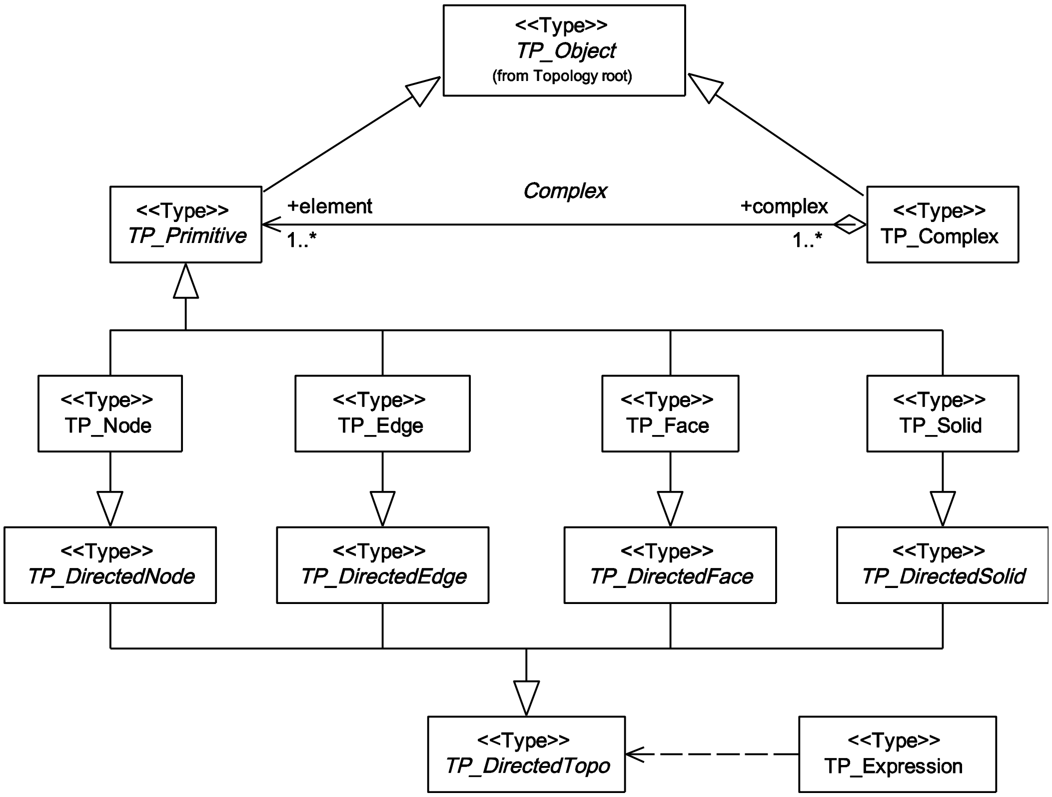


Figure 31 — Topologic primitives

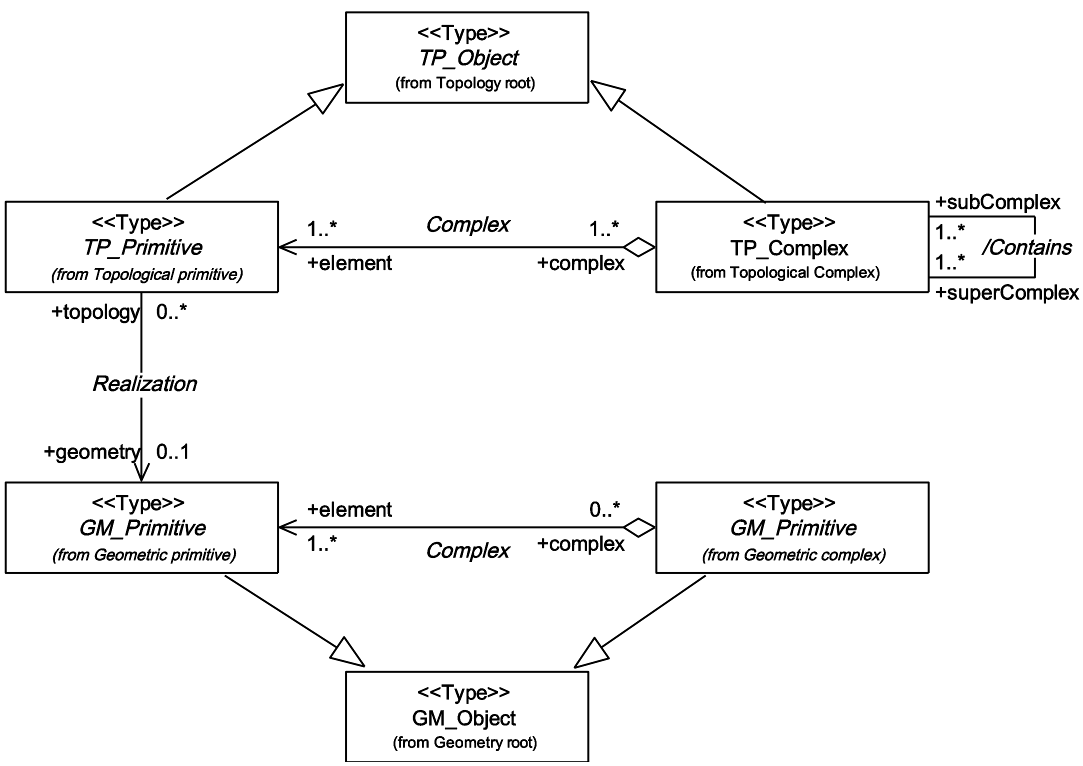


Figure 32 — Relationship between geometry and topology

The mapping of the different classes to the GML schema is explained in the subsequent subclauses showing details of the class hierarchy.

“TP_Object” is represented by the “AbstractTopology” object element. The “AbstractTopology” element may carry additional properties: an optional “description” element, zero or more “name” elements, an optional “identifier” element, and an optional “`gml:id`” attribute.

D.2.4.3 Topology primitive

The UML class diagrams in Figures D.25 to D.28 illustrate the profile of the “Topology root” package (compare with ISO 19107:2003, Figures 35 to 45).

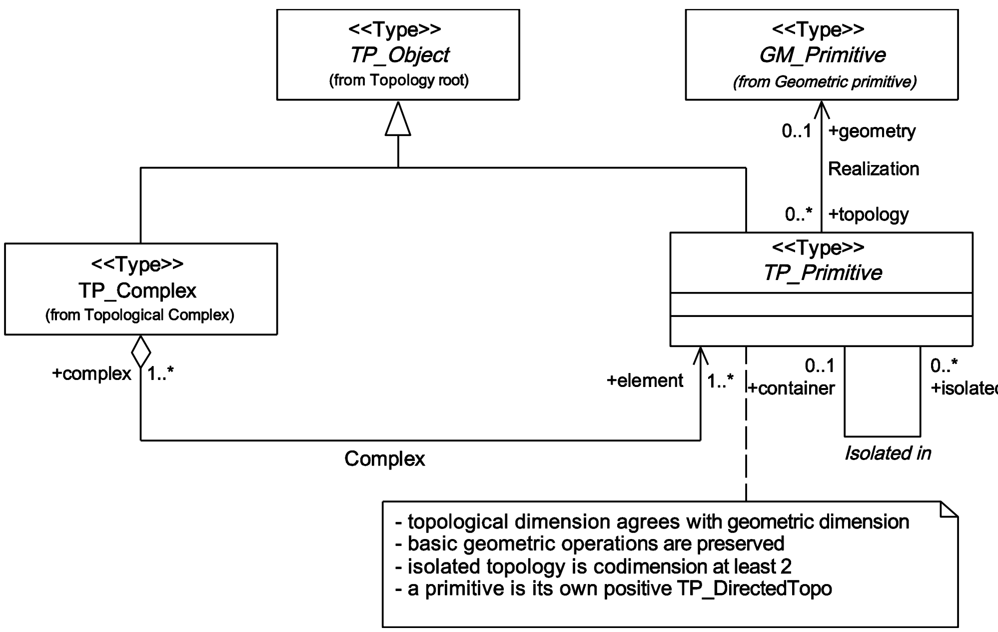


Figure 33 — Topology primitive

“TP_Primitive” is represented by the “AbstractTopoPrimitive” object element.

The “geometry” role is instantiated in the instantiable subtypes. This allows control of the geometry types at the other association end (dimensionality constraint): “pointProperty”, “curveProperty”, “surfaceProperty” and “solidProperty” respectively.

The “isolated” and “container” roles are represented as properties in “AbstractTopoPrimitive”.

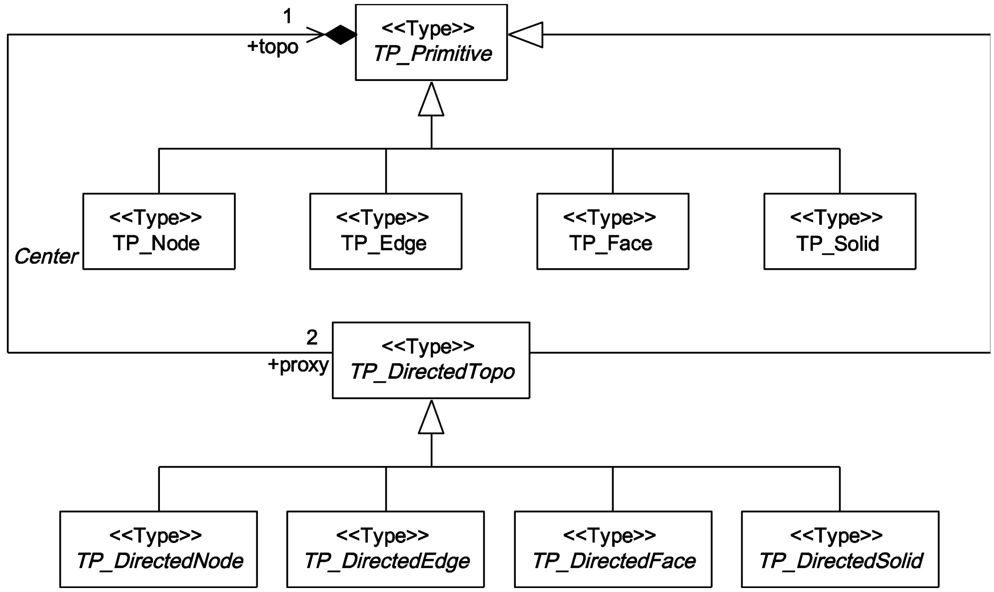


Figure 34 — Topology primitives

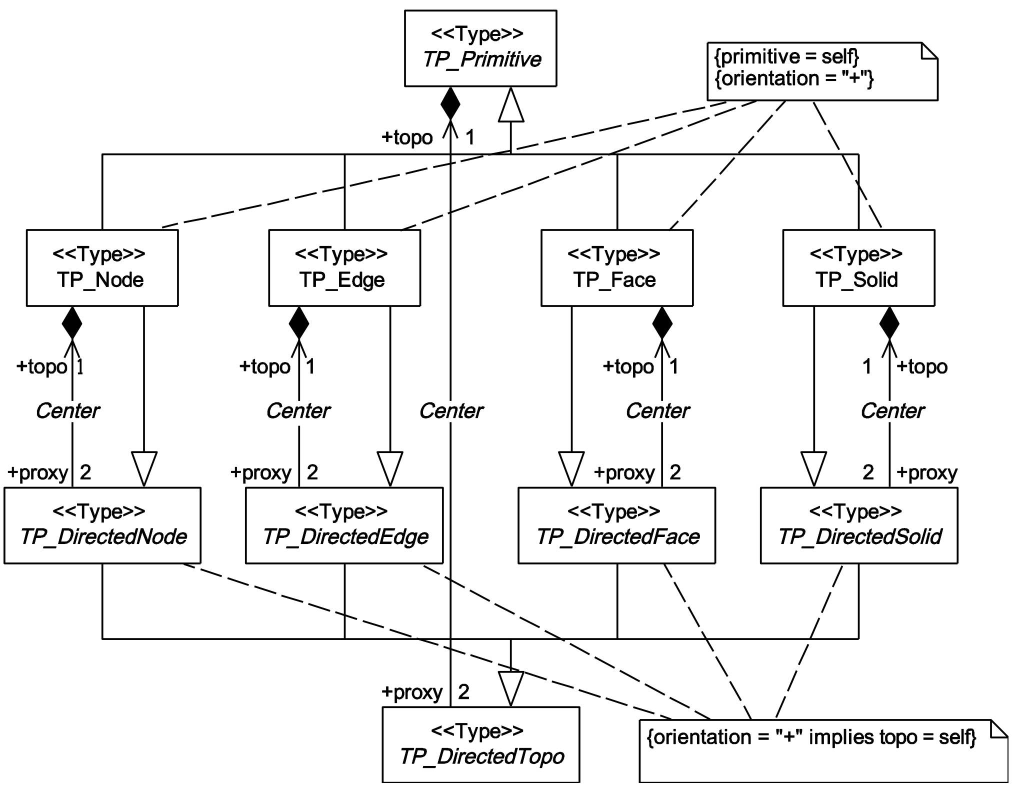


Figure 35 — Directed Topology primitives

“TP_Node” is represented by the “Node” object element in GML.

“TP_Edge” is represented by the “Edge” object element in GML.

“TP_Face” is represented by the “Face” object element in GML.

“TP_Solid” is represented by the “TopoSolid” object element in GML (the name “Solid” is already used for the 3-dimensional geometry primitive).

“TP_DirectedTopo” is not explicitly represented in GML, only its instantiable subtypes. A notable difference is that although the directed topology types are modelled as types they are represented as properties with an “orientation” attribute in GML.

“TP_DirectedNode” is represented by the “directedNode” property element in GML. The “topo” role is represented directly by the “Node” object element.

“TP_DirectedEdge” is represented by the “directedEdge” property element in GML. The “topo” role is represented directly by the “Edge” object element.

“TP_DirectedFace” is represented by the “directedFace” property element in GML. The “topo” role is represented directly by the “Face” object element.

“TP_DirectedSolid” is represented by the “directedTopoSolid” property element in GML. The “topo” role is represented directly by the “TopoSolid” object element.

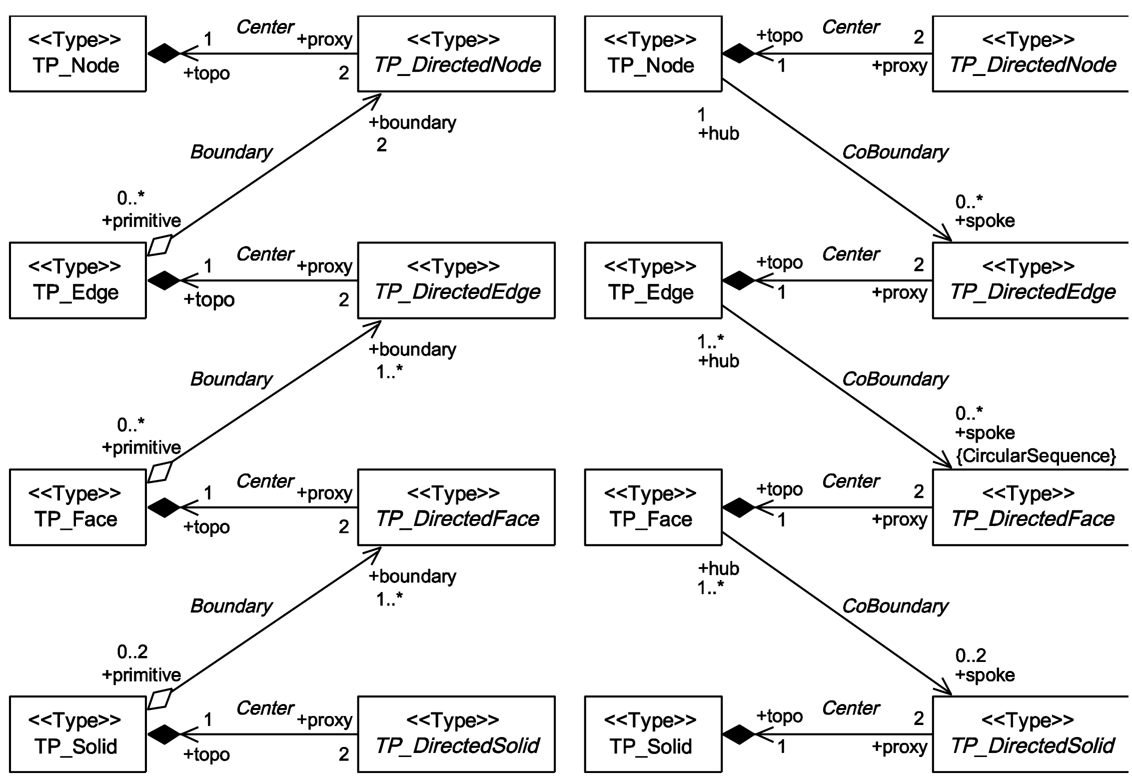


Figure 36 — Boundary and co-boundary relationships

The mapping of the “topo” role is already discussed above.

The “spoke” role is represented in GML by “directed Edge”, “directedFace” and “directedTopoSolid” properties respectively.

The “boundary” role is represented in GML by “directed Node”, “directedEdge” and “directedFace” properties respectively.

D.2.4.4 Topology complex

The UML class diagram in Figure D.29 illustrates the profile of the “Topology complex” package (compare ISO 19107:2003, Figure 46).

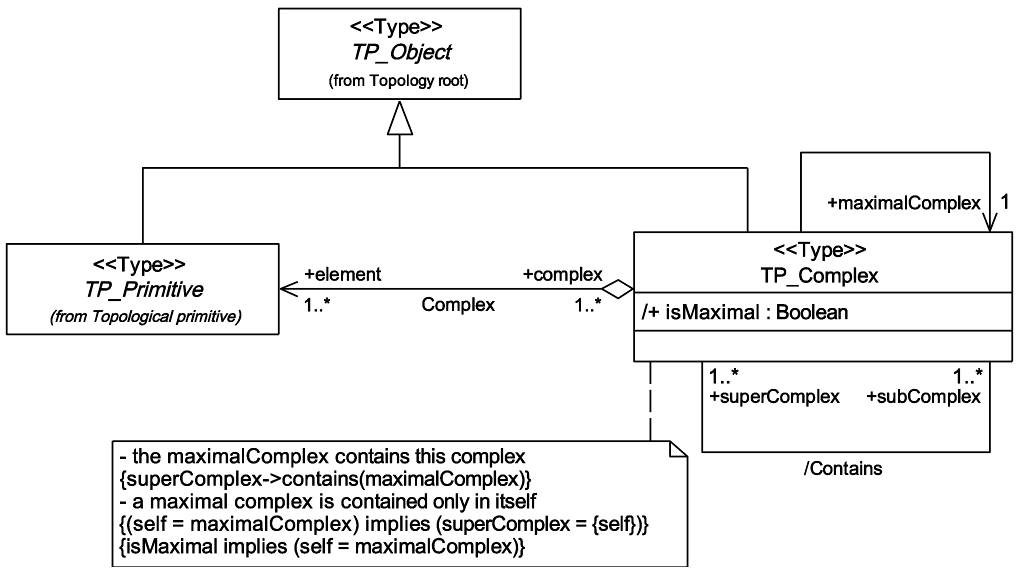


Figure 37 — Topology complex

“TP_Complex” is represented by the “TopoComplex” object element.

The “element” role is mapped to two properties — one regular association property “topoPrimitiveMember” and one array association property “topoPrimitiveMembers”.

The “subComplex” and “superComplex” roles are represented as association properties of the same name. The minimum multiplicity, however, is “0” for both properties in GML instead of “1”. This reflects that it is not required that this property is represented explicitly in a GML instance (note that it is a derived association).

The “maximalComplex” role is represented as an association property of the same name in GML.

D.2.4.5 Conformance

The rules governing conformance of a profile with ISO 19107 are described in ISO 19107:2003, Clause 2 and Annex A. Concerning the three criteria defined in Clause 2, GML topology covers the following levels:

Data Complexity:

— topological complexes;

— topological complexes with geometric realizations.

Dimensionality:

— 0-, 1-, 2- and 3-dimensional objects.

Functional Complexity:

— data types only.

Thus, the relevant conformance clauses of ISO 19107 are:

— A.3.1.1 to A.3.1.3;

— A.4.1.1 to A.4.1.3.

The conditions of these conformance clauses are met.

Note that the association “Realization” between TP_Complex and GM_Complex is not an explicit part of the profile, because the geometrical realization of the topology complex can be derived from the geometrical realization of the topological primitives.

D.2.5 ISO 19108 Temporal schema

D.2.5.1 Overview

The GML temporal schemas provides an implementation of ISO 19108:2002, 5.2 to 5.4.

The changes shown in Table D.5 have been applied to the packages of ISO 19108.

Table 14 — Description of the profile of ISO 19108

	Change	Explanation
1	The *Beginning* and *Ending* associations are implemented as unidirectional from TM_Period to TM_Instant	Maintaining bi-directional pointers deemed unnecessary in a static data encoding.
2	The *Beginning* and *Ending* associations from TM_Period are implemented as a choice of (a UML association with TM_Instant or a UML attribute of type TM_Position)	Allows a more compact encoding of the same information when a TM_Instant object with identity is not required.
3	The *Termination* and *Initiation* associations are implemented as unidirectional from TM_Edge to TM_Node	Maintaining bi-directional pointers deemed unnecessary in a static data encoding.
4	The *Realization* associations are implemented as unidirectional from TM_Instant and TM_Period to TM_Node and TM_Edge respectively	Maintaining bi-directional pointers deemed unnecessary in a static data encoding.
5	The *Basis* association is implemented as unidirectional from TM_Calendar to TM_CalendarEra	Maintaining bi-directional pointers deemed unnecessary in a static data encoding.
7	The *begin* and *end* attributes of TM_OrdinalEra are replaced by associations with TM_Node, with rolenames *start* and *end*	Practice in historical and geological sciences is that the termination points of an Ordinal Era are associated with events whose position may not be known precisely. This matches with the concept of TM_Node whose position is available indirectly.
8	The *origin* attribute of TM_CoordinateSystem is implemented as choice of (a UML association with TM_Instant or a UML attribute of type TM_Position)	Allows the origin to be specified in terms of an external event.

9	The *interval* attribute of TM_CoordinateSystem is implemented as TM_Interval Length	Allows the scale to be specified more precisely and flexibly.

NOTE Change 2 takes advantage of the <choice> structure which is provided by the XML Schema implementation language. This supports a more flexible and compact encoding, containing the same information, than would have been gained by mechanical application of the standard encoding rules.

Of the classes dealing with temporal relationships between features, described in 5.5 of ISO 19108:2002, only Feature Succession has been implemented directly. Components corresponding to the other relationships may be defined in GML application schemas, but are not discussed further here.

The mapping of the different classes to the GML schema is explained in the subsequent subclauses showing details of the class hierarchy.

The UML class diagrams in Figures D.30 to D.34 illustrate the profile of the "Temporal Objects" package (compare with ISO 19108:2002, Figures 2 to 6 and 11).

D.2.5.2 Temporal objects

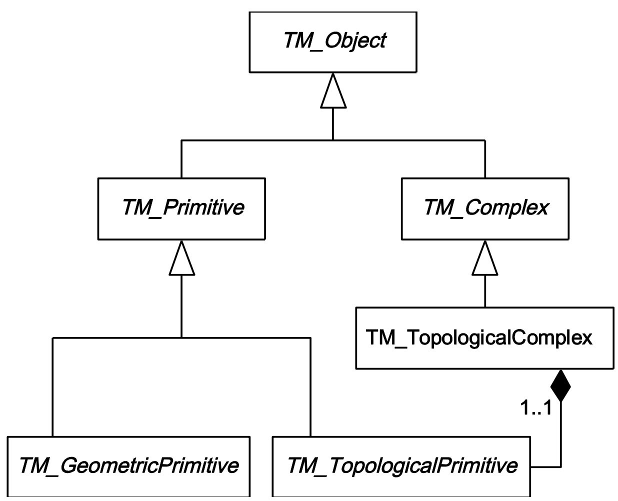


Figure 38 — Main hierarchy of temporal objects from ISO 19108

"TM_Object" is represented by the "AbstractTimeObject" object element. The "AbstractTimeObject" element may carry additional properties: an optional "description" element, zero or more "name" elements, an optional "identifier" element, and an optional "`gml:id`" attribute. These properties are inherited by all the components that are substitutable for AbstractTimeObject.

"TM_Primitive" is represented by the "AbstractTimePrimitive" object element.

"TM_GeometricPrimitive" is represented by the "AbstractTimeGeometricPrimitive" object element.

"TM_TopologicalPrimitive" is represented by the "AbstractTimeTopologyPrimitive" object element.

"TM_Complex" is represented by the "AbstractTimeComplex" object element.

"TM_TopologicalComplex" is represented by the "AbstractTimeTopologyComplex" object element.

D.2.5.3 Concrete temporal geometric primitives

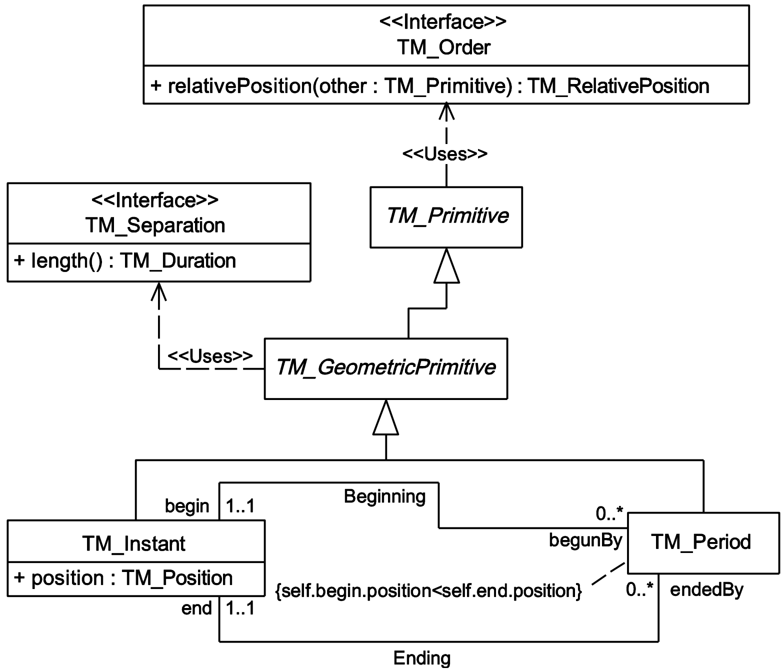


Figure 39 — Profile of temporal geometric objects adapted from ISO 19108

“TM_Primitive” is represented by the “AbstractTimePrimitive” object element. Additional properties “relatedTime” representing the result of “relativePosition(other:TM_Primitive)” operations has been added.

“TM_GeometricPrimitive” is represented by the “AbstractTimeGeometricPrimitive” object element. An additional property “abstractTimeLength” representing the result of the “length()” operation has been added.

“TM_Instant” is represented by the “TimeInstant” object element. The “position” attribute is represented by the “timePosition” property.

“TM_Period” is represented by the “TimePeriod” object element. The “begin” and “end” roles are represented by association properties of the same name in GML. These associations have an alternative representation in GML as follows: “end” is in a choice block with “endPosition” and “begin” with “beginPosition”, the latter in each case has simple content as discussed in 14.2.2.5.

D.2.5.4 Temporal duration

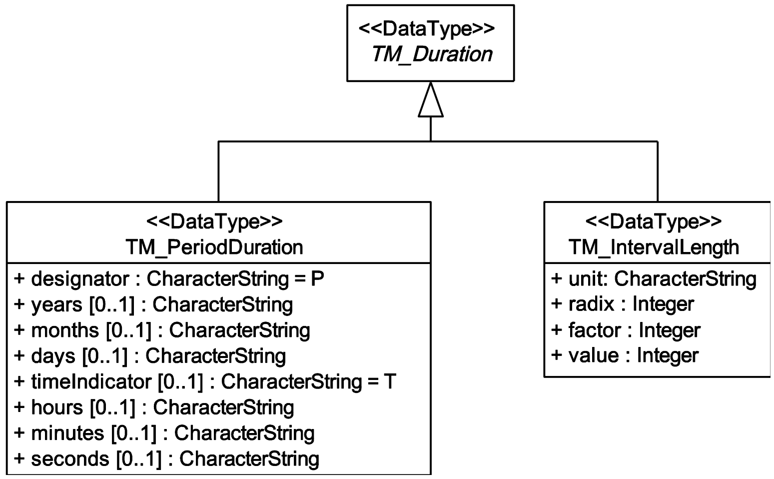


Figure 40 — DataTypes representing temporal duration ISO 19108

The GML property element “abstractTimeLength” is abstract, with either a “timeInterval or “duration” element substituting. These have XML Schema types which implement the data types shown in Figure D.32, as follows:

“TM_IntervalLength” is implemented using a simple content type constructed by adding the XML attributes “unit”, “radix” and “factor” to the XML Schema built-in type “decimal”.

“TM_PeriodDuration” is implemented by the XML Schema built-in type “duration” (see discussion in 14.2.2.8). The XML Schema type “duration” prescribes a literal value with the lexical form described in ISO 8601, which removes the need to implement the list of attributes of the TM_PeriodDuration class separately.

D.2.5.5 Temporal position

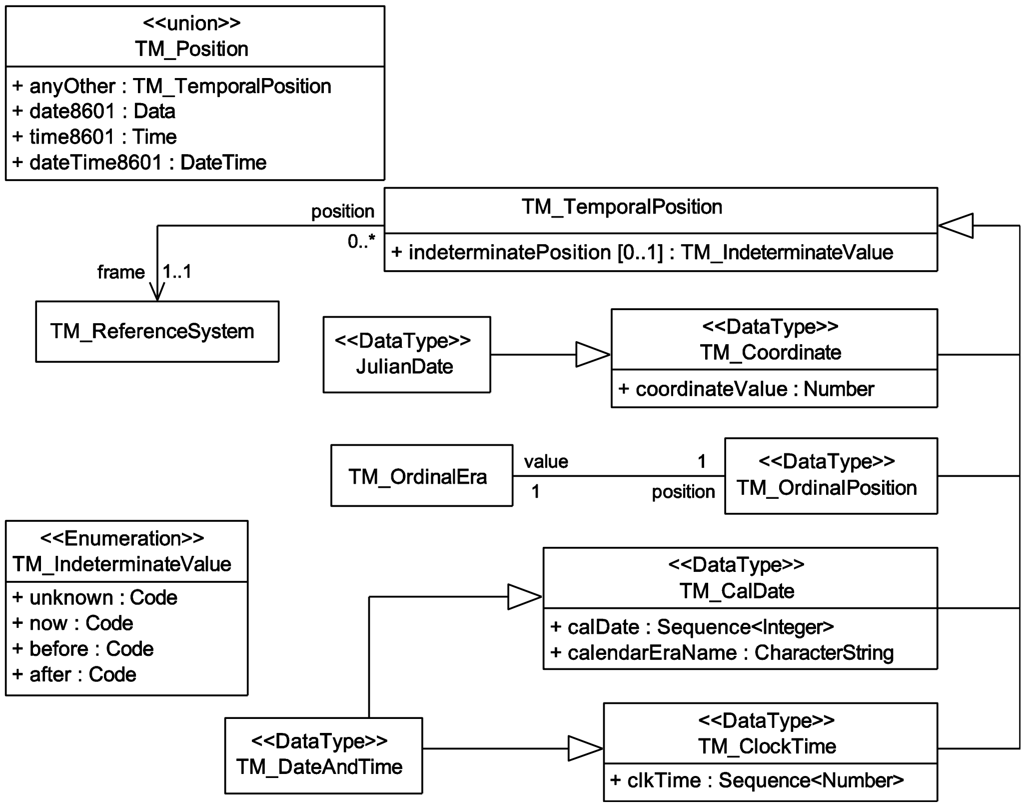


Figure 41 — Temporal position, from ISO 19108

Components represented as UML attributes of type TM_Position are represented as GML properties with XML Schema type `gml:TimePositionType`. This is a simple content type the details of whose derivation are described in 14.2.2.5. This represents the requirements shown in Figure D.33 as follows:

“TM_Coordinate”, which gives temporal position represented by a single number, is implemented by XML Schema type “decimal”;

“TM_OrdinalPosition”, which carries an association with a TM_OrdinalEra, is implemented by XML Schema type “anyURI”, which follows the pattern used in GML where associations are implemented through references;

“TM_CalDate”, which carries attributes consisting of sequence of numbers for the calendar date, and an era name, is implemented in `gml:CalDate` by a union (choice) of XML Schema types “date”, “gYear”, “gYearMonth”, whose lexical representations follow ISO 8601, to which an XML attribute “calendarEraName” is added;

“TM_ClockTime”, which carries a sequence of numbers describing an instant that recurs daily, is implemented by XML Schema type “time”, whose lexical representation follows ISO 8601;

“TM_DateAndTime” is implemented by XML Schema type “dateTime”, whose lexical representation follows ISO 8601;

The variants “date8601”, “time8601” and “dateTime8601”, shown in TM_Position, are implemented by the XML Schema types “date”, “time” and “dateTime”, already introduced;

“IndeterminatePosition” is represented using an XML attribute of the same name;

The role “frame” is implemented using an XML attribute of the same name, whose value has type “anyURI”, which follows the pattern used in GML where associations are implemented through references.

D.2.5.6 Temporal topology

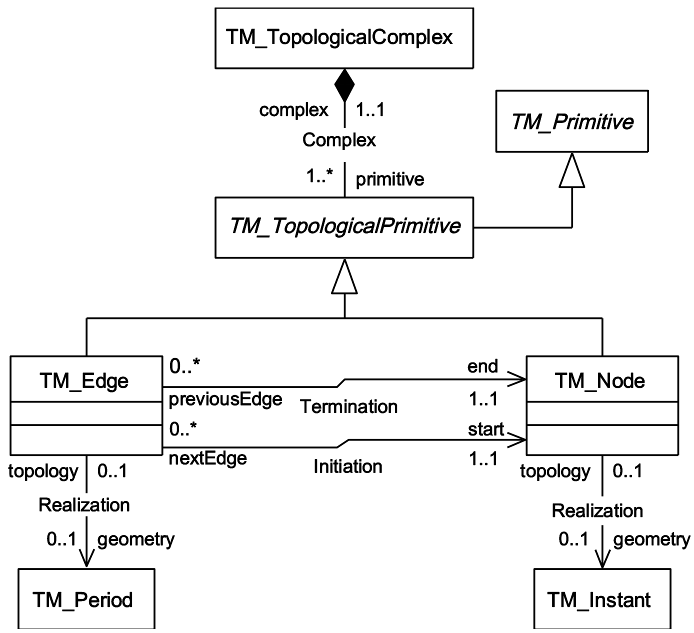


Figure 42 — Profile of temporal topology adapted from ISO 19108

“TM_TopologicalComplex” is represented by the “AbstractTimeTopologyComplex” object element. The “primitive” role is implemented by a property element of the same name.

“TM_TopologicalPrimitive” is represented by the “AbstractTimeTopologyPrimitive” object element. The “complex” role is implemented as a reference by a property element of the same name, though this is made optional.

“TM_Node” is implemented by the “TimeNode” object element. The “previousEdge” and “nextEdge” roles are implemented by property elements of the same name in GML. The “geometry” role is implemented by the “position” property.

“TM_Edge” is implemented by the “TimeEdge” object element. The “start” and “end” roles are implemented by property elements of the same name in GML. The “geometry” role is implemented by the “extent” property.

D.2.5.7 Temporal reference systems

The UML class diagrams in Figures D.35 to D.38 illustrate the profile of the “Temporal Reference Systems” package (compare with ISO 19108:2002, Figures 7 to 10).

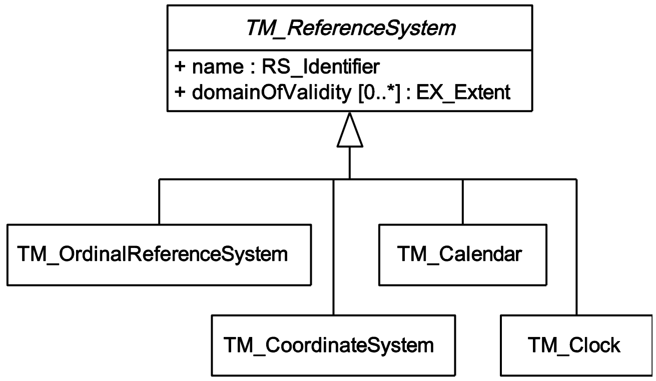


Figure 43 — Hierarchy of temporal reference systems from ISO 19108

“TM_ReferenceSystem” is implemented by the “AbstractTimeReferenceSystem” object element. The “AbstractTimeReferenceSystem” element may carry additional properties: an optional “description” element, one or more “name” elements, an optional “identifier” element, and a mandatory "`gml:id`" attribute. The “domainOfValidity” attribute is implemented using an XML attribute of the same name. This has XML Schema type “string” which implements the “description” attribute of EX_Extent (see ISO 19115).

These properties are inherited by all the components that are substitutable for AbstractTimeReferenceSystem.

D.2.5.8 Calendars and clocks

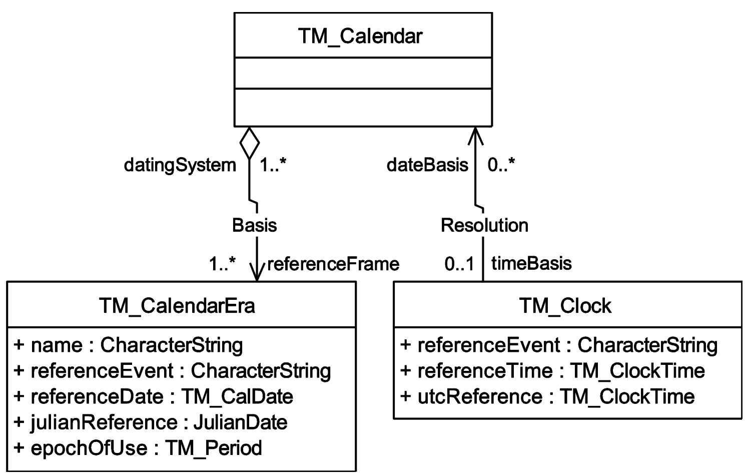


Figure 44 — Model for calendar and clock, from ISO 19108

“TM_Calendar” is implemented by the “TimeCalendar” object element. The “referenceFrame” role is implemented as a property element of the same name.

“TM_CalendarEra” is implemented by the “TimeCalendarEra” object element. The “referenceEvent”, “referenceDate”, “julianReference” and “epochOfUse” attributes are implemented as property elements of the same names.

“TM_Clock” is implemented by the “TimeClock” object element. The “referenceEvent”, “referenceTime”, and “utcReference” attributes are implemented as property elements of the same names. The “dateBasis” role is implemented as a property element of the same name.

D.2.5.9 Time coordinate systems

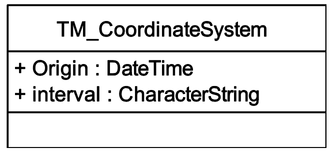


Figure 45 — Model for temporal coordinate system, from ISO 19108

“TM_CoordinateSystem” is implemented by the “TimeCoordinateSystem” object element. The “origin” attribute is implemented as a choice of property elements “origin”, which refers to a TimeInstant, or “originPosition” which encodes a position directly. The “interval” attribute is implemented as a property element of the same name, using TimeIntervalLengthType which follows ISO/IEC 11404.

D.2.5.10 Temporal ordinal reference system

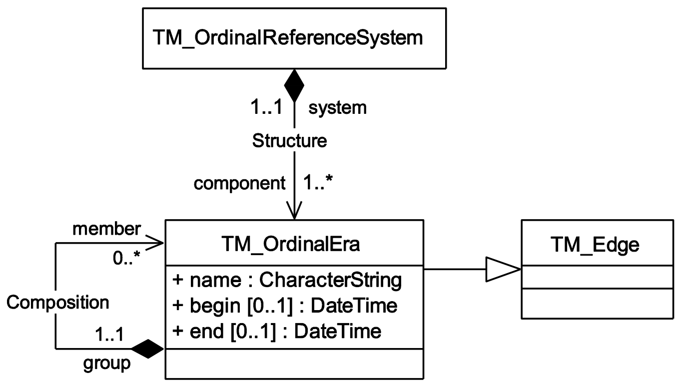


Figure 46 — Model for temporal ordinal reference system, adapted from ISO 19108

“TM_OrdinalReferenceSystem” is implemented by the “TimeOrdinalReferenceSystem” object element. The “component” role is implemented as a property element of the same name.

“TM_OrdinalEra” is implemented by the “TimeOrdinalEra” object element. The “name”, “begin” and “end” attributes are implemented by the “name”, “start” and “end” properties inherited from TimeEdge. The “member” role is implemented as property element of the same name. The “group” role is implemented as a reference by a property element of the same name. An optional “description” and a mandatory "`gml:id`" property are also inherited from TimeEdge.

D.2.5.11 Conformance

The rules governing conformance of a profile with ISO 19108 are described in ISO 19108:2002, Clause 2 and Annex A. Concerning the criteria defined in Clause 2, GML as an application schema for data transfer targets conformance with A.1. The conditions of this conformance clause are met by the profile specified above.

D.2.6 ISO 19109 rules for application schema

GML implements a subset of the general feature model defined in ISO 19109.

In addition extensions are implemented by GML. The general feature model is concerned only with feature types whereas an application schema (in GML or UML) will often deal with additional information types. Examples are data types, enumeration types, union types, etc. which are not specified explicitly in the general feature model.

ISO 19109 therefore specifies that only a one-way mapping from the general feature model to the application schema is possible.

Like in the case of UML, the mapping from the general feature model to the GML feature model described in XML Schema is in general straightforward.

The changes shown in Table D.6 have been applied to the general feature model of ISO 19109.

Table 15 — Description of the implementation of ISO 19109

	Change	Explanation
1	GF_FeatureOperation deleted	Operation are not supported
2	Multiplicity of GF_InheritanceRelation/supertype changed to “1”	Only single inheritance is supported and mapped to the substitution group mechanism of XML Schema

Some additional comments on the metaclasses of the general feature model are shown in Table D.7.

Table 16 — Remarks on the implementation of ISO 19109

	Change	Explanation
1	GF_AssociationType	In general, the composition of association roles to associations is not directly represented in GML application schemas. If the relationship between two association roles shall be expressed explicitly in the GML application schema, then the roles may be cross-referenced by - representing the qualified element name of the target object class in an appInfo annotation element `gml:targetElement` which is of type `xsd:string` - representing the qualified property name of the inverse association roles in an appInfo annotation element `gml:reverseProperty` which is of type `xsd:string` - optionally representing the name of the association in an appInfo annotation element `gml:associationName` which is of type `xsd:string`
2	GF_Constraint	Constraints may be mapped to schematron constraints or may be just expressed as text in documentation annotations.

The relationship between an application schema in UML and in GML is described as part of the Annexes E and F dealing with the mapping between ISO 19109 and GML application schemas.

D.2.7 ISO 19111 spatial referencing by coordinates

D.2.7.1 Overview

The full conceptual model of ISO 19111 is implemented in GML. The mapping is in general a straightforward mapping from the conceptual model to the GML schema. Only, schema components with a more complex mapping are detailed in D.2.7.

Wherever a GML object is associated with a Coordinate Reference System, this is implemented by an attribute (`srsName`) pointing to a `gml:AbstractCRS` element.

Additional attributes are defined in the content model of the same element carrying redundant information about the coordinate reference system. `srsDimension` is the dimension of the coordinate reference system as stated in the coordinate reference system definition. The `axisLabels` and `uomLabels` attributes are lists of the labels and units of measurement associated with the different axes of the coordinate reference system.

D.2.7.2 Identified object package

For the implementation in GML, the IO_IdentifiedObjectBase and the SC_CRS classes are merged into the IO_IdentifiedObject class to support a simpler XML encoding of the IO_IdentifiedObject class. Note that the “name” attribute of SC_CRS is mapped to the “name” attribute of IO_IdentifiedObject.

The IO_IdentifiedObject type is implemented by `gml:IdentifiedObject` element and its content model is derived-by-extension from `gml:DescriptionType`.

RS_Identifier is implemented by `gml:CodeType`. If used, the "version [0..1]" attribute shall be represented in the `codeSpace` attribute of `gml:CodeType`.

EXAMPLE A name for coordinate reference system "4326" of the International Association of Oil and Gas Producers' EPSG dataset (EPSG) may be represented using a URI as:

<name>http://www.opengis.net/def/crs/EPSG/0/4326</name>

The use of the optional `gml:description` property is supported to allow encoding additional information about each CRS object.

The "name" attribute is implemented by the `gml:identifier` property.

The "identifier" attribute is implemented by the `gml:name` property.

The "alias" attribute is implemented by the `gml:name` property, too.

NOTE In a reverse mapping, identifiers would be those `gml:name` with a value or `codeSpace` that could be mapped to an RS_Identifier and all other would become aliases.

D.2.7.3 Coordinate reference system package

The SC_CRS class is implemented by `gml:AbstractCRS`.

The "coordinateSystem"/"…CS" association role to a concrete subclass of the CS_CoordinateSystem class is implemented by `gml:coordinateSystem/gml:…CS` where the specific CS type name is substituted for ellipses.

The "datum" association role to a concrete subclass of the CD_Datum class is implemented by `gml:…Datum` where the specific datum type name is substituted for ellipses.

NOTE This name change is required because global names are used and otherwise the type of the target class could not be specified in the GML schema.

The "componentReferenceSystem" association role from the SC_CompoundCRS class to the SC_SingleCRS class, is implemented by the `gml:componentReferenceSystem` property.

The "baseCRS" association, from the SC_ProjectedCRS class to the SC_GeodeticCRS class, is implemented by the `gml:baseGeodeticCRS` property.

D.2.7.4 Coordinate system package

Besides the implementation rules specified in D.2.7.2 and D.2.7.3 the mapping is straightforward with the exception that the "axisUnitID" attribute of CS_CoordinateSystemAxis is implemented by the `uom` XML attribute.

D.2.7.5 Datum package

Besides the implementation rules specified in D.2.7.2 and D.2.7.3 the mapping is straightforward, except for the fact that the order of the properties in the CD_Datum has been changed in the mapping to `gml:AbstractDatum`.

D.2.7.6 Coordinate operation package

The CC_Conversion and CC_Transformation classes are implemented in two steps as separate abstract and concrete elements.

The parameterValue and method association roles from the CC_Operation class are implemented in the concrete `gml:Conversion` and `gml:Transformation` elements, to reduce the need to use XML Schema restriction.

The order of the properties in the CC_CoordinateOperation has been changed in the mapping to `gml:AbstractCoordinateOperation`.

The "coordOperation" association role, from the CC_ConcatenatedOperation class to the CC_CoordinateOperation class, is implemented by the `gml:coordOperation` property.

The "parameter" association roles to the CC_GeneralOperationParameter or CC_OperationParameter class are implemented by the `gml:generalOperationParameter` or the `gml:operationParameter` property depending on the target class.

D.2.8 ISO 19112 spatial referencing by geographic identifiers

GML does not provide a predefined schema for gazetteers. However, it does provide predefined properties for spatial references by geographic identifiers:

— The property <gml:locationName> contains a text that describes the location.

— The property <gml:locationReference> references a text that describes the location.

D.2.9 ISO 19115 metadata

GML does not provide an information model for metadata. Instead a mechanism to include or reference metadata is provided for all object elements.

NOTE As specified in 7.2.6, if metadata following the conceptual model of ISO 19115 is to be encoded in a GML document, the corresponding XML Schema specified in ISO/TS 19139 shall be used to encode the metadata information.

D.2.10 ISO 19118 encoding

The encoding rules described in E.2 conforms to ISO 19118 Level 1.

D.2.11 ISO 19123 coverages

The UML model of the GML profile defined in this annex describes a conceptual model of the abstract types defined in ISO 19123. Table D.8 maps GML Coverage object and property names to the corresponding class names and their attributes in ISO 19123 to ease the comparison with that standard.

Table 17 — Description of the implementation of ISO 19123

ISO 19123 construct	GML construct
CV_Coverage	AbstractCoverage (AbstractCoverageType)
domainExtent (attribute)	boundedBy (property)
domainElement (role name)	domainSet (property)
rangeElement (role name)	rangeSet (property)
AttributeValues	ValueArray, or AbstractScalarValueList
CoverageFunction (association)	coverageFunction (property)
CV_GridValuesMatrix	GridFunction
sequencingRule (attribute)	sequenceRule (property)
CV_SequenceRule	SequenceRuleType
CV_SequenceType	SequenceRuleNames
scanDirection (attribute)	order (attribute)
startSequence (attribute)	startPoint (property)
CV_Grid	Grid
CV_GridEnvelope	GridEnvelope
low (attribute)	low (property)
high (attribute)	high (property)
CV_RectifiedGrid	RectifiedGrid
origin (attribute)	origin (property)
offsetVectors (attribute)	(set of) offsetVector(s) (property)
CV_DiscretePointCoverage	MultiPointCoverage
CV_DiscreteCurveCoverage	MultiCurveCoverage
CV_DiscreteSurfaceCoverage	MultiSurfaceCoverage
CV_DiscreteGridPointCoverage	GridCoverage or RectifiedGridCoverage

The additional changes shown in Table D.9 have been applied to the coverage package of ISO 19123.

Table 18 — Description of the profile of ISO 19123

Change	Rationale
All subclasses of CV_ContinuousCoverage deleted	Currently not supported by GML.
Coordinate Reference System association deleted from CV_Coverage	Replaced by srsName (or frame) attributes on the Geometry or Temporal objects in the domain. The GML coverage package allows for domain objects to be in different coordinate reference systems (or reference frames).
CV_CommonPointRule deleted	Currently not supported by GML.
AttributeValues deleted	Replaced by a choice between the GML analogues: ValueArray or AbstractScalarValueList.
CV_GeometryValuePair deleted	The GML coverage encodes only the domain-range functional viewpoint.
CV_GridValuesMatrix deleted	The mapping between grid points and range values including sequence rule, etc. is contained in an object (GridFunction) that does not inherit from Grid.
CV_GridCell	Currently not supported by GML.

The UML class diagram in Figure D.39 illustrates the profile of the “Coverage root” package (compare with ISO 19123:2005, Figure 2).

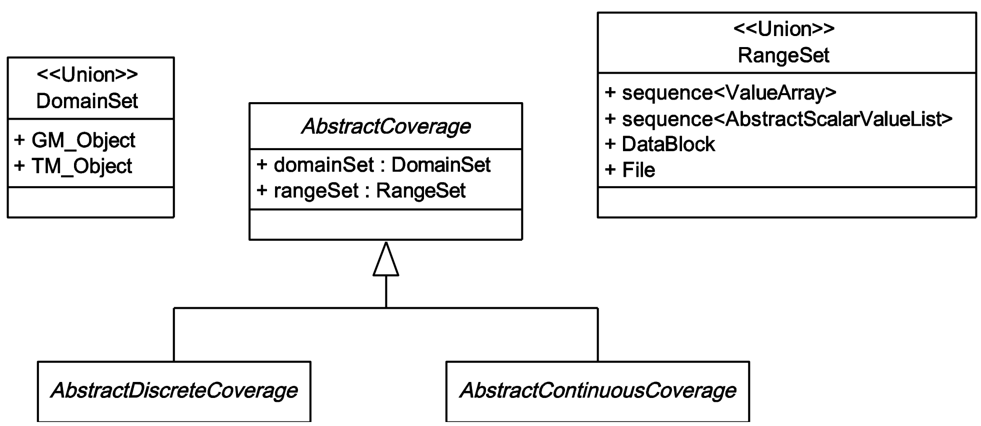


Figure 47 — Coverages overview

The UML class diagram in Figure D.40 illustrates the discrete coverages (compare with ISO 19123:2005, Figures 3, 4 and 5).

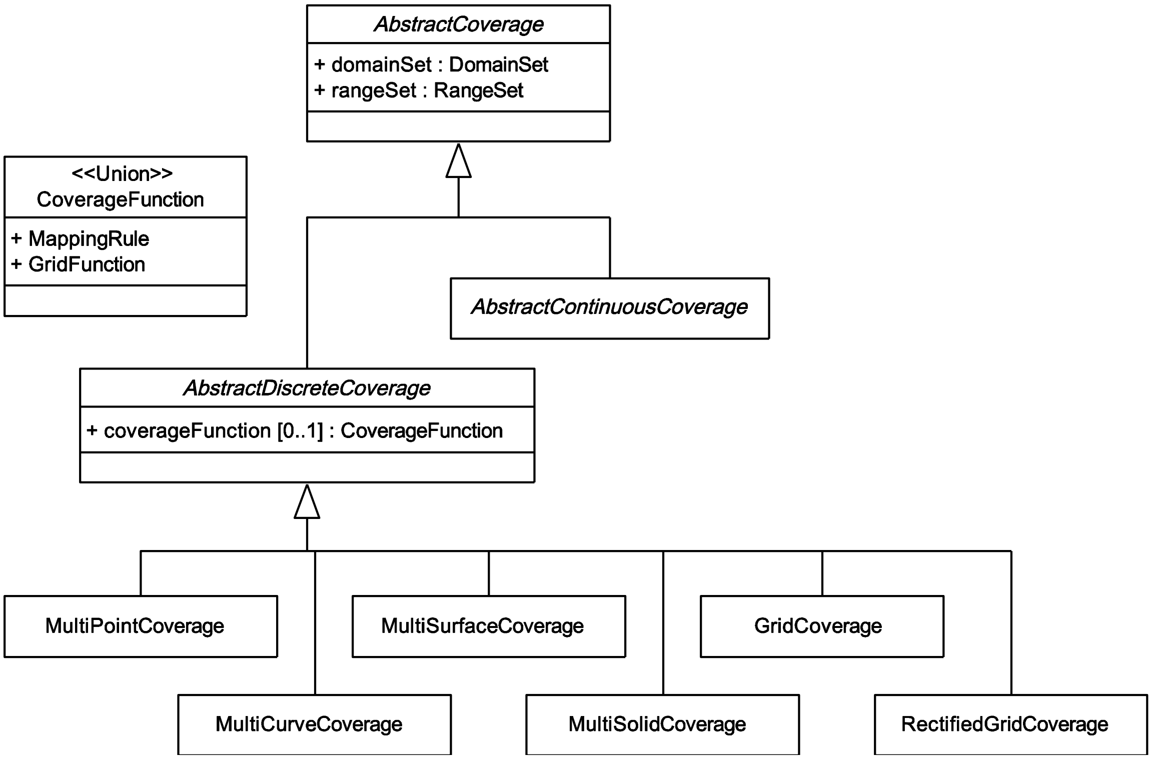


Figure 48 — Discrete coverages

The rules governing conformance of a profile of ISO 19123 are described in ISO 19123:2005, Clause 2 and Annex A.

The conditions of the following conformance clauses are met:

— A.1.1 Simple coverage interface

— A.1.2 Discrete coverage interface

— A.1.4 Quadrilateral grid coverage interface

Note that derived attributes are treated as operations and it is assumed that the derived attributes will be derived from the data by the application handling the GML instances.

D.3 Extension of the profile of the ISO 19100 series of International Standards

D.3.1 Overview

The following subclauses define the additional parts of GML that are not covered by the profile of the ISO 19100 series defined in D.2. UML is used as the conceptual schema language to describe the additional elements in accordance with ISO/TS 19103. For details on the semantics of the additional classes see Clauses 7 to 19.

The GML schema components have been grouped semantically based on the structure of the Clauses 7 to 19 and a package for each grouping is created. The required additional classes not documented in D.2 are defined in the corresponding package. The packages are part of a package “GML”.

D.3.2 Package "basicTypes"

In addition to the types from ISO/TS 19103, 8.2 defines a number of additional types that are used and required by other GML schema documents.

Most of these additions are the result of the capability to provide information about void information (nilReason attributes). It has been added to the GML schema based on user requirements and since the concept was considered to be of general utility (see Figures D.41 and D.42).

The list types are just convenience types to simplify the writing of the GML schema.

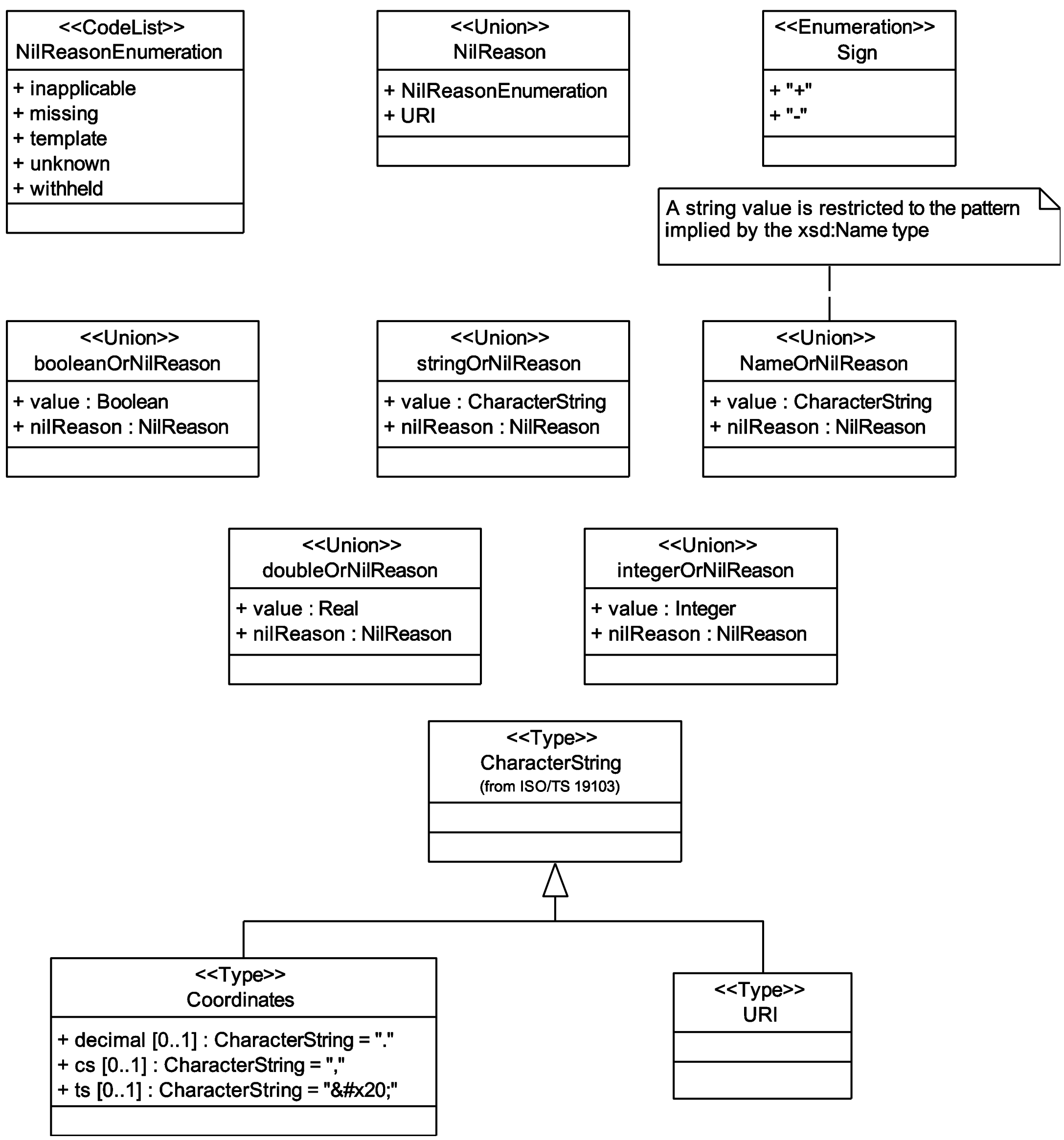


Figure 49 — Simple types

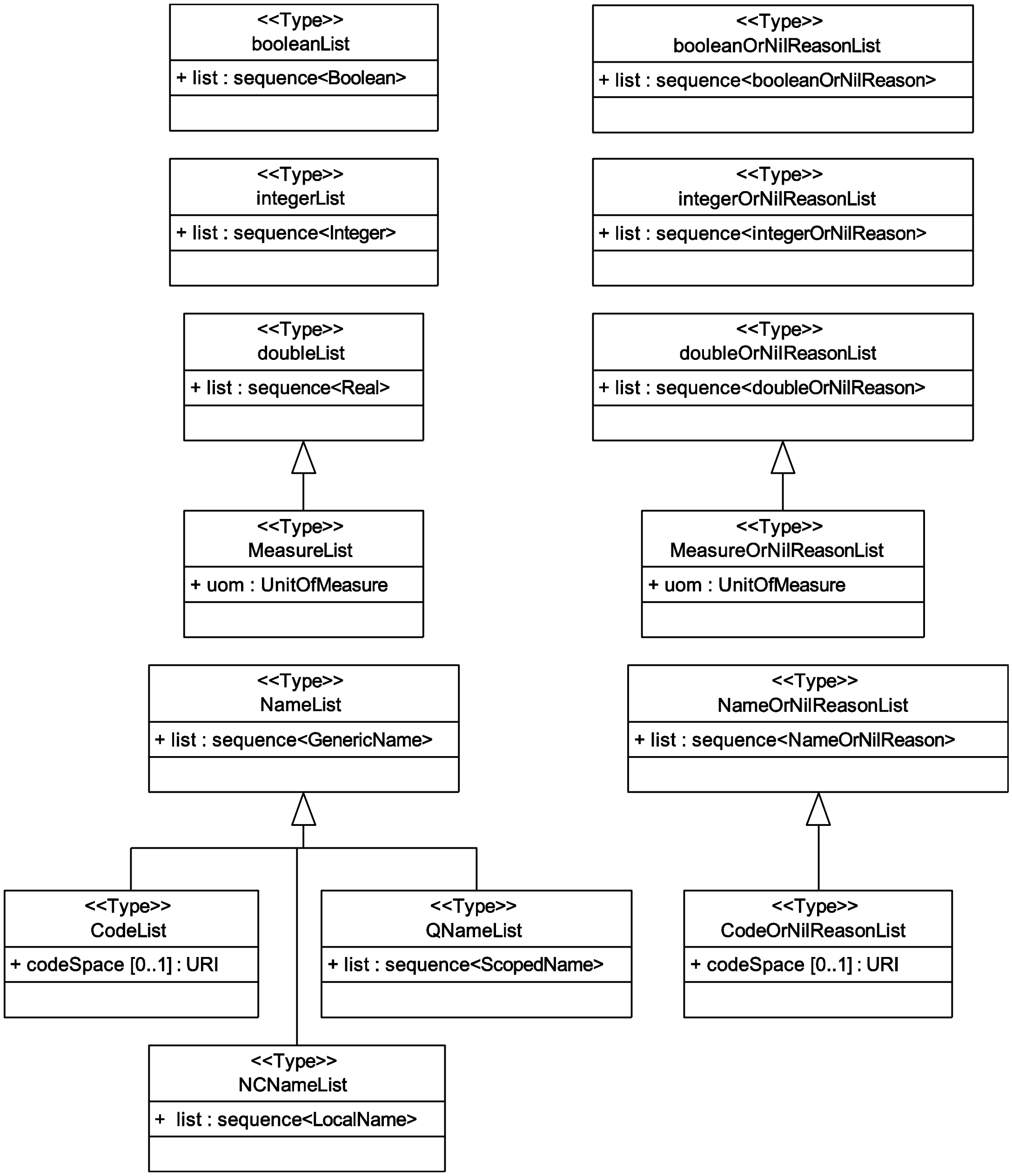


Figure 50 — Lists

D.3.3 Package "gmlBase"

In addition to the types from ISO/TS 19103, 7.2 defines few additional types that are used by other GML schema documents besides those that are part of the mapping from the conceptual schema to the XML Schema implementation. The representation of object types, association roles, etc. in XML Schema is described in Annex E.

`gml:AbstractObject`, `gml:AbstractGML` and `gml:AbstractFeature` (see D.3.4) make general concepts explicitly available for use in an application schema. This is required/useful, for example when a property may carry a value that is any feature (see Figures D.43 and D.44).

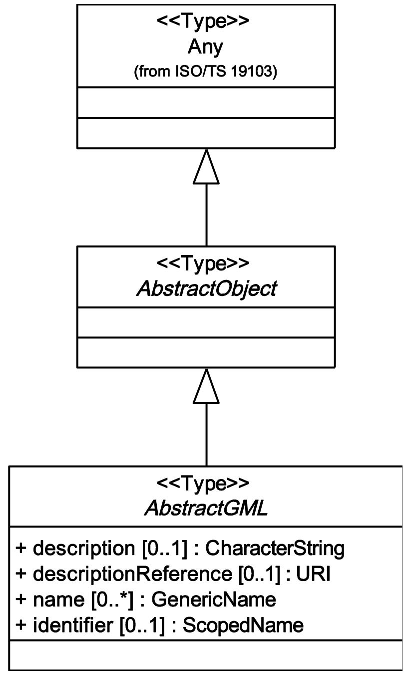


Figure 51 — Base types

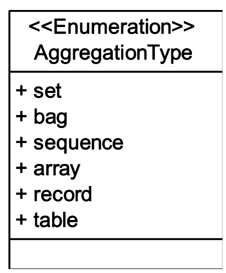


Figure 52 — Aggregation types of collection

D.3.4 Package "feature"

This subclause specifies additional types used in Clause 9. The types clarify the representation of feature types and common, predefined, optional property elements, namely names, an identifier, a description and a bounding envelope (see Figure D.45).

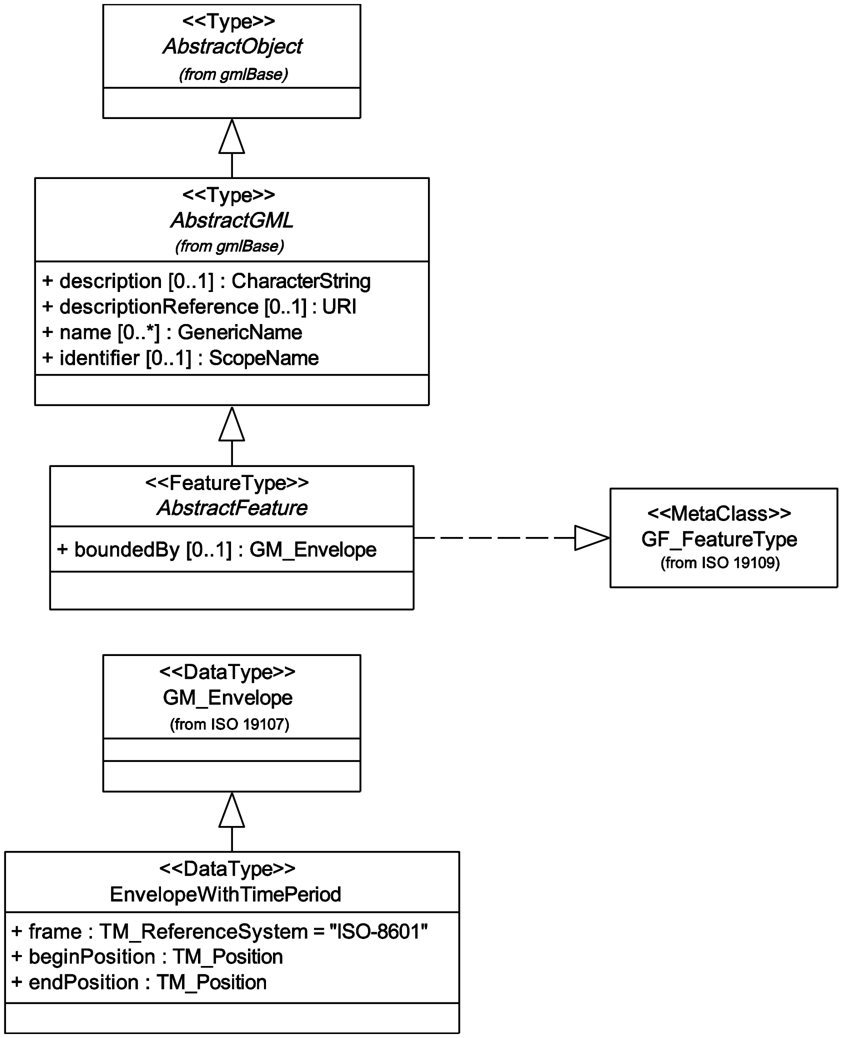


Figure 53 — Features

D.3.5 Package “geometryBasic0d1d”

This subclause specifies additional types used in 10.3 and 10.4.

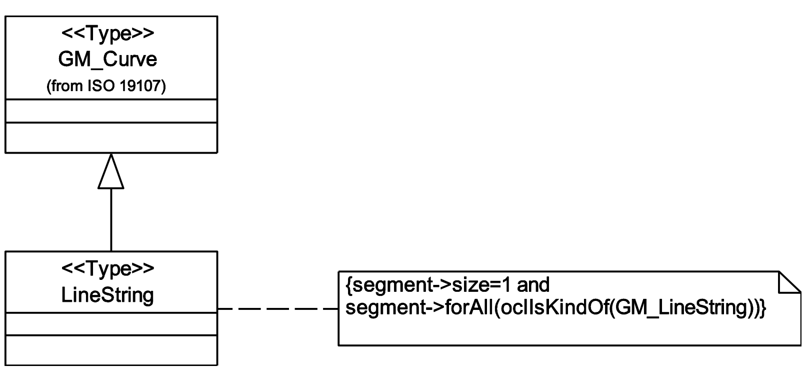


Figure 54 — GM_Curve with a single GM_LineString segment

An additional subtype of GM_Curve is added in GML. “LineString” is a special curve that consists of only GM_LineString segments. The XML representation of the “LineString” object element joins all the segments into one segment and its control points are represented as direct properties of the “LineString”. (See Figure D.46.)

The "LineString" type as specified above was added as a convenience type since it represents a typical case in practice.

D.3.6 Package “geometryBasic2d”

This subclause specifies additional types used in 10.4.5.

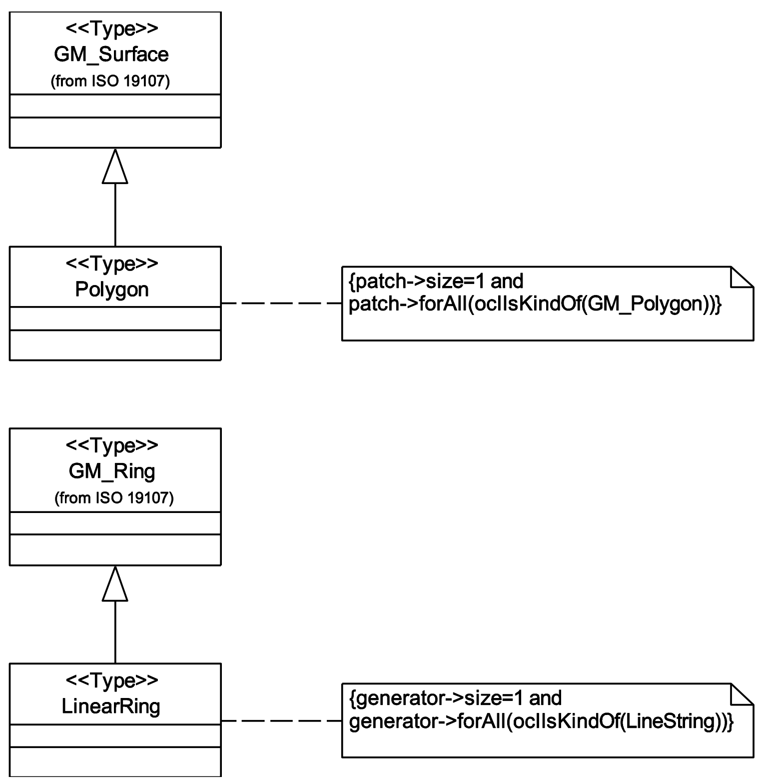


Figure 55 — A GM_Surface with a single GM_Polygon patch

An additional subtype of GM_Surface is added in GML. “Polygon” is a special surface that consists of a single GM_Polygon patch. Since only a single patch exists, the XML representation of the “Polygon” object element skips the patch level and the exterior and interior boundary properties of the patch are represented as direct properties of the “Polygon”.

In a similar way, “LinearRing” is a simple ring (a GM_Ring represented by a single line string).

The "Polygon" and "LinearRing" types as specified above were added as convenience types since they represent a typical case in practice. (See Figure D.47.)

D.3.7 Package “geometryPrimitives”

This subclause specifies additional types used in 10.5.10.

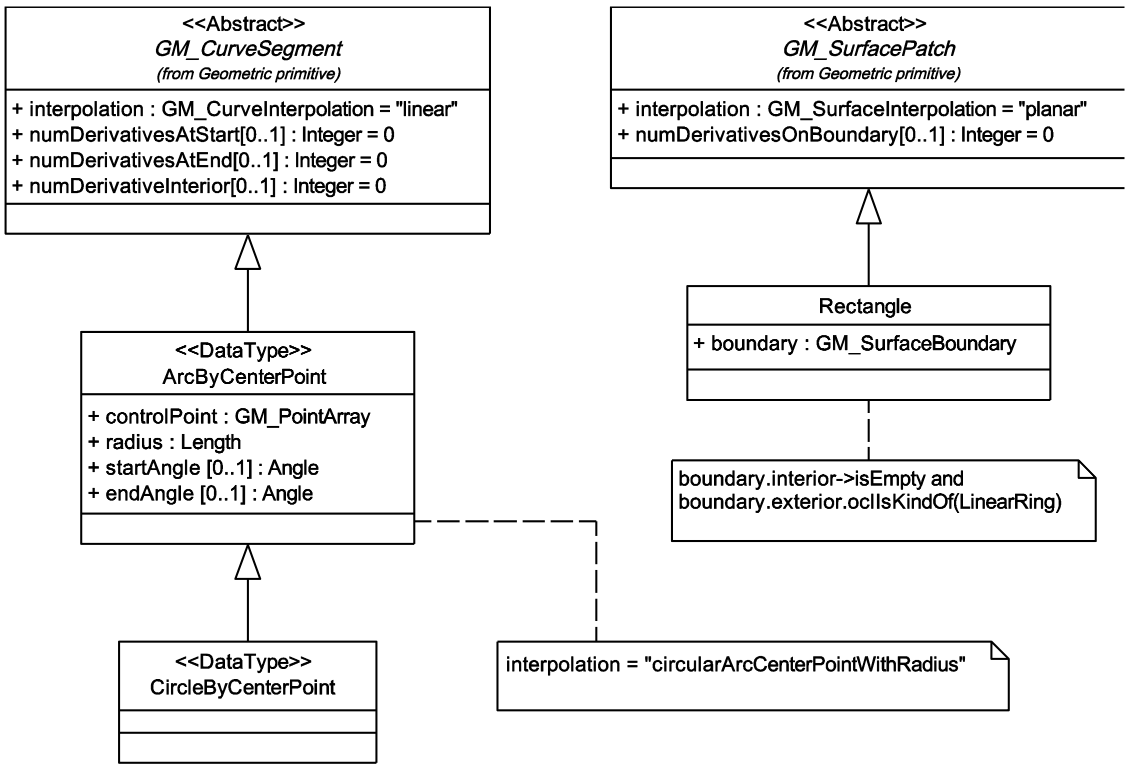


Figure 56 — Additional curve segments and surface patches

The curve segments “ArcByCenterPoint” and “CircleByCenterPoint” as well as the surface patch “Rectangle” have been defined in GML since these representations of a circle or arc are commonly used in several application domains (see Figure D.48).

D.3.8 Package “geometryAggregates”

This subclause specifies additional types used in 11.3.

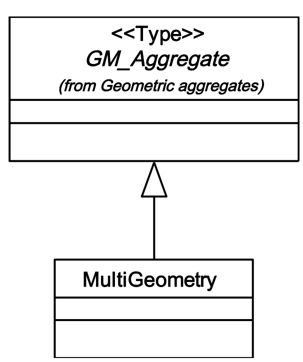


Figure 57 — Additional geometric aggregates

GML defines an instantiable geometric aggregate which is not restricted to elements of a single dimension: "MultiGeometry". This type has been added since such a collection of geometry objects is used in several applications and it is considered a generally useful concept. (See Figure D.49.)

D.3.9 Packages "coordinateOperations", "coordinateReferenceSystems", "coordinateSystems", "dataQuality", "datums", "referenceSystems"

D.3.9.1 Overview

D.3.10 specifies temporal reference systems as an additional coordinate reference system subtype along with types for temporal datums and time coordinate systems.

The mapping to the GML objects implementing these types (`gml:TemporalCRS`, `gml:TimeCS` and `gml:TemporalDatum`) is straightforward and follows the rules described in D.2.7.

D.3.9.2 UML schema of package "coordinateReferenceSystems"

Figure D.50 shows the UML class diagram of the coordinateReferenceSystems package relevant to temporal CRS.

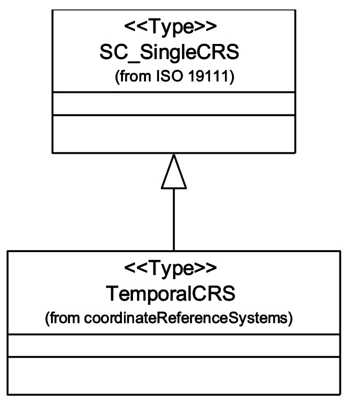


Figure 58 — TemporalCRS

Table 19 — Defining elements of TemporalCRS class

Description:	A 1D coordinate reference system used for the recording of time.
Stereotype:	(none)
Derived from	SC_SingleCRS
Association roles:	datum to TemporalDatum [1] coordinateSystem to CS_TimeCS [1] (associations inherited from SC_SingleCRS)
Public attributes:	6 attributes inherited from IO_IdentifiedObjectBase, RS_ReferenceSystem and SC_CRS.

D.3.9.3 UML schema of package "coordinateSystems"

A time coordinate system is a 1-dimensional coordinate system containing a single time axis and is used to describe the temporal position of a point in the specified time units from a specified time origin.

Figure D.51 shows the UML class diagram of the coordinateSystems package relevant to time coordinate systems. A restriction on the association between SC_SingleCRS and CS_CoordinateSystem is shown in the UML class diagram in Figure D.52.

There are restrictions on associations between Coordinate Reference System subtypes and Coordinate System subtypes are shown in the UML class diagram in Figure D.52.

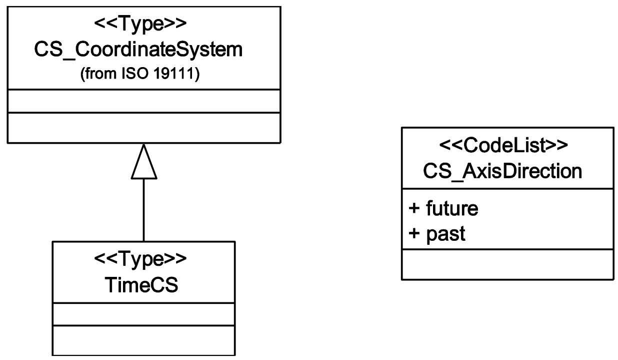


Figure 59 — TimeCS

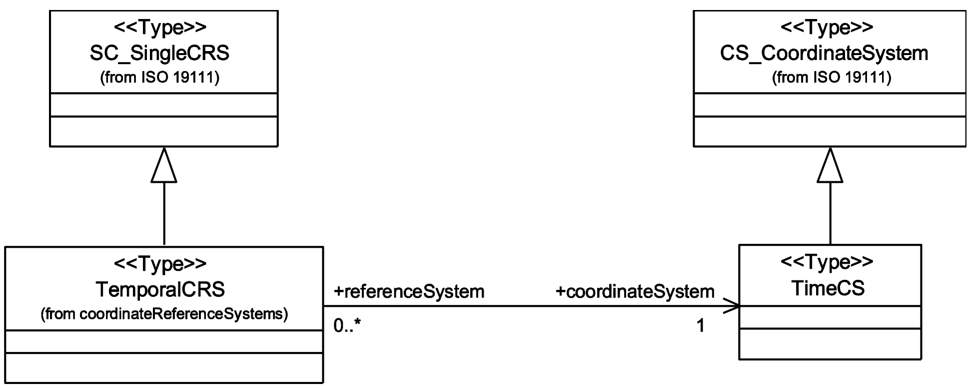


Figure 60 — Association between TemporalCRS and TimeCS

Table D.11 — Defining elements of TimeCS class

Description:	A one-dimensional coordinate system containing a time axis, used to describe the temporal position of a point in the specified time units from a specified time origin. A TimeCS shall have one axis association.
Stereotype:	(none)
Derived from	CS_CoordinateSystem
Association roles:	coordinateSystem from TemporalCRS [1]
	(associations inherited from CS_CoordinateSystem)
Public attributes:	4 attributes inherited from IO_IdentifiedObject and IO_IdentifiedObjectBase.

Table D.12 — Defining elements of CS_AxisDirection class

Description:	The direction of positive increase in the coordinate value for a coordinate system axis.				
Stereotype:	CodeList				
Derived from	(none)				
Association roles:	(none)				
Used by:	CS_CoordinateSystemAxis				
Public attributes:					
Attribute name	**UML identifier**	**Data type**	**Obligation**	**Maximum occurrence**	**Attribute description**
future	Future	CharacterString	C	1	Axis positive direction is towards the future.
past	Past	CharacterString	C	1	Axis positive direction is towards the past.
Condition: One and only one of the listed attributes shall be supplied.					

D.3.9.4 UML schema of package "datums"

Figure D.53 shows the UML class diagram of the datums package relevant to temporal datums. A restriction on the association between SC_SingleCRS and CD_Datum is shown in the UML class diagram in Figure D.54.

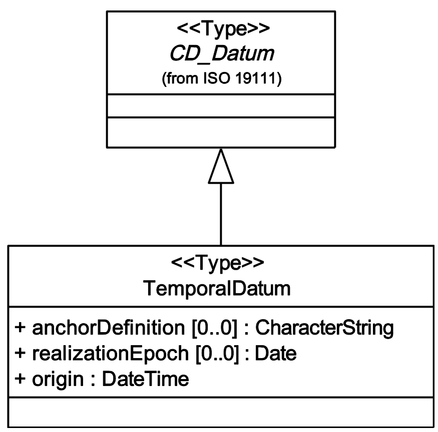


Figure 61 — TemporalDatum

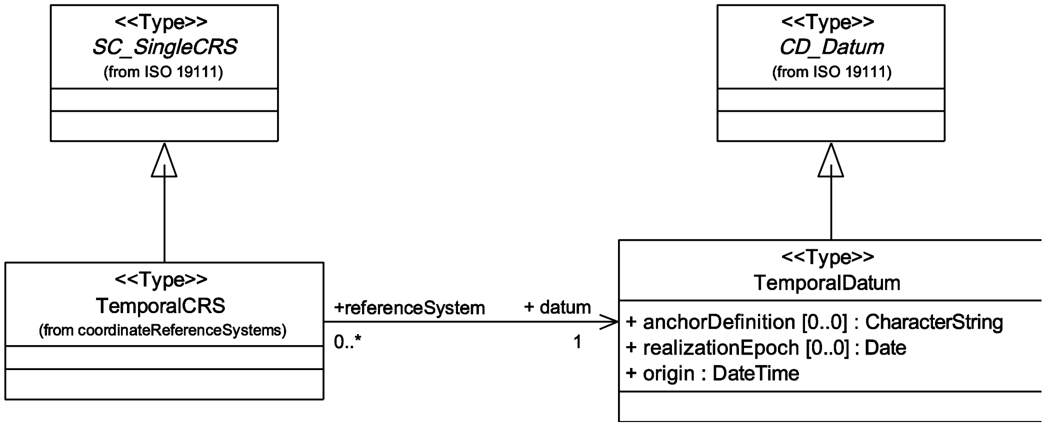


Figure 62 — Association between TemporalCRS and TemporalDatum

Table D.13 — Defining elements of TemporalDatum class

Description:	A temporal datum defines the origin of a temporal reference system.				
Stereotype:	(none)				
Derived from	CD_Datum				
Association roles:	datum from TemporalCRS [1]				
Public attributes:	8 attributes inherited from CD_Datum, IO_IdentifiedObject and IO_IdentifiedObjectBase, plus:				
Attribute name	**UML identifier**	**Data type**	**Obligation**	**Maximum occurrence**	**Attribute description**
Origin	origin	DateTime	M	1	The date and time origin of this temporal datum.
Of the 8 inherited attributes the following two are modified:					
Anchor definition	anchorDefinition	CharacterString	M	0	This attribute is not used by a temporal datum.
Realization epoch	realizationEpoch	Date	M	0	This attribute is not used by a temporal datum.

D.3.10 Package “topology”

This subclause specifies additional types used in Clause 13.

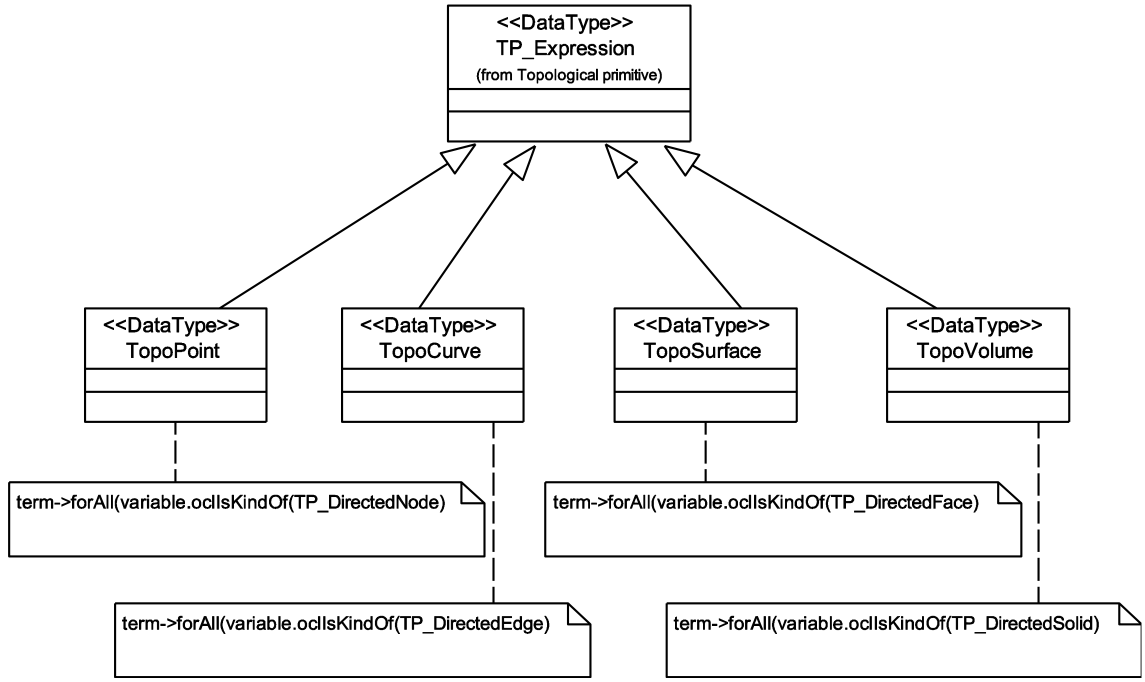


Figure 63 — Topological expressions

GML defines several data types containing (or referencing) directed topological primitives, one per dimension: “TopoPoint”, “TopoCurve”, “TopoSurface” and “TopoVolume”. These are convenience types which are intended to be used in properties of features in applications. (See Figure D.55.)

D.3.11 Package “dynamicFeature”

This subclause specifies additional types used in 14.5.

The dynamic feature concept has been added to GML, because a capability to express time varying properties has been considered a fundamental concept of geographic information. (See Figure D.56.)

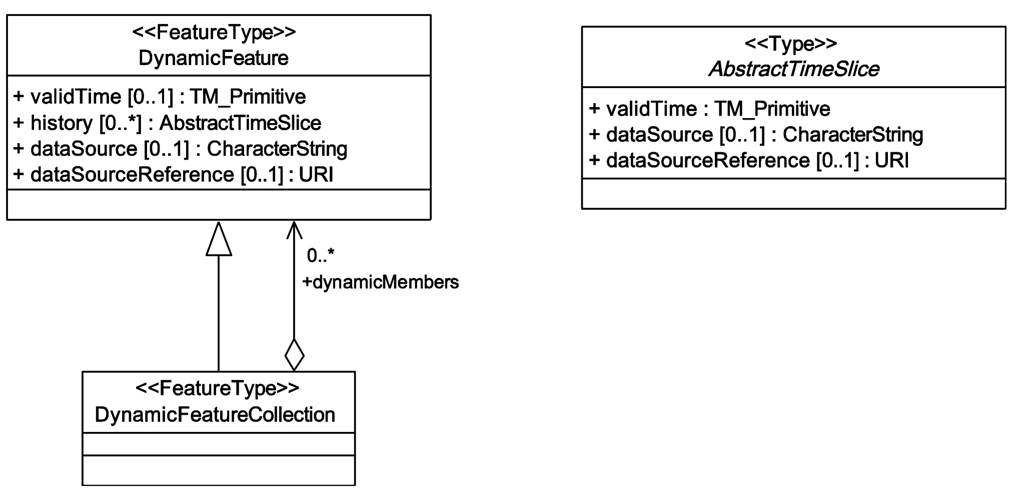


Figure 64 — Dynamic features

D.3.12 Package "dictionary"

This subclause specifies additional types used in Clause 15. The dictionary concept has been added to GML, because a capability to encode dictionaries of code lists, units and coordinate reference systems is fundamental for working with instance data and application schemas. (See Figure D.57.)

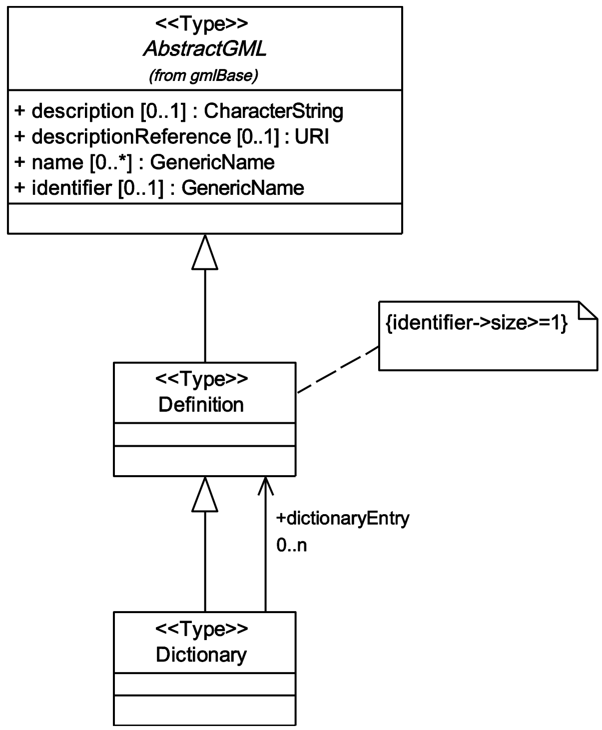


Figure 65 — Dictionaries

NOTE The "Dictionary" class had to be named GMLDictionary to avoid a naming conflict with the class with the same name in ISO/TS 19103.

D.3.13 Package "units"

This subclause specifies additional types used in 16.2. The schema has been specified as part of GML, because the model used in ISO/TS 19103 was not sufficient to express the information required about units of measurement (see Figure D.58).

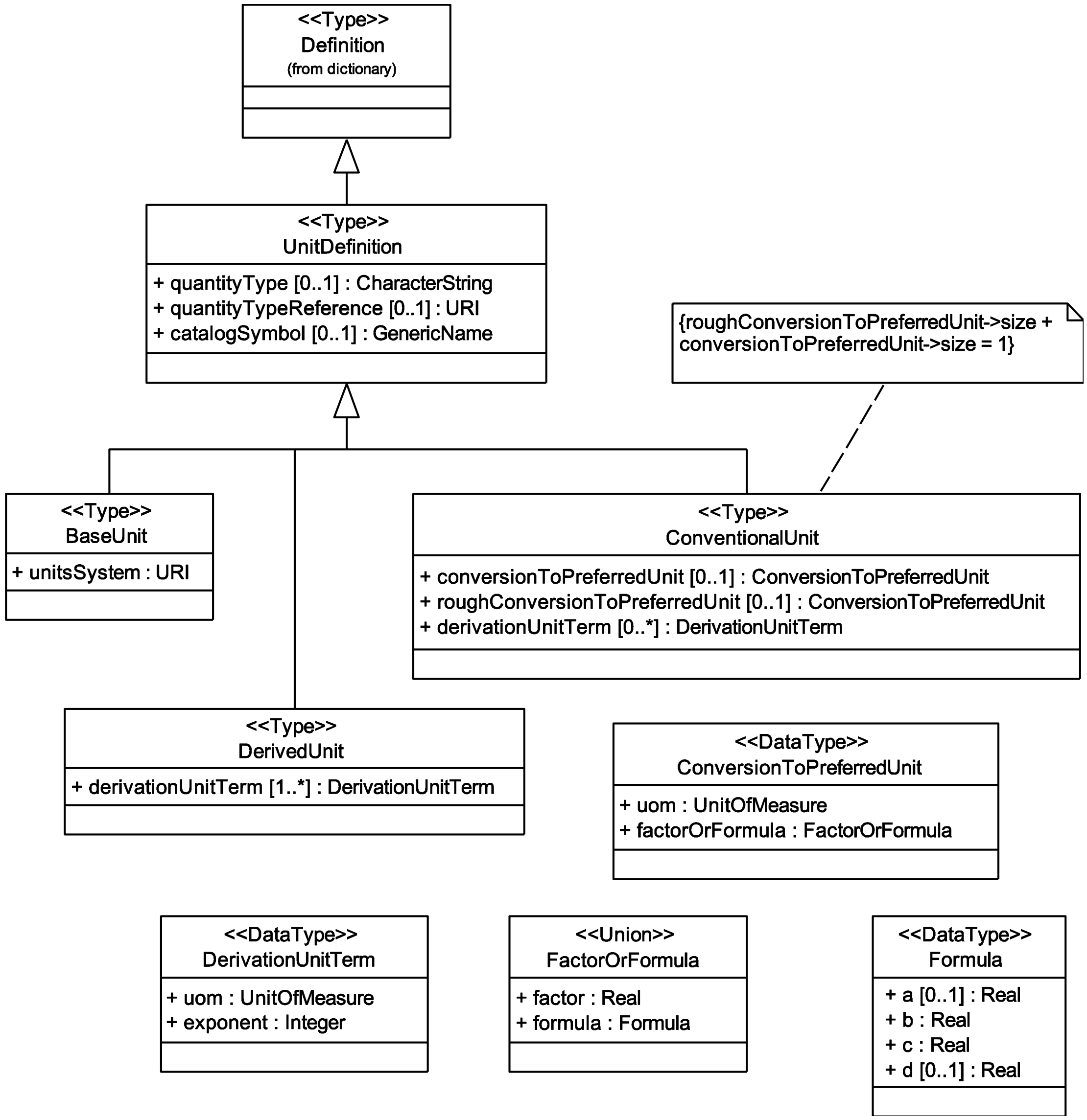


Figure 66 — Unit definitions

D.3.14 Package “measures”

This subclause specifies additional types used in 16.3. In addition to the subtypes of "Measure" as specified in ISO/TS 19103, another subtype has been used in the context of grids and thus has been specified as part of the GML schema (see Figure D.59).

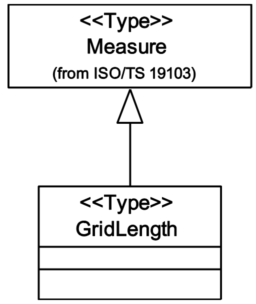


Figure 67 — Additional measures

D.3.15 Package “valueObjects”

This subclause specifies additional types used in 16.4. These types are used in the observations schema (see D.3.17).

The component hierarchy is illustrated in the UML class diagrams in Figures D.60 to D.64. UML generalization relationships are used to indicate XML Schema substitution group and choice group membership. UML composition relationships are used to indicate membership in an XML Schema type content model.

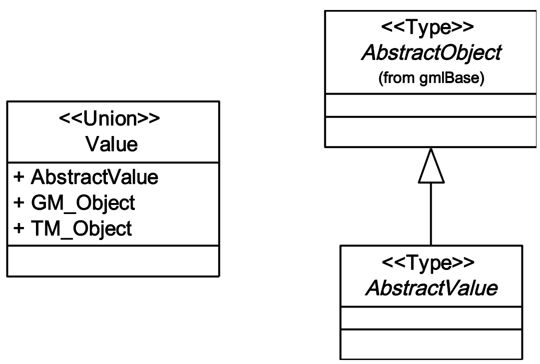


Figure 68 — Value expressions

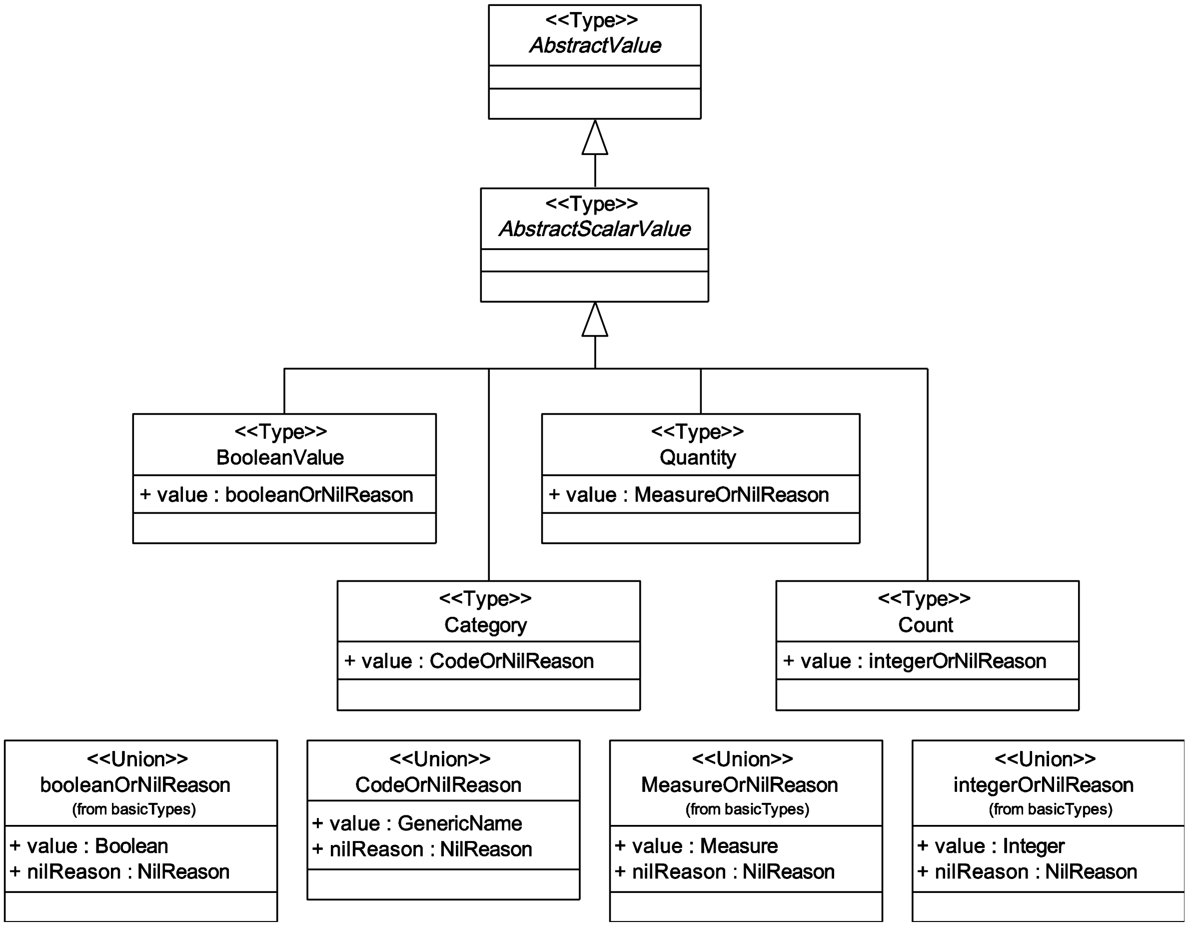


Figure 69 — Scalar values

NOTE To avoid a naming conflict with the class Boolean in ISO/TS 19103, the boolean value object class has been named BooleanValue.

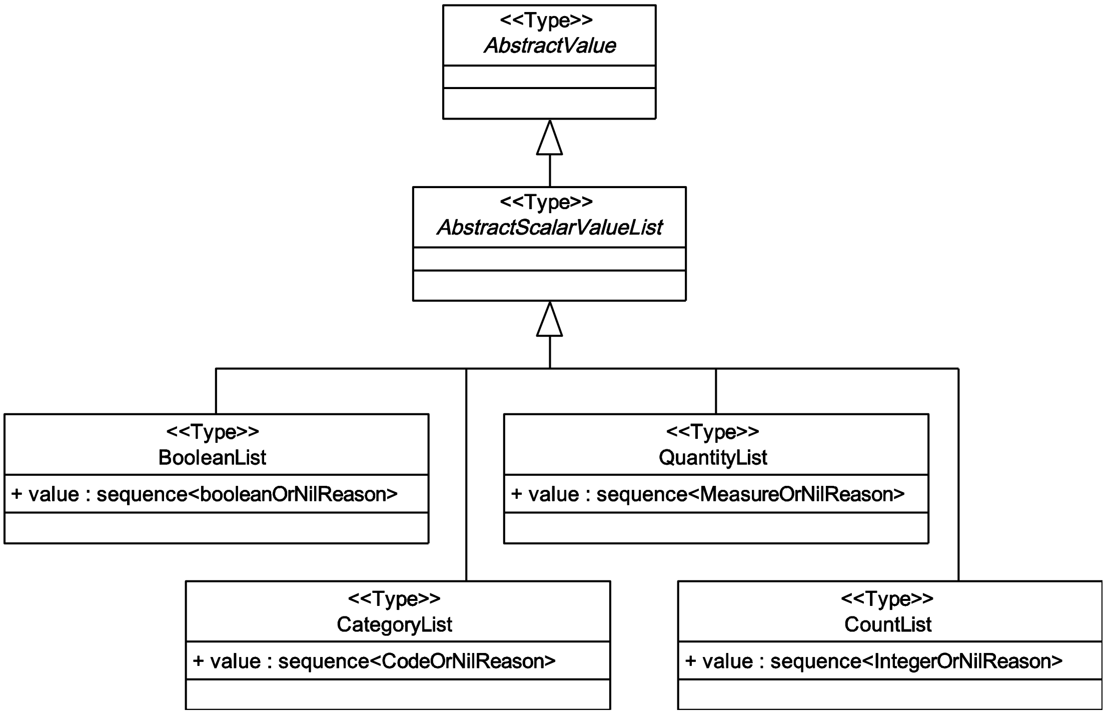


Figure 70 — Scalar value lists

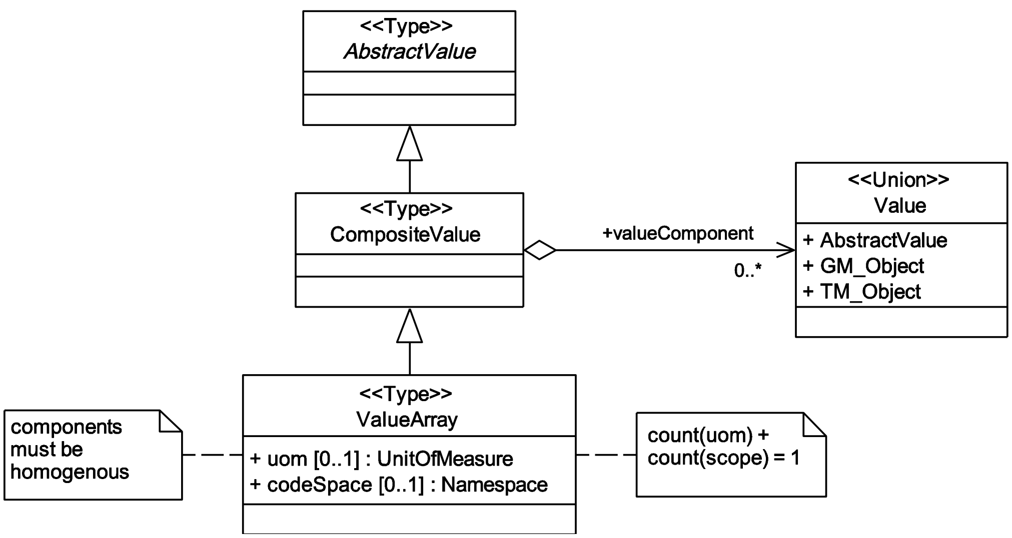


Figure 71 — Composite value

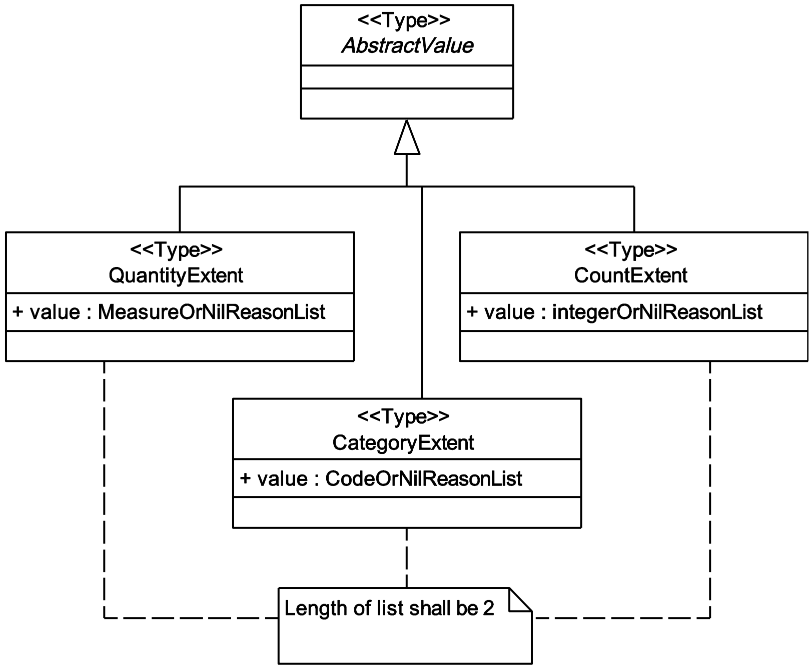


Figure 72 — Extents

D.3.16 Package "direction"

This subclause specifies additional types used in Clause 17 (see Figure D.65). These types are used in the observations schema (see D.3.17).

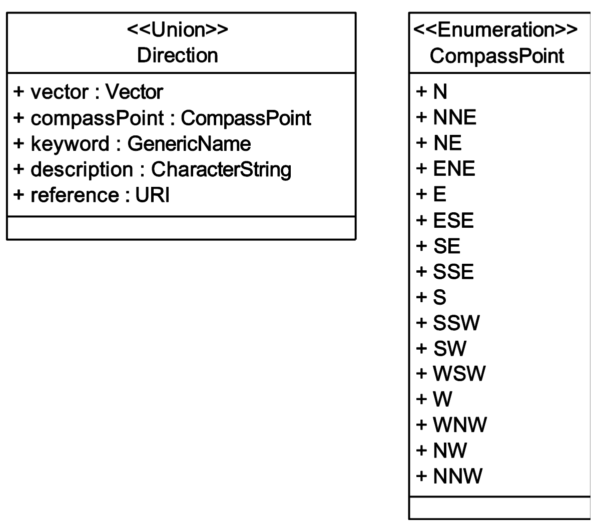


Figure 73 — Direction

D.3.17 Package "observation"

This subclause specifies additional types used in Clause 18 (see Figure D.66).

The observation concept has been added to GML, because the concept of observations is considered a fundamental concept of geographic information.

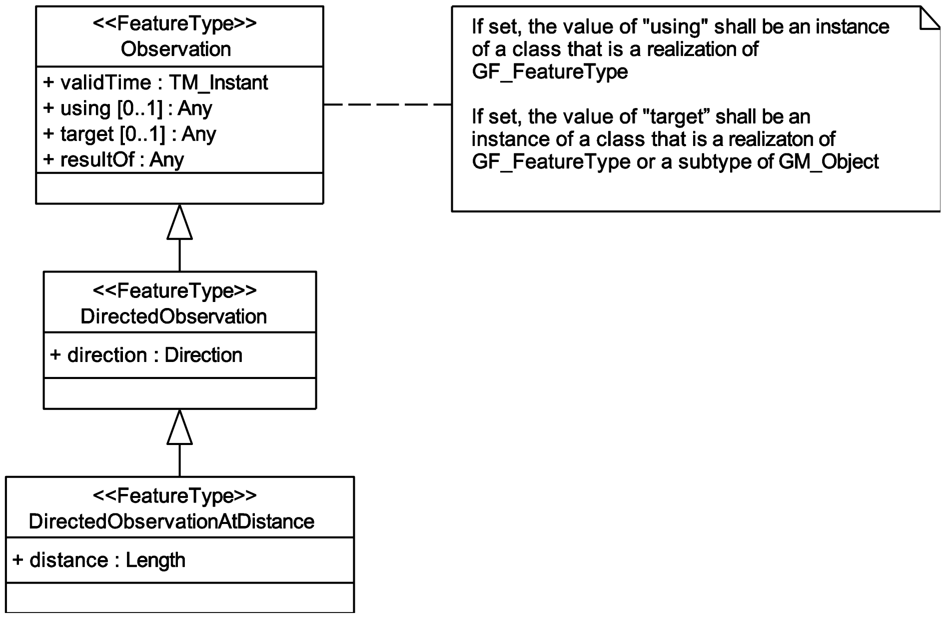


Figure 74 — Observation

Annex E
(normative)

UML-to-GML application schema encoding rules

E.1 General concepts

The mapping from an ISO 19109 conformant UML Application Schema to the corresponding GML application schema is based on a set of encoding rules. These encoding rules are compliant with the rules for GML application schemas and are based on ISO 19118.

The rules are derived from the rules for the GML model and syntax as described in Clauses 7 to 21, especially Clause 7. The encoding rules of ISO 19118:2005, Annex A, are used whenever possible and feasible.

The rules listed in this annex aim at an automatic mapping from an ISO 19109 and ISO/TS 19103 conformant UML application schema to a GML application schema (in accordance with the rules defined in Clause 21). As a result of this automation, the resulting GML application schema will not make full use of the capabilities of XML and XML Schema, but will provide an XML implementation conformant to the ISO 19100 series of International Standards with a well-defined, predictable XML grammar.

These rules do not prescribe that all GML application schemas shall be generated by using these rules. All schemas following the rules defined in Clause 21 are valid and conformant GML application schemas, whether they are handcrafted, automatically derived from a UML application schema or produced by some other means.

The schema encoding rules are based on the general idea that the class definitions in the application schema are mapped to type and element declarations in XML Schema, so that the objects in the instance model can be mapped to corresponding element structures in the XML document.

E.2 Encoding rules

E.2.1 General encoding requirements

E.2.1.1 Application schemas

E.2.1.1.1 General (application schema, packages)

To be a valid input into the mapping the UML Application Schema shall conform to all of the following rules. See ISO 19118:2005, A.2.1, for additional requirements.

The UML Application Schema shall conform to the rules defined in ISO 19109 and ISO/TS 19103.

The UML Application Schema shall be represented by a package with the stereotype <<Application Schema>>. This package shall contain (i.e. own directly or indirectly) all UML model elements to be mapped to object types in the GML application schema. The package may include other packages without the stereotype <<Application Schema>> to group the different UML model elements within the application schema.

The UML model shall be complete and not contain external references unless exceptions are explicitly stated below. Predefined classes may be imported from the standardized schemas of the ISO 19100 series of International Standards. The classes from the ISO 19100 series of International Standards that are implemented by the GML schema and used by the UML application schema shall be specified in a package with the name "ISO19100" or any sub-package of a package with that name.

Dependencies between packages shall be modelled explicitly. Permission elements with stereotype <<import>> or unspecified dependency elements between packages shall be used to express the dependency of elements in a package from elements in another package. All other dependency elements shall be ignored, see Figure E.1.

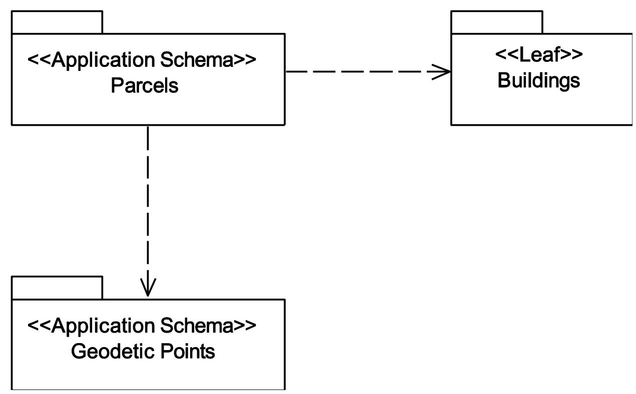


Figure 75 — Dependency between packages <informative>

The visibility of all UML elements shall be set to “public”. Only publicly visible elements shall be part of Application Schemas used for data interchange between applications.

Documentation of the elements in the UML model shall be stored in tagged values “documentation”.

A unique XML namespace shall be associated with the UML Application Schema. Tagged values “targetNamespace“ for the target namespace URI and “xmlns“ for the abbreviation shall be set if and only if the package represents a UML application schema.

The version number of a package representing a UML Application Schema shall be specified in a tagged value “version”, if applicable.

A GML profile may be associated with the application schema by a tagged value “gmlProfileSchema”. If provided, the value shall be a URL referencing the schema location of the GML profile.

If a package shall be mapped to its own XML Schema document, a tagged value "xsdDocument" shall be set providing a valid relative file name of the schema document. The tagged value shall be set for every package representing the UML Application Schema. All tagged values "xsdDocument" in a UML model shall be unique.

EXAMPLE The value of an "xsdDocument" tagged value might be "GeodeticPoints.xsd" or "schemas/Parcels.xsd".

E.2.1.1.2 Classes

All class names within the same Application Schema shall be unique and an "NCName" as defined by W3C XML Namespaces:1999.

Feature types shall be modelled as UML classes with stereotype <<FeatureType>>, see Figure E.2.

NOTE 1 Neither ISO 19109 nor ISO 19118:2005, Annex A, distinguishes between feature types and object types — ISO 19109 only considers feature types while ISO 19118:2005, Annex A, classifies all feature types as object types. However, the distinction is meaningful in GML and in practice often required in application schemas. The distinction made in this annex is a conformant refinement of ISO 19118:2005, Annex A.

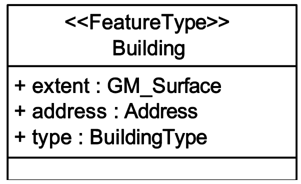


Figure 76 — A feature type <informative>

Object types shall be modelled as UML classes with no stereotype. Object types are types where the instances shall have an identity, but which are not feature types10).

EXAMPLE Examples of such types are geometries, topologies, reference systems. Instances of these types may have, for example, a name and an identifier.

UML classes with stereotype <<Type>> may have zero or more operations (these are not mapped to the GML application schema), attributes or associations.

The stereotype <<Abstract>> shall not be used in an Application Schema, because its use may be inconsistent with the use of correct UML notation, and thus misleading.

All instantiable subtypes of abstract types shall be either feature types, object types or data types.

Enumerations shall be modelled as UML classes with stereotype <<Enumeration>>.

Code lists shall be modelled as UML classes with stereotype <<CodeList>>, see Figure E.3.

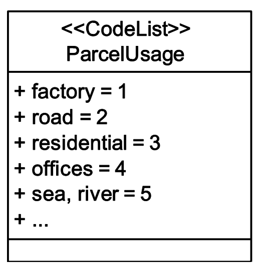
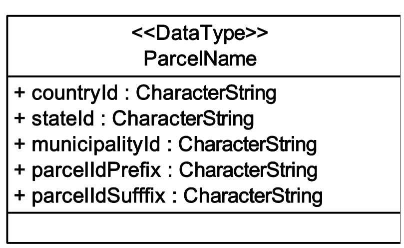


Figure 77 — A code list <informative>

Union types shall be modelled as UML classes with stereotype <<Union>> (as specified in ISO 19107).

All other data types shall be modelled as UML classes with stereotype <<DataType>>, see Figure E.4.



10) Object types are not considered explicitly in ISO 19109:2005. They appear only as value types of property types.

Figure 78 — A data type <informative>

UML classes of the ISO 19100 series of International Standards that are part of the GML profile and for which a GML base type has been provided in Table D.2 in the "GML type" column may be subclassed in the UML application schema. In the subclasses, additional properties may be added or properties of the subtype may be redefined with a restricted multiplicity or domain of values.

NOTE 2 Although redefinition of properties is supported, these redefined properties will be ignored in the conversion rules and it is the responsibility of the application to verify the constraints introduced by the redefinition. All classes with other stereotypes than those mentioned above may be part of the UML Application Schema, but will be ignored.

NOTE 3 When an Application Schema refers to types defined by other standards of the ISO 19100 series which are implemented by the GML schema, then the class names should match one of those listed in the first column of Table D.2.

A generalization relationship may be specified only between two classes that are either:

— both feature types,

— both object types, or

— both data types.

All generalization relationships between classes shall have no stereotype. All generalization relationships with other stereotypes will be ignored. The discriminator property of the UML generalization shall be blank.

If a class is a specialization of another class, then this class shall have only one supertype (no support for multiple inheritance).

All classes shall have a stereotype specifying the meaning of the class. Classes without a stereotype are treated as object types, see Figure E.5.

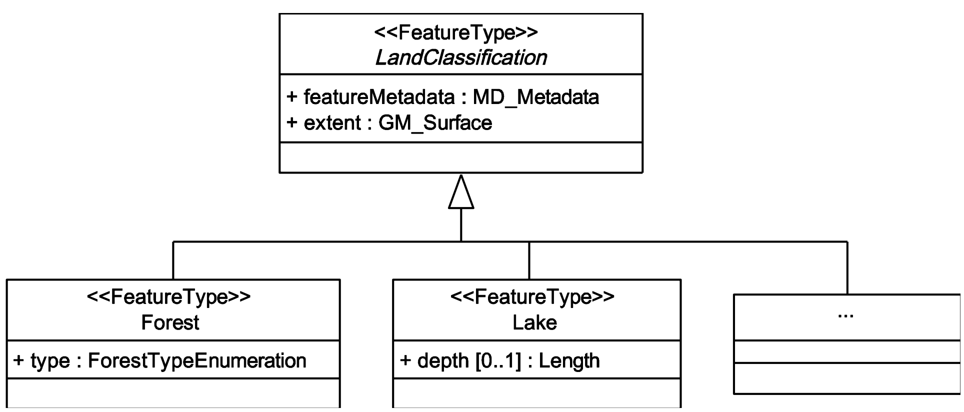


Figure 79 — Generalization relationship between feature types <informative>

E.2.1.1.3 Attributes

Every UML attribute of an abstract type, feature type, object type, data type or union type shall have a name and a type. The name shall be an "NCName" as defined by W3C XML Namespaces:1999. If its multiplicity is not “1”, the multiplicity shall be specified explicitly. An initial value may be specified for attributes with a number, string or enumeration type.

The type shall either be a predefined type (see E.2.1.1.5) or a class defined in the UML model.

Every UML attribute of an enumeration class shall have a name. The type information is left empty. No multiplicity, ordering or initial value information shall be attached to the attribute.

Every UML attribute of a code list class shall have a name. The type information is left empty. No multiplicity or ordering information shall be attached to the attribute. An initial value may be specified to document a code for the code list value. If it is omitted, the value (i.e. the attribute name) is used as the code.

The properties of a UML class are not ordered. To support the consistent ordering of the properties from the UML model in the conversion to XML Schema, a tagged value "sequenceNumber" (value domain: integer) shall be specified for every attribute. The value shall be unique for all attributes and association ends of a class.

E.2.1.1.4 Associations and association ends

Every UML association shall be an association with exactly two association ends. Both association ends shall connect to a feature, object or data type and shall have no stereotype or the stereotype <<association>> (otherwise the whole association will be ignored).

An association shall not contain any properties.

The rules for association ends are:

— If an association end is navigable it shall be marked as such and shall have a rolename. An association end with no name shall be ignored, even if it marked as navigable. If a name is provided, it shall be an "NCName" as defined by W3C XML Namespaces:1999.

— The multiplicity shall be given explicitly.

— The aggregation kind shall be specified explicitly if it is not "none".

— If the target class of an association end is a data type, then the aggregation kind shall be "composition".

Figure E.6 shows two example associations; one association is navigable in both directions and the other is an aggregation which is navigable in one direction only.

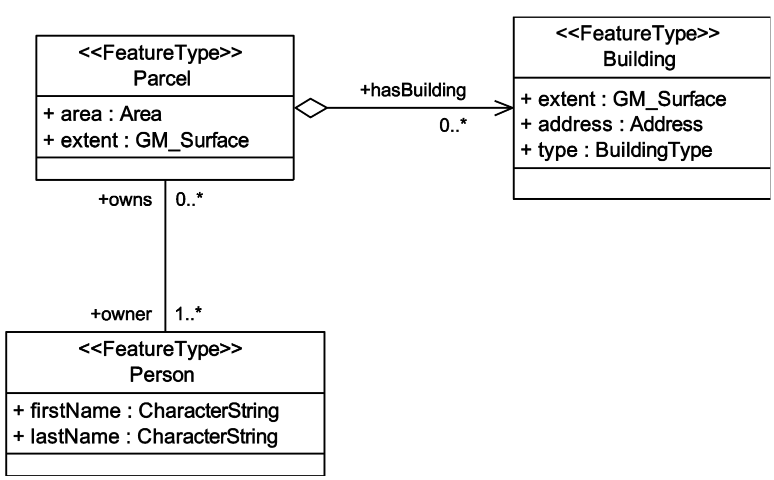


Figure 80 — Associations <informative>

The properties of a UML class are not ordered. To support the consistent ordering of the properties from the UML model in the conversion to XML Schema, a tagged value “sequenceNumber” (value domain: integer) shall be specified for every association end. The value shall be unique for all attributes and association ends of a class.

E.2.1.1.5 Predefined types

The predefined types from ISO/TS 19103 listed in E.2.4.4 are treated as “basic types” in the sense of ISO 19118:2005, Annex A (i.e. a canonical XML Encoding is attached to them).

E.2.1.1.6 OCL constraints

All OCL constraints are ignored. The assessment of the validity of the instance model with respect to these constraints is the task of the application processing the GML instances.

NOTE The Schematron language may be used to express OCL constraints as part of the XML Schema representing the GML application schema.

E.2.1.1.7 Other information

All other information in the UML Application Schema is not used in the encoding rules and is ignored.

E.2.1.2 Character repertoire and languages

“UTF-8” or “UTF-16” shall be used as the character encoding of the XML Schema files (with the associated character repertoire) in accordance with XML.

E.2.1.3 Exchange metadata

Exchange metadata may be specified for every feature or feature collection in a GML instance document by specifying in the application schema property elements whose content model is derived from "`gml:AbstractMetadataPropertyType`" as described in E.2.4.11 and E.2.4.13.

No specific schema for the exchange metadata is added to the GML application schema.

E.2.1.4 Dataset and object identification

Unique identifiers in accordance with XML's ID mechanism are used to identify objects.

NOTE The XML ID mechanism only requires that these identifiers are unique identifiers within the XML document in which they appear.

E.2.1.5 Update mechanism

No explicit update mechanism is defined for the features defined in the GML application schema. It is assumed that other mechanisms are used to update a data store.

NOTE An example is the “Transaction” operation of the OpenGIS® Web Feature Service Implementation Specification.

E.2.2 Input data structure

See ISO 19118:2005, A.3, for a description of the input data structure.

E.2.3 Output data structure

This encoding rule is based on the XML Recommendation 1.0 and the XML Linking Language (XLink) Version 1.0. The schema for the output data structure that governs the structure of the exchange format shall be a (set of) valid XML Schema(s) in accordance with XML Schema 1.0 and the Rules for Application Schemas (see Clause 21).

The XML Schema conversion rules are defined in the following Subclause.

E.2.4 Conversion rules

E.2.4.1 General concepts

The schema conversion rules define how XML Schema documents (XSDs) shall be derived from an application schema expressed in UML in accordance with ISO 19109. A number of general rules are defined in E.2.4 to describe the mapping from a UML model that follows the guidelines described in E.2.1.

NOTE In this annex the namespace "xsd:" is used to refer to the namespace of XML Schema, which is "http://www.w3.org/2001/XMLSchema". The namespace “gml:” refers to the namespace of GML, which is “http://www.opengis.net/gml/3.2”.

The rules are based on the GML model and syntax as described in Clauses 7 to 21 (especially Clauses 7, 9 and 21) and also on the encoding rules of ISO 19118:2005, Annex A.

The schema encoding rules are based on the general idea that the class definitions in the UML application schema are mapped to type and element declarations in XML Schema, so that the objects in the instance model can be mapped to corresponding element structures in the XML document.

Table E.1 gives an overview.

Table 20 — Schema encoding overview

Table: UML → GML application schema overview	
UML application schema	**GML application schema**
Package	One XML Schema document per package (default mapping)
<<Application Schema>>	XML Schema document
<<DataType>>	Global element, whose content model is a globally scoped XML Schema complexType, property type
<<Enumeration>>	Restriction of xsd:string with enumeration values
<<CodeList>>	Union of an enumeration and a pattern (default mapping, an alternative mapping is a reference to a dictionary)
<<Union>>	Choice group whose members are GML objects or features, or objects corresponding to DataTypes
<<FeatureType>>	Global element, whose content model is a globally scoped XML Schema type derived by direct/indirect extension of gml:AbstractFeatureType, property type
No stereotype or <<Type>>	Global element, whose content model is a globally scoped XML Schema type derived by direct/indirect extension of gml:AbstractGMLType, property type
Operations	Not encoded
Attribute	local xsd:element, the type is either a property type (if the type is a complex type) or a simple type.
Association role	local xsd:element, the type is always a property type (only named and navigable roles)
General OCL constraints	Not encoded

NOTE <<FeatureType>> is a new stereotype which does not appear in ISO/TS 19103 or ISO 19109, and is used to indicate that the type is a realization of GF_FeatureType and a specialization from AbstractFeature.

The multiplicity of attributes and association roles is mapped to “minOccurs” and “maxOccurs” attributes in <xsd:element> declarations. The detailed mapping rules are described below.

For different UML model elements, different tagged values are used to control the mapping from UML to XML Schema. The following Table E.2 provides a list of these tagged values.

Table 21 — Tagged values

UML model element	Associated tagged values
Package	— documentation — xsdDocument — targetNamespace (only <<Application Schema>>) — xmlns (only <<Application Schema>>) — version (only <<Application Schema>>) — gmlProfileSchema (only <<Application Schema>>)
Class	— documentation — noPropertyType — byValuePropertyType — isCollection — asDictionary (only <<CodeList>>) — xmlSchemaType (only <<Type>>)
Attribute and association end	— documentation — sequenceNumber — inlineOrByReference — isMetadata

E.2.4.2 UML packages

One XML Schema document is generated per package with the tagged value "xsdDocument" with the file name specified by the tagged value.

If the tagged value "xsdDocument" is set for a package, then the schema document contains all the XML Schema components resulting from the UML classes directly owned by the package. If the package is not a UML application schema, the schema document shall be included by the schema document that contains the schema components of the package that owns that package.

If the tagged value "xsdDocument" is not set for a package, all schema components are declared in the schema document that contains the schema components of the package that owns that package.

NOTE The tagged value is mandatory for all packages with the stereotype <<Application Schema>>, but optional for all other packages.

For every schema document, the "targetNamespace" and the "version" attributes of the root element shall be set in accordance with the tagged values of the same name in the package representing the UML Application Schema that owns the schema components within the schema document; if the "version" tagged value is not specified, the value "unknown" shall be used. In addition an "xmlns" attribute shall be specified for the target namespace with the value of the tagged value "xmlns" as the abbreviation.

EXAMPLE 1 “http://www.myorg.com/myns” may be a target namespace and “myns” may be the associated abbreviation used in the schema documents.

For every tagged value "gmlProfileSchema" of a package with the stereotype <<Application Schema>>, an element <gml:gmlProfileSchema> with the content of the tagged value shall be created in an appinfo annotation of the <schema> element as specified in 20.5.

The dependencies between the packages shall be used to determine the required imports of other schemas and additional includes of other schema documents:

— If the schema components specified by the target package of the dependency relationship are in the same target namespace as those of the supplier package, then the schema document specifying the schema components of the target package is "included".

— Otherwise the schema document representing the UML Application Schema package that contains the target package is "imported".

EXAMPLE 2 Mapping the information from Figure E.1 may result in:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.myorg.com/parcels" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:gp="http://www.myorg.com/geodeticPoints"
xmlns:pcl="http://www.myorg.com/parcels" xmlns:iso19115="http://www.isotc211.org/iso19115/"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified" version="2003-07-20">
    <include schemaLocation="Buildings.xsd"/>
    <import namespace="http://www.myorg.com/geodeticPoints" schemaLocation="GeodeticPoints.xsd"/>
    <import namespace="http://www.opengis.net/gml/3.2" schemaLocation="base/gml.xsd"/>
    <!-- … -->
</schema>
```

E.2.4.3 UML classes (general rules)

Recognized stereotypes for UML classes are: no stereotype, <<FeatureType>>, <<Type>>, <<DataType>>, <<Union>>, <<CodeList>>, and <<Enumeration>>. All classes will be mapped to the corresponding class category. All UML classes with other stereotypes will be ignored.

All UML classes shall have zero or one supertype.

All UML classes are mapped to named types. A suffix “Type” is added to the name of the type.

E.2.4.4 UML classes (basic types)

The basic types from the GML profile of ISO/TS 19103 listed in the left column of Table D.2 (starting with "CharacterString") are predefined and may be used as a data type of an attribute in an application schema conforming to ISO 19109. The mapping to a built-in type of XML Schema (“xsd:”) or GML (“gml:”) is specified. If multiple names are given in a cell of the table then the name in bold typeface shall be used as the default type of the mapping.

NOTE Multiple values in the right column are used to support also the reverse mapping in Annex F.

EXAMPLE ISO/TS 19103 Integer maps to “xsd:integer”.

If a class with the stereotype <<Type>> has a canonical XML Schema encoding (e.g. from XML Schema) the XML Schema typename corresponding to the data type shall be given as the value of the tagged value “xmlSchemaType”.

NOTE Canonical encodings may be preferred to structured encodings that follow the standard UML-to-GML encoding rules in some cases, for example where a compact structure based on “simpleContent” is already well known within the application domain.

E.2.4.5 UML classes (data types)

UML classes with stereotype <<DataType>> shall be mapped to XML Schema complex types.

NOTE Data types with other stereotypes, i.e. <<Enumeration>>, <<CodeList>> and <<Union>>, and predefined basic types are treated differently. See E.2.4.4, E.2.4.8, E.2.4.9, and E.2.4.10.

If the class has no supertype, it is a non-derived type in XML Schema; otherwise it extends its supertype which shall not be derived from `gml:AbstractGMLType` (directly or indirectly). Abstract superclasses without any attribute or navigable association role are ignored.

Global XML elements with appropriate settings for name (name of the UML class), type (name of the UML class plus “Type”), abstractness (if the class is abstract) and substitution groups (the qualified element name of the superclass or `gml:AbstractObject`, if the class has no superclass) shall be defined for these classes.

A named complex type shall be created for these classes (carrying the name of the class with a “PropertyType” suffix), if the class does not carry a tagged value "noPropertyType" with the value "true". The type follows the pattern for association properties as defined in GML (see 7.2.3), but without allowing Xlink attributes.

EXAMPLE The data type “ParcelName” from Figure E.4 may be mapped to:

```
<complexType name="ParcelNameType">
    <sequence>
        <element name="countryId" type="string"/>
        <element name="stateId" type="string"/>
        <element name="municipalityId" type="string"/>
        <element name="parcelIdPrefix" type="string"/>
        <element name="parcelIdSuffix" type="string" minOccurs=”0”/>
    </sequence>
</complexType>

<element name="ParcelName" type="ex:ParcelNameType" substitutionGroup="gml:AbstractObject"/>
<complexType name="ParcelNamePropertyType">
    <sequence>
        <element ref="ex:ParcelName"/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup" />
</complexType>
```

E.2.4.6 UML classes (feature types)

UML classes with stereotype <<FeatureType>> derive directly or indirectly from `gml:AbstractFeatureType`. If the class is a class without supertype, it extends directly `gml:AbstractFeatureType`; otherwise it extends its supertype which shall be derived from `gml:AbstractFeatureType` (again, directly or indirectly).

— Global XML elements with appropriate settings for name (name of the UML class), type (name of the UML class plus “Type”), abstractness (true, if the class is abstract) and substitution group (the name of the superclass or `gml:AbstractFeature`) are defined for these classes.

— If the class has a single association which is an aggregation or composition of a target class, the association role is converted to a property element, and the class carries a tagged value "isCollection" with the value "true", the attribute group `gml:AggregationAttributeGroup` is added to the complex type of the feature type.

— A named complex type shall be created for these classes (carrying the name of the class with a “PropertyType” suffix), if the class does not carry a tagged value "noPropertyType" with the value "true". The type follows the pattern for association properties as defined in GML (see 7.2.3).

— A named complex type shall be created for these classes (carrying the name of the class with a “PropertyByValueType” suffix), if the class carries a tagged value "byValuePropertyType" with the value "true". The type is a profile of the pattern for association properties as defined in GML restricted to the “by value” form (again, see 7.2.3).

EXAMPLE “Building” from Figure E.2 may be mapped to:

```
<complexType name="BuildingType">
	<complexContent>
		<extension base="gml:AbstractFeatureType">
			<sequence>
				<element name="extent" type="gml:SurfacePropertyType"/>
				<element name="address" type="pcl:AddressPropertyType"/>
				<element name="type" type="pcl:BuildingTypeType"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>

<complexType name="BuildingPropertyType">
	<sequence minOccurs="0">
		<element ref="pcl:Building"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
	<attributeGroup ref="gml:OwnershipAttributeGroup" />
</complexType>

<complexType name="BuildingPropertyByValueType">
	<sequence>
		<element ref="pcl:Building"/>
	</sequence>
	<attributeGroup ref="gml:OwnershipAttributeGroup" />
</complexType>

<element name="Building" type="pcl:BuildingType" substitutionGroup="gml:AbstractFeature"/>
```

E.2.4.7 UML classes (object types)

UML classes with no stereotype or stereotype <<Type>> derive directly or indirectly from `gml:AbstractGMLType`. If the class is a class without supertype it extends directly `gml:AbstractGMLType`, otherwise it extends its supertype which shall be derived from `gml:AbstractGMLType` (again, directly or indirectly), but not from `gml:AbstractFeatureType` (again, directly or indirectly).

— Global XML elements with appropriate settings for name (name of the UML class), type (name of the UML class plus “Type”), abstractness (true, if the class is abstract) and substitution group (the name of the supertype or “AbstractGML”) are defined for these classes.

— If the class has a single association which is an aggregation or composition of a target class, the association role is converted to a property element, and the class carries a tagged value "isCollection" with the value "true", the attribute group gml:AggregationAttributeGroup is added to the complex type of the object type.

— A named complex type shall be created for these classes (carrying the name of the class with a “PropertyType” suffix), if the class does not carry a tagged value "noPropertyType" with the value "true". The type follows the pattern for association properties as defined in GML (see 7.2.3).

— A named complex type shall be created for these classes (carrying the name of the class with a “PropertyByValueType” suffix), if the class carries a tagged value "byValuePropertyType" with the value "true". The type is a profile of the pattern for association properties as defined in GML restricted to the “by value” form (again, see 7.2.3).

EXAMPLE

```
<element name="Ellipse" type="ex:EllipseType" substitutionGroup="gml:AbstractCurveSegment"/>
```



```
<complexType name="EllipseType">
	<complexContent>
		<extension base="gml:AbstractCurveSegmentType">
		<sequence>
				<element name="center" type="gml:DirectPositionType"/>
				<element name="semiminor" type="gml:VectorType"/>
				<element name="semimajor" type="gml:VectorType"/>
		</sequence>
		</extension>
	</complexContent>
</complexType>
```

E.2.4.8 UML classes (enumerations)

UML classes with stereotype <<Enumeration>> are mapped to XML Schema simple types. The base type is “string”, the domain of values is restricted to the set of literal values as specified by the attribute names of the UML class.

EXAMPLE

```
<simpleType name="SignType">
	<restriction base="string">
		<enumeration value="-"/>
		<enumeration value="+"/>
	</restriction>
</simpleType>
```

E.2.4.9 UML classes (code lists)

A UML class with stereotype <<CodeList>> and without a tagged value "asDictionary" with the value "true" shall be mapped like an enumeration, but with the following differences:

— A facet "<pattern value=’other: \w{2,}’/>" shall be added that allows for any text value beside the predefined values; these free values are prefixed with “other: ”.

— If a code is specified for a code list value, only the code shall be represented as an enumeration facet.

— An encoded code value shall be qualified with an appinfo annotation with a `gml:description` element specifying the text value of the enumerated value.

EXAMPLE 1 The code list “ParcelUsage” from Figure E.3 may be represented as:

```
<simpleType name="ParcelUsageType">
	<union memberTypes="pcl:ParcelUsageEnumerationType pcl: ParcelUsageOtherType"/>
</simpleType>

<simpleType name="ParcelUsageEnumerationType">
	<restriction base="string">
		<enumeration value="1">
			<annotation>
				<appinfo><gml:description>factory</gml:description></appinfo>
			</annotation>
		</enumeration>
		<enumeration value="2">
			<annotation>
				<appinfo><gml:description>road</gml:description></appinfo>
```

```
                </annotation>
            </enumeration>
            <enumeration value="3">
                <annotation>
                    <appinfo><gml:description>residential</gml:description></appinfo>
                </annotation>
            </enumeration>
            <enumeration value="4">
                <annotation>
                    <appinfo><gml:description>offices</gml:description></appinfo>
                </annotation>
            </enumeration>
            <enumeration value="5">
                <annotation>
                    <appinfo><gml:description>sea, river</gml:description></appinfo>
                </annotation>
            </enumeration>
        </restriction>
</simpleType>

<simpleType name="ParcelUsageOtherType">
        <restriction base="string">
            <pattern value="other: \w{2,}"/>
        </restriction>
</simpleType>
```

Alternatively, if the class carries a tagged value "asDictionary" with the value "true", a `gml:Dictionary` shall be used to represent a code list.

EXAMPLE 2 The code list “ParcelUsage” from Figure E.3 may be represented in a GML dictionary document as:

```
<gml:Dictionary gml:id="CodeList" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/gml/3.2 gml.xsd">
        <gml:identifier codeSpace="http://www.someorg.de/cl.xml">My code lists</gml:identifier>
        <gml:dictionaryEntry>
            <gml:Dictionary gml:id="ParcelUsage">
                <gml:identifier codeSpace="http://www.someorg.de/cl.xml">ParcelUsage</gml:identifier>
                <gml:dictionaryEntry>
                    <gml:Definition gml:id="ParcelUsage_1">
                        <gml:description>factory</gml:description>
                        <gml:identifier codeSpace="http://www.someorg.de/cl.xml#ParcelUsage">1</gml:identifier>
                    </gml:Definition>
                </gml:dictionaryEntry>
                <gml:dictionaryEntry>
                    <gml:Definition gml:id="ParcelUsage_2">
                        <gml:description>road</gml:description>
                        <gml:identifier codeSpace="http://www.someorg.de/cl.xml#ParcelUsage">2</gml:identifier>
                    </gml:Definition>
                </gml:dictionaryEntry>
                <gml:dictionaryEntry>
                    <gml:Definition gml:id="ParcelUsage_3">
                        <gml:description>residential</gml:description>
                        <gml:identifier codeSpace="http://www.someorg.de/cl.xml#ParcelUsage">3</gml:identifier>
                    </gml:Definition>
                </gml:dictionaryEntry>
                <gml:dictionaryEntry>
                    <gml:Definition gml:id="ParcelUsage_4">
                        <gml:description>offices</gml:description>
                        <gml:identifier codeSpace="http://www.someorg.de/cl.xml#ParcelUsage">4</gml:identifier>
                    </gml:Definition>
                </gml:dictionaryEntry>
```



```
            <gml:dictionaryEntry>
                <gml:Definition gml:id="ParcelUsage_5">
                    <gml:description>sea, river</gml:description>
                    <gml:identifier codeSpace="http://www.someorg.de/cl.xml#ParcelUsage">5</gml:identifier>
                </gml:Definition>
            </gml:dictionaryEntry>
        </gml:Dictionary>
    </gml:dictionaryEntry>
</gml:Dictionary>
```

In an instance document the reference would then be encoded (using `gml:CodeType` as the content model, see E.2.4.11) for example as:

```
<usage codeSpace="http://www.someorg.de/example/cl.xml#ParcelUsage">1</usage>
```

The codeSpace attribute points to the dictionary, the value is the name of the entry in that dictionary.

The way a code list is encoded in a GML application schema also determines how property elements that carry the code lists as its value domain shall be encoded; see E.2.4.11.

E.2.4.10 UML classes (unions)

UML classes with stereotype <<Union>> are mapped as XML Schema complex types. These classes are mapped like data types (see E.2.4.5), but instead of a <xsd:sequence> of the properties, a <xsd:choice> is used so that exactly one of the properties is specified in an instance of a union.

EXAMPLE

```
<complexType name="RemoteResourceType">
    <choice>
        < element name="name" type="string"/>
        < element name="uri" type="anyURI"/>
    </choice>
</complexType>
```

E.2.4.11 UML attributes and association roles

A UML attribute or association role of an object or feature type is mapped to a local element with the same name in the complex type defining the content model of the object or feature type. The minOccurs and maxOccurs attributes are set in accordance with the definitions in the UML model (see ISO 19118:2005, Annex A, for details of the mapping). The type depends on the type of the value of the property in UML:

If the type of the value of the property is of simple content, then the type is used directly.

EXAMPLE 1 `<element name="count" type="integer"/>`

If the type of the value of the property is of complex content, then a property type shall be used. The default encoding of the property type allows both the inline or by-reference representation for feature and object types and the inline representation for data and union types. For feature and object types the representation may be restricted to inline or by-reference using a tagged value “inlineOrByReference” with the values “inline” or “byReference” respectively. If the tagged value is missing or its value is “inlineOrByReference” the default encoding shall be used.

If an attribute or association role is a metadata property, then the property type shall extend `gml:AbstractMetadataPropertyType` (see 7.2.6); a metadata property is a property with the tagged value "isMetadata" with the value "true" or whose value is a class defined by ISO 19115:2003. If an association role is the target end of an aggregation or composition, then the property type shall extend `gml:AbstractMemberType` (see 7.2.5.1) unless it is a metadata property. If an association role is the target end of a composition or an object-valued attribute, then the property element shall add a Schematron constraint that asserts that the `owns` attribute of the `gml:OwnershipAttributeGroup` is “true”. The Schematron constraint shall follow the following pattern:

```
<sch:pattern>
    <sch:rule context="qualified name of the object element">
        <sch:report test="qualified property name/@owns='true'">This property is a composition, values must be
owned</sch:report>
    </sch:rule>
</sch:pattern>
```

EXAMPLE 2 For a property ex:representativeLocation of a feature type ex:MyFeature that controls the point object describing the location this could be described as follows:

```
<element name="representativeLocation" type="gml:PointPropertyType">
    <annotation>
        <appinfo>
            <sch:pattern">
                <sch:rule context="ex:MyFeature">
                    <sch:report test="ex:representativeLocation/@owns='true'">This property is a composition, values
                    must be owned</sch:report>
                </sch:rule>
            </sch:pattern>
        </appinfo>
    </annotation>
</element>
```

If the property type is already specified in its application schema as a named type (this can be detected by inspecting the tagged values "noPropertyType" and "byValuePropertyType"), this schema component shall be referenced; otherwise, an anonymous property type shall be defined locally in the property element.

If the encoded property is an association end and the other association end of the association is also encoded in the GML application schema, the property name of the other association end shall be encoded in a `gml:reversePropertyName` element in an appinfo annotation of the property element (see 7.2.3.9).

EXAMPLE 3 By-reference or inline:

```
<element name="owner" type="ex:PersonPropertyType" minOccurs=”0”>
    <annotation>
        <appinfo>
                <gml:reversePropertyName>ex:owns</gml:reversePropertyName>
            </appinfo>
    </annotation>
</element>
…
<complexType name="PersonPropertyType">
    <sequence minOccurs="0">
        <element ref="ex:Person"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
    <attributeGroup ref="gml:OwnershipAttributeGroup" />
</complexType>
```

or

```
<element name="owner" minOccurs=”0”>
    <annotation>
        <appinfo>
            <gml:reversePropertyName>ex:owns</gml:reversePropertyName>
        </appinfo>
```



```
        </annotation>
        <complexType>
                <sequence minOccurs="0">
                        <element ref="ex:Person"/>
                </sequence>
                <attributeGroup ref="gml:AssociationAttributeGroup"/>
        </complexType>
</element>
```

Alternatively, the property type may support only one of the representations, inline or by-reference, depending on the tagged value "inlineOrByReference".

EXAMPLE 4 inline only:

```
<element name="owner" type="ex:PersonPropertyByValueType" minOccurs="0"/>
…
<complexType name="PersonPropertyByValueType">
        <sequence>
                <element ref="ex:Person"/>
        </sequence>
</complexType>
```

or

```
<element name="owner" minOccurs="0">
        <complexType>
                <sequence>
                        <element ref="ex:Person"/>
                </sequence>
        </complexType>
</element>
```

If only the by-reference representation is to be supported, then the property element shall be qualified with an appinfo annotation element `gml:targetElement` specifying the qualified element name of the target type.

```
        <element name="targetElement" type="string"/>
```

If the encoded property is an association end and the other association end of the association is also encoded in the GML application schema, the property name of the other association end shall be encoded in another appinfo annotation element `gml:reversePropertyName` specified above.

EXAMPLE 5 By-reference only:

```
<element name="owner" type="gml:ReferenceType" minOccurs="0">
        <annotation>
                <appinfo>
                        <gml:targetElement>ex:Person</gml:targetElement>
                        <gml:reversePropertyName>ex:owns</gml:reversePropertyName>
                </appinfo>
        </annotation>
</element>
```

Depending on the encoding of the class, a UML attribute of a code list or enumeration type is mapped to an element with either a string value (value domain: values of the enumeration or code list) or a value referencing the corresponding dictionary entry. In an instance, the dictionary may be explicitly referenced using the `codeSpace` attribute. A default value for the URI representing the dictionary may be provided using an appinfo annotation element `gml:defaultCodeSpace`.

```
        <element name="defaultCodeSpace" type="anyURI"/>
```


EXAMPLE 6 The code list “BuildingType” may be represented as:

```
<element name="type" type="ex:BuildingTypeType"/>
```

or

```
<element name="type" type="gml:CodeType">
	<annotation>
		<appinfo>
			<gml:defaultCodeSpace>http://www.someorg.de/example/cl.xml#BuildingType</gml:defaultCodeSpace>
		</appinfo>
	</annotation>
</element>
```

If a UML attribute or UML association role is redefined (i.e. a subclass contains an attribute or association role with the same name as in a supertype) then this property is not part of the content model of the subtype. It is the responsibility of an application to assert the compliance of instances with such constraints expressed in the conceptual model.

All attributes and association roles of a class shall be converted in the ascending sort order of the tagged value “sequenceNumber”.

E.2.4.12 Documentation

Tagged values “documentation” from elements in the UML model are mapped to annotation/documentation elements in the XML Schema files.

EXAMPLE

```
	<element name="curveProperty" type="gml:CurvePropertyType">
		<annotation>
			<documentation>This property element either references a curve via the XLink-attributes or contains the curve
element. curveProperty is the predefined property which can be used by GML application schemas whenever a GML feature
has a property with a value that is substitutable for AbstractCurve.</documentation>
		</annotation>
	</element>
```

E.2.4.13 Classes imported from the ISO 19100 series of International Standards

In addition to the rules defined above, the following rules apply when the UML Application Schema imports classes from the ISO 19100 series of International Standards.

Classes from the ISO 19100 series of International Standards that are implemented by the GML schema shall be recognized. The use of classes from the ISO 19100 series of International Standards shall be conformant with ISO 19109. The mapping of the relevant classes from the ISO 19100 series of International Standards is shown in Table D.2.

If a class from ISO 19115 and implemented in ISO/TS 19139 is used as the type of a property, then an anonymous property type extending gml:AbstractMetadataPropertyType shall be defined. The encapsulated object element is the corresponding object element for the metadata type as specified by ISO/TS 19139.

E.2.4.14 Classes imported from other conceptual models with a predefined XML encoding

In addition to the rules defined above, the following rules apply when the UML Application Schema imports classes from another UML model for which a standard XML encoding has already been specified.

Extensions to Table D.2 for the imported classes shall be specified. The table shall be distributed together with the application schema in UML.

The mapping of the relevant classes from the imported model to XML Schema is normatively specified by this table.

E.3 Example <informative>

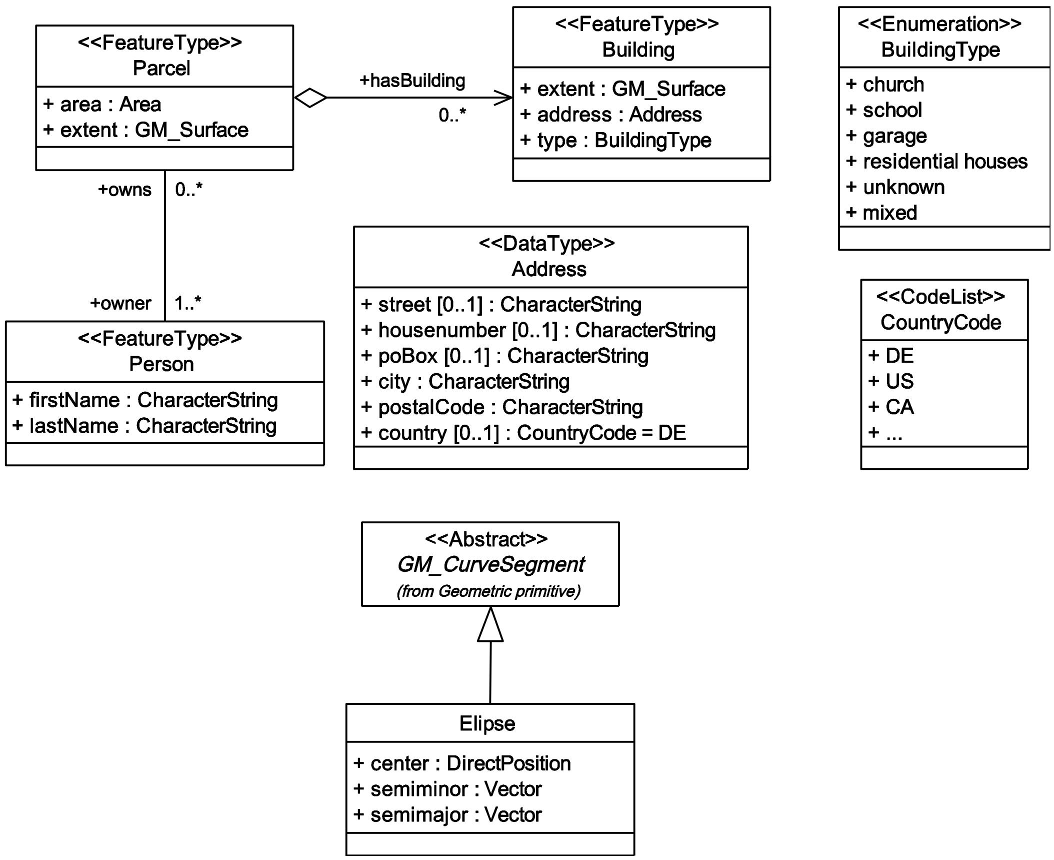


Figure 81 — Example application schema

The application schema shown in Figure E.7 may be encoded as

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.someorg.de/example" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:ex="http://www.someorg.de/example" xmlns:gml="http://www.opengis.net/gml/3.2" elementFormDefault="qualified"
version="1.0">
	<!-- ================================== -->
	<import namespace="http://www.opengis.net/gml/3.2" schemaLocation="./gml.xsd"/>
	<import namespace="http://www.w3.org/1999/xlink" schemaLocation="./xlinks.xsd"/>
	<!-- ================================== -->
	<element name="Parcel" substitutionGroup="gml:AbstractFeature">
		<complexType>
			<complexContent>
				<extension base="gml:AbstractFeatureType">
					<sequence>
						<element name="area" type="gml:AreaType"/>
						<element name="extent" type="gml:SurfacePropertyType"/>
```



```
                        <element name="owner" type="ex:PersonPropertyType" maxOccurs="unbounded">
                            <annotation>
                                <appinfo><gml:reverseProperty>ex:owns</gml:reverseProperty></appinfo>
                            </annotation>
                        </element>
                        <element name="hasBuilding" type="ex:BuildingPropertyType" minOccurs="0"
                            maxOccurs="unbounded"/>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>
    <complexType name="ParcelPropertyType">
        <sequence minOccurs="0">
            <element ref="ex:Parcel"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup" />
    </complexType>
    <!-- ================================== -->
    <element name="Building" substitutionGroup="gml:AbstractFeature">
        <complexType>
            <complexContent>
                <extension base="gml:AbstractFeatureType">
                    <sequence>
                        <element name="extent" type="gml:SurfacePropertyType"/>
                        <element name="address">
                            <complexType>
                                <sequence>
                                    <element name="Address" type="ex:AddressType"/>
                                </sequence>
                            </complexType>
                        </element>
                        <element name="type" type="ex:BuildingTypeType"/>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>
    <complexType name="BuildingPropertyType">
        <sequence minOccurs="0">
            <element ref="ex:Building"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup" />
    </complexType>
    <!-- ================================== -->
    <element name="Person" substitutionGroup="gml:AbstractFeature">
        <complexType>
            <complexContent>
                <extension base="gml:AbstractFeatureType">
                    <sequence>
                        <element name="firstName" type="string"/>
                        <element name="lastName" type="string"/>
                        <element name="owns" type="ex:ParcelPropertyType" minOccurs="0"
                        maxOccurs="unbounded">
                            <annotation>
                                <appinfo><gml:reverseProperty>ex:owner</gml:reverseProperty></appinfo>
                            </annotation>
                        </element>
                    </sequence>
                </extension>
```

```
            </complexContent>
        </complexType>
    </element>
    <complexType name="PersonPropertyType">
        <sequence minOccurs="0">
            <element ref="ex:Person"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup" />
    </complexType>
    <!-- ================================== -->
    <complexType name="AddressType">
        <sequence>
            <element name="street" type="string" minOccurs="0"/>
            <element name="housenumber" type="string" minOccurs="0"/>
            <element name="poBox" type="string" minOccurs="0"/>
            <element name="city" type="string"/>
            <element name="postalCode" type="string"/>
            <element name="country" type="ex:CountryCodeType" minOccurs="0" default="DE"/>
        </sequence>
    </complexType>
    <!-- ================================== -->
    <simpleType name="BuildingTypeType">
        <restriction base="string">
            <enumeration value="church"/>
            <enumeration value="school"/>
            <enumeration value="garage"/>
            <enumeration value="residential houses"/>
            <enumeration value="unknown"/>
            <enumeration value="mixed"/>
        </restriction>
    </simpleType>
    <!-- ================================== -->
    <simpleType name="CountryCodeType">
        <union memberTypes="ex:CountryCodeEnumerationType ex:CountryCodeOtherType"/>
    </simpleType>
    <simpleType name="CountryCodeEnumerationType">
        <restriction base="string">
            <enumeration value="DE"/>
            <enumeration value="US"/>
            <enumeration value="CA"/>
            <enumeration value="..."/>
        </restriction>
    </simpleType>
    <simpleType name="CountryCodeOtherType">
        <restriction base="string">
            <pattern value="other: \w{2,}"/>
        </restriction>
    </simpleType>
    <!-- ================================== -->
    <element name="Ellipse" type="ex:EllipseType" substitutionGroup="gml:AbstractCurveSegment"/>
    <complexType name="EllipseType">
        <complexContent>
            <extension base="gml:AbstractCurveSegmentType">
                <sequence>
                    <element name="center" type="gml:DirectPositionType"/>
                    <element name="semiminor" type="gml:VectorType"/>
                    <element name="semimajor" type="gml:VectorType"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

</schema>

Annex F
(normative)

GML-to-UML application schema encoding rules

F.1 General concepts

The mapping from a GML application schema to an ISO 19109 conformant application schema in UML is based on a set of encoding rules. These encoding rules are conformant with the rules for GML application schemas as described in Clauses 7 to 21, especially Clauses 7, 9 and 21.

The rules listed in F.2 aim at an automatic mapping from a GML application schema to an ISO 19109 and ISO/TS 19103 conformant UML application schema.

These rules do not prescribe that all GML application schemas shall be generated to fulfil the encoding requirements documented in this annex. All schemas following the rules defined in Clause 21 are valid and conformant GML application schemas.

This annex shall be used if there is a requirement in the application domain to derive an ISO 19109 conformant Application Schema in UML from a GML application schema.

The XML namespace abbreviation "xsd" is used to refer to the namespace of XML Schema, which is "http://www.w3.org/2001/XMLSchema".

The XML namespace abbreviation “gml” refers to the XML namespace of GML, which is “http://www.opengis.net/gml/3.2”.

In addition, GML imports definitions from the following namespaces:

The XML namespace abbreviation “xlink” refers to the XML namespace for xlinks, which is “http://www.w3.org/1999/xlink”.

The term “GML namespaces” is used below to refer to the namespaces “gml” and “xlink”.

F.2 Encoding rules

F.2.1 General encoding requirements

F.2.1.1 General remarks

The schema encoding rules are based on the general idea that the corresponding type and element declarations in XML Schema are mapped to class definitions in the UML application schema, so that element structures in the XML document can be mapped to the objects in the instance model.

F.2.1.2 GML schema

F.2.1.2.1 General

To be a valid input into the mapping, the GML application schema shall meet the requirements of the relevant conformance classes in 2.2, at least "All GML application schemas", "GML application schema to be converted to an ISO 19109 Application Schemas in UML" and "GML application schemas defining Features and Feature Collections".

The GML application schema shall have and contain definitions for only one target namespace.

The GML application schema may import definitions from XML namespaces other than its target namespace.

A GML application schema consists of a set of one or more XML schema documents such that:

— the documents have unique names;

— the documents contain xsd:include elements for other schema documents with the same target namespace;

— one top-level schema document for the GML application schema target namespace is not included by any other schema documents for the target namespace, but directly or indirectly includes all other schema documents for the target namespace, if any;

— the schema documents contain xsd:import elements for XML namespaces other than the target namespace, and for schema documents that contain definitions in those XML namespaces;

— all included and imported schema documents are accessible via the URI specified by the schemaLocation attribute on the xsd:include and xsd:import elements that reference them;

— a validating XML parser resolves all of the dependencies among the definitions contained in the set of schema documents;

— a validating XML parser validates the set of schema documents without error;

— a validating XML parser validates an XML instance document containing elements and attributes that represent all of the definitions from the target namespace of the GML application schema without error.

Documentation of the definitions contained in a GML application schema shall be stored in nested xsd:annotation and xsd:documentation elements within the schema definition elements.

The version of a GML application schema, if applicable, shall be contained in the version attribute of the xsd:schema element from the top-level schema for its target namespace.

All global type and element names within a GML application schema shall be unique.

The GML application schema shall not define any elements with anonymous types for objects.

The GML application schema shall not define any XML attributes or named groups.

Every complex type in a GML application schema shall either be a GML object type, a GML feature type, a GML data type or a GML property type.

Complex types with simple content shall not be defined in the GML application schema.

The name of all types defined in a GML application schema shall end with the suffix “Type”.

A suffix “RestrictionType” in the name of a complex type shall only be used for an abstract type that derives by restriction and which is a the base type of exactly one complex type that derives from this type by extension and has the same name as the restricted type except that “RestrictionType” is replaced by “Type”.

A suffix “PropertyType” in the name of a complex type shall only be used for an instantiable type that follows the pattern for by-reference-or-value property types of GML. A complex type (GML object type or GML feature type) with the same name shall exist that has “PropertyType” replaced by “Type”.

A suffix “PropertyByValueType” in the name of a complex type shall only be used for an instantiable type that follows the pattern for by- value property types of GML. A complex type (GML data type, GML object type or GML feature type) with the same name shall exist that has “PropertyByValueType” is replaced by “Type”.

NOTE These rules severely restrict the possible forms of GML application schemas.

F.2.1.2.2 GML object types including GML feature types

Each GML object type defined in a GML application schema shall have a content model that directly or indirectly derives from `gml:AbstractGMLType`.

Each GML object type of a particular kind defined in a GML application schema shall derive from the most specialized GML object type from the “http://www.opengis.net/gml/3.2” namespace of a similar kind (with matching semantics) that could possibly be used to define its content model. So GML object types defined in a GML application schema to represent geographic features (GML feature types) shall derive from `gml:AbstractFeatureType` instead of from `gml:AbstractGMLType`, GML object types defined in a GML application schema to represent geometric points shall derive from `gml:PointType` instead of from `gml:AbstractGeometryType`, etc.

GML object types defined in the GML application schema that derive from GML object types outside of the target namespace shall derive directly only from one of the GML object types listed in the third column of Table D.2 where there first column in the same row provides a class name of a class defined by the ISO 19100 series of International Standards or gml:AbstractGMLType or gml:AbstractFeatureType.

The schema definitions of abstract GML object types shall contain the attribute “abstract” with the value “true”.

The name of abstract GML object types shall begin with the prefix “Abstract”.

The schema definitions of GML object types for which no subtypes may be defined shall contain the attribute “final” with the value “all”.

The properties of the GML object type shall be specified in an xsd:sequence element.

F.2.1.2.3 Global elements for gml object types

One global XML element shall be defined for every GML object type defined in a GML application schema.

The name of this element shall be the name of the GML object type without the “Type”-suffix.

The element shall have a substitutionGroup attribute whose value is the name of a global XML element whose type is the base type of the GML object type.

F.2.1.2.4 Default property types for gml object types

A default GML property type may be defined in a GML application schema for every GML object type defined in that GML application schema.

The GML property type shall either use or inherit directly or indirectly from one of the property types specified in 7.2.3 or it shall be defined in accordance with the patterns specified in this subclause.

The name of this property type shall be the name of the GML object type with the “Type”-suffix replaced by “PropertyType”.

If no default property type is specified for a GML object type, an application schema shall use `gml:ReferenceType` as the default property type of the GML object type.

F.2.1.2.5 Inline property types for gml object types

A default GML property type for inline properties may be defined in a GML application schema for every GML object type defined in that GML application schema.

The GML property type shall either inherit directly or indirectly from `gml:InlinePropertyType`, or it shall be defined in accordance with the patterns specified in 7.2.3.8. The use of the `gml:AsscociationAttributeGroup` is prohibited in such properties.

The name of this property type shall be the name of the GML object type with the “Type”-suffix replaced by “PropertyByValueType”.

If no default property type for inline properties is specified for a GML object type, an application schema shall use `gml:AssociationRoleType` as the default property type for inline properties of the GML object type.

F.2.1.2.6 GML data types including GML union types

A complex type defined in a GML application schema that does not directly or indirectly derive from `gml:AbstractGMLType` is called a GML data type.

The properties of the GML data type shall take one of the following forms:

— The properties of the complex type as well as the properties of all of its base types are specified in an xsd:sequence element with minOccurs and maxOccurs values of “1”.

— The GML data type is not derived from any base type. In this case, the properties may be specified in either a single xsd:sequence element with minOccurs and maxOccurs values of “1” or a single xsd:choice element with minOccurs and maxOccurs values of “1”.

The content model of the complex type shall not include a `gml:id` attribute.

F.2.1.2.7 Default property types for GML data types

A default GML property type for inline properties may be defined in a GML application schema for every GML data type defined in that GML application schema.

The GML property type shall either inherit directly or indirectly from `gml:InlinePropertyType`, or it shall be defined in accordance with the patterns specified in 7.2.3.8. The use of the `gml:AsscociationAttributeGroup` is prohibited in such properties.

The name of this property type shall be the name of the GML data type with the “Type”-suffix replaced by “PropertyByValueType”.

If no default property type for inline properties is specified for a GML data type, an application schema shall use `gml:AssociationRoleType` as the default property type for inline properties of the GML data type.

F.2.1.2.8 Enumerations

A simple type defined in a GML application schema that is a restriction of xsd :string using only the xsd:enumeration facet is called an enumeration.

F.2.1.2.9 Code lists

A simple type defined in a GML application schema that is a union of an enumeration and a simple type that is a restriction of xsd:string using only one xsd:pattern facet with the value “other: \w{2,}” is called a code list.

Enumeration values may be qualified with an appInfo annotation (element `gml:codeListValue`) specifying that the enumeration value is the code value of another enumeration value; the associated enumeration value is given as the text value of the `gml:codeListValue` element.

F.2.1.2.10 Global elements for GML data types, enumerations and code lists

No global XML element shall be defined for enumerations or code lists defined in a GML application schema.

F.2.1.2.11 Predefined basic types

The simple types from the XML Schema and GML namespace listed in the fourth column of Table D.2 may be used in the GML application schema. No other simple types from these namespaces shall be used in a GML application schema.

F.2.1.2.12 GML properties

Every property of a GML object or feature type (except properties defined in the GML namespace) or of a GML data or union type shall be represented by a single, locally defined xsd:element. Locally defined means that the name and type of the element shall be given explicitly in the element declaration (no references to global XML elements). The element may carry minOccurs and maxOccurs values. The name of this element shall be the name of the property; the type shall be either a simple type or a property type.

F.2.1.2.13 Schematron constraints

All Schematron constraints are ignored.

F.2.1.2.14 Imported elements and types from other XML namespaces

If other XML Schema components are imported from other namespaces than XML Schema and GML, define the relevant entries as extensions to Table D.2.

F.2.1.2.15 Other information

All other information in the GML application schema is not used in the encoding rules and is ignored.

F.2.1.3 Character repertoire and languages

The character encoding used for the schemas determines the available character repertoire.

F.2.1.4 Exchange metadata

Exchange metadata may be specified for every Feature or Feature Collection in a GML instance document[11)]. No specific schema for the exchange metadata is added to the GML application schema.

F.2.1.5 Dataset and object identification

Unique `gml:id` identifiers in accordance with 7.2.4.5 and XML's ID mechanism shall be used to identify GML objects.

F.2.1.6 Update mechanism

No explicit update mechanism shall be defined for the feature types defined in a GML application schema. It is assumed that other mechanisms are used to update an instance model data store.

F.2.1.7 Input data structure

The schema for the input data structure is defined by the XML Schema 1.0 Part 1: Structures, Part 2: Datatypes W3C Recommendations, and the Rules for GML application schemas (see Clause 21).

F.2.2 Output data structure

See ISO 19118:2005, A.3, for a description of the output data structure.

F.2.3 Conversion rules

F.2.3.1 General concepts

The schema conversion rules defined in the following subclauses describe the mapping from a GML application schema that follows the guidelines described in F.2.1 to a UML application schema that conforms to the rules defined in ISO 19109 and ISO/TS 19103, using the encoding rules of ISO 19118:2005, Annex A, and in particular the generic instance model described in A.3. These rules are also based on the current rules for the GML model and syntax as described in Clauses 7 to 21 (especially Clause 7).

The schema conversion rules map definitions from a (set of) valid GML application schema documents (XSDs) to a set of UML packages. A top-level package with the stereotype <<Application Schema>> is created to contain all the other packages in this set. By default, one package is created in this set for each XSD in the GML application schema, including those directly or indirectly imported from XML namespaces other than the target namespace for the GML application schema, except for XSDs for the GML namespaces. The top-level package owns directly or indirectly all UML model elements mapped from object types in the GML application schema.

The declarations of the GML application schema may be arranged in a different package structure as long as the top-level package keeps its name and stereotype and all the model elements still belong directly or indirectly to this package.

The type and element declarations in the GML application schema are mapped to class definitions in the UML application schema, so that element structures in the GML XML document can be mapped to corresponding objects in the instance model.

---

11) By using the property elements whose content model has been derived from `gml:AbstractMetadataPropertyType` and, for example, the ISO/TS 19139 XML Schema encoding of ISO 19115:2003.

The UML model shall contain within a package with the name "ISO 19100" the applicable normative packages of the ISO 19100 series of International Standards or a strict profile of this model.

The UML model shall contain the UML package of all other GML application schemas imported by the GML application schema.

Table F.1 gives an overview; full details of the mapping are specified in the subsequent subclauses.

Table 22 — Schema encoding overview

Table: GML → UML Application Schema Overview	
GML application schema	**UML application schema**
GML application schema	Package <<ApplicationSchema>>
GML schema document {name} XSD	Package named {name}
Object and property type and global element for any object type that is a direct or indirect extension of `gml:AbstractFeatureType`	Class with stereotype <<FeatureType>>
Object and property type and global element for any object type that is a direct or indirect extension of `gml:AbstractGMLType`, other than those that extend `gml:AbstractFeatureType`	Class with no stereotype
Data and property type and global element for any object type that is not a direct or indirect extension of `gml:AbstractGMLType` and whose content model is a sequence of properties	Class with stereotype <<DataType>>
Restriction of `xsd:string` with enumeration values	Class with stereotype <<Enumeration>>
Union of an enumeration and a pattern	Class with stereotype <<CodeList>>
Data and property type and global element for any object type that is not a direct or indirect extension of `gml:AbstractGMLType` and whose content model is a choice of properties	Class with stereotype <<Union>>
Local xsd:element of a simpleType or a complexType with simpleContent or a type that does not directly or indirectly inherit from `gml:AbstractGMLType`	UML Attribute
Local xsd:element of a type that contains `gml:AssociationAttributeGroup`	UML Association Role
Schematron constraints	Not encoded

The multiplicity of attributes and association roles is derived from the minOccurs and maxOccurs attributes in local xsd:element declarations.

F.2.3.2 GML schema documents

A top-level package with the stereotype <<Application Schema>> is created to contain all the other packages generated for the GML application schema.

— The “targetNamespace” and “xmlns” tagged values are applied to the <<ApplicationSchema>> package with corresponding values for the target namespace of the GML application schema

EXAMPLE “http://www.myorg.com/myns” and “myns”.

— The “version” tagged value is applied to the <<ApplicationSchema>> package with the default value of “1.0”. If the “version” attribute of the xsd:schema element of the top-level schema document for the GML application schema exists and contains a non-empty value, its value replaces the default tagged value.

— The "xsdDocument" tagged value is set to the relative filename of the XML Schema document.

By default, one UML package is generated for each input schema document in the GML application schema, including those directly or indirectly imported from XML namespaces other than the target namespace of the GML application schema — except for XML Schema documents from the GML namespaces. Alternatively, a single XML Schema document may also be split into several UML packages.

The packages are generated in the <<ApplicationSchema>> package for the GML application schema with names that correspond to the names of the input schema documents.

The xsd:include and xsd:import statements in each input schema document are used to determine and set the dependencies of the packages generated in the <<Application Schema>> package.

F.2.3.3 GML object types

Every GML object type shall be mapped to a UML class.

If the object type directly or indirectly derives from `gml:AbstractFeatureType`, the stereotype of the class shall be <<FeatureType>>, otherwise no stereotype shall be set.

The name of the class shall be the same as the name of the global element of the GML object type.

The class shall be abstract, if and only if the GML object type is abstract.

If the GML object type is derived from another GML object type, then the class inherits from the corresponding superclass. If the base type is defined in the GML application schema or another imported GML application schema, then the superclass is the class corresponding to this GML object type. If the base type is defined in the GML namespace, then the superclass is determined by Table D.2. If the base type is listed in the third column of that table, then the superclass is the class in the first column of the same row.

The GML properties of the GML object type shall be mapped to attributes and association roles as described in F.2.3.9. Assign a tagged value "sequenceNumber" to all UML attributes and association roles created in this mapping with unique integer values in ascending order reflecting the order of the properties in the sequence of the object type.

F.2.3.4 GML object types (imported from the GML schema)

The complex types from the GML namespace listed in the left hand column of Table D.2 shall be mapped to the predefined UML classes implemented by the ISO geographic information standards profile of GML in the second column of the table.

F.2.3.5 Basic types

The simple types from the XML Schema and GML namespace shown in the right hand column of Table D.2 shall be mapped to the predefined UML classes implemented by the ISO geographic information standards profile of GML in the left hand column of the table.

F.2.3.6 GML data types

Every GML data type shall be mapped to a UML class. The name of the class shall be the same as the name of the complex type without the “Type”-suffix.

If the GML data type is derived from another GML data type (base type), then the class inherits from the corresponding superclass.

If the properties of the GML data type are embedded in an xsd:sequence element, the stereotype of the class shall be <<DataType>>, if they are embedded in an xsd:choice element, the stereotype of the class shall be set to <<Union>>.

The GML properties of the GML object type shall be mapped to attributes and association roles as described in F.2.3.9. Assign a tagged value "sequenceNumber" to all UML attributes and association roles created in this mapping with unique integer values in ascending order reflecting the order of the properties in the sequence of the object type.

F.2.3.7 Enumerations

A simple type defined in the GML application schema as a restriction of xsd:string with enumeration values shall be mapped to a class with the <<Enumeration>> stereotype in the UML application schema.

The name of the class shall be the name of the simple type.

Every xsd:enumeration facet without an xsd:appInfo annotation with a child element `gml:codeListValue` shall be mapped to a UML attribute with the value as the attribute name.

Every xsd:enumeration facet with an xsd:appInfo annotation with a child element `gml:codelistValue` shall be mapped to an initial value of the UML attribute with the same name as the value of the `gml:codelistValue` element. If no such UML attribute exists in the class, the facet shall be ignored.

F.2.3.8 Code lists

A simple type defined an the GML application schema as a union of an xsd:pattern restriction with the value “other:\w{2,}” and an enumeration shall be mapped to a class with the stereotype <<CodeList>> in the UML application schema.

The name of the class shall be the name of the simple type.

Every xsd:enumeration facet of the enumeration without an xsd:appInfo annotation with a child element `gml:codelistValue` shall be mapped to a UML attribute with the value as the attribute name.

Every xsd:enumeration facet of the enumeration with an xsd:appInfo annotation with a child element `gml:codelistValue` shall be mapped to an initial value of the UML attribute with the same name as the value of the `gml:codelistValue` element. If no such UML attribute exists in the class, the facet shall be ignored.

F.2.3.9 GML properties

If the type of a property element:

— is a simple type or the property type of GML data type, the property shall be mapped to a UML attribute with the corresponding type as the data type;

— is a property type of a GML object type (inline and/or by-reference) whose content model is directly or indirectly derived from gml:AbstractMemberType, the property shall be mapped to a UML association role of a UML aggregation to the class representing the target GML object type; if the content model of the property element contains an attribute "owns" with a fixed value of "true" (through a Schematron constraint) then the UML aggregation shall be change to a UML composition;

— is a property type of a GML object type (inline and/or by-reference), the property shall be mapped to a UML association role of a UML association to the class representing the target GML object type; if the property type supports only by-reference, the target GML object type shall be determined from the embedded xsd:appInfo annotation with a child element `gml:targetElement` specifying the qualified element name of the target type. The tagged value "inlineOrByReference" shall be set to "inline" for representations that allow only an inline encoding of the property value and to "byReference" for representations that allow only a by-reference encoding of the property value;

— is a property type of a GML object type (inline and/or by-reference) whose content model is directly or indirectly derived from gml:AbstractMetadataPropertyType, the UML attribute or association role shall carry a tagged value "isMetadata" with the value "true".

The name of the UML attribute or association role shall be the name of the GML property element.

The multiplicity of the UML attribute or association role shall be derived from the minOccurs and maxOccurs value of the GML property.

If the property element has an xsd:appInfo annotation with a child element `gml:reversePropertyName` embedded, then the association role shall be defined as part of the association between the two classes where the other association role has a name equal to the value of the `gml:reversePropertyName` element.

F.2.3.10 Documentation

XML Schema xsd:annotation/xsd:documentation elements in GML application schemas are mapped to “documentation” tagged values in the UML application schema.

Annex G
(informative)

Guidelines for subsetting the GML schema

G.1 General

An automated approach is recommended for subsetting the GML schema. This annex contains an informative XSLT reference implementation of a GML schema subset tool. The tool consists of three XSLT stylesheets; the three stylesheets are shown in G.2, G.3 and G.4 below.

To create a GML subset schema using this tool:

a) Transform gml.xsd using depends.xslt and an XSLT processor to produce gml.dep.

EXAMPLE 1 Using Xalan the command could be

```
$ java org.apache.xalan.xslt.Process -IN ../base/gml.xsd -XSL depends.xslt -OUT gml.dep
```

b) If the XSLT processor you are using cannot pass parameters to a stylesheet being processed, edit gmlSubset.xslt, and change the “wanted” parameter to contain a comma separated list (with a trailing comma) of the namespace-qualified global types and elements you want in your GML subset schema.

EXAMPLE 2 For example, change

```
<xsl:param name="wanted">,</xsl:param>
```

to

```
<xsl:param name="wanted">gml:featureProperty,gml:lineStringProperty,gml:polygonProperty,</xsl:param>
```

c) Transform gml.dep using gmlSubset.xslt, a parameter named “wanted” set to a comma separated list (with a trailing comma) of the namespace qualified global types and elements you want in your GML subset schema, and an XSLT processor to produce gmlSubset.xsd, which will contain the global types and elements specified in the “wanted” parameter and all of the global types and elements on which they directly or indirectly use.

d) The generated gmlSubset.xsd will include imports for the namespaces named “xlink” if your “wanted” list included or depended on any attribute from the corresponding namespace. Otherwise, it is a stand-alone GML subset schema that conforms to the requirements for GML profiles.

G.2 depends.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xlink="http://www.w3.org/1999/xlink">
    <!-- ============================================================
    This stylesheet is designed to be used on gml.xsd to produce gml.dep
    for use by the gml schema subset utility gmlSubset.xslt to produce a specialized
    gmlSubset.xsd that contains only the specified types and elements, and the types
```

```
    and elements on which they depend.
    ==============================================================-->
    <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
    <xsl:include href="utility.xslt"/>
    <!-- NEWLINE = &#xA; -->
    <xsl:param
name="schemas">gml.xsd,observation.xsd,dynamicFeature.xsd,coverage.xsd,topology.xsd,defaultStyle.xsd,coordinateReferenceSyst
ems.xsd,feature.xsd,valueObjects.xsd,grids.xsd,geometryComplexes.xsd,datums.xsd,coordinateSystems.xsd,coordinateOperations.x
sd,geometryAggregates.xsd,referenceSystems.xsd,dataQuality.xsd,geometryPrimitives.xsd,geometryBasic2d.xsd,direction.xsd,geom
etryBasic0d1d.xsd,measures.xsd,temporal.xsd,units.xsd,dictionary.xsd,gmlBase.xsd,basicTypes.xsd,</xsl:param>
    <xsl:param name="allSchemas">
        <xsl:call-template name="getUniqueSchemaList">
            <xsl:with-param name="list" select="$schemas"/>
            <xsl:with-param name="usePre"></xsl:with-param>
        </xsl:call-template>
    </xsl:param>
    <xsl:template match="/">
        <xsl:param name="docName">gml.xsd</xsl:param>
        <xsl:param name="top" select="true()"/>
        <xsl:param name="tns" select="//xsd:schema/@targetNamespace"/>
        <xsl:param name="vers" select="//xsd:schema/@version"/>
        <xsl:variable name="ltns">
            <xsl:for-each select="//xsd:schema/namespace::*">
                <xsl:if test="local-name() != 'targetNamespace' and string() = $tns">
                    <xsl:value-of select="local-name()"/>
                </xsl:if>
            </xsl:for-each>
        </xsl:variable>
        <xsl:variable name="tnsp">
            <xsl:choose>
                <xsl:when test="$ltns = ''">
                    <xsl:call-template name="getTargetNameSpacePrefix">
                        <xsl:with-param name="list" select="$tns"/>
                    </xsl:call-template>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$ltns"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:text>&#xA;</xsl:text>
        <xsl:choose>
            <xsl:when test="$top">
                <xsl:text disable-output-escaping="yes">&lt;depends version="</xsl:text><xsl:value-of select="$vers"/><xsl:text
disable-output-escaping="yes">"&gt;</xsl:text>
            </xsl:when>
            <xsl:otherwise>
                <xsl:for-each select="/xsd:schema">
                    <xsl:for-each select="xsd:complexType | xsd:group | xsd:simpleType | xsd:element | xsd:attribute |
xsd:attributeGroup">
                        <xsl:variable name="type" select="local-name()"/>
                        <xsl:choose>
                            <xsl:when test="$type = 'complexType' ">
                                <xsl:call-template name="complexType">
                                    <xsl:with-param name="docName" select="$docName"/>
                                    <xsl:with-param name="targetNamespace" select="$tnsp"/>
```



```
                        </xsl:call-template>
                    </xsl:when>
                    <xsl:when test="$type = 'group' ">
                        <xsl:call-template name="complexType">
                            <xsl:with-param name="docName" select="$docName"/>
                            <xsl:with-param name="targetNamespace" select="$tnsp"/>
                        </xsl:call-template>
                    </xsl:when>
                    <xsl:when test="$type = 'simpleType' ">
                        <xsl:call-template name="simpleType">
                            <xsl:with-param name="docName" select="$docName"/>
                            <xsl:with-param name="targetNamespace" select="$tnsp"/>
                        </xsl:call-template>
                    </xsl:when>
                    <xsl:when test="$type = 'element' ">
                        <xsl:call-template name="globalElement">
                            <xsl:with-param name="docName" select="$docName"/>
                            <xsl:with-param name="targetNamespace" select="$tnsp"/>
                        </xsl:call-template>
                    </xsl:when>
                    <xsl:when test="$type = 'attribute' ">
                        <xsl:call-template name="globalAtt">
                            <xsl:with-param name="docName" select="$docName"/>
                            <xsl:with-param name="targetNamespace" select="$tnsp"/>
                        </xsl:call-template>
                    </xsl:when>
                    <xsl:when test="$type = 'attributeGroup' ">
                        <xsl:call-template name="globalAtt">
                            <xsl:with-param name="docName" select="$docName"/>
                            <xsl:with-param name="targetNamespace" select="$tnsp"/>
                        </xsl:call-template>
                    </xsl:when>
                    <xsl:otherwise/>
                </xsl:choose>
            </xsl:for-each>
        </xsl:for-each>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:if test="$top">
    <xsl:call-template name="dependSchemas">
        <xsl:with-param name="list" select="$allSchemas"/>
    </xsl:call-template>
    <xsl:text disable-output-escaping="yes">&#xA;&lt;/depends&gt;&#xA;</xsl:text>
  </xsl:if>
</xsl:template>
<!-- ============================================================ -->
<xsl:template name="complexType">
  <xsl:param name="docName"/>
  <xsl:param name="targetNamespace"/>
  <xsl:variable name="name" select="@name"/>
  <xsl:if test="$name">
    <xsl:element name="def">
        <xsl:attribute name="name"><xsl:value-of select="$targetNamespace"/>:<xsl:value-of
select="$name"/></xsl:attribute>
        <xsl:attribute name="doc"><xsl:value-of select="$docName"/></xsl:attribute>
        <xsl:variable name="uses">
```

```
                    <xsl:apply-templates select="./xsd:complexContent|./xsd:simpleContent"/>
                    <xsl:call-template name="EltAndAtt"/>
                </xsl:variable>
                <!-- USES <xsl:value-of select="$uses"/> -->
                <xsl:call-template name="writeUses">
                    <xsl:with-param name="list" select="$uses"/>
                </xsl:call-template>
            </xsl:element>
        </xsl:if>
    </xsl:template>
    <!-- ============================================================ -->
    <xsl:template match="xsd:complexContent">
        <xsl:for-each select="descendant::xsd:extension">
            <xsl:value-of select="@base"/>
            <xsl:text>?extension|</xsl:text>
        </xsl:for-each>
        <xsl:for-each select="descendant::xsd:restriction">
            <xsl:value-of select="@base"/>
            <xsl:text>?restriction|</xsl:text>
        </xsl:for-each>
    </xsl:template>
    <!-- ============================================================ -->
    <xsl:template match="xsd:simpleContent">
        <xsl:for-each select="descendant::xsd:extension">
            <xsl:value-of select="@base"/>
            <xsl:text>?extension|</xsl:text>
        </xsl:for-each>
        <xsl:for-each select="descendant::xsd:restriction">
            <xsl:value-of select="@base"/>
            <xsl:text>?restriction|</xsl:text>
        </xsl:for-each>
    </xsl:template>
    <!-- ============================================================ -->
    <xsl:template name="EltAndAtt">
        <xsl:for-each select="descendant::xsd:element | descendant::xsd:group | descendant::xsd:attribute |
descendant::xsd:attributeGroup">
            <xsl:variable name="name" select="@type | @ref"/>
            <xsl:if test="$name and contains($name,':')">
                <xsl:value-of select="$name"/>
                <xsl:text>|</xsl:text>
            </xsl:if>
        </xsl:for-each>
    </xsl:template>
    <!-- ============================================================ -->
    <xsl:template name="simpleType">
        <xsl:param name="docName"/>
        <xsl:param name="targetNamespace"/>
        <xsl:variable name="name" select="@name"/>
        <xsl:if test="$name">
            <xsl:element name="def">
                <xsl:attribute name="name"><xsl:value-of select="$targetNamespace"/>:<xsl:value-of
select="$name"/></xsl:attribute>
                <xsl:attribute name="doc"><xsl:value-of select="$docName"/></xsl:attribute>
                <!-- SIMPLE <xsl:copy-of select="."/>-->
                <xsl:variable name="uses">
                    <xsl:for-each select="xsd:union">
```



```
                        <!-- UNION <xsl:value-of select="@memberTypes"/> -->
                            <xsl:variable name="members" select="@memberTypes"/>
                            <xsl:if test="$members">
                                <xsl:value-of select="translate($members,' ','|')"/>
                                <xsl:text>|</xsl:text>
                            </xsl:if>
                        </xsl:for-each>
                        <xsl:for-each select="xsd:list">
                            <xsl:variable name="items" select="@itemType"/>
                            <xsl:if test="$items">
                                <xsl:value-of select="$items"/>
                                <xsl:text>|</xsl:text>
                            </xsl:if>
                        </xsl:for-each>
                    </xsl:variable>
                     <!-- USES <xsl:value-of select="$uses"/> -->
                    <xsl:call-template name="writeUses">
                        <xsl:with-param name="list" select="$uses"/>
                    </xsl:call-template>
                </xsl:element>
            </xsl:if>
        </xsl:template>
        <!-- ============================================================ -->
        <xsl:template name="globalElement">
            <xsl:param name="docName"/>
            <xsl:param name="targetNamespace"/>
            <xsl:variable name="name" select="@name"/>
            <xsl:if test="$name">
                <xsl:element name="def">
                    <xsl:attribute name="name"><xsl:value-of select="$targetNamespace"/>:<xsl:value-of
select="$name"/></xsl:attribute>
                    <xsl:attribute name="doc"><xsl:value-of select="$docName"/></xsl:attribute>
                    <xsl:variable name="uses">
                        <xsl:variable name="type" select="@type"/>
                        <xsl:if test="$type and contains($type,':')">
                            <xsl:value-of select="$type"/>
                            <xsl:text>|</xsl:text>
                        </xsl:if>
                        <xsl:variable name="sub" select="@substitutionGroup"/>
                        <xsl:if test="$sub">
                            <xsl:value-of select="$sub"/>
                            <xsl:text>|</xsl:text>
                        </xsl:if>
                    </xsl:variable>
                    <!-- USES <xsl:value-of select="$uses"/> -->
                    <xsl:call-template name="writeUses">
                        <xsl:with-param name="list" select="$uses"/>
                    </xsl:call-template>
                </xsl:element>
            </xsl:if>
        </xsl:template>
        <!-- ============================================================ -->
        <xsl:template name="globalAtt">
            <xsl:param name="docName"/>
            <xsl:param name="targetNamespace"/>
            <xsl:variable name="name" select="@name"/>
```

```
            <xsl:if test="$name">
                <xsl:element name="def">
                    <xsl:attribute name="name"><xsl:value-of select="$targetNamespace"/>:<xsl:value-of
select="$name"/></xsl:attribute>
                    <xsl:attribute name="doc"><xsl:value-of select="$docName"/></xsl:attribute>
                    <xsl:variable name="uses">
                        <xsl:variable name="type" select="@type"/>
                        <xsl:if test="$type and contains($type,':')">
                            <xsl:value-of select="$type"/>
                            <xsl:text>|</xsl:text>
                        </xsl:if>
                        <xsl:call-template name="EltAndAtt"/>
                    </xsl:variable>
                    <!-- USES <xsl:value-of select="$uses"/> -->
                    <xsl:call-template name="writeUses">
                        <xsl:with-param name="list" select="$uses"/>
                    </xsl:call-template>
                </xsl:element>
            </xsl:if>
        </xsl:template>
        <!-- =========================================================== -->
        <xsl:template name="writeUses">
            <xsl:param name="list"/>
            <xsl:if test="$list != ''">
                <xsl:variable name="first" select="substring-before($list, '|')"/>
                <xsl:variable name="eor" select="substring-after($first, '?')"/>
                <xsl:variable name="use">
                    <xsl:choose>
                        <xsl:when test="contains($first, '?')">
                            <xsl:value-of select="substring-before($first, '?')"/>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="$first"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:variable>
                <xsl:variable name="testp">
                    <xsl:value-of select="$use"/>
                    <xsl:text>|</xsl:text>
                </xsl:variable>
                <xsl:variable name="testq">
                    <xsl:value-of select="$use"/>
                    <xsl:text>?</xsl:text>
                </xsl:variable>
                <xsl:variable name="rest" select="substring-after($list, '|')"/>
                <xsl:choose>
                    <xsl:when test="contains($rest, $testp)"/>
                    <xsl:when test="contains($rest, $testq)"/>
                    <xsl:when test="$use = ''"/>
                    <xsl:otherwise>
                        <xsl:element name="uses">
                            <xsl:attribute name="name"><xsl:value-of select="$use"/></xsl:attribute>
                            <xsl:if test="$eor != ''">
                                <xsl:attribute name="derivation"><xsl:value-of select="$eor"/> </xsl:attribute>
                            </xsl:if>
                        </xsl:element>
```



```
                </xsl:otherwise>
            </xsl:choose>
            <xsl:call-template name="writeUses">
                <xsl:with-param name="list" select="$rest"/>
            </xsl:call-template>
        </xsl:if>
    </xsl:template>
    <!-- ======================================================= -->
    <xsl:template name="dependSchemas">
        <xsl:param name="list"/>
        <xsl:if test="$list != ''">
            <xsl:variable name="first" select="substring-before($list, ',')"/>
            <xsl:variable name="rest" select="substring-after($list, ',')"/>
            <xsl:apply-templates select="document($first, /)">
                <xsl:with-param name="docName" select="$first"/>
                <xsl:with-param name="top" select="false()"/>
            </xsl:apply-templates>
            <xsl:choose>
                <xsl:when test="contains($rest,',')">
                    <xsl:call-template name="dependSchemas">
                        <xsl:with-param name="list" select="$rest"/>
                    </xsl:call-template>
                </xsl:when>
                <xsl:otherwise/>
            </xsl:choose>
        </xsl:if>
    </xsl:template>
    <!-- ============================================================== -->
    <!-- ============================================================== -->
</xsl:stylesheet>
```

G.3 gmlSubset.xslt

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xlink="http://www.w3.org/1999/xlink">
    <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
    <!-- ==============================================================
    This stylesheet is designed to be used on gml.dep (produced from
    gml.xsd by depends.xslt) to produce a specialized gmlSubset.xsd that
    contains only the types and elements specified in the "wanted" parameter,
    and the types and elements on which they depend. Note that the type and
    element items in the "wanted" parameter must include namespace prefixes,
    and that they must be separated by commas, including a trailing comma after
    the last item.
    ==============================================================-->
    <xsl:include href="utility.xslt"/>
    <xsl:param name="baseUri" select="document('../../base/gml.xsd')"/>
    <!-- sample1 <xsl:param name="wanted">gml:featureProperty,gml:lineStringProperty,gml:polygonProperty,</xsl:param> -->
    <!-- sample2 <xsl:param name="wanted">gml:GeodeticCRS,gml:AbstractCoverage,gml:track,</xsl:param> -->
    <!-- sample3     <xsl:param name="wanted">gml:AbstractFeatureCollection,gml:ItemStyleDescriptorType,
    gml:FeatureConstraintType,</xsl:param> -->
    <xsl:param name="wanted">gml:metaDataProperty,gml:Abstractassociation,gml:members,gml:Array,gml:curveProperty,
    gml:LineString,gml:LinearRing,gml:exterior,gml:interior,gml:surfaceMember,gml:surfaceProperty,gml:multiSurfaceProperty,
```

```
    gml:directedNode,gml:directedEdge,gml:directedFace,gml:IsolatedProperty,gml:featureProperty,gml:featureMembers,
    gml:AbstractFeatureCollection,gml:featureMember,gml:BaseStyleDescriptorType,</xsl:param>
    <xsl:template match="/">
        <xsl:variable name="wantedList">
            <xsl:call-template name="getWantedList">
                <xsl:with-param name="list" select="$wanted"/>
                <xsl:with-param name="from">BEGIN</xsl:with-param>
                <xsl:with-param name="depth">0</xsl:with-param>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="vers"  select="//depends/@version"/>
        <schema targetNamespace="http://www.opengis.net/gml/3.2" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:sch="http://purl.oclc.org/dsdl/schematron" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified" version="{$vers}">
            <annotation>
                <documentation>GML Subset schema for <xsl:value-of select="$wanted"/>  written by gmlSubset.xslt.
</documentation>
            </annotation>

            <xsl:if test="contains($wantedList,'xlink:')">
                <import namespace="http://www.w3.org/1999/xlink" schemaLocation="../../xlink/xlinks.xsd"/>
            </xsl:if>
            <xsl:call-template name="writeWantedList">
                <xsl:with-param name="list" select="$wantedList"/>
            </xsl:call-template>
        </schema>
    </xsl:template>
    <!-- =========================================================== -->
    <xsl:template name="getDocName">
        <xsl:param name="wanted"/>
        <xsl:for-each select="//depends/def[@name=$wanted]">
            <xsl:value-of select="@doc"/>
        </xsl:for-each>
    </xsl:template>
    <!-- =========================================================== -->
    <xsl:template name="getUses">
        <xsl:param name="wanted"/>
        <xsl:for-each select="//depends/def[@name=$wanted]">
            <xsl:for-each select="uses">
                <xsl:value-of select="@name"/>
                <xsl:text>,</xsl:text>
            </xsl:for-each>
        </xsl:for-each>
    </xsl:template>
    <!-- =========================================================== -->
    <xsl:template name="writeWanted">
        <xsl:param name="wanted"/>
        <xsl:choose>
            <xsl:when test="contains($wanted,'xlink:') ">
                <!-- XLINK <xsl:value-of select="$wanted"/> -->
            </xsl:when>
            <xsl:otherwise>
                <!-- OTHER <xsl:value-of select="$wanted"/> -->
                <xsl:variable name="docName">
                    <xsl:call-template name="getDocName">
                        <xsl:with-param name="wanted" select="$wanted"/>
```



```
                    </xsl:call-template>
                </xsl:variable>
                <xsl:variable name="localName">
                    <xsl:call-template name="removePrefix">
                        <xsl:with-param name="name" select="$wanted"/>
                        <xsl:with-param name="pre">:</xsl:with-param>
                    </xsl:call-template>
                </xsl:variable>
                <xsl:call-template name="Separator"/>
                <xsl:for-each select="document($docName,$baseUri)">
                    <xsl:for-each select="//xsd:schema/*[@name = $localName]">
                        <xsl:copy-of select="."/>
                    </xsl:for-each>
                </xsl:for-each>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
    <!-- ============================================================ -->
    <xsl:template name="writeWantedList">
        <xsl:param name="list"/>
        <xsl:if test="$list != ''">
            <xsl:variable name="first" select="substring-before($list, ',')"/>
            <xsl:variable name="rest" select="substring-after($list, ',')"/>
            <xsl:call-template name="writeWanted">
                <xsl:with-param name="wanted" select="$first"/>
            </xsl:call-template>
            <xsl:if test="contains($rest,',')">
                <xsl:call-template name="writeWantedList">
                    <xsl:with-param name="list" select="$rest"/>
                </xsl:call-template>
            </xsl:if>
        </xsl:if>
    </xsl:template>
    <!-- ============================================================ -->
    <xsl:template name="getWantedList">
        <xsl:param name="list"/>
        <xsl:param name="seen"/>
        <xsl:param name="from"/>
        <xsl:param name="depth"/>
        <xsl:if test="$list != ''">
            <xsl:variable name="first" select="substring-before($list, ',')"/>
            <xsl:variable name="firstSep" select="concat($first,',')"/>
            <xsl:variable name="rest" select="substring-after($list, ',')"/>
            <xsl:choose>
                <xsl:when test="contains($seen,$firstSep)">
                    <xsl:call-template name="getWantedList">
                        <xsl:with-param name="list" select="$rest"/>
                        <xsl:with-param name="seen" select="$seen"/>
                        <xsl:with-param name="from">REST</xsl:with-param>
                        <xsl:with-param name="depth" select="$depth + 1"/>
                    </xsl:call-template>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$firstSep"/>
                    <xsl:variable name="usesList">
                        <xsl:call-template name="getUses">
```

```
                <xsl:with-param name="wanted" select="$first"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="toDo" select="concat($usesList,$rest)"/>
        <xsl:variable name="nowSeen" select="concat($seen,$firstSep)"/>
        <xsl:call-template name="getWantedList">
            <xsl:with-param name="list" select="$toDo"/>
            <xsl:with-param name="seen" select="$nowSeen"/>
            <xsl:with-param name="from">USES</xsl:with-param>
            <xsl:with-param name="depth" select="$depth + 1"/>
        </xsl:call-template>
    </xsl:otherwise>
   </xsl:choose>
  </xsl:if>
 </xsl:template>
 <!-- ============================================================ -->
</xsl:stylesheet>
```

G.4 utility.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
 <!-- ===================================================== -->
 <xsl:template name="getTargetNameSpacePrefix">
  <xsl:param name="list"/>
  <xsl:if test="$list != ''">
   <xsl:variable name="first" select="substring-before($list, '/')"/>
   <xsl:variable name="rest" select="substring-after($list, '/')"/>
   <xsl:choose>
    <xsl:when test="contains($rest,'/')">
     <xsl:call-template name="getTargetNameSpacePrefix">
      <xsl:with-param name="list" select="$rest"/>
     </xsl:call-template>
    </xsl:when>
    <xsl:when test="$rest =''">
     <xsl:value-of select="$first"/>
    </xsl:when>
    <xsl:otherwise>
     <xsl:value-of select="$rest"/>
    </xsl:otherwise>
   </xsl:choose>
  </xsl:if>
 </xsl:template>
 <!-- ===================================================== -->
 <xsl:template name="getPathPrefix">
  <xsl:param name="file"/>
  <xsl:if test="contains($file,'/')">
   <xsl:variable name="pre" select="substring-before($file,'/')"/>
   <xsl:variable name="suf" select="substring-after($file,'/')"/>
   <xsl:choose>
    <xsl:when test="contains($suf,'/')">
     <xsl:value-of select="$pre"/><xsl:text>/</xsl:text>
     <xsl:call-template name="getPathPrefix">
      <xsl:with-param name="file" select="$suf"/>
```

```
                    </xsl:call-template>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:variable name="path">
                        <xsl:call-template name="removeSuffix">
                            <xsl:with-param name="name" select="$file"/>
                            <xsl:with-param name="suf" select="$suf"/>
                        </xsl:call-template>
                    </xsl:variable>
                    <xsl:value-of select="$path"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:if>
    </xsl:template>
    <!-- ===================================================== -->
    <xsl:template name="removePrefix">
        <xsl:param name="name"/>
        <xsl:param name="pre"/>
        <xsl:variable name="npName">
            <xsl:choose>
                <xsl:when test="contains($name,$pre)">
                    <xsl:value-of select="substring-after($name,$pre)"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$name"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:value-of select="$npName"/>
    </xsl:template>
    <!-- ===================================================== -->
    <xsl:template name="removeSuffix">
        <xsl:param name="name"/>
        <xsl:param name="suf"/>
        <xsl:variable name="nsName">
            <xsl:choose>
                <xsl:when test="contains($name,$suf)">
                    <xsl:value-of select="substring-before($name,$suf)"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$name"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:value-of select="$nsName"/>
    </xsl:template>
    <!-- ===================================================== -->
    <xsl:template name="lowerLeading">
        <xsl:param name="name"/>
        <xsl:variable name="ch1" select="substring($name, 1, 1)"/>
        <xsl:variable name="lc1" select="translate($ch1,'ABCDEFGHIJKLMNOPQRSTUVWXYZ','abcdefghijklmnopqrstuvwxyz')"/>
        <xsl:value-of select="concat($lc1, substring($name, 2))"/>
    </xsl:template>
    <!-- ===================================================== -->
    <xsl:template name="uniqueList">
        <xsl:param name="list"/>
```



```
        <xsl:param name="sep"/>
        <xsl:param name="seen"/>
        <xsl:param name="pre">../base/</xsl:param>
        <xsl:if test="$list != ''">
            <xsl:variable name="first" select="substring-before($list, $sep)"/>
            <xsl:variable name="firstSep" select="concat($first,$sep)"/>
            <xsl:variable name="rest" select="substring-after($list, $sep)"/>
            <xsl:choose>
                <xsl:when test="contains($seen,$firstSep)">
                    <xsl:call-template name="uniqueList">
                        <xsl:with-param name="list" select="$rest"/>
                        <xsl:with-param name="sep" select="$sep"/>
                        <xsl:with-param name="seen" select="$seen"/>
                    </xsl:call-template>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$firstSep"/>
                    <xsl:variable name="nowSeen" select="concat($seen, $firstSep)"/>
                    <xsl:call-template name="uniqueList">
                        <xsl:with-param name="list" select="$rest"/>
                        <xsl:with-param name="sep" select="$sep"/>
                        <xsl:with-param name="seen" select="$nowSeen"/>
                    </xsl:call-template>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:if>
    </xsl:template>
    <!-- ==================================================== -->
    <xsl:template name="getIncludedDocs">
        <xsl:param name="docName"/>
        <xsl:param name="usePre"/>
        <xsl:param name="seenList"/>
        <xsl:param name="sep">,</xsl:param>
        <xsl:value-of select="$docName"/>
        <xsl:text>,</xsl:text>
        <xsl:variable name="pathPre">
            <xsl:call-template name="getPathPrefix">
                <xsl:with-param name="file" select="$docName"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="callPathPre">
            <xsl:choose>
                <xsl:when test="$pathPre = '' or $pathPre = './' ">
                    <xsl:value-of select="$usePre"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$pathPre"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:for-each select="document($docName, / )">
            <xsl:for-each select="//xsd:include | //xsd:import">
                <xsl:variable name="iDoc" select="@schemaLocation"/>
                <xsl:variable name="iPathPre">
                    <xsl:call-template name="getPathPrefix">
                        <xsl:with-param name="file" select="$iDoc"/>
```



```
                </xsl:call-template>
            </xsl:variable>
            <xsl:variable name="iDocSuf">
                <xsl:call-template name="removePrefix">
                    <xsl:with-param name="name" select="$iDoc"/>
                    <xsl:with-param name="pre" select="$iPathPre"/>
                </xsl:call-template>
            </xsl:variable>
            <xsl:variable name="usePathPre">
                <xsl:choose>
                    <xsl:when test="$iPathPre = '' or $iPathPre = './' ">
                        <xsl:value-of select="$callPathPre"/>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="$iPathPre"/>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:variable>
            <xsl:variable name="uDoc">
                <xsl:value-of select="concat($usePathPre,$iDocSuf)"/>
            </xsl:variable>
            <xsl:variable name="uDocSep">
                <xsl:value-of select="concat($uDoc,$sep)"/>
            </xsl:variable>
            <xsl:if test="not(contains($seenList,$uDocSep))">
                <xsl:variable name="seenListPlus" select="concat($seenList,$uDocSep)"/>
                <xsl:call-template name="getIncludedDocs">
                    <xsl:with-param name="docName" select="$uDoc"/>
                    <xsl:with-param name="usePre" select="$usePathPre"/>
                    <xsl:with-param name="seenList" select="$seenListPlus"/>
                </xsl:call-template>
            </xsl:if>
        </xsl:for-each>
    </xsl:for-each>
</xsl:template>
<!-- ==================================================== -->
<xsl:template name="getDocumentList">
    <xsl:param name="list"/>
    <xsl:param name="seenList"/>
    <xsl:param name="usePre"/>
    <xsl:if test="$list != ''">
        <xsl:variable name="first" select="substring-before($list, ',')"/>
        <xsl:variable name="rest" select="substring-after($list, ',')"/>
        <xsl:variable name="included">
            <xsl:call-template name="getIncludedDocs">
                <xsl:with-param name="docName" select="$first"/>
                <xsl:with-param name="usePre" select="$usePre"/>
                <xsl:with-param name="seenList" select="$seenList"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:value-of select="$included"/>
        <xsl:variable name="seenListIncluded" select="concat($seenList,$included)"/>
        <xsl:if test="contains($rest,',')">
            <xsl:call-template name="getDocumentList">
                <xsl:with-param name="list" select="$rest"/>
                <xsl:with-param name="seenList" select="$seenListIncluded"/>
```

```
                    <xsl:with-param name="usePre" select="$usePre"/>
                </xsl:call-template>
            </xsl:if>
        </xsl:if>
    </xsl:template>
    <!-- ====================================================== -->
    <xsl:template name="getUniqueSchemaList">
        <xsl:param name="list"/>
        <xsl:param name="usePre"/>
        <xsl:variable name="allSchemas">
            <xsl:call-template name="getDocumentList">
                <xsl:with-param name="list" select="$list"/>
                <xsl:with-param name="usePre" select="$usePre"/>
            </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="uniqueSchemas">
            <xsl:call-template name="uniqueList">
                <xsl:with-param name="list" select="$allSchemas"/>
                <xsl:with-param name="sep">,</xsl:with-param>
            </xsl:call-template>
        </xsl:variable>
        <xsl:value-of select="$uniqueSchemas"/>
    </xsl:template>
    <!-- ============================================================ -->
     <xsl:template name="Separator">
   <xsl:param name="comment" select="'=========='"/>
   <xsl:text>&#xA;</xsl:text>
   <xsl:text disable-output-escaping="yes">
      &lt;!-- ==================</xsl:text><xsl:value-of select="$comment"/><xsl:text disable-output-
escaping="yes">====================== --&gt;
  </xsl:text>
    <!-- ============================================================ -->
  </xsl:template>
</xsl:stylesheet>
```

Annex H
(informative)

Default styling

H.1 General

GML has been designed to strictly separate data content from the graphical or other presentation of that data. GML feature descriptions thus do not contain any information related to the presentation of that feature.

This annex provides schema components for defining sets of styling rules that when applied to an associated GML dataset generate a graphical visualization of that data using W3C Scalable Vector Graphics (SVG). These styling rules enable the creation of SVG documents based on data elements including feature type names, thematic or spatial feature properties, etc.

A capability that allows to define styles for GML data is considered to be essential for the portrayal of GML data. The default styling schema components provide a means for this, however, there are known issues, most notably that harmonization with the existing and more widely implemented OpenGIS Implementation Specification Styled-Layer Descriptor (SLD) is required. Potentially also a revision of ISO 19117 should be considered in this step. In addition, additional enhancements should be considered in this process, e.g. the introduction of a style dictionary to separate features from their styling information more clearly. Therefore, the default style schema components are not normative in this International Standard.

Note that it is not considered essential that these style description schema components "live" in the GML namespace, however, it is important that a standardized style description capability exists. When a generally accepted styling schema exists, this annex may be removed.

The default style schema components described in this annex are intended to be used as a separate model that can be "plugged-in" to a GML dataset.

EXAMPLE A typical usage would be to provide a persistent style associated with a particular feature type.

The term "default" signifies a loose relationship to the associated GML data, and the style information that is assigned to this data set may be used for styling but may also be completely ignored. The utilization of the associated default styling rules is thus to be determined by the styling application.

The notion of style as defined in this annex is effectively an association between a GML object (e.g. a feature, geometry, or topology) and a graphical presentation element expressed in SVG. For example, a style may express that the default graphical presentation of a `gml:Curve` representing the centreline of a road feature is to be an SVG path with a particular stroke-width and stroke-colour for the SVG path.

The default style schema components also depend on W3C Synchronized Multimedia Integration Language (SMIL) schemas.

The relation of the default style information and GML data instances is achieved through the `gml:defaultStyle` property. The property may be assigned to the instance by assigning it to the feature type in the associated application schema. Since GML is a feature-based encoding, a default style always applies to a feature, features or feature collections. Default styling enables the graphical presentation of such features based on their properties.

NOTE The conceptual model for the default styling schema and the description of the implementation of ISO 19117 is not part of this annex as it is informative in this International Standard.

H.2 Top-level styling elements

H.2.1 Overview

The connection between a GML data set and a styling description is established through the single property, `gml:defaultStyle`. The value of this property, the `gml:Style` object, contains all styling descriptions. The `gml:defaultStyle` property must be specified in feature type definition in the application schema so that it can be used to associate the feature element with the styling rules.

H.2.2 defaultStyle

The `gml:defaultStyle` property is a property defined as a global element and can be assigned to any feature defined in an application schema. The definition of the property is as follows:

```
<element name="defaultStyle" type="gml:DefaultStylePropertyType"/>

<complexType name="DefaultStylePropertyType">
  <sequence minOccurs="0">
    <element ref="gml:AbstractStyle"/>
  </sequence>
  <attribute name="about" type="anyURI"/>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

The `gml:defaultStyle` property may contain an attribute `about`. This can be used in a feature collection to assign default styles to features in the collection. In this case the `gml:defaultStyle` property is attached to the collection (the application schema could permit any number of such properties), each containing inline or referencing the styling rule information. If the `about` attribute is used, then it may reference any feature (or feature collection); if it is not used then the feature style applies to the parent feature of the `gml:defaultStyle` property to which the about attribute is attached.

This property can be included in a feature via the application schema defining the feature type.

EXAMPLE The following `exp:Road` feature type definition illustrates the inclusion of the `gml:defaultStyle` property:

```
<element name="Road" type="exp:RoadType" substitutionGroup="gml:AbstractFeature"/>

<complexType name="RoadType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element ref="gml:centerLineOf"/>
        <element ref="gml:defaultStyle"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

H.2.3 Style

The `gml:Style` object is the default concrete value of the `gml:defaultStyle` property. It is the top-level styling object that encapsulates all other, partial style descriptions. Its definition is as follows:

```
<element name="Style" type="gml:StyleType" substitutionGroup="gml:AbstractStyle"/>

<complexType name="StyleType">
	<complexContent>
		<extension base="gml:AbstractStyleType">
			<sequence>
				<element ref="gml:featureStyle" minOccurs="0" maxOccurs="unbounded"/>
				<element ref="gml:graphStyle" minOccurs="0"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

The content model of the `gml:Style` object is derived by extension from `gml:AbstractStyleType`. This type serves as an abstract base type for extensibility purposes, i.e. creating custom style objects, and it does not add any new content to the `gml:AbstractGMLType` from which it derives.

```
<element name="AbstractStyle" type="gml:AbstractStyleType" abstract="true"
	substitutionGroup="gml:AbstractGML"/>

<complexType name="AbstractStyleType" abstract="true">
  <complexContent>
    <extension base="gml:AbstractGMLType"/>
  </complexContent>
</complexType>
```

However, it is not assumed that creating custom style objects will be the usual practice since `gml:Style` provides rich capabilities for describing styles.

The definition of the `gml:Style` object presented previously in the text is itself a proper example of using the extensibility mechanism and shows how these rules can be applied in the schema:

— The content model of a concrete style object derives from `gml:AbstractStyleType`.

— The concrete style object is substitutable for `gml:AbstractStyle`.

The function of the styling elements in the `gml:Style` object, namely `gml:featureStyle` and `gml:graphStyle` is to describe styles for two aspects of GML data: individual features and topology graphs that consist of collections of features. Note that elements that describe styles for particular aspects of features, namely, feature style, graph style, geometry style, topology style and label style are often called style descriptors.

H.3 Feature style

H.3.1 FeatureStyle

A feature style descriptor is assigned to a `gml:Style` through the `gml:featureStyle` property. It allows, like other GML properties, to specify the value inline or remotely.

```
<element name="featureStyle" type="gml:FeatureStylePropertyType"/>

<complexType name="FeatureStylePropertyType">
	<sequence minOccurs="0">
		<element ref="gml:FeatureStyle"/>
	</sequence>
	<attribute name="about" type="anyURI"/>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

Its value is a `gml:FeatureStyle` — the feature style descriptor. A feature style descriptor describes the styling information for a set of feature instances. The set is defined by the selection mechanisms that are part of this style descriptor. The style applies to each feature in the set independently — no relations that might exist among features in the set are significant.

NOTE 1 The opposite case is graph style where the style applies to a set of features as a whole.

The definition of the feature style descriptor is as follows:

```
<element name="FeatureStyle" type="gml:FeatureStyleType" substitutionGroup="gml:AbstractGML"/>

<complexType name="FeatureStyleType">
    <complexContent>
        <extension base="gml:AbstractGMLType">
            <sequence>
                <element name="featureConstraint" type="string" minOccurs="0"/>
                <element ref="gml:geometryStyle" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="gml:topologyStyle" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="gml:labelStyle" minOccurs="0"/>
            </sequence>
            <attribute name="featureType" type="string"/>
            <attribute name="baseType" type="string"/>
            <attribute name="queryGrammar" type="gml:QueryGrammarEnumeration"/>
        </extension>
    </complexContent>
</complexType>
```

Feature instances to which the style applies are selected using one of the attributes `featureType` or `baseType` and `gml:featureConstraint` element. These two attributes shall be used exclusively, with or without the `gml:featureConstraint` element.

NOTE 2 In a revision, the `gml:featureConstraint` property elements should become an attribute.

H.3.2 featureType

The simplest and most common way of relating features and styles is by using this attribute. Its value will be the declared name of a feature, instances of which we want to style.

EXAMPLE If the value is `exp:Road`, the `gml:FeatureStyle` object will simply apply to all Road features. The value of this attribute is always the name of the element from the application schema that declares the feature.

H.3.3 baseType

Another way of selecting the feature instances to which the style applies is to specify, as the value of this attribute, the name of the base type from which feature or features derive. This is always the name of an XML Schema complex type. Any complex type from the derivation chain can be used; the style applies to any feature instance that ultimately derives from it.

EXAMPLE If `gml:AbstractFeatureType` is used as the value of the attribute, the style applies to all feature instances in a data set.

H.3.4 featureConstraint

This property is used to further constrain the feature instance set to which the style applies. It is optional and its value is an XPath expression. If the property does not exist, the style applies to all feature instances selected by `featureType` or `baseType` attribute.

H.3.5 queryGrammar

The value of this property which is defined as an enumeration specifies the grammar that is used in the content of the `gml:featureConstraint` element. The enumeration allows for three values: "Xpath", "Xquery" and "other".

Styling features means styling a particular aspect or aspects of a feature. We can style feature geometry, topology or display arbitrary text string. Feature style contains three style descriptors for respective purposes: `gml:GeometryStyle`, `gml:TopologyStyle` and `gml:LabelStyle`.

H.4 Geometry style

The value of the `gml:geometryStyle` property is `gml:GeometryStyle` descriptor which describes the style for one geometry of a feature. Any number of geometry style descriptors can be assigned to one feature style. This is usually required for features with multiple geometry properties.

```
<element name="geometryStyle" type="gml:GeometryStylePropertyType"/>

<complexType name="GeometryStylePropertyType">
	<sequence minOccurs="0">
		<element ref="gml:GeometryStyle" />
	</sequence>
	<attribute name="about" type="anyURI"/>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>

<element name="GeometryStyle" type="gml:GeometryStyleType" substitutionGroup="gml:AbstractGML"/>

<complexType name="GeometryStyleType">
	<complexContent>
		<extension base="gml:BaseStyleDescriptorType">
			<sequence>
				<choice>
					<element ref="gml:symbol"/>
					<element name="style" type="string"/>
				</choice>
				<element ref="gml:labelStyle" minOccurs="0"/>
			</sequence>
			<attribute name="geometryProperty" type="string"/>
			<attribute name="geometryType" type="string"/>
		</extension>
	</complexContent>
</complexType>
```

The `gml:geometryStyle` is defined in the same manner as other GML properties which allow for referencing the value remotely or inline.

The `geometryProperty` attribute on the `gml:GeometryStyle` specifies the name of the geometry property of a feature to which this geometry style descriptor applies. It is necessary to specify the geometry type using `geometryType` attribute as well since the application schema of the geometry property may allow different geometries as its value.

The property `gml:symbol` is described in H.7.2.

The property style has been deprecated.

H.5 Topology style

The value of the `gml:topologyStyle` property is a `gml:TopologyStyle` descriptor which describes the style for one topology property. Similarly to the `gml:GeometryStyle`, a feature can have multiple topology properties, thus multiple topology style descriptors can be specified within one feature style.

```
<element name="topologyStyle" type="gml:TopologyStylePropertyType"/>

<complexType name="TopologyStylePropertyType">
	<sequence minOccurs="0">
		<element ref="gml:TopologyStyle"/>
	</sequence>
	<attribute name="about" type="anyURI"/>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>

<element name="TopologyStyle" type="gml:TopologyStyleType" substitutionGroup="gml:AbstractGML"/>

<complexType name="TopologyStyleType">
	<complexContent>
		<extension base="gml:BaseStyleDescriptorType">
			<sequence>
				<choice>
					<element ref="gml:symbol"/>
					<element name="style" type="string"/>
				</choice>
				<element ref="gml:labelStyle" minOccurs="0"/>
			</sequence>
			<attribute name="topologyProperty" type="string"/>
			<attribute name="topologyType" type="string"/>
		</extension>
	</complexContent>
</complexType>
```

The `gml:topologyStyle` property is defined in the same manner as other GML properties which allow for referencing the value remotely or inline.

The `topologyProperty` attribute on the `gml:TopologyStyle` descriptor specifies the name of the topology property of a feature to which this topology style descriptor applies. It is necessary to specify the topology type using `topologyType` attribute as well since the application schema of the topology property may allow different topologies as its value.

The property `gml:symbol` is described in H.7.2.

The property style has been deprecated.

H.6 Label style

The value of the `gml:labelStyle` property is `gml:LabelStyle` descriptor which describes the style for the text that is to be displayed along with the graphical representation of a feature. The content of the label is not necessarily defined in the GML data set. More precisely, the content can be static text specified in the style itself and the text from the GML data set.

Label style has two elements: `gml:style` that specifies the style and `gml:label` that is used to compose the label content.

```
<element name="labelStyle" type="gml:LabelStylePropertyType"/>
```



```
<complexType name="LabelStylePropertyType">
	<sequence minOccurs="0">
		<element ref="gml:LabelStyle"/>
	</sequence>
	<attribute name="about" type="anyURI"/>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>

<element name="LabelStyle" type="gml:LabelStyleType" substitutionGroup="gml:AbstractGML"/>

<complexType name="LabelStyleType">
	<complexContent>
		<extension base="gml:BaseStyleDescriptorType">
			<sequence>
				<element name="style" type="string"/>
				<element name="label" type="gml:LabelType"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

The `gml:labelStyle` property is defined in the same manner as other GML properties which allow for referencing the value remotely or inline.

The `gml:style` element is used to specify the style of the rendered text. The type of this element is `string` and the CSS2 (Cascading Style Sheet Version 2.0) styling expressions grammar is used to express graphic properties.

EXAMPLE 1 The following feature style shows the use of the `gml:style` element in a geometry style context.

```
<gml:FeatureStyle featureType="exp:City">
  <gml:GeometryStyle>
    <gml:style>fill:blue;stroke:white</gml:style>
  </gml:GeometryStyle>
</gml:FeatureStyle>
```

As noted, the `gml:label` property on the `gml:LabelStyle` descriptor holds the textual content that can be composed of static text and the text extracted from the GML data.

```
<complexType name="LabelType" mixed="true">
  <sequence>
    <element name="LabelExpression" type="string" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute ref="gml:transform"/>
</complexType>
```

The content model is mixed to allow both text content and unbounded number of `gml:LabelExpression` elements. The value of a `gml:LabelExpression` element is an XPath expression that selects the value of some property of the feature.

EXAMPLE 2 Consider this GML data fragment and corresponding gml:label style:

```
<exp:City>
  <gml:name>Belgrade</gml:name>
  <exp:size>1,700,000</exp:size>
  <gml:extentOf>
  …
  </gml:extentOf>
</exp:City>
```



```
<gml:FeatureStyle featureType="exp:City">
  <gml:LabelStyle>
    <gml:style>font-family:Verdana;font-size:16;fill:red</gml:style>
    <gml:label>
      City:
      <gml:LabelExpression>//City/name</gml:LabelExpression>
      , Size:
      <gml:LabelExpression>//City/size</gml:LabelExpression>
    </gml:label>
  </gml:LabelStyle>
</gml:FeatureStyle>
```

This label style will result in the following text being displayed:

City: Belgrade, Size: 1,700,000

H.7 Common styling elements

H.7.1 Overview

Some common styling elements are used in multiple style descriptors. The `gml:symbol` element is used by geometry and topology style descriptors. The `spatialResolution`, `styleVariation` and `animation` attributes are declared in `gml:BaseStyleDesriptorType`, and inherited by geometry, topology, label and graph style descriptors.

H.7.2 symbol

The `gml:symbol` property element specifies a graphical symbol used to render a geometry or a topology. A symbol is a description of graphical attributes of a graphical object without a particular, implicit meaning. It can be a description of a line, circle, polygon or more complex drawing. Using the symbol element, we can specify a particular symbol in two ways:

— Remote: Just like any other remote property, the symbol property has the `gml:AssociationAttributeGroup` attributes that allow for specifying a link pointing to a remote object.

— Inline: The value of the `gml:symbol` property is the any specifier. This allows for specifying an arbitrary grammar for the symbol.

This element has two additional attributes: `symbolType` and `transform`. The `symbolType` attribute is an enumeration and can take one of three values: "svg", "xpath" or "other". Applications will rely on the value of this attribute to decide how to interpret the symbol.

The `transform` attribute allows to specify a transformation expression that will be applied to the symbol in the rendering phase. Its type is `string` and the value is specified in SVG (transform attribute).

H.7.3 styleVariation

The function of then `gml:styleVariation` property element is manifold:

— Styling labels: Label style does not have a symbol associated with it since the content is not graphical but is given textually. This property can be used to specify its style attributes.

— Styling symbol variations: One symbol is often used in different cases with slight modifications. It would be cumbersome to create and manage large number of virtually identical symbols; it is easier to create and use only one symbol and express minor differences in its style using this property.

— Parametrized styles: Parametrized styles are styles whose attributes depend on some property of the feature being styled.

EXAMPLE A city might be styled differently depending on its population. The `gml:styleVariation` property allows for specifying such dependencies.

The content model of this property is:

```
<complexType name="StyleVariationType">
  <simpleContent>
    <extension base="string">
      <attribute name="styleProperty" type="string" use="required"/>
      <attribute name="featurePropertyRange" type="string" use="optional"/>
    </extension>
  </simpleContent>
</complexType>
```

It has two attributes: `styleProperty` and `featurePropertyRange`. The value of the `styleProperty` is an SVG styling attribute name, such as “stroke”, “fill”, etc. It specifies what attribute of the style the property sets or overrides. The value of the `styleVariation` element is the value of the styling attribute specified by the `styleProperty`. The value may be a constant expression or an XPath expression.

The `featurePropertyRange` attribute defines the subset of features to which the variation applies. Its value is an XPath expression.

EXAMPLE The following shows two variations of the symbol style for a City feature. The feature is styled using a circle symbol. The radius of the circle depends on the population of the city, and is also calculated differently depending whether the population of the city is greater or less than 2 million.

```
<gml:FeatureStyle featureType="exp:City">
  <gml:GeometryStyle>
    <gml:styleVariation
      styleProperty="r"
      featurePropertyRange="population &gt;= 2000000">population div 1000000</gml:styleVariation>
    <gml:styleVariation
      styleProperty="r"
      featurePropertyRange="population &lt; 2000000">population div 1000000</gml:styleVariation>
    <gml:symbol xlink:href="http://www.opengis.org/symbols/City.xml#City"/>
  </gml:GeometryStyle>
</gml:FeatureStyle>
```

H.7.4 spatialResolution

The value of the `gml:spatialResolution` property element is a gml:MeasureType. In GML default styling, the meaning of this element is based on the corresponding definition in ISO 19115, where it is defined as a factor that provides a general understanding of the density of spatial data in the data set. Other than this informal definition, GML does not specify the exact use of this attribute. Application developers can use `gml:spatialResolution` in different ways.

EXAMPLE 1 It can be used as a map scale denominator (1:50,000, 1:25000, etc.).

EXAMPLE 2 Applications can also use its value to determine how to draw features in different scales. For example, a city and its features are typically drawn in more detail on a large scale map, and perhaps only as a single symbol on a small scale map or a coastline can be drawn in detail on a large scale map, while a small scale map application can omit some coordinates for better performance.

H.7.5 animation

Animation attributes are used to describe the animation behaviour of the geometry, topology, label or graph. These attributes are defined in W3C SMIL (SMIL 2.0 BasicAnimation Elements), see Table H.1.

Table 23 — Attributes used for animation

Attribute	Used for
animate	Generic attribute animation
animateMotion	Moving an element along the path
animateColor	Animating colour attributes
set	Setting the value of an attribute for a specified duration

H.8 Graph style

The `gml:graphStyle` property of the `gml:FeatureStyle` descriptor has as its value the `gml:GraphStyle` descriptor which describes style attributes of a graph formed by a set of features. The definitions of the graph style property and descriptor are shown in the following listing:

```
<element name="graphStyle" type="gml:GraphStylePropertyType"/>

<complexType name="GraphStylePropertyType">
	<sequence minOccurs="0">
		<element ref="gml:GraphStyle"/>
	</sequence>
	<attribute name="about" type="anyURI"/>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>

<element name="GraphStyle" type="gml:GraphStyleType" substitutionGroup="gml:AbstractGML"/>

<complexType name="GraphStyleType">
	<complexContent>
		<extension base="gml:BaseStyleDescriptorType">
			<sequence>
				<element name="planar" type="boolean" minOccurs="0"/>
				<element name="directed" type="boolean" minOccurs="0"/>
				<element name="grid" type="boolean" minOccurs="0"/>
				<element name="minDistance" type="double" minOccurs="0"/>
				<element name="minAngle" type="double" minOccurs="0"/>
				<element name="graphType" type="gml:GraphTypeType" minOccurs="0"/>
				<element name="drawingType" type="gml:DrawingTypeType" minOccurs="0"/>
				<element name="lineType" type="gml:LineTypeType" minOccurs="0"/>
				<element name="aestheticCriteria" type="gml:AesheticCriteriaType"
					minOccurs="0" maxOccurs="unbounded"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>

<simpleType name="GraphTypeType">
	<restriction base="string">
```



```
                <enumeration value="TREE"/>
                <enumeration value="BICONNECTED"/>
        </restriction>
</simpleType>

<simpleType name="DrawingTypeType">
        <restriction base="string">
                <enumeration value="POLYLINE"/>
                <enumeration value="ORTHOGONAL"/>
        </restriction>
</simpleType>

<simpleType name="LineTypeType">
        <restriction base="string">
                <enumeration value="STRAIGHT"/>
                <enumeration value="BENT"/>
        </restriction>
</simpleType>

<simpleType name="AesheticCriteriaType">
        <restriction base="string">
                <enumeration value="MIN_CROSSINGS"/>
                <enumeration value="MIN_AREA"/>
                <enumeration value="MIN_BENDS"/>
                <enumeration value="MAX_BENDS"/>
                <enumeration value="UNIFORM_BENDS"/>
                <enumeration value="MIN_SLOPES"/>
                <enumeration value="MIN_EDGE_LENGTH"/>
                <enumeration value="MAX_EDGE_LENGTH"/>
                <enumeration value="UNIFORM_EDGE_LENGTH"/>
                <enumeration value="MAX_ANGULAR_RESOLUTION"/>
                <enumeration value="MIN_ASPECT_RATIO"/>
                <enumeration value="MAX_SYMMETRIES"/>
        </restriction>
</simpleType>
```

The `gml:graphStyle` property is defined in the same manner as other GML properties which allow for referencing the value remotely or inline.

Graph style descriptor describes the style for a graph as a whole, not for individual graph elements. It inherits from the base content model common styling properties described in the section H.7.

This descriptor adds to the base content model a group of properties specific to graph styling — they describe the graph in terms of its specific characteristics. The properties are described in Table H.2.

Table 24 — Elements used in graph styling

Element	Type	Use
planar	boolean	If true, the graph edges do not cross (planar graph); if false they may cross
directed	boolean	If true the graph is directed; if false it is not directed
grid	boolean	If true, the coordinates of vertices, crossings and bends have integer values, otherwise they may have decimal values
minDistance	double	A recommendation for the minimum distance between vertices and non-incident edges
minAngle	double	A recommendation for the minimum angle between consecutive incident edges (angular resolution)
graphType	An enumeration	The type of the graph. The value may be TREE or BICONNECTED
drawingType	An enumeration	The type of the drawing with respect to the orthogonality of edges. The value may be POLYLINE or ORTHOGONAL
lineType	An enumeration	Determines whether there will be any bent edges. The value may be STRAIGHT or BENT
aestheticCriteria	An enumeration	A recommendation for the general outline of the graph in accordance with a particular aesthetic criteria. The value may be one of the following: MIN_CROSSINGS, MIN_AREA, MIN_BENDS, MAX_BENDS, UNIFORM_BENDS, MIN_SLOPES, MIN_EDGE_LENGTH, MAX_EDGE_LENGTH, UNIFORM_EDGE_LENGTH, MAX_ANGULAR_RESOLUTION, MIN_ASPECT_RATIO or MAX_SYMMETRIES

Annex I
(informative)

Backwards compatibility with earlier versions of GML

I.1 Overview

This annex specifies the deprecated schema components and their replacements (see 5.3).

I.2 Base schema components

I.2.1 remoteSchema

The attribute `remoteSchema` was provided to indicate a schema which constrains the description of the remote resource referenced by the xlink. The use of this attribute has been deprecated, `xlink:role` (see 8.1) may be used for the same purpose.

```
<attribute name="remoteSchema" type="anyURI"/>
```

I.2.2 member

A concrete property element named "member" was previously declared as follows:

```
<element name="member" type="gml:AssociationRoleType"/>
```

Property elements defined in an application schema shall be used instead.

I.2.3 ArrayAssociationType

For a property that will only be encoded inline, the property type pattern was explicitly encoded as:

```
<complexType name="ArrayAssociationType">
    <sequence>
        <element ref="gml:AbstractObject" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

This type has been replaced by derived types of `gml:AbstractMemberType` (see 7.2.5.1).

I.2.4 members

A concrete property element named "members" was previously declared as follows:

```
<element name="members" type="gml:ArrayAssociationType"/>
```

Property elements defined in an application schema shall be used instead.

I.2.5 featureProperty, featureMember, featureMembers

The concrete elements `gml:featureMember` *and* `gml:featureProperty` *used the* `gml:AssociationRoleType` *pattern in their content model, and were declared as follows:*

```
<element name="featureMember" type="gml:FeaturePropertyType"/>
<element name="featureProperty" type="gml:FeaturePropertyType"/>
```

The concrete elements `gml:featureMembers` *contains an array of features, and was declared as follows:*

```
<element name="featureMembers" type="gml:FeatureArrayPropertyType"/>
```

These property elements have been superseded by elements defined in application schemas.

I.2.6 StringOrRefType

`gml:StringOrRefType` *is a type provided to contain extended text values. It is defined as follows:*

```
<complexType name="StringOrRefType">
    <simpleContent>
        <extension base="string">
            <attributeGroup ref="gml:AssociationAttributeGroup"/>
        </extension>
    </simpleContent>
</complexType>
```

The use of remote references in this type has been deprecated. This type was previously available wherever there was a need for a "text" type property. It is of string type, so the text can be included inline, but the value could also have been referenced remotely via an xlink:href attribute. If the remote reference was present, then the value obtained by traversing the link was considered to be the value.

To refer to a remote string value, the xlink:href attribute of an element of type gml:ReferenceType (see 7.2.3.7), e.g. gml:descriptionReference, shall be used instead.

I.2.7 Array, ArrayType, Bag, BagType

Two concrete collections of objects were provided, but have been deprecated. GML object collections shall be constructed in application schema as described in 7.2.5 instead.

A `gml:Bag` *was for general collections with no implication about the type, order or uniqueness of the member objects:*

```
<element name="Bag" type="gml:BagType" substitutionGroup="gml:AbstractGML"/>

<complexType name="BagType">
    <complexContent>
        <extension base="gml:AbstractGMLType">
            <sequence>
                <element ref="gml:member" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="gml:members" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

A `gml:Array` *was intended to be used for a collection whose member objects are of homogeneous type and where their order is significant:*

```
<element name="Array" type="gml:ArrayType" substitutionGroup="gml:AbstractGML"/>
```

```
<complexType name="ArrayType">
    <complexContent>
        <extension base="gml:AbstractGMLType">
            <sequence>
                <element ref="gml:members" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

I.2.8 metaDataProperty, MetaDataPropertyType, AbstractMetaData, AbstractMetaDataType

The schema components specified in this subclause are superseded by the schema components specified in 7.2.6.

This property contains or refers to a metadata package that contains metadata properties in an encoding used in a previous version of GML. This element has been deprecated and is superseded by elements whose content model is derived from `gml:AbstractMetadataPropertyType`. More detail is provided in 7.2.6.

```
<element name="metaDataProperty" type="gml:MetaDataPropertyType"/>

<complexType name="MetaDataPropertyType">
    <sequence minOccurs="0">
        <element ref="gml:AbstractMetaData"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
    <attribute name="about" type="anyURI"/>
</complexType>
```

The optional "about" attribute carries a URI which points to an element or range of elements, or other resource to which this metadata refers.

The value of the metaDataProperty is an abstract element `gml:AbstractMetaData` that acts as a placeholder for "any package of metadata properties", defined as follows:

```
<element name="AbstractMetaData" type="gml:AbstractMetaDataType" abstract="true"
    substitutionGroup="gml:AbstractObject"/>

<complexType name="AbstractMetaDataType" abstract="true" mixed="true">
    <sequence/>
    <attribute ref="gml:id"/>
</complexType>
```

I.2.9 GenericMetaData, GenericMetaDataType

For convenience, a generic concrete MetaData element was provided in a previous version of GML. This element has been deprecated and is superseded by the schema components specified in 7.2.6.

```
<element name="GenericMetaData" type="gml:GenericMetaDataType" substitutionGroup="gml:AbstractMetaData"/>

<complexType name="GenericMetaDataType" mixed="true">
    <complexContent mixed="true">
        <extension base="gml:AbstractMetaDataType">
            <sequence>
                <any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

I.3 Basic types, Null

gml:Null is superseded by nillable and nilReason attributes as specified in 8.2.3.2.

The gml:Null element was declared as follows:

```
<element name="Null" type="gml:NilReasonType"/>
```

EXAMPLE This element might appear in data instance documents as follows:

```
<gml:Null>withheld</gml:Null>

<gml:Null>http://my.big.org/explanations/theDogAteIt</gml:Null>
```

The first example uses one of the built-in values for Null. The second example contains a reference to an explanation available elsewhere, identified by a URI.

The purpose in providing the gml:Null element was as follows: In order to construct a content model where a value may be omitted, the cardinality constraint expressed in XML Schema using the construction `minOccurs="0"` might be used. However, this approach carries the risk that the reason for the value not being present may be misinterpreted. As an alternative the element `gml:Null` may be included as a member of a `choice` group, alongside an element of the data type of a "normal" value.

EXAMPLE The content model described by the schema fragment

```
<element name="footprint">
  <complexType>
    <choice>
      <element ref="gml:Envelope"/>
      <element ref="gml:Null"/>
    </choice>
  </complexType>
</element>
```

allows either of the following data instances to be valid:

```
<footprint>
  <gml:Envelope> … </gml:Envelope>
</footprint>

<footprint>
  <gml:Null>inapplicable</gml:Null>
</footprint>
```

This allows the hypothetical element "footprint" to appear in an instance document, optionally containing an explicit marker indicate why it has no value, instead of having semantics inferred from the absence of a value.

I.4 Features

I.4.1 location, LocationPropertyType, LocationKeyWord, LocationString

The `gml:location` element was a convenience property that described the generalized location of the feature. It was defined as follows:

```
<element name="location" type="gml:LocationPropertyType"/>

<complexType name="LocationPropertyType">
  <sequence >
```



```
      <choice>
        <element ref="gml:AbstractGeometry"/>
        <element ref="gml:LocationKeyWord"/>
        <element ref="gml:LocationString"/>
        <element ref="gml:Null"/>
      </choice>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
```

The value of a location may be a geometry, a location string, a location keyword, or a null.

`gml:location` *and* `gml:LocationPropertyType` *have been deprecated.*

NOTE The flexible content model of the location property has proven to be difficult to implement in practice.

A location string is text which should describe the location. It was declared as follows:

```
<element name="LocationString" type="gml:StringOrRefType"/>
```

`gml:LocationKeyWord` *has been deprecated and is superseded by* `gml:locationName`*.*

The location keyword is a code usually selected from a controlled list. It was declared as follows:

```
<element name="LocationKeyWord" type="gml:CodeType"/>
```

`gml:LocationString` *has been deprecated and is superseded by* `gml:locationReference` *and* `gml:locationName (see` *9.4.2*`).`

I.4.2 priorityLocation, priorityLocationType

A property `gml:priorityLocation` *was provided for GML application schema developers that wish to provide prioritized locations for their features. A* `gml:priorityLocation` *has the following content model:*

```
<element name="priorityLocation" type="gml:PriorityLocationPropertyType"
      substitutionGroup="gml:location"/>

<complexType name="PriorityLocationPropertyType">
  <complexContent>
    <extension base="gml:LocationPropertyType">
      <attribute name="priority" type="string" />
    </extension>
  </complexContent>
</complexType>
```

Note that this simply adds a priority string to the base `gml:location` *property to assign levels of importance to the different locations.*

I.4.3 BoundedFeatureType

A simple restriction of `gml:AbstractFeatureType` *was previously offered making the optional boundedBy property mandatory.* `gml:BoundedFeatureType` *was defined as follows:*

```
<complexType name="BoundedFeatureType" abstract="true">
    <complexContent>
        <restriction base="gml:AbstractFeatureType">
            <sequence>
                <group ref="gml:StandardObjectProperties"/>
```

```
                    <element ref="gml:boundedBy"/>
                    <element ref="gml:location" minOccurs="0"/>
                </sequence>
            </restriction>
        </complexContent>
    </complexType>
```

I.4.4 AbstractFeatureCollectionType, AbstractFeatureCollection, FeatureCollection, FeatureCollectionType

GML feature collections in previous versions of GML were derived by extension or restriction from `gml:AbstractFeatureCollectionType`, defined as follows:

```
<complexType name="AbstractFeatureCollectionType" abstract="true">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <element ref="gml:featureMember" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="gml:featureMembers" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The `gml:featureMember` property (but not the `gml:featureMembers` property) follows the association pattern and may thus refer to a "remote" feature by means of the `xlink:ref` attribute.

The compositing property `gml:featureMembers` encloses a set of members of the Feature Collection regardless of their semantic type as features. `gml:featureMember` encloses or references a single feature instance. `gml:featureMember` and `gml:featureMembers` properties may appear on the same Feature Collection, but there may be only one `gml:featureMembers` property.

GML feature collections are themselves valid GML features and may have `gml:location` and other properties as defined in their GML application schema.

```
    <element name="AbstractFeatureCollection" type="gml:AbstractFeatureCollectionType" abstract="true"
    substitutionGroup="gml:AbstractFeature"/>
```

This abstract element `gml:AbstractFeatureCollection` serves as the head of a substitution group which may contain any elements whose content model is derived from `gml:AbstractFeatureType`. This may be used as a variable in the construction of content models.

The schema also provides a concrete feature collection:

```
<element name="FeatureCollection" type="gml:FeatureCollectionType" substitutionGroup="gml:AbstractFeature"/>

<complexType name="FeatureCollectionType">
  <complexContent>
    <extension base="gml:AbstractFeatureCollectionType"/>
  </complexContent>
</complexType>
```

Users of the concrete `gml:FeatureCollection` should note that it allows any valid GML feature as a member.

The content model of a GML feature collection in previous versions of GML was derived from `gml:AbstractFeatureCollectionType`. This in turn derives from `gml:AbstractFeatureType`. Hence feature collections are features, and are in general substitutable for `gml:AbstractFeature`.

The schema components specified in this subclause are deprecated and superseded by the rules for GML feature collections specified in 9.9.1.

I.4.5 Spatial properties

In general the definition of feature properties is the responsibility of the application schema designer. GML previously had defined a set of predefined spatial property elements to associate instances of these spatial types with features. These have been deprecated, application schema specific property names shall be used instead.

a) ***Descriptive names*** that provide a set of property names that are often used in application schemas. These are:

```
<element name="centerOf" type="gml:PointPropertyType"/>
<element name="position" type="gml:PointPropertyType"/>
<element name="extentOf" type="gml:SurfacePropertyType"/>
<element name="edgeOf" type="gml:CurvePropertyType"/>
<element name="centerLineOf" type="gml:CurvePropertyType"/>

<element name="multiLocation" type="gml:MultiPointPropertyType"/>
<element name="multiCenterOf" type="gml:MultiPointPropertyType"/>
<element name="multiPosition" type="gml:MultiPointPropertyType"/>
<element name="multiCenterLineOf" type="gml:MultiCurvePropertyType"/>
<element name="multiEdgeOf" type="gml:MultiCurvePropertyType"/>
<element name="multiCoverage" type="gml:MultiSurfacePropertyType"/>
<element name="multiExtentOf" type="gml:MultiSurfacePropertyType"/>
```

These property elements provide common role names for the geometry of geographic features. However, the specific semantics of these role names is not defined.

b) ***Formal names*** *that denote spatial properties in a manner based on the type of geometry or topology allowed as a property value. These are names based on the name of the spatial type with a suffix “Property”. These property types are usually defined for use within the GML schema itself. They shall not be used for property elements in application schemas. All formal names that are not used within the GML schema itself are deprecated:*

```
<element name="topoComplexProperty" type="gml:TopoComplexPropertyType"/>
<element name="multiPointProperty" type="gml:MultiPointPropertyType"/>
<element name="multiCurveProperty" type="gml:MultiCurvePropertyType"/>
<element name="multiSurfaceProperty" type="gml:MultiSurfacePropertyType"/>
<element name="multiSolidProperty" type="gml:MultiSolidPropertyType"/>
<element name="multiGeometryProperty" type="gml:MultiGeometryPropertyType"/>
<element name="pointArrayProperty" type="gml:PointArrayPropertyType"/>
<element name="curveArrayProperty" type="gml:CurveArrayPropertyType"/>
<element name="surfaceArrayProperty" type="gml:SurfaceArrayPropertyType"/>
<element name="solidArrayProperty" type="gml:SolidArrayPropertyType"/>
```

The specific semantics of these role names (e.g. "What does multiPointProperty of an object mean?") is not defined.

I.5 Coordinate geometry, geometric primitives

I.5.1 coordinates

```
<element name="coordinates" type="gml:CoordinatesType" />
```

The `gml:coordinates` *element is deprecated and replaced by* `gml:posList` *(see 10.1.4.2).*

I.5.2 pos in EnvelopeType

The properties `gml:lowerCorner` *and* `gml:upperCorner` *within* `gml:EnvelopeType` *shall be used instead (see* 10.1.4.6*).*

I.5.3 pointRep

```
<element name="pointRep" type="gml:PointPropertyType"/>
```

This property element has been deprecated. Use `gml:pointProperty` *instead, see 10.3.2.*

I.5.4 polygonPatches

```
<element name="polygonPatches" type="gml:SurfacePatchArrayPropertyType" substitutionGroup="gml:patches"/>
```

`gml:polygonPatches` *encapsulates the polygon patches of the polyhedral surface.* `gml:patches` *shall be used instead.*

I.5.5 trianglePatches

```
<element name="trianglePatches" type="gml:SurfacePatchArrayPropertyType" substitutionGroup="gml:patches"/>
```

`gml:trianglePatches` *encapsulates the triangles of the triangulated surface.* `gml:patches` *shall be used instead.*

I.6 Coordinate reference systems

I.6.1 baseGeographicCRS

```
<element name="baseGeographicCRS" type="gml:GeographicCRSPropertyType"/>
```

`gml:baseGeographicCRS` *is an association role to the geographic coordinate reference system used by this projected CRS. This property element is deprecated and replaced by* `gml:baseGeodeticCRS` *(see 12.3.3.15).*

I.6.2 GeographicCRS

```
<element name="GeographicCRS" type="gml:GeographicCRSType" substitutionGroup="gml:AbstractSingleCRS"/>

<complexType name="GeographicCRSType">
	<complexContent>
		<extension base="gml:AbstractCRSType">
			<sequence>
				<element ref="gml:usesEllipsoidalCS"/>
				<element ref="gml:usesGeodeticDatum"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:GeographicCRS` *is a coordinate reference system based on an ellipsoidal approximation of the geoid; this provides an accurate representation of the geometry of geographic features for a large portion of the Earth's surface.* `gml:GeographicCRS` *is deprecated and replaced by* `gml:GeodeticCRS` *(see 12.3.3.4).*

```
<complexType name="GeographicCRSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:GeographicCRS"/>
```

```
		</sequence>
		<attributeGroup ref="gml:AssociationAttributeGroup"/>
	</complexType>
```

`gml:GeographicCRSPropertyType` is a property type for association roles to a geographic coordinate reference system, either referencing or containing the definition of that reference system. This property type has been deprecated and replaced by `gml:GeodeticCRSPropertyType` (see 12.3.3.9).

I.6.3 GeocentricCRS

```
<element name="GeocentricCRS" type="gml:GeocentricCRSType" substitutionGroup="gml:AbstractSingleCRS"/>

<complexType name="GeocentricCRSType">
	<complexContent>
		<extension base="gml:AbstractCRSType">
			<sequence>
				<choice>
					<element ref="gml:usesCartesianCS"/>
					<element ref="gml:usesSphericalCS"/>
				</choice>
				<element ref="gml:usesGeodeticDatum"/>
			</sequence>
		</extension>
	</complexContent>
</complexType>
```

`gml:GeocentricCRS` is a 3D coordinate reference system with the origin at the approximate centre of mass of the Earth. A geocentric CRS deals with the Earth's curvature by taking a 3D spatial view. `gml:GeocentricCRS` is deprecated and replaced by `gml:GeodeticCRS` (see 12.3.3.4).

```
<complexType name="GeocentricCRSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:GeocentricCRS"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:GeocentricCRSPropertyType` is a property type for association roles to a geocentric coordinate reference system, either referencing or containing the definition of that reference system. This property type has been deprecated and replaced by `gml:GeodeticCRSPropertyType` (see 12.3.3.9).

I.6.4 uom

```
<attribute name="uom" type="anyURI"/>
```

The `uom` attribute provides an identifier of the unit of measure and has been deprecated. Local `uom` attributes shall be used instead.

I.6.5 ObliqueCartesianCS

```
<element name="ObliqueCartesianCS" type="gml:ObliqueCartesianCSType"
substitutionGroup="gml:AbstractCoordinateSystem"/>

<complexType name="ObliqueCartesianCSType">
	<complexContent>
		<extension base="gml:AbstractCoordinateSystemType"/>
	</complexContent>
</complexType>
```


gml:ObliqueCartesianCS is a two- or three-dimensional coordinate system with straight axes that are not necessarily orthogonal. An ObliqueCartesianCS shall have two or three gml:usesAxis associations. This element and type have been deprecated and are replaced by gml:AffineCS and its type (see 12.4.4.19).

```
<complexType name="ObliqueCartesianCSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:ObliqueCartesianCS"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

gml:ObliqueCartesianCSPropertyType is a property type for association roles to an oblique-Cartesian coordinate system, either referencing or containing the definition of that coordinate system. This property type has been deprecated and is replaced by gml:AffineCSPropertyType.

I.6.6 TemporalCS

```
<element name="TemporalCS" type="gml:TemporalCSType" substitutionGroup="gml:AbstractCoordinateSystem"/>

<complexType name="TemporalCSType">
	<complexContent>
		<extension base="gml:AbstractCoordinateSystemType"/>
	</complexContent>
</complexType>
```

gml:TemporalCS is a one-dimensional coordinate system containing a single time axis, used to describe the temporal position of a point in the specified time units from a specified time origin. A TemporalCS shall have one gml:usesAxis property element. This element and type have been deprecated and are replaced by gml:TimeCS and its type (see 12.4.4.7).

```
<complexType name="TemporalCSPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:TemporalCS"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

gml:TemporalCSPropertyType is a property type for association roles to a temporal coordinate system, either referencing or containing the definition of that coordinate system. This property type has been deprecated.

I.6.7 greenwichLongitude

The use of the deprecated gml:AngleChoiceType in gml:greenwichLongitude (see 12.5.3.7) has been removed.

I.6.8 AbstractOperation

```
<element name="AbstractOperation" type="gml:AbstractCoordinateOperationType" abstract="true"
 substitutionGroup="gml:AbstractSingleOperation"/>
```

gml:AbstractOperation is a parameterized mathematical operation on coordinates that transforms or converts coordinates to another coordinate reference system. This coordinate operation uses an operation method, usually with associated parameter values. However, operation methods and parameter values are directly associated with concrete subtypes, not with this abstract type. A GML application schema shall not extend or restrict this abstract complexType. This element has been deprecated and replaced by gml:AbstractSingleOperation (see 12.6.2.7).

```
<complexType name="OperationPropertyType">
	<sequence minOccurs="0">
		<element ref="gml:AbstractOperation"/>
	</sequence>
	<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

`gml:OperationPropertyType` is a property type for association roles to an abstract operation, either referencing or containing the definition of that operation. This property type has been deprecated and replaced by `gml:AbstractSingleOperationPropertyType` (see 12.6.2.8).

I.6.9 dmsAngleValue

```
<element name="dmsAngleValue" type="gml:DMSAngleType"/>
```

`gml:dmsAngleValue` is a value of an angle operation parameter, in either degree-minute-second format or single value format. This property element has been deprecated.

I.6.10 Renamed property elements

Previous versions of GML contained property elements that used a different naming convention than recommended in ISO/TS 19103 and used elsewhere in GML or because the property name in the underlying conceptual model has been changed. These property elements have been deprecated, and the property elements that are mentioned in the substitution group shall be used instead.

```
<element name="methodFormula" type="gml:CodeType" substitutionGroup="gml:formula"/>
<element name="anchorPoint" type="gml:CodeType" substitutionGroup="gml:anchorDefinition"/>
<element name="generalOperationParameter" type="gml:AbstractGeneralOperationParameterPropertyType"
	substitutionGroup="gml:parameter"/>
<element name="valueOfParameter" type="gml:OperationParameterPropertyType"
	substitutionGroup="gml:operationParameter"/>
<element name="valuesOfGroup" type="gml:OperationParameterGroupPropertyType"
	substitutionGroup="gml:group"/>
<element name="includesParameter" type="gml:AbstractGeneralOperationParameterPropertyType"
	substitutionGroup="gml:parameter"/>
<element name="definedByConversion" type="gml:GeneralConversionPropertyType"
	substitutionGroup="gml:conversion"/>
<element name="includesSingleCRS" type="gml:SingleCRSPropertyType"
	substitutionGroup="gml:componentReferenceSystem"/>
<element name="usesEllipsoidalCS" type="gml:EllipsoidalCSPropertyType"
	substitutionGroup="gml:ellipsoidalCS"/>
<element name="usesCartesianCS" type="gml:CartesianCSPropertyType" substitutionGroup="gml:cartesianCS" />
<element name="usesSphericalCS" type="gml:SphericalCSPropertyType" substitutionGroup="gml:sphericalCS"/>
<element name="usesGeodeticDatum" type="gml:GeodeticDatumPropertyType"
	substitutionGroup="gml:geodeticDatum"/>
<element name="usesVerticalCS" type="gml:VerticalCSPropertyType"
	substitutionGroup="gml:verticalCS"/>
<element name="usesVerticalDatum" type="gml:VerticalDatumPropertyType"
	substitutionGroup="gml:verticalDatum"/>
<element name="usesCS" type="gml:CoordinateSystemPropertyType" substitutionGroup="gml:coordinateSystem" />
<element name="usesEngineeringDatum" type="gml:EngineeringDatumPropertyType"
	substitutionGroup="gml:engineeringDatum"/>
<element name="usesAffineCS" type="gml:AffineCSPropertyType"
	substitutionGroup="gml:affineCS"/>
<element name="usesImageDatum" type="gml:ImageDatumPropertyType" substitutionGroup="gml:imageDatum"/>
<element name="usesObliqueCartesianCS" type="gml:ObliqueCartesianCSPropertyType"/>
<element name="usesTimeCS" type="gml:TimeCSPropertyType" substitutionGroup="gml:timeCS"/>
<element name="usesTemporalCS" type="gml:TemporalCSPropertyType"/> (use gml:timeCS instead)
<element name="usesTemporalDatum" type="gml:TemporalDatumPropertyType"
```

```
        substitutionGroup="gml:temporalDatum"/>
    <element name="usesAxis" type="gml:CoordinateSystemAxisPropertyType" substitutionGroup="gml:axis"/>
    <element name="usesPrimeMeridian" type="gml:PrimeMeridianPropertyType"
        substitutionGroup="gml:primeMeridian"/>
    <element name="usesEllipsoid" type="gml:EllipsoidPropertyType" substitutionGroup="gml:ellipsoid"/>
    <element name="usesSingleOperation" type="gml:CoordinateOperationPropertyType"
        substitutionGroup="gml:coordOperation"/>
    <element name="usesOperation" type="gml:CoordinateOperationPropertyType"
        substitutionGroup="gml:coordOperation"/>
    <element name="usesMethod" type="gml:OperationMethodPropertyType" substitutionGroup="gml:method"/>
    <element name="usesValue" type="gml:AbstractGeneralParameterValuePropertyType"
        substitutionGroup="gml:parameterValue"/>
    <element name="usesParameter" type="gml:AbstractGeneralOperationParameterPropertyType"
        substitutionGroup="gml:generalOperationParameter"/>
    <element name="includesValue" type="gml:AbstractGeneralParameterValueType"
    substitutionGroup="gml:parameterValue"/>
```

I.6.11 …Ref property elements

Previous versions of GML contained predefined property elements in the coordinate reference system schema documents. These property elements have been deprecated, property elements should be specified in the application schema.

```
<element name="crsRef" type="gml:CRSPropertyType"/>
<element name="singleCRSRef" type="gml:SingleCRSPropertyType"/>
<element name="compoundCRSRef" type="gml:CompoundCRSPropertyType"/>
<element name="verticalCRSRef" type="gml:VerticalCRSPropertyType"/>
<element name="projectedCRSRef" type="gml:ProjectedCRSPropertyType"/>
<element name="derivedCRSRef" type="gml:DerivedCRSPropertyType"/>
<element name="engineeringCRSRef" type="gml:EngineeringCRSPropertyType"/>
<element name="temporalCRSRef" type="gml:TemporalCRSPropertyType"/>
<element name="imageCRSRef" type="gml:ImageCRSPropertyType"/>
<element name="geocentricCRSRef" type="gml:GeocentricCRSPropertyType"/>
<element name="coordinateSystemAxisRef" type="gml:CoordinateSystemAxisPropertyType"/>
<element name="coordinateSystemRef" type="gml:CoordinateSystemPropertyType"/>
<element name="ellipsoidalCSRef" type="gml:EllipsoidalCSPropertyType"/>
<element name="cartesianCSRef" type="gml:CartesianCSPropertyType"/>
<element name="temporalCSRef" type="gml:TemporalCSPropertyType"/>
<element name="linearCSRef" type="gml:LinearCSPropertyType"/>
<element name="userDefinedCSRef" type="gml:UserDefinedCSPropertyType"/>
<element name="sphericalCSRef" type="gml:SphericalCSPropertyType"/>
<element name="polarCSRef" type="gml:PolarCSPropertyType"/>
<element name="cylindricalCSRef" type="gml:CylindricalCSPropertyType"/>
<element name="datumRef" type="gml:DatumPropertyType"/>
<element name="geodeticDatumRef" type="gml:GeodeticDatumPropertyType"/>
<element name="ellipsoidRef" type="gml:EllipsoidPropertyType"/>
<element name="primeMeridianRef" type="gml:PrimeMeridianPropertyType"/>
<element name="engineeringDatumRef" type="gml:EngineeringDatumPropertyType"/>
<element name="imageDatumRef" type="gml:ImageDatumPropertyType"/>
<element name="verticalDatumRef" type="gml:VerticalDatumPropertyType"/>
<element name="temporalDatumRef" type="gml:TemporalDatumPropertyType"/>
<element name="coordinateOperationRef" type="gml:CoordinateOperationPropertyType"/>
<element name="singleOperationRef" type="gml:SingleOperationPropertyType"/>
<element name="referenceSystemRef" type="gml:CRSPropertyType"/>
<element name="generalConversionRef" type="gml:GeneralConversionPropertyType"/>
<element name="generalTransformationRef" type="gml:GeneralTransformationPropertyType"/>
<element name="concatenatedOperationRef" type="gml:ConcatenatedOperationPropertyType"/>
<element name="passThroughOperationRef" type="gml:PassThroughOperationPropertyType"/>
<element name="conversionRef" type="gml:ConversionPropertyType"/>
<element name="transformationRef" type="gml:TransformationPropertyType"/>
```

```
<element name="abstractGeneralOperationParameterRef"
    type="gml:AbstractGeneralOperationParameterPropertyType"/>
<element name="operationParameterGroupRef" type="gml:OperationParameterPropertyType"/>
<element name="operationParameterRef" type="gml:OperationParameterPropertyType"/>
<element name="operationMethodRef" type="gml:OperationMethodPropertyType"/>
<element name="obliqueCartesianCSRef" type="gml:ObliqueCartesianCSPropertyType"/>
<element name="verticalCSRef" type="gml:VerticalCSPropertyType"/>
<element name="operationRef" type="gml:OperationPropertyType"/>
```

I.7 Temporal information and dynamic features

I.7.1 SuccessionType

A temporal non-linear graph is a network composed of time edges. This provides a framework for describing successions of feature instances or feature property values described by “substitution”, “division”, “fusion” and "initiation". In support of application schemas that require this capability, the `gml:SuccessionType` was provided in previous versions of GML to ensure a consistent terminology. However, since it was an isolated concept the type has been deprecated. It is defined as follows:

```
<simpleType name="SuccessionType">
  <restriction base="string">
    <enumeration value="substitution"/>
    <enumeration value="division"/>
    <enumeration value="fusion"/>
    <enumeration value="initiation"/>
  </restriction>
</simpleType>
```

I.7.2 MovingObjectStatus

`gml:MovingObjectStatus` *is one example of how* `gml:AbstractTimeSlice` *may be extended. This element provides a method to capture a record of the status of a moving object. It is declared as follows:*

```
<element name="MovingObjectStatus" type="gml:MovingObjectStatusType" substitutionGroup="gml:AbstractTimeSlice"/>
```



```
<complexType name="MovingObjectStatusType">
  <complexContent>
    <extension base="gml:AbstractTimeSliceType">
      <sequence>
        <choice>
          <element name="position" type="gml:GeometryPropertyType"/>
          <element ref="gml:pos"/>
          <element ref="gml:locationName"/>
          <element ref="gml:locationReference"/>
          <element ref="gml:location"/>
        </choice>
        <element name="speed" type="gml:MeasureType" minOccurs="0"/>
        <element name="bearing" type="gml:DirectionPropertyType" minOccurs="0"/>
        <element name="acceleration" type="gml:MeasureType" minOccurs="0"/>
        <element name="elevation" type="gml:MeasureType" minOccurs="0"/>
        <element ref="gml:status" minOccurs="0"/>
        <element ref="gml:statusReference" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

A `gml:MovingObjectStatus` element allows the user to describe the present location, along with the speed, bearing, acceleration and elevation of an object in a particular time slice.

Additional information about the current status of the object may be recorded in the `gml:status` or `gml:statusReference` property elements, declared as follows:

```
<element name="status" type="gml:StringOrRefType"/>
<element name="statusReference" type="gml:ReferenceType"/>
```

I.7.3 track

The `gml:track` property element has been deprecated, gml:history (see 14.5.7) should be used instead. It is declared in the schema as follows:

```
<element name="track" type="gml:HistoryPropertyType" substitutionGroup="gml:history"/>
```

I.8 Definitions and dictionaries

I.8.1 DefinitionCollection

```
<element name="DefinitionCollection" type="gml:DictionaryType" substitutionGroup="gml:Definition"/>
```

The alias for dictionaries, gml:DefinitionCollection, has been deprecated, gml:Dictionary (see 15.2.2) shall be used instead.

For remote definition references gml:dictionaryEntry shall be used, gml:indirectEntry has been deprecated. If a Definition object contained within a Dictionary uses the descriptionReference property to refer to a remote definition, then this enables the inclusion of a remote definition in a local dictionary, giving a handle and identifier in the context of the local dictionary.

I.8.2 definitionMember

```
<element name="definitionMember" type="gml:DictionaryEntryType" substitutionGroup="gml:dictionaryEntry"/>
```

The alias gml:definitionMember has been deprecated, gml:dictionaryEntry shall be used instead (see 15.2.3).

I.8.3 indirectEntry, IndirectEntryType, DefinitionProxy, DefinitionProxyType

If a definition is to be included by reference, in its context within the current collection, then the deprecated `gml:indirectEntry` property element was to be used in previous versions of GML. This element is declared as follows:

```
<element name="indirectEntry" type="gml:IndirectEntryType"/>

<complexType name="IndirectEntryType">
  <sequence>
    <element ref="gml:DefinitionProxy"/>
  </sequence>
</complexType>

<element name="DefinitionProxy" type="gml:DefinitionProxyType" substitutionGroup="gml:Definition"/>

<complexType name="DefinitionProxyType">
  <complexContent>
    <extension base="gml:DefinitionType">
      <sequence>
        <element ref="gml:definitionRef"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

A `gml:DefinitionProxy` carries a mandatory handle (`gml:id`), and contains a reference to a definition represented elsewhere. This entry is expected to be convenient in allowing multiple elements in one XML document to contain short (abbreviated XPointer) references, which are resolved to an external definition provided in a Dictionary element in the same XML document.

The reference is carried by a `gml:definitionRef` element which is declared as follows:

```
<element name="definitionRef" type="gml:ReferenceType"/>
```

This uses the `gml:ReferenceType`. The remote entry referenced may be in a dictionary in the same or different XML document.

I.9 Units, measures and values

I.9.1 dmsAngle

The `gml:dmsAngle` element was used to record the value of an angle in degree-minute-second or degree-minute format, but has been deprecated including all dependant schema component, because for machine-to-machine communication decimal degrees shall be used (see 16.3.3). It uses the following schema declarations:

```
<element name="dmsAngle" type="gml:DMSAngleType"/>

<complexType name="DMSAngleType">
  <sequence>
    <element ref="gml:degrees"/>
    <choice minOccurs="0">
      <element ref="gml:decimalMinutes"/>
      <sequence>
        <element ref="gml:minutes"/>
        <element ref="gml:seconds" minOccurs="0"/>
      </sequence>
    </choice>
```



```
  </sequence>
</complexType>

<element name="degrees" type="gml:DegreesType"/>

<complexType name="DegreesType">
  <simpleContent>
    <extension base="gml:DegreeValueType">
      <attribute name="direction">
        <simpleType>
          <restriction base="string">
            <enumeration value="N"/>
            <enumeration value="E"/>
            <enumeration value="S"/>
            <enumeration value="W"/>
            <enumeration value="+"/>
            <enumeration value="-"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </simpleContent>
</complexType>

<simpleType name="DegreeValueType">
  <restriction base="nonNegativeInteger">
    <maxInclusive value="359"/>
  </restriction>
</simpleType>

<element name="decimalMinutes" type="gml:DecimalMinutesType"/>

<simpleType name="DecimalMinutesType">
  <restriction base="decimal">
    <minInclusive value="0.00"/>
    <maxExclusive value="60.00"/>
  </restriction>
</simpleType>

<element name="minutes" type="gml:ArcMinutesType"/>

<simpleType name="ArcMinutesType">
  <restriction base="nonNegativeInteger">
    <maxInclusive value="59"/>
  </restriction>
</simpleType>

<element name="seconds" type="gml:ArcSecondsType"/>

<simpleType name="ArcSecondsType">
  <restriction base="decimal">
    <minInclusive value="0.00"/>
    <maxExclusive value="60.00"/>
  </restriction>
</simpleType>
```

I.9.2 degrees

*The **degrees** element allows an integer number of degrees with identification of the angle direction. This element is intended to be used within geographic positions, and has an XML attribute **direction** that may take values*

"N" or "S" for Latitude, meaning North or South of the equator;

"E" or "W" for Longitude, meaning East or West of the prime meridian;

"+" or "-" for other angles, in the specified rotational direction from a specified reference direction.

I.9.3 decimalMinutes

Decimal number of arc-minutes for use within a degree-minute angular value.

I.9.4 minutes

Integer number of arc-minutes for use within a degree-minute-second angular value.

I.9.5 seconds

Number of arc-seconds for use within a degree-minute-second angular value.

I.9.6 AngleChoiceType

To support the choice of either encoding for angles in a content model, a convenience type `gml:AngleChoiceType` is provided. This element contains another element, either an angle or a dmsAngle. It is declared in the schema as follows:

```
<complexType name="AngleChoiceType">
  <choice>
    <element ref="gml:angle"/>
    <element ref="gml:dmsAngle"/>
  </choice>
</complexType>
```

I.10 Directions

The properties `gml:horizontalAngle` and `gml:verticalAngle` in `gml:DirectionVectorType` (see 17.3) have been deprecated and superseded by `gml:vector` (see 10.1.4.5).

I.11 Coverages

I.11.1 MappingRule

`gml:CoverageMappingRule` (see 19.3.12) is the replacement for the deprecated `gml:MappingRule`:

```
<element name="MappingRule" type="gml:StringOrRefType"/>
```

I.11.2 IncrementOrder

The deprecated `gml:order` property has the content model (limited to 2-dimensional coverages):

```
<simpleType name="IncrementOrder">
  <restriction base="string">
    <enumeration value="+x+y"/>
    <enumeration value="+y+x"/>
    <enumeration value="+x-y"/>
    <enumeration value="-x-y"/>
```

```
  </restriction>
</simpleType>
```

The enumeration value here indicates the incrementing order to be used on the first 2 axes, i.e. "+x-y" means that the points on the first axis are to be traversed from lowest to highest and the points on the second axis are to be traversed from highest to lowest. The points on all other axes (if any) beyond the first 2 are assumed to increment from lowest to highest.

If the order attribute is omitted it is assumed to have the value "+x+y".

The element has been superseded by `gml:axisOrder` (see 19.3.14).

I.11.3 Domain set properties

```
<element name="multiPointDomain" type="gml:DomainSetType" substitutionGroup="gml:domainSet"/>
<element name="multiCurveDomain" type="gml:DomainSetType" substitutionGroup="gml:domainSet"/>
<element name="multiSurfaceDomain" type="gml:DomainSetType" substitutionGroup="gml:domainSet"/>
<element name="multiSolidDomain" type="gml:DomainSetType" substitutionGroup="gml:domainSet"/>
<element name="gridDomain" type="gml:DomainSetType" substitutionGroup="gml:domainSet"/>
<element name="rectifiedGridDomain" type="gml:DomainSetType" substitutionGroup="gml:domainSet"/>
```

These properties have been deprecated, because all previous uses of derivation-by-restriction in property elements have been removed. See the note in 21.2.6. `gml:domainSet` shall be used instead.

Annex J
(informative)

Modularization and dependencies

The GML schema described in this International Standard has been modularized informatively in Annex C to help creating profiles where only a topical subset of GML is required for a specific application. For example, a GML 2.1 application schema migrating to this International Standard without adding any new capabilities could define a profile that includes the schema document feature.xsd. Such a profile would not contain the new ISO 19136 definitions for coordinate reference systems, topology, coverages, dynamic features, and observations. However, it would contain of all of the basic types that have been added since GML 2.1.

The default modularization of the GML schema creates the dependencies among the GML base schemas shown in Figure J.1 below. A dashed arrow in the figure indicates that the schema at the tail of the arrow depends upon the schema at the head of the arrow. A dependency may occur where one schema <include>s another schema in the “gml” namespace. For example, feature.xsd <include>s geometryBasic2d.xsd. A dependency may also occur where one schema <import>s another schema for a namespace other than “gml”, for example, gmlBase.xsd <import>s xlinks.xsd from the “xlink” namespace.

There are now seven schema documents in GML upon which no other GML schema documents depend. These top-level schemas are the roots of partially overlapping hierarchies of GML schema documents:

— observation.xsd

— dynamicFeature.xsd

— coverage.xsd

— topology.xsd

— defaultStyle.xsd

— coordinateReferenceSystems.xsd

— temporalReferenceSystems.xsd.

A profile that needs definitions from more than one of these GML topical subset schema hierarchies may use a custom top-level schema document that contains contain multiple <include>s for just the appropriate schema documents, thereby excluding unwanted GML type definitions.

However, when an application schema will be used in a processing environment that lacks CPU, memory and/or I/O bandwidth, for example, in a mobile hand-held device, an absolutely minimal import of GML schema components is often desired. The custom top-level schema document approach described above might bring in an unacceptably large number of unwanted definitions from each GML schema included in the custom top-level schema document of the GML profile. The solution is to create a single GML subset schema that contains exactly the required GML type and element definitions. However, creating such a GML subset schema by hand using a text or XML editor to cut and paste definitions is a tedious and error-prone process because it involves analyzing type definition dependencies across the many GML schema documents. An automated approach is recommended instead. An informative sample implementation of a GML schema subset tool is included in Annex G. Subset schemas, however they are produced, are profiles of GML as described in Clause 20.

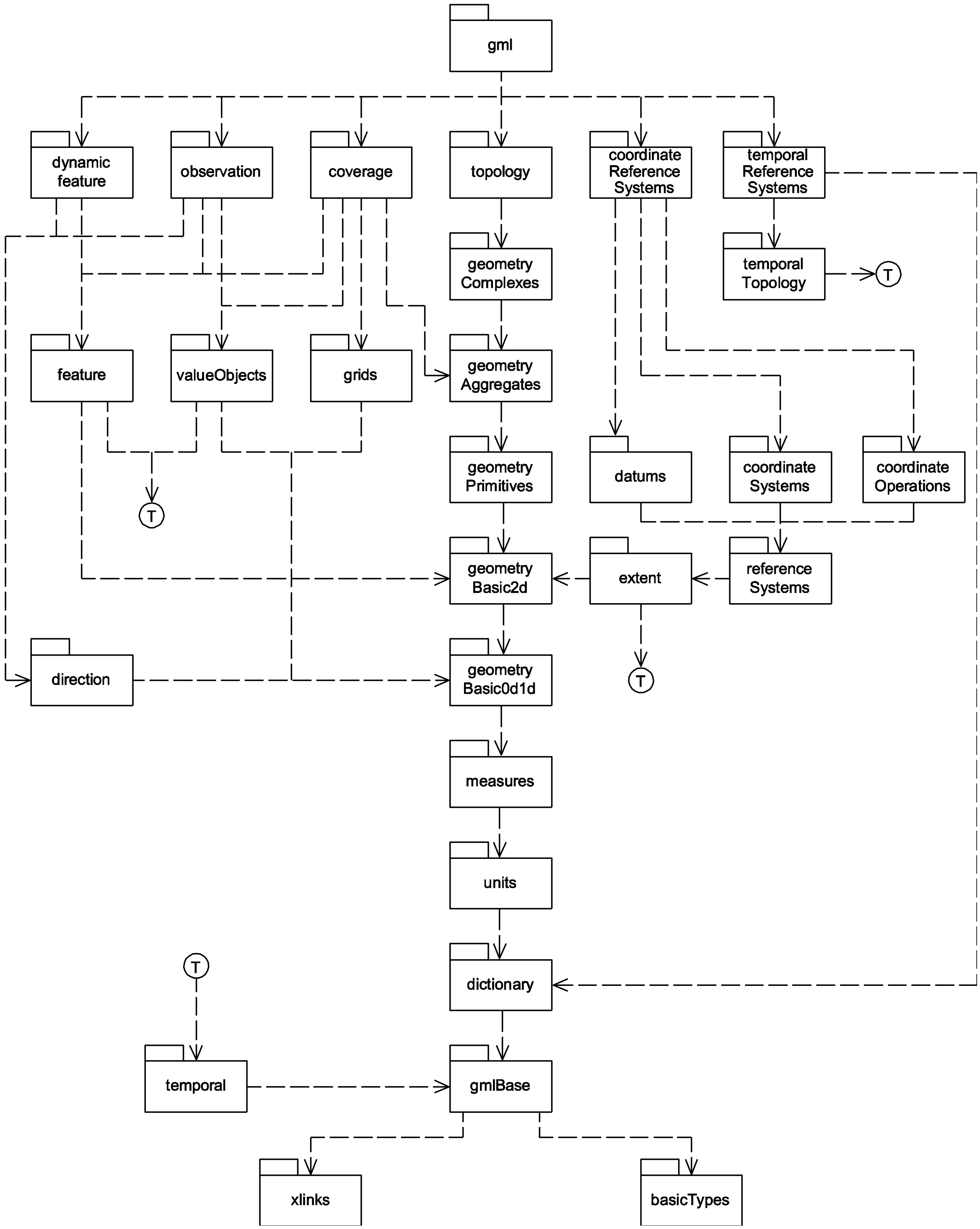


Figure 82 — Schema dependencies

Bibliography

[1] ISO 1000, *SI units and recommendations for the use of their multiples and of certain other units*

[2] ISO 8879, *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*

[3] ISO/IEC TR 10000-1:1998 *Information technology — Framework and taxonomy of International Standardized Profiles — Part 1: General principles and documentation framework*

[4] ISO 19101:2002, *Geographic information — Reference model*

[5] ISO 19105, *Geographic information — Conformance and testing*

[6] ISO 19106:2004, *Geographic information — Profiles*

[7] ISO 19110:2005, *Geographic information — Methodology for feature cataloguing*

[8] ISO 19117, *Geographic information — Portrayal*

[9] ISO 19133, *Geographic information — Location-based services — Tracking and navigation*

[10] ISO 19137:2007, *Geographic information — Core profile of the spatial schema*

[11] ISO 19141, *Geographic information — Schema for moving features*

[12] ISO/IEC 19501:2005, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

[13] Cover Pages: *Geography Markup Language (GML)*, available at <http://xml.coverpages.org/geographyML.html>

[14] LANGRAN, G. *Time in Geographic Information Systems.* London: Taylor & Francis Ltd. 1992

[15] KAUFMAN, M., and WAGNER, D. *Drawing Graphs,* Springer LNCS 2025, 1998

[16] BATTISTA, G., EADES, P., TAMASSIA, R., and TOLLIS, I. *Graph Drawing*, Prentice Hall 1999

[17] OpenGIS® Abstract Specification Topic 5, *The OpenGIS Feature*, OGC document 99-105r2

[18] OpenGIS® Abstract Specification Topic 6, *The Coverage Type*, OGC document 00-106

[19] OpenGIS® Abstract Specification Topic 8, *Relationships between Feature*, OGC document 99-108r2

[20] OpenGIS® Abstract Specification Topic 10, *Feature Collections*, OGC document 99-110

[21] OGC *Best Practice, Observations and Measurements*, OGC document 05-087r4

[22] UCUM, *Unified Code for Units of Measure*, Schadow, G., and McDonald, C.J. (eds.), available at <http://aurora.rg.iupui.edu/UCUM>

[23] W3C XML Base, *XML Base*, W3C Recommendation (27 June 2001)

[24] W3C XPath, *W3C XML Path Language (XPath) Version 1.0*, W3C Recommendation (16 November, 1999)

[25] W3C XPointer Framework, *XPointer Framework*, W3C Recommendation (25 March 2003)

[26] W3C XPointer element() Scheme, *XPointer element()*, W3C Recommendation (25 March 2003)

[27] W3C XPointer xmlns() Scheme, *XPointer xmlns()*, W3C Recommendation (25 March 2003)

[28] W3C XPointer xpointer() Scheme, *XPointer xpointer()*, W3C Working Draft (19 December 2002)

[29] W3C SMIL, *Synchronized Multimedia Integration Language (SMIL 2.0)*, W3C Recommendation (07 August 2001)

[30] W3C SVG, *Scalable Vector Graphics (SVG) 1*, W3C Recommendation (14 January 2003)

Index

