OGC GeoSPARQL - A Geographic Query Language for RDF Data

Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <yyyy-mm-dd>

External identifier of this OGC® document: http://www.opengis.net/doc/IS/geosparql/1.0

Internal reference number of this OGC® document: YY-DOC

Version: 1.0.1

Category: OGC® Implementation Specification

Category: OGC® Implementation Specification

Editors: <TBC>

OGC GeoSPARQL - A Geographic Query Language for RDF Data

Document type: OGC® Standard

Document subtype: Encoding

Document stage:Approved for Public Release

Document language: English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the

Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Table of Contents

Chapter 1. Preface

This standard defines a set of SPARQL extension functions [W3C SPARQL], a set of RIF rules [W3C RIF Core], and a core RDF/OWL [12,14,16] vocabulary for geographic information based on the General Feature Model [7], Simple Features [ISO 19125-1], Feature Geometry [6] and SQL MM [4].

Chapter 2. Submitting organizations

The following organizations submitted this Implementation Specification to the Open Geospatial Consortium Inc.:

a) Australian Bureau of Meteorology

b) Bentley Systems, Inc.

c) CSIRO

d) Defence Geospatial Information Working Group (DGIWG)

e) GeoConnections - Natural Resources Canada

f) Interactive Instruments GmbH

g) Oracle America

h) Ordnance Survey

1. Raytheon Company

j) Traverse Technologies, Inc.

k) US Geological Survey (USGS)

Chapter 3. Submission contact points

All questions regarding this submission should be directed to the editor or the submitters:

CONTACT	COMPANY
Matthew Perry	Oracle America
John Herring	Oracle America

Chapter 4. Revision history

Date	Release	Author	Paragraph modified	Description
27 Oct. 2009	Draft	Matthew Perry	Clause 6	Technical Draft
11 Nov. 2009	Draft	John R. Herring	All	Creation
6 Jan. 2010	Draft	John R. Herring	All	Comment responses
30 March 2010	Draft	Matthew Perry	All	Comment responses
26 Oct. 2010	Draft	Matthew Perry	All	Revision based on working group discussion
28 Jan. 2011	Draft	Matthew Perry	All	Revision based on working group discussion
18 April 2011	Draft	Matthew Perry	All	Restructure with multiple conformance classes
02 May 2011	Draft	Matthew Perry	Clause 6 and Clause 8	Move Geometry Class from core to geometryExtension
05 May 2011	Draft	Matthew Perry	All	Update URIs
13 Jan. 2012	Draft	Matthew Perry	All	Revision based on Public RFC
16 April 2012	Draft	Matthew Perry	All	Revision based on adoption vote comments
19 July 2012	1.0	Matthew Perry	All	Revision of URIs based on OGC Naming Authority recommendations

Chapter 5. Changes to the OGC® Abstract Specification

The OGC® Abstract Specification does not require changes to accommodate this OGC® standard.

Chapter 6. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights. However, to date, no such rights have been claimed or identified.

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 specification set forth in this document, and to provide supporting documentation.

Chapter 7. Introduction

The W3C Semantic Web Activity is defining a collection of technologies that enables a “web of data” where information is easily shared and reused across applications. Some key pieces of this technology stack are the RDF (Resource Description Framework) data model [12,14], the OWL Web Ontology Language [16] and the SPARQL protocol and RDF query language [W3C SPARQL].

RDF is, among other things, a data model built on edge-node "graphs." Each link in a graph consists of three things (with many aliases depending on the mapping from other types of data models):

- Subject (start node, instance, entity, feature)
- Predicate (verb, property, attribute, relation, member, link, reference)
- Object (value, end node, non-literal values can be used as a Subject)

Any of the three values in a single triple can be represented via a URI (with an optional fragment identifier). Subjects and objects are called nodes and can be represented as a blank node (usually with a local identifier with no meaning). Objects can also be represented as a literal value. Note that the same node may play the role of a Subject in some edges, and the role of the Object in others.

[RDF Triple] | *img/01.png*

Figure 1. RDF Triple

Almost all data can be presented or represented in RDF. In particular, it is an easy match to the (feature-instance-by-id, attribute, value) tuples of the General Feature Model [7], and for the relational model as (table primary key, column, value).

From http://dbpedia.org/page/SPARQL:

> SPARQL is an RDF query language; its name is a recursive acronym that stands for SPARQL Protocol and RDF Query Language. It was standardized by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium, and is considered a key semantic web technology. On 15 January 2008, SPARQL became an official W3C Recommendation. SPARQL allows [a] query to consist of triple patterns, conjunctions, disjunctions, and optional patterns.

SPARQL queries work on RDF representations of data by finding patterns that match templates in the query, in effect finding information graphs in the RDF data based on the templates and filters (constraints on nodes and edges) expressed in the query. This query template is represented in the SPARQL query by a set of parameterized “query variables” appearing in a sequence of RDF triples and filters. If the query processor finds a set of triples in the data (converted to an RDF graph in some predetermined standard manner) then the values that the “query variables” take on in those triples become a solution to the query request. The values of the variables are returned in the query result in a format based on the “SELECT” clause of the query (similar to SQL).

In addition to predicates defined in this manner, the SPARQL query may contain filter functions

that can be used to further constrain the query. Several mechanisms are available to extend filter functions to allow for predicates calculated directly on data values. The SPARQL specification [W3C SPARQL] in section 11.6 (http://www.w3.org/TR/rdf-sparql-query/#extensionFunctions) describes the mechanism for invocation of such a filter function.

The OGC GeoSPARQL standard supports representing and querying geospatial data on the Semantic Web. GeoSPARQL defines a vocabulary for representing geospatial data in RDF, and it defines an extension to the SPARQL query language for processing geospatial data.

The GeoSPARQL standard follows a modular design; it comprises several different components.

- A *core* component defines top-level RDFS/OWL classes for spatial objects.
- A *topology vocabulary* component defines RDF properties for asserting and querying topological relations between spatial objects.
- A *geometry* component defines RDFS data types for serializing geometry data, geometry-related RDF properties, and non-topological spatial query functions for geometry objects.
- A *geometry topology* component defines topological query functions.
- An *RDFS entailment* component defines a mechanism for matching implicit RDF triples that are derived based on RDF and RDFS semantics.
- A *query rewrite* component defines rules for transforming a simple triple pattern that tests a topological relation between two features into an equivalent query involving concrete geometries and topological query functions.

Each of the components described above forms a requirements class for GeoSPARQL. Implementations can provide various levels of functionality by choosing which requirements classes to support. For example, a system based purely on qualitative spatial reasoning may support only the core and topological vocabulary components.

In addition, GeoSPARQL is designed to accommodate systems based on qualitative spatial reasoning and systems based on quantitative spatial computations. Systems based on qualitative spatial reasoning, (e.g. those based on the Region Connection Calculus [1, 9]) do not usually model explicit geometries, so queries in such systems will likely test for binary spatial relationships between features rather than between explicit geometries. To allow queries for spatial relations between features in quantitative systems, GeoSPARQL defines a series of query transformation rules that expand a feature-only query into a geometry-based query. With these transformation rules, queries about spatial relations between features will have the same specification in both qualitative systems and quantitative systems. The qualitative system will likely evaluate the query with a backward-chaining spatial “reasoner”, and the quantitative system can transform the query into a geometry-based query that can be evaluated with computational geometry.

Chapter 8. OGC GeoSPARQL – A Geographic Query Language for RDF Data

8.1. Scope

This document defines GeoSPARQL: a spatial extension to the SPARQL query language for geographic information. GeoSPARQL provides the following features:

- An RDF/OWL vocabulary for representing spatial information consistent with the Simple Features model
- A set of SPARQL extension functions for spatial computations
- A set of RIF rules for query transformation

GeoSPARQL does not define a comprehensive vocabulary for representing spatial information. It instead defines a core set of classes, properties and datatypes that can be used to construct query patterns. Many useful extensions to this vocabulary are possible, and we intend for the GIS community to develop additional vocabulary for describing spatial information. Other standards groups are actively working to develop such a vocabulary (e.g. ISO/TC 211).

8.2. Conformance

Conformance with this specification shall be checked using all the relevant tests specified in Annex A (normative). The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in ISO 19105: Geographic information — Conformance and Testing [5].

This document establishes several requirements classes and corresponding conformance classes. Any GeoSPARQL implementation claiming conformance with one of the conformance classes shall pass all the tests in the associated abstract test suite. Requirements and conformance test URIs defined in this document are relative to `http://www.opengis.net/spec/geosparql/1.0/`. Table 1 summarizes the conformance classes in GeoSPARQL. Many conformance classes are parameterized. For parameterized conformance classes, the list of parameters is given within parenthesis.

Table 1. Conformance Classes

Conformance class	Description	Subclause of the abstract test suite
Core	Defines top-level spatial vocabulary components	A.1
Topology Vocabulary Extension (relation_family)	Defines topological relation vocabulary	A.2
Geometry Extension (serialization, version)	Defines geometry vocabulary and non-topological query functions	A.3

Geometry Topology Extension (serialization, version, relation_family)	Defines topological query functions for geometry objects	A.4
RDFS Entailment Extension (serialization, version , relation_family)	Defines a mechanism for matching implicit RDF triples that are derived based on RDF and RDFS semantics	A.5
Query Rewrite Extension (serialization, version, relation_family)	Defines query transformation rules for computing spatial relations between spatial objects based on their associated geometries	A.6

Dependencies between each GeoSPARQL requirements class are shown below in Figure 2. To support a requirements class for a given set of parameter values, an implementation must support each dependent requirements class with the same set of parameter values.

[02] | *img/02.png*

Figure 2. Requirements Class Dependency Graph

8.3. Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this document are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

ISO 19125-1, *Geographic information — Simple feature access — Part 1: Common architecture*

OGC 07-036, *Geography Markup Language (GML) Encoding Standard*, Version 3.2.1

RFC 2396, *RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax*

W3C RIF Core, *RIF Core Dialect*, W3C Recommendation (22 June 2010)

W3C SPARQL, *SPARQL Query Language for RDF*, W3C Recommendation (15 January 2008)

W3C SPARQL Entailment, *SPARQL 1.1 Entailment Regimes*, W3C Recommendation

W3C SPARQL Protocol, *SPARQL Protocol for RDF*, W3C Recommendation (15 January 2008)

W3C SPARQL Result Format, *SPARQL Query Results XML Format*, W3C Recommendation (15 January 2008)

8.4. Terms and definitions

For the purposes of this document, the terms and definitions given in the above references apply.

8.5. Conventions

8.5.1. Symbols (and abbreviated terms)

In this draft candidate standard, the following common acronyms are used:

GeoJSON	Geographic JavaScript Object Notation
GFM	General Feature Model (as defined in ISO 19109)
GML	Geography Markup Language
KML	Keyhole Markup Language
OWL	OWL 2 Web Ontology Language
RCC	Region Connection Calculus
RDF	Resource Description Framework
RDFS	RDF Schema
RIF	Rule Interchange Format
SPARQL	SPARQL Protocol and RDF Query Language
WKT	Well Known Text (as defined by Simple Features or ISO 19125)
W3C	World Wide Web Consortium (http://www.w3.org/)
XML	Extensible Markup Language

8.5.2. XML Namespaces

The following XML namespace prefixes are used throughout this document:

ogc:	http://www.opengis.net/
geo:	http://www.opengis.net/ont/geosparql#
geof:	http://www.opengis.net/def/function/geosparql/
geor:	http://www.opengis.net/def/rule/geosparql/
sf:	http://www.opengis.net/ont/sf#
gml:	http://www.opengis.net/ont/gml#
my:	http://example.org/ApplicationSchema#
xsd:	http://www.w3.org/2001/XMLSchema#
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
owl:	http://www.w3.org/2002/07/owl#

8.5.3. Placeholder URIs

The URI `ogc:geomLiteral` is used in requirement specifications as a placeholder for the geometry literal serialization used in a fully-qualified conformance class, e.g. `<http://www.opengis.net/ont/geosparql#wktLiteral>`. The URI `ogc:asGeomLiteral` is used in requirement specifications as a placeholder for the geometry literal serialization property used in a fully-qualified conformance class, e.g. `geo:asWKT`.

8.5.4. RDF Serializations

Two RDF serializations are used in this document. Terse RDF Triple Language (turtle) [10] is used for RDF snippets placed within the main body of the document, and RDF/XML [11] is used for the examples in Annex B.

8.6. Core

This clause establishes the **core** requirements class, with URI `/req/core`, which has a single corresponding conformance class, **core**, with URI `/conf/core`. This requirements class defines a set of classes and properties for representing geospatial data. The resulting vocabulary can be used to construct SPARQL graph patterns for querying appropriately modeled geospatial data. RDFS and OWL vocabulary have both been used so that the vocabulary can be understood by systems that support only RDFS entailment and by systems that support OWL-based reasoning.

8.6.1. SPARQL

Req 1 Implementations shall support the SPARQL Query Language for RDF [W3C SPARQL], the SPARQL Protocol for RDF [W3C SPARQL Protocol] and the SPARQL Query Results XML Format [W3C SPARQL Result Format].
`/req/core/sparql-protocol`

8.6.2. Classes

Two main classes are defined: `geo:SpatialObject` and `geo:Feature`. The class `geo:Feature` is equivalent to the UML class `GFI_Feature` defined in [8].

8.6.2.1. Class: geo:SpatialObject

The class `geo:SpatialObject` is defined by the following:

```
geo:SpatialObject a rdfs:Class,
                    owl:Class;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "Spatial Object"@en;
    rdfs:comment "The class Spatial Object represents everything that can
                 have a spatial representation. It is superclass of feature
                 and geometry"@en .
```

Req 2 Implementations shall allow the RDFS class `geo:SpatialObject` to be used in SPARQL graph patterns.
`/req/core/spatial-object-class`

8.6.2.2. Class: geo:Feature

The class `geo:Feature` is equivalent to the class `GFI_Feature` [8] and is defined by the following:

```
geo:Feature a rdfs:Class,
              owl:Class;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "Feature"@en;
    rdfs:subClassOf geo:SpatialObject;
    owl:disjointWith geo:Geometry;
    rdfs:comment "This class represents the top-level feature type. This
                  class is equivalent to GFI_Feature defined in ISO 19156,
                  and it is superclass of all feature types."@en .
```

Req 3 Implementations shall allow the RDFS class `geo:Feature` to be used in SPARQL graph patterns.
`/req/core/feature-class`

8.7. Topology Vocabulary Extension (relation_family)

This clause establishes the *Topology Vocabulary Extension (relation_family)* parameterized requirements class, with URI `/req/topology-vocab-extension`, which has a single corresponding conformance class *Topology Vocabulary Extension (relation_family)*, with URI `/conf/topology-vocab-extension`. This requirements class defines a vocabulary for asserting and querying topological relations between spatial objects. The class is parameterized so that different families of topological relations may be used, e.g. RCC8, Egenhofer. These relations are generalized so that they may connect features as well as geometries.

A DE-9IM pattern, which specifies the spatial dimension of the intersections of the interiors, boundaries and exteriors of two geometric objects, is used to describe each spatial relation. Possible pattern values are `-1` (empty), `0, 1, 2, T` (true) `= {0, 1, 2}, F` (false) = {-1}, *` (don't care)` = {-1, 0, 1, 2}`. In the following descriptions, the notation `X/Y` is used denote applying a spatial relation to geometry types `X` and `Y` (i.e., `x` *relation* `y` where `x` is of type `X` and `y` is of type `Y`). The symbol `P` is used for 0- dimensional geometries (e.g. points). The symbol `L` is used for 1-dimensional geometries (e.g. lines), and the symbol `A` is used for 2-dimensional geometries (e.g. polygons). Consult the Simple Features specification [ISO 19125-1] for a more detailed description of DE-9IM intersection patterns.

8.7.1. Parameters

The following parameter is defined for the *Topology Vocabulary Extension* requirements class.

relation_family: Specifies the set of topological spatial relations to support.

8.7.2. Requirements for Simple Features Relation Family (relation_family=Simple Features)

This clause defines requirements for the *Simple Features* relation family.

Req 4 Implementations shall allow the properties `geo:sfEquals`, `geo:sfDisjoint`, `geo:sfIntersects`, `geo:sfTouches`, `geo:sfCrosses`, `geo:sfWithin`, `geo:sfContains`, `geo:sfOverlaps` to be used in SPARQL graph patterns.
`/req/topology-vocab-extension/sf-spatial-relations`

Topological relations in the *Simple Features* family are summarized in Table 1. Multi- row intersection patterns should be interpreted as a logical `OR` of each row.

Table 2. Simple Features Topological Relations

Relation Name	Relation URI	Domain/Range	Applies To Geometry Types	DE-9IM Intersection Pattern
equals	`geo:sfEquals`	`geo:SpatialObject`	`All`	`(TFFFTFFFT)`
disjoint	`geo:sfDisjoint`	`geo:SpatialObject`	`All`	`(FF**FF****)`
intersects	`geo:sfIntersects`	`geo:SpatialObject`	`All`	`(T******** *T******* ***T***** ****T****)`
touches	`geo:sfTouches`	`geo:SpatialObject`	`All except P/P`	`(FT******* F**T***** F***T****)`
within	`geo:sfWithin`	`geo:SpatialObject`	`All`	`(T*F**F***)`
contains	`geo:sfContains`	`geo:SpatialObject`	`All`	`(T*****FF*)`
overlaps	`geo:sfOverlaps`	`geo:SpatialObject`	`A/A, P/P, L/L`	`(T*T***T**)` for `A/A, P/P`; `(1*T***T**)` for `L/L`
crosses	`geo:sfCrosses`	`geo:SpatialObject`	`P/L, P/A, L/A, L/L`	`(T*T***T**)` for `P/L, P/A, L/A`; `(0********)` for `L/L`

8.7.3. Requirements for Egenhofer Relation Family (relation_family=Egenhofer)

This clause defines requirements for the *Egenhofer* relation family. Consult references [2] and [3] for a more detailed discussion of *Egenhofer* relations.

Req 5 Implementations shall allow the properties `geo:ehEquals`, `geo:ehDisjoint`, `geo:ehMeet`, `geo:ehOverlap`, `geo:ehCovers`, `geo:ehCoveredBy`, `geo:ehInside`, `geo:ehContains` to be used in SPARQL graph patterns.

/req/topology-vocab-extension/eh-spatial-relations

Topological relations in the *Egenhofer* family are summarized in Table 2. Multi-row intersection patterns should be interpreted as a logical OR of each row.

Table 3. Egenhofer Topological Relations

Relation Name	Relation URI	Domain/Range	Applies To Geometry Types	DE-9IM Intersection Pattern
equals	geo:ehEquals	geo:SpatialObject	All	(TFFFTFFFT)
disjoint	geo:ehDisjoint	geo:SpatialObject	All	(FF*FF****)
meet	geo:ehMeet	geo:SpatialObject	All except P/P	(FT****** F**T***** F***T****)
overlap	geo:ehOverlap	geo:SpatialObject	All	(T*T***T**)
covers	geo:ehCovers	geo:SpatialObject	A/A, A/L, L/L	(T*TFT*FF*)
covered by	geo:ehCoveredBy	geo:SpatialObject	A/A, L/A, L/L	(TFF*TFT**)
inside	geo:ehInside	geo:SpatialObject	All	(TFF*FFT**)
contains	geo:ehContains	geo:SpatialObject	All	(T*TFF*FF*)

8.7.4. Requirements for RCC8 Relation Family (relation_family=RCC8)

This clause defines requirements for the *RCC8* relation family. Consult references [1] and [9] for a more detailed discussion of *RCC8* relations.

Req 6 Implementations shall allow the properties geo:rcc8eq, geo:rcc8dc, geo:rcc8ec, geo:rcc8po, geo:rcc8tppi, geo:rcc8tpp, geo:rcc8ntpp, geo:rcc8ntppi to be used in SPARQL graph patterns.

/req/topology-vocab-extension/rcc8-spatial-relations

Table 4. RCC8 Topological Relations

Relation Name	Relation URI	Domain/Range	Applies To Geometry Types	DE-9IM Intersection Pattern
equals	geo:rcc8eq	geo:SpatialObject	A/A	(TFFFTFFFT)
disconnected	geo:rcc8dc	geo:SpatialObject	A/A	(FFTFFTTTT)
externally connected	geo:rcc8ec	geo:SpatialObject	A/A	(FFTFTTTTT)
partially overlapping	geo:rcc8po	geo:SpatialObject	A/A	(TTTTTTTTT)
tangential proper part inverse	geo:rcc8tppi	geo:SpatialObject	A/A	(TTTFTTFFT)

Relation Name	Relation URI	Domain/Range	Applies To Geometry Types	DE-9IM Intersection Pattern
tangential proper part	geo:rcc8tpp	geo:SpatialObject	A/A	(TFFTTFTTT)
non-tangential proper part	geo:rcc8ntpp	geo:SpatialObject	A/A	(TFFTFFTTT)
non-tangential proper part inverse	geo:rcc8ntppi	geo:SpatialObject	A/A	(TTTFFTFFT)

8.7.5. Equivalent RCC8, Egenhofer and Simple Features Topological Relations

Table 4 summarizes the equivalences between *Egenhofer*, *RCC8* and *Simple Features* spatial relations for closed, non-empty regions. The symbol + denotes logical OR, and the symbol ¬ denotes negation.

Table 5. Equivalent Simple Features, RCC8 and Egenhofer relations

Simple Features	RCC8	Egenhofer
equals	equals	equals
disjoint	disconnected	disjoint
intersects	¬ disconnected	¬ disjoint
touches	externally connected	meet
within	non-tangential proper part + tangential proper part	inside + coveredBy
contains	non-tangential proper part inverse + tangential proper part inverse	contains + covers
overlaps	partially overlapping	overlap

8.8. Geometry Extension (serialization, version)

This clause establishes the *Geometry Extension (serialization, version)* parameterized requirements class, with URI /req/geometry-extension, which has a single corresponding conformance class *Geometry Extension (serialization, version)*, with URI /conf/geometry-extension. This requirements class defines a vocabulary for asserting and querying information about geometry data, and it defines query functions for operating on geometry data.

As part of the vocabulary, an RDFS datatype is defined for encoding detailed geometry information as a literal value. A literal representation of a geometry is needed so that geometric values may be treated as a single unit. Such a representation allows geometries to be passed to external functions for computations and to be returned from a query.

Other schemes for encoding simple geometry data in RDF have been proposed. The W3C Basic Geo vocabulary (http://www.w3.org/2003/01/geo/) is one popular vocabulary. These simple vocabularies have limitations, for example the inability to specify different datums and coordinate systems, and were therefore not used in GeoSPARQL. Note that most existing geometry data encoded using these vocabularies can easily be converted into GeoSPARQL representations. The SPARQL query below creates `geo:wktLiteral` values from W3C Basic Geo geometries.

```
PREFIX w3cGeo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geo: <http://www.opengis.net/#geosparql>

SELECT (STRDT(CONCAT("POINT(",?long," ",?lat,")"),
              geo:wktLiteral) AS ?wktLit)
WHERE { ?point w3cGeo:long ?long .
        ?point w3cGeo:lat ?lat }
```

8.8.1. Parameters

The following parameters are defined for the *Geometry Extension* requirements class.

serialization: Specifies the serialization standard to use when generating geometry literals and also the supported geometry types.

Note that the serialization chosen strongly affects the geometry conceptualization. The WKT serialization aligns the geometry types with ISO 19125 Simple Features [ISO 19125-1], and the GML serialization aligns the geometry types with ISO 19107 Spatial Schema [6].

version: Specifies the version of the serialization format used.

8.8.2. Geometry Classes and Properties

A single root geometry class is defined: `geo:Geometry`. The class `geo:Geometry` is equivalent to the UML class `GM_Object` defined in [6]. In addition, properties are defined for describing geometry data and for associating geometries with features.

8.8.2.1. Class: geo:Geometry

The class `geo:Geometry` is equivalent to `GM_Object` [6] and is defined by the following:

```
geo:Geometry a rdfs:Class,
               owl:Class;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "Geometry"@en;
    rdfs:subClassOf geo:SpatialObject;
    owl:disjointWith geo:Feature;
    rdfs:comment "The class represents the top-level geometry type. This class
                  is equivalent to the UML class GM_Object defined in ISO 19107,
                  and it is superclass of all geometry types."@en .
```

Req 7 Implementations shall allow the RDFS class `geo:Geometry` to be used in SPARQL graph patterns.
`/req/geometry-extension/geometry-class`

8.8.2.2. Standard Properties for geo:Feature

Properties are defined for associating geometries with features.

Req 8 Implementations shall allow the properties `geo:hasGeometry` and `geo:hasDefaultGeometry` to be used in SPARQL graph patterns.
`/req/geometry-extension/feature-properties`

Property: geo:hasGeometry

The property `geo:hasGeometry` is used to link a feature with a geometry that represents its spatial extent. A given feature may have many associated geometries.

```
geo:hasGeometry a rdf:Property,
                  owl:ObjectProperty;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "has Geometry"@en;
    rdfs:comment "A spatial representation for a given feature."@en;
    rdfs:domain geo:Feature;
    rdfs:range geo:Geometry .
```

8.8.2.3. Property: geo:hasDefaultGeometry

The property `geo:hasDefaultGeometry` is used to link a feature with its default geometry. The default geometry is the geometry that should be used for spatial calculations in the absence of a request for a specific geometry (e.g. in the case of query rewrite).

```
geo:hasDefaultGeometry a rdf:Property,
                           owl:ObjectProperty;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "has Default Geometry"@en;
    rdfs:comment "The default geometry to be used in spatial calculations,
                 usually the most detailed geometry."@en;
    rdfs:subPropertyOf geo:hasGeometry;
    rdfs:domain geo:Feature;
    rdfs:range geo:Geometry .
```

GeoSPARQL does not restrict the cardinality of the `geo:hasDefaultGeometry` property. It is thus possible for a feature to have more than one distinct default geometry or to have no default geometry. This situation does not result in a query processing error; SPARQL graph pattern matching simply proceeds as normal. Certain queries may, however, give logically inconsistent results. For example, if a feature my:f1 has two asserted default geometries, and those two geometries are disjoint polygons, the query below could return a non-zero count on a system

supporting the GeoSPARQL Query Rewrite Extension (rule `geor:sfDisjoint`).

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>

SELECT (COUNT(*) AS ?cnt)
WHERE { :f1 geo:sfDisjoint :f1 }
```

Such cases are application-specific data modeling errors and are therefore outside of the scope of the GeoSPARQL specification.

8.8.3. Standard Properties for geo:Geometry

Properties are defined for describing geometry metadata.

Req 9 Implementations shall allow the properties `geo:dimension`, `geo:coordinateDimension`, `geo:spatialDimension`, `geo:isEmpty`, `geo:isSimple`, `geo:hasSerialization` to be used in SPARQL graph patterns.
`/req/geometry-extension/geometry-properties`

8.8.3.1. Property: geo:dimension

The dimension is the topological dimension of this geometric object, which must be less than or equal to the coordinate dimension. In non-homogeneous collections, this will return the largest topological dimension of the contained objects.

```
geo:dimension a rdf:Property,
                owl:DatatypeProperty;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "dimension"@en;
    rdfs:comment "The topological dimension of this geometric object, which
                  must be less than or equal to the coordinate dimension. In
                  non-homogeneous collections, this is the largest
                  topological dimension of the contained objects."@en;
    rdfs:domain geo:Geometry;
    rdfs:range xsd:integer .
```

8.8.3.2. Property: geo:coordinateDimension

The coordinate dimension is the dimension of direct positions (coordinate tuples) used in the definition of this geometric object.

```
geo:coordinateDimension a rdf:Property,
                        owl:DatatypeProperty;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "coordinate dimension"@en;
    rdfs:comment "The number of measurements or axes needed to describe the
                 position of this geometry in a coordinate system."@en;
    rdfs:domain geo:Geometry;
    rdfs:range xsd:integer .
```

8.8.3.3. Property: geo:spatialDimension

The spatial dimension is the dimension of the spatial portion of the direct positions (coordinate tuples) used in the definition of this geometric object. If the direct positions do not carry a measure coordinate, this will be equal to the coordinate dimension.

```
geo:spatialDimension a rdf:Property,
                       owl:DatatypeProperty;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "spatial dimension"@en;
    rdfs:comment "The number of measurements or axes needed to describe the
                 spatial position of this geometry in a coordinate
                 system."@en;
    rdfs:domain geo:Geometry;
    rdfs:range xsd:integer .
```

Property: geo:isEmpty

The `geo:isEmpty` Boolean will be set to true only if the geometry contains no points.

```
geo:isEmpty a rdf:Property, owl:DatatypePropterty;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "is empty"@en;
    rdfs:comment "(true) if this geometric object is the empty Geometry. If
                 true, then this geometric object represents the empty point
                 set for the coordinate space."@en;
    rdfs:domain geo:Geometry;
    rdfs:range xsd:boolean .
```

Property: geo:isSimple

The `geo:isSimple` Boolean will be set to true, only if the geometry contains no self- intersections, with the possible exception of its boundary.

```
geo:isSimple a rdf:Property,
               owl:DatatypePropterty;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "is simple"@en;
    rdfs:comment "(true) if this geometric object has no anomalous geometric
                 points, such as self intersection or self tangency."@en;
    rdfs:domain geo:Geometry;
    rdfs:range xsd:boolean .
```

8.8.3.4. Property: geo:hasSerialization

The `geo:hasSerialization` property is used to connect a geometry with its text- based serialization (e.g., its WKT serialization).

```
geo:hasSerialization a rdf:Property,
                       owl:DatatypePropterty;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "has serialization"@en;
    rdfs:comment "Connects a geometry object with its text-based
                  serialization."@en;
    rdfs:domain geo:Geometry;
    rdfs:range rdfs:Literal .
```

8.8.4. Requirements for WKT Serialization (serialization=WKT)

This section establishes the requirements for representing geometry data in RDF based on WKT as defined by Simple Features [ISO 19125-1].

8.8.4.1. RDFS Datatypes

This section defines one RDFS Datatype: `http://www.opengis.net/ont/geosparql#wktLiteral`.

RDFS Datatype: geo:wktLiteral

```
geo:wktLiteral a rdfs:Datatype;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "Well-known Text Literal"@en;
    rdfs:comment "A Well-known Text serialization of a geometry object."@en .
```

Req 10 All RDFS Literals of type `geo:wktLiteral` shall consist of an optional URI identifying the coordinate reference system followed by Simple Features Well Known Text (WKT) describing a geometric value. Valid `geo:wktLiterals` are formed by concatenating a valid, absolute URI as defined in [RFC 2396], one or more spaces (Unicode U+0020 character) as a separator, and a WKT string as defined in Simple Features [ISO 19125-1].
`/req/geometry-extension/wkt-literal`

For geo:wktLiterals, the beginning URI identifies the spatial reference system for the geometry. The OGC maintains a set of CRS URIs under the http://www.opengis.net/def/crs/ namespace. This leading spatial reference system URI is optional. In the absence of a leading spatial reference system URI, the following spatial reference system URI will be assumed: <http://www.opengis.net/def/crs/OGC/1.3/CRS84> This URI denotes WGS 84 longitude-latitude.

Req 11 The URI <http://www.opengis.net/def/crs/OGC/1.3/CRS84> shall be assumed as the spatial reference system for `geo:wktLiteral`s that do not specify an explicit spatial reference system URI..
/req/geometry-extension/wkt-literal-default-srs

Req 12 Coordinate tuples within `geo:wktLiteral`s shall be interpreted using the axis order defined in the spatial reference system used.
/req/geometry-extension/wkt-axis-order

The example geo:wktLiteral below encodes a point geometry using the default WGS84 geodetic longitude-latitude spatial reference system for Simple Features 1.0:

```
"Point(-83.38 33.95)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
```

A second example below encodes the same point using <http://www.opengis.net/def/crs/EPSG/0/4326>: a WGS 84 geodetic latitude-longitude spatial reference system (note that this spatial reference system defines a different axis order):

```
"<http://www.opengis.net/def/crs/EPSG/0/4326>
Point(33.95 -83.38)"^^<http://www.opengis.net/ont/geosparql#wktLiteral>
```

Req 13 An empty RDFS Literal of type geo:wktLiteral shall be interpreted as an empty geometry.
/req/geometry-extension/wkt-literal-empty

8.8.4.2. Serialization Properties

The geo:asWKT property is defined to link a geometry with its WKT serialization.

Property: geo:asWKT

Req 14 Implementations shall allow the RDF property geo:asWKT to be used in SPARQL graph patterns.
/req/geometry-extension/geometry-as-wkt-literal

The property geo:asWKT is used to link a geometric element with its WKT serialization.

```
geo:asWKT a rdf:Property,
            owl:DatatypeProperty;
    rdfs:subPropertyOf geo:hasSerialization;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "as WKT"@en;
    rdfs:comment "The WKT serialization of a geometry."@en;
    rdfs:domain geo:Geometry;
    rdfs:range geo:wktLiteral .
```

8.8.5. Requirements for GML Serialization (serialization=GML)

This section establishes requirements for representing geometry data in RDF based on GML as defined by Geography Markup Language Encoding Standard [OGC 07-036].

8.8.5.1. RDFS Datatypes

This section defines one RDFS Datatype: `http://www.opengis.net/ont/geosparql#gmlLiteral`.

RDFS Datatype: geo:gmlLiteral

```
geo:gmlLiteral a rdfs:Datatype;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "GML literal"@en;
    rdfs:comment "The datatype of GML literal values"@en .
```

Valid `geo:gmlLiteral`s are formed by encoding geometry information as a valid element from the GML schema that implements a subtype of `GM_Object`. For example, in GML 3.2.1 this is every element directly or indirectly in the substitution group of the element `{http://www.opengis.net/ont/gml/3.2}AbstractGeometry`. In GML 3.1.1 and GML 2.1.2 this is every element directly or indirectly in the substitution group of the element `{http://www.opengis.net/ont/gml}_Geometry`.

Req 15 All `geo:gmlLiteral`s shall consist of a valid element from the GML schema that implements a subtype of `GM_Object` as defined in [OGC 07-036].
`/req/geometry-extension/gml-literal`

The example `geo:gmlLiteral` below encodes a point geometry in the WGS 84 geodetic longitude-latitude spatial reference system using GML version 3.2:

```
"<gml:Point
        srsName=\"http://www.opengis.net/def/crs/OGC/1.3/CRS84\"
        xmlns:gml=\"http://www.opengis.net/ont/gml\">
    <gml:pos>-83.38 33.95</gml:pos>
</gml:Point>"^^<http://www.opengis.net/ont/geosparql#gmlLiteral>
```

Req 16 An empty `geo:gmlLiteral` shall be interpreted as an empty geometry.

/req/geometry-extension/gml-literal-empty

Req 17 Implementations shall document supported GML profiles.
/req/geometry-extension/gml-profile

8.8.5.2. Serialization Properties

This document defines the geo:asGML property to link a geometry with its serialization.

Property: geo:asGML

Req 18 Implementations shall allow the RDF property geo:asGML to be used in SPARQL graph patterns.
/req/geometry-extension/geometry-as-gml-literal

The property geo:asGML is used to link a geometric element with its GML serialization.

```
geo:asGML a rdf:Property;
    rdfs:subPropertyOf geo:hasSerialization;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "as GML"@en;
    rdfs:comment "The GML serialization of a geometry."@en;
    rdfs:domain geo:Geometry;
    rdfs:range geo:gmlLiteral .
```

8.8.6. Requirements for GeoJSON Serialization (serialization=GEOJSON)

This section establishes the requirements for representing geometry data in RDF based on GeoJSON.

8.8.6.1. RDFS Datatypes

This section defines one RDFS Datatype: http://www.opengis.net/ont/geosparql#geoJSONLiteral.

RDFS Datatype: geo:geoJSONLiteral

```
geo:geoJSONLiteral a rdfs:Datatype;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.1>;
    rdfs:label "GeoJSON Literal"@en;
    rdfs:comment "A GeoJSON serialization of a geometry object."@en .
```

Valid `geo:geoJSONLiteral`s are formed by encoding geometry information as a Geometry object as defined in the GeoJSON specification [RFC 7946].

Req XX All `geo:geoJSONLiteral`s shall consist of the Geometry objects as defined in the GeoJSON specification [RFC 7946].

/req/geometry-extension/geoJSON-literal

Req XX All RDFS Literals of type `geo:geoJSONLiteral` do not contain a CRS definition. All literals of this type shall according to the GeoJSON specification only be encoded in and assumed to use the WGS84 geodetic longitude-latitude spatial reference system (urn:ogc:def:crs:OGC::CRS84).
/req/geometry-extension/geoJSON-literal-crs

The example `geo:geoJSONLiteral` below encodes a point geometry using the default WGS84 geodetic longitude-latitude spatial reference system for Simple Features 1.0:

```
"{\"type\":\"Point\", \"coordinates\":[-
83.38,33.95]}"^^<http://www.opengis.net/ont/geosparql#geoJSONLiteral>
```

Req XX An empty RDFS Literal of type `geo:geoJSONLiteral` shall be interpreted as an empty geometry, i.e. {"geometry":null} in GeoJSON .
/req/geometry-extension/geoJSON-literal-empty

8.8.6.2. Serialization Properties

The `geo:asGeoJSON` property is defined to link a geometry with its GeoJSON serialization.

Property: geo:asGeoJSON

Req XX Implementations shall allow the RDF property `geo:asGeoJSON` to be used in SPARQL graph patterns.
/req/geometry-extension/geometry-as-geojson-literal

The property `geo:asGeoJSON` is used to link a geometric element with its GeoJSON serialization.

```
geo:asGeoJSON a rdf:Property,
            owl:DatatypeProperty;
    rdfs:subPropertyOf geo:hasSerialization;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.1>;
    rdfs:label "as GeoJSON"@en;
    rdfs:comment "The GeoJSON serialization of a geometry."@en;
    rdfs:domain geo:Geometry;
    rdfs:range geo:geoJSONLiteral .
```

8.8.7. Requirements for KML Serialization (serialization=KML)

This section establishes the requirements for representing geometry data in RDF based on KML.

8.8.7.1. RDFS Datatypes

This section defines one RDFS Datatype: `http://www.opengis.net/ont/geosparql#kmlLiteral`.

RDFS Datatype: geo:kmlLiteral

```
geo:kmlLiteral a rdfs:Datatype;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.1>;
    rdfs:label "KML Literal"@en;
    rdfs:comment "A KML serialization of a geometry object."@en .
```

Valid `geo:kmlLiteral`s are formed by encoding geometry information as a Geometry object as defined in the KML specification [https://www.ogc.org/standards/kml/].

Req XX All `geo:kmlLiteral`s shall consist of the Geometry objects as defined in the KML specification [https://www.ogc.org/standards/kml/].
/req/geometry-extension/kml-literal

Req XX All RDFS Literals of type `geo:kmlLiteral` do not contain a CRS definition. All literals of this type shall according to the KML specification only be encoded in and assumed to use the WGS84 geodetic longitude-latitude spatial reference system (urn:ogc:def:crs:OGC::CRS84).
/req/geometry-extension/kml-literal-crs

The example `geo:kmlLiteral` below encodes a point geometry using the default WGS84 geodetic longitude-latitude spatial reference system for Simple Features 1.0:

```
"<Point xmlns=\"http://www.opengis.net/kml/2.2\"><coordinates>-
83.38,33.95</coordinates></Point>"^^<http://www.opengis.net/ont/geosparql#kmlLiteral>
```

Req XX An empty RDFS Literal of type `geo:kmlLiteral` shall be interpreted as an empty geometry .
/req/geometry-extension/kml-literal-empty

8.8.7.2. Serialization Properties

The `geo:asKML` property is defined to link a geometry with its KML serialization.

Property: geo:asKML

Req XX Implementations shall allow the RDF property `geo:asKML` to be used in SPARQL graph patterns.
/req/geometry-extension/geometry-as-kml-literal

The property `geo:asKML` is used to link a geometric element with its KML serialization.

```
geo:asKML a rdf:Property,
            owl:DatatypeProperty;
    rdfs:subPropertyOf geo:hasSerialization;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.1>;
    rdfs:label "as KML"@en;
    rdfs:comment "The KML serialization of a geometry."@en;
    rdfs:domain geo:Geometry;
    rdfs:range geo:kmlLiteral .
```

8.8.8. Non-topological Query Functions

This clause defines SPARQL functions for performing non-topological spatial operations.

Req 19 Implementations shall support `geof:distance`, `geof:buffer`, `geof:convexHull`, `geof:intersection`, `geof:union`, `geof:difference`, `geof:symDifference`, `geof:envelope` and `geof:boundary` as SPARQL extension functions, consistent with the definitions of the corresponding functions (`distance`, `buffer`, `convexHull`, `intersection`, `difference`, `symDifference`, `envelope` and `boundary` respectively) in Simple Features [ISO 19125-1].
`/req/geometry-extension/query-functions`

An invocation of any of the following functions with invalid arguments produces an error. An invalid argument includes any of the following:

- An argument of an unexpected type
- An invalid geometry literal value
- A geometry literal from a spatial reference system that is incompatible with the spatial reference system used for calculations
- An invalid units URI

For further discussion of the effects of errors during FILTER evaluation, consult Section 11 of the SPARQL specification [W3C SPARQL] (http://www.w3.org/TR/rdf-sparql- query/#tests).

Note that returning values instead of raising an error serves as an extension mechanism of SPARQL.

From Section 11.3.1 of the SPARQL specification [W3C SPARQL] (http://www.w3.org/TR/rdf-sparql-query/#operatorExtensibility):

> SPARQL language extensions may provide additional associations between operators and operator functions; this amounts to adding rows to the table above. No additional operator may yield a result that replaces any result other than a type error in the semantics defined above. The consequence of this rule is that SPARQL extensions will produce at least the same solutions as an unextended implementation, and may, for some queries, produce more solutions.

This extension mechanism is intended to allow GeoSPARQL implementations to simultaneously support multiple geometry serializations. For example, a system that supports `geo:wktLiteral` serializations may also support `geo:gmlLiteral` serializations and consequently would not raise an error if it encounters multiple geometry datatypes while processing a given query.

Several non-topological query functions use a unit of measure URI. The OGC has defined some standard units of measure URIs under the `http://www.opengis.net/def/uom/OGC/1.0/` namespace, for example `<http://www.opengis.net/def/uom/OGC/1.0/metre>`.

8.8.8.1. Function: geof:distance

```
geof:distance (geom1: ogc:geomLiteral, geom2: ogc:geomLiteral,
               units: xsd:anyURI): xsd:double
```

Returns the shortest distance in units between any two Points in the two geometric objects as calculated in the spatial reference system of `geom1`.

8.8.8.2. Function: geof:buffer

```
geof:buffer (geom: ogc:geomLiteral, radius: xsd:double,
             units: xsd:anyURI): ogc:geomLiteral
```

This function returns a geometric object that represents all Points whose distance from `geom1` is less than or equal to the `radius` measured in `units`. Calculations are in the spatial reference system of `geom1`.

8.8.8.3. Function: geof:convexHull

```
geof:convexHull (geom1: ogc:geomLiteral): ogc:geomLiteral
```

This function returns a geometric object that represents all Points in the convex hull of `geom1`. Calculations are in the spatial reference system of `geom1`.

8.8.8.4. Function: geof:intersection

```
geof:intersection (geom1: ogc:geomLiteral,
                   geom2: ogc:geomLiteral): ogc:geomLiteral
```

This function returns a geometric object that represents all Points in the intersection of `geom1` with `geom2`. Calculations are in the spatial reference system of `geom1`.

8.8.8.5. Function: geof:union

```
geof:union (geom1: ogc:geomLiteral, geom2: ogc:geomLiteral,
            ): ogc:geomLiteral
```

This function returns a geometric object that represents all Points in the union of `geom1` with `geom2`. Calculations are in the spatial reference system of `geom1`.

8.8.8.6. Function: geof:difference

```
geof:difference (geom1: ogc:geomLiteral, geom2: ogc:geomLiteral,
                 ): ogc:geomLiteral
```

This function returns a geometric object that represents all Points in the set difference of `geom1` with `geom2`. Calculations are in the spatial reference system of `geom1`.

8.8.8.7. Function: geof:symDifference

```
geof:symDifference (geom1: ogc:geomLiteral,
                    geom2: ogc:geomLiteral,
                    ): ogc:geomLiteral
```

This function returns a geometric object that represents all Points in the set symmetric difference of `geom1` with `geom2`. Calculations are in the spatial reference system of `geom1`.

8.8.8.8. Function: geof:envelope

```
geof:envelope (geom1: ogc:geomLiteral): ogc:geomLiteral
```

This function returns the minimum bounding box of `geom1`. Calculations are in the spatial reference system of `geom1`.

8.8.8.9. Function: geof:boundary

```
geof:boundary (geom1: ogc:geomLiteral): ogc:geomLiteral
```

This function returns the closure of the boundary of `geom1`. Calculations are in the spatial reference system of `geom1`.

Req 20 Implementations shall support `geof:getSRID` as a SPARQL extension function.
`/req/geometry-extension/srid-function`

8.8.8.10. Function: geof:getsrid

```
geof:getSRID (geom: ogc:geomLiteral): xsd:anyURI
```

Returns the spatial reference system URI for `geom`.

8.9. Geometry Topology Extension (relation_family, serialization, version)

This clause establishes the *Geometry Topology Extension (relation_family, serialization, version)* parameterized requirements class, with URI `/req/geometry-topology-extension`, which defines a collection of topological query functions that operate on geometry literals. This class is parameterized to give implementations flexibility in the topological relation families and geometry serializations that they choose to support. This requirements class has a single corresponding conformance class *Geometry Topology Extension (relation_family, serialization, version)*, with URI `/conf/geometry-topology-extension`.

The Dimensionally Extended Nine Intersection Model (DE-9IM) has been used to define the relation tested by the query functions introduced in this section. Each query function is associated with a defining DE-9IM intersection pattern. Possible pattern values are `-1` (empty), `0`, `1`, `2`, `T` (true) = `{0, 1, 2}`, `F` (false) = `{-1}`, `*` (don't care) = `{-1, 0, 1, 2}`. In the following descriptions, the notation `X/Y` is used denote applying a spatial relation to geometry types `X` and `Y` (i.e., `x` *relation* `y` where `x` is of type `X` and `y` is of type `Y`). The symbol `P` is used for 0-dimensional geometries (e.g. points). The symbol `L` is used for 1- dimensional geometries (e.g. lines), and the symbol `A` is used for 2-dimensional geometries (e.g. polygons). Consult the Simple Features specification [ISO 19125-1] for a more detailed description of DE-9IM intersection patterns.

8.9.1. Parameters

relation_family: Specifies the set of topological spatial relations to support.

serialization: Specifies the serialization standard to use for geometry literals.

version: Specifies the version of the serialization format used.

8.9.2. Common Query Functions

Req 21 Implementations shall support `geof:relate` as a SPARQL extension function, consistent with the relate operator defined in Simple Features [ISO 19125-1].
`/req/geometry-topology-extension/relate-query-function`

```
geof:relate (geom1: ogc:geomLiteral, geom2: ogc:geomLiteral,
             pattern-matrix: xsd:String): xsd:boolean
```

Returns `true` if the spatial relationship between `geom1` and `geom2` corresponds to one with acceptable values for the specified pattern-matrix. Otherwise, this function returns `false`. `pattern-matrix` represents a DE-9IM intersection pattern consisting of `T` (true) and `F` (false) values. The spatial

reference system for geom1 is used for spatial calculations.

8.9.3. Requirements for Simple Features Relation Family (relation_family=Simple Features)

This clause establishes requirements for the *Simple Features* relation family.

Req 22 Implementations shall support geof:sfEquals, geof:sfDisjoint, geof:sfIntersects, geof:sfTouches, geof:sfCrosses, geof:sfWithin, geof:sfContains, geof:sfOverlaps as SPARQL extension functions, consistent with their corresponding DE-9IM intersection patterns, as defined by Simple Features [ISO 19125-1].
/req/geometry-topology-extension/sf-query-functions

Boolean query functions defined for the Simple Features relation family, along with their associated DE-9IM intersection patterns, are shown in Table 5 below. Multi-row intersection patterns should be interpreted as a logical OR of each row. Each function accepts two arguments (geom1 and geom2) of the geometry literal *serialization* type *specified* by serialization and version. Each function returns an xsd:boolean value of true if the specified relation exists between geom1 and geom2 and returns false otherwise. In each case, the spatial reference system of geom1 is used for spatial calculations.

Table 6. Simple Features Query Functions

Query Function	Defining DE-9IM Intersection Pattern
geof:sfEquals(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean	(TFFFTFFFT)
geof:sfDisjoint(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean	(FF*FF**)
geof:sfIntersects(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean	(FT******* F**T***** F***T****)
geof:sfTouches(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean	(FT******* F**T***** F***T****)
geof:sfCrosses(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean	(T*T***T**) for P/L, P/A, L/A; (0*T***T**) for L/L
geof:sfWithin(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean	(T*F**F***)
geof:sfContains(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean	(T*****FF*)
geof:sfOverlaps(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean	(T*T***T**) for A/A, P/P; (1*T***T**) for L/L

8.9.4. Requirements for Egenhofer Relation Family (relation_family=Egenhofer)

This clause establishes requirements for the *Egenhofer* relation family. Consult references [2] and [3] for a more detailed discussion of *Egenhofer* relations.

Req 23 Implementations shall support `geof:ehEquals`, `geof:ehDisjoint`, `geof:ehMeet`, `geof:ehOverlap`, `geof:ehCovers`, `geof:ehCoveredBy`, `geof:ehInside`, `geof:ehContains` as SPARQL extension functions, consistent with their corresponding DE-9IM intersection patterns, as defined by Simple Features [ISO 19125-1].
`/req/geometry-topology-extension/eh-query-functions`

Boolean query functions defined for the *Egenhofer* relation family, along with their associated DE-9IM intersection patterns, are shown in Table 6 below. Multi-row intersection patterns should be interpreted as a logical OR of each row. Each function accepts two arguments (`geom1` and `geom2`) of the geometry literal serialization type specified by *serialization* and *version*. Each function returns an `xsd:boolean` value of `true` if the specified relation exists between `geom1` and `geom2` and returns `false` otherwise. In each case, the spatial reference system of geom1 is used for spatial calculations.

Table 7. Egenhofer Query Functions

Query Function	Defining DE-9IM Intersection Pattern
`geof:ehEquals(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(TFFFTFFFT)`
`geof:ehDisjoint(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(FF*FF****)`
`geof:ehMeet(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(FT******* F**T***** F***T****)`
`geof:ehOverlap(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(T*T***T**)`
`geof:ehCovers(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(T*TFT*FF*)`
`geof:ehCoveredBy(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(TFF*TFT**)`
`geof:ehInside(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(TFF*FFT**)`
`geof:ehContains(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(T*TFF*FF*)`

8.9.4.1. Requirements for RCC8 Relation Family (relation_family=RCC8)

This clause establishes requirements for the *RCC8* relation family. Consult references [1] and [9] for a more detailed discussion of *RCC8* relations.

Req 24 Implementations shall support `geof:rcc8eq`, `geof:rcc8dc`, `geof:rcc8ec`, `geof:rcc8po`, `geof:rcc8tppi`, `geof:rcc8tpp`, `geof:rcc8ntpp`, `geof:rcc8ntppi` as SPARQL extension functions, consistent with their corresponding DE-9IM intersection patterns, as defined by Simple Features [ISO 19125-1].
`/req/geometry-topology-extension/rcc8-query-functions`

Boolean query functions defined for the *RCC8* relation family, along with their associated DE-9IM intersection patterns, are shown in Table 7 below. Each function accepts two arguments (`geom1` and `geom2`) of the geometry literal serialization type specified by *serialization* and *version*. Each function returns an `xsd:boolean` value of `true` if the specified relation exists between `geom1` and `geom2` and

returns `false` otherwise. In each case, the spatial reference system of geom1 is used for spatial calculations.

Table 8. RCC8 Query Functions

Query Function	Defining DE-9IM Intersection Pattern
`geof:rcc8eq(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(TFFFTFFFT)`
`geof:rcc8dc(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(FFTFFTTTT)`
`geof:rcc8ec(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(FFTFTTTTT)`
`geof:rcc8po(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(TTTTTTTTT)`
`geof:rcc8tppi(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(TTTFTTFFT)`
`geof:rcc8tpp(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(TFFTTFTTT)`
`geof:rcc8ntpp(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(TFFTFFTTT)`
`geof:rcc8ntppi(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral ): xsd:boolean`	`(TTTFFTFFT)`

8.10. RDFS Entailment Extension (relation_family, serialization, version)

This clause establishes the *RDFS Entailment Extension (relation_family, serialization, version)* parameterized requirements class, with URI `/req/rdfs-entailment-extension`, which defines a mechanism for matching implicitly derived RDF triples in GeoSPARQL queries. This class is parameterized to give implementations flexibility in the topological relation families and geometry types that they choose to support. This requirements class has a single corresponding conformance class *RDFS Entailment Extension (relation_family, serialization, version)*, `with URI /conf/rdfs-entailment-extension`.

8.10.1. Parameters

relation_family: Specifies the set of topological spatial relations to support.

serialization: Specifies the serialization standard to use for geometry literals.

version: Specifies the version of the serialization format used.

8.10.2. Common Requirements

The basic mechanism for supporting RDFS entailment has been defined by the W3C SPARQL 1.1 RDFS Entailment Regime [W3C SPARQL Entailment].

Req 25 Basic graph pattern matching shall use the semantics defined by the RDFS Entailment Regime [W3C SPARQL Entailment].
/req/rdfs-entailment-extension/bgp-rdfs-ent

8.10.3. Requirements for WKT Serialization (serialization=WKT)

This section establishes the requirements for representing geometry data in RDF based on WKT as defined by Simple Features [ISO 19125-1].

8.10.3.1. Geometry Class Hierarchy

The Simple Features specification presents a geometry class hierarchy. It is straightforward to represent this class hierarchy in RDFS and OWL by constructing URIs for geometry classes using the following pattern: `http://www.opengis.net/ont/sf#{geometry class}` and by asserting appropriate `rdfs:subClassOf` statements.

The example RDF snippet below encodes the Polygon class from Simple Features 1.0.

```
sf:Polygon a rdfs:Class,
             owl:Class;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "Polygon"@en;
    rdfs:subClassOf sf:Surface;
    rdfs:comment "A planar surface defined by 1 exterior boundary and 0 or
                  more interior boundaries"@en .
```

Req 26 Implementations shall support graph patterns involving terms from an RDFS/OWL class hierarchy of geometry types consistent with the one in the specified version of Simple Features [ISO 19125-1].
/req/rdfs-entailment-extension/wkt-geometry-types

8.10.4. Requirements for GML Serialization (serialization=GML)

This section establishes requirements for representing geometry data in RDF based on GML as defined by Geography Markup Language Encoding Standard [OGC 07-036].

8.10.4.1. Geometry Class Hierarchy

An RDF/OWL class hierarchy can be generated from the GML schema that implements `GM_Object` by constructing URIs for geometry classes using the following pattern: `http://www.opengis.net/ont/gml#{GML Element}` and by asserting appropriate `rdfs:subClassOf` statements.

The example RDF snippet below encodes the Polygon class from GML 3.2.

```
gml:Polygon a rdfs:Class,
              owl:Class;
    rdfs:isDefinedBy <http://www.opengis.net/spec/geosparql/1.0>;
    rdfs:label "Polygon"@en;
    rdfs:subClassOf gml:SurfacePatch;
    rdfs:comment "A planar surface defined by 1 exterior boundary and 0 or
                  more interior boundaries."@en .
```

Req 27 Implementations shall support graph patterns involving terms from an RDFS/OWL class hierarchy of geometry types consistent with the GML schema that implements `GM_Object` using the specified *version* of GML [OGC 07-036].
`/req/rdfs-entailment-extension/gml-geometry-types`

8.11. Query Rewrite Extension (relation_family, serialization, version)

This clause establishes the *Query Rewrite Extension (relation_family, serialization, version)* parameterized requirements class, with URI `/req/query-rewrite-extension`, which has a single corresponding conformance class *Query Rewrite Extension (relation_family, serialization, version)*, with URI `/conf/query-rewrite-extension`. This requirements class defines a set of RIF rules [15] that use topological extension functions defined in Clause 9 to establish the existence of direct topological predicates defined in Clause 7. One possible implementation strategy is to transform a given query by expanding a triple pattern involving a direct spatial predicate into a series of triple patterns and an invocation of the corresponding extension function as specified in the RIF rule.

The following rule specified using the RIF Core Dialect [W3C RIF Core] is used as a template to describe rules in the remainder of this clause. `ogc:relation` is used as a placeholder for the spatial relation URIs defined in Clause 7, and `ogc:function` is used as a placeholder for the spatial functions defined in Clause 9.

```
Forall ?f1 ?f2 ?g1 ?g2 ?g1Serial ?g2Serial
    (?f1[ogc:relation->?f2] :-
        Or(
            And
                # feature  feature rule
                (?f1[geo:hasDefaultGeometry->?g1]
                 ?f2[geo:hasDefaultGeometry->?g2]
                 ?g1[ogc:asGeomLiteral->?g1Serial]
                 ?g2[ogc:asGeomLiteral->?g2Serial]
                 External(ogc:function (?g1Serial,?g2Serial)))
            And
                # feature  geometry rule
                (?f1[geo:hasDefaultGeometry->?g1]
                 ?g1[ogc:asGeomLiteral->?g1Serial]
                 ?f2[ogc:asGeomLiteral->?g2Serial]
                 External(ogc:function (?g1Serial,?g2Serial)))
            And
                # geometry - feature rule
                (?f2[geo:hasDefaultGeometry->?g2]
                 ?f1[ogc:asGeomLiteral->?g1Serial]
                 ?g2[ogc:asGeomLiteral->?g2Serial]
                 External(ogc:function (?g1Serial,?g2Serial)))
            And
                # geometry - geometry rule
                (?f1[ogc:asGeomLiteral->?g1Serial]
                 ?f2[ogc:asGeomLiteral->?g2Serial]
                 External(ogc:function (?g1Serial,?g2Serial)))
    )
)
```

8.11.1. Parameters

relation_family: Specifies the set of topological spatial relations to support.

serialization: Specifies the serialization standard to use for geometry literals.

version: Specifies the version of the serialization format used.

8.11.2. Requirements for Simple Features Relation Family (relation_family=Simple Features)

This clause defines requirements for the *Simple Features* relation family. Table 8 specifies the function and property substitutions for each rule in the *Simple Features* relation family.

Req 28 Basic graph pattern matching shall use the semantics defined by the RIF Core Entailment Regime [W3C SPARQL Entailment] for the RIF rules [W3C RIF Core] `geor:sfEquals`, `geor:sfDisjoint`, `geor:sfIntersects`, `geor:sfTouches`, `geor:sfCrosses`, `geor:sfWithin`, `geor:sfContains`, `geor:sfOverlaps`.
`/req/query-rewrite-extension/sf-query-rewrite`

Table 9. Simple Features Query Transformation Rules

Rule	ogc:relation	ogc:function
geor:sfEquals	geo:sfEquals	geof:sfEquals
geor:sfDisjoint	geo:sfDisjoint	geof:sfDisjoint
geor:sfIntersects	geo:sfIntersects	geof:sfIntersects
geor:sfTouches	geo:sfTouches	geof:sfTouches
geor:sfCrosses	geo:sfCrosses	geof:sfCrosses
geor:sfWithin	geo:sfWithin	geof:sfWithin
geor:sfContains	geo:sfContains	geof:sfContains
geor:sfOverlaps	geo:sfOverlaps	geof:sfOverlaps

8.11.3. Requirements for Egenhofer Relation Family (relation_family=Egenhofer)

Req 29 Basic graph pattern matching shall use the semantics defined by the RIF Core Entailment Regime [W3C SPARQL Entailment] for the RIF rules [W3C RIF Core] `geor:ehEquals`, `geor:ehDisjoint`, `geor:ehMeet`, `geor:ehOverlap`, `geor:ehCovers`, `geor:ehCoveredBy`, `geor:ehInside`, `geor:ehContains`.
`/req/query-rewrite-extension/eh-query-rewrite`

Table 10. Simple Features Query Transformation Rules

Rule	ogc:relation	ogc:function
geor:ehEquals	geo:ehEquals	geof:ehEquals
geor:ehDisjoint	geo:ehDisjoint	geof:ehDisjoint
geor:ehMeet	geo:ehMeet	geof:ehMeet
geor:ehOverlap	geo:ehOverlap	geof:ehOverlap
geor:ehCovers	geo:ehCovers	geof:ehCovers
geor:ehCoveredBy	geo:ehCoveredBy	geof:ehCoveredBy
geor:ehInside	geo:ehInside	geof:ehInside
geor:ehContains	geo:ehContains	geof:ehContains

8.11.4. Requirements for RCC8 Relation Family (relation_family=RCC8)

This clause defines requirements for the *RCC8* relation family. Table 10 specifies the function and property substitutions for each rule in the *RCC8* relation family.

Req 30 Basic graph pattern matching shall use the semantics defined by the RIF Core Entailment Regime [W3C SPARQL Entailment] for the RIF rules [W3C RIF Core] `geor:rcc8eq`, `geor:rcc8dc`, `geor:rcc8ec`, `geor:rcc8po`, `geor:rcc8tppi`, `geor:rcc8tpp`, `geor:rcc8ntpp`, `geor:rcc8ntppi`.
`/req/query-rewrite-extension/rcc8-query-rewrite`

Table 11. Simple Features Query Transformation Rules

Rule	ogc:relation	ogc:function
geor:rcc8eq	geo:rcc8eq	geof:rcc8eq
geor:rcc8dc	geo:rcc8dc	geof:rcc8dc
geor:rcc8ec	geo:rcc8ec	geof:rcc8ec
geor:rcc8po	geo:rcc8po	geof:rcc8po
geor:rcc8tppi	geo:rcc8tppi	geof:rcc8tppi
geor:rcc8tpp	geo:rcc8tpp	geof:rcc8tpp
geor:rcc8ntpp	geo:rcc8ntpp	geof:rcc8ntpp
geor:rcc8ntppi	geo:rcc8ntppi	geof:rcc8ntppi

8.11.5. Special Considerations

The applicability of GeoSPARQL rules in certain circumstances has intentionally been left undefined.

The first situation arises for triple patterns with unbound predicates. Consider the query pattern below:

```
{ my:feature1 ?p my:feature2 }
```

When using a query transformation strategy, this triple pattern could invoke none of the GeoSPARQL rules or all of the rules. Implementations are free to support either of these alternatives.

The second situation arises when supporting GeoSPARQL rules in the presence of RDFS Entailment. The existence of a topological relation (possibly derived from a GeoSPARQL rule) can entail other RDF triples. For example, if `geo:sfOverlaps` has been defined as an `rdfs:subPropertyOf` the property `my:overlaps`, and the RDF triple `my:feature1 geo:sfOverlaps my:feature2` has been derived from a GeoSPARQL rule, then the RDF triple `my:feature1 my:overlaps my:feature2` can be entailed. Implementations may support such entailments but are not required to.

8.12. Future Work

Many future extensions of this standard are possible. Obvious extensions are to define new conformance classes for other standard serializations of geometry data (e.g. KML, GeoJSON). In addition, significant work remains in developing vocabularies for spatial data, and expanding the GeoSPARQL vocabularies with OWL axioms to aid in logical spatial reasoning would be a valuable contribution. There are also large amounts of existing feature data represented either in a GML file (or similar serialization) or in a datastore supporting the general feature model. It would be beneficial to develop standard processes for converting (or virtually converting and exposing) this data to RDF.

8.13. Annex A

8.14. (normative)

8.15. Abstract Test Suite

8.15.1. A.1 Conformance Class: *Core*

Conformance Class URI: `/conf/core`

8.15.1.1. A.1.1 `/conf/core/sparql-protocol`

Requirement: `/req/core/sparql-protocol`

Implementations shall support the SPARQL Query Language for RDF [W3C SPARQL], the SPARQL Protocol for RDF [W3C SPARQL Protocol] and the SPARQL Query Results XML Format [W3C SPARQL Result Format].

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that the implementation accepts SPARQL queries and returns the correct results in the correct format, according to the SPARQL Query Language for RDF, the SPARQL Protocol for RDF and SPARQL Query Results XML Format W3C specifications.

c.) Reference: Clause 6.1 Req 1

d.) Test Type: Capabilities

8.15.1.2. A.1.2 `/conf/core/spatial-object-class`

Requirement: `/req/core/spatial-object-class`

Implementations shall allow the RDFS class geo:SpatialObject to be used in SPARQL graph patterns.

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that queries involving geo:SpatialObject return the correct result on a test dataset.

c.) Reference: Clause 6.2.1 Req 2 d.) Test Type: Capabilities

8.15.1.3. A.1.3 `/conf/core/feature-class`

Requirement: `/req/core/feature-class` Implementations shall allow the RDFS class geo:Feature to be used in SPARQL graph patterns.

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that queries involving geo:Feature return the correct result on a test dataset.

c.) Reference: Clause 6.2.2 Req 3

d.) Test Type: Capabilities

8.15.2. A.2 Conformance Class: Topology Vocabulary Extension (relation_family) Conformance Class URI: /conf/topology-vocab-extension

8.15.2.1. A.2.1 relation_family = Simple Features

A.2.1.1 /conf/topology-vocab-extension/sf-spatial-relations

Requirement: /req/topology-vocab-extension/sf-spatial-relations Implementations shall allow the properties geo:sfEquals, geo:sfDisjoint, geo:sfIntersects, geo:sfTouches, geo:sfCrosses, geo:sfWithin, geo:sfContains, geo:sfOverlaps to be used in SPARQL graph patterns.

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that queries involving these properties return the correct result for a test dataset.

c.) Reference: Clause 7.2 Req 4

d.) Test Type: Capabilities

8.15.2.2. A.2.2 relation_family = Egenhofer

A.2.2.1 /conf/topology-vocab-extension/eh-spatial-relations

Requirement: /req/topology-vocab-extension/eh-spatial-relations Implementations shall allow the properties geo:ehEquals, geo:ehDisjoint, geo:ehMeet, geo:ehOverlap, geo:ehCovers, geo:ehCoveredBy, geo:ehInside, geo:ehContains to be used in SPARQL graph patterns.

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that queries involving these properties return the correct result for a test dataset.

c.) Reference: Clause 7.3 Req 5

d.) Test Type: Capabilities

8.15.2.3. A.2.3 relation_family = RCC8

A.2.3.1 /conf/topology-vocab-extension/rcc8-spatial-relations

Requirement: /req/topology-vocab-extension/rcc8-spatial-relations Implementations shall allow the properties `geo:rcc8eq, geo:rcc8dc, geo:rcc8ec, geo:rcc8po, geo:rcc8tppi, geo:rcc8tpp, geo:rcc8ntpp, `geo:rcc8ntppi to be used in SPARQL graph patterns

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that queries involving these properties return the correct result for a test

dataset.

c.) Reference: Clause 7.4 Req 6

d.) Test Type: Capabilities

8.15.3. A.3 Conformance Class: Geometry Extension (serialization, version) Conformance Class URI: `/conf/geometry-extension

8.15.3.1. A.3.1 Tests for all Serializations

A.3.1.1 /conf/geometry-extension/geometry-class

Requirement: /req/geometry-extension/geometry-class Implementations shall allow the RDFS class geo:Geometry to be used in SPARQL graph patterns.

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that queries involving geo:Geometry return the correct result on a test dataset.

c.) Reference: Clause 8.2.1 Req 7 d.) Test Type: Capabilities

A.3.1.2 /conf/geometry-extension/feature-properties

Requirement: /req/geometry-extension/feature-properties Implementations shall allow the properties geo:hasGeometry and geo:hasDefaultGeometry to be used in SPARQL graph patterns.

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that queries involving these properties return the correct result for a test dataset.

c.) Reference: Clause 8.3 Req 8

d.) Test Type: Capabilities

A.3.1.3 /conf/geometry-extension/geometry-properties

Requirement: /req/geometry-extension/geometry-properties Implementations shall allow the properties geo:dimension, geo:coordinateDimension, geo:spatialDimension, geo:isEmpty, geo:isSimple, geo:hasSerialization to be used in SPARQL graph patterns.

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that queries involving these properties return the correct result for a test dataset.

c.) Reference: Clause 8.4 Req 9

d.) Test Type: Capabilities

A.3.1.4 /conf/geometry-extension/query-functions

Requirement: /req/geometry-extension/query-functions Implementations shall support geof:distance, geof:buffer, geof:convexHull, geof:intersection, geof:union, geof:difference, geof:symDifference, geof:envelope and `geof:boundary as SPARQL extension functions, consistent with the definitions of the corresponding functions (distance, buffer, convexHull, intersection, difference, symDifference, envelope and boundary respectively) in Simple Features [ISO 19125-1].

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that a set of SPARQL queries involving each of the following functions returns the correct result for a test dataset when using the specified serialization and version: geof:distance, geof:buffer, geof:convexHull, geof:intersection, geof:union, geof:difference, geof:symDifference, geof:envelope and geof:boundary.

c.) Reference: Clause 8.7 Req 19

d.) Test Type: Capabilities

A.3.1.5 /conf/geometry-extension/srid-function

Requirement: /req/geometry-extension/srid-function Implementations shall support `geof:getSRID as a SPARQL extension function.

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that a SPARQL query involving the geof:getSRID function returns the correct result for a test dataset when using the specified serialization and version.

c.) Reference: Clause 8.7 Req 20

d.) Test Type: Capabilities

8.15.3.2. A.3.2 serialization = WKT

A.3.2.1 /conf/geometry-extension/wkt-literal

Requirement: /req/geometry-extension/wkt-literal All RDFS Literals of type geo:wktLiteral shall consist of an optional URI identifying the coordinate reference system followed by Simple Features Well Known Text (WKT) describing a geometric value. Valid geo:wktLiterals are formed by concatenating a valid, absolute URI as defined in [RFC 2396], one or more spaces (Unicode U+0020 character) as a separator, and a WKT string as defined in Simple Features [ISO 19125-1].

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that queries involving geo:wktLiteral values return the correct result for a test dataset.

c.) Reference: Clause 8.5.1 Req 10

d.) Test Type: Capabilities

A.3.2.2 /conf/geometry-extension/wkt-literal-default-srs

Requirement: /req/geometry-extension/wkt-literal-default-srs The URI http://www.opengis.net/def/crs/OGC/1.3/CRS84 shall be assumed as the spatial reference system for geo:wktLiterals that do not specify an explicit spatial reference system URI.

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that queries involving geo:wktLiteral values without an explicit encoded spatial reference system URI return the correct result for a test dataset.

c.) Reference: Clause 8.5.1 Req 11

d.) Test Type: Capabilities

A.3.2.3 /conf/geometry-extension/wkt-axis-order

Requirement: /req/geometry-extension/wkt-axis-order Coordinate tuples within geo:wktLiterals shall be interpreted using the axis order defined in the spatial reference system used.

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that queries involving geo:wktLiteral values return the correct result for a test dataset.

c.) Reference: Clause 8.5.1 Req 12

d.) Test Type: Capabilities

A.3.2.4 /conf/geometry-extension/wkt-literal-empty

Requirement: /req/geometry-extension/wkt-literal-empty An empty RDFS Literal of type geo:wktLiteral shall be interpreted as an empty geometry.

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that queries involving empty geo:wktLiteral values return the correct result for a test dataset.

c.) Reference: Clause 8.5.1 Req 13

d.) Test Type: Capabilities

A.3.2.5 /conf/geometry-extension/geometry-as-wkt-literal

Requirement: /req/geometry-extension/geometry-as-wkt-literal Implementations shall allow the RDF property geo:asWKT to be used in SPARQL graph patterns.

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that queries involving the geo:asWKT property return the correct result for a test dataset.

c.) Reference: Clause 8.5.2 Req 14

d.) Test Type: Capabilities

8.15.3.3. A.3.3 serialization = GML

A.3.3.1 `/conf/geometry-extension/gml-literal`

Requirement: `/req/geometry-extension/gml-literal` All `geo:gmlLiterals` shall consist of a valid element from the GML schema that implements a subtype of GM_Object as defined in [OGC 07-036].

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that queries involving geo:gmlLiteral values return the correct result for a test dataset.

c.) Reference: Clause 8.6.1 Req 15

d.) Test Type: Capabilities

A.3.3.2 `/conf/geometry-extension/gml-literal-empty`

Requirement: `/req/geometry-extension/gml-literal-empty` An empty `geo:gmlLiteral` shall be interpreted as an empty geometry.

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that queries involving empty `geo:gmlLiteral` values return the correct result for a test dataset.

c.) Reference: Clause 8.6.1 Req 16

d.) Test Type: Capabilities

A.3.3.3 `/conf/geometry-extension/gml-profile`

Requirement: `/req/geometry-extension/gml-profile` Implementations shall document supported GML profiles.

a.) Test purpose: check conformance with this requirement

b.) Test method: Examine the implementation's documentation to verify that the supported GML profiles are documented.

c.) Reference: Clause 8.6.1 Req 17

d.) Test Type: Documentation

A.3.3.4 `/conf/geometry-extension/geometry-as-gml-literal`

Requirement: `/req/geometry-extension/geometry-as-gml-literal` Implementations shall allow the RDF property geo:asGML to be used in SPARQL graph patterns.

a.) Test purpose: check conformance with this requirement

b.) Test method: verify that queries involving the `geo:asGML` property return the correct result for a test dataset.

c.) Reference: Clause 8.6.2 Req 18

d.) Test Type: Capabilities

8.15.4. A.4 Conformance Class: Geometry Topology Extension (relation_family, serialization, version)

Conformance Class URI: `/conf/geometry-topology-extension`

8.15.4.1. A.4.1 Tests for all relation families

A.4.1.1 `/conf/geometry-topology-extension/relate-query-function`

Requirement: `/req/geometry-topology-extension/relate-query-function` Implementations shall support `geof:relate as a SPARQL extension function, consistent with the relate operator defined in Simple Features [ISO 19125-1].

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that a set of SPARQL queries involving the `geof:relate function returns the correct result for a test dataset when using the specified serialization and version.

c.) Reference: Clause 9.2 Req 21

d.) Test Type: Capabilities

8.15.4.2. A.4.2 relation_family = Simple Features

A.4.2.1 `/conf/geometry-topology-extension/sf-query-functions`

Requirement: `/req/geometry-topology-extension/sf-query-functions` Implementations shall support `geof:sfEquals`, `geof:sfDisjoint`, `geof:sfIntersects`, `geof:sfTouches`, `geof:sfCrosses`, `geof:sfWithin`, `geof:sfContains`, `geof:sfOverlaps` as SPARQL extension functions, consistent with their corresponding DE-9IM intersection patterns, as defined by Simple Features [ISO 19125-1].

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that a set of SPARQL queries involving each of the following functions returns the correct result for a test dataset when using the specified serialization and version: `geof:sfEquals`, `geof:sfDisjoint`, `geof:sfIntersects`, `geof:sfTouches`, `geof:sfCrosses`, `geof:sfWithin`, `geof:sfContains`, `geof:sfOverlaps`.

c.) Reference: Clause 9.3 Req 22

d.) Test Type: Capabilities

8.15.4.3. A.4.3 relation_family = Egenhofer

A.4.3.1 `/conf/geometry-topology-extension/eh-query-functions`

Requirement: `/req/geometry-topology-extension/eh-query-functions` Implementations shall support `geof:ehEquals`, `geof:ehDisjoint`, `geof:ehMeet`, `geof:ehOverlap`, `geof:ehCovers`, `geof:ehCoveredBy`, `geof:ehInside`, `geof:ehContains` as SPARQL extension functions, consistent with their corresponding DE-9IM intersection patterns, as defined by Simple Features [ISO 19125- 1].

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that a set of SPARQL queries involving each of the following functions returns the correct result for a test dataset when using the specified serialization and version: `geof:ehEquals`, `geof:ehDisjoint`, `geof:ehMeet`, `geof:ehOverlap`, `geof:ehCovers`, `geof:ehCoveredBy`, `geof:ehInside`, `geof:ehContains`.

c.) Reference: Clause 9.4 Req 23

d.) Test Type: Capabilities

8.15.4.4. A.4.4 relation_family = RCC8

A.4.4.1 `/conf/geometry-topology-extension/rcc8-query-functions`

Requirement: `/req/geometry-topology-extension/rcc8-query-functions` Implementations shall support `geof:rcc8eq, `geof:rcc8dc`, `geof:rcc8ec`, `geof:rcc8po`, `geof:rcc8tppi`, `geof:rcc8tpp`, `geof:rcc8ntpp`, `geof:rcc8ntppi` as SPARQL extension functions, consistent with their corresponding DE-9IM intersection patterns, as defined by Simple Features [ISO 19125-1].

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that a set of SPARQL queries involving each of the following functions returns the correct result for a test dataset when using the specified serialization and version: `geof:rcc8eq`, `geof:rcc8dc`, `geof:rcc8ec`, `geof:rcc8po`, `geof:rcc8tppi`, `geof:rcc8tpp`, `geof:rcc8ntpp`, `geof:rcc8ntppi`.

c.) Reference: Clause 9.5 Req 24

d.) Test Type: Capabilities

8.15.5. A.5 Conformance Class: RDFS Entailment Extension (relation_family, serialization, version)

Conformance Class URI: `/conf/rdfs-entailment-extension

8.15.5.1. A.5.1 Tests for all implementations

A.5.1.1 `/conf/rdfsentailmentextension/bgp-rdfs-ent`

Requirement: `/req/rdfs-entailment-extension/bgp-rdfs-ent` Basic graph pattern matching shall use the semantics defined by the RDFS Entailment Regime [W3C SPARQL Entailment].

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that a set of SPARQL queries involving entailed RDF triples returns the correct result for a test dataset using the specified serialization, version and relation_family.

c.) Reference: Clause 10.2 Req 25

d.) Test Type: Capabilities

8.15.5.2. A.5.2 serialization=WKT

A.5.2.1 `/conf/rdfs-entailment-extension/wkt-geometry-types`

Requirement: `/req/rdfs-entailment-extension/wkt-geometry-types` Implementations shall support graph patterns involving terms from an RDFS/OWL class hierarchy of geometry types consistent with the one in the specified version of Simple Features [ISO 19125-1].

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that a set of SPARQL queries involving WKT Geometry types returns the correct result for a test dataset using the specified version of Simple Features. c.) Reference: Clause 10.3.1 Req 26

d.) Test Type: Capabilities

8.15.5.3. A.5.3 serialization=GML

A.5.3.1 `/conf/rdfs-entailment-extension/gml-geometry-types`

Requirement: `/req/rdfs-entailment-extension/gml-geometry-types` Implementations shall support graph patterns involving terms from an RDFS/OWL class hierarchy of geometry types consistent with the GML schema that implements GM_Object using the specified version of GML [OGC 07-036].

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that a set of SPARQL queries involving GML Geometry types returns the correct result for a test dataset using the specified version of GML.

c.) Reference: Clause 10.4.1 Req 27

d.) Test Type: Capabilities

8.15.6. A.6 Conformance Class: Query Rewrite Extension (relation_family, serialization, version)

Conformance Class URI: `/conf/query-rewrite-extension`

8.15.6.1. A.6.1 relation_family = Simple Features

A.6.1.1 `/conf/query-rewrite-extension/sf-query-rewrite`

Requirement: `/req/query-rewrite-extension/sf-query-rewrite` Basic graph pattern matching shall

use the semantics defined by the RIF Core Entailment Regime [W3C SPARQL Entailment] for the RIF rules [W3C RIF Core] geor:sfEquals, geor:sfDisjoint, geor:sfIntersects, geor:sfTouches, geor:sfCrosses, geor:sfWithin, geor:sfContains, geor:sfOverlaps.

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that queries involving the following query transformation rules return the correct result for a test dataset when using the specified serialization and version: geor:sfEquals, geor:sfDisjoint, geor:sfIntersects, geor:sfTouches, geor:sfCrosses, geor:sfWithin, geor:sfContains and `geor:sfOverlaps.

c.) Reference: Clause 11.2 Req 28

d.) Test Type: Capabilities

8.15.6.2. A.6.2 relation_family = Egenhofer

A.6.2.1 `/conf/query-rewrite-extension/eh-query-rewrite

Requirement: /req/query-rewrite-extension/eh-query-rewrite Basic graph pattern matching shall use the semantics defined by the RIF Core Entailment Regime [W3C SPARQL Entailment] for the RIF rules [W3C RIF Core] `geor:ehEquals, geor:ehDisjoint, geor:ehMeet, geor:ehOverlap, geor:ehCovers, geor:ehCoveredBy, geor:ehInside, geor:ehContains.

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that queries involving the following query transformation rules return the correct result for a test dataset when using the specified serialization and version: geor:ehEquals, geor:ehDisjoint, geor:ehMeet, geor:ehOverlap, geor:ehCovers, geor:ehCoveredBy, geor:ehInside, geor:ehContains.

c.) Reference: Clause 11.3 Req 29 d.) Test Type: Capabilities

8.15.6.3. A.6.3 relation_family = RCC8

A.6.3.1 `/conf/query-rewrite-extension/rcc8-query-rewrite

Requirement: /req/query-rewrite-extension/rcc8-query-rewrite Basic graph pattern matching shall use the semantics defined by the RIF Core Entailment Regime [W3C SPARQL Entailment] for the RIF rules [W3C RIF Core] `geor:rcc8eq, geor:rcc8dc, geor:rcc8ec, geor:rcc8po, geor:rcc8tppi, geor:rcc8tpp, geor:rcc8ntpp, geor:rcc8ntppi.

a.) Test purpose: check conformance with this requirement

b.) Test method: Verify that queries involving the following query transformation rules return the correct result for a test dataset when using the specified serialization and version: geor:rcc8eq, geor:rcc8dc, geor:rcc8ec, geor:rcc8po, geor:rcc8tppi, geor:rcc8tpp, geor:rcc8ntpp, geor:rcc8ntppi.

c.) Reference: Clause 11.4 Req 30 d.) Test Type: Capabilities

8.16. Annex B

8.17. (informative)

8.18. GeoSPARQL Examples

8.18.1. B.1 Example Data

The following RDF/XML data encodes application-specific spatial data. The resulting spatial data is illustrated in Figure 3. The RDF statements define the feature class my:PlaceOfInterest, and two properties are created for associating geometries with features: my:hasExactGeometry and my:hasPointGeometry. my:hasExactGeometry is designated as the default geometry for the my:PlaceOfInterest feature class.

All the following examples use the parameter values relation_family = Simple Features, serialization = WKT, and version = 1.0.

[RDF Triple] | *img/03.png*

Figure 3. Illustration of spatial data

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:sf="http://www.opengis.net/ont/sf#"
    xmlns:geo="http://www.opengis.net/ont/geosparql#"
    xmlns:my="http://example.org/ApplicationSchema#">

<!-- Integration with GeoSPARQL classes and properties -->
<rdfs:Class rdf:about=
    "http://example.org/ApplicationSchema#PlaceOfInterest">
    <rdfs:subClassOf rdf:resource=
        "http://www.opengis.net/ont/geosparql#Feature"/>
</rdfs:Class>
<rdf:Property rdf:about=
    "http://example.org/ApplicationSchema#hasExactGeometry">
    <rdfs:subPropertyOf rdf:resource=
        "http://www.opengis.net/ont/geosparql#hasGeometry"/>
    <rdfs:subPropertyOf rdf:resource=
        "http://www.opengis.net/ont/geosparql#hasDefaultGeometry"/>
</rdf:Property>
<rdf:Property rdf:about=
    "http://example.org/ApplicationSchema#hasPointGeometry">
    <rdfs:subPropertyOf rdf:resource=
        "http://www.opengis.net/ont/geosparql#hasGeometry"/>
</rdf:Property>

<!-- Instance-level statements -->
```

```
<!-- A -->
<my:PlaceOfInterest rdf:about="http://example.org/ApplicationSchema#A">
    <my:hasExactGeometry rdf:resource=
        "http://example.org/ApplicationSchema#AExactGeom"/>
    <my:hasPointGeometry rdf:resource=
        "http://example.org/ApplicationSchema#APointGeom"/>
</my:PlaceOfInterest>
<sf:Polygon rdf:about="http://example.org/ApplicationSchema#AExactGeom">
    <geo:asWKT rdf:datatype=
        "http://www.opengis.net/ont/geosparql#wktLiteral">
        <![CDATA[
            <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
            Polygon((-83.6 34.1, -83.2 34.1, -83.2 34.5,
                    -83.6 34.5, -83.6 34.1))
        ]]>
    </geo:asWKT>
</sf:Polygon>
<sf:Point rdf:about="http://example.org/ApplicationSchema#APointGeom">
    <geo:asWKT rdf:datatype=
        "http://www.opengis.net/ont/geosparql#wktLiteral">
        <![CDATA[
            <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
            Point(-83.4 34.3)
        ]]>
    </geo:asWKT>
</sf:Point>

<!-- B -->
<my:PlaceOfInterest rdf:about="http://example.org/ApplicationSchema#B">
    <my:hasExactGeometry rdf:resource=
        "http://example.org/ApplicationSchema#BExactGeom"/>
    <my:hasPointGeometry rdf:resource=
        "http://example.org/ApplicationSchema#BPointGeom"/>
</my:PlaceOfInterest>
<sf:Polygon rdf:about="http://example.org/ApplicationSchema#BExactGeom">
    <geo:asWKT rdf:datatype=
        "http://www.opengis.net/ont/geosparql#wktLiteral">
        <![CDATA[
            <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
            Polygon((-83.6 34.1, -83.4 34.1, -83.4 34.3,
                     -83.6 34.3, -83.6 34.1))
        ]]>
    </geo:asWKT>
</sf:Polygon>
<sf:Point rdf:about="http://example.org/ApplicationSchema#BPointGeom">
    <geo:asWKT rdf:datatype=
        "http://www.opengis.net/ont/geosparql#wktLiteral">
        <![CDATA[
            <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
            Point(-83.5 34.2)
        ]]>
```

```
    </geo:asWKT>
</sf:Point>

<!-- C -->
<my:PlaceOfInterest rdf:about="http://example.org/ApplicationSchema#C">
    <my:hasExactGeometry rdf:resource=
        "http://example.org/ApplicationSchema#CExactGeom"/>
    <my:hasPointGeometry rdf:resource=
        "http://example.org/ApplicationSchema#CPointGeom"/>
</my:PlaceOfInterest>
<sf:Polygon rdf:about="http://example.org/ApplicationSchema#CExactGeom">
    <geo:asWKT rdf:datatype=
        "http://www.opengis.net/ont/geosparql#wktLiteral">
        <![CDATA[
            <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
            Polygon((-83.2 34.3, -83.0 34.3, -83.0 34.5,
                     -83.2 34.5, -83.2 34.3))
        ]]>
    </geo:asWKT>
</sf:Polygon>
<sf:Point rdf:about="http://example.org/ApplicationSchema#CPointGeom">
    <geo:asWKT rdf:datatype=
        "http://www.opengis.net/ont/geosparql#wktLiteral">
        <![CDATA[
            <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
            Point(-83.1 34.4)
        ]]>
    </geo:asWKT>
</sf:Point>

<!-- D -->
<my:PlaceOfInterest rdf:about="http://example.org/ApplicationSchema#D">
    <my:hasExactGeometry rdf:resource=
        "http://example.org/ApplicationSchema#DExactGeom"/>
    <my:hasPointGeometry rdf:resource=
        "http://example.org/ApplicationSchema#DPointGeom"/>
</my:PlaceOfInterest>
<sf:Polygon rdf:about="http://example.org/ApplicationSchema#DExactGeom">
    <geo:asWKT rdf:datatype=
        "http://www.opengis.net/ont/geosparql#wktLiteral">
        <![CDATA[
            <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
            Polygon((-83.3 34.0, -83.1 34.0, -83.1 34.2,
                     -83.3 34.2, -83.3 34.0))
        ]]>
    </geo:asWKT>
</sf:Polygon>
<sf:Point rdf:about="http://example.org/ApplicationSchema#DPointGeom">
    <geo:asWKT rdf:datatype=
        "http://www.opengis.net/ont/geosparql#wktLiteral">
        <![CDATA[
```

```
            <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
        Point(-83.2 34.1)
        ]]>
    </geo:asWKT>
</sf:Point>

<!-- E -->
<my:PlaceOfInterest rdf:about="http://example.org/ApplicationSchema#E">
    <my:hasExactGeometry rdf:resource=
        "http://example.org/ApplicationSchema#EExactGeom"/>
</my:PlaceOfInterest>
<sf:LineString rdf:about=
    "http://example.org/ApplicationSchema#EExactGeom">
    <geo:asWKT rdf:datatype=
        "http://www.opengis.net/ont/geosparql#wktLiteral">
        <![CDATA[
            <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
            LineString((-83.4 34.0, -83.3 34.3))
        ]]>
    </geo:asWKT>
</sf:LineString>

<!-- F -->
<my:PlaceOfInterest rdf:about="http://example.org/ApplicationSchema#F">
    <my:hasExactGeometry rdf:resource=
        "http://example.org/ApplicationSchema#FExactGeom"/>
</my:PlaceOfInterest>
<sf:Point rdf:about="http://example.org/ApplicationSchema#FExactGeom">
    <geo:asWKT rdf:datatype=
        "http://www.opengis.net/ont/geosparql#wktLiteral">
        <![CDATA[
            <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
        Point(-83.4 34.4)
        ]]>
    </geo:asWKT>
</sf:Point>

</rdf:RDF>
```

8.18.2. B.2 Example Queries

This Section illustrates the use of GeoSPARQL functions through a series of example queries.

Example 1: *Find all features that feature* `my:A` *contains, where spatial calculations are based on* `my:hasExactGeometry`.

```
PREFIX my: <http://example.org/ApplicationSchema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>

SELECT ?f
WHERE {
    my:A my:hasExactGeometry ?aGeom .
    ?aGeom geo:asWKT ?aWKT .
    ?f my:hasExactGeometry ?fGeom .
    ?fGeom geo:asWKT ?fWKT .
    FILTER (geof:sfContains(?aWKT, ?fWKT) &&
            !sameTerm(?aGeom, ?fGeom))
)
```

Result:

?f
my:B
my:F

Example 2: *Find all features that are within a transient bounding box geometry, where spatial calculations are based on* `my:hasPointGeometry`.

```
PREFIX my: <http://example.org/ApplicationSchema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>

SELECT ?f
WHERE { ?f my:hasPointGeometry ?fGeom .
        ?fGeom geo:asWKT ?fWKT .
        FILTER (geof:sfWithin(?fWKT,
                "<http://www.opengis.net/def/crs/OGC/1.3/CRS84>
                Polygon ((-83.4 34.0, -83.1 34.0,
                          -83.1 34.2, -83.4 34.2,
                          -83.4 34.0))"^^geo:wktLiteral))
)
```

Result:

?f
my:D

Example 3: *Find all features that touch the union of feature* `my:A` *and feature* `my:D`*, where computations are based on* `my:hasExactGeometry`.

```
PREFIX my: <http://example.org/ApplicationSchema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>

SELECT ?f
WHERE { ?f my:hasExactGeometry ?fGeom .
        ?fGeom geo:asWKT ?fWKT .
        my:A my:hasExactGeometry ?aGeom .
        ?aGeom geo:asWKT ?aWKT .
        ?my:D my:hasExactGeometry ?dGeom .
        ?dGeom geo:asWKT ?dWKT .
        FILTER (geof:sfTouches(?fWKT,
                geof:union(?aWKT, ?dWKT)))
}
```

Result:

?f
my:C

Example 4: *Find the 3 closest features to feature my:C, where computations are based on* my:hasExactGeometry.

```
PREFIX uom: <http://www.opengis.net/def/uom/OGC/1.0/>
PREFIX my: <http://example.org/ApplicationSchema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/geosparql/function>

SELECT ?f
WHERE { my:C my:hasExactGeometry ?cGeom .
        ?cGeom geo:asWKT ?cWKT .
        ?f my:hasExactGeometry ?fGeom . ?fGeom geo:asWKT ?fWKT .
        FILTER (?fGeom != ?cGeom) }
ORDER BY ASC (geof:distance(?cWKT, ?fWKT,
              uom:metre))
LIMIT 3
```

Result:

?f
my:A
my:D
my:E

8.18.3. B.3 Example Rule Application

This section illustrates the query transformation strategy for implementing GeoSPARQL rules.

Example 5: *Find all features or geometries that overlap feature* `my:A`.

Original Query:

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>

SELECT ?f
WHERE { ?f geo:sfOverlaps my:A }
```

Transformed Query (application of transformation rule geor:sfOverlaps):

```
PREFIX my: <http://example.org/ApplicationSchema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>

SELECT ?f
WHERE { { # check for asserted statement
          ?f geo:sfOverlaps my:A }
        UNION
        { # feature □ feature
          ?f geo:hasDefaultGeometry ?fGeom .
          ?fGeom geo:asWKT ?fSerial .
          my:A geo:hasDefaultGeometry ?aGeom .
          ?aGeom geo:asWKT ?aSerial .
          FILTER (geof:sfOverlaps(?fSerial, ?aSerial)) }
        UNION
        { # feature □ geometry
          ?f geo:hasDefaultGeometry ?fGeom .
          ?fGeom geo:asWKT ?fSerial .
          my:A geo:asWKT ?aSerial .
          FILTER (geof:sfOverlaps(?fSerial, ?aSerial)) }
        UNION
        { # geometry □ feature
          ?f geo:asWKT ?fSerial .
          my:A geo:hasDefaultGeometry ?aGeom .
          ?aGeom geo:asWKT ?aSerial .
          FILTER (geof:sfOverlaps(?fSerial, ?aSerial)) }
        UNION
        { # geometry □ geometry
          ?f geo:hasDefaultGeometry ?fGeom .
          ?fGeom geo:asWKT ?fSerial .
          my:A geo:hasDefaultGeometry ?aGeom .
          ?aGeom geo:asWKT ?aSerial .
          FILTER (geof:sfOverlaps(?fSerial, ?aSerial)) }
}
```

Result:

?f
my:D
my:DExactGeom
my:E
my:EExactGeom

Bibliography

[1] Cohn, Anthony G., Brandon Bennett, John Gooday, Nicholas Mark Gotts, *Qualitative Spatial Representation and Reasoning with the Region Connection Calculus*, GeoInformatica, 1, 275–316, 1997.

[2] Egenhofer Max, *A Formal Definition of Binary Topological Relationships*, Third International Conference on Foundations of Data Organization and Algorithms (FODO), Paris, France, W. Litwin and H. Schek (eds.), Lecture Notes in Computer Science, Vol. 367, Springer-Verlag, pp. 457-472, June 1989. available at http://www.spatial.maine.edu/~max/RC2.html

[3] Egenhofer, Max and J. Herring *Categorizing Binary Topological Relations Between Regions, Lines, and Points*, Geographic Databases, Technical Report, Department of Surveying Engineering, University of Maine, 1990. (revised versions in NCGIA Technical Report 91-7 and NCGIA Technical Report 94-1)

[[13249]] [4] ISO/IEC 13249-3, *Information technology — Database languages — SQL multimedia and application packages — Part 3: Spatial*

[[19105]] [5] ISO 19105, *Geographic information – Conformance and testing*

[[19107]] [6] ISO 19107, *Geographic information — Spatial schema*

[[19109]] [7] ISO 19109, *Geographic information — Rules for application schemas*

[[19156]] [8] ISO 19156, *Geographic information — Observations and measurements*

[9] Randell, D. A., Cui, Z. and Cohn, A. G.: A spatial logic based on regions and connection, Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning, Morgan Kaufmann, San Mateo, pp. 165–176, 1992.

[10] W3C Turtle, *Turtle – Terse RDF Triple Language*, W3C Team Submission (28 March 2011)

[11] W3C RDF XML, *RDF/XML Syntax Specification (Revised)*, W3C Recommendation (10 February 2004)

[12] W3C RDF, *Resource Description Framework (RDF): Concepts and Abstract Syntax*, W3C Recommendation (10 February 2004)

[13] W3C RDF Semantics, *RDF Semantics*, W3C Recommendation (10 February 2004)

[14] W3C RDFS, *RDF Vocabulary Description Language 1.0: RDF Schema*, W3C Recommendation (10 February 2004)

[15] W3C RIF Overview, *RIF Overview*, W3C Working Group Note (22 June 2010)

[16] W3C OWL2, *OWL 2 Web Ontology Language Document Overview*, W3C Recommendation (27 October 2009)

[17] W3C XML, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, W3C Recommendation (26 November 2008)

[18] W3C XML Namespaces, *Namespaces in XML 1.0 (Second Edition)*, W3C Recommendation (16 August 2006)

[19] W3C XML Schema Part 1, *XML Schema Part 1: Structures (Second Edition)*, W3C Recommendation (28 October 2004)

[20] W3C XML Schema Part 2, *XML Schema Part 2: Datatypes (Second Edition)*, W3C Recommendation (28 October 2004)