



IEEE Standard for Policy Language for Dynamic Spectrum Access Systems

IEEE Communications Society

Developed by the
Dynamic Spectrum Access Networks Standards Committee

IEEE Std 1900.5.1™-2020

IEEE Standard for Policy Language for Dynamic Spectrum Access Systems

Developed by the

Dynamic Spectrum Access Networks Standards Committee
of the
IEEE Communications Society

Approved 3 December 2020

IEEE SA Standards Board

Abstract: A vendor-independent policy language for managing the functionality and behavior of dynamic spectrum access networks based on the language requirements defined in IEEE Std 1900.5™, IEEE Standard Policy Language Requirements and System Architectures for Dynamic Spectrum Access Systems, is defined in this standard.

Keywords: Description Logic, Dynamic Spectrum Access, First Order Logic, Horn-based rules, IEEE 1900.5.1™, IEEE 1900.5.2™, Policy Language, RIF Framework for Logic Dialect (RIF-FLD), rule interchange format (RIF), Spectrum Consumption Modeling, Web Ontology Language (OWL 2)

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2021 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 26 February 2021. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-1-5044-7261-6 STD24537
Print: ISBN 978-1-5044-7262-3 STDPD24537

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <https://www.ieee.org/about/corporate/governance/p9-26.html>.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (<https://standards.ieee.org/ipr/disclaimers.html>), appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA, and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or completeness of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to results and workmanlike effort. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE is the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents.**

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and Standards Coordinating Committees are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the [IEEE SA myProject system](#). An IEEE Account is needed to access the application.

Comments on standards should be submitted using the [Contact Us](#) form.

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These

include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; <https://www.copyright.com/>. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit [IEEE Xplore](#) or [contact IEEE](#). For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the [IEEE SA Website](#). Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in [IEEE Xplore](#). Users are encouraged to periodically check for errata.

Patents

IEEE Standards are developed in compliance with the [IEEE SA Patent Policy](#).

IMPORTANT NOTICE

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. IEEE Standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

Participants

At the time this IEEE standard was completed, the Policy Language and Architectures for Managing Cognitive Radio for Dynamic Spectrum Access Applications Working Group had the following membership:

Tony Rennier, *Chair*
Darcy-Swain Walsh, *Vice Chair*
Carlos Caicedo, *Secretary*

David Chester
Colb Harper
Nilesh Khamberkar

Mieczyslaw Kokar
Yuriy Posherstnik
Venkatesha Prasad

Sam Schmitz
Reinhard Schrage
John Stine

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention:

Demetrio Bucaneg Jr
Carlos Caicedo
Juan Carreon
Pin Chang
Todor Cooklev
Marc Emmelmann
Avraham Freedman
Tim Godfrey
Randall C. Groves
Marco Hernandez
Werner Hoelzl
Oliver Holland
Atsushi Ito

Piotr Karocki
Stuart Kerry
Mieczyslaw Kokar
Alexander Lackpour
Hyeong Ho Lee
Kin Hong Lee
Jeffery Masters
Edward McCall
Nicks A. Nikjoo
Arumugam Paventhan
Clinton Powell
Venkatesha Prasad

Ranga Reddy
Tony Rennier
Maximilian Riegel
Benjamin Rolfe
Thomas Starai
John Stine
Walter Struppler
Mark-Rene Uchida
John Vergis
Lisa Ward
Hung-Yu Wei
Andreas Wolf
Oren Yuen

When the IEEE SA Standards Board approved this standard on 3 December 2020, it had the following membership:

Gary Hoffman, *Chair*
Jon Walter Rosdahl, *Vice Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Ted Burse
Doug Edwards
J. Travis Griffith
Grace Gu
Guido R. Hiertz
Joseph L. Koepfinger*

David J. Law
Howard Li
Dong Liu
Kevin Lu
Paul Nikolich
Damir Novosel
Dorothy Stanley

Mehmet Ulema
Lei Wang
Sha Wei
Philip B. Winston
Daidi Zhong
Jingyi Zhou

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 1900.5.1-2020, IEEE Standard for Policy Language for Dynamic Spectrum Access Systems.

Acknowledgments

Copyright © 2015 W3C® (MIT, ERCIM, Keio, Beihang). This software or document includes material copied from or derived from the following documents:

OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). W3C Recommendation, 11 December 2012. Available at <http://www.w3.org/TR/owl2-syntax/>

OWL 2 Web Ontology Language Direct Semantics (Second Edition). W3C Recommendation, 11 December 2012. Available at <http://www.w3.org/TR/owl2-direct-semantics/>

OWL 2 Web Ontology Language Profiles (Second Edition). W3C Recommendation, 11 December 2012. Available at http://www.w3.org/TR/owl2-profiles/#Feature_Overview_3

RIF Framework for Logic Dialects (Second Edition). W3C Recommendation, 5 February 2013. Available at http://www.w3.org/TR/rif-fld/#Semantic_Framework

RIF Core Dialect (Second Edition). W3C Recommendation 5 February 2013. Available at <http://www.w3.org/TR/rif-core/>

OWL Web Ontology Language Reference, M. Dean, G. Schreiber, Editors, W3C Recommendation, 10 February 2004. Available at <http://www.w3.org/TR/owl-ref/>

Resource Description Framework (RDF): Concepts and Abstract Syntax], G. Klyne, J. Carroll, Editors, W3C Recommendation, 10 February 2004, <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>. Available at <http://www.w3.org/TR/rdf-concepts/>

RIF Basic Logic Dialect (Second Edition) Harold Boley, Michael Kifer, eds. W3C Recommendation, 5 February 2013, <http://www.w3.org/TR/2013/REC-rif-bld-20130205/>. Available at <http://www.w3.org/TR/rif-bld/>

RIF Core Dialect (Second Edition) Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres, Dave Reynolds, eds. W3C Recommendation, 5 February 2013, <http://www.w3.org/TR/rif-core/>

RIF Datatypes and Built-Ins 1.0 (Second Edition) Axel Polleres, Harold Boley, Michael Kifer, eds. W3C Recommendation, 5 February 2013, <http://www.w3.org/TR/2013/REC-rif-dtb-20130205/>. Available at <http://www.w3.org/TR/rif-dtb/>

RIF In RDF (Second Edition) Sandro Hawke, Axel Polleres, eds. W3C Working Group Note, 5 February 2013, <http://www.w3.org/TR/2013/NOTE-rif-in-rdf-20130205/>.

RIF Combination with XML data (Second Edition) Christian de Sainte Marie, editor. W3C Working Group Note, 5 February 2013, <http://www.w3.org/TR/2013/NOTE-rif-xml-data-20130205/>. Available at <http://www.w3.org/TR/rif-xml-data/>

RIF Overview, Kifer M. and Boley H. (Editors), W3C Rule Interchange Format Working Group Note. Available at <http://www.w3.org/TR/rif-overview/>

OWL 2 RL in RIF (Second Edition) Dave Reynolds, editor. W3C Working Group Note, 5 February 2013, <http://www.w3.org/TR/2013/NOTE-rif-owl-rl-20130205/>.

RIF Framework for Logic Dialects (Second Edition) Harold Boley, Michael Kifer, eds. W3C Recommendation, 5 February 2013, <http://www.w3.org/TR/2013/REC-rif-flt-20130205/>. Available at <http://www.w3.org/TR/rif-flt/>

RIF Production Rule Dialect (Second Edition) Christian de Sainte Marie, Gary Hallmark, Adrian Paschke, eds. W3C Recommendation, 5 February 2013, <http://www.w3.org/TR/2013/REC-rif-prd-20130205/>. Available at <http://www.w3.org/TR/rif-prd/>

RIF RDF and OWL Compatibility (Second Edition) Jos de Bruijn, Chris Welty, eds. W3C Recommendation, 5 February 2013, <http://www.w3.org/TR/2013/REC-rif-rdf-owl-20130205/>. Available at <http://www.w3.org/TR/rif-rdf-owl/>

RIF Test Cases (Second Edition) Stella Mitchell, Leora Morgenstern, Adrian Paschke, eds. W3C Working Group Note, 5 February 2013, <http://www.w3.org/TR/2013/NOTE-rif-test-20130205/>. Available at <http://www.w3.org/TR/rif-test/>

RIF Use Cases and Requirements (Second Edition) Adrian Paschke, Leora Morgenstern, David Hirtle, Allen Ginsberg, Paula-Lavinia Patranjan, Frank McCabe, eds. W3C Working Group Note, 5 February 2013, <http://www.w3.org/TR/2013/NOTE-rif-ucr-20130205/>.

RIF Approved Test Cases, Mitchell, S. (Editor), RIF Approved Test Cases. Available at <http://www.w3.org/2005/rules/wiki/Category:Approved>

Contents

1. Overview	12
1.1 Introductory prevailing conditions.....	12
1.2 Scope	12
1.3 Purpose	13
1.4 Word usage.....	13
1.5 Notion of “policy” in its various uses	14
1.5.1 Current state	14
1.5.2 Cognitive radio environment.....	14
1.5.3 Underlying concepts	15
1.6 Related work.....	36
1.6.1 IEEE Std 1900.5-2011	36
2. Normative references.....	36
3. Definitions, acronyms, and abbreviations	36
3.1 Definitions	36
3.2 Acronyms and abbreviations	36
4. Policy Language syntax representation	37
4.1 Logical structure	37
4.1.1 OWL 2 syntax—General definitions and OWL 2 constructs.....	38
4.1.2 RIF syntax—General definitions and RIF constructs	51
4.1.3 DSA Policy Language syntax presentation and syntactical constraints	53
4.1.4 RIF-DSA safeness criteria	62
4.2 Physical structure.....	69
4.2.1 OWL 2 RL serialization.....	69
4.2.2 RIF-DSA and plain XML document.....	69
4.2.3 RIF and RDF/RDFS document.....	74
4.2.4 RIF and OWL 2 Document	74
4.2.5 Importing XML data and XML schemas into RIF	75
4.2.6 Importing RDF and OWL in RIF	76
4.3 Conformance	78
4.4 Extensibility.....	78
5. Policy Language semantics	78
5.1 Semantics.....	78
5.2 Purpose of policy reasoner.....	79
5.3 Predicate logic as a programming language	80
5.3.1 Orders of logical systems	80
5.4 DSA Policy Language—RIF-DSA logics	88
5.4.1 Horn notation—Definite programs	88
5.4.2 Horn notation—Normal programs	88
5.4.3 Horn notation—Unit and goal clauses	88
5.4.4 RIF-DSA versus Horn notation.....	89
5.5 Expressivity of computable functions in DSA Policy Language.....	89
5.5.1 Primitive recursive functions	89
5.5.2 Minimalization— μ -Operator	91
5.5.3 Characteristic functions.....	91
5.5.4 Transformation into horn-based predicates.....	93
5.5.5 Examples.....	95
5.5.6 Mapping of computable functions into DSA clauses.....	98
5.6 Negation	99
5.6.1 Elimination of negation by new predicate introduction	99

5.6.2 Negation by NAF	101
5.7 Reasoning in DSA Policy Language	105
5.7.1 Unification	105
5.7.2 Recursion	105
5.7.3 General first order language interpretations.....	107
5.7.4 DSA Policy Language interpretation specifics	108
5.7.5 Distinguishing slots in object-oriented languages and RIF	112
5.7.6 OWL 2 compatible RIF reasoning	113
5.8 Definite program execution	114
5.9 General program execution.....	117
5.10 Object model representation	118
5.11 Entity capability and state model representation	119
5.12 Condition, decision tree, classification tree, table, and/or rule model	119
5.13 Extensibility	124
6. Runtime environment	124
6.1 Runtime environment for sensing and perceiving	124
6.2 Procedural attachments	124
7. Case-based language analysis.....	124
7.1 Applicability of deontic logic for policy language	124
7.1.1 Policy model using deontic logic	125
7.1.2 Functional view of cognitive radio modeling using Upper Ontology Nuvio	125
7.1.3 Custom domain model/ontology	129
7.2 IEEE Std 1900.5-2011 Use Case 1	131
7.2.1 Programmatic XSD-based schema transform into OWL 2 ontology	131
7.2.2 Sample XSLT style sheet to transform SCM schema into OWL 2 ontology	135
7.3 IEEE Std 1900.5-2011 Use Case 2	145
8. Origins and relationship to other standards	152
Annex A (informative)	153
A.1 Requirements and objectives	153
Annex B (informative) Listing of used CRO knobs and meters, schematic diagrams	159
B.1 Cognitive radio ontology: classes (232).....	159
B.2 SDRontology: objectproperties (156)	161
B.3 SDRontology: dataproperties (39)	163
B.4 SDRontology: individuals (19)	163
B.5 SDRontology: datatypes (5).....	164
Annex C (informative) NUVIO upper ontology class.....	166
Annex D (informative) OWL 2 and RIF syntax components	167
D.1 RIF-DSA.....	167
D.1.1 RIF-DSA—Specialization of RIF-FLD	167
D.1.2 RIF-DSA—Constructive description	172
D.2 OWL 2	182
D.2.1 OWL 2 Syntax Presentation—General definitions	182
D.2.2 OWL 2 Syntax Presentation—Definitions of OWL 2 constructs.....	183
D.2.3 OWL 2 RL syntactical constraints	184
D.3 XPath 2.0 EBNF grammar	186
Annex E (informative) Examples of automated generation, deployment, conformance, and enforcement.....	187
E.1 Type-handling in RIF+XML documents.....	187

Annex F (informative) NAF semantics—Handling nonmonotonic semantics	189
F.1 Semantics of a definite program	189
F.2 Semantics of a normal program	190
F.3 Formal definition	191
F.4 Least fixpoint lfp versus greatest fixpoint gfp	192
F.5 Resolution with graph-theoretic procedure	193
Annex G (informative) Algorithms	195
G.1 Property chain strict order algorithm	195
G.2 Algorithm to check for RIA regularity	195
Annex H (informative) Bibliography	199

IEEE Standard for Policy Language for Dynamic Spectrum Access Systems

1. Overview

1.1 Introductory prevailing conditions

In the development of this standard, the following conditions were to be met:

- a) The development of this standard was required to follow the specifications of IEEE Std 1900.5TM-2011.¹
- b) According to IEEE Std 1900.5-2011, the language should be based both on the Web Ontology Language (OWL 2) and on a rules language compatible with OWL 2, that is, be compatible with a suitable rule interchange format (RIF) dialect.
- c) The development of the standard required some linguistic constructs that are not available in current RIF dialects as RIF Basic Logic Dialect (RIF-BLD), RIF Production Rule Dialect (RIF-PRD), and RIF-Core.
- d) Consequently, this new standard is introducing a new dialect of RIF [RIF Spectrum Consumption Modeling (SCM) Dialect based on RIF Framework for Logic Dialect (RIF-FLD) called “RIF-DSA”].
- e) Since this standard is addressed to the community of radio and networks community, to make the language usable to this community, introductory material on formal languages like RDF, OWL, and RIF, as well as logic and theory of computation, needed to be included.
- f) One requirement of IEEE Std 1900.5-2011 was that the language should be able to use ontologies to define policies. Toward this end, an example ontology was used. This ontology is not part of the standard.

1.2 Scope

This document defines a standard for a formal language to specify policies for radio devices that use cognitive patterns in their operating behavior.

¹ Information on references can be found in Clause 2.

IEEE Std 1900.5-2011 defines cognitive radios as a type of radio in which the device *is aware* of the radio's environment (which can include location, time, or other operational parameters related to communication systems) and internal state and can *make decisions* about the radio operating behavior based on that information and predefined objectives.

To facilitate productive, cooperative behavior in a dynamic ecosystem of communicating devices, the process of decision-making in cognitive radios requires dynamically adaptable guidelines, expressed as policies.

Adaptive knowledge capturing, organization, representation, and reasoning is required as environmental awareness and dynamic, objective-steered, decision-making cannot be achieved on a basis of static data and procedures. This standard uses ontological modeling in conjunction with logic rules to provide the informational basis for knowledge representation and entailing statements for cognitive radios.

Among other desired operating features that provide significant enhancements over classic hardware-defined radios, cognitive radios are capable of dynamically assessing and accessing available spectrum in wireless communications.

A set of base policies augmented by dynamic updates serves as foundation for a cognitive behavior. Dynamic updates solicited by the local device or received asynchronously by other remote cognitive radios can result in local policy and data amendments.

Evidently, cognitive radios need to operate under real-time conditions. Therefore, all computational processes involved in information processing including cognition relevant algorithms and their respective underlying structures are required to facilitate these operating requirements in terms of efficiency and effectiveness.

1.3 Purpose

The purpose of this standard is to define a language empowering the user to perform the following:

- a) Programmatically define policies
- b) Define these policies universally and unambiguously by means of strict adherence to First Order Logic
- c) Enable exchange of policies with other participating entities ensuring semantical correctness across the entire platform
- d) Meet the requirements and objectives as stipulated in IEEE Std 1900.5-2011, which are listed in A.1

1.4 Word usage

The word *shall* indicates mandatory requirements strictly to be followed to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).^{2,3}

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should* equals *is recommended that*).

² The use of the word *must* is deprecated and cannot be used when stating mandatory requirements, *must* is used only to describe unavoidable situations.

³ The use of *will* is deprecated and cannot be used when stating mandatory requirements, *will* is only used in statements of fact.

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

1.5 Notion of “policy” in its various uses

1.5.1 Current state

IETF RFC 3198 defines “policy” from the following two perspectives:

- A definite goal, course, or method of action to guide and determine present and future decisions. “Policies” are implemented or executed within a particular context (such as policies defined within a business unit)
- Policies as a set of rules to administer, manage, and control access to network resources

IEEE Std 1900.5-2011 defines “policy” as follows:

- A set of rules governing the behavior of a system

Despite the differences in these definitions, they share the call for some set of actions invoked by rules based on an assessment of the current situation.

In the realm of information processing, policies are usually expressed as algorithmically enforceable, deterministic statements of intent, to ensure that all devices subjected to a common policy also share a common situation assessment and implementation of relevant rules. One way to achieve this is to define policies by common datasets and event-driven actions based on these datasets. Implementations of these systems usually require installing standardized software modules.

Interaction with environmental incidents is then achieved by tying specific events to specific components that are compared to specific conditions that then in turn result in specific actions that are invoked by programs mostly written in a procedural language, for example, Java.

This way, an intrinsically complicated chain of event–condition–action (ECA) triples is created that is not only difficult to validate for logical soundness and completeness but is also difficult to maintain in a distributed environment.

This scenario results in policy actions to be modeled by a finite state machine automaton in which environmental parameters render discrete software states that in turn trigger certain predefined output actions. The advantage of this approach lies in the complete determinism of the runtime behavior of the finite state machine, but with increasing complexity in the environmental parameters and resulting possible reactions, this advantage turns into a disadvantage as all possible scenarios need to be considered by a policy designer up front.

1.5.2 Cognitive radio environment

Cognitive radios form a set of heterogeneous devices with widely varied capabilities. The establishment of a common policy or set of policies will therefore also require effective communication between these heterogeneous devices. Moreover, each device can find itself in a different situation than its neighbor. An approach of defining policy adhering decisions and thereof resulting actions based on statically defined assessments of a current situation as in a finite state machine design does not suit into the versatile and flexible world of cognitive radios.

This standard follows an innovative approach by empowering participating radios to make their own decisions on how to achieve objectives within the boundaries of an overarching set of policies.

It defines a semantic-preserving, knowledge-representing declarative language based on a standardized ontology language and production rule language to define policies and their respective rules, which are expressed in terms of the concepts in an (or more than one) underlying ontology.

Unlike the specific procedural ECA policy model that uniquely specifies each corresponding ECA triple, documents/programs written in a declaratively defined policy and subjected as input to an adequate logical reasoner gives the policy designer a higher level of expressiveness as they may define what should be done, not necessarily how to do it. It allows for the specification of criteria for arriving at a set of target objectives, which are logical consequences based on a model defined in an ontology and logical rules from a semantical structure defined in a rule language. Rationality, that is, logical consequence, is then generated by the policy engine reasoner that in turn can trigger actions in the embedding software executing system, thereby enforcing the policy into real actions and making sure the respective objective has been met or satisfied.

1.5.3 Underlying concepts

This standard uses current knowledge representation techniques in the form of ontologies in conjunction with logic-based computation in the form of Horn rules, which are generalized in the sense of Clark program completion to capture the requirement of negated clauses. An IEEE 1900.5.1 program essentially comprises a document of generalized Horn rules, defined in RIF-FLD, that may import OWL 2 RL ontologies presented in Extensible Markup Language (XML) serialization of OWL/Resource Description Framework (RDF) or may import other XML-based documents with or without accompanying schema definitions.

It builds on existing World Wide Web Consortium (W3C) standards for knowledge representation and logical reasoning.

1.5.3.1 Ontology

Computational ontologies are a means to formally model the structure of a system (concrete or abstract), that is, the relevant entities and relations that emerge from its observation, which are useful to our purposes.

Ontologies are part of a family of logic-based knowledge representation formalisms, which are based on Description Logics (DL). Description Logics form a decidable fragment of First Order Logic (FOL). Full FOL is well known to be only semi-decidable. Ontologies describe their domain in terms of concepts (a.k.a. classes), roles (a.k.a. properties or relationships), and individuals.

In particular, the policy language builds on concepts (classes) from a cognitive radio ontology (CRO, CRO2) that serves as a conceptual knowledge representation for a cognitive radio.

The CRO2 itself serves as an intermediary between more general base concepts like, for example, object, process, and quantity, which have been patterned in upper ontologies like, for example, DOLCE or NUVIO, and more specific concepts like Future Multiband Multiwaveform Modular Tactical Radio (FM3TR) radio components or transmitter application programming interfaces (APIs). Moreover, a mechanism is available that transforms the Spectrum Consumption Modeling XML Schema Definition (XSD) defined in IEEE Std 1900.5.2™ into an OWL 2 ontology in a fully automated way, thereby making the SCM data structures available to an OWL 2 reasoner.

Components of a cognitive radio are modeled as object subclasses, and actions that components may engage in are modeled as process subclasses. Abstract properties, that is, roles in description logic terms, are used to model relationships between objects and processes.

An important issue when designing or deciding on an ontology language is to choose a trade-off between language expressiveness and resulting complexity of the most relevant reasoning problems. This particularly applies to the real-time operating conditions of cognitive radios.

OWL 2 is the current web ontology language standard of the W3C consortium. Benefitting from experiences with the original OWL 1 specification, OWL 2 has undergone the following important improvements:

- a) Syntactic changes to make some common statements easier to construct
- b) New constructs to improve expressiveness
- c) Extended support for data types
- d) Definition of three sublanguages, called “profiles” (EL, QL, RL), offering different levels of trade-off between expressiveness and reasoning complexity

OWL 2 RL is a profile of OWL 2 aimed at applications that require scalable reasoning without sacrificing too much expressive power. It is designed to accommodate OWL 2 applications that can trade the full expressivity of the language for efficiency, as well as Resource Description Framework Schema (RDFS) applications that need some added expressivity. Furthermore, as this standard builds on an amalgamation of ontological and rule-based reasoning, OWL 2 RL is the prime candidate for ontological knowledge representation as OWL 2 RL reasoning systems themselves can be implemented using rule-based reasoning engines.

In most relevant scenarios, OWL 2 RL reasoning complexity is also polynomial-time (PTIME)-complete.⁴

In complexity theory, computational algorithms are assigned to complexity classes dependent on several parameters. PTIME is defined as the class of problems solvable by a *deterministic* algorithm where the completion time used by an algorithm is bounded by a polynomial function dependent on the size of the input (i.e., roughly n^c , for n is the size of the input and c is a constant). PTIME algorithms are also referred to as *tractable*.

For a complexity class $\mathcal{C}$, an algorithm A is said to be *$\mathcal{C}$ -complete* if A itself belongs to complexity class $\mathcal{C}$ and every other algorithm B belonging to class $\mathcal{C}$ can be polynomially reduced to A (i.e., A is at least as hard as any other problem in $\mathcal{C}$).

An algorithm A is said to be *$\mathcal{C}$ -hard* if every other algorithm B belonging to class $\mathcal{C}$ can be polynomially reduced to A , but A itself is not known to belong to complexity class $\mathcal{C}$; that is, A is at least as hard as any other problem in $\mathcal{C}$ but might be even harder.

In other words, using a PTIME-complete algorithm ensures that tight polynomial runtime bounds are known for that algorithm, an indispensable base property for an algorithm in a real-time environment.

Computational considerations are an important factor to assess the suitability of an algorithm particularly in a real-time environment as the following tables show.

⁴ Let L be a language accepted by a halting Turing machine M . Assume that for some function $f : \mathbb{N} \rightarrow \mathbb{N}$, it is the case that the number of transitions M makes before accepting or rejecting a string s is at most $f(|s|)$, where $|s|$ is the length of s . If M is deterministic, then we write $L \in \text{DTIME}(f)$; if M is nondeterministic, then we write $L \in \text{NTIME}(f)$. We define the class PTIME of polynomial-time computable problems as $\text{PTIME} = \bigcup_{k \in \mathbb{N}} \text{DTIME}(n^k)$ and the class NP of problems computable by nondeterministic polynomial-time Turing machines as $\text{NP} = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k)$.

Time complexity function: n computation steps with 1 μs/step						
	Size n					
	10	20	30	40	50	60
n	.00001 s	.00002 s	.00003 s	.00004 s	.00005 s	.00006 s
n ²	.0001 s	.0004 s	.0009 s	.0016 s	.0025 s	.0036 s
n ³	.001 s	.008 s	.027 s	.064 s	.125 s	.216 s
n ⁵	.1 s	3.2 s	24.3 s	1.7 min	5.2 min	13.0 min
2 ⁿ	.001 s	1.0 s	17.9 min	12.7 days	35.7 years	366 centuries
3 ⁿ	.059 s	58 min	6.5 years	3855 centuries	2 × 10 ⁸ centuries	1.3 × 10 ¹³ centuries
n!	3.629 s	771 centuries	8.4 × 10 ¹⁶ centuries			

Time complexity function: Improvement for size of problem solvable in 1 h by processor computational increase			
	Present	10 ² increase	10 ³ increase
n	N ₁	100 × N ₁	1000 × N ₁
n ²	N ₂	10 × N ₂	31.6 × N ₂
n ³	N ₃	4.64 × N ₃	10 × N ₃
n ⁵	N ₄	2.5 × N ₄	3.98 × N ₄
2 ⁿ	N ₅	N ₅ + 6.64	N ₅ + 9.97
3 ⁿ	N ₆	N ₆ + 4.19	N ₆ + 6.29
n!	13 (1 μs/step)	14 (1 μs/100 steps)	15 (1 μs/1000 steps)

Observe the well-known Stirling's formula, with $0 < \Theta < 1$, in the following equation:

$$n! = \sqrt{2\pi} \frac{n^{n+\frac{1}{2}}}{e^{n-\frac{\Theta}{12}}} = \sqrt{2n\pi} e^{\frac{\Theta}{12}} \left(\frac{n}{e}\right)^n$$

The well-known “Traveling Salesman Problem”⁵ to find an optimal route for $n + 1$ cities requires $n! / 2$ steps. If a route is given, for example, by guessing, checking of, that is, deciding the validity of, the route can be done in polynomial time. This type of decision problem is known to belong to class NP-complete (nondeterministic polynomial), whereas the route optimization problem is known to belong to class NP-hard.

However, even if an algorithm is known to have exponential time complexity, it may be preferable to an algorithm performing polynomially if its input can be reduced to a reasonable size; for example, an algorithm of complexity 2^n is performing faster than an algorithm of complexity n^5 for $n \leq 20$.

Therefore, the three main reasons for using the OWL 2 RL profile are as follows:

- OWL 2 RL is fully decidable.
- PTIME computational properties.
- Ease of integration of OWL 2 RL and rules, as an OWL 2 RL inferencer could be implemented using rules that in turn can then be used for expressing additional rules that will be necessary for expressing policies for Dynamic Spectrum Access.

Summarizing, the OWL 2 RL profile is designed such that all relevant reasoning tasks are tractable; that is, it can be solved by a deterministic algorithm with polynomial-time performance, which is a requirement stipulated in IEEE Std 1900.5-2011.

⁵ For a modest problem of 10 cities, $9! / 2 = 181\,440$ itineraries would have to be examined. With some confidence, this can be carried out by a computer, but what if there were 40 cities to visit? The number of itineraries is now gigantic: $39! / 2$, which is larger than 5×10^{44} . Even if one could examine 10^{15} tours per second—a pace that is much too fast even for the most powerful supercomputers, existing or projected—the required time for completing this calculation would be several billion lifetimes of the universe!

1.5.3.1.1 Description Logic notation

Although OWL 2 RL inferencing can be implemented by rule-based entailment, ontological and rule-based logical systems come from different historical backgrounds and use different notations. The following table gives an example showing the Class and ObjectProperty constructors that are supported in OWL 2 RL, also showing the naming correspondences between OWL 2 Functional Syntax, Description Logic Syntax, and First Order Logic Syntax.

OWL 2 constructor	DL syntax	Example	FOL syntax
Class/Concept Assertion	$a : C$	HarryPotter : Wizard	$C(a)$
Role/Property/Relation Assertion	$\langle a, b \rangle : P$	$\langle \text{HarryPotter}, \text{RonWeasley} \rangle : \text{hasFriend}$	$P(a, b)$
Negation	$\neg C$	$\neg \text{Wizard}$	$\neg C(x)$
SubClassOf	$C \sqsubseteq D$	$\text{Muggle} \sqsubseteq \neg \text{Wizard}$	$\forall x. C(x) \rightarrow D(x)$
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	$\text{Human} \sqcap \text{Male}$	$C_1(x) \wedge \dots \wedge C_n(x)$
unionOf	$C_1 \sqcup \dots \sqcup C_n$	$\text{Wizard} \sqcup \text{Student}$	$C_1(x) \vee \dots \vee C_n(x)$
oneOf	$\{a_1, \dots, a_n\}$	$\{\text{HarryPotter}\} \sqcup \{\text{RonWeasley}\}$	$x = a_1 \vee \dots \vee x = a_n$
transitiveProperty	$P^+ \sqsubseteq P$	$\text{ancestor}^+ \sqsubseteq \text{ancestor}$	$\forall x, y, z. (P(x, y) \wedge P(y, z)) \rightarrow P(x, z)$
inverseProperty	$P \equiv Q^-$	$\text{isFriend} \equiv \text{hasFriend}^-$	$\forall x, y. P(x, y) \Leftrightarrow Q(y, x)$
someValuesFrom	$\exists P.C$	$\exists \text{hasChild.Wizard}$	$\exists y. (P(x, y) \wedge C(y))$
allValuesFrom	$\forall P.C$	$\forall \text{hasPet}. (\{\text{Owl}\} \sqcup \{\text{Toad}\} \sqcup \{\text{Cat}\})$	$\forall y. (P(x, y) \rightarrow C(y))$
minCardinality	$\geq n P.C$	$\geq 2 \text{hasChild}$	$\exists y_1, \dots, y_n. \bigwedge_{1 \leq i \leq n} (P(x, y_i) \wedge C(y_i)) \wedge \bigwedge_{1 \leq i < j \leq n} y_i \neq y_j$
maxCardinality	$\leq n P.C$	$\leq 1 \text{hasChild}$	$\exists^{\leq n} y. P(x, y), \text{ with } \exists^{\leq n} y. P(x, y) \equiv \forall y_1 \dots \forall y_{n+1} P(x, y_1) \dots \wedge P(x, y_{n+1}) \rightarrow \bigvee_{1 \leq i < j \leq n+1, i \neq j} y_i = y_j$
functionalProperty	$T \sqsubseteq \leq 1 P$	hasMother	$\forall x, y, z. (P(x, y) \wedge P(x, z)) \rightarrow y = z$

$$\exists^{\geq n} x. P(x) \equiv \exists x_1 \exists x_2 \dots \exists x_n P(x_1) \wedge P(x_2) \dots \wedge P(x_n) \wedge \bigwedge_{1 \leq i < j \leq n, i \neq j} x_i \neq x_j$$

$$\exists^{\leq n} x. P(x) \equiv \forall x_1 \forall x_2 \dots \forall x_n \forall x_{n+1} P(x_1) \wedge P(x_2) \dots \wedge P(x_n) \wedge P(x_{n+1}) \rightarrow \bigvee_{1 \leq i < j \leq n+1, i \neq j} x_i = x_j$$

Several classes of DL ontologies have been defined with varying degrees of expressivity. See the list in 1.5.3.1.2.

1.5.3.1.2 Description Logic naming convention⁶

There are many varieties of Description Logic, and there is an informal naming convention, which roughly describes the operators allowed. The expressivity is encoded in the label for a logic starting with one of the following basic logics:

$\mathcal{AL}$	Attributive language. This is the base language that allows the following:
—	Atomic negation (negation of concept names that do not appear on the left-hand side of axioms)
—	Concept intersection
—	Universal restrictions
—	Limited existential quantification
$\mathcal{FL}$	Frame-based description language ^[2] that allows the following:
—	Concept intersection
—	Universal restrictions
—	Limited existential quantification

⁶ Unicode chars taken from <http://milde.users.sourceforge.net/LUCR/Math/unimathsymbols.pdf>.

—	Role restriction
$\mathcal{EL}$	Allows the following:
—	Concept intersection
—	Existential restrictions (of full existential quantification)

It is then followed by any of the following extensions:

$\mathcal{F}$	Functional properties
$\mathcal{E}$	Full existential qualification (existential restrictions that have fillers other than $\mathbf{T}$)
$\mathcal{U}$	Concept union
$\mathcal{C}$	Complex concept negation
$\mathcal{H}$	Role hierarchy (subproperties - <code>rdfs:subPropertyOf</code>)
$\mathcal{R}$	Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness
$\mathcal{O}$	Nominals. (Enumerated classes of object value restrictions - <code>owl:oneOf</code> , <code>owl:hasValue</code>)
$\mathcal{I}$	Inverse properties
$\mathcal{N}$	Cardinality restrictions (<code>owl:cardinality</code> , <code>owl:maxCardinality</code>)
$\mathcal{Q}$	Qualified cardinality restrictions (available in OWL 2, cardinality restrictions that have fillers other than $\mathbf{T}$)
$(\mathcal{D})$	Use of datatype properties, data values, or data types

1.5.3.1.2.1 Exceptions

Some canonical DLs that do not exactly fit this convention are as follows:

$\mathcal{S}$	Abbreviation for $\mathcal{ALC}$ with transitive roles
$\mathcal{FL}$	Sublanguage of $\mathcal{FL}$, which is obtained by disallowing role restriction. This is equivalent to $\mathcal{AL}$ without atomic negation
$\mathcal{FL}_0$	Sublanguage of $\mathcal{FL}$, which is obtained by disallowing limited existential quantification
$\mathcal{EL}^{++}$	Alias for $\mathcal{ELRO}$

1.5.3.1.2.2 Examples

As an example, $\mathcal{ALC}$ is a centrally important description logic from which comparisons with other varieties can be made. $\mathcal{ALC}$ is simply $\mathcal{AL}$ with a complement of any concept allowed, not just atomic concepts.

A further example, the description logic $\mathcal{SHOIQ}$ is the logic $\mathcal{ALC}$ plus extended cardinality restrictions, and transitive and inverse roles. The naming conventions are not purely systematic so that the logic $\mathcal{ALCOIN}$ might be referred to as $\mathcal{ALCNO}$ and abbreviations are made where possible; for example, $\mathcal{ALC}$ is used instead of the equivalent $\mathcal{ALU}\mathcal{E}$.

The Protégé ontology editor supports $\mathcal{SHOIQ}^{(D)}$. Three major biomedical informatics terminology bases, [SNOMED CT](#), [GALEN](#), and [GO](#), are expressible in $\mathcal{EL}$ (with additional role properties).

OWL 2 provides the expressiveness of *SHOIQ^D*, OWL-DL is based on *SHOIN^D*, and OWL-Lite is *SHIF^D*.

1.5.3.1.3 Example

The following excerpt is an example of an SCMImport ontology resulting from the automatic transformation of the SCM XSD schema of IEEE Std 1900.5.2 via an W3C Extensible Stylesheet Language Transformations (XSLT) stylesheet. As XSD schemas are serialized into tree structures as are RDF/XML ontologies, the XSLT stylesheet basically constructs a tree mapping from an XSD tree structure into an RDF/XML-encoded ontology tree structure. Once an appropriate XSLT stylesheet has been designed, the transformation process can be done anytime as a batch job by an XSLT processor, for example, in Java, whenever necessary, that is, whenever a change in the underlying XSD schema necessitates a change in the mapped ontology.

This example will be used to show the results, inferences, and consequences when using Policy Language constructs, that is, programs and documents, as defined in this standard.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://www.ieee.org/DyspanSC/1900/5/SCM"
targetNamespace="http://www.ieee.org/DyspanSC/1900/5/SCM" elementFormDefault="qualified">
  <element name="txModel" type="tns:TxModelType"/>
  <complexType name="TxModelType">
    <sequence>
      <element name="systemID" type="string" minOccurs="0" maxOccurs="1"/>
      <element name="purpose" minOccurs="1" maxOccurs="1">
        <simpleType>
          <restriction base="string">
            <enumeration value="AUTHORIZATION"/>
            <enumeration value="CONSTRAINT"/>
            <enumeration value="CONSUMPTION"/>
          </restriction>
        </simpleType>
      </element>
      <element name="confidence" type="tns:ConfidenceType" minOccurs="0" maxOccurs="1"/>
      <element name="referencePower" type="tns:ReferencePowerType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="spectrumMask" type="tns:SpectrumMaskType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="underlayMask" type="tns:UnderlayMaskType" minOccurs="0" maxOccurs="1"/>
      <element name="scmPowerMap" type="tns:SCMPowerMapType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="scmPropagationMap" type="tns:SCMPropagationMapType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="intermodulationMask" type="tns:IntermodulationMaskType" minOccurs="0" maxOccurs="unbounded"/>
      <element name="scmPlatform" type="tns:SCMPlatformType" minOccurs="0" maxOccurs="unbounded"/>
      <element name="scmLocation" type="tns:SCMLocationType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="scmSchedule" type="tns:SCMScheduleType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="minimumPowerSpectralFluxDensity" type="double" minOccurs="0" maxOccurs="1"/>
      <element name="scmPolicyOrProtocol" type="tns:SCMPolicyOrProtocolType" minOccurs="0" maxOccurs="1"/>
    </sequence>
  </complexType>
  <element name="rxModel" type="tns:RxModelType"/>
  <complexType name="RxModelType">
    <sequence>
      <element name="systemID" type="string" minOccurs="0" maxOccurs="1"/>
      <element name="purpose" type="string" minOccurs="1" maxOccurs="1"/>
      <element name="confidence" type="tns:ConfidenceType" minOccurs="0" maxOccurs="1"/>
      <element name="referencePower" type="tns:ReferencePowerType" minOccurs="2" maxOccurs="unbounded"/>
      <element name="underlayMask" type="tns:UnderlayMaskType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="scmPowerMap" type="tns:SCMPowerMapType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="scmPropagationMask" type="tns:SCMPropagationMapType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="intermodulationMask" type="tns:IntermodulationMaskType" minOccurs="0" maxOccurs="unbounded"/>
      <element name="scmPlatform" type="tns:SCMPlatformType" minOccurs="0" maxOccurs="unbounded"/>
      <element name="scmLocation" type="tns:SCMLocationType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="scmSchedule" type="tns:SCMScheduleType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="scmPolicyOrProtocol" type="tns:SCMPolicyOrProtocolType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  ....
  <complexType name="SCMPowerMapType">
```

```

<sequence>
  <element name="orientation" type="tns:OrientationType" minOccurs="1" maxOccurs="1"/>
  <element name="scanningRegion" type="tns:ScanningRegionType" minOccurs="0" maxOccurs="1"/>
  <element name="gainMap" type="tns:GainMapType" minOccurs="1" maxOccurs="1"/>
  <element name="locationIndex" type="int" minOccurs="0" maxOccurs="1"/>
  <element name="confidence" type="tns:ConfidenceType" minOccurs="0" maxOccurs="1"/>
</sequence>
</complexType>
....
<complexType name="SCMPropagationMapType">
  <sequence>
    <element name="propMap" type="tns:PropMapType" minOccurs="1" maxOccurs="1"/>
    <element name="antennaHeight" type="tns:AntennaHeightType" minOccurs="0" maxOccurs="1"/>
    <element name="locationIndex" type="int" minOccurs="0" maxOccurs="1"/>
    <element name="confidence" type="tns:ConfidenceType" minOccurs="0" maxOccurs="1"/>
  </sequence>
</complexType>

```

Schema	Ontology class	Equivalent to
<complexType name="TxModelType">	scmClass: scmClsTypeTxModel	(hasScmObjELEMsystemID max 1 scmClsELEMsystemID) and (hasScmObjELEMpurpose exactly 1 scmClsELEMpurpose) and (hasScmObjTYPEintermodulationMask only scmClsTypeIntermodulationMask) and (hasScmObjTYPEscmPlatform only scmClsTypeSCMPlatform) and (hasScmObjTYPEreferencePower min 1 scmClsTypeReferencePower) and (hasScmObjTYPEscmLocation min 1 scmClsTypeSCMLocation) and (hasScmObjTYPEscmPowerMap min 1 scmClsTypeSCMPowerMap) and (hasScmObjTYPEscmPropagationMap min 1 scmClsTypeSCMPropagationMap) and (hasScmObjTYPEscmSchedule min 1 scmClsTypeSCMSchedule) and (hasScmObjTYPEspectrumMask min 1 scmClsTypeSpectrumMask) and (hasScmObjELEMminimumPowerSpectralFluxDensity max 1 scmClsELEMminimumPowerSpectralFluxDensity) and (hasScmObjTYPEconfidence max 1 scmClsTypeConfidence) and (hasScmObjTYPEscmPolicyOrProtocol max 1 scmClsTypeSCMPolicyOrProtocol) and (hasScmObjTYPEunderlayMask max 1 scmClsTypeUnderlayMask)
<element name="systemID" type="string" minOccurs="0" maxOccurs="1"/>	scmClass: scmClsELEMsystemID	scmDataRangeELEMsystemID some xsd:string
<element name="purpose" minOccurs="1" maxOccurs="1"/>	scmClass: scmClsELEMpurpose	scmDataRangeELEMpurpose some xsd:string
	scmClass: scmClsATTRpurpose	scmDataRangeATTRpurpose some xsd:string
<element name="scmPowerMap" type="tns:SCMPowerMapType" minOccurs="1" maxOccurs="unbounded"/>		
<complexType name="SCMPowerMapType">	scmClass: scmClsTypeSCMPowerMap	(hasScmObjTYPEgainMap exactly 1 scmClsTypeGainMap) and (hasScmObjTYPEorientation exactly 1 scmClsTypeOrientation) and (hasScmObjELEMlocationIndex max 1 scmClsELEMlocationIndex) and (hasScmObjTYPEconfidence max 1 scmClsTypeConfidence) and (hasScmObjTYPEscanningRegion max 1 scmClsTypeScanningRegion)

Schema	Ontology object property	Domain	Range
<complexType name="TxModelType">	hasScmObjTYPE: hasScmObjTYPEtxModel	scmClsTypeSCMSet or scmClsTypeSystemModel or scmClsTypeUberModel	scmClsTypeTxModel
<element name="systemID" type="string" minOccurs="0" maxOccurs="1"/>	hasScmObjELEM: hasScmObjELEMsystemID	scmClsTypeRxModel or scmClsTypeSystemModel or scmClsTypeTxModel	scmClsELEMsystemID
<element name="purpose" minOccurs="1" maxOccurs="1"/>	hasScmObjELEM: hasScmObjELEMpurpose	scmClsTypeRxModel or scmClsTypeTxModel	scmClsELEMpurpose
	hasScmObjATTR: hasScmObjATTRpurpose	scmClsTypeSCMSet or scmClsTypeSystemModel	scmClsATTRpurpose
<complexType name="SCMPowerMapType">	hasScmObjTYPE: hasScmObjTYPEscmPowerMap	scmClsTypeRxModel or scmClsTypeTxModel	scmClsTypeSCMPowerMap

Schema	Ontology data property	Domain	Range
<element name="systemID" type="string" minOccurs="0" maxOccurs="1"/>	scmDataRangeELEMstring: scmDataRangeELEMsystemID		some xsd:string
<element name="purpose" minOccurs="1" maxOccurs="1"/>	scmDataRangeATTRstring		{"AUTHORIZATION", "CONSTRAINT", "CONSUMPTION"}

In short, the transformation process maps the inclusion of XSD structures within an outer structure by creating appropriate Classes, ObjectProperties, and DataProperties.

Every XSD element is mapped to its Class. Inclusion into a higher element is mapped by generating an appropriate ObjectProperty; for example, element Lower is included in element Higher, which is mapped by defining an ObjectProperty that takes HigherClass as its domain and LowerClass as its range. This applies to XSD complex types as they contain other types (complex or simple). XSD simple types have data elements; hence, there are two properties: an ObjectProperty that links the data element into its Class and a DataProperty that defines the valid data type for this element.

After mapping the XSD schema into its ontological structure, the ontology is filled with data that for the purpose of this example is created randomly, for example:

Individual	Class	ObjectProperty	Object individual or DataProperty
rxModel-d1e3	scmClsTypeRxModel inferred by ObjectProperty Definitions	hasScmObjELEMsystemID	systemID-d1e5
		hasScmObjELEMpurpose	purpose-d1e8
		hasScmObjTYPEconfidence	confidence-d1e11
		hasScmObjTYPEreferencePower	referencePower-d1e26
		hasScmObjTYPEreferencePower	referencePower -d1e33
		hasScmObjTYPEreferencePower	referencePower -d1e20
		hasScmObjTYPEunderlayMask	underlayMask-d1e205
		hasScmObjTYPEunderlayMask	underlayMask-d1e48
		hasScmObjTYPEunderlayMask	underlayMask-d1e232
		hasScmObjTYPEscmPowerMap	scmPowerMap-d1e525
		hasScmObjTYPEscmPowerMap	scmPowerMap-d1e796
		hasScmObjTYPEscmPowerMap	scmPowerMap-d1e691
		hasScmObjTYPEscmPowerMap	scmPowerMap-d1e426
		hasScmObjTYPEscmPropagationMask	scmPropagationMask-d1e967
		hasScmObjTYPEscmPropagationMask	scmPropagationMask-d1e1120
		hasScmObjTYPEscmPropagationMask	scmPropagationMask-d1e1267
		hasScmObjTYPEscmPropagationMask	scmPropagationMask-d1e1417
		hasScmObjTYPEintermodulationMask	intermodulationMask-d1e1616
		hasScmObjTYPEintermodulationMask	intermodulationMask-d1e1559
		hasScmObjTYPEscmPlatform	scmPlatform-d1e1718
		hasScmObjTYPEscmPlatform	scmPlatform-d1e1724
		hasScmObjTYPEscmPlatform	scmPlatform-d1e1712
		hasScmObjTYPEscmLocation	scmLocation-d1e1739
		hasScmObjTYPEscmLocation	scmLocation -d1e2023
		hasScmObjTYPEscmSchedule	scmSchedule-d1e2345
		hasScmObjTYPEscmSchedule	scmSchedule-d1e2360
		hasScmObjTYPEscmSchedule	scmSchedule-d1e2324
		hasScmObjTYPEscmPolicyOrProtocol	scmPolicyOrProtocol-d1e2372

		hasScmObjTYPEscmPolicyOrProtocol	scmPolicyOrProtocol-d1e2378
scmPropagationMask-d1e1267	scmClsTypeSCMPropagationMap inferred by ObjectProperty Definitions	hasScmObjTYPEpropMap	propMap-d1e1269
		hasScmObjELEMlocationIndex	locationIndex-d1e1407
		hasScmObjTYPEconfidence	confidence-d1e1410
scmPowerMap-d1e525	scmClsTypeSCMPowerMap inferred by ObjectProperty Definitions	hasScmObjTYPEorientation	orientation-d1e527
		hasScmObjTYPEscanningRegion	scanningRegion-d1e560
		hasScmObjTYPEgainMap	gainMap-d1e623
		hasScmObjELEMlocationIndex	locationIndex-d1e686
locationIndex-d1e1116	scmClsELEMlocationIndex inferred by DataProperty Definitions	scmDataRangeELEMlocationIndex	"88"^^xsd:init
locationIndex-d1e1407		scmDataRangeELEMlocationIndex	"87"^^xsd:init
locationIndex-d1e2018		scmDataRangeELEMlocationIndex	"7"^^xsd:init
locationIndex-d1e2314		scmDataRangeELEMlocationIndex	"73"^^xsd:init
locationIndex-d1e2368		scmDataRangeELEMlocationIndex	"51"^^xsd:init
locationIndex-d1e686		scmDataRangeELEMlocationIndex	"87"^^xsd:init
locationIndex-d1e957		scmDataRangeELEMlocationIndex	"57"^^xsd:init
systemID-d1e5	scmELEMsystemID inferred by DataProperty Definitions	scmDataRangeELEMpurpose	"e3k"^^xsd:string
purpose-d1e8	scmELEMpurpose inferred by DataProperty Definitions	scmDataRangeELEMpurpose	"AUTHORIZATION"^^xsd:string

These data would represent a partial structure of one rxModel, such as follows:

```

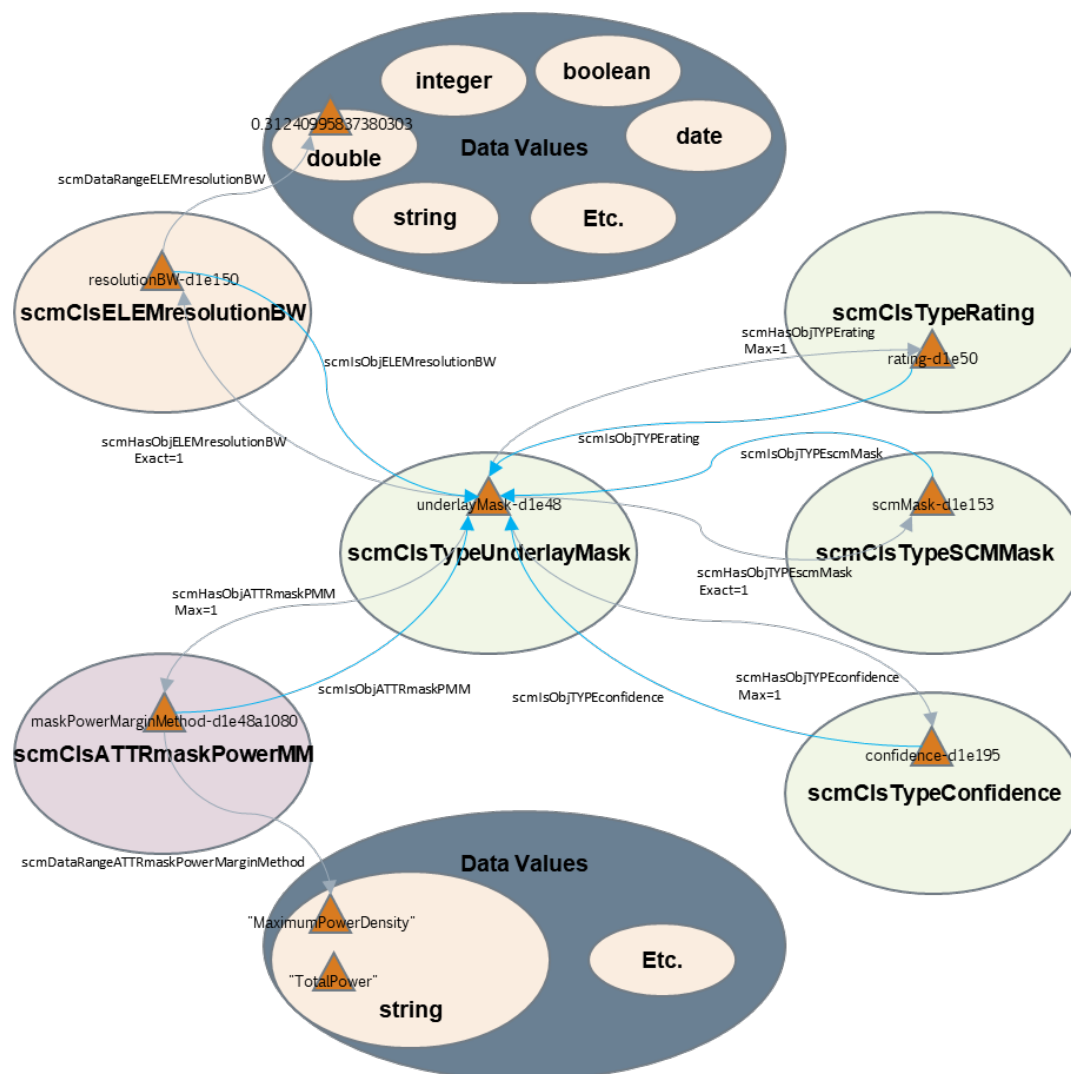
rxModel-d1e3
  systemId
    "e3k"
  purpose
    "AUTHORIZATION"
  confidence
  referencePower
  referencePower
  referencePower
  underlayMask
  underlayMask
  underlayMask
  scmPowerMap
    orientation
    scanningRegion
    gainMap
    locationIndex
      87 (accidentally identical to below locationIndex value as randomly chosen)
  scmPowerMap
  scmPowerMap
  scmPowerMap
  scmPropagationMask
  scmPropagationMask
  scmPropagationMask
    propMap
    locationIndex
      87 (accidentally identical to above locationIndex value as randomly chosen)
  scmPropagationMask
  scmIntermodulationMask
  scmIntermodulationMask

```

scmPlatform
scmPlatform
scmPlatform
scmLocation
scmLocation
scmSchedule
scmSchedule
scmSchedule
scmPolicyOrProtocol
scmPolicyOrProtocol

Note that although the values of the two listed locationIndex elements are identical, they are still different individuals as their position and thereby semantics differ.

Or the following figure illustrates another part of the SCM XSD schema.



1.5.3.2 Example

XSD Definition:

```
<complexType name="UnderlayMaskType">
  <sequence>
    <element name="rating" type="tns:RatingType" minOccurs="0" maxOccurs="1"/>
    <element name="resolutionBW" type="double" minOccurs="1" maxOccurs="1"/>
    <element name="scmMask" type="tns:SCMMaskType" minOccurs="1" maxOccurs="1"/>
    <element name="confidence" type="tns:ConfidenceType" minOccurs="0" maxOccurs="1"/>
  </sequence>
  <attribute name="maskPowerMarginMethod">
    <simplifiableType>
      <restriction base="string">
        <enumeration value="TotalPower"/>
        <enumeration value="MaximumPowerDensity"/>
      </restriction>
    </simplifiableType>
  </attribute>
</complexType>
```

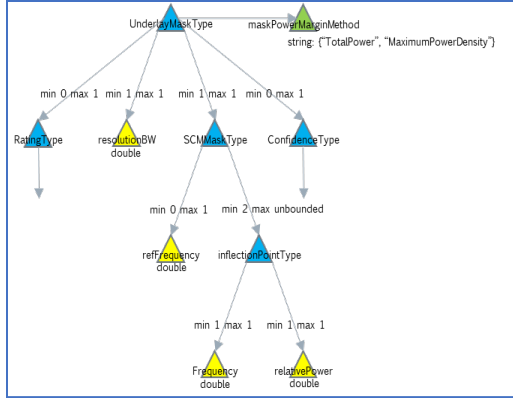
This example might comprise the following data. (Note that all data values have been generated randomly for demonstration purposes.)

UnderlayMask		
maskPowerMM		"MaximumPowerDensity"
rating		
resolutionBW		0.31240995837380303
scmMask		
confidence		

scmMask			
refFrequency			0.41806779374141345
inflectionPoint			
	frequency		0.8183843728079355
	relativePower		0.13869431150941813
inflectionPoint			
	frequency		0.7025515921987794
	relativePower		0.08709246715421293
inflectionPoint			
	frequency		0.9790438191524995
	relativePower		0.5207156711031952
inflectionPoint			
	frequency		0.9790438191524995
	relativePower		0.5207156711031952

confidence			
derivation			"Measured"
measure			0.9136669674393392
maxDwellTime			0.6279451397663207

rating			
ratedBW			0.8879678854377293
bwRatedList			
	bwRating		
		ratedBW	0.5739193323731228
		adjustment	0.309678052874383
		ratedBW	0.8383749090874749
		adjustment	0.9460398036991707
ratedBTP			0.5913791437377655
btpRatedList			
	btpRating		
		adjustment	0.7640732716259591
		btp	0.0995346553540335
	btpRating		
		adjustment	0.7381408796325543
		btp	0.5135368966975137
	btpRating		
		adjustment	0.5644742269868681
		btp	0.27362642250630664
	btpRating		
		adjustment	0.08175300775285621
		btp	0.45783901797041415
dcRatedList			
	dcRating		
		dc	0.689072366097927
		maxDwellTime	0.06284368845980981
		adjustment	0.48654810626249345
	dcRating		
		dc	0.6141235309390909
		maxDwellTime	0.5367390363896238
		adjustment	0.8295265108446387
hopRated			
popIndex			17



Structural View of UnderlayMask in SCM Schema XSD:

Ontology Definitions:

XSD Definition:

```
<complexType name="UnderlayMaskType">
  <sequence>
    <element name="rating" type="tns:RatingType" minOccurs="0" maxOccurs="1"/>
    <element name="resolutionBW" type="double" minOccurs="1" maxOccurs="1"/>
    <element name="scmMask" type="tns:SCMMaskType" minOccurs="1" maxOccurs="1"/>
    <element name="confidence" type="tns:ConfidenceType" minOccurs="0" maxOccurs="1"/>
  </sequence>
  <attribute name="maskPowerMarginMethod">
    <simpleType>
      <restriction base="string">
        <enumeration value="TotalPower"/>
        <enumeration value="MaximumPowerDensity"/>
      </restriction>
    </simpleType>
  </attribute>
</complexType>
```

Manchester Syntax OWL 2 Class Definition:

```
Class: scmClsTypeUnderlayMask
EquivalentTo:
  (hasScmObjTYPErating max 1 scmClsTypeRating) and
  (hasScmObjELEMresolutionBW exactly 1 scmClsELEMresolutionBW) and
  (hasScmObjTYPEscmMask exactly 1 scmClsTypeSCMMask) and
  (hasScmObjTYPEconfidence max 1 scmClsTypeConfidence) and
  (hasScmObjATTRmaskPowerMarginMethod max 1 scmClsATTRmaskPowerMarginMethod)
SubClassOf: scmTypes
```

RDF/XML triples for OWL 2 Individuals generated by XSLT:

```
<owl:NamedIndividual rdf:about="underlayMask-d1e48">
  <hasScmObjTYPErating rdf:resource="rating-d1e50"/>
  <hasScmObjELEMresolutionBW rdf:resource="resolutionBW-d1e150"/>
  <hasScmObjTYPEscmMask rdf:resource="scmMask-d1e153"/>
  <hasScmObjTYPEconfidence rdf:resource="confidence-d1e195"/>
  <hasScmObjATTRmaskPowerMarginMethod rdf:resource="maskPowerMarginMethod-d1e48a1080"/>
</owl:NamedIndividual>
isScmObj{ATTR ELEM TYPE}* inverseProperty hasScmObj{ATTR ELEM TYPE}*
hasScmObj{ATTR ELEM TYPE}* subProperty hasScmObj{ATTR ELEM TYPE}
hasScmObj{ATTR ELEM TYPE} subProperty hasScmObj
isScmObj{ATTR ELEM TYPE}* subProperty isScmObj{ATTR ELEM TYPE}
isScmObj{ATTR ELEM TYPE} subProperty isScmObj
hasScmObj subProperty scmObjectProperty
isScmObj subProperty scmObjectProperty
scmObjectProperty isTransitive
Therefore: All elements/types under one for example, rxModel are connected via property scmObject !
```

The ontology classes are then recursively defined by their respective contents.

Therefore, the class `scmClsTypeUnderlayMask` is defined as the intersection of those classes/sets of individuals that are linked via the respective ObjectProperties to range values in the appropriate class. Hence, an individual `underlayMask-d1e48` belongs to to class `scmClsTypeUnderlayMask` if and only if it is linked via ObjectProperty, such as follows:

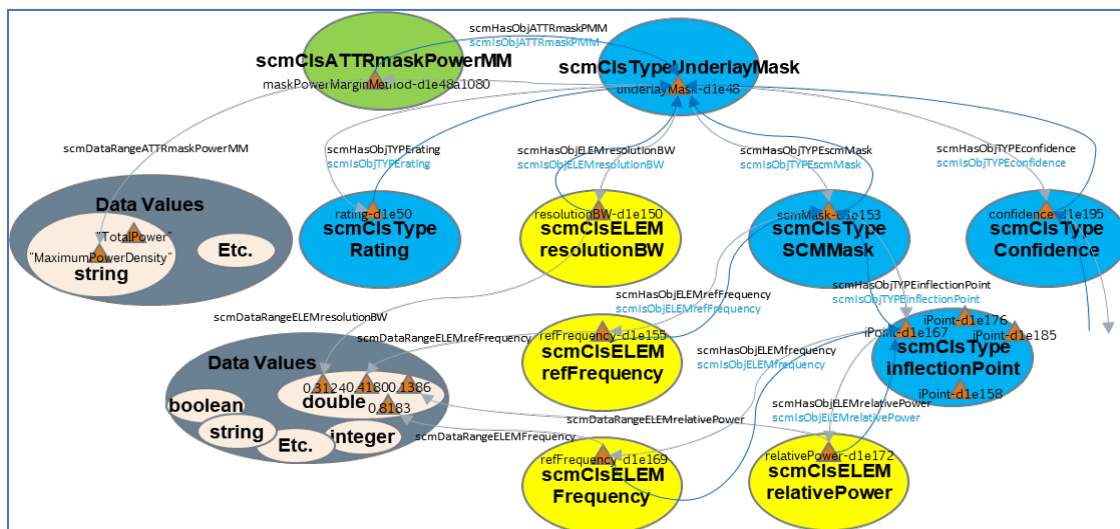
- hasScmObjTYPErating to another individual, for example, `rating-d1e50`, that itself belongs to class `scmClsTypeRating`
- hasScmObjTYPEscmMask to another individual, for example, `scmMask-d1e153`, that itself belongs to class `scmClsTypeSCMMask`
- hasScmObjTYPEconfidence to another individual, for example, `confidence-d1e195`, that itself belongs to class `scmClsTypeConfidence`
- hasScmObjELEMresolutionBW to another individual, for example, `resolutionBW-d1e150`, that itself belongs to class `scmClsELEMresolutionBW`
- hasScmObjATTRmaskPowerMarginMethod to another individual, for example, `maskPowerMarginMethod-d1e48a1080`, that itself belongs to class `scmClsATTRmaskPowerMarginMethod`

Individual `resolutionBW-d1e150` itself belongs to class `scmClsELEMresolutionBW` if and only if it is linked via DataProperty `scmdataRangeELEMresolutionBW` to a data value in the defined range of the DataProperty, for example, value `0.31240995837380303`.

Likewise, individual `maskPowerMarginMethod-d1e48a1080` itself belongs to class `scmClsATTRmaskPowerMarginMethod` if and only if it is linked via DataProperty `scmdataRangeATTRmaskPowerMarginMethod` to a data value in the defined range of the DataProperty, for example, value `MaximumPowerDensity`. The valid data range of DataProperty `scmdataRangeATTRmaskPowerMarginMethod` is restricted by definition of the DataProperty to strings of the set `{"MaximumPowerDensity", "TotalPower"}`.

By appropriate definition of ObjectProperties, DataProperties, and Classes, the ontology reasoner infers by itself the allocation of individuals to their proper classes. Hence, a CR would just receive an SCM XSD schema-compatible XML file, read in the data, and the reasoner allocates the various individuals to their proper classes.

If the SCM XSD schema undergoes revision changes, a new run of an XSLT stylesheet processor would automatically adapt the changes into the structure of the SCM ontology.



ObjectProperty hasScmObjTYPEscmMask is defined as shown in this table; other ObjectProperties of type hasScmObjTYPE/ELEM/ATTR follow this pattern.

ObjectProperty:	scmHasObjTYPEscmMask
SubPropertyOf:	scmHasObjTYPE
Characteristics:	Asymmetric, Functional
Domain:	scmClsTypeSpectrumMask or scmClsTypeUnderlayMask
Range:	scmClsTypeSCMMask
InverseOf:	scmIsObjTYPEscmMask
ObjectProperty:	scmHasObjTYPE
SubPropertyOf:	scmHasObj
InverseOf:	scmIsObjTYPE

Object Properties isScmObjTYPE/ELEM/ATTR are the inverse of hasScmObjTYPE/ELEM/ATTR; therefore, if underlayMask-d1e48[hasScmObjTYPErating→rating-d1e150] then rating-d1e150[isScmObjTYPErating→underlayMask-d1e48] holds.

ObjectProperties hasScmObj and isScmObj are defined as super-properties of hasScmObjTYPE/ELEM/ATTR and isScmObjTYPE/ELEM/ATTR, respectively. They are also defined as transitive, therefore, if

underlayMask-d1e48[hasScmObjTYPErating→rating-d1e150]
rating-d1e150[hasScmObjTYPEbtpRatedList→btpRatedList-d1e79]
btpRatedList-d1e79[hasScmObjTYPEbtpRating→btpRating-d1e99]

then

underlayMask-d1e48[hasScmObjRating→rating-d1e150]
rating-d1e150[hasScmObjBtpRatedList→btpRatedList-d1e79]
btpRatedList-d1e79[hasScmObjBtpRating→btpRating-d1e99]

and due to transitivity of hasScmObj

underlayMask-d1e48[hasScmObjRating→btpRating-d1e99]

and analogously for isScmObj.

With these structures in place, the following DLQuery would give all individuals anywhere in the SCM that have subordinate individuals with frequency values that are greater than or equal to 0.7 within the SCM. As DLQueries can be stored as ontology classes, this coincides with a definition of a class of individuals with just those properties.

hasScmObj some (scmDataRangeELEMfrequency min 1 xsd:double[>="0.7"^^xsd:double])

imAmplificationMask-d1e1669
imCombiningMask-d1e1567
imCombiningMask-d1e1621
inflectionPoint-d1e1641
inflectionPoint-d1e1683
inflectionPoint-d1e176
inflectionPoint-d1e167
inflectionPoint-d1e185
inflectionPoint-d1e212
intermodulationMask-d1e1559
intermodulationMask-d1e1616
inflectionPoint-d1e1659

inflectionPoint-d1e381
inflectionPoint-d1e1605
rxModel-d1e3
scmData-d1e1
scmMask-d1e153
scmMask-d1e210
scmMask-d1e367
underlayMask-d1e205
underlayMask-d1e221
underlayMask-d1e232
underlayMask-d1e48

The next query provides a finer specification in that only subordinate individuals of class `scmClsTypeSCMMask` that have frequency values greater than or equal to 0.7 are wanted. Set-theoretically, this solution set is simply the intersection of the previous solution set with class `scmClsTypeSCMMask`.

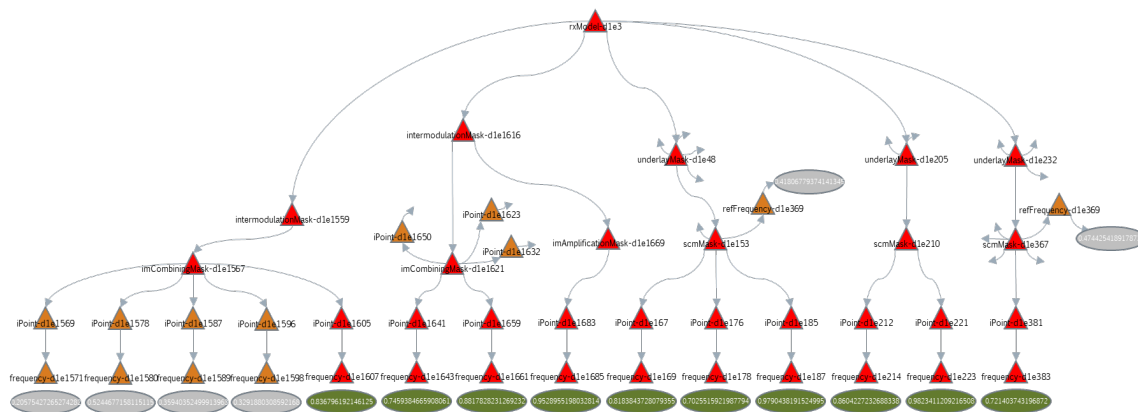
hasScmObj some (scmDataRangeELEMfrequency min 1 xsd:double[>="0.7"^^xsd:double]) and
scmClsTypeSCMMask

imAmplificationMask-d1e1669
imCombiningMask-d1e1567
imCombiningMask-d1e1621
scmMask-d1e210
scmMask-d1e367
scmMask-d1e153

Then the next query provides a yet finer specification in that only those subordinate individuals of class `scmClsTypeSCMMask` are wanted that have frequency values greater than or equal to 0.7 and `refFrequency` values of greater than or equal to 0.4. Set-theoretically, again this solution set is simply the intersection of the first solution set with class `scmClsTypeSCMMask` intersected with those individuals that fall in the proper range of property `scmDataRangeELEMrefFrequency`.

hasScmObj some (scmDataRangeELEMfrequency min 1 xsd:double[>="0.7"^^xsd:double]) and
scmClsTypeSCMMask
hasScmObj some (scmDataRangeELEMrefFrequency min 1 xsd:double[>="0.4"^^xsd:double])

imAmplificationMask-d1e1669
scmMask-d1e153
scmMask-d1e367



1.5.3.3 Rules⁷

While OWL 2 RL serves to express and model structural knowledge, that is, knowledge pertaining to CR components, intercomponent relationships and obligatory, permissive, or prohibitive actions resulting thereof, a *rule language* is used to steer environmentally aware behavior by exploiting this structural knowledge expressed in ontologies into actions.

To be able to implement the necessary functionalities of dynamic spectrum access systems, the rule language defined in this standard allows for inclusion of externally defined procedural attachments—programs written in a procedural language that can be invoked by the rules.

As cognitive radios may also communicate facts from their respective knowledge bases to other CRs, any data exchange with remote cognitive radios not only requires adherence to suitable communication protocols but also the ability to process knowledge base updates preserving the semantics of the received information. This, too, is accomplished by the standardized use of logical rules.

Many different logical rule systems are available. The W3C has facilitated the following standardized semantics preserving interchange formats. Logical rule systems fall into two broad categories—*logic-based* and *rules with actions*—and consequently the W3C has defined two different interchange formats, named dialects:

- a) RIF-BLD (Basic Logic Dialect)
- b) RIF-PRD (Production Rule Dialect)
- c) RIF-Core, defined as proper, that is, not maximum possible, intersection of dialects RIF-BLD and RIF-PRD

As an overarching dialect, the W3C has also defined the following:

- d) RIF-FLD (Framework for Logic Dialects)

The RIF-FLD is a formalism for specifying all logic dialects of RIF, that is, RIF-BLD and RIF-Core (albeit not RIF-PRD, as this is not a logic-based RIF dialect).

RIF-FLD is a formalism in which both syntax and semantics are described through a number of mechanisms that are commonly used for various logic languages but are rarely brought all together. An amalgamation of several different mechanisms is required because the framework must be broad enough to accommodate several different types of logic languages and because various advanced mechanisms are needed to facilitate translation into a common framework. RIF-FLD gives precise definitions to these mechanisms but allows well-defined aspects to vary. The design of RIF envisions that future standard logic dialects will be based on RIF-FLD. Therefore, for any RIF dialect to become a standard, its development should start as a specialization of RIF-FLD and extensions to (or deviations from) FLD should be justified.

IEEE Std 1900.5-2011 stipulates the necessity to express negation in statements. As negation is only properly supported as part of RIF-FLD, this standard uses the formalisms defined in RIF-FLD to define its own RIF dialect RIF-DSA, which is as close as possible to RIF-Core but due to required negation support expands RIF-Core.

Rules and ontologies complement each other. Neither is a subset of the other. However, OWL 2 RL has been defined such that it may be implemented using RIF-Core rules.

⁷ Currently, only interpreters (not compilers) are available to process rules, for example, Python FUXI, HORNLOG, and so on.

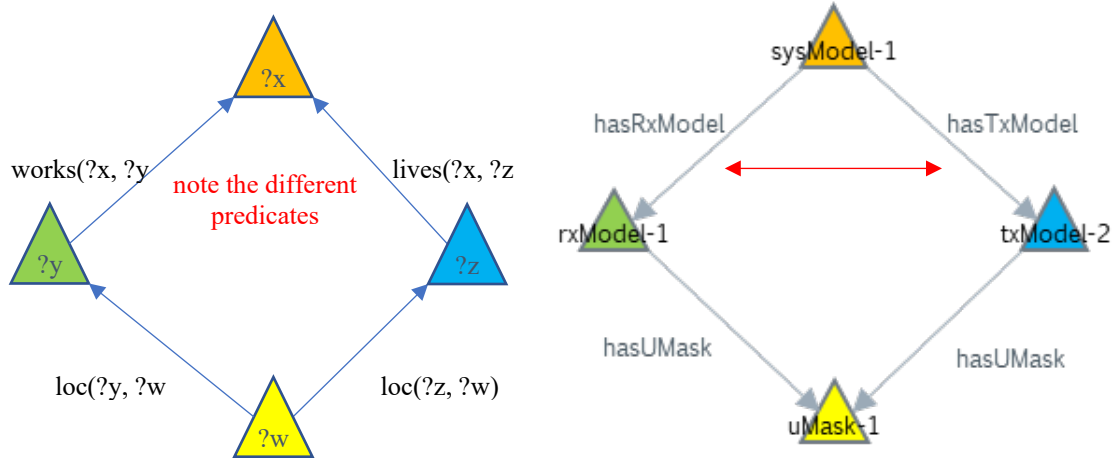
All OWL 2 profiles are ontology definition languages. Ontologies are based on Description Logic (DL), which in turn is a decidable subset of First Order Logic (FOL) where the decidability is due in large part to DLs having (a form of) the tree model property. This property says that a DL class C has a model (an interpretation I in which C^I is nonempty) iff⁸ C has a tree-shaped model, that is, one in which the interpretation of properties defines a tree-shaped directed graph (Gaifman graph).

For example, the DL class $\exists P.C$ corresponds to the FOL formula $\exists y.(P(x, y) \wedge C(y))$, where the property predicate P acts as a so-called “guard.” One obvious consequence of this restriction is that it is impossible to describe classes whose instances are related to another anonymous individual via differing property paths.

This requirement restricts the way variables and quantifiers can be used. In particular, quantifiers must be relativized via atomic formulas (as in the guarded fragment of FOL⁹); that is, the quantified variable must occur in a property predicate along with the free variable (DL classes correspond to formulas with one free variable).

For example, it is impossible in OWL 2 to assert certain “diamond-shaped relations”; that is, individuals who live and work at the same location are “LocalWorkers.” This is easily achieved with a Horn rule: any RIF dialect, for example:

$\text{LocalWorker}(?x) \leftarrow \text{work}(?x, ?y) \wedge \text{live}(?x, ?z) \wedge \text{loc}(?y, ?w) \wedge \text{loc}(?z, ?w)$ ¹⁰



In the case of an SCM ReceiverModel rxModel , two object property chains might be defined as follows:

$\text{sysRxModel}(x, z) = \text{hasScmObjTYPErxModel}(x, y) \circ \text{hasScmObjTYPEunderlayMask}(y, z)$
 $\text{sysTxModel}(x, z) = \text{hasScmObjTYPEtxModel}(x, y) \circ \text{hasScmObjTYPEunderlayMask}(y, z)$

Composition of object properties works like object property transitivity; in fact, transitivity is a special case of composition. Two individuals are connected via a composition property iff a third individual exists that connects them. For example, in the previous case:

$x[\text{sysRxModel} \rightarrow z]$ iff there is a ' y_1 ' such that $x[\text{hasScmObjTYPErxModel} \rightarrow y_1]$ and $y_1[\text{hasScmObjTYPEunderlayMask} \rightarrow z]$.

⁸ “iff” is used throughout this document as short for “if and only if” as it is used in the same way in the underlying logical and mathematical literature.

⁹ See Grädel [B2]. (The numbers in brackets correspond to those of the bibliography in Annex H.)

¹⁰ Variables are marked by a leading “?” character in RIF.

$x[\text{sysTxModel} \rightarrow z]$ iff there is a ' y_2 ' such that $x[\text{hasScmObjTYPEtxModel} \rightarrow y_2]$ and $y_2[\text{hasScmObjTYPEunderlayMask} \rightarrow z]$.

An object property P is transitive exactly when $P \sqsubseteq P \circ P$.

Then, when building classes on, for example, range criterium over the above two object property chains, one will always have to quantize over one of the two independent variables, such as follows:

$$\text{rxSysModelUMaskCls}(x) \equiv \exists \text{ sysRxModel. } \{\text{range of underlay masks}\}$$

$$\text{txSysModelUMaskCls}(x) \equiv \exists \text{ sysTxModel. } \{\text{range of underlay masks}\}$$

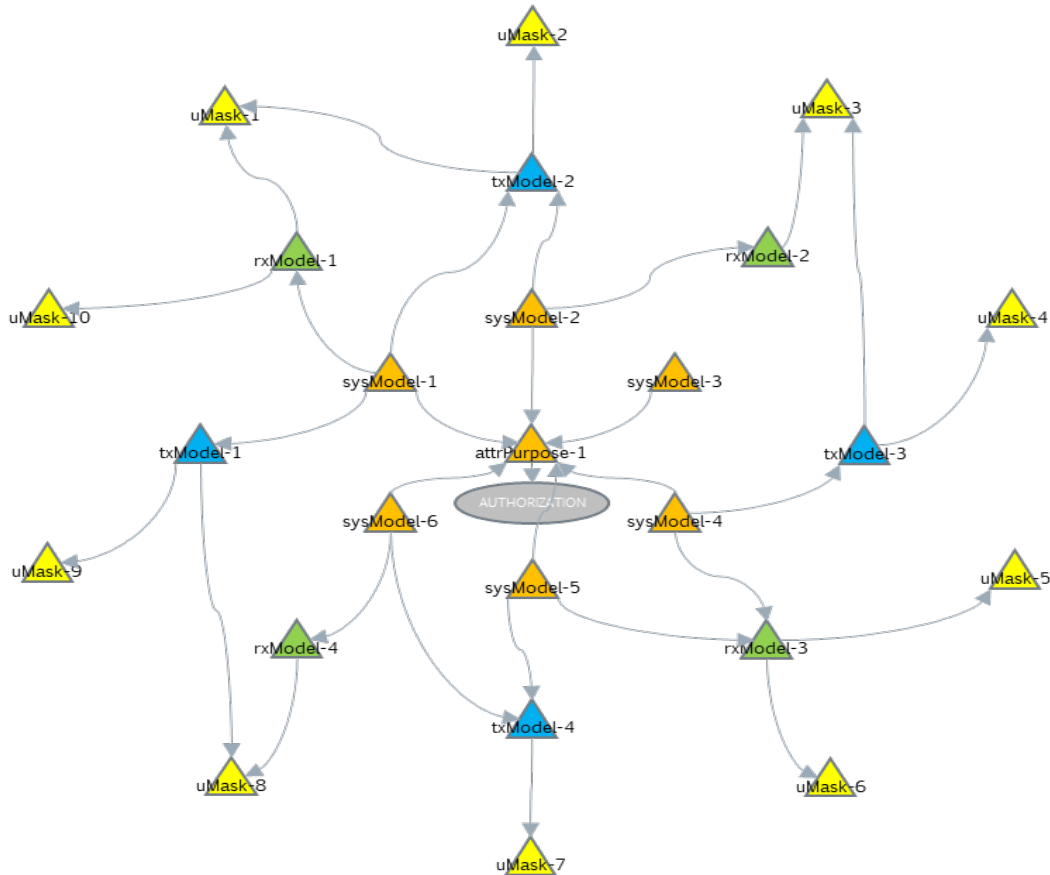
and then building the following intersection:

$$\text{sysModelUMaskCls} \equiv \text{rxSysModelUMaskCls}(x) \sqcap \text{txSysModelUMaskCls}(x)$$

would yield system models that have underlay masks in the set of underlay mask individuals, *but not necessarily identical ones*, whereas rule

$$\begin{aligned} \text{sysModelUMaskCls} (?x) \leftarrow & \text{hasScmObjTYPErxModel} (?x, ?y) \wedge \text{hasScmObjTYPEunderlayMask} (?y, ?w) \wedge \\ & \text{hasScmObjTYPEtxModel} (?x, ?z) \wedge \text{hasScmObjTYPEunderlayMask} (?z, ?w) \end{aligned}$$

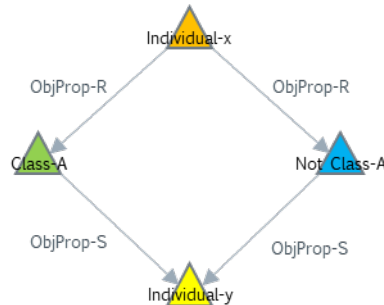
precisely gives the requested class as now both `hasScmObjTYPEunderlayMask` predicates have the identical “ $?z$ ” parameter as arguments.



On the other hand, rules cannot assert the information that a person is either a man or a woman or that Signal is either SignalInRadio or SignalInSpace.¹¹

This information is easily expressed in OWL 2 using disjoint union.¹²

OWL 2 can map the following “diamond-shaped” relationship:



One key difference between ontologies and rule systems lies in the Open World Assumption (OWA) and in the ability of ontology reasoners to infer the existence of new individuals from the knowledge basis, whereas rule systems use the full or partial variants of the Closed World Assumption (CWA) and infer only on existing individuals; for example, a person not listed in the telephone directory does not infer that the person does not have a telephone (OWA), whereas if there is no flight to ABC listed in the departure list of, for example, NYC then we can infer that there is no direct flight between these two destinations (CWA).¹³

Ontologies serve as knowledge representation systems. The CWA is not intuitive in ontologies as one does NOT want to infer that, for example, $\text{Person} \sqsubseteq \text{American}$, that is, all persons are Americans, just because a certain ontology only contains information on American persons.

Combining ontology and rule languages is tantamount to combining a logic whose decidability is due to the fact that we can restrict our attention to “nice” models (but with individuals whose existence may be implied by a knowledge base as in ontologies) with a logic whose decidability is owed to restricting attention to “known” individuals (but with arbitrary relations between them as in rules). Unsurprisingly, this and similar combinations may become undecidable.

A problem is defined as “undecidable” if there is logical proof that an algorithmic solution to that problem does not exist. In other words, there is irrefutable logical proof that an algorithm for that particular circumstance can and will never be found.

For rule extensions of Description Logic languages such as OWL 2, the undecidability of the consistency problem is often proved by showing that the ontology extension with rules makes it possible to encode a known undecidable domino or tiling problem as an ontology consistency problem. In particular, it is well known that such combined rule and ontology languages only need the ability to represent an infinite

¹¹ Disjunctive formulas like $a \vee b$ cannot be expressed in logic programs.

The logical dependency of a fact “a” entailed from “b” or from “c” can be expressed by means of two clauses $a \leftarrow b$ and $a \leftarrow c$. However, the fact that $(a \text{ or } b)$ is entailed by c cannot be expressed in Horn rules.

In logic programming, a disjunctive statement $a \vee b$ can only be proven by proving a or by proving b .

¹² Although axiom *DisjointUnion*($C \text{ } CE_1 \dots CE_n$) axiom is not supported in OWL 2 RL, this can be accomplished by combining the sequence of following axioms: *EquivalentClasses*($C \text{ } \text{ObjectUnionOf}(CE_1 \dots CE_n)$) and *DisjointClasses*($CE_1 \dots CE_n$).

¹³ The OWA forms a monotonic inference system, whereas the CWA is an example of a nonmonotonic inference rule. An inference rule is nonmonotonic if the *addition* of new axioms can *decrease* the set of theorems that previously held. In this example, we only need to add a direct flight between ABC and NYC; then the previous entailment that there is no direct flight between the two locations does not hold any more.

two-dimensional grid in order for consistency to become undecidable. With the addition of rules, such an encoding is trivial.¹⁴

For example, given two ontology properties *x-succ* and *y-succ*, the rule:

$$\text{sameAs}(\text{?c}; \text{?e}) \leftarrow \text{x-succ}(\text{?a}; \text{?b}) \wedge \text{y-succ}(\text{?b}; \text{?c}) \wedge \text{y-succ}(\text{?a}; \text{?d}) \wedge \text{x-succ}(\text{?d}; \text{?e})$$

along with the assertion that every grid node is related to exactly one other node by each of *x-succ* and *y-succ* allows such a grid to be represented. This would be possible even without the use of the *sameAs* atom in the rule consequent—it would only be necessary to establish appropriate relationships with a “diagonal” property:

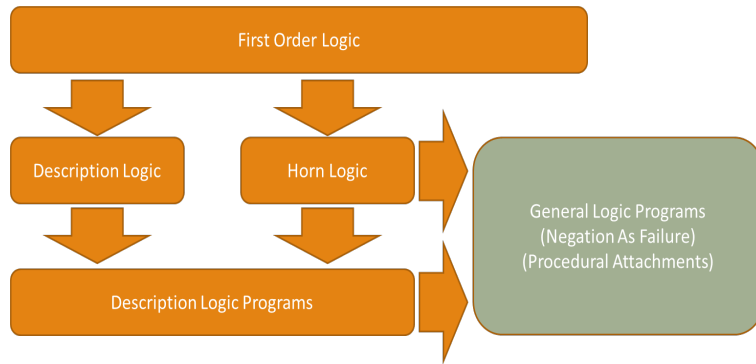
$$\begin{aligned} \text{diagonal}(\text{?a}; \text{?c}) &\leftarrow \text{x-succ}(\text{?a}; \text{?b}) \wedge \text{y-succ}(\text{?b}; \text{?c}) \\ \text{diagonal}(\text{?a}; \text{?e}) &\leftarrow \text{y-succ}(\text{?a}; \text{?d}) \wedge \text{x-succ}(\text{?d}; \text{?e}) \end{aligned}$$

and additionally assert that every grid node is related to *exactly one* other node by diagonal.

Therefore, to maintain decidability, syntactical restrictions need to be enforced on the combination of rule languages with ontology languages.

1.5.3.4 Relationship with First Order Logic

The task of defining a language is futile if the language cannot be used in a meaningful way to transport meaning between the two communicating entities. In terms of a formal language, this equates to the ability to produce an interpreter and/or compiler that translates the high-level formal language into algorithms that are in machine-readable and executable format. Therefore, the formal language should disallow and even actively prohibit to express programs that are known to be algorithmically undecidable.



The meaning of *SHOIQ* is usually given by a model theory. However, *SHOIQ* can also be seen in terms of a correspondence to FOL, where classes correspond to unary predicates, properties correspond to binary predicates, and subclass/property axioms correspond to implication.

To be more precise, individuals are equivalent to FOL constants, classes, and class expressions are equivalent to FOL formulas with one free variable, and properties (and property expressions when supported by the DL) are equivalent to FOL formulas with two free variables. Class and property inclusion axioms are equivalent to FOL sentences consisting of an implication between two formulas with the free variables universally quantified at the outer level. For example, a DL axiom of the form $C \sqsubseteq D$ is

¹⁴ This problem can be formalized as follows. A tiling system is a quadruple $D = (D, d_0, H, V)$, where D is a finite set of tiles, with a countably infinite supply of copies of each tile, and $d_0 \in D$, $H \subseteq D \times D$, $V \subseteq D \times D$. A *tiling* by D is a function $f: \mathbb{N} \times \mathbb{N} \rightarrow D$ such that the following hold:

$f(0, 0) = d_0$
 $(f(m, n), f(m + 1, n)) \in H$ for all $m, n \in \mathbb{N}$
 $(f(m, n), f(m, n + 1)) \in V$ for all $m, n \in \mathbb{N}$

It is a well-known theorem that the problem of determining, given a tiling system, whether there is a tiling by that system is undecidable.

equivalent to a FOL sentence of the form $\forall x. C(x) \rightarrow D(x)$. DL axioms of the form $a : C$ and $\langle a, b \rangle : P$ correspond to ground atoms $C(a)$ and $P(a, b)$. Finally, DL axioms asserting the transitivity of a property P , the functionality of a property P , and that property Q is the inverse¹⁵ of property P are, respectively, equivalent to FOL sentences of the form:

$$\forall x, y, z. (P(x, y) \wedge P(y, z)) \rightarrow P(x, z)$$

$$\forall x, y, z. (P(x, y) \wedge P(x, z)) \rightarrow y = z$$

$$\forall x, y. P(x, y) \Leftrightarrow Q(y, x)$$

The previous figure summarizes these equivalences and shows the FOL formulas corresponding to the DL class expressions described as follows, where a ; b are constants and x is the free variable. These formulas can be composed in the obvious way, for example: $\exists R. (C \sqcap D) \equiv \exists y. (P(x, y) \wedge (C(y) \wedge D(y)))$.

Description Logic	First Order Logic	RIF-DSA
$a : C$	$C(a)$	ClassAssertion(C a) or $\langle a \rangle \# C$
$\langle a, b \rangle : P$	$P(a, b)$	ObjectPropertyAssertion(P a b) or $\langle a \rangle [\langle P \rangle \rightarrow b]$
$C \sqsubseteq D$	$\forall x: C(x) \Rightarrow D(x)$	SubClassOf(C D) or Forall ?x (D(?x) :- C(?x))
$P^+ \sqsubseteq P$	$\forall x, y, z: (P(x, y) \wedge P(y, z) \Rightarrow P(x, z))$	TransitiveObjectProperty(P) or Forall ?x ?y ?z (P(?x ?z) :- And(P(?x ?y) P(?y ?z)))
$\top \sqsubseteq \leq 1P$	$\forall x, y, z: (P(x, y) \wedge P(x, z) \Rightarrow (y = z))$	FunctionalObjectProperty(P) or Forall ?x ?y ?z (?x = ?z) :- And(P(?x ?y) P(?x ?z)))
$P \equiv Q^-$ or $P \equiv \text{Inv}(Q)$	$\forall x, y: (P(x, y) \Leftrightarrow Q(y, x))$	InverseObjectProperties(P Q) or Forall ?x ?y (P(?x ?y) :- Q(?y ?x))
$C_1 \sqcap C_2 \sqcap \dots \sqcap C_n$	$C_1(x) \wedge C_2(x) \wedge \dots \wedge C_n(x)$	EquivalentClasses(AndClass ObjectIntersectionOf(C ₁ C ₂ ... C _n)) or Forall ?x (AndClass(?x) :- And(C ₁ (?x) C ₂ (?x) ... C _n (?x)))
$C_1 \sqcup C_2 \sqcup \dots \sqcup C_n$	$C_1(x) \vee C_2(x) \vee \dots \vee C_n(x)$	EquivalentClasses(OrClass ObjectUnionOf(C ₁ C ₂ ... C _n)) or Forall ?x (OrClass(?x) :- Or(C ₁ (?x) C ₂ (?x) ... C _n (?x)))
$\neg C$	$\neg C(x)$	EquivalentClasses(NotClass ObjectComplementOf(C))
$\{a_1, a_2, \dots, a_n\}$	$x = a_1 \vee x = a_2 \vee \dots \vee x = a_n$	EquivalentClasses(C ObjectOneOf(a ₁ a ₂ ... a _n))
$\exists P.C$	$\exists y: (P(x, y) \wedge C(y))$	EquivalentClasses(D ObjectSomeValuesFrom(P C))
$\forall P.C$	$\forall y: (P(x, y) \Rightarrow C(y))$	EquivalentClasses(D ObjectAllValuesFrom(P C))
$\geq nP.C$	$\exists y_1, y_2, \dots, y_n: (\bigwedge_{1 \leq i \leq n} (P(x, y_i) \wedge C(y_i)) \wedge \bigwedge_{1 \leq i < j \leq n} y_i \neq y_j)$	EquivalentClasses(D ObjectMinCardinality(n P C))
$\leq (n-1)P.C$	$\forall y_1, y_2, \dots, y_n: (\bigwedge_{1 \leq i \leq n} (P(x, y_i) \wedge C(y_i)) \Rightarrow \bigvee_{1 \leq i < j \leq n} y_i = y_j)$	EquivalentClasses(D ObjectMaxCardinality(n-1 P C))

¹⁵ Equivalence axioms can be reduced to a symmetrical pair of inclusion axioms.

As a notational convention, we will throughout the standard use a and b for constants and w , x , y , and z for variables.

1.6 Related work

1.6.1 IEEE Std 1900.5-2011

IEEE Std 1900.5-2011 sets a framework for requirements on a policy language and associated architecture for cognitive wireless devices.

2. Normative references

The following referenced documents are indispensable for the application of this document. Each referenced document is cited and its relationship to this document is explained. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 1900.5TM-2011, IEEE Standard for Policy Language Requirements and System Architectures for Dynamic Spectrum Access Systems.^{16, 17}

IEEE Std 1900.5.2TM, IEEE Standard for Method for Modeling Spectrum Consumption.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.¹⁸

Resource Description Framework (RDF): A standard by the W3C for expressing statements about resources.

Resource Description Framework Schema (RDFS): A set of classes with certain properties using the RDF extensible knowledge representation language, providing basic elements for the description of ontologies, otherwise called RDF vocabularies, intended to structure RDF resources.

3.2 Acronyms and abbreviations

API	Application Programming Interface
CR	Cognitive Radio
CRO, CRO2	Cognitive Radio Ontology, version 2
CWA	Closed World Assumption

¹⁶ The IEEE standards or products referred to in Annex H are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.

¹⁷ IEEE publications are available from the Institute of Electrical and Electronics Engineers (<http://standards.ieee.org/>).

¹⁸ *IEEE Standards Dictionary Online* is available at: <http://dictionary.ieee.org>. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.

DL	Description Logic
DNF	Disjunctive Normal Form
FOL	First Order Logic
IRI	International Resource Identifier
mgu	Most General Unifier
NAF	Negation as Failure
OWA	Open World Assumption
OWL 2	Web Ontology Language; standard by World Wide Web Consortium (W3C)
OWL 2 EL, QL, RL	Profiles of OWL 2
PCC	Policy Conformance Components
PR	Policy Reasoner
PTIME	Polynomial Time
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RIA	Role Inclusion Axiom
RIF	Rule Interchange Format
RIF-BLD	RIF Basic Logic Dialect
RIF-FLD	RIF Framework for Logic Dialect
RIF-PRD	RIF Production Rule Dialect
RIF-DSA	RIF Dynamic Spectrum Access Dialect based on RIF-FLD defined in this standard
SCM	Spectrum Consumption Modeling
SSR	System Strategy Reasoner
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations
XSD	XML Schema Definition

4. Policy Language syntax representation

CR implementations conforming to this standard store and process policies for Dynamic Spectrum Access in the form of XML documents comprising combinations of OWL 2 and RIF-DSA statements.

For conciseness, this standard uses the terser presentation styles endorsed by W3C.

4.1 Logical structure

As mentioned, this standard is built on concepts from OWL 2 RL and RIF-FLD. The next two major paragraphs show syntax presentations of the underlying OWL 2 RL and RIF-FLD syntactical definitions.

An OWL 2 ontology is a formal description of a domain of interest. OWL 2 ontologies consist of the following three different syntactic categories:

- a) Entities, such as classes, properties, and individuals, are identified by IRIs. They form the primitive terms of an ontology and constitute the basic elements of an ontology. For example, a class `Nuvio:UnitOfMeasure` can be used to represent the set of all unit types that are used in relevant measurements. Similarly, the object property `Nuvio:hasUnitOfMeasure` can be used to represent the value–unit relationship. Finally, the individual `Nuvio:Watt` can be used to represent a particular individual unit called a “Watt.”
- b) Expressions represent complex notions in the domain being described. For example, a class expression describes a set of individuals in terms of the restrictions on the individuals’ characteristics, for example, `scmClsTypeAntennaHeight` is defined as being the intersection of classes equivalent to `(hasScmObjELEMheight exactly 1 scmClsELEMheight)` and `(hasScmObjELEMreference exactly 1 scmClsELEMreference)`.
- c) Axioms are statements that are asserted to be true in the domain being described. For example, using a subclass axiom, one can state that the class `Nuvio:PowerUnit` is a subclass of the class `Nuvio:UnitOfMeasure`.

These three syntactic categories are used to express the *logical* part of OWL 2 ontologies—that is, they are interpreted under a precisely defined semantics that allows useful inferences to be drawn. For example, if an individual `scm:antennaHeight-d1e1107` is assigned certain object properties like *reference* and *height*, then from the OWL 2 semantics, one can derive that `scm:antennaHeight-d1e1107` is an instance of `scm:scmClsTypeAntennaHeight`.

4.1.1 OWL 2 syntax—General definitions and OWL 2 constructs

Several syntax notational styles are being supported to define an OWL 2 ontology.

An OWL 2 implementation conforming to this standard shall implement the RDF/XML syntax as specified in this subclause. This syntax uses RDF subject–predicate–object (spo) triples to describe ontological entities. These RDF spo-triples are transformed into XML elements as defined in a further W3C standard to obtain a serialized text document. Ontology descriptions rendered via RDF/XML tend to be rather lengthy and the following terser syntactical styles are available:

- Manchester syntax is used by the Protégé OWL editor originating from Stanford University.
- Functional syntax is used for documentation in W3C standards.
- OWL 2 can also be serialized directly into XML without using RDF spo-triples.
- Turtle style syntax is favored by a number of authors and is used in tools and query languages like SPARQL and SHACL.

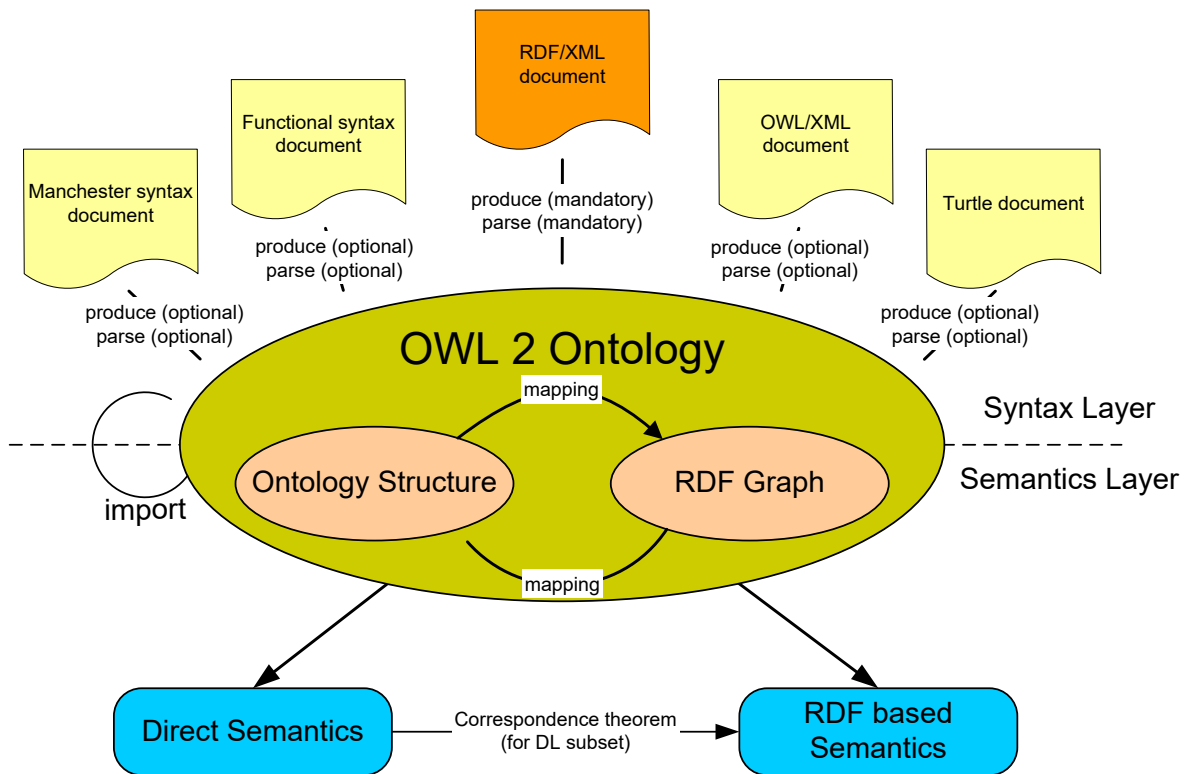


Figure 1—OWL 2 environmental embedding

Following the W3C example, this standard uses the functional-style syntax description for OWL 2. The latest complete and normative functional-style description of the full OWL 2 language may be retrieved from the W3C website.¹⁹

The core part of W3C OWL 2 specification²⁰—called the *structural specification*—is independent of the concrete exchange syntaxes for OWL 2 ontologies. The structural specification describes the conceptual structure of OWL 2 ontologies and thus provides a normative abstract representation for all (normative and nonnormative) syntaxes of OWL 2. This allows for a clear separation of the essential features of the language from issues related to any particular syntax. Furthermore, such a structural specification of OWL 2 provides the foundation for the implementation of OWL 2 tools such as APIs and reasoners.

Each OWL 2 ontology represented as an instance of this conceptual structure can be converted into an RDF graph; conversely, *most* OWL 2 ontologies represented as RDF graphs can be converted into the conceptual structure defined in that document.

Current general OWL 2 Syntax definitions can be found in D.1.

Current definitions of OWL 2 Constructs can be found in D.2.2.

¹⁹ See Annex C.

²⁰ See *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax* [B8].

4.1.1.1 OWL 2 RL syntax presentation and syntactical constraints

As mentioned, the OWL 2 RL is a profile of OWL 2. An OWL 2 *profile* (commonly called a *fragment* or a *sublanguage* in computational logic) comprises a subset version of OWL 2 depending on its intended use; for example, it may trade some expressive power for the efficiency of reasoning as in OWL 2 RL.

As such, the OWL 2 RL profile imposes restrictions on full OWL 2 in the way in which constructs are used, thereby facilitating the implementation of reasoning systems using rule-based reasoning engines, while providing desirable computational guarantees. These restrictions are designed to avoid the need to infer the existence of individuals not explicitly present in the knowledge base (for compatibility with rule-based systems) and therefore avoid the otherwise uprising possibility of nondeterministic reasoning. This is achieved by restricting the use of constructs, including variables, to certain syntactic positions.

Being a fragment of OWL 2, each OWL 2 RL ontology must naturally as well satisfy all global restrictions on axioms defined in the appropriate section of the OWL 2 structural specification.

However, minimum cardinality clauses like the requirement that a class contain only range data from a certain object property, if at least a certain minimum of individuals are mapped by that property, are not included in OWL 2 RL as there is no proper mechanism to implement such a minimum cardinality clause via a rule language. As these cardinality clauses are generated, for example, through XSLT styles sheets, the DSA policy language needs to include these.

4.1.1.2 OWL 2 RL entities

Entities are the fundamental building blocks of OWL 2 ontologies, and they define the vocabulary, that is, the named terms, of an ontology.

4.1.1.3 Classes

Classes can be understood as sets of individuals.

4.1.1.4 Datatypes

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions.

4.1.1.5 Object properties

Object properties connect pairs of individuals.

4.1.1.6 Data properties

Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called *attributes*. The following data properties with the IRIs *owl:topDataProperty* and *owl:bottomDataProperty* are available in OWL 2 as built-in data properties with a predefined semantics:

- The data property with IRI *owl:topDataProperty* connects all possible individuals with all literals.
- The data property with IRI *owl:bottomDataProperty* does not connect any individual with a literal.

4.1.1.7 Annotation properties

Annotation properties can be used to provide an annotation for an ontology, axiom, or an IRI.

4.1.1.8 Individuals

Individuals in the OWL 2 syntax represent actual objects from the domain. There are two types of individuals in the syntax of OWL 2. *Named individuals* are given an explicit name that can be used in any ontology to refer to the same object. *Anonymous individuals* do not have a global name and are thus local to the ontology in which they are contained.

4.1.1.9 Literals

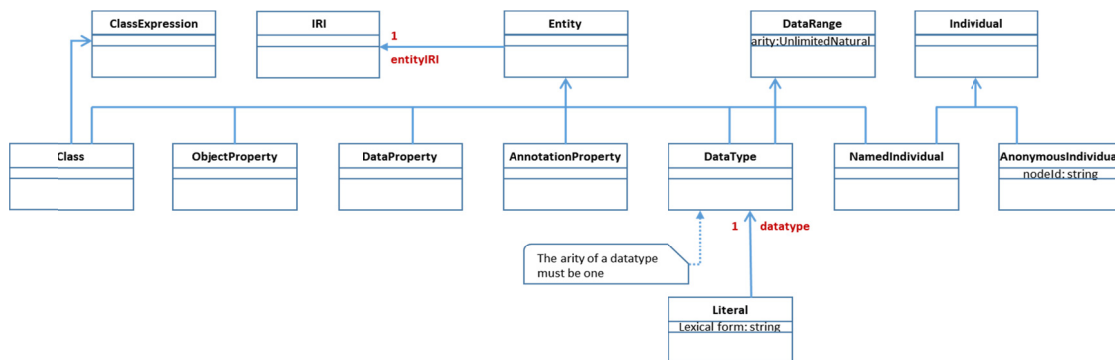
Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals [RDF Concepts] and can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string, and a datatype; the datatypes supported in OWL 2 are described in more detail in 4.1.1.15 and 4.1.3.1.5. A literal consisting of a lexical form “abc” and a datatype identified by the IRI *datatypeIRI* is written as “abc”^^*datatypeIRI*.

4.1.1.10 Entity restrictions in OWL RL

Entities are defined in OWL 2 RL in the same way as in the OWL 2 structural specification with the following distinctiveness:

OWL 2 RL versus OWL 2	
Supported	Not supported
owl:Nothing	owl:topObjectProperty
owl:Thing	owl:bottomObjectProperty
Supported but restricted in use	owl:topDataProperty
	owl:bottomDataProperty

4.1.1.11 Entities, literals, and anonymous individuals in OWL 2



Source. Figure adapted from *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax* [B8].

As outlined in Clause 1, when combining rule systems with ontologies, care must be taken to avoid the need to infer the existence of individuals that are not explicitly present in the knowledge base (as a pure ontology reasoner might do) as this could easily lead to undecidability of the entire rule and ontology combination. This care must be taken at the syntactical level as once a rule/ontology combined document/program is submitted to a reasoner, the reasoner has no way of detecting this and may enter an indefinite and/or infinite loop. In any case, the result of any reasoner output in this scenario is invalid.

Decidability is maintained by restricting the use of constructs to certain syntactic positions. For example, in SubClassOf axioms, the constructs in the subclass and superclass expressions must follow the usage patterns shown in the following table.

Supported subclass expressions	
a class other than <i>owl:Thing</i>	
an enumeration of individuals	ObjectOneOf
intersection of class expressions	ObjectIntersectionOf
union of class expressions	ObjectUnionOf
existential quantification to a class expression	ObjectSomeValuesFrom
existential quantification to a data range	DataSomeValuesFrom
existential quantification to an individual	ObjectHasValue
existential quantification to a literal	DataHasValue
Supported superclass expressions	
a class other than <i>owl:Thing</i>	
intersection of classes	ObjectIntersectionOf
negation	ObjectComplementOf
universal quantification to a class expression	ObjectAllValuesFrom
existential quantification to an individual	ObjectHasValue
at-most 0/1 cardinality restriction to a class expression	ObjectMaxCardinality 0/1
universal quantification to a data range	DataAllValuesFrom
existential quantification to a literal	DataHasValue
at-most 0/1 cardinality restriction to a data range	DataMaxCardinality 0/1

For example, the following excerpt from the SCM XSD would result in an unsupported construct via the automatic XSLT stylesheet transformation as clause ‘minOccurs=“n”’ would be transformed into RDF predicate owl:minQualifiedCardinality, which is not supported.

Likewise would any attribute maxOccurs=“n” with $n \geq 2$ be transformed into RDF predicate owl:maxQualifiedCardinality that is unsupported for values n other 0 or 1.

```
<complexType name="SCMPolygonType">
  <sequence>
    <element name="side" type="tns:SideType" minOccurs="3"21
maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

XSLT OWL 2 RDF transformation into:

```
<owl:Class rdf:about="http://ece.neu.edu.crf/SDROntology.owl#scmClsTypeSCMPolygon">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty
rdf:resource="http://ece.neu.edu.crf/SDROntology.owl#hasScmObjTYPEside"/>
          <owl:onClass
```

²¹ Please note that number restrictions ($\leq nR.C$) and ($\geq nR.C$) are restricted to simple roles. Intuitively, these are (possibly inverse) roles that are not implied by the composition of other roles. The reason for this restriction is that, without it, satisfiability of SHIQ-concepts is undecidable (Horrocks et al. [B3]) even for a logic without inverse roles and with only unqualifying number restrictions (these are number restrictions of the form ($\leq nR.T$) and ($\geq nR.T$)).

```

rdf:resource="http://ece.neu.edu.crf/SDROntology.owl#scmClsTypeSide"/>
                                <owl:minQualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">3</owl:minQualifiedCardinality>

                                </owl:Restriction>

                                </owl:intersectionOf>

                                </owl:Class>

                                </owl:equivalentClass>

                                <rdfs:subClassOf rdf:resource="http://ece.neu.edu.crf/SDROntology.owl#scmTypes"/>

</owl:Class>

```

4.1.1.12 OWL 2 RL class expressions

There are three types of class expressions in OWL 2 RL, as follows:

- subClassExpression** production defines the class expressions that can occur as subclass expressions in **SubClassOf** axioms.
- superClassExpression** production defines the class expressions that can occur as superclass expressions in **SubClassOf** axioms.
- equivClassExpressions** production defines the classes that can occur in **EquivalentClasses** axioms.

A subclass axiom **SubClassOf**(CE_1 CE_2) states that the class expression CE_1 is a subclass of the class expression CE_2 . Roughly speaking, this states that CE_1 is more specific than CE_2 . Subclass axioms are a fundamental type of axioms in OWL 2 and can be used to construct a class hierarchy. Other kinds of class expression axiom can be seen as syntactic shortcuts for one or more subclass axioms.

4.1.1.13 OWL 2 RL syntax class expressions

zeroOrOne := '0' | '1'

subClassExpression :=
Class other than *owl:Thing* |
subObjectIntersectionOf |
subObjectUnionOf |
ObjectOneOf |
subObjectSomeValuesFrom |
ObjectHasValue |
DataSomeValuesFrom |
DataHasValue

subObjectIntersectionOf :=
'ObjectIntersectionOf' '(' **subClassExpression** **subClassExpression** { **subClassExpression** } ')'

subObjectUnionOf :=
'ObjectUnionOf' '(' **subClassExpression** **subClassExpression** { **subClassExpression** } ')'

subObjectSomeValuesFrom :=

'ObjectSomeValuesFrom' '(' **ObjectPropertyExpression** **subClassExpression** ')' |
'ObjectSomeValuesFrom' '(' **ObjectPropertyExpression** *owl:Thing* ')'

superClassExpression :=

Class other than *owl:Thing* |

superObjectIntersectionOf |

superObjectComplementOf |

superObjectAllValuesFrom |

ObjectHasValue |

superObjectMaxCardinality |

DataAllValuesFrom |

DataHasValue |

superDataMaxCardinality

superObjectIntersectionOf :=

'ObjectIntersectionOf' '(' **superClassExpression** **superClassExpression** { **superClassExpression** } ')'

superObjectComplementOf :=

'ObjectComplementOf' '(' **subClassExpression** ')'

superObjectAllValuesFrom :=

'ObjectAllValuesFrom' '(' **ObjectPropertyExpression** **superClassExpression** ')'

superObjectMaxCardinality :=

'ObjectMaxCardinality' '(' **zeroOrOne** **ObjectPropertyExpression** [**subClassExpression**] ')'

'ObjectMaxCardinality' '(' **zeroOrOne** **ObjectPropertyExpression** *owl:Thing* ')'

superDataMaxCardinality :=

'DataMaxCardinality' '(' **zeroOrOne** **DataPropertyExpression** [**DataRange**] ')'

equivClassExpression :=

Class other than *owl:Thing* |

equivObjectIntersectionOf |

ObjectHasValue |

DataHasValue

equivObjectIntersectionOf :=

'ObjectIntersectionOf' '(' **equivClassExpression** **equivClassExpression** { **equivClassExpression** } ')'

4.1.1.14 OWL 2 RL data ranges

A data range expression is restricted in OWL 2 RL to the predefined datatypes admitted in OWL 2 RL and the intersection of data ranges.

4.1.1.15 OWL 2 RL datatypes

Datatypes supported by OWL 2 RL are as follows:

Supported datatypes in OWL 2 RL			
rdf:PlainLiteral	rdf:XMLLiteral	rdfs:Literal	xsd:decimal
xsd:integer	xsd:nonNegativeInteger	xsd:nonPositiveInteger	xsd:positiveInteger
xsd:negativeInteger	xsd:long	xsd:int	xsd:short
xsd:byte	xsd:unsignedLong	xsd:unsignedInt	xsd:unsignedShort
xsd:unsignedByte	xsd:float	xsd:double	xsd:string
xsd:normalizedString	xsd:token	xsd:language	xsd:Name

xsd:NCName	xsd:NMTOKEN	xsd:boolean	xsd:hexBinary
xsd:base64Binary	xsd:anyURI	xsd:dateTime	xsd:dateTimeStamp
Unsupported datatypes in OWL 2 RL			
owl:real	owl:rational		

The supported datatypes are also listed in Annex D.²²

4.1.1.16 OWL 2 RL syntax data ranges

DataRange := Datatype | DataIntersectionOf

DataIntersectionOf := 'DataIntersectionOf' '(' DataRange DataRange {DataRange } ')'

4.1.1.17 OWL 2 RL object property expressions

Property expressions in OWL 2 RL are identical to the property expressions in the OWL 2 structural specification.²³ For convenience, these are also listed here in functional-style syntax.

As the policy language defined in this standard is a combination of OWL 2 constructs and RIF rules, and rules take precedence in notation, the object property expressions are used as predicate constructs in the policy language.

Example:

OWL 2 functional-style syntax (from Cognitive Radio Ontology 2)	Comment
SubObjectPropertyOf(CRO2:hasSampleRate Nuvio:hasObjectProperty)	Having a CR sample rate implies having a Nuvio Object
ObjectPropertyAssertion(CRO2:hasSampleRate CRO2:cr-id1 CRO2:smplRate-id1)	Assert that cr-id1 of CRO2 has smplRate-id1
Implies the fact ObjectPropertyAssertion(Nuvio:hasObjectProperty CRO2:cr-id1 CRO2:smplRate-id1)	
Policy Language presentation syntax	
Forall ?x ?y (?x[Nuvio:hasObjectProperty → ?y] :- ?x[CRO2:hasSampleRate → ?y])	?x having ?y as value for predicate CRO2:hasSampleRate in the condition part implies ?x having ?y as value for predicate Nuvio:hasObjectProperty in the conclusion, that is, head
CRO2:cr-id1[CRO2:hasSampleRate → CRO2:smplRate-id1]	Assert fact
Above rule is therefore “true” in condition part for ?x=CRO2:cr-id1 and ?y=smplRate-id1, hence, imply head, that is, assert: CRO2:cr-id1 [Nuvio:hasObjectProperty → smplRate-id1]	

List of OWL 2 RL object properties in BNR form and Functional-Style Syntax:

ObjectPropertyAxiom :=

SubObjectPropertyOf	EquivalentObjectProperties	
DisjointObjectProperties	InverseObjectProperties	
ObjectPropertyDomain	ObjectPropertyRange	

²² See D.2.3.1 specifically.

²³ See OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax [B8].

FunctionalObjectProperty		InverseFunctionalObjectProperty	
ReflexiveObjectProperty		IrreflexiveObjectProperty	
SymmetricObjectProperty		AsymmetricObjectProperty	
TransitiveObjectProperty			

SubObjectPropertyOf :=
'SubObjectPropertyOf' '(' axiomAnnotations subObjectPropertyExpression
superObjectPropertyExpression ')'

subObjectPropertyExpression :=
ObjectPropertyExpression | propertyExpressionChain

propertyExpressionChain :=
'ObjectPropertyChain' '(' ObjectPropertyExpression ObjectPropertyExpression
{ ObjectPropertyExpression } ')'

superObjectPropertyExpression := ObjectPropertyExpression

EquivalentObjectProperties :=
'EquivalentObjectProperties' '(' axiomAnnotations ObjectPropertyExpression

ObjectPropertyExpression { ObjectPropertyExpression } ')'

DisjointObjectProperties :=
'DisjointObjectProperties' '(' axiomAnnotations ObjectPropertyExpression ObjectPropertyExpression
{ ObjectPropertyExpression } ')'

InverseObjectProperties :=
'InverseObjectProperties' '(' axiomAnnotations ObjectPropertyExpression ObjectPropertyExpression
)'

ObjectPropertyDomain :=
'ObjectPropertyDomain' '(' axiomAnnotations ObjectPropertyExpression ClassExpression ')'

ObjectPropertyRange :=
'ObjectPropertyRange' '(' axiomAnnotations ObjectPropertyExpression ClassExpression ')'

FunctionalObjectProperty :=
'FunctionalObjectProperty' '(' axiomAnnotations ObjectPropertyExpression ')'

InverseFunctionalObjectProperty :=
'InverseFunctionalObjectProperty' '(' axiomAnnotations ObjectPropertyExpression ')'

ReflexiveObjectProperty :=
'ReflexiveObjectProperty' '(' axiomAnnotations ObjectPropertyExpression ')'

IrreflexiveObjectProperty :=
'IrreflexiveObjectProperty' '(' axiomAnnotations ObjectPropertyExpression ')'

SymmetricObjectProperty :=
'SymmetricObjectProperty' '(' axiomAnnotations ObjectPropertyExpression ')'

AsymmetricObjectProperty :=
'AsymmetricObjectProperty' '(' axiomAnnotations ObjectPropertyExpression ')'

TransitiveObjectProperty :=
'TransitiveObjectProperty' '(' **axiomAnnotations** **ObjectPropertyExpression** ')'

4.1.1.18 OWL 2 RL Axioms

Axioms are those statements that are asserted as always being true, and all true statements in an ontology must be derived from one or more axioms. In other words, the true statements in an ontology are formed as syntactically correct derivations from one or more below listed axioms.

OWL 2 RL redefines all axioms of the OWL 2 structural specification that refer to class expressions. In particular, it restricts various class axioms to use the appropriate form of class expressions (i.e., one of `subClassExpression`, `superClassExpression`, or `equivClassExpression`), and it disallows the `DisjointUnion` axiom.

OWL 2 RL axioms about property expressions are as in the above structural specification, the only differences being that class expressions in property domain and range axioms are restricted to `superClassExpression`, and that the use of reflexive properties is disallowed.

OWL 2 RL restricts class expressions in positive assertions to `superClassExpression`. All other assertions are the same as in the above structural specification.

All other axioms in OWL 2 RL are defined as in the structural specification.²⁴

4.1.1.19 OWL 2 RL class axioms

ClassAxiom :=
SubClassOf | **EquivalentClasses** | **DisjointClasses**

SubClassOf :=
'SubClassOf' '(' **axiomAnnotations** **subClassExpression** **superClassExpression** ')'

EquivalentClasses :=
'EquivalentClasses' '(' **axiomAnnotations** **equivClassExpression** **equivClassExpression**
{ **equivClassExpression** } ')'

DisjointClasses :=
'DisjointClasses' '(' **axiomAnnotations** **subClassExpression** **subClassExpression**
{ **subClassExpression** } ')'

4.1.1.19.1 OWL 2 RL object property axioms

ObjectPropertyDomain :=
'ObjectPropertyDomain' '(' **axiomAnnotations** **ObjectPropertyExpression** **superClassExpression** ')'

ObjectPropertyRange :=
'ObjectPropertyRange' '(' **axiomAnnotations** **ObjectPropertyExpression** **superClassExpression** ')'

DataPropertyDomain :=
'DataPropertyDomain' '(' **axiomAnnotations** **DataPropertyExpression** **superClassExpression** ')'

ObjectPropertyAxiom :=
SubObjectPropertyOf | **EquivalentObjectProperties** | **DisjointObjectProperties**

²⁴ See D.2.3.4.

InverseObjectProperties	ObjectPropertyDomain	ObjectPropertyRange
FunctionalObjectProperty	InverseFunctionalObjectProperty	IrreflexiveObjectProperty
SymmetricObjectProperty	AsymmetricObjectProperty	TransitiveObjectProperty

4.1.1.19.2 OWL 2 RL class assertion axiom

ClassAssertion :=
'ClassAssertion' '(' axiomAnnotations superClassExpression Individual ')'

4.1.1.19.3 OWL 2 RL key class axiom

HasKey :=
'HasKey' '(' axiomAnnotations subClassExpression '(' { ObjectPropertyExpression } ')' '('
{ DataPropertyExpression } ')')'

4.1.1.19.4 OWL 2 RL property chains²⁵

OWL 2 RL allows the definition of role chains, such as follows:

- $\text{sysRxModel}(x,z) = \text{hasScmObjTYPErxModel}(x,y) \circ \text{hasScmObjTYPEunderlayMask}(y,z)$
- $\text{sysTxModel}(x,z) = \text{hasScmObjTYPEtxModel}(x,y) \circ \text{hasScmObjTYPEunderlayMask}(y,z)$

In this example, x is connected to z by role (aka ObjectProperty) sysRxModel iff:

- a) x is connected to some y via ObjectProperty $\text{hasScmObjTYPErxModel}$
- b) y is connected to z via ObjectProperty $\text{hasScmObjTYPEunderlayMask}(y,z)$

Hence, if:

- systemModel-1 is an individual of class $\text{scmClsTypeSystemModel}$
- rxModel-2 is an individual of class scmClsTypeRxModel
- underlayMask-3 is an individual of class $\text{scmClsTypeUnderlayMask}$
- rxModel-2 is part of systemModel-1 's RxModel structure
- underlayMask-3 is part of rxModel-1 's underlayMask structure

Then:

- systemModel-1 is connected to underlayMask-3 via ObjectProperty sysRxModel

Analogously for ObjectProperty sysTxModel :

ObjectProperty chains are defined as $\text{SubObjectPropertyOf}(\text{ObjectPropertyChain}(\text{ObjProp1 ObjProp2} \dots \text{ObjPropN}); \text{ObjPropChainName})$ in OWL 2 RL Functional Syntax.

While ObjectProperty chains are very useful tools in ontology definitions, the unrestricted use of these leads to undecidability even without the addition of any rules.

²⁵ For effective computability of property chains, a.k.a. *RLAs*, see G.1.

To maintain decidability, ObjectProperties (roles) are first classified as simple or as nonsimple, and then a strict partial order is attempted to be defined on the nonsimple roles. If there is no strict partial order definable on the nonsimple roles, the entire ontology MUST be rejected as it may cause undecidability.

The procedure for defining simple, nonsimple roles and a strict partial order on the nonsimple roles is as follows:

Generally, a role chain axiom (or RIA) is a statement of the form $r_1 \circ \dots \circ r_n \sqsubseteq r$,²⁶ where $r_1, \dots, r_n, r$ are roles (a.k.a. “Object Properties”).

Given a role hierarchy, it is useful to distinguish those roles that can be created by role chains of length greater than one from those that cannot. Consequently, we define nonsimple roles as follows:

- S1 Every role r occurring in a RIA $r_1 \circ \dots \circ r_n \sqsubseteq r$ where $n > 1$ is nonsimple.
- S2 Every role r occurring in a simple role inclusion $s \sqsubseteq r$ with a non-simple s is itself nonsimple.
- S3 If r is nonsimple then so is $\text{Inv}(r)$, also denoted as r^- .
- S4 No other role is nonsimple.

Let R^n denote the set of all nonsimple roles of a role hierarchy and call all the other roles “simple” denoted by $R^s = R \setminus R^n$.

For example, consider the following role hierarchy:

- a) $\text{motherOf} \sqsubseteq \text{parentOf}$
- b) $\text{parentOf} \sqsubseteq \text{ancestorOf}$
- c) $\text{ancestorOf} \circ \text{ancestorOf} \sqsubseteq \text{ancestorOf}$
- d) $\text{ancestorOf} \sqsubseteq \text{Inv}(\text{descendantOf})$

Then by S1 and due to item 3), ancestorOf is nonsimple. Therefore, $\text{Inv}(\text{descendantOf})$ is nonsimple via item 4) and S2.

The above follows via S3 in that also $\text{Inv}(\text{ancestorOf})$ and descendantOf must be nonsimple. Finally, S4 ensures that motherOf , $\text{Inv}(\text{motherOf})$, parentOf , and $\text{Inv}(\text{parentOf})$ are simple.

Similar constructs occur naturally and frequently in a policy language based on ontological design.

To ensure decidability of the ensuing logic, arbitrary role hierarchies cannot be allowed but have to be restricted to those that have the property of being “regular.” Formally, a role hierarchy is regular if there is a *strict* partial order “ $<$ ” on the nonsimple roles R^n , such as follows:

- $S < R$ if and only if $\text{Inv}(S) < R$
- Every RIA is of one of the forms R1, R2, R3, R4, R5:

- | | | |
|----|-------------------------------------|-----------------|
| R1 | $r \circ r$ | $\sqsubseteq r$ |
| R2 | $\text{Inv}(r)$ | $\sqsubseteq r$ |
| R3 | $s_1 \circ \dots \circ s_n$ | $\sqsubseteq r$ |
| R4 | $r \circ s_1 \circ \dots \circ s_n$ | $\sqsubseteq r$ |
| R5 | $s_1 \circ \dots \circ s_n \circ r$ | $\sqsubseteq r$ |

such that r is a (noninverse) role name r , and $s_i < r$ for $i = 1, \dots, n$ whenever s_i is nonsimple.

²⁶ symbol $\sqsubseteq$ denotes subsumption (inclusion) in Description Logic; that is, the left side is subsumed (included) in the right side.