



LarKC

The Large Knowledge Collider

a platform for large scale integrated reasoning and Web-search

FP7 – 215535

D1.1.4 Improved Knowledge Representation Formalism

Coordinator: Florian Fischer

With contributions from: Florian Fischer, Barry Bishop

Quality Assessor: Dunja Mladenic

Quality Controller: Reto Krummenacher

Document Identifier:	LarKC/2008/D1.1.4/V0.1
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	version 1.0
Date:	March 29, 2010
State:	final
Distribution:	public



EXECUTIVE SUMMARY

This report is an extension of the work in D1.1.3 toward a recommended minimal representation formalism for LarKC. The goal of D1.1.3 was to define a standard minimal representation language for LarKC reasoning components. The methodology employed was to examine existing formalisms and to select one of these if suitable. If no suitable formalism was found then desirable features of existing formalisms would be identified and these would be combined to create the definition of a new language for use in LarKC.

It was indeed found that no suitable standard language existed, since at the time OWL 2 RL had a slightly different specification which forbade the use of meta-modelling (to be explained in the following document). Therefore a new language was created from subsets of OWL and this language was called L2. However, when considering a second version of this language (the purpose of this deliverable), recent modifications to the emerging OWL 2 standard mean that OWL 2 RL could after all be considered for a suitable minimal representation language for LarKC, due to a change in policy regarding meta-modelling.

This deliverable explains these recent changes, how they are desirable (and even necessary for LarKC) and goes on to compare the complexity/scalability of the original L2 language and the latest OWL 2 RL specification. Furthermore, this document also outlines an architecture for the implementation of a conformant OWL 2 RL reasoner prototype.



DOCUMENT INFORMATION

IST Project Number	FP7 – 215535	Acronym	LarKC
Full Title	The Large Knowledge Collider: a platform for large scale integrated reasoning and Web-search		
Project URL	http://www.larkc.eu/		
Document URL			
EU Project Officer	Stefano Bertolo		

Deliverable	Number	1.1.4	Title	Improved Knowledge Representation Formalism
Work Package	Number	1	Title	Conceptual Framework & Evaluation

Date of Delivery	Contractual	M24	Actual	
Status	version 1.0		final	<input checked="" type="checkbox"/>
Nature	prototype <input type="checkbox"/> report <input checked="" type="checkbox"/> dissemination <input type="checkbox"/>			
Dissemination Level	public <input checked="" type="checkbox"/> consortium <input type="checkbox"/>			

Authors (Partner)	Florian Fischer (UIBK), Barry Bishop (UIBK)			
Resp. Author	Florian Fischer		E-mail	florian.fischer@sti2.at
	Partner	UIBK	Phone	+xx (xxx) xxx-xxxx

Abstract (for dissemination)	
Keywords	knowledge representation, ontology reasoning, Web scale reasoning

Version Log			
Issue Date	Rev No.	Author	Change
28 Sep 2009	1	Florian Fischer	Document creation.
29 Sep 2009	2	Florian Fischer	Introduction, Background
3 Oct 2009	3	Florian Fischer	Appendix, Language semantics
4 Nov 2009	4	Barry Bishop	Proof reading
29 Mar 2010	5	Christoph Fuchs	Incorporation of QA feedback



PROJECT CONSORTIUM INFORMATION
















Participant's name	Partner	Contact
Semantic Technology Institute Innsbruck, Universitaet Innsbruck	 	Prof. Dr. Dieter Fensel Semantic Technology Institute (STI), Universitaet Innsbruck, Innsbruck, Austria Email: dieter.fensel@sti-innsbruck.at
AstraZeneca AB		Bosse Andersson AstraZeneca Lund, Sweden Email: bo.h.andersson@astrazeneca.com
CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA		Emanuele Della Valle CEFRIEL - SOCIETA CONSORTILE A RE- SPONSABILITA LIMITATA Milano, Italy Email: emanuele.dellavalle@cefriel.it
CYCROP, RAZISKOVANJE IN EKSPERI- MENTALNI RAZVOJ D.O.O.		Michael Witbrock CYCROP, RAZISKOVANJE IN EKSPERIMEN- TALNI RAZVOJ D.O.O., Ljubljana, Slovenia Email: witbrock@cyc.com
Höchstleistungsrechenzentrum, Universitaet Stuttgart		Georgina Gallizo Höchstleistungsrechenzentrum, Universitaet Stuttgart Stuttgart, Germany Email : gallizo@hlrs.de
MAX-PLANCK GESELLSCHAFT ZUR FOERDERUNG DER WISSENSCHAFTEN E.V.		Dr. Lael Schooler, Max-Planck-Institut für Bildungsforschung Berlin, Germany Email: schooler@mpib-berlin.mpg.de
Ontotext AD		Atanas Kiryakov, Ontotext Lab, Sofia, Bulgaria Email: naso@ontotext.com
SALTLUX INC.		Kono Kim SALTLUX INC Seoul, Korea Email: kono@saltlux.com
SIEMENS AKTIENGESELLSCHAFT		Dr. Volker Tresp SIEMENS AKTIENGESELLSCHAFT Muenchen, Germany Email: volker.tresp@siemens.com
THE UNIVERSITY OF SHEFFIELD		Prof. Dr. Hamish Cunningham, THE UNIVERSITY OF SHEFFIELD Sheffield, UK Email: h.cunningham@dcs.shef.ac.uk
VRIJE UNIVERSITEIT AMSTERDAM		Prof. Dr. Frank van Harmelen, VRIJE UNIVERSITEIT AMSTERDAM Amsterdam, Netherlands Email: Frank.van.Harmelen@cs.vu.nl
THE INTERNATIONAL WIC INSTI- TUTE, BEIJING UNIVERSITY OF TECHNOLOGY		Prof. Dr. Ning Zhong, THE INTERNATIONAL WIC INSTITUTE Mabeshi, Japan Email: zhong@maebashi-it.ac.jp
INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER		Dr. Paul Brennan, INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER Lyon, France Email: brennan@iarc.fr
INFORMATION RETRIEVAL FACILITY		Dr. John Tait, Dr. Paul Brennan, INFORMATION RETRIEVAL FACILITY Vienna, Austria Email: john.tait@ir-facility.org



TABLE OF CONTENTS

1	INTRODUCTION	6
2	LARKC MINIMAL KR IN THE CONTEXT OF RECENT ADVANCES	7
2.1	Background	7
2.1.1	L2	7
2.1.2	OWL 2 RL	7
2.2	Comparison of L2 and OWL 2 RL	8
2.2.1	Language Features	9
2.2.2	Language Semantics	11
2.2.3	Data-type Support	12
2.2.4	Summary	12
3	CONCLUSION	14
	REFERENCES	14
A	ENTAILMENT RULES	16
A.1	OWL 2 RL	16
A.2	L2	23
B	AXIOMATIC TRIPLES	24



1. Introduction

This document describes further progress in regard to knowledge representation within the LarKC project. It is a conceptual continuation of the work documented in D1.1.3, which presented the “Initial knowledge representation formalism” called L2, and the efforts reflected in D1.1.2, which empirically analyzed the performance of inference algorithms using the language primitives and entailment rules identified for L2.

The main motivation for L2 was to ensure a basic level of interoperability between plug-ins by defining a common basic representation formalism. This formalism needs to support LarKC’s overall vision of an extensible framework and only forms an agreed-upon baseline for the different reasoning plug-ins.

Some initial requirements identified in D1.1.3 include: compatibility and alignment with existing Web standards such as RDF(S), a useful degree of expressivity while retaining tractability of inference, and ease of integration of existing ontologies and data-sets for the sake of interoperability. A core issue captured in these requirements is that because of the heterogeneity and noisy nature of “Web-data” we cannot necessarily assume any specific syntactic restrictions on the use of RDF to hold. Thus a sufficient degree of meta-modeling (i.e. treating concepts as instances and vice-versa) and layering on top of RDFS were identified as crucial. For this reason L2 was designed as an extension of RDFS in the style of OWL Full, albeit with limited expressivity in order to meet scalability requirements. In other words, L2 assigned a meaning to all RDF graphs, which includes the meaning defined through the RDFS semantics (albeit without axiomatic triples).

The requirements identified in D1.1.3 remain key issues and in light of these we identify recent promising results developed within the standardization efforts of the OWL 2 working group. In particular the OWL 2 RL profile [6] has emerged to be a suitable standard approach that meets LarKC’s requirements, and hence the rest of this document explains how adopting OWL 2 RL as the LarKC minimal knowledge representation formalism is a logical next step after L2, how this will be beneficial to the LarKC project, how L2 and OWL 2 RL differ, and the implications for components with respect to conformance.

The remainder of this document is structured as follows: In section 2.1 we briefly lay out the background, the work done so far, and the underlying motivation for both L2 and OWL 2 RL, while in section 2.2 we provide a more detailed comparison of the two languages. Finally, section 3 summarizes the main points of this document.



2. LarKC Minimal KR in the Context of Recent Advances

2.1 Background

2.1.1 L2

L2 was specified as an initial minimal knowledge representation formalism in order to meet the specific requirements of LarKC. It is layered on top of the RDF Schema semantics [1] although it explicitly allows the exclusion of the RDFS axiomatic triples in order to facilitate efficient implementations. The reasons for a full layering have been briefly explained in the introduction of this document, and stem from the underlying requirements of LarKC and properties of data found on the Web [10].

A further guideline for the design of L2 came from analyzing existing implementations that perform RDF based inference at a large scale. These systems usually implement inference support for a subset of OWL that corresponds very closely the pD^* fragment identified in [9]. This approach is a genuine semantic extension of RDFS as well (including meta-modeling), which does not assume extensionality and tries to retain useful parts of OWL, while removing some language elements and entailments for the sake of efficiency. Within the OWL 1 language stack, such a layering on top of RDFS is only retained by OWL-Full [5]. Unfortunately, the Description Logic based variants of OWL (OWL-Lite, OWL-DL) do not allow meta-modeling [4], rather they require a strict separation of identifiers for classes, individuals and properties. This imposes syntactic restrictions on RDF graphs which is not easy to reconcile with the requirements of LarKC.

In order to get a clearer understanding about the actual scalability behavior of L2 we performed an empirical analysis by implementing a suitable set of entailment rules for OWLIM and ran a series of performance tests. The results were reported in D1.1.2 and showed L2 to have scalability behavior comparable to more optimized rule-sets, e.g. those provided for pD^* , when performing forward-chaining.

L2 can be considered an OWL fragment (or “profile”) in the sense that it allows an efficient sub-set of inferences to be made. Therefore the OWL 2 functional-style syntax [7] can be used as a high-level syntax. However, any surface syntax with an appropriate mapping to the underlying RDF primitives can be used as the fundamental design aspects of the language are independent of the particular syntax employed. A concrete example for an alternative syntax is the Manchester Syntax¹ of OWL 2, which is a frame-based and more user-friendly, compact syntax.

In the following section we reflect on the outcome of the standardization work within the W3C OWL 2 working group and how this can be aligned with L2 and LarKC’s requirements.

2.1.2 OWL 2 RL

OWL 2 RL is one of the profiles [6] developed as part of the OWL 2 standardization process (a W3C Proposed Recommendation at the time of the writing of this document) and was inspired by Description Logic Programs(DLP) [3] as well as pD^* [9]. A profile is a fragment of OWL 2 that eliminates specific language elements (expressivity) for more efficient reasoning and in turn scalability. The design of OWL 2 RL

¹<http://www.w3.org/TR/owl2-manchester-syntax/>



deliberately targets applications that require scalable reasoning over large instance sets with a reasonable level of expressivity, i.e. applications that would often deal with high amounts of instance data, but would only use a schema that can be expressed in RDFS with some additional OWL constructs.

As such it is a syntactic OWL 2 subset whose semantics are defined via a partial axiomatization of the OWL 2 RDF-base semantics (OWL 2 Full), similar to the approach taken in [9] and also for L2. This means that the semantics are defined as a number of first-order implications that can be used as a direct starting point for implementations. Thus, OWL 2 RL is as well applicable to an implementation on common rule engines or extended RDBMS and typical reasoning tasks, such as consistency checking, subsumption checking, instance checking, or query answering, can be solved in polynomial time with respect to the size of the data.

However, as has been observed, it is in fact possible to apply the corresponding entailment rules to arbitrary RDF graphs as well as syntactically restricted restricted RDF graphs. Thus, an important feature of the OWL 2 RL profile is that it is basically the reconciliation of two different language variants (initially called OWL R Full and OWL R DL) proposed during the standardization process, which operated on a subset of OWL 2 Full and OWL 2 DL respectively. As an outcome of this, the final OWL 2 RL profile allows the application of the same set of inference rules to RDF graphs that are syntactically restricted (in order to comply with Description Logic limitations) as well as arbitrary RDF graphs. In the first case the inference process will yield conclusions that are a subset of those a Description Logic reasoner would supply (basically DLP), whereas in the second case the reasoner can derive *more* conclusions corresponding to an (incomplete w.r.t OWL 2 Full) subset of OWL 2 Full inferences. In this way it is possible to formally and clearly establish conformance for reasoners performing DLP-based inference as well as for systems that perform large inference by implementing pd*-style semantics [9] within a single profile. More importantly for LarkKC, OWL 2 RL's layering on top of RDFS allows for the processing of arbitrary RDF graphs (including those that use meta-modeling) in a tractable way while avoiding the undecidability of OWL Full. Technical details are presented in *Theorem PR1* in [6].

2.2 Comparison of L2 and OWL 2 RL

In this section we will provide a closer comparison between L2 and the OWL 2 RL profile. For this purpose we will briefly give an overview of the language features of each of them. For the sake of brevity we do not repeat the complete language definitions, but only list the basic features and vocabulary for each language. More in-depth information and definitions can be found in D1.1.3 and [6] along with [8]. In this overview we point out restrictions on the use of the vocabularies of L2 and OWL 2 RL. Following this overview we focus on actual language semantics and associated entailment rules. We do not list the complete rule-sets here, but only discuss notable differences. The rule-sets can be found in Appendix A and these are described in more depth in the respective specifications.



owl:sameAs owl:SymetricProperty owl:TransitiveProperty
 owl:inverseOf owl:equivalentClass owl:equivalentProperty

Table 2.1: Additional L2 vocabulary

2.2.1 Language Features

L2 is a semantic extension of RDF(S) with added vocabulary elements and their intended semantics, in the same fashion as OWL Full is defined. However L2 extends RDF(S) with a limited subset of the OWL Full vocabulary. The L2 vocabulary, as an extension of the RDF and RDFS vocabularies, adds the primitives listed in Table 2.1 *in addition* to those provided by RDF(S). Using this vocabulary along with the RDF(S) vocabulary allows us to express the following language primitives in L2:

- Class and property definitions
- Class and property hierarchies
- Domain and range restrictions
- Class, property, individual equivalence
- Transitive properties
- Symmetric properties
- Inverse properties

OWL 2 RL is an OWL 2 profile and thus any suitable OWL 2 syntax can be used and it inherits OWL 2's structural specification as well [7]. OWL 2 RL also inherits a large number of language constructs from OWL that are not directly concerned with language semantics (e.g. `owl:versionIRI`) and additional syntactic shortcuts. Furthermore, several of the language elements replace or refine RDF elements, e.g. `owl:ObjectProperty` is a subclass of `rdf:Property`. The complete list of language elements is shown in Table 2.2 on page 10. So, OWL 2 RL is very similar in spirit to L2, but adds additional language constructs. The OWL 2 RL language definition is also more complex in the sense that it not only defines the set of supported constructs, it also restricts the places in which these additional constructs can be used (see Table 2.3 on page 11). L2 does *not* have such restrictions in place and instead focuses on a more basic vocabulary that has no restrictions on where it can be used.

In addition to the language primitives supported by L2, the following constructs are supported in OWL 2 RL, although with specific syntactic restrictions placed on their use.

- Intersection of classes
- Union of classes
- Enumeration of individuals
- Existential quantification
- Universal quantification
- Negation
- Cardinality restrictions

The syntactic restrictions are intended to avoid two potential sources of intractability, namely the need to infer the existence of unnamed individuals (basically blank nodes in the consequent of a rule) and the need to examine different branches of the search space in a nondeterministic fashion. The restrictions imposed in regard to certain language constructs are essentially the same as for DLP [3] – restrictions on class constructors in order to ensure that it is possible to map them to corresponding



```

owl:AllDifferent owl:AllDisjointClasses owl:AllDisjointProperties
owl:allValuesFrom owl:annotatedProperty owl:annotatedSource
owl:annotatedTarget owl:Annotation owl:AnnotationProperty
owl:assertionProperty owl:AsymmetricProperty owl:Axiom
owl:backwardCompatibleWith owl:bottomDataProperty owl:cardinality owl:Class
owl:complementOf owl:DataRange owl:datatypeComplementOf
owl:DatatypeProperty owl:deprecated owl:DeprecatedClass
owl:DeprecatedProperty owl:differentFrom owl:disjointWith owl:distinctMembers
owl:equivalentClass owl:equivalentProperty owl:FunctionalProperty owl:hasKey
owl:hasSelf owl:hasValue owl:imports owl:incompatibleWith owl:intersectionOf
owl:InverseFunctionalProperty owl:inverseOf owl:IrreflexiveProperty
owl:maxCardinality owl:maxQualifiedCardinality owl:members
owl:minCardinality owl:minQualifiedCardinality owl:NamedIndividual
owl:NegativePropertyAssertion owl:Nothing owl:ObjectProperty owl:onClass
owl:onDataRange owl:onDatatype owl:oneOf owl:onProperty owl:onProperties
owl:Ontology owl:OntologyProperty owl:priorVersion owl:propertyChainAxiom
owl:propertyDisjointWith owl:qualifiedCardinality owl:Restriction owl:sameAs
owl:someValuesFrom owl:sourceIndividual owl:SymmetricProperty
owl:targetIndividual owl:targetValue owl:Thing owl:TransitiveProperty
owl:unionOf owl:versionInfo owl:versionIRI owl:withRestrictions

```

Table 2.2: OWL 2 RL vocabulary

Horn-rules and thus make sure that it is feasible to implement suitable reasoners using rule-engines while still retaining desirable performance.

Examples for these restrictions are disjunction and conjunction. A conjunction in a class expression can easily be rewritten into a single first-order implication or a pair of implications, independent of the side of a subclass axioms it occurs in:

$$C_1 \sqcap C_2 \sqsubseteq D \equiv D(x) \leftarrow C_1(x) \wedge C_2(x)$$

If a conjunction occurs on the right-hand-side of a subclass axiom it becomes an implication with a conjunction in the head, which could in turn be split into two independent first-order rules:

$$\begin{aligned}
C \sqsubseteq D_1 \sqcap D_2 &\equiv D_1(x) \wedge D_2(x) \leftarrow C(x) \\
&D_1(x) \leftarrow C(x) \\
&D_2(x) \leftarrow C(x)
\end{aligned}$$

In the case of a disjunction of two classes, it becomes relevant on which side of the subclass expression it occurs. If the disjunction is in on the left-hand-side of the subclass expression, the result is simply a disjunction in the body of the resulting rule, which could simply be split again.

$$\begin{aligned}
C_1 \sqcup C_2 \sqsubseteq D &\equiv D(x) \leftarrow C_1(x) \vee C_2(x) \\
&D(x) \leftarrow C_1(x) \\
&D(x) \leftarrow C_2(x)
\end{aligned}$$

However if a disjunction is used on the right-hand-side of a subclass expression this essentially results in a disjunction in the head of the corresponding first-order rule,



Subclass Expressions
Classes other than <code>owl:Thing</code> An enumeration of individuals The intersection of class expressions The union of class expressions Existential quantification to a class expression, data range, individual, or literal
Superclass Expressions
Classes other than <code>owl:Thing</code> The intersection of class expressions Negation Universal quantification to a class expression or a data range Existential quantification to an individual or literal Maximum of 0/1 cardinality restrictions to a class expression or data range

Table 2.3: Syntactic restrictions on class expressions of OWL 2 RL

which cannot be handled in an efficient way using a common rule engine [2]:

$$C \sqsubseteq D_1 \sqcup D_2 \equiv D_1(x) \vee D_2(x) \leftarrow C(x)$$

As in the above example, syntactic restrictions are achieved by restricting the use of constructs to certain syntactic positions. In general class expressions can only be composed according to the overview in Table 2.3 on page 11. Property expressions in OWL 2 RL do not have such restrictions and are the same as the property expressions in [7]. In terms of axioms, OWL 2 RL restricts class axioms to only include class expressions under the restrictions outlined above and property axioms to only include legal superclass expressions in domain and range restrictions. Furthermore the use of reflexive properties is disallowed, as in L2.

In summary OWL 2 RL provides several additional language elements and auxiliary tools (e.g. annotations, high-level syntax) to users. However, its expressivity comes at the cost of additional restrictions on language elements.

2.2.2 Language Semantics

The semantics for both L2 and OWL 2 RL are defined as a set of entailment rules operating on RDF triples. OWL 2 RL is a syntactic fragment of OWL 2 and is defined by means of a partial axiomatization of the OWL 2 RDF-based semantics [8] by universally quantified first-order implications that can be directly understood as a set of entailment rules and simply applied to an arbitrary RDF graph, just as in the case of L2. OWL 2 RL establishes conformance to language semantics in a slightly more complicated fashion (as already pointed out in Section 2.1 by admitting both DLP based inference (using a DL reasoner), which operates on restricted RDF graphs, as well as rule engines which apply the OWL 2 RL entailment rules to arbitrary graphs – including those that make use of features that go beyond the capabilities of a pure DL-based reasoner (i.e. by using meta-modeling). The principal restriction is that IRIs should only be used for one type of entity (a class, an individual, etc) with several additional ones defined in [6]. L2 allows meta-modeling and can simply be regarded as an additional layer on RDFS that includes some tractable parts of OWL Full.



Both languages agree on the decision to skip axiomatic triples that need to be satisfied by every RDF and RDFS interpretation (see Appendix B) by default since they i) are usually not relevant for practical query answering, ii) can cause potential performance problems due to a huge bloat when forward-chaining. However, both OWL 2 RL and L2 explicitly allow axiomatic triples to be included if desired.

In order to define its semantics OWL 2 RL requires more entailment rules than L2, in total 74 (see Appendix A). However the rules in the definition of OWL 2 RL are by no means the minimal set of rules that could be used and practical implementations are expected to a more efficient minimal set. Furthermore around 50 of these rules overlap with the RDFS entailment rules or simply subsume the RDFS semantics, e.g. the definition of class equivalence is already available implicitly in RDFS. Thus there are only a small number of rules that actually add further semantics to the language and these mostly relate to additional semantic constructs for class and property axioms and schema vocabulary (see Tables A.3, A.5, and A.5). Several of the rules overlap with L2, in particular those concerned with finer characterization of properties (as symmetric, inverse, transitive, etc). Rules that clearly go beyond L2 are concerned with the intersection and union of classes, property chains (defining the composition of two or more properties as being the sub property of another one), and cardinalities (or keys).

In summary, the additional entailment rules of OWL 2 RL can be considered basic extensions of L2. However, it needs to be noted that these additional rules are of a slightly higher complexity when implemented naively and furthermore that a special propositional symbol (or similar functionality) needs to be supported by the underlying rule engine in order to indicate a contradiction – this is required to indicate that the initial RDF graph was inconsistent.

2.2.3 Data-type Support

A point in which OWL 2 RL clearly goes beyond the possibilities provided by L2 is data-type support: L2 essentially only includes the RDF semantics for data-types, which imposes minimal conditions on a data-types and only includes a single built-in data-type (`rdf:XMLLiteral`). The RDFS semantics also allow the extended notion of a *D-Interpretation* which is concerned with a subset of the XML Schema data-types. Since a specific *data-type map* may define the value spaces for certain data-types as disjoint, it becomes possible for an RDF graph to have no satisfying D-Interpretation – in this case the graph becomes inconsistent with respect to data-types (a *data-type clash*). L2 does not include this option. However, OWL 2 RL covers a comparatively large subset of the XML Schema data-types, which are usually suitable for implementation on rule based systems and can be practically relevant for several use-cases. The set of supported data-types is shown in Table 2.4

2.2.4 Summary

In this section we summarize the main differences between L2 and OWL 2 RL in a concise fashion.

- Both L2 and OWL 2 RL target similar application scenarios that require additional tractable reasoning on top-of RDFS;



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
--

Table 2.4: OWL 2 RL datatypes

- L2 and OWL 2 RL support a common core set of language primitives. OWL 2 RL additionally provides further language features that cleanly extend L2, but places some additional syntactic constraints on these additional language features (see Table 2.3);
- Both languages are defined in terms of first-order entailment rules (essentially Horn rules) and can be implemented on rule-engines;
- L2 and OWL 2 RL inferences are both applicable to arbitrary RDF graphs, including those that employ meta-modeling;
- Both L2 and OWL 2 RL allow axiomatic triples to be omitted in order to avoid potential scalability pitfalls;
- Inference within OWL 2 RL has the same complexity as for L2 (PTIME-complete in general). However, its entailment rules are slightly more complex than L2's and require some additional implementation effort;
- OWL 2 RL has more extensive support for data-types.

In summary it can be said that OWL 2 RL is a straightforward extension of L2 when applied to general RDF graphs, which is the main application scenario within LarKC.



3. Conclusion

In this deliverable we have revisited the specification of L2, the initial recommended LarKC knowledge representation formalism, in the light of recent standardization activities. We compared L2 with one of the new OWL 2 profiles, namely OWL 2 RL, which was created with very similar objectives to those that guided the design of L2.

Both OWL 2 RL and L2 are defined via entailment rules that can be applied to general RDF graphs and both can deal with RDF-style meta-modeling, although OWL 2 RL also admits DL reasoners to perform more limited inference while still being compliant. Both languages relax the requirement to include the RDF(S) axiomatic triples in order to meet their scalability aims, but optionally allow them to be included. Most importantly (but not surprisingly), both languages support roughly the same language primitives, with OWL 2 RL being more comprehensive.

The standardization work towards OWL 2 RL has resulted in a useful, scalable and moderately expressive profile within the OWL 2 language stack that forms a relevant bridge between RDFS and more comprehensive knowledge representation communities. Being implementable both on rule engines and in a restricted fashion on Description Logic reasoners is an additional bonus.

Furthermore, the added expressivity of OWL 2 RL when compared to L2 comes at the price of slightly more complex inference and restrictions on some of the modeling primitives. However, since those restrictions have no impact on any of the core language primitives which overlap with L2, we think that it is feasible to go forward using OWL 2 RL as the baseline language for LarKC, since it is a straightforward extension of the functionality provided by L2.



REFERENCES

- [1] D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. *W3C Recommendation*, 2, 2004.
- [2] T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15(3):289–323, 1995.
- [3] B. Grosz, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: Combining Logic Programs with Description Logic. 2003.
- [4] W3C OWL Working Group. OWL 2 Web Ontology Language: Document Overview. *W3C Working Draft*, 2009.
- [5] D.L. McGuinness, F. van Harmelen, et al. OWL Web Ontology Language Overview. *W3C Recommendation*, 10:2004–03, 2004.
- [6] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and editors C. Lutz. OWL 2 Web Ontology Language: Profiles. *W3C Proposed Recommendation*, 2009.
- [7] B. Motik, P.F. Patel-Schneider, and editors B. Parsia. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. *W3C Candidate Recommendation*, 2009.
- [8] M. Schneider, J. Carroll, I. Herman, and editors P.F. Patel-Schneider. OWL 2 Web Ontology Language RDF-Based Semantics. *W3C Proposed Recommendation*, 2009.
- [9] Herman J. ter Horst. Combining rdf and part of owl with rules: Semantics, decidability, complexity. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 668–684. Springer, 2005.
- [10] T.D. Wang, B. Parsia, and J. Hendler. A survey of the web ontology landscape. *Lecture Notes in Computer Science*, 4273:682, 2006.



A. Entailment Rules

A.1 OWL 2 RL

In the following we list the complete set of (non data-type related) entailment rules for OWL 2 RL (also see [6]). We have kept the same syntactic style and functional grouping as in the original specification document, including the simplification used to represent constructs that map to RDF containers (list constructs in particular), which are not used for L2. In this case `LIST[h, e1, ..., en]` is used as a syntactic shortcut for a conjunction of triples in order to simplify the presentation of the rules. For the formal expansion of this shortcut please see the official reference document.

The set of OWL 2 RL entailment rules is split into several different groups. Table A.1 is concerned with the semantics of equality. It defines the reflexivity, transitivity, etc, of `owl:sameAs`. While the rules are slightly different they essentially capture the same behavior as the corresponding L2 rules. The rules in table A.3 capture the basic semantics of schema vocabulary for properties and classes, and partially subsume the RDFS semantics or extend them with functionality that already existed implicitly in RDFS (e.g. class equivalence). In turn Table A.2 captures the semantics of the class vocabulary added on top of RDFS. Table A.4 and Table A.5 cover axioms which capture additional information about classes and properties respectively – this functionality is to a large degree also available in L2.

Table A.1: Entailment rules encoding the semantics of (in-)equality

	IF	THEN
eq-ref	?s ?p ?o	?s owl:sameAs ?s ?p owl:sameAs ?p ?o owl:sameAs ?o
eq-sym	?x owl:sameAs ?y	?y owl:sameAs ?x
eq-trans	?x owl:sameAs ?y ?y owl:sameAs ?z	?x owl:sameAs ?z
eq-rep-s	?s owl:sameAs ?s' ?s ?p ?o	?s' ?p ?o
eq-rep-p	?p owl:sameAs ?p' ?s ?p ?o	?s ?p' ?o
eq-rep-o	?o owl:sameAs ?o' ?s ?p ?o	?s ?p ?o'
eq-diff1	?x owl:sameAs ?y ?x owl:differentFrom ?y	false
eq-diff2	?x rdf:type owl:AllDifferent ?x owl:members ?y LIST[?y, ?z ₁ , ..., ?z _n] ?z _i owl:sameAs ?z _j	false



eq-diff3	?x rdf:type owl:AllDifferent ?x owl:distinctMembers ?y LIST[?y, ?z ₁ , ..., ?z _n] ?z _i owl:sameAs ?z _j	false
----------	--	-------

Table A.2: Basic entailment rules concerned with the semantics of classes

	IF	THEN
cax-sco	?c ₁ rdfs:subClassOf ?c ₂ ?x rdf:type ?c ₁	?x rdf:type ?c ₂
cax-eqc1	?c ₁ owl:equivalentClass ?c ₂ ?x rdf:type ?c ₁	?x rdf:type ?c ₂
cax-eqc2	?c ₁ owl:equivalentClass ?c ₂ ?x rdf:type ?c ₂	?x rdf:type ?c ₁
cax-dw	?c ₁ owl:disjointWith ?c ₂ ?x rdf:type ?c ₁ ?x rdf:type ?c ₂	false
cax-adc	?x rdf:type owl:AllDisjointClasses ?x owl:members ?y LIST[?y, ?c ₁ , ..., ?c _n] ?z rdf:type ?c _i ?z rdf:type ?c _j	false

Table A.3: Entailment rules covering the schema semantics of RDFS as well as additional OWL constructs

	IF	THEN
scm-cls	?c rdf:type owl:Class	?c rdfs:subClassOf ?c ?c owl:equivalentClass ?c ?c rdfs:subClassOf owl:Thing owl:Nothing rdfs:subClassOf ?c
scm-sco	?c ₁ rdfs:subClassOf ?c ₂ ?c ₂ rdfs:subClassOf ?c ₃	?c ₁ rdfs:subClassOf ?c ₃
scm-eqc1	?c ₁ , owl:equivalentClass ?c ₂	?c ₁ rdfs:subClassOf ?c ₂ ?c ₂ rdfs:subClassOf ?c ₁
scm-eqc2	?c ₁ rdfs:subClassOf ?c ₂ ?c ₂ rdfs:subClassOf ?c ₁	?c ₁ owl:equivalentClass ?c ₂
scm-dp	?p rdf:type owl:DatatypeProperty	?p rdfs:subPropertyOf ?p ?p owl:equivalentProperty ?p
scm-spo	?p ₁ rdfs:subPropertyOf ?p ₂ ?p ₂ rdfs:subPropertyOf ?p ₃	?p ₁ rdfs:subPropertyOf ?p ₃ ?p ₂ owl:equivalentProperty ?p ₃



scm-eqp1	?p ₁ owl:equivalentProperty ?p ₂	?p ₁ rdfs:subPropertyOf ?p ₂ ?p ₂ rdfs:subPropertyOf ?p ₁
scm-eqp2	?p ₁ rdfs:subPropertyOf ?p ₂ ?p ₂ rdfs:subPropertyOf ?p ₁	?p ₁ owl:equivalentProperty ?p ₂
scm-dom1	?p rdfs:domain ?c ₁ ?c ₁ rdfs:subClassOf ?c ₂	?p rdfs:domain ?c ₂
scm-dom2	?p ₂ rdfs:domain ?c ?p ₁ rdfs:subPropertyOf ?p ₂	?p ₁ rdfs:domain ?c
scm-rng1	?p rdfs:range ?c ₁ ?c ₁ rdfs:subClassOf ?c ₂	?p rdfs:range ?c ₂
scm-rng2	?p ₂ rdfs:range ?c ?p ₁ rdfs:subPropertyOf ?p ₂	?p ₁ rdfs:range ?c
scm-hv	?c ₁ owl:hasValue ?i ?c ₁ owl:onProperty ?p ₁ ?c ₂ owl:hasValue ?i ?c ₂ owl:onProperty ?p ₂ ?p ₁ rdfs:subPropertyOf ?p ₂	?c ₁ rdfs:subClassOf ?c ₂
scm-svf1	?c ₁ owl:someValuesFrom ?y ₁ ?c ₁ owl:onProperty ?p ?c ₂ owl:someValuesFrom ?y ₂ ?c ₂ owl:onProperty ?p ?y ₁ rdfs:subClassOf ?y ₂	?c ₁ rdfs:subClassOf ?c ₂
scm-svf2	?c ₁ owl:someValuesFrom ?y ?c ₁ owl:onProperty ?p ₁ ?c ₂ owl:someValuesFrom ?y ?c ₂ owl:onProperty ?p ₂ ?p ₁ rdfs:subPropertyOf ?p ₂	?c ₁ rdfs:subClassOf ?c ₂
scm-avf1	?c ₁ owl:allValuesFrom ?y ₁ ?c ₁ owl:onProperty ?p ?c ₂ owl:allValuesFrom ?y ₂ ?c ₂ owl:onProperty ?p ?y ₁ rdfs:subClassOf ?y ₂	?c ₁ rdfs:subClassOf ?c ₂
scm-avf2	?c ₁ owl:allValuesFrom ?y ?c ₁ owl:onProperty ?p ₁ ?c ₂ owl:allValuesFrom ?y ?c ₂ owl:onProperty ?p ₂ ?p ₁ rdfs:subPropertyOf ?p ₂	?c ₂ rdfs:subClassOf ?c ₁
scm-int	?c owl:intersectionOf ?x LIST[?x, ?c ₁ , ..., ?c _n]	?c rdfs:subClassOf ?c ₁ ?c rdfs:subClassOf ?c ₂ ... ?c rdfs:subClassOf ?c _n
scm-uni	?c owl:unionOf ?x LIST[?x, ?c ₁ , ..., ?c _n]	?c ₁ rdfs:subClassOf ?c ?c ₂ rdfs:subClassOf ?c ... ?c _n rdfs:subClassOf ?c

Table A.4: Entailment rules capturing the semantics of class axioms

	IF	THEN
cls-thing		owl:Thing rdf:type owl:Class
cls-nothing1		owl:Nothing rdf:type owl:Class
cls-nothing2	?x rdf:type owl:Nothing	false
cls-int1	?c owl:intersectionOf ?x LIST[?x, ?c ₁ , ..., ?c _n] ?y rdf:type ?c ₁ ?y rdf:type ?c ₂ ... ?y rdf:type ?c _n	?y rdf:type ?c
cls-int2	?c owl:intersectionOf ?x LIST[?x, ?c ₁ , ..., ?c _n] ?y rdf:type ?c	?y rdf:type ?c ₁ ?y rdf:type ?c ₂ ... ?y rdf:type ?c _n
cls-uni	?c owl:unionOf ?x LIST[?x, ?c ₁ , ..., ?c _n] ?y rdf:type ?c _i	?y rdf:type ?c
cls-com	?c ₁ owl:complementOf ?c ₂ ?x rdf:type ?c ₁ ?x rdf:type ?c ₂	false
cls-svf1	?x owl:someValuesFrom ?y ?x owl:onProperty ?p ?u ?p ?v ?v rdf:type ?y	?u rdf:type ?x
cls-svf2	?x owl:someValuesFrom owl:Thing ?x owl:onProperty ?p ?u ?p ?v	?u rdf:type ?x
cls-avf	?x owl:allValuesFrom ?y ?x owl:onProperty ?p ?u rdf:type ?x ?u ?p ?v	?v rdf:type ?y
cls-hv1	?x owl:hasValue ?y ?x owl:onProperty ?p ?u rdf:type ?x	?u ?p ?y
cls-hv2	?x owl:hasValue ?y ?x owl:onProperty ?p ?u ?p ?y	?u rdf:type ?x
cls-maxc1	?x owl:maxCardinality "0"^^xsd:nonNegativeInteger ?x owl:onProperty ?p ?u rdf:type ?x ?u ?p ?y	false



cls-maxc2	?x owl:maxCardinality "1"^^xsd:nonNegativeInteger ?x owl:onProperty ?p ?u rdf:type ?x ?u ?p ?y ₁ ?u ?p ?y ₂	?y ₁ owl:sameAs ?y ₂
cls-maxqc1	?x owl:maxQualifiedCardinality "0"^^xsd:nonNegativeInteger) ?x owl:onProperty ?p ?x owl:onClass ?c ?u rdf:type ?x ?u ?p ?y ?y rdf:type ?c	false
cls-maxqc2	?x owl:maxQualifiedCardinality "0"^^xsd:nonNegativeInteger ?x owl:onProperty ?p ?x owl:onClass owl:Thing ?u rdf:type ?x ?u ?p ?y	false
cls-maxqc3	?x owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger ?x owl:onProperty ?p ?x owl:onClass ?c ?u rdf:type ?x ?u ?p ?y ₁ ?y ₁ rdf:type ?c ?u ?p ?y ₂ ?y ₂ rdf:type ?c	?y ₁ owl:sameAs ?y ₂
cls-maxqc4	?x owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger ?x owl:onProperty ?p ?x owl:onClass owl:Thing ?u rdf:type ?x ?u ?p ?y ₁ ?u ?p ?y ₂	?y ₁ owl:sameAs ?y ₂
cls-oo	?c owl:oneOf ?x LIST[?x, ?y ₁ , ..., ?y _n]	?y ₁ rdf:type ?c ... ?y _n rdf:type ?c

Table A.5: Entailment rules capturing the semantics of property axioms

	IF	THEN
prp-dom	?p rdfs:domain ?c ?x ?p ?y	?x rdf:type ?c



prp-rng	?p rdfs:range ?c x? ?p ?y	?y rdf:type ?c
prp-fp	?p rdf:type owl:FunctionalProperty ?x ?p ?y ₁ ?x ?p ?y ₂	?y ₁ owl:sameAs ?y ₂
prp-ifp	?p rdf:type owl:InverseFunctionalProperty ?x ₁ ?p ?y ?x ₂ ?p ?y	?x ₁ owl:sameAs ?x ₂
prp-irp	?p rdf:type owl:IrreflexiveProperty ?x ?p ?x	false
prp-symp	?p rdf:type owl:SymmetricProperty ?x ?p ?y	?y ?p ?x
prp-asymp	?p rdf:type owl:AsymmetricProperty ?x ?p ?y ?y ?p ?x	false
prp-trp	?p rdf:type owl:TransitiveProperty ?x ?p ?y ?y ?p ?z	?x ?p ?z
prp-spo1	?p ₁ rdfs:subPropertyOf ?p ₂ ?x ?p ₁ ?y	?x ?p ₂ ?y
prp-spo2	?p owl:propertyChainAxiom ?x LIST[?x, ?p ₁ , ..., ?p _n] ?u ₁ ?p ₁ ?u ₂ ?u ₂ ?p ₂ ?u ₃ ... ?u _n ?p _n ?u _{n+1}	?u ₁ ?p ?u _{n+1}
prp-eqp1	?p ₁ owl:equivalentProperty ?p ₂ ?x ?p ₁ ?y	?x ?p ₂ ?y
prp-eqp2	?p ₁ owl:equivalentProperty ?p ₂ ?x ?p ₂ ?y	?x ?p ₁ ?y
prp-pdw	?p ₁ owl:propertyDisjointWith ?p ₂ ?x ?p ₁ ?y ?x ?p ₂ ?y	false
prp-adp	?x rdf:type owl:AllDisjointProperties ?x owl:members ?y LIST[?y, ?p ₁ , ..., ?p _n] ?u ?p _i ?v ?u ?p _j ?v	false
prp-inv1	?p ₁ owl:inverseOf ?p ₂ ?x ?p ₁ ?y	?y ?p ₂ ?x
prp-inv2	?p ₁ owl:inverseOf ?p ₂ ?x ?p ₂ ?y	?y ?p ₁ ?x



prp-key	?c owl:hasKey ?u LIST[?u, ?p ₁ , ..., ?p _n] ?x rdf:type ?c ?x ?p ₁ ?z ₁ ... ?x ?p _n ?z _n ?y rdf:type ?c ?y ?p ₁ ?z ₁ ... ?y ?p _n ?z _n	?x owl:sameAs ?y
prp-npa1	?x owl:sourceIndividual ?i ₁ ?x owl:assertionProperty ?p ?x owl:targetIndividual ?i ₂ ?i ₁ ?p ?i ₂	false
prp-npa2	?x owl:sourceIndividual ?i ?x owl:assertionProperty ?p ?x owl:targetValue ?lt ?i ?p ?lt	false



A.2 L2

This section lists the L2 entailment rules that are meant to be used in addition to the RDFS entailment rules listed in [1]. Rule (1) and (2) cover symmetry and transitivity of properties. Rules (3a) and (3b) formalize the reflexivity of individual equivalence. Rule (4) captures reflexivity and rule (5) transitivity of individual equivalence. Rule (6) and (7) cover the semantics of inverse properties, including its reflexivity. Rules (8) and (9) denote that individuals that are classes or properties are considered sub-classes or sub-properties of themselves. These rules are important to facilitate basic meta-modeling in the language. Rules (11a), (11b) and (11c) express the semantics of class equivalence, while (12a), (12b) and (12c) do the same for property equivalence.

Table A.6: L2 semantics as first-order implications

	IF	THEN
1	?p rdf:type owl:SymmetricProperty ?v ?p ?w	?w ?p ?v
2	?p rdf:type owl:TransitiveProperty ?u ?p ?v ?v ?p ?w	?u ?p ?w
3a	?v ?p ?w	?v owl:sameAs ?v
3b	?v ?p ?w	?w owl:sameAs ?w
4	?v owl:sameAs ?w	?w owl:sameAs ?v
5	?u owl:sameAs ?v ?v owl:sameAs ?w	?u owl:sameAs ?w
6	?p owl:inverseOf ?q ?v ?p ?w	?w ?q ?v
7	?p owl:inverseOf ?q ?v ?q ?w	?w ?p ?v
8	?v rdf:type rdfs:Class ?v owl:sameAs ?w	?v rdfs:subClassOf ?w
9	?p rdf:type rdfs:Property ?p owl:sameAs ?q	?p rdfs:subPropertyOf ?q
10	?u ?p ?v ?u owl:sameAs ?w ?v owl:sameAs ?q	?w ?p ?q
11a	?v owl:equivalentClass ?w	?v rdfs:subClassOf ?w
11b	?v owl:equivalentClass ?w	?w rdfs:subClassOf ?v
11c	?v rdfs:subClassOf ?w ?w rdfs:subClassOf ?v	?v owl:equivalentClass ?w
12a	?v owl:equivalentProperty ?w	?v rdfs:subProperty ?w
12b	?v owl:equivalentProperty ?w	?w rdfs:subProperty ?v
12c	?v rdfs:subPropertyOf ?w ?w rdfs:subPropertyOf ?v	?v owl:equivalentProperty ?w



B. Axiomatic Triples

Table B.1: The RDF axiomatic triples

<pre>rdf:type rdf:type rdf:Property . rdf:subject rdf:type rdf:Property . rdf:predicate rdf:type rdf:Property . rdf:object rdf:type rdf:Property . rdf:first rdf:type rdf:Property . rdf:rest rdf:type rdf:Property . rdf:value rdf:type rdf:Property . rdf:_1 rdf:type rdf:Property . rdf:_2 rdf:type rdf:Property rdf:nil rdf:type rdf:List .</pre>

Table B.2: The RDFS axiomatic triples

<pre>rdf:type rdfs:domain rdfs:Resource . rdfs:domain rdfs:domain rdf:Property . rdfs:range rdfs:domain rdf:Property . rdfs:subPropertyOf rdfs:domain rdf:Property . rdfs:subClassOf rdfs:domain rdfs:Class . rdf:subject rdfs:domain rdf:Statement . rdf:predicate rdfs:domain rdf:Statement . rdf:object rdfs:domain rdf:Statement . rdfs:member rdfs:domain rdfs:Resource . rdf:first rdfs:domain rdf:List . rdf:rest rdfs:domain rdf:List . rdfs:seeAlso rdfs:domain rdfs:Resource . rdfs:isDefinedBy rdfs:domain rdfs:Resource . rdfs:comment rdfs:domain rdfs:Resource . rdfs:label rdfs:domain rdfs:Resource . rdf:value rdfs:domain rdfs:Resource . rdf:type rdfs:range rdfs:Class . rdfs:domain rdfs:range rdfs:Class . rdfs:range rdfs:range rdfs:Class . rdfs:subPropertyOf rdfs:range rdf:Property . rdfs:subClassOf rdfs:range rdfs:Class . rdf:subject rdfs:range rdfs:Resource . rdf:predicate rdfs:range rdfs:Resource . rdf:object rdfs:range rdfs:Resource . rdfs:member rdfs:range rdfs:Resource . rdf:first rdfs:range rdfs:Resource . rdf:rest rdfs:range rdf:List . rdfs:seeAlso rdfs:range rdfs:Resource .</pre>



```
rdfs:isDefinedBy rdfs:range rdfs:Resource .
rdfs:comment rdfs:range rdfs:Literal .
rdfs:label rdfs:range rdfs:Literal .
rdf:value rdfs:range rdfs:Resource .
rdf:Alt rdfs:subClassOf rdfs:Container .
rdf:Bag rdfs:subClassOf rdfs:Container .
rdf:Seq rdfs:subClassOf rdfs:Container .
rdfs:ContainerMembershipProperty rdfs:subClassOf rdf:Property .
rdfs:isDefinedBy rdfs:subPropertyOf rdfs:seeAlso .
rdf:XMLLiteral rdf:type rdfs:Datatype .
rdf:XMLLiteral rdfs:subClassOf rdfs:Literal .
rdfs:Datatype rdfs:subClassOf rdfs:Class .
rdf:_1 rdf:type rdfs:ContainerMembershipProperty .
rdf:_1 rdfs:domain rdfs:Resource .
rdf:_1 rdfs:range rdfs:Resource .
rdf:_2 rdf:type rdfs:ContainerMembershipProperty .
rdf:_2 rdfs:domain rdfs:Resource .
rdf:_2 rdfs:range rdfs:Resource .
...
```