TripCom

Triple Space Communication

FP6 – 027324

Deliverable

D2.3

Ontology and rule representation in a tuple space

Vassil Momtchev (Ontotext)
Stijn Heymans (LFUI)
Jos de Bruijn (LFUI)
Axel Pollers (NUIG)
Lyndon Nixon (FUB)

April 25, 2008

Executive Summary

This deliverable describes the approaches, design and implementation of reasoning into Triple Space. We introduce two alternative approaches for extending the Triple Space knowledge representation formalisms using RDF native reasoner (TRREE) and F-logic datalog reasoner. We conclude that the solution that best matches the goals with respect to the scalablity and current specifications of TripCom use cases is the TRREE based one. In scenarios where user-defined rules have more importance and a better expressivity is required, it is advisable to use WSML-Flight and the IRIS based solution.

Document Information

IST Project Number	FP6 – 027324	Acronym	TripCom
Full Title	Triple Space Communication		
Project URL	http://www.tripcom.org/		
Document URL			
EU Project Officer	Werner Janusch		

Deliverable	Number	2.3	Title	Ontology and rule representation in a tuple space
Work Package	Number	2	Title	Triple Space Knowledge Representation

Date of Delivery	Contractual	M24	Actual	31-Mar-08
Status	version 0.2		final ☐	
Nature	prototype ☐ report ☒ dissemination ☐			
Dissemination Level	public ☒ consortium ☐			

Authors (Partner)	Vassil Momtchev (Ontotext)		
Resp. Author	Vassil Momtchev(Ontotext)	E-mail	vassil.momtchev@ontotext.com
	Partner Ontotext	Phone	+359-2-8091553

Abstract (for dissemination)	This deliverable describes the approaches, design and implementation of reasoning into Triple Space. We introduce two alternative approaches for extending the Triple Space knowledge representation formalisms using RDF native reasoner (TRREE) and F-logic reasoner (IRIS). We conclude that the solution that best matches the goals with respect to the scalablity and current specifications of TripCom use cases is the TRREE based one. In scenarios where user-defined rules have more importance and a better expressivity is required, it is advisable to use WSML-Flight and the IRIS based solution.
Keywords	triplespace, reasoning, OWL, WSML, rules

Version Log			
Issue Date	Rev No.	Author	Change
27 June 2007	1	Vassil Momtchev	Initial document structure proposal
10 October	2	Vassil Momtchev	Draft of knowledge representation in a triplespace
21 December 2007	3	Stijn Heymans and Jos de Bruijn	Entailment regimes in F-Logic
22 February 2007	4	Vassil Momtchev	Entailment regimes in TRREE and LP section

Project Consortium Information

Acronym	Partner	Contact
Semantic Technology Institute Innsbruck http://www.sti-innsbruck.at	STI	Prof. Dr. Dieter Fensel Semantic Technology Institute (STI) Innsbruck, Austria E-mail: dieter.fensel@sti-innsbruck.at
National University of Ireland, Galway http://www.deri.ie	NUIG	Dr. Laurentiu Vasiliu Digital Enterprise Research Institute (DERI) Galway, Ireland Email: laurentiu.vasiliu@deri.org
University of Stuttgart http://www.iaas.uni-stuttgart.de/	USTUTT	Prof.Dr. Frank Leymann Inst. für Architektur von Anwendungssystemen (IAAS) Stuttgart, Germany E-mail: frank.leymann@informatik.uni-stuttgart.de
Vienna university of Technology http://www.complang.tuwien.ac.at/	TUW	Prof.Dr. eva Kühn Institut für Computersprachen Vienna, Austria E-mail: eva@complang.tuwien.ac.at
Free University Berlin http://www.ag-nbi.de/	FUB	Prof. Dr.-Ing. Robert Tolksdorf AG Netzbasierte Informationssysteme Berlin, Germany E-mail : tolk@inf.fu-berlin.de
Ontotext Lab, Sirma Group Corp. http://www.ontotext.com/	ONTO	Atanas Kiryakov, Vassil Momtchev, Ontotext Lab, Sirma Group Corp. Sofia, Bulgaria E-mail: vassil.momtchev@ontotext.com
Profium OY http://www.profium.com/	Profium	Dr. Janne Saarela Profium OY Espoo, Finland E-mail: janne.saarela@profium.com
CEFRIEL SCRL. http://www.cefriel.it/	CEFRIEL	Davide Cerri CEFRIEL SCRL. Milano, Italy E-mail: cerri@cefriel.it
Telefonica I+D http://www.tid.es/	TID	Noelia Pérez Crespo Telefonica I+D Madrid, España E-mail: npc@tid.es

Table of Contents

List of Abbreviations

ANSI	American National Standards Institute
BSD	Berkeley Software Distribution
DAWG	Data Access Working Group
DBMS	Database Management Systems
ER	Entity Relationship
FOAF	Friend Of a Friend
HTTP	Hyper Text Transfer Protocol
iTQL	Interactive Tucana Query Language
JRDF	Java RDF
LAN	Local Area Network
LGPL	GNU Lesser General Public Licence
N3	Notation 3
N3QL	N3 Query Language
NDM	Oracle Spatial Network Data Model
OASIS	Organization for the Advancement of Structured Information Standards
ORDI	Ontology Representation and Data Integration
OWL	Web Ontology Language
OWLIM	OWL In Memory
RDBMS	Relational DBMS
RDF	Resource Description Framework
RDFS	RDF Schema
RDQL	RDF Data Query Language
ROI	RDF Input/Output
SAIL	Storage And Inference Layer
SOFA	Simple Ontology Framework API
SOAP	Simple Object Access Protocol
SeRQL	Sesame RDF Query Language
SEQUEL	Structured English Query Language
SPARQL	SPARQL Protocol And RDF Query Language
SQL	Structured Query Language
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
WSMO	Web Service Modeling Language
XML	Extensible Markup Language
YARS	Yet Another RDF Store
YARSQL	YARS Query Language

1 Introduction

According to the TripCom project work plan, work package WP2 deals with the issues of representing semantic data models such as RDF and OWL/WSML in ways that are efficient and the same time consistent with the principles of space-based computing, and in the scope of the project, that match the requirements of large scale Semantic Web applications and services. So far, the project concentrated on the use of RDF as formalism to describe application data and administrative metadata. The role of this deliverable is to look beyond this fundamental formalism and to consider representations that are more expressive than RDF(S). In other words, we look at means and models to integrate language families such as OWL and WSML.

There are several reasons why we expect to provide added value by integrating languages other than RDF(S) with Triple Space:

- Representing ontological structure in RDF graphs is awkward, as instance data and structural metadata end up in one and the same data construct.
- Semantic Web languages that are located on top of RDF can provide additional reasoning capabilities and modeling constructs.
- However, more expressive axiomatic expressions given in e.g. OWL or WSML are lost when transformed to RDF and have to be extracted again in extra procedures before the actual reasoning task can be executed at the level of these languages.

We start with the evaluation of the current Triple Space data models and investigate to what extent RDF could suffice as representational formalism. We look at models, which are backwards compatible with RDF, for representing the greater expressiveness of ontologies through triples, alongside existing RDF-based syntaxes.

This deliverable focuses in particular on extending the Triple Space knowledge modeling framework toward Logic Programming (e.g. by applying Datalog, F-Logic or the TRREE rule language part of the ORDI storage infrastructure used in TripCom implementation). In this respect Chapter 3 discusses the advantages of Logic Programming over Description Logic. The results of the discussion give the arguments that led to the decision to follow a route towards a more expressive Triple Space.

Further on we propose two alternative Logic Programming-based approaches to represent knowledge in triplespaces. The first one is based on the use of the rule language extension of the RDF reasoner TRREE, which supports SPARQL as query language and implements TripCom storage infrastructure interfaces. The application of TRREE is thus an obvious continuation of the work already done in the TripCom project. The second proposal is aligned with recent achievements of the IRIS datalog reasoner. This approach requires the RDF data model to be converted into F-Logic molecules in order to allow for an integral language framework. Further details on these two proposals towards a backward compatible extension towards Logic Programming are given in Chapter 5 (TRREE) and Chapter 4 (IRIS).

Before concluding the deliverable, Chapter 6 evaluates these extended TripCom knowledge representation models and discusses their use and applicability in the context of the project's use cases. The results of this analysis provide valuable feedback for the continuation of the project with respect to formalism to use, the reasoning support to implement and integrate with the Triple Space kernel releases.

2 Semantic tuple model evaluation

This chapter outlines the most important aspects of the tuple model used in TripCom project. An evaluation of the main features is carried out with respect to realization of semantic-aware space based applications and the advantages over traditional Linda-like systems.

The original Linda system operates only with primitive data types. This imposes restrictions over the expressivity of the tuples and the mechanism for co ordination. [27] and [23] describe coordination systems that resolve this problem by introducing more expressive models using objects, where template matching is enriched with the polymorphic behaviour of Java classes. Despite the significant increase of expressivity and compatibility with the existing object-oriented platforms, the approach may be considered conflicting with the original tuple space paradigm for a complete decoupling of the clients. sTuple is a system to go even further and combines the Tuple Space technology with the Semantic Web, [17]. It is an extension of JavaSpaces [28] introduces the concept of *semantic tuple*. *Semantic tuple* is a tuple having at least one of its fields typed as an RDF types. Hence, it is possible to be annotated by information in an ontology and used to infer implicit knowledge.

2.1 Semantic triple

TripCom tuple space model is bound to RDF data model according the specification in section 4 of [27]. Three data types URI, literal and blank node are used to construct statements in the form subject-predicate- object. The subject indicates a resource expressed by URI (identifiable) or blank node (not-identifiable), predicate represent a relationship determined by URI and the object is another resource or literal. Thus, a new type of semantic tuple is required to comply with the formal semantic of a RDF triple. Semantic triple is a specialization of the semantic tuple as a RDF triple, which has as first element a subject expressed as resource (URI or blank node), second element predicate (URI) and third element object (URI, blank node or literal). The association of TripCom tuple space model with the concept of *semantic triples* increases flexibility because:

- Compliance with the original Linda system design.
- Compatibility with core Semantic Web formal languages, which can be represented in the RDF data model and the concept of semantic triple.
- Simple mapping to already existing infrastructure such as reasoner and triple stores.

Despite the simplicity of the model in the context of multi-agent systems the idea of universal truth does not ensure completly the required level of flexibility. The proposed model for semantic triple addressing is conceptually correct, however it does not comply with the Linda or Semantic Web technologies. So, a new type of semantic tuple even further specialized is required.

2.2 Semantic tripleset statements

Conceptually the approach to use *semantic triples* is correct, however it is hardly feasible to associate meta-data to RDF statements without to step into the reification bloat (four triples to refer a single statement) or use tuple identifiers, which leads to sever compliance problems and consistency check overhead. The *semantic tripleset statement* is a specialization of the semantic triple to include also the associated meta-data like triplespace identifier or further meta-information defined by the TS Ontology. The complete model is specified in [20], so a very short overview will be presented to explain the semantics of the formal definition. Triplesets represent groups (or sets) of contextualized triples (quads). Each tripleset can be considered as a tag, which can be associated with a specific triple. Triplesets are independent from the *named graphs* – triples from different *named graphs* can coexist in one and the same tripleset. In contrast to the *named graphs*, the semantics of the triplesets impose linking of or association triples, instead of copying them.

For instance, the sample semantic triple below denotes a specific history of a patient. Using the simple knowledge representation formalism of the semantic triples, it is impossible to conclude who was the author of the statement or what was the space it was written to.

```
<patientEPSInst1, hasSocialHistory, p_history1>
```

The same piece of knowledge could be enriched using a named graph to indicate the space identifier and triplesets to tag the statement to a set of meta-data instances, further described in the same graph.

```
<patientEPSInst1, hasSocialHistory, p_history1, spaceURI, <author1>>
<author1, type, Author, spaceURI, <metaData>>
<author1, hasName, "Vassil Momtchev", spaceURI, <metaData>>
```

According to the prescribed semantics of the named graph, two different tuples may coexist if written in different spaces (named graphs). However, in the context of single space all meta-data is linked to a single statement and if it is deleted the relation between data and meta-data is lost.

2.3 Triplespace

This section presents an evaluation of different possibilities for represent information in a triplespace, whether the triplespace is composed of data or knowledge (i.e, allowed to duplicate triples), what is the scope of blank node identifiers (they should be limited to the current triplespace only), and the relation between different triplespaces on data model level (i.e, all statements of super triplespaces are contained also in the current triplespace)

A triplespace can be seen as a specialized form of tuplespace which contains semantic tuples, as opposed to tuples whose fields contain instances of datatypes as in classical tuplespace systems. Semantic tuple fields contain instances of concepts whose meaning is described in knowledge models such as RDF Schema. Semantic tuples form an atomic unit of meaning (a knowledge statement). A triplespace can be seen as a bag of semantic tuples and hence as a collection of statements of knowledge.

Client access to the global Triple Space is always in context of one or more triplespaces. Each triplespace forms a clearly distinct, self-contained model of knowledge within the global knowledge stored in Triple Space. The fact that triplespaces can be interpreted as knowledge models and that every triplespace represents only a part of the global knowledge model within Triple Space raises issues that were not part of the classical tuplespace model. We will briefly discuss the issues here and outline the decisions we make in Triple Space.

- *Scope of triples*: Triplespaces also have parent and child spaces. Hence the knowledge they contain is visible to their parents and the knowledge of their children is visible to them. Thefore we must also recognize that the following issues may relate not only to an individual triplespace but to what we term 'the scope of the triplespace', meaning the triplespace itself plus all of its child triplespaces, all of their children and so forth.

- *Duplication*: In a triplespace, it is possible to emit a tuple which represents the same statement multiple times. However, from a knowledge modelling perspective a statement is equally true regardless if stated once or multiple times. So Triple Space can contain tuples which duplicate the same knowledge statement when interpreted as semantic triples yet we assume every tuple inserted to be in itself unique, regardless of its content. On the other hand, we decide that client interaction in a triplespace through the Triple Space API assumes the interpretation of tuple content as knowledge. When deleting matching statements the removal of a statement in a triplespace will also remove its duplicates in other tuples - within the scope of that triplespace.

- *Anonymous concepts*: It is possible to refer to anonymous concepts, i.e. concept which are not identified by any URI. This is the existential qualifier in First-Order Logic, or blank nodes in RDF. Since they lack an unique identifier, it is not possible in a triplespace to know which blank nodes potentially refer to the same concept. This is only possible through the Triple Space API, in which statements are emitted together to the triplespace with the same blank node identifier inside the same out operation. Hence "identical" blank nodes can only exist within the same triplespace (not one of them defined in a parent or child space).

- *Contradiction*: In a triplespace, content is interpreted as a model of knowledge which can be statements but also axioms which constraint those statements (e.g. a boss may only have one secretary). Some axioms are implicit in the knowledge representation (e.g. OWL- DL would disallow that a class is also an instance), others are explicit (through RDF Schema statements for example). Since Semantic Web knowledge models assume incompleteness of knowledge (Open World Assumption) constraint interpretation often attempts to avoid contradiction (e.g. if boss hasSecretary Sandra and hasSecretary Joanne, it could be assumed Sandra and Joanne are the same person). However, further axioms can be used to flag a contradiction and avoid faulty reasoning (Sandra differentFrom Joanne). In triplespaces, we have to decide if we allow contradictions, and if we interpret knowledge according to a Closed or Open World Assumption. This case seems to depend on which knowledge representations we choose to use, e.g. in the Semantic Web paradigm (rather than Logic Programming) we interpret statements

according to the Open World Assumption and will allow contradictions across triplespaces (i.e. among non- overlapping triplespaces). Within the scope of a triplespace, it may be desirable to allow space owners to specify if a space permits contradictory statements, either within itself or within its scope (including all child spaces). A possible rule of thumb may be to allow contradictions within the same space as default (and leave clients interacting over that space to resolve contradictions among themselves) and disallow child spaces from contradicting statements in the parent space.

3 Reasoning in a tuple space

The TripCom architecture [22] is a new form of network-based communication, which faciliates the development of large and complex integration projects. The proposed generic architecture can be used to realize a high number of different use case scenarios, so different levels of semantic support are required.

In this chapter we discuss different reasoning scenarios and what are the aspects in the interpretation of a triplespace. So far, the triplespace is realized using semantic tripleset data model, which is a simple data model. The data model uses a regular syntax to represent complex data structures. However, in TripCom this is not enough and the data should be further interpreted to derive meaningful implicit knowledge. We investigate reasoning with ontologies represented in knowledge representation frameworks such as Description Logics (OWL) and Logic Programming (F-Logic and TRREE rule language) in the context of RDF discuss how RDF reasoning can be integrated with higher fragments OWL and WSML respectively.

Here we provide a brief summary Description Logic (DL) and Logic Programming (LP). A detailed presentation and discussion of DL and LP formalism can be found in D3.2 [25].

Logical programming (LP) is used mainly as a common name for a family of rule-based logical languages, similar to PROLOG. One of the most popular LP languages is Datalog; still another is F-Logic, with its different flavours supported by Flora 2 and Ontobroker. Logic programming partly evolved as an extension of relational databases with deductive features. Therefore, LP typically focuses on efficient query answering over a bounded data set, and is often used in data-intensive applications that require managing large amounts of data like the scenarios used in the triple space. As a rule of thumb, systems based on LP are more efficient and scalable compared to the ones based on DL. However, a direct comparison is not possible, because these two paradigms support semantics of quite different nature - there are definitions that can be expressed in LP, while this is impossible in OWL-DL, and vice versa.

Motik in [21] motivates several important reasons to try to integrate OWL (DL formalism) with Logic Programs by investigating the following modeling shortcomings:

- OWL is not suitable for modeling general relation structures involving objects which are not connected in tree-like manner.
- OWL is not able to express polyadic predicates used to model arities beyond unary and binary relations
- Definition of integrity contraints and modeling exceptions.
- OWL by definition implies open world reasoning and it cannot answer negative queries. However, close-world query answering may be implemented in compatible way with the OWL semantics.

Description Logic Programs (DLP) is specified by Grosof [10] and emerged as a new dialect, offering a prospective compromise between expressive power, efficient reasoning, and compatibility. DLP is named to denote a common language that is able to integrate knowledge bases described in Description Logic with Logic Programs. In effect, OWL-DLP is the most expressive sub-language of OWL-DL that can be efficiently mapped to Datalog. OWL-DLP is simpler than OWL-Lite. Compared to

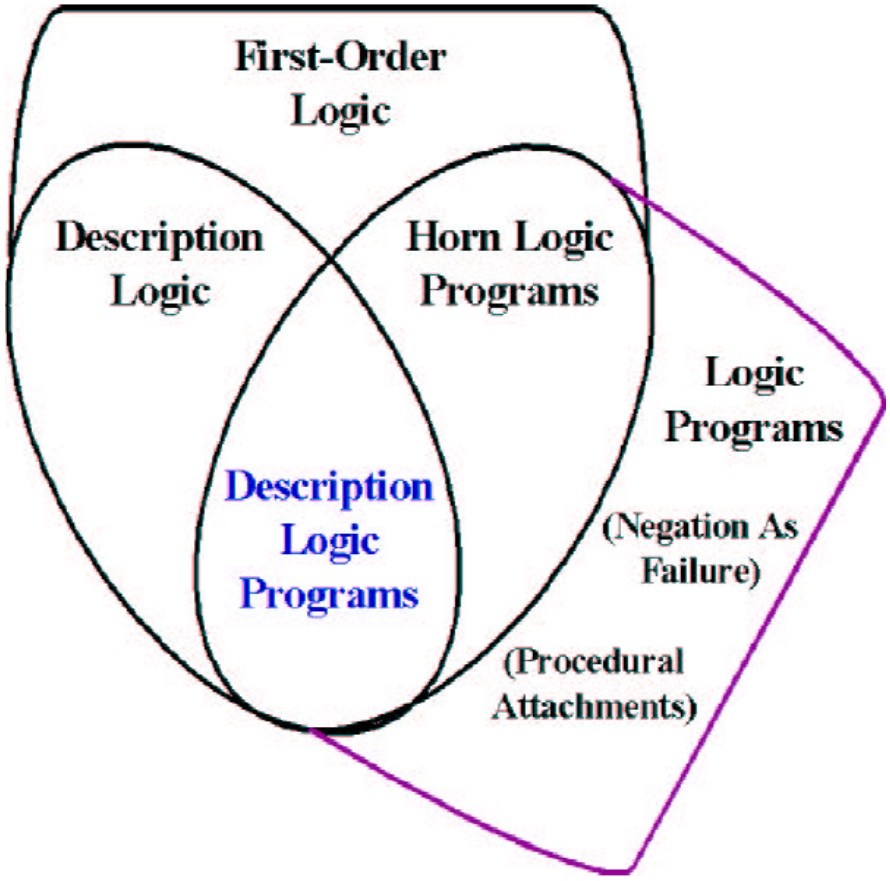

Figure 3.1: DLP defined as explicit intersection of DL with LP. [10].

the Lite and DL dialects, the semantics of OWL-DLP is easier to align to the one of RDFS. And yet, alignment can only be achieved through the enforcement of additional modelling constraints and transformations.

Even the simplest fragment of OWL (OWL-Lite) renders the concept satisfiability reasoning task to exponential time complete problem[1]. DL-Lite is notable exception and it is the first description logic ontology language with polynomial complexity for query answering [7]. Nevertheless, we base our work on LP formalisms, because they provide polynomial query answering complexity for static knowledge bases and are subject of better optimization techniques.

We propose the following architecture to intergate two alternative solutions. In the next chapter we investigate in details the different layers.

[1]http://www.cs.man.ac.uk/ ezolin/dl/

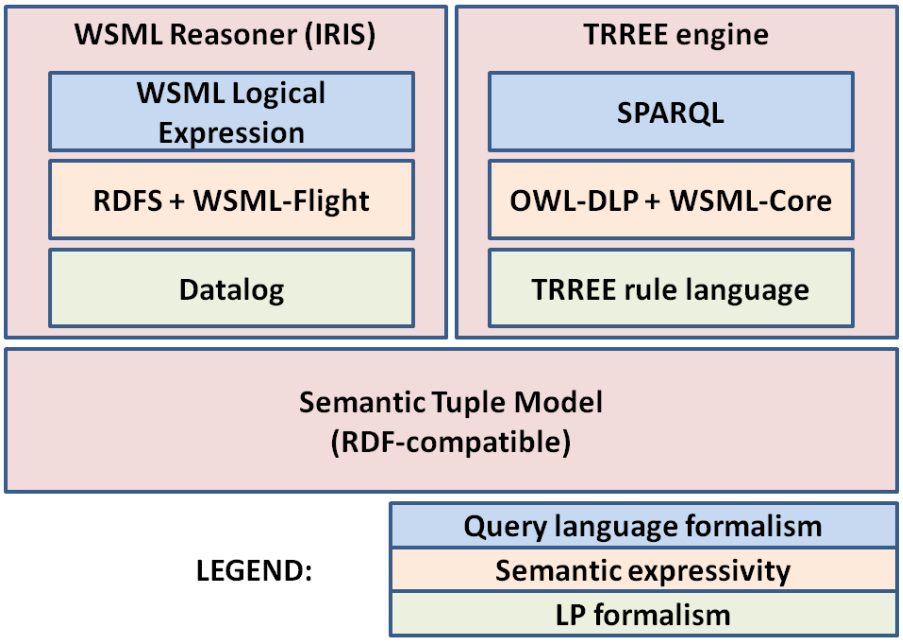

Figure 3.2: Two alternative solutions to support different ontology languages.

4 Entailment regimes in F-Logic

In this chapter we base our discussion on reasoning in the presence of both WSML ontologies and RDF Tuples on [5]. In that paper the embeddings of the various kinds of RDF entailment in F-Logic are explored. It is shown that the embeddings of simple, RDF, and RDFS entailment, as well as a large fragment of extensional RDFS entailment, fall in the Datalog fragment of F-Logic, allowing the use of optimization techniques from the area of deductive databases for reasoning with RDF. Given the close connection between WSML and F-Logic (see [2]) this translation thus allows reasoning with RDF by WSML reasoners such as IRIS[1].

Using earlier results on the relationship between F-Logic and Description Logics (DLs), an embedding of a large fragment of extensional RDFS is defined in a tractable description logic, namely DL-Lite, allowing efficient reasoning over the ontology vocabulary with existing DL reasoners.

Furthermore, using these embeddings, RDFS can be extended with (WSML) rules and/or general axioms.

4.1 Introduction

The Resource Description Framework RDF [12], together with RDFS, constitutes the basic language for the Semantic Web. The RDF semantics specification [12] defines four increasingly expressive types of entailment, namely simple, RDF, RDFS, and extensional RDFS (eRDFS) entailment[2]. We refer to these kinds of entailment as *entailment regimes*.

The standard knowledge representation and reasoning paradigms of Description Logics (DL) [1] and Logic Programming (LP) [19], which are both based on classical first-order logic, are used on the the semantic Web (e.g. [14, 6]). However, so far, little research has been done into the formal relationships[3] between RDF and the logical languages which are being considered for the semantic Web. We show how to bridge the gap between these formalisms by demonstrating several embeddings of the RDF(S) entailment regimes in logic, and showing how RDF(S) can be extended with (LP) rules and (DL) logical axioms.

We use F-Logic [18], a syntactical extension of standard first-order logic, for our embeddings. It turns out that the attribute value construct in F-Logic is exactly equivalent to the triple construct in RDF, and the typing and subclassing constructs in F-Logic are very close to those in RDF. Additionally, F-Logic, like RDFS, has the possibility of using the same identifier as a class, an instance, or a property identifier.

4.2 Frame Logic

We follow the defintion of F-Logic[4] in [4]. For the full definition of the F-Logic semantics, we refer the reader to [18, 4].

[1] `http://iris-reasoner.org/`

[2] Note that the definition of extensional RDFS entailment is not normative.

[3] A notable exception is [3].

[4] Note that F-Logic is also often used as an extension of nonmonotonic logic programming; however, we follow the original definition which is strictly first-order.

The signature of an F-language $\mathcal{L}$ is of the form $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F}$ and $\mathcal{P}$ disjoint sets of function and predicate symbols, each with an associated arity $n \geq 0$. Terms and atomic formulas are defined in the usual way. A molecule in F-Logic is one of the following statements: (i) an *is-a* assertion of the form $C : D$, (ii) a *subclass-of* assertion of the form $C :: D$, or (iii) a data molecule of the form $C[D \twoheadrightarrow E]$, with C, D, E terms. An F-Logic molecule is *ground* if it does not contain variables.

Formulas of an F-language $\mathcal{L}$ are either atomic formulas, molecules, or compound formulas, which are constructed in the usual way from atomic formulas, molecules, and the logical connectives $\neg, \wedge, \vee, \supset$, the quantifiers $\exists, \forall$ and the auxiliary symbols $), ($. The Horn and Datalog subset of F-Logic are defined in the usual way (see e.g. [4]).

An *F-structure* is a tuple $\mathbf{I} = \langle U, \prec_U, \in_U, \mathbf{I}_F, \mathbf{I}_P, \mathbf{I}_{\twoheadrightarrow} \rangle$. Here, $\prec_U$ is an irreflexive partial order on the domain U and $\in_U$ is a binary relation over U. We write $a \preceq_U b$ when $a \prec_U b$ or $a = b$, for $a, b \in U$. For each F-structure must hold that if $a \in_U b$ and $b \preceq_U c$ then $a \in_U c$. An n-ary function symbol $f \in F$ is interpreted as a function over the domain U: $\mathbf{I}_F(f) : U^n \to U$. An n-ary predicate symbol $p \in P$ is interpreted as a relation over the domain U: $\mathbf{I}_P(p) \subseteq U^n$. $\mathbf{I}_{\twoheadrightarrow}$ associates a binary relation over U with each $k \in U$: $\mathbf{I}_{\twoheadrightarrow}(k) \subseteq U \times U$. Variable assignments are defined as usual.

Given an F-structure $\mathbf{I}$, a variable assignment B, and a term t of $\mathcal{L}$, $t^{\mathbf{I},B}$ is defined as: $x^{\mathbf{I},B} = x^B$ for variable symbol x and $t^{\mathbf{I},B} = \mathbf{I}_F(f)(t_1^{\mathbf{I},B}, \ldots, t_n^{\mathbf{I},B})$ for t of the form $f(t_1, \ldots, t_n)$.

Satisfaction of atomic formulas and molecules ϕ in $\mathbf{I}$, given the variable assignment B, denoted $(\mathbf{I}, B) \models_{\mathsf{f}} \phi$, is defined as: $(\mathbf{I}, B) \models_{\mathsf{f}} p(t_1, \ldots, t_n)$ iff $(t_1^{\mathbf{I},B}, \ldots, t_n^{\mathbf{I},B}) \in \mathbf{I}_P(p)$, $(\mathbf{I}, B) \models_{\mathsf{f}} t_1 : t_2$ iff $t_1^{\mathbf{I},B} \in_U t_2^{\mathbf{I},B}$, $(\mathbf{I}, B) \models_{\mathsf{f}} t_1 :: t_2$ iff $t_1^{\mathbf{I},B} \preceq_U t_2^{\mathbf{I},B}$, $(\mathbf{I}, B) \models_{\mathsf{f}} t_1[t_2 \twoheadrightarrow t_3]$ iff $\langle t_1^{\mathbf{I},B}, t_3^{\mathbf{I},B} \rangle \in \mathbf{I}_{\twoheadrightarrow}(t_2^{\mathbf{I},B})$, and $(\mathbf{I}, B) \models_{\mathsf{f}} t_1 = t_2$ iff $t_1^{\mathbf{I},B} = t_2^{\mathbf{I},B}$. This extends to arbitrary formulas in the usual way.

The notions of a model and of validity are defined as usual. A theory $\Phi \subseteq \mathcal{L}$ *F-entails* a formula $\phi \in \mathcal{L}$, denoted $\Phi \models_{\mathsf{f}} \phi$, iff for all F-structures $\mathbf{I}$ such that $\mathbf{I} \models_{\mathsf{f}} \Phi$, $\mathbf{I} \models_{\mathsf{f}} \phi$.

Classical first-order logic (classical FOL) is equvalent to F-Logic without molecules. *Contextual first-order logic* (contextual FOL) is classical FOL where $\mathcal{F}$ and $\mathcal{P}$ are not required to be disjoint, function and predicate symbols do not have an associated arity, and for every structure $\mathbf{I} = \langle U, \prec_U, \in_U, \mathbf{I}_F, \mathbf{I}_P, \mathbf{I}_{\twoheadrightarrow} \rangle$, $\mathbf{I}_F$ assigns a function $\mathbf{I}_F(f) : U^n \to U$ to every $f \in \mathcal{F}$ for every nonnegative integer n and $\mathbf{I}_P$ assigns a relation $\mathbf{I}_P(p) \subseteq U^n$ to every $p \in \mathcal{P}$ for every nonnegative integer n. We denote satisfaction and entailment in classical and contextual FOL using the symbols $\models$ and $\models_c$, respectively.

4.3 RDF Embedding and Extension

In this section we first consider an embedding of the RDF entailment regimes in F-Logic, after which we discuss embeddings of the eRDFS entailment regime in FOL and DL. For the definition of the different entailment regimes we refer the reader to [12].

Let S, E be RDF graphs, $x \in \{s, rdf, rdfs, erdfs\}$ be the simple (resp., RDF, RDFS, eRDFS) entailment regime, we denote entailment, i.e. S x-entails E, with $S \models_x E$.

$$
\begin{array}{l}
\Psi^{s} = \emptyset \\
\Psi^{rdf} = \Psi^{s} \cup \{tr(\langle s,p,o\rangle) \mid \langle s,p,o\rangle \text{ is an RDF axiomatic triple}\} \cup \\
\quad \{\forall x(\exists y, z(y[x \twoheadrightarrow z]) \supset x[\texttt{type} \twoheadrightarrow \texttt{Property}])\} \\
\Psi^{rdfs} = \Psi^{rdf} \cup \{tr(\langle s,p,o\rangle) \mid \langle s,p,o\rangle \text{ is an RDFS axiomatic triple}\} \cup \\
\quad \{\forall x,y,z(x[y \twoheadrightarrow z] \supset x[\texttt{type} \twoheadrightarrow \texttt{Resource}] \wedge z[\texttt{type} \twoheadrightarrow \texttt{Resource}]), \\
\quad \forall u,v,x,y(x[\texttt{domain} \twoheadrightarrow y] \wedge u[x \twoheadrightarrow v] \supset u[\texttt{type} \twoheadrightarrow y]), \\
\quad \forall u,v,x,y(x[\texttt{range} \twoheadrightarrow y] \wedge u[x \twoheadrightarrow v] \supset v[\texttt{type} \twoheadrightarrow y]), \\
\quad \forall x(x[\texttt{type} \twoheadrightarrow \texttt{Property}] \supset x[\texttt{subPropertyOf} \twoheadrightarrow x]), \\
\quad \forall x,y,z(x[\texttt{subPropertyOf} \twoheadrightarrow y] \wedge y[\texttt{subPropertyOf} \twoheadrightarrow z] \supset \\
\qquad x[\texttt{subPropertyOf} \twoheadrightarrow z]), \\
\quad \forall x,y(x[\texttt{subPropertyOf} \twoheadrightarrow y] \supset x[\texttt{type} \twoheadrightarrow \texttt{Property}] \wedge \\
\qquad y[\texttt{type} \twoheadrightarrow \texttt{Property}] \wedge \forall z_1,z_2(z_1[x \twoheadrightarrow z_2] \supset z_1[y \twoheadrightarrow z_2])), \\
\quad \forall x(x[\texttt{type} \twoheadrightarrow \texttt{Class}] \supset x[\texttt{subClassOf} \twoheadrightarrow \texttt{Resource}]), \\
\quad \forall x,y(x[\texttt{subClassOf} \twoheadrightarrow y] \supset x[\texttt{type} \twoheadrightarrow \texttt{Class}] \wedge \\
\qquad y[\texttt{type} \twoheadrightarrow \texttt{Class}] \wedge \forall z(z[\texttt{type} \twoheadrightarrow x] \supset z[\texttt{type} \twoheadrightarrow y])), \\
\quad \forall x(x[\texttt{type} \twoheadrightarrow \texttt{Class}] \supset x[\texttt{subClassOf} \twoheadrightarrow x]), \\
\quad \forall x,y,z(x[\texttt{subClassOf} \twoheadrightarrow y] \wedge y[\texttt{subClassOf} \twoheadrightarrow z] \supset \\
\qquad x[\texttt{subClassOf} \twoheadrightarrow z]), \\
\quad \forall x(x[\texttt{type} \twoheadrightarrow \texttt{ContainerMembershipProperty}] \supset \\
\qquad x[\texttt{subPropertyOf} \twoheadrightarrow \texttt{member}])\} \\
\Psi^{erdfs} = \Psi^{rdfs} \cup \{\forall x,y(\forall u,v(u[x \twoheadrightarrow v] \supset u[\texttt{type} \twoheadrightarrow y]) \supset \\
\qquad x[\texttt{domain} \twoheadrightarrow y]), \\
\quad \forall x,y(\forall u,v(u[x \twoheadrightarrow v] \supset v[\texttt{type} \twoheadrightarrow y]) \supset x[\texttt{range} \twoheadrightarrow y]), \\
\quad \forall x,y(x[\texttt{type} \twoheadrightarrow \texttt{Property}] \wedge y[\texttt{type} \twoheadrightarrow \texttt{Property}] \wedge \\
\qquad \forall u,v(u[x \twoheadrightarrow v] \supset u[y \twoheadrightarrow v]) \supset x[\texttt{subPropertyOf} \twoheadrightarrow y]), \\
\quad \forall x,y(x[\texttt{type} \twoheadrightarrow \texttt{Class}] \wedge y[\texttt{type} \twoheadrightarrow \texttt{Class}] \wedge \\
\qquad \forall u(u[\texttt{type} \twoheadrightarrow x] \supset u[\texttt{type} \twoheadrightarrow y]) \supset x[\texttt{subClassOf} \twoheadrightarrow y])\}
\end{array}
$$

Table 4.1: Axiomatization of the RDF semantics

4.3.1 Embedding RDF in F-Logic

We first define the embedding of RDF graphs in F-Logic, without taking into account the specific entailment regime, using an embedding function tr. The RDF graph is translated to a conjunction of data molecules, where URIs are constants, and blank nodes are existentially quantified variables. In the remainder, we assume that every RDF graph is finite. Given a triple $\langle s,p,o\rangle$ (resp., graph S), $bl(\langle s,p,o\rangle)$ (resp., $bl(S)$) denotes the set of blank nodes occurring in the triple (resp., graph).

Definition 1. *Let $\langle s,p,o\rangle$ be a triple and S a graph.*

$$
\begin{array}{rcl}
tr(\langle s,p,o\rangle) & = & s[p \twoheadrightarrow o] \\
tr(S) & = & \exists\, bl(S)(\bigwedge\{tr(\langle s,p,o\rangle) \mid \langle s,p,o\rangle \in S\})
\end{array}
$$

Depending on the entailment regime x, we add a set of formulas Ψ^x to the embedding of the graph. Ψ^x, defined in Table 4.1, axiomatizes the semantics of the entailment regime x.[5]

If ϕ is an F-Logic formula in prenex normal form with only existential quantifiers, then ϕ^{sk} denotes the *Skolemization* of ϕ, i.e. every existentially quantified variable is replaced by a new constant not occurring in the formula or its context (the theory in which it occurs, or any of the surrounding theories, e.g. those participating in an entailment relation). If Φ is an F-Logic theory, then Φ^{sk} denotes a skolemization of Φ.

[5]For brevity, we leave out the namespace of the RDF vocabulary; for example, `type` is short for `rdf:type`.

The following Proposition follows immediately from the definition of the translations.

Proposition 1. *Let S be an RDF graph. Then, $tr(S)^{sk} \cup \Psi^x$, with $x \in \{s, rdf, rdfs\}$, can be equivalently rewritten to a set of F-Logic Datalog formulas.*

Note that Ψ^{erdfs} cannot be equivalently rewritten to a set of Datalog formulas, due to the universal quantification in the antecedents of the implications in Ψ^{erdfs}.

We now show the correspondence between entailment in the original RDF semantics and entailment in the F-Logic embedding.

Theorem 1. *Let S, E be RDF graphs and $x \in \{s, rdf, rdfs, erdfs\}$ an entailment regime. Then,* $S \models_x E$ *if and only if* $tr(S) \cup \Psi^x \models_{\mathsf{f}} tr(E)$.

The following corollary follows immediately from Theorem 1 and the classical results about Skolemization.

Corollary 1. *Let S, E be RDF graphs and $x \in \{s, rdf, rdfs, erdfs\}$ be an entailment regime. Then,* $S \models_x E$ *if and only if* $tr(S)^{sk} \cup \Psi^x \models_{\mathsf{f}} tr(E)$.

Since, by Proposition 1, $tr(S)^{sk}$, $tr(S)^{sk} \cup \Psi^{rdf}$ and $tr(S)^{sk} \cup \Psi^{rdfs}$ are equivalent to sets of Horn formulas, this result implies that simple, RDF, and RDFS entailment can be computed using existing F-Logic rule reasoners such as FLORA-2, and Ontobroker, as well as other rule reasoners[6]. Notice that, in the corollary, $tr(E)$ can be seen as a boolean conjunctive query (i.e. a yes/no query) and the existentially quantified variables (blank nodes) in $tr(E)$ are the non-distinguished variables.

The final embedding in F-Logic we consider is a direct embedding of the extensional RDFS semantics which eliminates part of the RDFS vocabulary, yielding a set of Horn formulas. We first define the notion of *nonstandard use* of the RDFS vocabulary. Nonstandard use of the RDFS vocabulary intuitively corresponds to using the vocabulary in locations where it has not been intended, for example $\langle$`type`, `subPropertyOf`, $a\rangle$.

We say that a term t occurs in a property position if it occurs as the predicate of a triple, as the subject or object of a `subPropertyOf` triple, as the subject of a `domain` or `range` triple, or as the subject in a triple $\langle t,$ `type`, `Property`$\rangle$ or $\langle t,$ `type`, `ContainerMembershipProperty`$\rangle$. A term t occurs in a class position if it occurs as the subject or object of a `subClassOf` triple, as the object of a `domain`, `range`, or `type` triple, or as the subject of a triple $\langle t,$ `type`, `Class`$\rangle$.

Definition 2. *Let S be an RDF graph. Then S has* nonstandard use *of the RDFS vocabulary if* `type`*,* `subClassOf`*,* `domain`*,* `range`*, or* `subPropertyOf` *occurs in a non-property position in S, or* `ContainerMembershipProperty`*,* `Resource`*,* `Class`*, or* `Property` *occurs in S.*

[6]Note that the attribute value construct $a[b \twoheadrightarrow c]$ is the only construct specific to F-Logic which is used in the embeddings. Since it does not carry any specific semantics, it may be straightforwardly embedded using a ternary predicate $attval(a, b, c)$. Notice also that all rules are safe, and thus Datalog engines may be used.

Definition 3. *Let* $\langle s, p, o\rangle$ *be an RDF triple, then*

$$tr^{erdfs}(\langle s, \texttt{type}, o\rangle) = s:o,$$
$$tr^{erdfs}(\langle s, \texttt{subClassOf}, o\rangle) = \forall x(x:s \supset x:o),$$
$$tr^{erdfs}(\langle s, \texttt{subPropertyOf}, o\rangle) = \forall x, y(x[s \twoheadrightarrow y] \supset x[o \twoheadrightarrow y]),$$
$$tr^{erdfs}(\langle s, \texttt{domain}, o\rangle) = \forall x, y(x[s \twoheadrightarrow y] \supset x:o),$$
$$tr^{erdfs}(\langle s, \texttt{range}, o\rangle) = \forall x, y(x[s \twoheadrightarrow y] \supset y:o), \text{ and}$$
$$tr^{erdfs}(\langle s, p, o\rangle) = s[p \twoheadrightarrow o], \text{ otherwise.}$$

Let S *be an RDF graph. Then,*

$$tr^{erdfs}(S) = \exists\, bl(S)(\bigwedge\{tr^{erdfs}(\langle s, p, o\rangle) \mid \langle s, p, o\rangle \in S\} \cup \{tr^{erdfs}(\langle s, p, o\rangle) \mid \langle s, p, o\rangle \text{ is an RDFS axiomatic triple with no nonstandard use of the RDFS vocabulary}\})$$

Theorem 2. *Let* S, E *be RDF graphs with no nonstandard use of the RDFS vocabulary. Then,*

$$S \models_{erdfs} E \text{ iff } tr^{erdfs}(S) \models_{\mathsf{f}} tr^{erdfs}(E).$$

Furthermore, $(tr^{erdfs}(S))^{sk}$ *is a conjunction of F-Logic Datalog formulas. If, additionally,* E *does not contain* `subClassOf`*,* `domain`*,* `range`*, or* `subPropertyOf`*, then* $tr^{erdfs}(E)$ *is a conjunction of atomic molecules with an existential prefix, and*

$$S \models_{erdfs} E \text{ iff } (tr^{erdfs}(S))^{sk} \models_{\mathsf{f}} tr^{erdfs}(E).$$

Since $(tr^{erdfs}(S))^{sk}$ is a set of Datalog formulas, we have that, if the RDF graphs fulfills certain (natural) conditions, query answering techniques from the area of deductive databases can be used for checking eRDFS entailment.

4.3.2 Embedding Extensional RDFS in First-Order Logic

An F-Logic theory Φ is *translatable* to contextual FOL if it has no :: molecules and for molecules of the forms $t_1[t_2 \twoheadrightarrow t_3]$ and $t_1:t_2$ holds that t_2 is a constant symbol.

Let Φ be an F-Logic theory which is translatable to contextual FOL, then $(\Phi)^{FO}$ is the contextual FOL theory obtained from Φ by:
– replacing every $t_1[t_2 \twoheadrightarrow t_3]$ with $t_2(t_1, t_3)$, and
– replacing every $t_1:t_2$ with $t_2(t_1)$.

The following is a straightforward generalization of a result in [4].

Proposition 2. *Let* Φ *(resp.,* ϕ*) be an equality-free F-Logic theory (resp., formula) which is translatable to contextual FOL. Then,* $\Phi \models_{\mathsf{f}} \phi$ *iff* $(\Phi)^{FO} \models_c (\phi)^{FO}$.

An RDF graph S is a *non-higher-order* RDF graph if S does not contain blank nodes in class or property positions and does not contain nonstandard use of the RDFS vocabulary. A non-higher order RDF graph S is a *classical* RDF graph if the sets of URIs occurring in class and property positions in S (and its context, e.g. entailing or entailed graph) are mutually disjoint, and disjoint with the sets of all URIs not occurring in class or property positions in S (and its context).

Theorem 3. *Let* S, E *be non-higher-order (resp., classical) RDF graphs. Then,* $(tr^{erdfs}(S))^{FO}$, $(tr^{erdfs}(E))^{FO}$ *are theories of contextual (resp., classical) FOL and* $S \models_{erdfs} E$ *iff* $(tr^{erdfs}(S))^{FO} \models_c (tr^{erdfs}(E))^{FO}$ *(resp.,* $(tr^{erdfs}(S))^{FO} \models (tr^{erdfs}(E))^{FO}$*).*

4.4 Complexity of RDF

The complexity of simple and RDFS entailment is well known, and the complexity of RDF and extensional RDFS entailment follow immediately. Note that, although the set of axiomatic triples is infinite, only a finite subset needs to be taken into account when checking the entailment.

Proposition 3 ([11, 16, 3])**.** *Let S, E be RDF graphs, then the problems $S \models_s E$, $S \models_{rdf} E$, and $S \models_{rdfs} E$ are NP-complete in the combined size of S and E, and polynomial in the size of S. If E is ground, then the respective problems are in P. Additionally, the problem $S \models_{erdfs} E$ is NP-hard.*

From the embedding in F-Logic, together with the complexity of nonrecursive Datalog, we obtained the following novel characterization of the complexity of simple and RDF entailment.

Proposition 4. *Let S, E be RDF graphs. Then, the problems $S \models_s E$ and $S \models_{rdf} E$ are in LogSpace with respect to the size of S, and with respect to the combined size of the graphs if E is ground.*

From the correspondence between FOL and Description Logics and earlier complexity results for the Description Logic $DL\text{-}Lite_{\mathcal{R}}$ [8] we obtain the following results.

Theorem 4. *Let S, E be RDF graphs with no nonstandard use of the RDFS vocabulary. Then, the problem of deciding $S \models_{erdfs} E$ is NP-complete in the size of the graphs, and polynomial if E is ground.*

Regime	Restrictions on S	Restrictions on E	Complexity
$x \in \{s, rdf, rdfs\}$	none	none	NP-complete
$x \in \{s, rdf\}$	none	ground	LogSpace
$x \in \{rdfs\}$	none	ground	P
$x \in \{erdfs\}$	none	none	NP-hard
$x \in \{erdfs\}$	no nonst. RDFS	no nonst. RDFS	NP-complete
$x \in \{erdfs\}$	no nonst. RDFS	ground, no nonst. RDFS	P

Table 4.2: Complexity of Entailment $S \models_x E$, measured in the combined size of S and E

Table 4.2 summarizes the complexity of the different entailment regimes; "No nonst. RDFS" stands for "no nonstandard use of the RDFS vocabulary". The results in the first and third line of the Table were obtained in [11, 3, 16]. To the best of our knowledge, the other results in the table are novel.

4.5 RDF Extensions

In this section we consider extensions of RDF graphs with logical rules and general theories.

Definition 4. *An F-Logic extended RDF graph is a tuple* $eS = \langle S, \Phi, x\rangle$*, with* S *an RDF graph,* Φ *an F-Logic theory, and* $x \in \{s, rdf, rdfs, erdfs\}$ *an entailment regime.*

eS is satisfiable (resp., valid) if $tr(S) \cup \Psi^x \cup \Phi$ *is satisfiable (resp., valid), and eS entails an F-Logic formula* ϕ *(resp., RDF graph E), denoted* $eS \models \phi$ *(resp.,* $eS \models E$*), if* $tr(S) \cup \Psi^x \cup \Phi \models_{\mathsf{f}} \phi$ *(resp.,* $tr(S) \cup \Psi^x \cup \Phi \models_{\mathsf{f}} tr(E)$*).*

The following proposition follows immediately from Theorem 1.

Proposition 5. *Let* S, E *be RDF graphs, and* $x \in \{s, rdf, rdfs, erdfs\}$ *an entailment regime. Then,*

$$\langle S, \emptyset, x\rangle \models E \;\; \textit{iff} \;\; S \models_x E.$$

Considering such F-Logic extended RDF graphs, there is a discrepancy between the RDF and F-Logic constructs used for asserting class membership ($a[\texttt{type} \twoheadrightarrow C]$ vs. $a\!:\!C$) and asserting the subclass relation ($A[\texttt{subClassOf} \twoheadrightarrow B]$ vs. $A\!::\!B$). Therefore, the interaction between the RDF graph and the F-Logic theory might not be as expected.

Consider, for example, the RDF graph $S = \{\langle A, \texttt{subClassOf}, B\rangle\}$ and the F-Logic theory $\Phi = \{a\!:\!A\}$. Consider now the F-Logic extended RDF graph $T = \langle S, \Phi, rdfs\rangle$. One might intuitively expect $T \models a\!:\!B$. This is, however, not the case, because of the lack of interaction between the RDFS vocabulary and the F-Logic language constructs.

We overcome this limitation by using the so-called *RDF interaction axioms (RIA)*:

$$\begin{aligned}\Psi^{RIA} = \{ \;\; & \forall x, y(x[\texttt{type} \twoheadrightarrow y] \supset x\!:\!y),\\ & \forall x, y(x[\texttt{subClassOf} \twoheadrightarrow y] \supset x\!::\!y)\}.\end{aligned}$$

Definition 5. *An F-Logic extended RDF graph eS is RIA-satisfiable (resp., valid) if* $tr(S) \cup \Psi^x \cup \Phi \cup \Psi^{RIA}$ *is satisfiable (resp., valid), and eS RIA-entails an F-Logic formula* ϕ *(resp., RDF graph E), denoted* $eS \models_{RIA} \phi$ *(resp.,* $eS \models_{RIA} E$*), if* $tr(S) \cup \Psi^x \cup \Phi \cup \Psi^{RIA} \models_{\mathsf{f}} \phi$ *(resp.,* $tr(S) \cup \Psi^x \cup \Phi \cup \Psi^{RIA} \models_{\mathsf{f}} tr(E)$*).*

The following proposition follows from Proposition 5 and the structure of the RIA axioms.

Proposition 6. *Let* S, E *be RDF graphs, and* $x \in \{s, rdf, rdfs, erdfs\}$ *an entailment regime. Then,*

$$\langle S, \emptyset, x\rangle \models_{RIA} E \;\; \textit{iff} \;\; S \models_x E.$$

Theorem 3 sanctions the extension of a subset of eRDFS with DL or FOL axioms:

Definition 6. *A contextual (resp., classical) FOL-extended RDF graph is a tuple* $\langle S, \Phi\rangle$ *where S is a non-higher-order (resp., classical) RDF graph, and* Φ *is a contextual (resp., classical) FOL theory.*

$\langle S, \Phi\rangle$ is satisfiable (resp., valid) if $(tr^{erdfs}(S))^{FO} \cup \Phi$ *is satisfiable (resp., valid).*

A contextual FOL extended RDF graph $\langle S, \Phi\rangle$ entails a contextual FOL formula ϕ (resp., non-higher-order RDF graph E) if $(tr^{erdfs}(S))^{FO} \cup \Phi \models_c \phi$ *(resp.,* $(tr^{erdfs}(S))^{FO} \cup \Phi \models_c (tr^{erdfs}(E))^{FO}$*).*

A classical FOL extended RDF graph $\langle S, \Phi\rangle$ entails a classical FOL formula ϕ (resp., classical RDF graph E) if $(tr^{erdfs}(S))^{FO} \cup \Phi \models \phi$ *(resp.,* $(tr^{erdfs}(S))^{FO} \cup \Phi \models (tr^{erdfs}(E))^{FO}$*).*

Proposition 7. *Let S, E be non-higher-order (resp., classical) RDF graphs. Then, $\langle S, \emptyset\rangle \models_c E$ (resp., $\langle S, \emptyset\rangle \models E$) iff $S \models_{erdfs} E$.*

The following results about the complexity of reasoning with extended RDF graphs follow immediately from the complexity results obtained in the previous section, and the complexity of the considered extensions.

We first consider RDF graphs extended with F-Logic Datalog rules.

Theorem 5. *Given an F-Logic extended RDF graph $eS = \langle S, \Phi, x\rangle$, with Φ a set of F-Logic Datalog rules, and a ground atom or molecule α, then*

- *if $x \in \{s, rdf, rdfs\}$, the problem of deciding $eS \models_{RIA} \alpha$ is polynomial, and*
- *if $x = erdfs$ and S has no nonstandard use of the RDFS vocabulary, the problem of deciding $eS \models \alpha$ is polynomial.*

We now consider RDF graphs, under the eRDFS entailment regime, extended with *DL-Lite*$_\mathcal{R}$ axioms. *DL-Lite*$_\mathcal{R}$ [8] is a Description Logic which largely subsumes the expressiveness of extensional RDFS, and for which most of the reasoning tasks are tractable (i.e. polynomial). We consider *DL-Lite*$_\mathcal{R}$ as defined in [8], and refer to this as *classical DL-Lite*$_\mathcal{R}$. We also consider a variant of *DL-Lite*$_\mathcal{R}$ in which the sets of class, rule, and individual identifiers are not disjoint, and refer to this *contextual DL-Lite*$_\mathcal{R}$ (cf. contextual FOL).

Theorem 6. *Let $eS = \langle S, \Phi\rangle$ be a contextual (resp., classical) FOL extended RDF graph, such that S is ground, and Φ is the FOL equivalent of a contextual DL-Lite$_\mathcal{R}$ knowledge base $\mathcal{K}$, and let Φ' be the FOL equivalent of a contextual (resp., classical) DL-Lite$_\mathcal{R}$ knowledge base $\mathcal{K}'$. Then, the problem $eS \models_c \Phi'$ (resp., $eS \models \Phi'$) is polynomial.*

5 Entailment regimes in TRREE rule language

Triple Reasoning and Rule Entailment Engine (TRREE) is a native RDF rule-entailment engine, which is currently integrated in the TripCom storage implementation optimized for high-performance reasoning [26]. A forward-chaining algorithm is used to compute the inference closure and materialize all implicit statements. Different semantic levels are supported with the assignement of configurable rule sets. In this chapter we will overview the main aspects to implement reasoning in a triplespace using the TRREE engine. First, the expressivity of the R-entailment rules and their simliarity to TRREE rule language in terms of semantics and syntax is covered, later extension of the existing rule set is proposed toward the support of Web Service Modeling Language[1] (WSML) and rules support.

5.1 R-entailment rules

In [15], Horst defines RDFS extensions towards rule support and specified a fragment of OWL more expressive than DLP, which we name OWL-Horst. The notion of R-entailment is used to describe a set of entailment rules to transform a source target RDF graph. Compared to the D-entailment defined by Hayes, [12] the R-entailment is more general in defining RDFS semantics and is based on generalized RDF graphs, where blank nodes are allowed for predicates (a possibility disallowed in RDF). Rules without premises are used to declare axiomatic statements and rules without consequences imply inconsistency.

The D-entailment rules in [12] are extended in two steps:

- D* adds entailment for literal datatypes.
- pD* adds rules for partial OWL support.

The partial OWL support includes the primitives: FunctionalProperty, InverseFunctionalProperty, SymmetricProperty, TransitiveProperty, equivalentClass, eqivalentProperty, sameAs, hasValue, inverseOf, someValueFrom, allValuesFrom, differentFrom and disjointWith. The former two primitives are implemented through inconsistency rules in the case of P-clashes, [15]. Also, there is only one directional support (iff-semantics) of someValueFrom and allValuesFrom primitives.

OWL-Horst offers several distinctive characteristics:

- No constrains over the RDFS semantics exist. It is backward compatible to RDFS in contrast to other existing fragments like OWL-Lite or OWL-DLP.
- Unlike SWRL [13], R-entailment provides extensions without DL-related constraints.
- Lower computational complexity than SWRL and approaches to combine DL with rules.

[1] http://www.wsmo.org/wsml/

5.2 TRREE rule language

The TRREE rule language is formal language used to define the entailment rule sets. In this section we will cover the main aspects of the syntax and the used semantics. Formally a rule set is composed by three sections Prefices, Axioms and Rules enclosed in curly braces .

- Prefices is an optional section used to define commonly used namespaces. Each prefix/namespace pair is specified on a separate line.
- Axioms is an optional section to set axiomatic triples to be asserted by default in repository. The triples are used to define meta-level primitives like rdf:type or rdfs:Class.
- Rules is a mandatory section that encloses all the rule definition. Each rule is composed by Id, one or more premises and one or more corollaries. The premises and corollaries are composed by subject, predicate and object components where the values can use a variable (single letter), a full URI or short name with prefix. Constraints could be associated next to each premises to enforce most often performance optimizations. The left-hand side argument of constrains must be a variable, while the right-hand can be either variable, full URI or short name.

The TRREE rule format and the semantics enforced is analogous to R-entailment with several differences:

- Free variables in the head, which are not bound in the body are interpreted as blank node identifiers.
- The variable inequality constraint can be specified in the triple patterns or the rule body.
- Operator [cut] is associated with the rule premises.
- No support of inconsistency rules is provided.
- Special axiom construct is provided for the rules with empty bodies.

5.3 WSML extensions

This section presents possible strategies toward extending the existing TRREE reasoning/query engine in the direction of WSML and rule support. TRREE rule language as a forward-chaining engine for positive rules over triples can cover two fragments WSML Core, as a compatible logical language to the DLP fragment of Description Horn Logics. Later, on we estimate the theoretical limits for expansion of the rule language and support of a subset of WSML-Flight, which we call WSML-Flight–.

5.3.1 F-Logic and WSML atoms

We first review the kind of rules which are conceptually allowed in WSML, namely, WSML-Core and -Flight which may be seen as syntactic variants of F-Logic programming. Next, we will check how this corresponds to WSML/RDF and then we will define the subset of F-Logic programming rules in WSML-Core, which can be transformed to constructs handled by TRREE. We briefly introduce F-Logic here (a longer introduction of the language is available in the next section), as it is the semantic basis of WSML Core. An F-Logic program is a set of rules of the form:

```
h :- a1, a2, . . ., an, naf b1, naf b2, . . ., naf bn .
```

Where each of

```
h, a1, . . . , am, b1, . . . , bn
```

is an atom of one of the following forms: o1 : o2 (class membership), o1 :: o2 (explicit subconcept declaration), o1[o2 → o3], o1[o2 ⟹ o3]. We additionally assume safety of rules, ie. each variable in h and b^1, ... , b^n, also appears in one of the a^1, ... , a^m, a syntactic restriction which is made in both WSML's Core and Flight variants.

Note that ':-' corresponds to "wsml:impliedBy" in WSML's axiom syntax and the F-logic atoms correspond to the following atoms in WSML: o1 memberOf o2 (class membership), o1 subConceptOf o2 (explicit subconcept declaration), o1[o2 hasValue o3], o1[o2 ofType o3]. Additionally, WSML conceptual syntax allows atoms of the following kind: o1[o2 impliesType o3]

5.3.2 WSML/RDF

For every WSML/F-Logic atom:

```
o1 memberOf o2,
o1 subConceptOf o2,
o1[o2 hasValue o3],
o1[o2 ofType o3],
o1[o2 impliesType o3]
```

There are triple representations of WSML/RDF[9], namely:

```
// o1 memberOf o2
o1 rdf:type o2 .
// o1 subConceptOf o2 .
o1 rdfs:subClassOf o2 .
// o1[o2 hasValue o3] .
o1 o2 o3 .
// o1[o2 ofType o3] .
o1 rdf:type wsml:Concept;
part-whole:hasPart directly :x .
:x rdf:type wsmlrdf:AttributeDefinition ;
wsmlrdf:ofTypeConcept o1 ;
wsmlrdf:forAttribute o2 ;
```

```
wsmlrdf:ofType o3 .
// o1[o2 impliesType o3]
o1 rdf:type wsml:Concept ;
part-whole:hasPart directly :y .
:y rdf:type wsml:AttributeDefinition ;
wsmlrdf:impliesTypeConcept o1 ;
wsmlrdf:impliesTypeAttribute o2 ;
wsmlrdf:impliesType o3 .
```

In the snippet above :x and :y are new blank nodes. The full translation between WSML and RDF languages is specified in [9].

5.3.3 WSML logical expressions

WSML logical expressions in WSML-Core and WSML-Flight allow the definition of rules and integrity constraints. Note here, that the allowed syntax of WSML-Flight goes beyond the rules we have mentioned in the Section 4 above in that it allows all rules which can be transformed to rules of the above form by monotonic Lloyd-Topor transformations, see Section 5.6. "WSML-Flight Logical Expression Syntax" of [24]. WSML-Core logical expressions are more restricted in that after applying the Lloyd-Topor transformation, only positive rules (no "naf" atoms) with the following restrictions are allowed.

- Var1[attr hasValue Var2] impliedBy Var1[attr hasValue Var3] and Var3[p hasValue Var2]. Such a rule corresponds semantically to an OWL transitive role statement.
- Var1[attr hasValue Var2] impliedBy Var2[attr hasValue Var1]. Such a rule corresponds semantically to an OWL symmetric role statement.
- Var1[attr1 hasValue Var2] impliedBy Var1[attr2 hasValue Var2]. Such a rule corresponds semantically to an OWL/RDFS sub-property statement.
- Var1[attr1 hasValue Var2] equivalent Var2[attr2 hasValue Var1]. Such a rule corresponds semantically to an OWL inverse role statement.

Moreover, WSML-Core allows rules which, after application of Lloyd- Topor result in safe rules such that head atoms have the form VarRoot memberOf o2 and all body atoms are either of the form Var memberOf o2 or Vari[attr hasValue Varj] or Vari[attr hasValue Value or Object], such that the graph built by adding an edge from Vari to Varj , labeled with attr occurring builds a tree with root variable VarRoot. Such rules are translatable in a DL TBox axiom:

$$Expr \subseteq o2$$

where Expr is a DL expression consisting of a conjunction formed by class names and nested existential property restrictions or inverse property restrictions.

Instead of giving a detailed translation here, we give an example:

```
?X memberOf c :- ?X memberOf d1, ?X[attr1 hasValue ?Y],
 ?Z[attr2 hasValue ?Y],
 ?Z memberOf d2,
 ?Z memberOf d3.
```

will be translated to:

$$\geq 1 attr1 \ni attr2^{-}.(d2 \sqcap d3) \subseteq c$$

To create this expression by annotating in the variable tree all variables (nodes) with a list of attached concepts, given by the respective memberOf statements. If this list is empty, you put a minCardinality restriction, otherwise, you put an existential property restriction over the intersection of all classes in the list, traversing the attribute labeled tree, over (depending the direction of the edge) the respective property or its inverse. We remark again, that the tree structure of the variable graph is mandatory, otherwise the translation does not work, eg. the following is not a valid WSML-Core rule:

```
?X memberOf c :- ?X memberOf d1, ?X[attr1 hasValue ?Y],
?Y[attr2 hasValue ?Y], ?Y memberOf d2,
?Y memberOf d3.
```

5.3.4 Beyond WSML-Core

An analysis of the way TRREE makes obvious, that nonmonotonic features of WSML Flight, the next variant of WSML above WSML-Core in the WSML language stack, make a direct adoption of WSML-Flight infeasible in combination with the approach of materialization used in TRREE. So, rules with negation as failure in the body must be forbidden in order to obtain a TRREE compatible fragment of WSML-Flight. Positive WSML-Flight rules, can be translated to TREE rules after Lloyd Topor transformation, using the correspondences in Section 5.3.2 straightforwardly. However, this makes dynamic addition and recursive processing of TRREE rules necessary i.e., whenever a new rule is inserted, the new TRREE rule has to be matched against the triples already in the repository, and more rules might recursively fire as usual in the processing of TRREE when adding new consequences from these WSML rules. A simple translation to OWL axioms which wouldn't make the insertion of TRREE rules necessary, is only possible for WSML-Core compatible rules as already outlined above, i.e., a positive WSML Flight rule:

```
h :- a1, a2, . . ., an.
```

would translate to a TRREE rule:

```
WSML/RDF version of a1
WSML/RDF version of a2
...
WSML/RDF version of an
------------------------
WSML/RDF version of h
```

Integrity constraints, denoted syntactically by rules with WSML axioms with an empty head and the symbol "!-" instead of ":-", being somewhat external to the logical inferences made in WSML-Flight, could be modeled by special TRREE rules which allow to infer inconsistency:

```
!- a1 and a2 and . . . and an.
```

would translate to:

```
WSML/RDF version of a1
WSML/RDF version of a2
...
WSML/RDF version of an
------------------------
err err err .
```

where (err err err) is a special triple denoting inconsistency.

Negative literals can not be safely allowed in the WSML-Flight fragment coverable by TRREE, since, if negation as failure being checked dynamically during insertion of a triple set, would possibly fire constraints depending on the order you insert triples. Here, a small example, assuming that TRREE had the possibility to check non-entailment of triples in rule bodies, marked with naf. Take the following constraint in WSML-Flight:

```
!- ?X memberOf concept1 and ?X[attr1 hasValue ?Y] and naf ?Y memberOf concept2.
```

Supposedly, a corresponding TRREE inconsistency check rule could look as follows:

```
?X rdf:type concept1 .
?X attr1 hasValue ?Y .
naf ?Y rdf:type concept2 .
------------------------
err err err .
```

However, take the following instance definitions:

```
o1 memberOf concept1 .
o1[attr1 hasValue o2] .
o2 memberOf concept2 .
```

Obviously, if you check the constraint before:

```
o2 memberOf concept2 .
```

was inserted, you be able to fire the constraint rule and infer err, which was not the case if you parsed the instance triples in a different order. This makes obvious, that constraint checking with naf is incompatible with TRREE's inference strategy of firing rules "per inserted triple". In case you allow these nonmonotonic constraints, a strategy of treating them as queries which are evaluated on demand or on "bulk" seems more advisable. Note that, the constraint above is logically equivalent to the declaration concept concept1 attr1 ofType concept1 in WSML-Flight conceptual syntax. Thus, for constraints involving naf also applies for WSML's ofType declarations.

6 Evaluation of triplespace knowledge representation

To evaluate the needs of the TripCom use case scenarios with respect to the knowledge representation model, use case ontologies and sample queries are considered.

In WP8A, a digital content marketplace is implemented, where business transactions are maintained between Content and Service Providers. The ontological definitions are focused on the description of actors, contents, transactions, auction management and customers. These definitions are complete by using RDFS semantics only, however in some cases using more expressive semantics or rule languages is needed or handy to provide a better knowledge capture in this scenario:

- Inverse relationships can be very useful to improve the representation of several relationships in this use case. For example, we have a relationship Actor *participatesIn* Film. It will be very convenient to define an inverse relationship which indicates in which Film an Actor has participated, to provide further information for useful queries from a customer point of view.

- Rules are needed in this scenario in order to capture some complex but useful relationships. As an example, it might be very useful for a customers to look for similar films to one he really liked. This similarity can be modelled through rules, defining three levels of similarity:

  1. Films are tightly related if they are films from the same genre, and share director and some actors (like The Godfather and The Godfather II. Both are crime films, directed by Francis Ford Coppola and share many actors).
  2. Films are quite related if they are films from the same genre and share some actor (like The Godfather and Scarface. Both are crime films starred by Al Pacino).
  3. Films are loosely related if the belong to the same genre (i.e: like The Lion King and Shreck, which are kid films).

- Filter queries can be useful for the implementation of black lists, which define a list of non-desired providers to trade with (because a previous contracted service from this provider was not satisfactory). In this context, it can be very useful to ask questions of this type: *Give me the name, of films that can be provided by any provider from Spain except X*. Where this X means a provider belonging to the Service Provider's black list.

  The open and close world assumptions are two different approaches to interpret the existing data and generate knowledge. The close world assumption (CWA) is scenario when it is expected that the data model contains all available information at one place and everything is considered false unless otherwise proven. A typical example of CWA based application are the database management systems in which query returning no results is typically interpreted as false. The open world assumption (OWA) contrary to CWA assumes incomplete nature of the information. As consequence it is optimized for scenarios where the data is intentionally underspecified, thus allow other users to reuse and further extend it.

WP8A provides 13 SPARQL queries, which can be categorized as follows:

- Projection (graph matching) 6
- Select distinct 1
- With limit 1
- Filter on string 1
- Filter on dateTime 1
- Filter on bound, optional 1
- Union 1
- Negation 1

SPARQL query language is used in WP8a to interact with the data model. Thus, we have to implement a native RDF data model to support efficient SPARQL queries. The best choice to realize the use case requirements is to use the TRREEAdapter implementation distributed in ORDI framework, which supports SPARQL and limited semantic expressivity. TRREE engine can be initialized with the OWL-Horst rule set to support inverse relationships using owl:inverseOf property. Rules that are needed in the scenario to capture relationships of type "tightly related", "quite related" or "loosely related" films have to be implemented with TRREE rule language in a WP8a specific ruleset as follows:

```
x rdf:type a .
y rdf:type a . [Constraint y != x]
a rdf:type daml:FilmAsset [Constraint a != <daml:FilmAsset>]
------------------------
x daml:relatedFilm y .
```

The property daml:relatesFilm may be further used to specified "quite related" and "tightly related":

```
x daml:relatedFilm y . [Constraint x != y]
x daml:hasActor a .
y daml:hasActor a .
------------------------
x daml:quiteRelatedFilm y .
```

Negation queries are used in the sense of naf instead of not (!) operator. To express negation queries in SPARQL, OPTIONAL and FILTER clauses have to be used:

```
SELECT ?name
WHERE { ?x daml:name ?name.
OPTIONAL { ?x daml:blackListedBy ?y }
FILTER (!bound(?y)) }
```

In WP8B, the EPS ontologies are modeled in WSML-Core, which uses global attributes to be fully compatible with RDFS semantics. Thus, any information currently managed is expressible in RDFS. On the basis of these ontologies, sample data could be defined to represent instances of knowledge in the scenario domain. The WP8b scenario model patient summaries with specific generic concepts like `concept Person` linked to `concept PatientSummary` using the `hasPatient` attribute. It should be noted that the concept Person is also used to represent other actors (such as practitioners or relatives) when it's linked via the `hasGeneralPractitioner` attribute. The following listing reports a simplified WSML representation of the previous description:

```
concept PatientSummary
  hasPatient impliesType Person
  hasGeneralPractitioner impliesType Person

concept Person
```

On the other hand, another different ontology based on the HL7 CDA standard[1] models patient and general practitioner, with specific concepts as follow:

```
concept Patient

concept Author
```

In order to cope with such heterogeneity, the `concept Person` of the EPS ontology has to be mediated with:

- the `concept Patient` of the CDA ontology when the concept Person is linked with the PatientSummary using the `hasPatient` attribute;
- the `concept Author` of the CDA ontology when the concept Person is linked with the PatientSummary using the `hasGeneralPractitioner` attribute.

The RDFS semantics does not allow to specify such two semantic mappings. Rules are needed to distinguish between the two meanings of the concept Person. Thus, the EPS scenario needs a middleware able to cope with heterogeneity between different formats. A solution may be to semantically mediate between the different ontologies that model various eHealth standards. The use of semantic mediation requires a language able to support the definition of axioms and rules. This work is beyond the scope of the current deliverable and should be addressed by the Mediation Manager [22].

We see that this scenario generally requires projection queries with graph matching. A few cases show the need for datatype operations, DISTINCT, LIMIT and UNION, all of which are supported in SPARQL. One case needed negation (a film in which person 1 acted and person 2 did not act). SPARQL by default does not support negation.

WP8B provided 11 queries, which can be categorized as follows:

- Projection (graph matching) 1
- Select distinct 4

[1]Clinical Document Architecture: a standard format for exchanging generic clinical data.

- Order by, limit 2
- Completeness 1
- Filter on dateTime 3
- Union 2
- Boolean 2

WP8b poses challenge concerning the scale of the information. TRREEAdapter implementation provides completeness in the context of a single ORDI data model. The task of distributed reasoning is out of scope of the project, however it has been addressed by the WP8b use case by generating region specific data sets. Every EPS record is associated with a specific region that is responsible to manage the information. Hence, every patient is associated with a general practioner operating in his area. Still, the information could be distributed to multiple spaces like a patient to be associated with general practioner in another region, but this will lead to lack of completenes with respect to reasoning.

7 Conclusion

This deliverable provides a first step towards Triple Space implementations that support more expressive Semantic Web languages than the fundamental RDF(S) language. Through an evaluation of the current specifications of the TripCom project in respect to tuple and space models we find that in many cases, where application rely on the publication of more complex data constructs, simple RDF graphs are not expressive enough. On the one hand construct demand more sophisticated reasoning support, while on the other modeling them as triples results in large graphs that mix the actual information and the related structural metadata. The same problem arises when using more expressive formalism with an RDF serialization which is the case both OWL and WSML. Further to the problem of mixing information and metadata, this adds the issue of loss of the more expressive axiomatic expressions that must be approximated with RDF constructs.

A second result of this deliverable shows that Logic Programming, i.e. Datalog, F-Logic (WSML-Flight), or the presented TRREE rule language, are better suited to match the requirements Description Logic, as LP-based approaches are more scalable and the reasoner support better optimized. Hence, we propose to extend the TripCom knowledge models towards WSML-Flight rather than towards the OWL language family.

In line with this conclusion the deliverable presented two alternative approaches for extending the Triple Space knowledge representation formalisms.

TRREE . The TRREE rule language is a Datalog-like language that is however slightly less expressive than Datalog. Chapter 5 shows how the TRREE language can be used to implement WSML-Core (the least expressive WSML language) and how it potentially could also be applied to implement nearly WSML-Flight. A respective TRREE rule engine is shipped with ORDI and supports a SPARQL query language interface.

IRIS . IRIS is a datalog reasoner, hence in order to be compatible with the existing Triple Space specifications, the complete RDF data model has to be converted to rules and facts. This procedures turned out to be less scalable as the first one, and moreover there is yet a lack of persistency support. On the other hand, IRIS as dedicated Logic Programming reasoner has better support for rule following and query resolution algorithms.

By help of a use case-driven evaluation of the two approaches we conclude that the solution that best matches the goals and current specifications of TripCom, and that best supports the requirements of the use cases with respect to expressivity and semantics is the former, the TRREE-based one. The main reasons for that decision are scalability inherent support for SPARQL, the standardized query langauge for RDF, and hence the Semantic Web. Still, as stated above, in scenarios where user-defined rules have more importance, it is advisable to use WSML-Flight and the IRIS-based solution. According to the current needs the former approach was implemented in course of this work, while the latter was not.

References

[1] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] Jos de Bruijn, Dieter Fensel, Uwe Keller, Michael Kifer Holger Lausen, Reto Krummenacher, Axel Polleres, and Livia Predoiu. Web service modeling language (WSML). W3C Member Submission 3 June 2005, 2005.

[3] Jos de Bruijn, Enrico Franconi, and Sergio Tessaris. Logical reconstruction of normative RDF. In *OWL: Experiences and Directions Workshop (OWLED-2005)*, Galway, Ireland, November 2005.

[4] Jos de Bruijn and Stijn Heymans. Translating ontologies from predicate-based to frame-based languages. In *Proceedings of the Second International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML-2006)*, Athens, Georgia, USA, November 10-11 2006. IEEE.

[5] Jos de Bruijn and Stijn Heymans. RDF and logic: Reasoning and extension. In *Proceedings of the 6th International Workshop on Web Semantics (WebS 2007), in conjunction with the 18th International Conference on Database and Expert Systems Applications (DEXA 2007)*, Regensburg, Germany, September 3–7 2007. IEEE Computer Society Press.

[6] Jos de Bruijn, Holger Lausen, Axel Polleres, and Dieter Fensel. The web service modeling language: An overview. In *Proceedings of the 3rd European Semantic Web Conference (ESWC2006)*, number 4011 in Lecture Notes in Computer Science, pages 590–604, Budva, Montenegro, June 2006. Springer-Verlag.

[7] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: Tractable description logics for ontologies. In *AAAI-05*, pages 602–607, 2005.

[8] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.

[9] Jos de Bruijn. D32v0.1 wsml/rdf, 2007.

[10] B. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic, 2003.

[11] Claudio Gutierrez, Carlos Hurtado, and Alberto O. Mendelzon. Foundations of semantic web databases. In *ACM Symposium on Principles of Database Systems (PODS)*, 2004.

[12] Patrick Hayes. RDF semantics. Technical report, W3C, 2004. W3C Recommendation 10 February 2004.

[13] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A semantic web rule language combining OWL and RuleML. Member submission 21 may 2004, W3C, 2004.

[14] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[15] Herman J. ter Horst. Combining rdf and part of owl with rules: Semantics, decidability, complexity. In *Proceedings of the 4th International Semantic Web Conference (ISWC 2005)*, Galway, Ireland, 2005.

[16] Herman J. ter Horst. Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *Journal of Web Semantics*, 3(2–3):79–115, 2005.

[17] Deepali Khushraj, Ora Lassila, and Timothy W. Finin. sTuples: Semantic Tuple Spaces. In *MobiQuitous*, pages 268–277, 2004.

[18] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *JACM*, 42(4):741–843, 1995.

[19] John W. Lloyd. *Foundations of Logic Programming (2nd edition)*. Springer-Verlag, 1987.

[20] Vassil Momtchev and Atanas Kiryakov. Specification of the store architecture and interfaces. TripCom Deliverable D1.2, 2006.

[21] Boris Motik, Ian Horrocks, Riccardo Rosati, and Ulrike Sattler. Can OWL and Logic Programming Live Together Happily Ever After? In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *Proc. of the 5th Int. Semantic Web Conference (ISWC 2006)*, volume 4273 of *LNCS*, pages 501–514, Athens, GA, USA, November 5–9 2006. Springer.

[22] Martin Murth, Gerson Joskowicz, and eva Kuhn et al. Triple space reference architecture. TripCom Deliverable D6.2, 2007.

[23] Lyndon Nixon, Elena Paslaru Bontas Simperl, Reto Krummenacher, Francisco Martin-Recuerdaa, Martin Murth, Geri Joskowicz, and eva Kuhn. Specification and implementation of a semantic linda model. TripCom Deliverable D3.1, 2007.

[24] Eyal Oren, Holger Lausen, and Jos de Bruijn. Languages for WSMO. Deliverable D16.0, WSMO, http://www.wsmo.org/, 2004. Available from http://www.wsmo.org/2004/d16/d16.0/v0.2/.

[25] Janne Saarela, Tommi Koivula, Lyndon Nixon, Axel Polleres, and David de Francisco. State of the art and triple space specific requirements of semantic query languages. TripCom Deliverable D3.2, 2007.

[26] Brahmananda Sapkota, Vassil Momtchev, and Omair Shafiq. High-performance storage implementation. TripCom Deliverable D1.3, 2008.

[27] Elena Simperl, Lyndon Nixon, Reto Krummenacher, Vassil Momtchev, and Henar Munoz. Representing rdf semantics in tuples. TripCom Deliverable D2.1, 2007.

[28] Sun Microsystems, Inc. *JavaSpace Specification, Revision 0.4*, 1997.