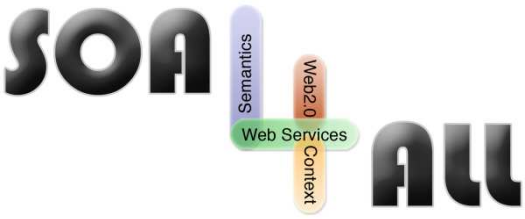

Project Number: **215219**

Project Acronym: **SOA4All**

Project Title: **Service Oriented Architectures for All**

Instrument: **Integrated Project**

Thematic Priority: **Information and Communication Technologies**

D3.1.3 Defining the Features of the WSML-DL v2.0 Language

Activity N:	Activity 2 - Core Research and Development	
Work Package:	WP3 - Service Annotation and Reasoning	
Due Date:		M12
Submission Date:		10/09/2009
Start Date of Project:		01/03/2008
Duration of Project:		36 Months
Organisation Responsible of Deliverable:		UIBK
Revision:		1.0
Author(s):	Barry Bishop Florian Fischer Pascal Hitzler Markus Krötzsch Sebastian Rudolph Yiorgos Trimponias Gulay Unel UIBK (All)	
Reviewers(s):	Sudhir Agarwal Nathalie Steinmetz UKARL	

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	30.01.2009	Parts 1+2	Florian Fischer
0.2	02.02.2009	Complete draft	Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph
0.3	04.02.2009	Editing and corrections	Barry Bishop, Gulay Unel
1.0	05.03.2009	Incorporating reviewer comments	Pascal Hitzler
Final	10.03.2009	Overall fortmat and quality revision	Malena Donato

Table of Contents

Table of Figures

Glossary of Acronyms

Acronym	Definition
D	Deliverable
EC	European Commission
WP	Work Package
HLDD	High Level Design Document
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
LP	Logic Programming
DL	Description Logic
OWL	Web Ontology Language
DLP	Description Logic Program
BT	British Telecom
C2C	Consumer to Consumer

Executive summary

In order to automate tasks such as location and composition, Semantic Web Services must be described in a well-defined formal language. The Web Services Modeling Language (WSML) is based on the conceptual model of the Web Service Modeling Ontology (WSMO) and as such can be used for modeling Web services, ontologies, and related aspects.

WSML is actually a family of several language variants, each of which is based upon a different logical formalism. The family of languages is unified under one syntactic umbrella, with a concrete syntax for modeling ontologies.

This deliverable, along with others, defines an updated version of the WSML language stack, in order to bring it in line with the scalability requirements of reasoning in SOA4All and realign it with new research results and others standards. Thus, this document describes WSML-DL v2.0, a WSML language variant of the Description Logics paradigm featuring a favorable trade-off between expressivity and scalability.

By capturing the semantics of the ELP knowledge representation formalism, the updated WSML-DL variant greatly improves scalability and lends itself to evaluation using rule-based reasoners.

1. Introduction

SOA4All's aim is to facilitate a Web where billions of parties are exposing and consuming services via advanced Web technology. The outcome of the project will be a framework and infrastructure "that integrates four complementary and revolutionary technical advances into a coherent and domain independent service delivery platform":

- Web principles and technology as the underlying infrastructure for the integration of services at a worldwide scale.
- Web 2.0 as a means to structure human-machine cooperation in an efficient and cost-effective manner.
- Semantic Web technology as a means to abstract from syntax to semantics as required for meaningful service discovery.
- Context management as a way to process in a machine understandable way user needs that facilitates the customization of existing services for the needs of users.

Thus, one basic technological building block is Semantic Web technology, which abstracts from pure syntax to semantics. Ontologies are used as a semantic data model, by which means services gain machine-understandable annotations. This information makes the development of high quality techniques for automated selection, construction, etc. possible. Furthermore, precise formal models allow for the expression of context-specific rules and constraints, which can be taken into account during the inference process. The basic building blocks for this are formal languages for describing resources in a clear and unambiguous way.

The Web Service Modeling Language WSML [1] is such a formal language for the specification of ontologies and different aspects of Web services, based on the conceptual model of WSMO [2]. Several different WSML language variants exist, which are founded upon different logical formalisms. The main formalisms exploited for this purpose are Description Logics [3], Logic Programming [4], and First-Order Logic [5]. Furthermore, WSML has been influenced by F-Logic [6] and frame-based representation systems.

This deliverable introduces a revised version of the WSML-DL variant of the WSML language family, in order to bring it up to date with recent research in both the Description Logic and Logic Programming paradigm. It belongs to a set of conceptually related M12 deliverables, namely:

- D3.1.1 Defining the features of the WSML-Quark language
- D3.1.2 Defining the features of the WSML-Core v2.0 language
- **D3.1.3 Defining the features of the WSML-DL v2.0 language**
- D3.1.4 Defining the features of the WSML-Rule v2.0 language

These four deliverables form the foundation for a redefinition of WSML that brings it in line with the tractability requirements of SOA4ALL, which envisions "billions of parties exposing services". Working with and reasoning over the vast datasets that are implied by this vision poses a significant scalability challenge.

A lot of current standards and knowledge representation formalisms for the Web feature very high worst-case complexity results, ranging from EXPTime-complete to NEXPTIME-complete. For example, such worst-case results apply to the OWL language family as well as to the former version of WSML-DL, which is a notational variant of the Description Logic SHIQ(D) [7].

In general, tableaux-based methods for Description Logics behave rather efficiently in regard to TBox (schema) reasoning, however in general they do not scale very well when faced with

a large ABox (a large instance set) [8].

In order to support tractable inference at a Web-scale there have been proposals for more lightweight representation formalisms such as the DL-Lite family of languages [9], EL++ [10], as well as tractable fragments of OWL like DLP [11], OWL-Horst [12], or L2 [13]. Several of these proposals are in the process of being adopted in the upcoming OWL 2 standard as so-called profiles [14]. This deliverable is thus part of an effort to align WSML with these research and standardization efforts.

1.1 Purpose and Scope

1.1.1 Audience

This document is intended as a reference of the features of the WSML language. In turn its main audience are users who want to model Web services and ontologies using WSML, as well as technical staff building tools (i.e. reasoners) that use the WSML language.

Inside the consortium, this mainly applies to partners involved in technical work packages within Activity cluster A2 – "Core R&D Activities". For outside parties beyond the consortium it can serve as an introduction to WSML.

1.1.2 Scope

The main purpose of this deliverable is to present the features of the reworked WSML-DL v2.0 language variant, particularly to describe the changes made in regard to the existing WSML-DL language[1].

We describe the modeling elements in WSML-DL v2.0, restrictions imposed on the language, and a motivation for them. Beyond the definition of the conceptual and the logical expression syntax of the language itself we also outline the steps involved in a practical implementation and explain the relation with the other language variants within the WSML stack and their respective layering.

1.2 Structure of the document

The remainder of this deliverable is structured as follows: Section 2 clarifies the relation of WSML-DL v2.0 and the WSML language in relation to the SOA4All project and other deliverables. Section 3 defines the WSML-DL v2.0 language by describing the individual language elements and pointing out the particular restrictions placed on them for this language variant. It then proceeds to outline the algorithmization of WSML-DL v2.0 on rule-based reasoners. Finally, it clarifies the relation WSML-DL v2.0 and the other WSML language variants and their layering. Section 4 concludes this deliverable and points out some next steps for future work.

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

2. Technical deliverable remarks

2.1 Deliverable relation with the architecture of the project

The work conducted towards a reworked WSML language stack in WP3 conceptually belongs to the Base Layer of the SOA4All architecture (see ***Figure 1***). In the SOA4All architecture, different elements are distributed in three different layers according to their functional dependencies on each other.

The Base Layer contains elements such as (1) formal languages and ontologies, (2) reasoner and (3) semantic spaces as the publication and communication element of the infrastructure.

The Web Enabled Service platform (the second layer), consists of (4) Service Ranking and Selection, (5) Service Location, (6) Service Adaptation and Service, (7) Service Grounding, (8) Service Delivery, (9) Service Monitoring and Management and (10) Service Context.

Finally, in the User Layer are components such as (11) Service Modeling, (12) Service Provisioning and (13) Service Consumption.

The “Semantic Service Bus” ties all these components together and serves as the infrastructural backbone. In ***Figure 1*** the Semantic Service Bus is indicated by the outer “envelope” around the other components and shows the possibility of being connected to other buses as an extension.

The changes to the WSML family have direct consequences for the reasoning components to be developed in WP3, which directly process descriptions created using these formal languages.

Furthermore, any component from the second layer, which operates on (i) semantically annotated Web Services or (ii) relies on an ontology for other reasons will make use of WSML, at least indirectly. This most directly applies to WP5 for the purpose of service location, discovery and ranking, as well as to WP6 for the purpose of service composition.

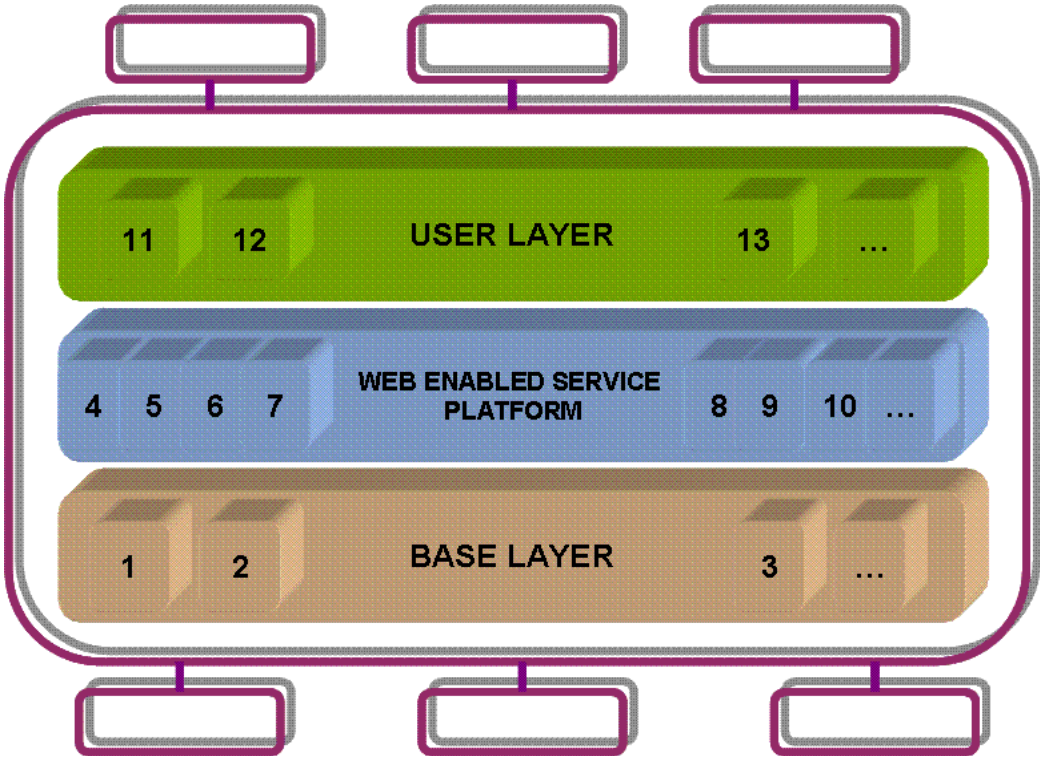

Figure 1 SOA4All Semantic Service Bus

2.2 Deliverable relation with the use-cases

This section clarifies the relation of the WSML language family with the use-case activities in SOA4All and points out direct applications of WSML as they are apparent at the time of the writing.

2.2.1 End-user Integrated Enterprise Service Delivery Platform

As the End-user Integrated Enterprise Service Delivery Platform case study will fully use service annotation and reasoning about such annotations, it will also make direct use of WSML and the reasoner components associated with it.

This use-case aims for an open, dynamic and lightweight service platform instead of heavyweight existing solutions, which are hard to set up and maintain due to their complexity. An envisioned outcome (among several) from the end user's perspective is a tool to compose processes[2] from services and reuse services in a visual tool without requiring an in-depth technical background. Apart from the requirements that stem from **service composition** an envisioned outcome of the use-case is to provide support for publishing, finding and reusing existing processes. In order to find processes in repositories **search** mechanisms based on semantic descriptions (and hence WSML descriptions) are required.

2.2.2 W21C BT Infrastructure

This use-case will create a semantically enhanced and expanded version of BT's Web21c

[2] In the loose sense of a "business process" composed from various subtasks (services) in order to accomplish a specific goal.

platform [15], which will result in a framework for the delivery of service, both by BT itself and third parties. This requires in-depth technical knowledge and the aim of the case study is to simplify the process of **discovering**, **integrating**, using and sharing BTs capabilities on this platform. Thus, in the BT W21C case study the focus is shifted slightly by using **service location** technologies to discover capabilities within the BT Web21c infrastructure.

Reasoning with formal service semantics forms the basis for **composition** tools that will enhance and aide the creation of more complex services. Furthermore, unambiguous descriptions of services facilitate the **selection** of services for the end user. WSML will thus be used directly in this work package.

2.2.3 C2C Service eCommerce

One of the focuses of this use-case in WP9 is to investigate the impact and sustainability of future C2C eCommerce applications based on services and to enable eCommerce as a common distribution channel for end users by means of SOA4All. In this scenario, non-technical end users can make use of existing services and combine them to build eCommerce applications in order to market and sell their own products.

This use-case again entails several tasks that are based on annotation and (WSML) reasoning, among them easy **composition** of services**, service location, ranking** and **selection** in the case of similar services. In this sense the scenario demonstrates almost all parts of the SOA4ALL concept including service discovery, integration, etc. and as such heavily relies on the formal languages work conducted in WP3.

3. WSML-DL Language Definition

3.1 Motivation

Version 1.0 of WSML-DL[3] was based on the SHIQ(D) Description Logic and thus was essentially aligned with OWL 1.0 DL.[4] Because of new developments in requirements on ontology languages, and in particular because of the forthcoming revision of OWL – known as OWL 2 [16] – a revision of WSML-DL is due.

We sketch here the main points in this revision of WSML-DL and direct the reader towards the complete WSML language specification [22], where the syntax of all WSML variants is laid out in detail and in relation to each other (see Section 4 of the indicated document).

The revision was to meet a number of criteria to ensure that it is up to date with state-of-the-art developments in OWL and in ontology languages. They are as follows.

- Align it with OWL 2.
- Achieve higher efficiency of reasoning by using tractable fragments.
- Incorporate new results on the integration of OWL and rules.
- Make it implementable on a Datalog reasoner in order to be able to further integrate tools for the different WSML variants.

These requirements have been met by the revision. In particular,

- WSML-DL v2.0 is a tractable language which encompasses – and is thus fully compatible with – all three designated tractable fragments of OWL 2, namely OWL EL, OWL RL, and OWL QL (see Section 3.2),
- it allows for the expression of certain types of rules which are not expressible in the OWL 2 tractable fragments, but expressible in OWL 2 itself,
- it allows for the expression of Datalog rules which are not expressible in OWL 2 DL,
- it fully adheres to the open-world semantics and is thus fully compatible with the OWL DL semantics, and
- it is implementable on a Datalog reasoner via a polynomial-time transformation into Datalog.

An implementation of WSML-DL v2.0 is under way and progress towards the first prototype will be reported in the M18 deliverable D3.2.4 “First Prototype Description Logic Reasoner for WSML-DL v2.0”.

3.2 Approach

The definition of WSML-DL v2.0 is based on recent research developments that have been communicated in [17]. In essence, WSML-DL v2.0 is a syntactic variant of the language ELP [17], which is the most expressive polynomial-time language based on OWL currently known. It is particularly suited for WSML since it incorporates rules into an OWL-style language, and as such is in line with the conceptual build-up of the WSML layering that includes both OWL and rule-like languages. It covers, or is closely related to, a number of well-known formalisms, which we list in the following.

[3] http://www.wsmo.org/2004/d16/d16.7/v0.1/20040719/

[4] http://www.w3.org/TR/owl-features/

Datalog

Datalog is essentially the Horn fragment of first-order logic restricted to formulae without function symbols, but is usually endowed with a Herbrand semantics. It is a prominent and very well known language, which has been studied extensively. Since Datalog can be understood as a fragment of first-order logic, it can be read under first-order semantics, which is compatible with the Herbrand perspective in that it retains all the positive consequences which can be drawn under the Herbrand semantics. ELP contains Datalog but semantically reads it as DL-safe rules [18], and this reading again is compatible with the Herbrand perspective in the same sense.

DLP

Description Logic Programs (DLP) [11] can be roughly described as the logical fragment formed by the naïve intersection of Description Logics and Logic Programming. The practical use of DLP is actually twofold: First of all. it is a tractable formalism that captures many of the ontologies found on the Web, and secondly it can serve as a basic interoperability layer between Description Logic and Logic Programming based formalisms. ELP contains DLP.

OWL 2 RL

The OWL 2 RL profile [14], which is part of the forthcoming OWL 2 standard, can be considered to be roughly based on DLP.

OWL 2 RL [14] is a fragment that is customized to support reasoning with rule-based engines. It is a profile that is intended to form a proper extension of RDFS while still being computationally tractable. As such, it realizes a weakened form of the OWL 2 Full semantics and is very similar in spirit to DLP and OWL-Horst [12]. OWL 2 RL semantics is provided as a partial axiomatization in the form of additional entailment rules directly on the RDF serialization of OWL 2, in a similar fashion as for OWL-Horst. ELP contains OWL 2 RL.

OWL 2 QL

OWL 2 QL [14] is based on DL-Lite [9], which is actually not a single language fragment but rather a family of languages with several slight variations, and tailored for reasoning over large instance sets combined with a relatively inexpressive TBox. More specifically, many important reasoning tasks can be performed in logarithmic space with respect to the size of the ABox. The variant picked up in OWL 2 is DL-Lite$_R$ and supports property inclusion axioms. Other variants instead support (inverse) functionality on object properties. A notable feature goes hand in hand with DL-Lite's complexity results: Since query answering over knowledge bases has polytime data complexity and since it is possible to separate TBox and ABox reasoning for the evaluation of a query, it is possible to delegate the ABox reasoning to a standard SQL database engine. Moreover, increasing the expressiveness of the DL-Lite languages also increases space complexity to at least NLogSpace. Some languages from the DL-Lite family, and thus also OWL 2 QL, are the maximally expressive Description Logics having the feature of being able to use database engines (along with all the optimizations employed in them) for query evaluation, and thus also support very efficient query answering for large instance sets. As such OWL 2 QL is optimized for data complexity. ELP contains OWL 2 QL.

OWL 2 EL

OWL 2 EL [14], which is based on the Description Logic EL++ [10], is a fragment that is tractable in regard to several key inference tasks such as consistency checking, subsumption checking, instance checking or classification – they can be decided in polynomial time. On the other hand it is still expressive enough to adequately model many real world problems. The most prominent constructs from OWL 2 that are disallowed in OWL 2 EL are disjunction, negation, enumerations of multiple elements, inverse and irreflexive object properties,

functional object properties, symmetric and asymmetric object properties as well as several constructs involving universal quantification. ELP contains OWL 2 EL.

EL++ Rules

EL++ Rules [19] is a rules-extension of OWL 2 EL that is of polynomial time complexity and still contained in OWL 2 DL. The main expressive extension comes from allowing Datalog-like rules under the restriction that the pattern of variables occurring in rule bodies must be tree-shaped (in a sense which will be explained later). Furthermore, EL++ class expressions can be used in place of predicates in the Datalog rules. ELP contains EL++ Rules.

3.3 ELP in a Nutshell

The essential ELP language features are the following, which we state in an intuitive and somewhat simplified manner. A formal and thorough treatment of all language features can be found in [17].

OWL 2 EL

ELP contains all the expressive features of OWL 2 EL [14], also known as the description logic EL++ that essentially features conjunction, existential quantification, nominals and role chains.

Role Conjunctions

ELP allows the expression of conjunctions of roles – these can be used in place of normal roles. (Some global restrictions on their use apply.)

Tree-shaped EL++ Rules

ELP allows the expression of Datalog-like rules with the following features.

- EL++ complex class expressions and role expressions are used in the place of atomic predicates.
- The pattern of variables occurring in a rule body is tree-shaped. To check for tree-shapedness, intuitively speaking, the body variables are understood as vertices in a graph, and there is an edge between two variables if they are connected by a role in the rule body. If the resulting graph is actually a tree, then it qualifies for an ELP rule body.
- The head of the rule, which must be a role name or a class expression, must refer to the root of the tree. In the case of a role name, the root must refer to the first parameter of the role (when expressed as a binary relation).

The tree-shapedness of rules bodies is required in order to guarantee decidability of the language. Its origin lies in the fact that in description logics similar patterns occur when translating complex class descriptions to first-order predicate logic.

Non-tree-shaped EL++ Rules with DL-safe variables

ELP allows a rule body to be non-tree-shaped, but in this case it must consist of several tree-shaped parts which are connected only by shared, so-called safe variables. Cycles, even such involving safe variables, must not occur. Safe variables are semantically interpreted in the sense that they can only be instantiated by instances that explicitly occur in the knowledge base. As such, they are a generalisation of DL-safe rules as mentioned above

(and also below).

DL-safe Datalog Rules

ELP allows for the expression of any Datalog rule, but the rule is interpreted as a DL-safe rule, i.e. all variables occurring in the rule body are semantically interpreted in the sense that they can only be instantiated by instances which explicitly occur in the knowledge base.

Safe Range Restrictions

ELP allows for the expression of range restrictions in the sense known from OWL. Some global restrictions on their use apply, though.

3.4 WSML-DL v2.0 Syntax Definition

In this section the WSML-DL v2.0 logical expression syntax is given. It is a syntactic variant of ELP. Further details can be found in the WSML v2.0 syntax specification [22].

Definition 3.1. A WSML-Core vocabulary *V* follows the following restrictions:

- V_C, V_D, V_R, V_I and V_{ANN} are the sets of concept, datatype, relation, instance and annotation identifiers. These sets are all subsets of the set of IRIs and are pairwise disjoint.
- The set of attribute names is equivalent to V_R
- The set of relation identifiers V_R is split into two disjoint sets, V_{RA} and V_{RC}, which correspond to relations with an abstract and relations with a concrete range, respectively.
- We assume that V_{RA} contains the two identifiers topObjectProperty (i.e. the relation which relates any two things) and bottomObjectProperty (i.e. the relation is empty), and that V_{RC} contains the two identifiers topDataProperty and bottomDataProperty (which are the corresponding notions relating to datatypes).

The arguments of a datatype wrapper in WSML-Core can only be strings, integers or decimals. No other arguments, also not variables, are allowed for such data terms.

Definition 3.2. Any WSML-Core vocabulary V is a WSML-DL vocabulary.

Let *V* be a WSML-Core vocabulary and let V_V be a set of variable identifiers. We use the convention that the names of such variables always begin with a question mark, e.g. $?x \in V_V$. Let V_S be a set disjoint from V_V that we call the set of *safe* variables. We will adhere to the convention that the names of such variables always begin with an exclamation mark, e.g. $!x \in V_S$.

Now, let $\gamma \in V_C$, let Γ be either an identifier in V_C or a list of identifiers in V_C, let Δ be either an identifier in V_D or a list of identifiers in V_D, let $\chi \in V_I$, $x \in V_V$, $\varphi \in V_I \cup V_V \cup V_S$, let ψ be either an identifier in $V_I \cup V_V \cup V_S$ or a list of identifiers in V_I, let $p,q \in V_{RA}$, $s,t \in V_{RC}$, and let *Val* be either a data value or a list of data values.

Definition 3.3. The set of atomic formulae, also called *molecules* in *L(V)* is defined as follows:

- φ **memberOf** Γ is an atomic formula in *L(V)*
- γ **subConceptOf** Γ is an atomic formula in *L(V)*
- $\{\chi\}(x)$ is an atomic formula in *L(V)*

- γ[*s* **ofType** Δ] is an atomic formula in *L(V)*
- γ[*s* **impliesType** Δ] is an atomic formula in *L(V)*
- γ[*p* **impliesType** Γ] is an atomic formula in *L(V)*
- φ[*p* **hasValue** ψ] is an atomic formula in *L(V)*
- φ[*s* **hasValue** *Val*] is an atomic formula in *L(V)*
- If *α* and *β* ∈ V_I then *α* = *β* is an atomic formula in *L(V)*

Let ψ,φ be arbitrary WSML variables. We call molecules of the form φ **memberOf** Γ or {χ}(*x*) *a-molecules*, and molecules of the forms ψ [*p* **hasValue** φ] *b-molecules*, respectively. If F is such an a-molecule or b-molecule, then we sometimes write it as F<φ>, F<*x*>, respectively F<ψ,φ> to indicate the (safe or non-safe) variables occurring in it. In some cases, it is allowed to use instance identifiers in the place of variables, and we will explicitly point them out.

Definition 3.4. We define the set of relation descriptions in *L(V)*. We denote each relation description G also in the form G<ψ,φ>, where ψ,φ∈$V_V \cup V_S$, and we call <ψ,φ> the distinguished variable pair of G.

- Any b-molecule G<ψ,φ> is a relation description in *L(V)*.
- If *G*<ψ,φ> and H<ψ,φ> are relation descriptions in *L(V)*, then *G* **and** *H* is a relation description in *L(V)* with distinguished variable pair <ψ,φ>.

Definition 3.5. We define the set of concept descriptions in *L(V)*. We denote each concept description F also in the form F<φ>, where φ∈V_V, *and we call* φ the distinguished variable of F.

- Any a-molecule F<φ> is a concept description in *L(V)*.
- **true**, **false** are concept descriptions in *L(V)*. We adopt the notational convention that **true = true**<φ> and **false** = **false**<φ> for any φ.
- If F_1<φ> and F_2<φ> are concept descriptions in *L(V)* and *G*<ψ,φ> and H<ψ,!χ> are relation descriptions in *L(V)*, then
  - F_1 **and** F_2 is a concept description in *L(V)* with distinguished variable φ,
  - **exists** φ (*G* **and** F_1) is a concept description in *L(V)* with distinguished variable ψ,
  - **exists** !χ (ψ [H hasValue !χ]) is a concept description in *L(V)* with distinguished variable ψ,
  - φ[G **hasValue** φ] is a concept description in *L(V)* with distinguished variable φ.

Definition 3.6. Let $F,F_1,\ldots,F_n$, be WSML-DL concept descriptions, γ∈V_C and p∈V_{RA}, let $G,H,G_1,\ldots,G_{n-1}$ be relation descriptions and let φ,$\varphi_1,\ldots$, φ_n∈V_V be distinct. Furthermore, let $f_1,\ldots,f_k$ be a-molecules and $g_1,\ldots,g_m$ be b-molecules and let χ_1, ,…, χ_k,ψ_1,ς_1,…, ψ_m,ς_m∈$V_I \cup V_S$. The set of WSML-DL formulae in *L(V)* is defined as follows:

- Any atomic formula which does not contain a WSML variable is a formula in *L(V)*.
- F<φ_1> **implies** F_1<φ_1> is a formula in *L(V)*.
- F<φ_1> **impliedBy** F_1<φ_1> is a formula in *L(V)*.
- F<φ_1> **equivalent** F_1<φ_1> is a formula in *L(V)*.
- F_1<φ_1> **and** G_1<φ_1,φ_2> **and** F_2<φ_2> **and** … **and** F_{n-1}<φ_{n-1}> **and** G_{n-1}<φ_{n-1},φ_n> **and** F_n<φ_n> **and** f_1<χ_1> **and** … **and** f_k<χ_k> **and** g_1<ψ_1,ς_1> **and** … **and** g_m<ψ_m,ς_m> **implies** F<φ_1> is a formula in *L(V)*. If F_i = **true** or if G_i = topObjectRole, it can be omitted.

- $F\langle\varphi_1\rangle$ **impliedBy** $F_1\langle\varphi_1\rangle$ **and** $G_1\langle\varphi_1,\varphi_2\rangle$ **and** $F_2\langle\varphi_2\rangle$ **and** … **and** $F_{n-1}\langle\varphi_{n-1}\rangle$ **and** $G_{n-1}\langle\varphi_{n-1},\varphi_n\rangle$ **and** $F_n\langle\varphi_n\rangle$ **and** $f_1\langle\chi_1\rangle$ **and** … **and** $f_k\langle\chi_k\rangle$ **and** $g_1\langle\psi_1,\varsigma_1\rangle$ **and** … **and** $g_m\langle\psi_m,\varsigma_m\rangle$ is a formula in *L(V)*. If F_i = **true** or if G_i = topObjectRole, it can be omitted.
- $F_1\langle\varphi_1\rangle$ **and** $G_1\langle\varphi_1,\varphi_2\rangle$ **and** $F_2\langle\varphi_2\rangle$ **and** … **and** $F_{n-1}\langle\varphi_{n-1}\rangle$ **and** $G_{n-1}\langle\varphi_{n-1},\varphi_n\rangle$ **and** $F_n\langle\varphi_n\rangle$ **and** $f_1\langle\chi_1\rangle$ **and** … **and** $f_k\langle\chi_k\rangle$ **and** $g_1\langle\psi_1,\varsigma_1\rangle$ **and** … **and** $g_m\langle\psi_m,\varsigma_m\rangle$ **implies** $G\langle\varphi_1, \varphi_n\rangle$ is a formula in *L(V)*. If F_i = **true** or if G_i = topObjectRole, it can be omitted.
- $G\langle\varphi_1, \varphi_n\rangle$ **impliedBy** $F_1\langle\varphi_1\rangle$ **and** $G_1\langle\varphi_1,\varphi_2\rangle$ **and** $F_2\langle\varphi_2\rangle$ **and** … **and** $F_{n-1}\langle\varphi_{n-1}\rangle$ **and** $G_{n-1}\langle\varphi_{n-1},\varphi_n\rangle$ **and** $F_n\langle\varphi_n\rangle$ **and** $f_1\langle\chi_1\rangle$ **and** … **and** $f_k\langle\chi_k\rangle$ **and** $g_1\langle\psi_1,\varsigma_1\rangle$ **and** … **and** $g_m\langle\psi_m,\varsigma_m\rangle$ is a formula in *L(V)*. If F_i = **true** or if G_i = topObjectRole, it can be omitted.
- φ_1 **memberOf** γ **impliedBy** $\varphi[\ p$ **hasValue** $\varphi_1\ ]$ is a formula in *L(V)*.
- $\varphi[\ p$ **hasValue** $\varphi_1\ ]$ **implies** φ_1 **memberOf** γ is a formula in *L(V)*.

Furthermore, the following global restrictions must be satisfied.

- Relation descriptions which are not b-molecules must not contain non-simple relation names. Thereby, a relation name is called *non-simple* if it occurs on the right-hand side of **implies** (respectively, on the left-hand side of **impliedBy**) while on the corresponding left-hand side (respectively, right-hand side) there (1) occurs more than one relation description or (2) there occurs a non-simple relation. Simplicity of relations is a technical notion borrowed from description logics – this global restriction is needed to ensure decidability of the language.
- Concept descriptions occurring in the same formula must not share variables unless they are distinguished or safe. Furthermore, cycles must not occur, which means that it is not allowed that a variable (safe or not) is reachable from another variable (safe or not) via two distinguished paths along relation descriptions. For determining cycles, topObjectRole is ignored.
- If there is a formula $\varphi[\ p$ **hasValue** $\varphi_1\ ]$ **implies** φ_1 **memberOf** γ (respectively, φ_1 **memberOf** γ **impliedBy** $\varphi[\ p$ **hasValue** $\varphi_1\ ]$) and at the same time a formula A **implies** $\varphi_2[\ p$ **hasValue** $\varphi_3\ ]$ (respectively, $\varphi_2[\ p$ **hasValue** $\varphi_3\ ]$ **impliedBy** A), then φ_3 **memberOf** γ must occur in A.

The differences between WSML-DL v1.0 and WSML-DL v2.0 are substantial, since they reflect the change from the description logic SHIQ(D) to the ELP language. Essentially, most complex description logic features which lead to high computational complexity are now removed.

3.5 Algorithmisation

Reasoning in ELP – and thus in WSML-DL v2.0 – can be realised by compiling ELP knowledge bases into Datalog. The compilation is possible with polynomial time-dependency on the size of the knowledge base. Full details of the algorithmisation can be found in [17]. A brief and non-rigorous description of these steps follows:

1. Re-write ELP knowledge-base to normal form, i.e. transcribe description logic constructors to rules, e.g. DL intersection becomes LP conjunction. The generated rule-base is equisatisfiable, the size of which is polynomial in the size of the input rule-base.
2. Re-write rules to remove conjunction in rule-heads, i.e. create one new rule for each head literal.
3. Re-write rules to have no more than 3 variables, i.e. "reduce the forest structure of rule bodies" as per Theorem 16 from [17].

4. Ground all safe variables with all known instance names.
5. Remove range restrictions by replacing each occurrence with a rule that infers an instance's membership of a special 'Range' predicate.
6. Renormalize again as per step 1.
7. Complete the generation of a Datalog rule-set by adding special symbols and rules for roles, concepts and individuals.

3.6 Relation with other WSML Variants and Language Layering

As mentioned earlier, WSML actually consists of distinctly different language variants, identified for their particular properties in terms of modeling and performance of reasoning tasks. They differ in expressiveness as well as in their underlying logical formalism. This allows users of the language to decide on (i) the level of expressivity and thus also on (ii) the associated complexity, as well as (iii) the style of modeling which they want to use, on a case by case basis – depending on the requirements of a specific application.

The relation between the different WSML variants is depicted in ***Figure 2***. As can be seen, WSML-Quark and WSML-Core 2.0 form a common, lightweight, yet increasingly expressive foundation for extensions towards the paradigms of both Description Logic (in the form of WSML-DL v2.0) and Logic Programming (in the form of WSML-Flight 2.0 and WSML-Rule 2.0). Consequently, WSML-DL v2.0 and WSML-Flight/Rule 2.0 are both layered on WSML-Core 2.0, which defines a common subset. WSML-Core v2.0 is in turn layered upon WSML-Quark.

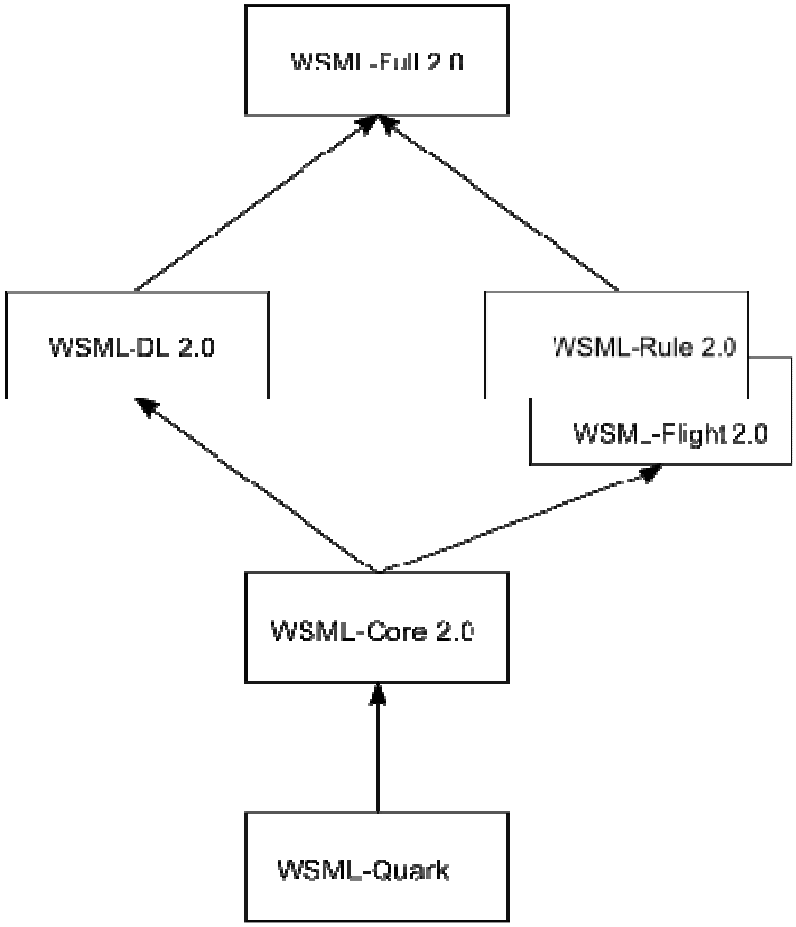

Figure 2 WSML Language Layering

WSML-Quark is a very lightweight and intuitive language variant that allows for the simple organization of concepts in to a hierarchical classification system. WSML-Quark can be used as a very efficient stepping stone towards more formal and complex WSML language variants.

WSML-Core 2.0 inherits many features from the first version of WSML-Core, which was based on DLP [11] - formed by the intersection of the Description Logic SHIQ and Horn Logic. It has been adjusted to align results of ongoing standardization efforts, most notably OWL 2 RL [14], as well research results such as the L2 language [13], which has similar language features, albeit specified directly at the level of RDF. Furthermore, WSML-Core 2.0 forms the common subset between the DL and LP based variants of WSML.

WSML-DL v2.0 is the Description Logic variant of WSML, based on ELP [17], which is based on the tractable DL EL++ [10], and covers OWL 2 RL, OWL 2 EL and OWL 2 QL, while at the same time retaining polynomial combined complexity.

WSML-Flight 2.0 is the least expressive of the two LP-based variants. Compared with WSML-Core, it adds features such as meta-modeling, constraints, and non-monotonic (stratified) negation. WSML-Flight is semantically equivalent to Datalog with equality and integrity constraints.

WSML-Rule 2.0 extends WSML-Flight 2.0 with further features from Logic Programming, namely the use of function symbols, unsafe rules, and unstratified negation. Due to the intended tractability goals, WSML-Rule 2.0 relies on the Well-Founded Semantics [20] in place of the more general Stable Model Semantics for the purpose of query answering.

WSML-Full 2.0 finally reconciles the DL and LP variants of WSML in a more expressive superset. While the specification of WSML-Full is still open at this stage, the use of hybrid MKNF knowledge bases forms a possible option. [21] defines the well-founded semantics for this approach, which still preserves tractable data complexity.

3.7 Future Work

Foremost is the realisation of a prototype implementation of WSML-DL v2.0. This is ongoing work and that will be reported in the M18 deliverable D3.2.4 “First Prototype Description Logic Reasoner for WSML-DLv2.0”.

Further future work will focus on application of WSML-DL in SOA4All use cases, and possibly – if the need arises – on further tool support for such applications.

On the more foundational side, the implementation of a WSML-DL v2.0 reasoner will be achieved with the first implementation of a standalone ELP reasoner, which will necessarily include:

1. A software component for representing ELP knowledge-bases
2. A software component for translating from ELP to Datalog representations
3. A Datalog reasoner – IRIS enhanced with instance equivalence (equality in rule heads)

Further on, when WP3 will start to consider methods for defining WSML-Full 2.0, (D3.2.8 - Month 36), consideration will be given to implementing the algorithmisation of hybrid ELP following [21].

4. Conclusions

This deliverable reported on the redefinition of WSML-DL which is based on the ELP language, and is thus a modern redesign based on state-of-the-art developments in tractable ontology languages and the integration of OWL and rules. It

- modernises WSML-DL in keeping with the spirit of the previous WSML-DL 1.0,
- aligns WSML-DL with the new OWL 2,
- covers all tractable profiles of OWL 2,
- is of polynomial-time complexity,
- incorporates new results on the integration of OWL and rules, and
- is implementable on top of a Datalog reasoner.

5. References

[1] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel, “The Web Service Modeling Language WSML,” *WSML Final Draft D*, vol. 16, 2005.

[2] D. Roman, H. Lausen, and U. Keller, “Web Service Modeling Ontology (WSMO),” *WSMO Working Draft*, 2004.

[3] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, “The Description Logic Handbook,” Cambridge University Press, 2nd edition, 2007.

[4] J.W. Lloyd, "Foundations of logic programming," Springer, 1987.

[5] M. Fitting, "First-Order Logic and Automated Theorem Proving," Springer, 1996.

[6] M. Kifer and G. Lausen, “F-logic: a higher-order language for reasoning about objects, inheritance, and scheme,” *In: Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, 1989, pp. 134-146.

[7] I. Horrocks, U. Sattler, and S. Tobies, “Practical reasoning for very expressive description logics,” *Logic Journal of the IGPL*, vol. 8, 2000, pp. 239-263.

[8] U. Hustadt, B. Motik, and U. Sattler, “Data Complexity of Reasoning in Very Expressive Description Logics,” In: L. Kaelbling and A. Saffiotti (eds.), *IJCAI-05, Proceedings of the 19th International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, 2005*. Professional Book Center 2005, pp. 466-371.

[9] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, “DL-Lite: Tractable Description Logics for Ontologies,” In: *Proceedings of the National Converence on Artificial Intelligence*, AAAI Press, 2005, p. 602.

[10] F. Baader, S. Brandt, and C. Lutz, “Pushing the EL Envelope,” *In: Proceedings of the International Joint Conference on Artificial Intelligence 2005, Lawrence Erlbaum Associates Ltd.,* 2005, p. 364.

[11] B. GROSOF, I. HORROCKS, R. VOLZ, and S. DECKER, “Description Logic Programs: Combining Logic Programs with Description Logic,” *In: Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, 20-24 May 2003*. ACM, 2003, pp. 48-57.

[12] H.J. ter Horst, “Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity,” *Proc. of ISWC*, Springer, 2005, pp. 6-10.

[13] F. Fischer, U. Keller, A. Kiryakov, Z. Huang, V. Momtchev, E. Simperl, D. Fensel, and R. Dumitru, “D1.1.3 Initial Knowledge Representation Formalism,” *LarKC Deliverable*, 2004.

[14] B. Motik, C. Bernardo, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, “OWL 2 Web Ontology Language: Profiles,” *W3C Working Draft*, Dec. 2008.

[15] “Web21C SDK.” http://web21c.bt.com/

[16] B. Motik et al., "OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax," *W3C Working Draft* 02 December 2008. http://www.w3.org/TR/2008/WD-owl2-syntax-20081202/

[17] M. Krötzsch, S. Rudolph, and P. Hitzler, “ELP: Tractable rules for OWL 2,” *Proceedings of the 7th International Semantic Web Conference (ISWC2008)*, Springer, 2008.

[18] B. Motik, U. Sattler, and R. Studer, "Query Answering for OWL-DL with rules," *Journal of Web Semantics* 3(1):41-60, 2005.

[19] M. Krötzsch, S. Rudolph, and P. Hitzler, "Description Logic Rules," In: M.Ghallab et al., *Proceedings of the 18th European Conference on Artificial Intelligence, ECAI2008, Patras, Greece, July 2008*. IOS Press, 2008, pp. 80-84.

[20] A. Van Gelder, K.A. Ross, and J.S. Schlipf, "The well-founded semantics for general logic programs," *Journal of the ACM (JACM)*, vol. 38, 1991, pp. 619-649.

[21] M. Knorr, J.J. Alferes, and P. Hitzler, "A Coherent Well-founded Model for Hybrid MKNF Knowledge Bases," In: M.Ghallab et al., *Proceedings of the 18th European Conference on Artificial Intelligence, ECAI2008, Patras, Greece, July 2008*. IOS Press, 2008, pp. 99.

[22] WSML Language specification v1.0, http://www.wsmo.org/TR/d16/d16.1/