



D16.7v0.1. WSML-Core

WSML Working Draft 2 August 2004

This version:

<http://www.wsmo.org/2004/d16/d16.7/v0.1/20040802/>

Latest version:

<http://www.wsmo.org/2004/d16/d16.7/v0.1/>

Previous version:

<http://www.wsmo.org/2004/d16/d16.7/v0.1/20040719/>

Editors:

Jos de Bruijn
Douglas Foxvog

Authors:

Jos de Bruijn
Douglas Foxvog
Eyal Oren
Dieter Fensel

Reviewer:

Ian Horrocks

This document is also available in non-normative [PDF](#) version.

Copyright © 2004 [DERI](#)®, All Rights Reserved. [DERI](#) liability, trademark, document use, and software licensing rules apply.

Abstract

We introduce WSML-Core, an ontology modeling language, based on the semantics of OWL Lite⁻ and the conceptual model of ontologies presented in WSMO.

We describe each modeling element in the syntax of WSML-Core and describe the semantics through a mapping to (the OWL Lite⁻ subset of) the OWL abstract syntax, which has a model-theoretic semantics associated with it. When analyzing the applicability of WSML-Core to the current version of the WSMO use cases we see some limitations of WSML-Core, mainly the lack of datatypes.

Table of contents

- [1. Introduction](#)
- [2. WSML-Core Element Elements](#)
 - [2.1 Basic WSML-Core Syntax](#)

- [2.2 Ontologies](#)
 - [2.3 Namespace definitions](#)
 - [2.4 Non-functional properties](#)
 - [2.5 Import Ontology Definitions](#)
 - [2.6 Mediators](#)
 - [2.7 Concept definitions](#)
 - [2.8 Relation definitions](#)
 - [2.9 Instance definitions](#)
 - [2.10 Axiom Definitions](#)
 - [3. Mapping WSML-Core to OWL Lite⁻](#)
 - [4. Mapping OWL Lite⁻ to WSML-Core](#)
 - [5. WSML-Core in the WSMO Use Cases](#)
 - [6. Conclusions](#)
 - [References](#)
 - [Acknowledgement](#)
-

1. Introduction

The major goal of the [WSML](#) working group is to develop a formal language for the description of Semantic Web Services based on the [WSMO](#) conceptual model. As is described in deliverable [D16.0 \[Oren, 2004\]](#), there are several WSML languages with different underlying logical formalisms. The two main logical formalisms exploited in different WSML languages are Description Logics [\[Baader et al., 2003\]](#) (exploited in WSML-DL) and Rule Languages [\[Lloyd, 1987\]](#) (exploited in WSML-RL). WSML-Core (described in this document) exploits the intersection of both formalisms.

We introduce WSML-Core, an ontology modeling language, based on the semantics of OWL Lite⁻ [\[de Bruijn et al., 2004\]](#) and the conceptual model of ontologies presented in WSMO [\[Roman et al., 2004\]](#). [Chapter 2](#) describes all the elements of the WSML-Core language. [Chapter 3](#) specifies the mapping from WSML-Core to the OWL Lite⁻ abstract syntax, thereby actually defining the semantics of WSML-Core. [Chapter 4](#) provides a mapping from the OWL Lite⁻ abstract syntax to WSML-Core. [Chapter 5](#) summarizes the applicability of WSML-Core to the use cases of WSMO D3.2 [\[Stollberg et al., 2004\]](#). Finally, we present some conclusions in [Chapter 6](#).

2. WSML-Core Elements

In this chapter we explain the ontology modeling elements in the WSML-Core language. The modeling elements are based on the WSMO conceptual model of ontologies [\[Roman et al., 2004\]](#). Each description is accompanied by an example. Most of the examples were taken from [\[Stollberg et al., 2004\]](#).

The recommended namespace to be used to reference this version of WSML-Core is <http://www.wsmo.org/2004/d16/d16.7/v0.1/>.

In this chapter, we explain the elements of WSML-Core. Each description is accompanied by an example. Most of the examples were taken from [\[Stollberg et al., 2004\]](#).

2.1 Basic WSML-Core Syntax

The syntax for WSMML-Core has a frame-like style, where a collection of information about a class or property is given in one large syntactic construct, instead of being divided into a number of atomic chunks or even being divided into several triples (as when writing using an RDF syntax). It is possible to spread the information about a particular class, relation, instance or axiom over several constructs, but we do not recommend this.

Identifiers in WSMML-Core follow the conventions of WSMO [Roman et al., 2004]. An identifier is either a URI or a Qualified Name (QName). QNames without a prefix are resolved with the default namespace. Argument lists are separated by commas. Statements in WSMML-Core start with a keyword and can be multi-lined. There is no specific end-of-statement syntax -- a keyword for a new statement is sufficient.

2.2 Ontologies

An ontology definition in WSMML-Core starts with the **ontology** keyword followed by a full URI, which serves as the identifier of the ontology and, when there is no explicit target namespace definition, as the target namespace of the definitions in the ontology specification document.

An example:

```
ontology <http://www.wsmo.org/2004/d3/d3.2/v0.1/20040628/resources/po.wsmml>
```

2.3 Namespace definitions

Below the **ontology** URI declaration, there is an optional block of namespace definitions, which is preceded by the **namespace** keyword. The **namespace** keyword is optionally followed by the default namespace and a number of namespace definitions. Each namespace definition, except for the default namespace, consists of the chosen prefix, an '=' and the URI, which identifies the namespace. Finally, the (optional) target namespace is specified with the use of the **target-namespace** keyword. If the target namespace is designated by the URI which is specified by the **ontology** declaration, the **target-namespace** line is redundant and may be omitted.

An example:

```
namespace  
  http://www.wsmo.org/ontologies/purchase#,  
  dc=http://purl.org/dc/elements/1.1#,  
  wsmml=http://www.wsmo.org/2004/d2/v1.0#  
  
target-namespace http://www.wsmo.org/ontologies/purchase#
```

2.4 Non-functional properties

Following the **namespace** definition block is an optional non-functional properties definition block, identified by the keyword **non-functional-properties**. Following the keyword is a list of properties and property values. The recommended properties are the properties of the Dublin Core [Weibel et al. 1998], but the list of properties is extensible and thus the user can choose to use properties coming from different sources. WSMO defines one property, absent in the Dublin Core, namely **version**. The value of each of the properties is either a URI or a literal. If a property has multiple values, these are separated by commas.

An example:

non-functional-properties

dc:title "Purchase ontology"
dc:creator "DERI International"
dc:subject "Buyer", "Seller", "Product", "Price",
"Payment method", "Delivery"
dc:description "General purchase order request ontology"
dc:publisher "DERI International"
dc:contributor "Armin Haller"
dc:date "2004-07-19"
dc:type <http://www.wsmo.org/2004/d2/v0.3/20040329/#ontos>
dc:format "text/plain"
dc:language "en-US"
dc:relation
<http://www.wsmo.org/2004/d3/d3.2/v0.1/20040607/resources/loc.wsml>,
<http://www.wsmo.org/ontologies/dateTime#>
dc:coverage "ID:7029392 Name:World"
dc:rights <http://www.deri.org/privacy.html>
version "Revision: 1.12"

2.5 Import Ontology definitions

Following the **non-functional-properties** definition block is an optional imported ontologies definition block, identified by the keyword **import-ontologies**. Following the keyword is a list of URIs identifying the ontologies being imported. An **import-ontologies** definition serves to merge ontologies, just as an OWL **import** statement does. This means the logical definitions in the imported ontologies are simply appended to the definitions of the importing ontology.

An example:

import-ontologies
<http://www.wsmo.org/ontologies/dateTime#>,
<http://www.wsmo.org/ontologies/currency#>

2.6 Mediators

Mediators are used to import other ontologies. This concept of mediation between ontologies is more flexible than the **import-ontologies** statement, which is used to import an WSMML ontology into another WSMML ontology. The ontology import mechanism simply appends the definitions in the imported ontology to the importing ontology. The importing ontology should contain the axioms to relate the definitions in the imported ontology to the local definitions. This mechanism enforces a strong coupling between the ontologies, which is undesirable in the general case and it does not allow importing parts of ontologies.

By externalizing the mediation from the ontology, WSMO allows loose coupling of ontologies; the mediator is responsible for relating the different ontologies to each other. A mediator can provide, for example, translation services between ontology languages or adjust for argument-order differences.

An example:

usedMediators
<http://www.wsmo.org/2004/d3/d3.2/v0.1/20040628/resources/owlFactbookMediator.wsml>,

2.7 Concept definitions

A concept definition starts with the **concept** keyword, which is followed by the identifier of the concept. This is followed by an optional **non-functional-properties** block, zero or more implicitly anded direct superconcept definitions (using the **subconcept-of** keyword followed by the identifier for a named concept), and zero or more attribute specifications. An attribute specification consists of the identifier of a relation, the **oftype** keyword and the identifier of a concept of which the attribute values must be instances.

A concept can only be a **subconcept-of** named concepts. We do not allow complex concepts definitions as superconcepts, as is allowed in OWL.

An example:

```
concept Human
  non-functional-properties
    dc:description "Members of the species Homo sapiens"
  subconcept-of Primate
  subconcept-of LegalAgent
  parentOf otype Human
```

2.8 Relation definitions

A relation definition starts with the **relation** keyword, which is followed by the identifier of the relation and an optional specialization of a direct super-relation. This is followed by an optional **non-functional-properties** block. In WSM-Core, relations are restricted to binary relations, which correspond with objectproperties in OWL Lite⁻. Parameters can be used to restrict the domain and range of the property (denoted by the **domain** and **range** keywords, respectively). In fact, no other parameters are allowed.

Particular aspects of the relation are denoted by particular keywords. The relation can be a specialization of a different named relation, denoted with the **subrelation-of** keyword. Transitive, symmetric, and inverse properties are denoted with the **transitive**, **symmetric**, and **inverse-of** keywords, respectively. For a complete account, see Table 1 in [Chapter 3](#).

An example:

```
relation hasAncestor subrelation-of hasRelative
  non-functional-properties
    dc:description "Relation between ancestors"
  transitive
  domain otype Person
  range otype Person
```

2.9 Instance definitions

An instance definition starts with the **instance** keyword, followed by the identifier of the instance, the **member-of** keyword and the name of the concept to which the instance belongs. An instance corresponds with an individual in OWL Lite⁻. The

member-of keyword identifies the concept to which the instance belongs. This definition is followed by the attribute values associated with the instance. Each property filler consists of the property identifier, the keyword **hasvalues**^[1] and the value for the attribute.

An example:

```
instance innsbruckHbf member-of station
  name hasvalues IBKHbf
  code hasvalues INN
  locatedIn hasvalues loc:innsbruck
```

The second way of accessing instances, as described in [\[Roman et al., 2004\]](#), is by providing a link to an instance store. Instance stores contain large numbers of instances and they are linked to the ontology with the use of a mediator. In order to distinguish mediators used for linking ontologies and mediators used for linking instance stores, we introduce the keyword **instance-stores** followed by one or more identifiers to indicate the instance stores.

An example:

```
instance-stores
  med:personnel-store,
  med:client-store
```

2.10 Axiom Definitions

An axiom definition starts with the **axiom** keyword, followed by the name (identifier) of the axiom. This is followed by an optional **non-functional-properties** block) and then by a logical expression preceded by the **logical-expression** keyword. The language allowed for the logical expression is detailed in Table 2 in [Chapter 3](#).

An example:

```
axiom example-axiom
  logical-expression
    "lessThan(?xDistance, ?yDistance) <-
      meters(?xDistance, ?xMeters) and
      meters(?yDistance, ?yMeters) and
      lessThan(?xMeters, ?yMeters)."
```

3. Mapping WSML-Core to OWL Lite^ˆ

In this chapter we define the semantics of WSML-Core through a mapping to the OWL Lite^ˆ abstract syntax [\[de Bruijn et al., 2004\]](#), which is a subset of the OWL abstract syntax. The syntax and semantics of the OWL abstract syntax can be found in [\[Patel-Schneider et al., 2004\]](#).

Table 1 shows the mapping between the WSML-Core conceptual syntax and OWL Lite^ˆ. Through the mapping to the OWL abstract syntax, the precise semantics of the WSML-Core primitives is defined. Please note in WSML-Core, each construct can have non-functional properties associated with it. This is not reflected in the table. Rather, there is a separate row in the table, which addresses the non-functional

properties and the way they are reflected in the OWL abstract syntax. The annotation properties occur inside each class, property or instance definition.

We need to make two notes here about the WSML-Core syntax compared to the syntax presented in WSMO-Standard [Roman et al., 2004]. First of all, the attribute definitions in the concept signature (see the second row of Table 1) are interpreted as universal value restrictions in OWL Lite⁻. This does not correspond with the intuition behind these attribute definitions, as was pointed out in [de Bruijn et al., 2004]. Second, we on the one hand restrict the WSMO-Standard syntax for relations by allowing only two parameters (**domain** and **range**) in order to restrict the relations to those of binary arity. On the other hand, we extend the WSMO-Standard syntax with the keywords **subrelation-of**, **transitive**, **symmetric**, and **inverse-of** in order to capture the epistemology of the ObjectProperties in OWL.

In Table 2, we have identified exactly which logical expressions can be translated to OWL Lite⁻. These logical expressions add little^[2] in expressivity over the conceptual syntax, but sometimes the user prefers a different way of modeling axioms. Furthermore, this syntax for logical expressions can be directly used in goal descriptions and capability descriptions of web services for the modeling of assumptions, pre-conditions, effects and post-conditions. All and only those logical expressions that are either in the first column of Table 2 or can be reduced to logical expressions in Table 2 are valid in WSML-Core.

| WSML-Core conceptual syntax | OWL Lite ⁻ Abstract syntax | Remarks |
|---|--|---|
| <i>Logical Definitions</i> | | |
| ontology O | Ontology(O) | All definitions and axioms in the ontology are nested inside the Ontology statement, according to the other mappings in this table. |
| concept C subconcept-of D_1, \dots, D_n P_1 of type C_1 \vdots P_n of type C_n | Class(C partial $D_1 \dots D_n$ restriction(P_1 allValuesFrom C_1) ... restriction(P_n allValuesFrom C_n)) | |
| relation R [subrelation-of R_1, \dots, R_n] [transitive] [symmetric] [inverse-of (R_0)] [domain of type C_1, \dots, C_n] [range of type D_1, \dots, D_n] | ObjectProperty(R super(R_1) ... super(R_n) [inverseOf(R_0)] [Symmetric] [Transitive] domain(C_1) ... domain(C_n) range(D_1) ... range(D_n)) | |
| instance o member-of C_1, \dots, C_n A_1 has values o_1 \vdots | Individual(o type(C_1) ... type(C_n) value(A_1 o_1) ... value(A_n o_n)) | |

| | | |
|--|--|---|
| A_n hasvalues o_n | | |
| <i>Extra-Logical definitions</i> | | |
| non-functional-properties $P_1 v_1$ \cdot \cdot $P_n v_n$ [version v_0] | $\text{annotation}(P_1 v_1)$ \cdot \cdot $\text{annotation}(P_n v_n)$ [$\text{annotation}(\text{owl:versionInfo } v_0)$] | Annotation properties are nested inside other definitions, such as ontology, individual, class and property definitions. For some strange reason, if annotations occur on the ontology level, they should be written with a capital 'A' (e.g. $\text{Annotation}(\text{owl:versionInfo } "\$Revision: 1.49 \$")$). |
| import-ontologies $O_1,$ \cdot \cdot O_n | | The OWL abstract syntax does not provide a construct for the import of ontologies, although the RDF/XML serialization does provide this facility through the import statement. |
| usedMediators $M_1,$ \cdot \cdot M_n | | OWL does not have the concept of a mediator. Therefore, this construct cannot be translated to OWL. |
| Table 1: Mapping between WSMML-Core and OWL Lite ⁻ abstract syntax | | |

| WSMML-Core Logical Expression | OWL Lite ⁻ Abstract Syntax | Remarks |
|--|---|---------|
| C subconcept-of D | Class (C partial D) | |
| ?x member-of D <-> ?x member-of C ₁ and ... and ?x member-of C _n | Class (D complete C ₁ ... C _n) | |
| C[R ofType D] | Class (C partial restriction(R allValuesFrom D)) | |
| R subrelation-of S | ObjectProperty (R super(S)) | |
| ?x member-of C <- R(?x,?y) | ObjectProperty (R domain(C)) | |

| | | |
|--|--|---|
| ?y member-of C <- R(?x,?y) | ObjectProperty (R range(S)) | |
| S(?y,?x) <- R(?x,?y) R(?y,?x) <- S(?x,?y) | ObjectProperty (R inverseOf(S)) | |
| R(?y,?x) <- R(?x,?y) | ObjectProperty (R Symmetric) | |
| R(?x,?y) <- R(?x,?y) and R(?y,?z) | ObjectProperty (R Transitive) | |
| o member-of C [R ₁ hasvalues o ₁ , : R _n hasvalues o _n] | Individual(o type(C) value(R ₁ o ₁) ... value(R _n o _n) | Both the member-of C and all the property values are optional. |
| Table 2: Mapping between WSMML-Core logical expressions and OWL Lite ⁻ abstract syntax | | |

4. Mapping OWL Lite⁻ to WSMML-Core

Table 3 shows the mapping between OWL Lite⁻ (abstract syntax) and WSMML-Core. The table shows the corresponding conceptual syntax and any additional logical expression that might be necessary to capture the semantics of the OWL Lite⁻ statement. The table shows for each construct supported by OWL Lite⁻ the WSMML-Core syntax in terms of the conceptual model and additional logical expressions necessary to capture the construct.

As we can see from Table 3, all of OWL Lite⁻ can be captured in WSMML-Core. Most of the axioms can be captured with the conceptual model. However, some axioms rely on additional logical expressions.

| OWL Lite ⁻ Abstract syntax | WSMML-Core conceptual syntax | WSMML-Core logical expression |
|---|---|-------------------------------|
| <i>Logical Definitions</i> | | |
| Class(C partial D ₁ ... D _n) | concept C subconcept-of D_i (in case D _i is a named class) P _i of type C_i (in case D _i is a value restriction of the form P _i allValuesFrom C _i) | |

| | | |
|--|---|---|
| Class(C complete $D_1 \dots D_n$) | concept C subconcept-of D_1, \dots, D_n | ? x member-of C <- ? x member-of D_1 and ... and ? x member-of D_n |
| EquivalentClasses($C_1 \dots C_n$) | | ? x member-of C_i <- $1 = i = n$? x member-of C_j $1 = j = n$ |
| ObjectProperty(R super(R_1) ... super(R_n) domain(C_1) ... domain(C_n) range(D_1) ... range(D_n)) | relation R subrelation-of R_1, \dots, R_n domain of type C_1, \dots, C_n range of type D_1, \dots, D_n | |
| [inverseOf(R_0)] | relation R inverse-of(R_0) | |
| [Symmetric] | relation R symmetric | |
| [Transitive] | relation R transitive | |
| EquivalentProperties(R_1 ... R_n) | | $R_i(?x,?y)$ <- $1 \leq i \leq n$ $R_j(?x,?y)$ $1 \leq j \leq n$ |
| Individual(o type(C_1) ... type(C_n) value(R_1 o_1) ... value(R_n o_n)) | instance o member-of C_1, \dots, C_n R_1 hasvalues o_1 . . R_n hasvalues o_n | |
| <i>Extra-Logical Definitions</i> | | |
| annotation(P v) | non-functional-properties P v | |

Table 3: Mapping between OWL Lite⁻ abstract syntax and WSML-Core

5. WSML-Core in the WSMO Use Cases

This chapter enumerates the listings in the WSMO Use Cases document [Stollberg et al., 2004] and identifies whether the listing is in WSML-Core and if not, why it is not. Furthermore, we describe reductions, which would be necessary to make the listing fall inside WSML-Core. Note that WSML-Core does not distinguish between single-valued and set-valued attributes. Note also that in the evaluation we do not take into account inter-dependencies between the ontologies.

When evaluating the goal and web service descriptions, we only focus on the logical expressions, which are used for the assumptions, pre-conditions, effects and

post-conditions.

Note that we have not evaluated the mediator descriptions, because they were not available in enough detail.

In the reason why a particular listing is not in WSML-Core we distinguish:

data types

WSML-Core does not support the use of datatypes, such as strings or integers.

data type predicates

Because the use of special datatype predicates, such as '>' or '<' for the domain of integers, is quite common we distinguish this special category in the list. Another reason for including this feature in the list is that certain popular ontology languages (e.g. OWL) do not support datatype predicates.

mixing abstract and concrete domains

A concrete domain corresponds with the domain of a datatype. The abstract domain corresponds with the elements defined in the ontology. The separation between abstract and concrete domains stems from the separation of these domain in Description Logic languages and the Description Logic-based ontology language OWL.

integrity constraints

Integrity constraints can be used in axiom definitions to express arbitrary restrictions.

equality

| Listing | In WSML-Core? | Why not? | Steps to make it fall inside WSML-Core |
|---|---------------|---|---|
| <i>Ontologies</i> | | | |
| 1: Domain Ontology "International Train Ticket" | no | - data types - data type predicates - integrity constraints - equality | - Remove all (3) axioms - Replace occurrences of data types with abstract concepts |
| 2: Domain Ontology "Date and Time" | no | - data types - data type predicates - mixing abstract and concrete domains - integrity constraints - equality | - Complete remodeling of ontology |
| 3: Domain Ontology "Purchase" | no | - data types | - Replace occurrences of data types with abstract concepts |
| 4: Domain Ontology "Locations" | no | - data types - data type predicates - mixing abstract and concrete | - Replace occurrences of data types with abstract concepts - Remove all axioms |

| | | | |
|---|----|---|---|
| | | domains - integrity constraints - equality (and also '<' and '>'; in the head!) | |
| <i>Goals and Web Services</i> | | | |
| 5: Goal - buying a train ticket online | no | - data types | - Replace occurrences of data types with abstract concepts |
| 6: ÖBB Web Service for Booking Online Train Tickets for Austria and Germany | no | - data types - data type predicates - equality | - Removing all restrictions written in the preCondition and the postCondition |

It turns out that in the listings in [\[Stollberg et al., 2004\]](#) data types are often used, as well as data type predicates. An extension of WSML-Core with, for example, data type groups [\[Pan and Horrocks, 2004\]](#), could solve most of the problems.

In many of the listings in the use cases document, abstract concepts are mixed with datatypes. This seems to be done mostly because it was necessary to create user-defined data types (cf. [\[Pan and Horrocks, 2004\]](#)). If user-defined data types were to be introduced, a lot of this mixing of domains could be easily eliminated. Furthermore, it turns out that very often, integrity constraints are used. An extension with integrity constraints can solve this problem.

6. Conclusions

In this deliverable we have introduced the syntax of WSML-Core, which is based on the WSMO conceptual model for ontologies [\[Roman et al., 2004\]](#). We have given a precise characterization of the semantics through a mapping to the OWL abstract syntax [\[Patel-Schneider et al., 2004\]](#).

From the evaluation of the ontologies developed in the WSMO use cases [\[Stollberg et al., 2004\]](#) we have seen that there are still some things lacking in WSML-Core, which are essential for the modeling of ontologies. The main elements lacking in WSML-Core are datatypes and datatype predicates. WSML-Core currently does not have these feature because it is based on OWL Lite⁻ [\[de Bruijn et al., 2004\]](#) and OWL Lite⁻ lacks these features. However, [\[de Bruijn et al., 2004\]](#) already proposed datatype groups [\[Pan and Horrocks, 2004\]](#) as an extension to OWL Lite⁻, which can then be integrated in WSML-Core.

References

[Baader et al., 2003] F. Baader, D. Calvanese, and D. McGuinness: *The Description Logic Handbook*, Cambridge University Press, 2003.

[de Bruijn et al., 2004] J. de Bruijn, A. Polleres, and D. Fensel. *OWL lite⁻*. Deliverable d20v0.1, WSML, 2004. Available from

<http://www.wsmo.org/2004/d20/v0.1/>.

[Dean et al., 2004] M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein: *OWL Web Ontology Language Reference*, W3C Recommendation, 10 February 2004. Available at <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>.

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

[Oren, 2004] E. Oren (eds.): *Languages for WSMO*, WSML deliverable D16.0 version 0.1 Working Draft 2 August 2004. available from <http://www.wsmo.org/2004/d16/d16.0/v0.2/20040802/>.

[Pan and Horrocks, 2004] J. Z. Pan and I. Horrocks, I.: *OWL-E: Extending OWL with expressive datatype expressions*. IMG Technical Report IMG/2004/KR-SW-01/v1.0, Victoria University of Manchester, 2004. Available from <http://dl-web.man.ac.uk/Doc/IMGTR-OWL-E.pdf>.

[Patel-Schneider et al., 2004] P. F. Patel-Schneider, P. Hayes, and I. Horrocks: *OWL web ontology language semantics and abstract syntax*. Recommendation 10 February 2004, W3C, 2004. Available from <http://www.w3.org/TR/owl-semantics/>.

[Roman et al., 2004] D. Roman, H. Lausen (eds.): *Web Service Modeling Ontology - Standard (WSMO - Standard)*, WSMO deliverable D2 version 1.0 Working Draft 29 July 2004. available from <http://www.wsmo.org/2004/d2/v1.0/20040729/>.

[Stollberg et al., 2004] M. Stollberg, H. Lausen, A. Polleres, and R. Lara (eds.): *WSMO Use Case Modeling and Testing*, WSMO d3.2v0.1, version 2004-06-28. available from <http://www.wsmo.org/2004/d3/d3.2/v0.1/20040628/>.

[Weibel et al. 1998] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf: *RFC 2413 - Dublin Core Metadata for Resource Discovery*, September 1998.

Acknowledgement

The work is funded by the European Commission under the projects [DIP](#), [Knowledge Web](#), [SEKT](#), [SWWS](#), [Esperanto](#), and [h-TechSight](#); by [Science Foundation Ireland](#) under the [DERI-Lion](#) project; and by the Vienna city government under the [CoOperate](#) program.

The editors would like to thank to all the [members of the WSML working group](#) for their advice and input into this document.

[1]Note that in WSML-Core there is no keyword **hasValue** for single-valued attributes, because there are not cardinality restrictions in WSML-Core. Only set-valued attributes are allowed through the **hasValues** keyword.

[2]The only expressivity added by the logical expressions over the conceptual syntax is the complete class definition, which is an equivalence relation between the defined class and the intersection (conjunction) of the defining classes.

\$Date: 2004/08/02 10:08:20 \$