Rules for the Semantic Web

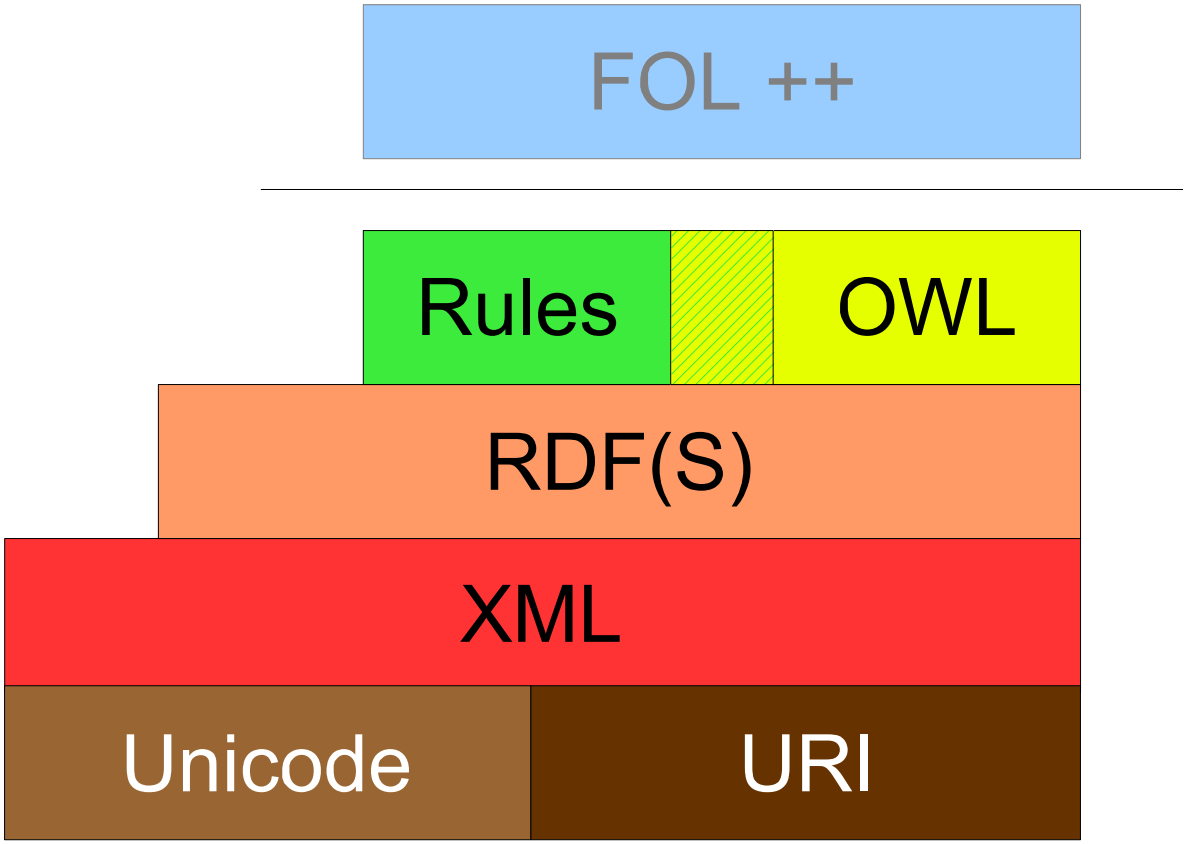
The WSML Approach

Jos de Bruijn
jos.debruijn@deri.org

Digital Enterprise Research Institute (DERI)
National University of Ireland, Galway
University of Innsbruck, Austria

April 27, 2005

Web Rule Language in its Context

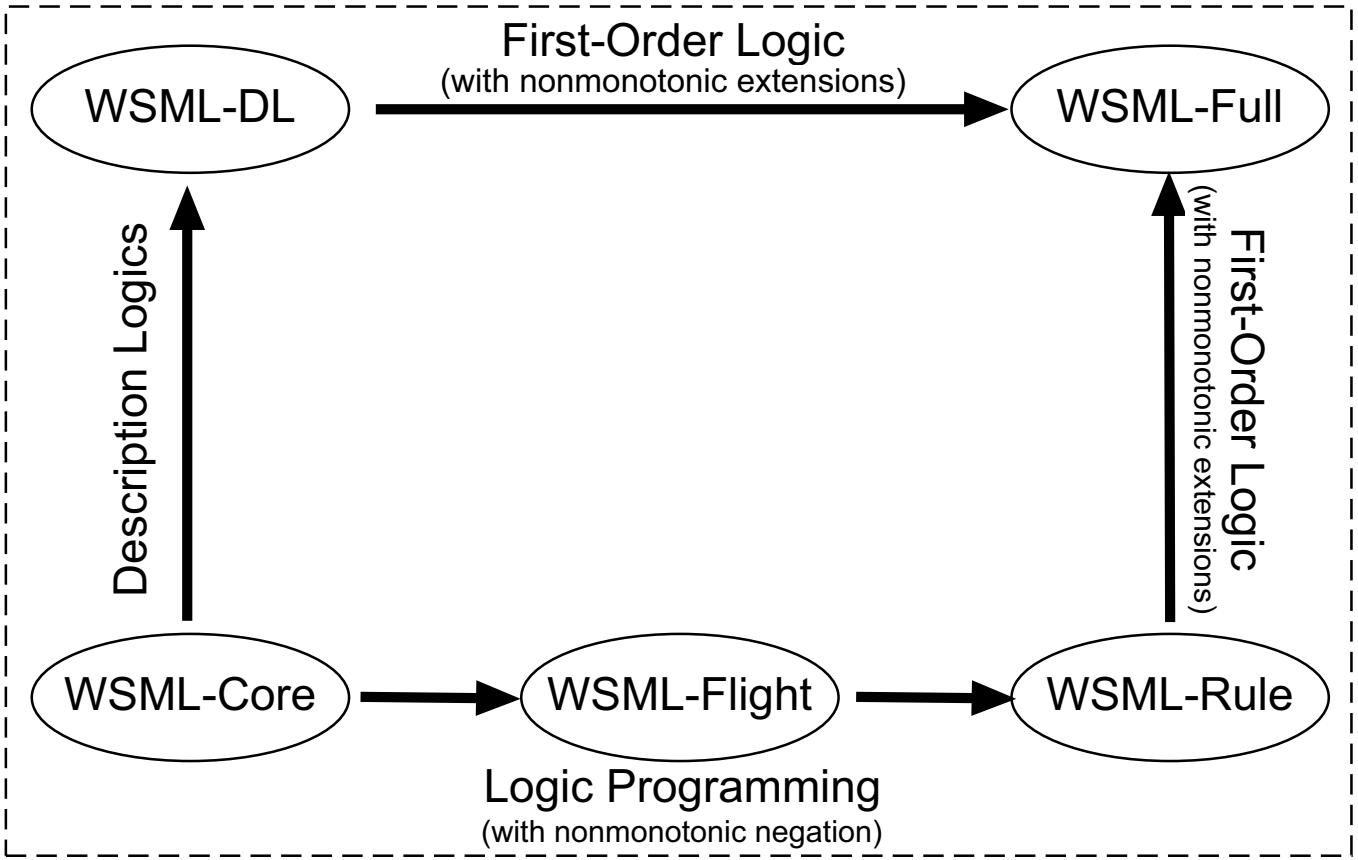


Outline

The Web Service Modeling Language WSML

- A language for the Semantic description of Web Services
- Based on the Web Service Modeling Ontology WSMO
  - Ontologies
  - Web Services
  - Goals
  - Mediators
- Disregard Web Service-related aspects in this presentation
- Semantics based on well-known logical language paradigms:
  - Description Logics
  - Logic Programming
  - Frame Logic
- WSML distinguishes between:
  - Conceptual modeling
  - Logical expressions

WSML Language Variants



Prologue

By Example

```
// Specification of the WSML variant
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"
// Namespace prefix declaration
namespace {_"http://www.example.org/example#",
  dc _"http://purl.org/dc/elements/1.1/"}
// WSML specifications
ontology _"http://www.example.org/exampleOntology"
 [...]
goal _"http://www.example.org/exampleGoal"
 [...]
```

etc...

WSML Specification

A WSML specification has the following structure:

- Type of specification (Ontology/Web Service/Goal/Mediator)
- Header
  - Non-Functional Properties
  - Imported Ontologies
  - Used Mediators
- Content of the specification

Ontologies

Header

[.. prologue ..]

ontology _"http://www.example.org/ontologies/example"

nonFunctionalProperties

dc#title **hasValue** "WSML example ontology"

endNonFunctionalProperties

importsOntology {_"http://www.wsmo.org/ontologies/location"}

usesMediator {_"http://www.wsmo.org/mediators/"}

Ontologies

Concepts

- Form the basic terminology of the domain of discourse
- May be organized in a hierarchy (using **subConceptOf**)
- Has a number of attributes:
  - Attributes have a type:
    - Type constraint (**ofType**)
    - Type inference (**impliesType**)
  - Attributes may have cardinality constraints
  - Attributes may have a number of features:
    - Transitive
    - Symmetric
    - Reflexive
    - Inverse of another attribute

Ontologies

Concepts - example

```
concept Person subConceptOf {Primate, LegalAgent}
  nfp
 // Related axiom
     dc#relation hasValue personUncle
   endnfp
 // A functional attribute (maximal cardinality=1)
    hasName ofType (0 1) _string
 // hasParent is the inverse of hasChild
    hasChild inverseOf(hasParent) ofType Person
    hasParent ofType Person
    hasBrother ofType Person
```

Ontologies
Relations

- Inspired by relations in mathematics
- Have arbitrary arity
- May have typing associated with its arguments
- May be organized in a hierarchy (using **subRelationOf**)

relation Marriage (ofType Person, ofType Person, ofType _date)
nfp
dc#description **hasValue** "Marriage is a relation between two persons, which are the participants in the marriage, and the date in the marriage."
endnfp

Ontologies
Instances

- Are the objects in the domain
- May be member of one or more concepts
- May have a number of attribute values associated with it

instance john **memberOf** Person
nfp
dc#description **hasValue** "The person John Smith"
endnfp
hasName **hasValue** "John Smith"

Ontologies
Relation Instances

- Are tuples in a relation

relationInstance Marriage(john,mary,_date(2005,03,03))
nfp
dc#description **hasValue** "John and Mary married on 2005-03-03."
endnfp

Ontologies
Axioms

- Refine concept and relation definitions in Ontologies using logical expressions
- Add arbitrary knowledge and constraints
- Entry point for logical expressions, rules in ontology
- Allowed logical expressions depend on WSML variant

```
axiom personUncle
nfp
  dc#description hasValue "The brother of a person's parent is that
    person's uncle."
endnfp
definedBy
  ?x[hasUncle hasValue ?z] impliedBy ?x[hasParent hasValue ?y] and
    ?y[hasBrother hasValue ?z].
```

Logical Expression syntax

- Used for refining Ontologies and specifying Web Service functionality
- Allow to use the full expressive power of the underlying logic
- Frame syntax (F-Logic)
- Logic Programming constructs
  - Negation-as-failure
  - LP implication
- Variables are implicitly universally quantified outside the formula
- Symbols resemble natural language and are unambiguous
- WSML variants restrict allowed logical expressions

Examples

// a simple rule ; the brother of someone's parent is that person's
// uncle
?x[hasUncle **hasValue** ?z] :– ?x[hasParent **hasValue** ?y] **and**
?y[hasBrother **hasValue** ?z].

// the same person cannot be both a man and a woman (constraint)
!– ?x **memberOf** Man **and** ?x **memberOf** Woman.

// every person has a father
?x **memberOf** Person **implies exists** ?y (?x[father **hasValue** ?y]) .

WSML XML Syntax

- Syntax for exchange over the Web
- Translation between human-readable and XML syntax
- XML Schema for WSML has been defined

WSML XML

Example

```
<!ENTITY ex "http://www.example.org/ontologies/example#">
<!ENTITY wsml "http://www.wsmo.org/wsml/wsml-syntax#">
<wsml xmlns="&wsml;"
 variant="http://www.wsmo.org/wsml/wsml-syntax/wsml-flight">
  <importsOntology>
    http://www.wsmo.org/ontologies/location
  </importsOntology>
  <concept name="&ex;Person">
    <nonFunctionalProperties>[..]</nonFunctionalProperties>
    <attribute name="&ex;hasName" type="constraining">
      <range>&wsml;string</range>
      <maxCardinality>1</maxCardinality>
    </attribute>
      [..]
  </concept>
    [..]
</wsml>
```

WSML RDF Syntax

- Interoperability with RDF applications
- Maximal reuse of RDF and RDFS vocabulary
- WSML RDF includes most of RDF
- Translation between human-readable and RDF syntax
- For logical expressions, XML literals are used

WSML RDF

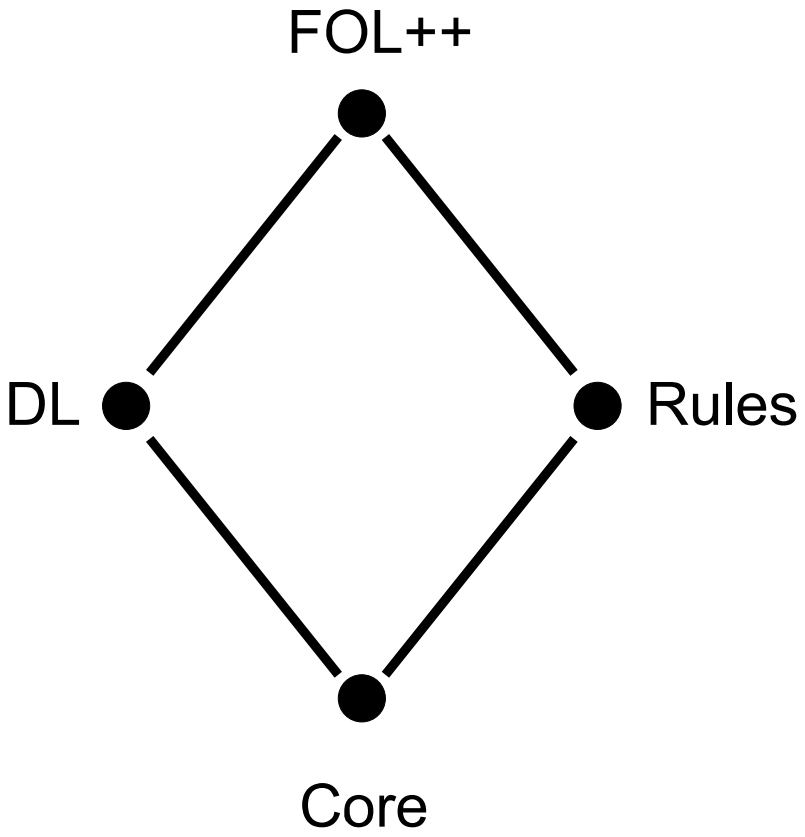
Example

```
<http://www.example.org/ontology> rdf#type wsml#ontology
<http://www.example.org/ontology> wsml#variant
    <http://www.wsmo.org/wsml/wsml-syntax/wsml-flight>
<http://www.example.org/ontology> wsml#nfp _:nfp1
_:nfp1 dc#title "WSML example ontology"^^xsd#string
<http://www.example.org/ontology> wsml#importsOntology
    <http://www.wsmo.org/ontologies/location>
<http://www.example.org/ontology> wsml#hasConcept ex#Person
ex#Person wsml#hasAttribute _:att1
_:att1 wsml#attribute ex#hasName
_:att1 wsml#ofType xsd#string
_:att1 wsml#maxCardinality "1"^^xsd:integer
<http://www.example.org/ontology> wsml#hasAxiom
    ex#personUncle
ex#personUncle rdfs#isDefinedBy
    "<impliedByLP>..</impliedByLP>"^^rdf#XMLLiteral
```

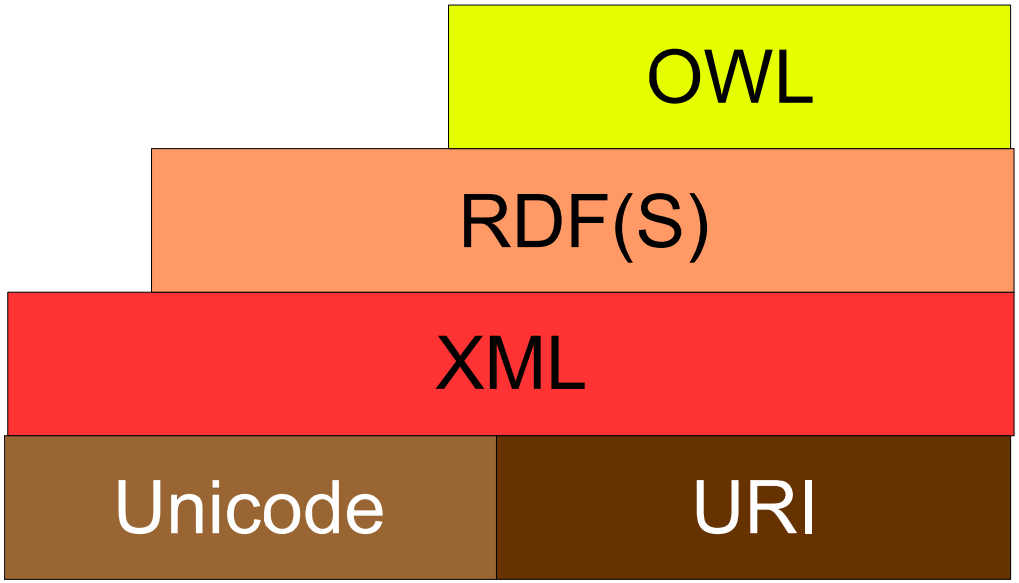
Key Features of WSML

- One framework for a set of Layered Languages
- Normative, Human-readable Syntax
- Separation of conceptual modeling and logical expressions
- Semantics based on well-known formalisms
- Relation between DL and Rules through common subset
- Web Language
- Frame-based syntax

The WSML Approach to language layering



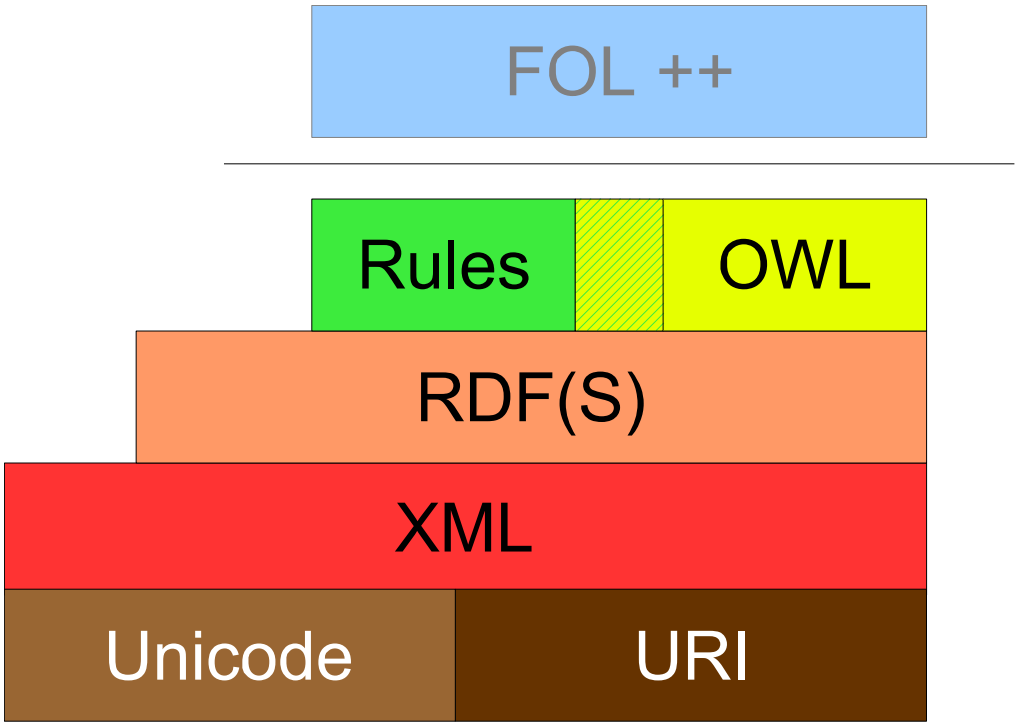
Current Languages on the Semantic Web



How to Incorporate rules?

- Layering Rules on top of OWL (e.g. SWRL)
- Hybrid approach (e.g. CARIN/AL-Log)
- Using a common subset for interoperation (e.g. DLP)

Common subset



- Maintain nice properties of each of the underlying logics
- Reuse existing implementations of rules and description logic
- Allow straightforward extension in both directions

Conclusions

WSML position on a Rules language for the Web:

1. Relation between DL and Rules through common subset
2. Rules-based ontology language
3. Ontology meta-model independent from underlying logic
4. Separation of conceptual modeling and logical expressions
5. Normative, Human-readable Syntax
6. Semantics based on well-known formalisms; allows integration with existing systems
7. Web Language
8. Frame-based syntax

WSML resources

http://www.wsmo.org/wsml/wsml-syntax#

Web Rule Language in its Context