# The Web Service Modeling Language WSML

## An Overview

Jos de Bruijn[1], Holger Lausen[1], Axel Polleres[1,2] and Dieter Fensel[1]

[1] Digital Enterprise Research Institute (DERI)
{jos.debruijn,holger.lausen,dieter.fensel}@deri.org

[2] Universidad Rey Juan Carlos
axel.polleres@urjc.es

ESWC2006

# Outline

# Outline

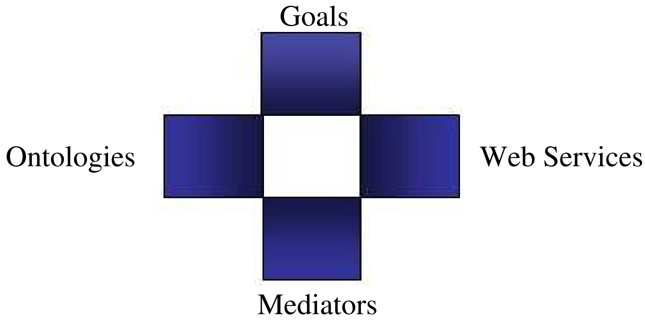
# Outline

# The Web Service Modeling Ontology WSMO

Introduction

- An ontology for Semantic Web Services
- Provides conceptual model for SWS
- Based on the Web Service Modeling Framework WSMF
- Principles of WSMO:
  - Ontology-based descriptions
  - Strict decoupling of components
  - Strong mediation between components
  - Interface vs. Implementation

# The Web Service Modeling Ontology WSMO



# The Web Service Modeling Ontology WSMO

Ontologies

- Provide terminology for:
  - Data exchanged between service requesters and providers
  - Description of other WSMO elements
- Ontologies consist of:
  - Concepts
    - Attributes
  - Relations
  - Functions
  - Instances
  - Axioms

# The Web Service Modeling Ontology WSMO

Web Service descriptions

- Functionality offered by the Web Service
- Functional description, in the form of a *capability*:
  - Assumptions
    - Cannot be checked
    - Usually indicate dependency on real world
  - Preconditions
    - Conditions over the input
  - Effects
    - Changes in the real world as a result of execution of the Web Service
  - Postconditions
    - Relation between the input and the output

# The Web Service Modeling Ontology WSMO

Web Service descriptions (cont'd)

- Behavioral description, in the form of an *interface*:
  - Choreography
    - How to interact with the service
  - Orchestration
    - Use of external Web Service to realize the functionality
  - Both choreography and orchestration are decompositions of the capability

# The Web Service Modeling Ontology WSMO

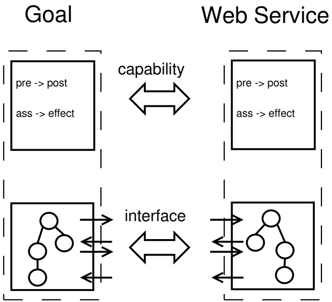
Goals

- Functionality requested from the Web Service
- Description symmetric to Web Service description:
  - Capability
  - Interface

# The Web Service Modeling Ontology WSMO

Goals

- Functionality requested from the Web Service
- Description symmetric to Web Service description:
  - Capability
  - Interface



# The Web Service Modeling Ontology WSMO

Mediators

- Connect heterogeneous components
- Resolve heterogeneity in different levels
  - Data - differences in data representation
  - Protocol - differences in interaction styles
  - Process - differences in business processes

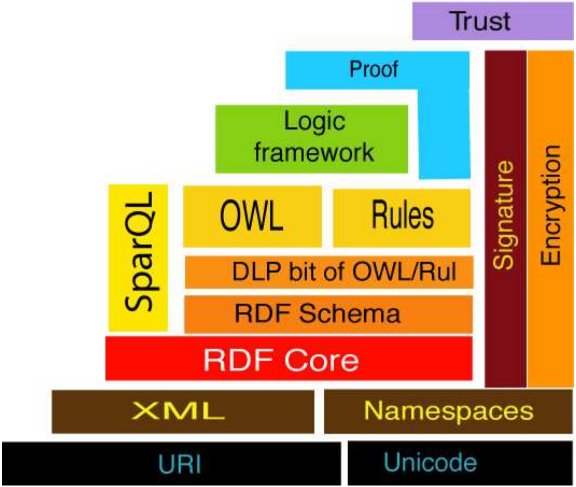
# The Web Service Modeling Ontology WSMO

Types of Mediators

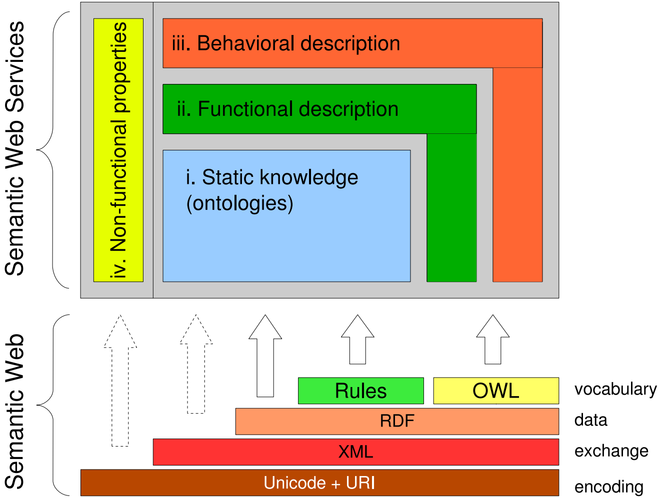
- OO Mediators
  - Connect ontologies to any other component (including mediators)
  - Resolve mismatches conflicts between ontologies
- WW Mediators
  - Link Web Services to services they depend on
  - Resolve representation differences through OO Mediators
- WG Mediators
  - Link Goals and Web Services
  - Resolve differences in data, protocol and process between requester and provider
- GG Mediators
  - Connect generic and refined Goals

# Outline

# Semantic Web Languages



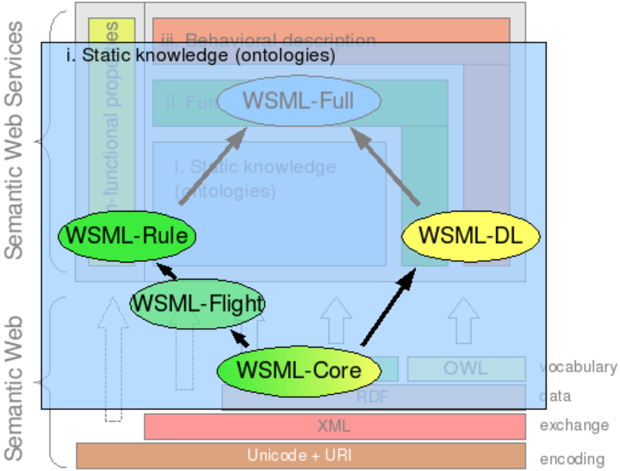
# Semantic Web Service Languages



# The Web Service Modeling Language WSML

1. A language for the Semantic description of Web Services
2. Based on the Web Service Modeling Ontology WSMO
3. One syntactic framework for a set of layered languages
4. Normative "human-readable" surface syntax
5. Separation of
   - Conceptual modeling
   - Logical modeling
6. Semantics based on well known formalisms
   - Description Logics
   - Logic Programming
   - Frame Logic
7. Web language
8. Frame-based syntax

# WSML Variants



# Why not use OWL?

# Why not use OWL?

- Semantic Web is not only about Description Logics!

# Why not use OWL?

- Semantic Web is not only about Description Logics!
- "Inferring style" restrictions of OWL not useful in all settings

# Why not use OWL?

- Semantic Web is not only about Description Logics!
- "Inferring style" restrictions of OWL not useful in all settings
- WSML investigates use of
  - Logic Programming
  - Description Logics

  in common framework

# Why not use OWL?

- Semantic Web is not only about Description Logics!
- "Inferring style" restrictions of OWL not useful in all settings
- WSML investigates use of
  - Logic Programming
  - Description Logics

  in common framework
- WSML-DL close to OWL DL

# Why not use OWL?

- Semantic Web is not only about Description Logics!
- "Inferring style" restrictions of OWL not useful in all settings
- WSML investigates use of
  - Logic Programming
  - Description Logics

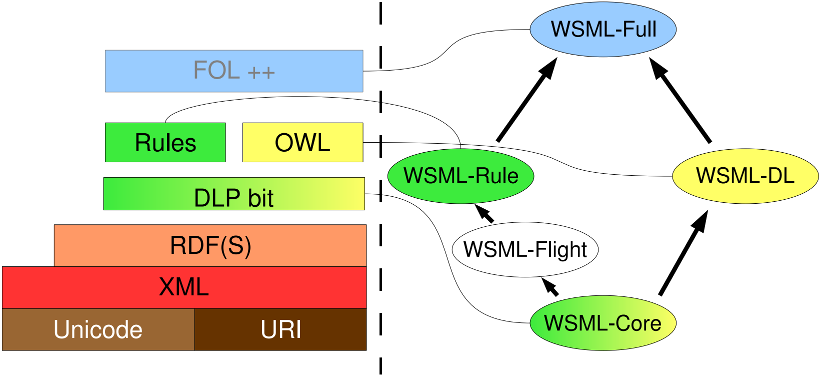
  in common framework
- WSML-DL close to OWL DL
- Interoperation between LP and DL through common subset

# Why not use OWL?

- Semantic Web is not only about Description Logics!
- "Inferring style" restrictions of OWL not useful in all settings
- WSML investigates use of
  - Logic Programming
  - Description Logics

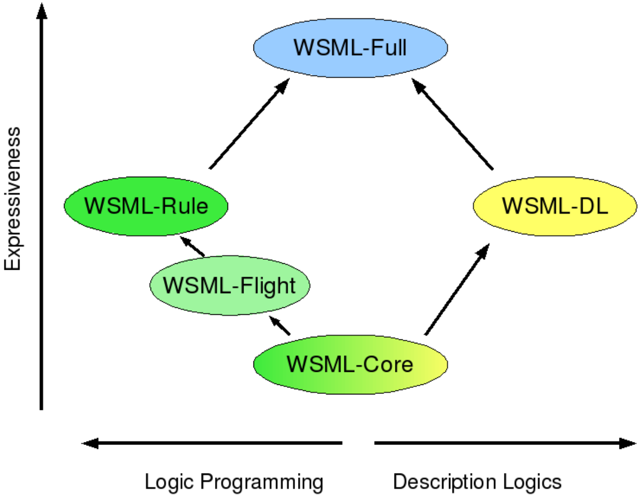
  in common framework
- WSML-DL close to OWL DL
- Interoperation between LP and DL through common subset
- Expressive integration of DL and LP topic of ongoing research (Eiter et al., KR2004; Rosati, KR2006)

# WSML and the Semantic Web



# Outline

# WSML Language Variants



# WSML-Core

- Basic interoperability layer between Description Logics and Logic Programming paradigms
- Based on Description Logic Programs
  - Expressive intersection of Description Logic $\mathcal{SHIQ}$ and Datalog
  - Allows to take advantage of many years of established research in Databases and Logic Programming
  - Allows reuse of existing efficient Deductive Database and Logic programming reasoners
- Some limitations in conceptual modeling of Ontologies
  - No cardinality constraints
  - Only "inferring" range of attributes
  - No meta-modeling

# WSML-Core Logical Expressions

- Limitations in logical expressions
  - From Description Logic point-of-view, there is a lack of:
    - Existentials
    - Disjunction
    - (Classical) negation
    - Equality
  - From Logic Programming point-of-view, there is a lack of:
    - N-ary predicates
    - Chaining variables over predicates
    - (Default) negation
    - Function symbols

# WSML-DL

- Extension of WSML-Core
- Based on the Description Logic $\mathcal{SHIQ}$
  - Entailment is decidable
  - Close to DL species of Web Ontology Language OWL
  - Many efficient subsumption reasoners
- Some limitations in conceptual modeling of Ontologies
  - No cardinality constraints
  - Only "inferring" range of attributes
  - No meta-modeling
- Limitations in logical expressions
  - From Logic Programming point-of-view, there is a lack of:
    - N-ary predicates
    - Chaining variables over predicates
    - (Default) negation

# WSML-Flight

- Extension of WSML-Core
- Based on the Datalog, with negation under Perfect Model Semantics
  - Ground entailment is decidable
  - Allows to take advantage of many years of established research in Databases and Logic Programming
  - Allows reuse of existing efficient Deductive Database and Logic programming reasoners
- No limitations in conceptual modeling of Ontologies
  - Cardinality constraints
  - Value constraints for attributes
  - Meta-modeling

# WSML-Flight Logical Expressions

- Syntax based on Datalog fragment of F-Logic, extended with negation-as-failure
- Arbitrary Datalog rules:
  - N-ary predicates
  - Chaining variables over predicates
- From Description Logic point-of-view, there is a lack of:
  - Existentials
  - Disjunction
  - (Classical) negation
  - Equality
- From Logic Programming point-of-view, there is a lack of:
  - Function symbols

# WSML-Rule

- Extension of WSML-Flight
- Based on Horn fragment of F-Logic, with negation under Perfect Model Semantics
  - Ground entailment is undecidable
  - Turing complete
  - Allows to take advantage of many years of established research in Logic Programming
  - Allows reuse of existing efficient Logic programming reasoners
- Extends WSML-Flight logical expressions with:
  - Function symbols
  - Unsafe rules
- From Description Logic point-of-view, there is a lack of:
  - Existentials
  - Disjunction
  - (Classical) negation
  - Equality

# WSML-Full

- Extension of WSML-Rule *and* WSML-DL
- Based on First Order Logic with nonmonotonic extensions
  - Entailment is undecidable
  - Very expressive
- Extends WSML-DL logical expressions with:
  - Chaining variables over predicates
  - Function symbols
  - Nonmonotonic negation
  - N-ary predicates
- Extends WSML-Rule with:
  - Existentials
  - Disjunction
  - Classical negation
  - Equality
- Specification of WSML-Full is open research issue

# Outline

# Identifiers

- Internationalized Resource Identifiers (IRIs) are basic identifiers
  - Concepts, attributes, relations, instances, etc... are all IRIs
  - IRI is successor of URI
  - Using in newer W3C recommondations, e.g., XML, RDF
  - e.g., _"http://www.wsmo.org/wsml/wsml-syntax#", _"http://example.org/myOntology#myConcept"
- sQNames
  - Abbreviations for IRIs ("serialized QNames")
  - e.g., wsml#concept, dc#title, ont#location
- Data values
  - Elementary data values: strings, int, decimals
  - Structured data values
    - Derived from XML Schema Datatypes
    - date, float, etc...
    - e.g., _date(2005,6,23), _float(12.567)

# Prologue

By Example

**wsmlVariant** _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

**namespace** {_"http://www.example.org/example#",
dc _"http://purl.org/dc/elements/1.1/"}

**ontology** _"http://www.example.org/exampleOntology"
[...]
**goal** _"http://www.example.org/exampleGoal"
[...]

*etc...*

# Prologue

By Example

```
// Specification of the WSML variant
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

namespace {_"http://www.example.org/example#",
  dc _"http://purl.org/dc/elements/1.1/"}


ontology _"http://www.example.org/exampleOntology"
 [...]
goal _"http://www.example.org/exampleGoal"
 [...]
```

*etc...*

# Prologue

By Example

```
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/"}

ontology _"http://www.example.org/exampleOntology"
 [...]
goal _"http://www.example.org/exampleGoal"
 [...]
```

*etc...*

# Prologue

By Example

**wsmlVariant** _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"

**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

# Outline

# 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/"}
```

# 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

# 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
```

# 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**

# 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"

# 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**

# Axioms

- Refine concept and relation definitions in Ontologies using logical expressions
- Add arbitrary knowledge and constraints
- 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].
```

# Web Services

A Web Service specification has the following structure:

- Type of specification (**webService**) and identifier
- Header
  - Non-Functional Properties
  - Imported Ontologies
  - Used Mediators
- Capability
  - Functional description of Web Service
- Interfaces
  - Behavioural description of Web Service
  - Communications pattern of Web Service

**webService** _"http://www.example.org/exampleService"
  **capability** ...

  **interface** ...

# Capability

- Syntactical framework for Functional description
- Functionality defined through logical expressions:
  - Preconditions
  - Postconditions
  - Assumptions
  - Effects
- Shared variables
  - Variables shared by description elements
  - Quantified over the entire capability

# Capability

Example

```
capability
 sharedVariables ?x,?y,...
 precondition
   definedBy
     ..
 postcondition
   definedBy
     ..
 assumption
   definedBy
     ..
 effect
   definedBy
     ..
```

# Interfaces

- Choreography
  - Communication interface of Web Service
- Orchestration
  - Usage of external Web Services
- Currently, choreography and orchestration are external to WSML

**interface**
**choreography** _"http://example.org/choreographies/1"
**orchestration** _"http://example.org/orchestration/1"

# Goals

- Describe requested functionality
- Description symmetric to Web Services:
  - Header
  - Capability
  - Interfaces

**goal** _"http://www.example.org/exampleGoal"

**capability**

...

**interface**

...

# Mediators

- Mediators connect WSML elements in two ways:
  - Referencing mediators through **usesMediator**
  - Specifying **source** and **target** in mediator specification
- Mediation is achieved by mediation service (**usesService**)
  - Web Service
  - Goal

**wgMediator** _"http://www.example.org/exampleMediator"

**source** _"http://www.example.org/exampleGoal"

**target** _"http://www.example.org/exampleService"

**usesService** _"http://www.example.org/mediationService"

# Outline

# Logical Expression syntax

- Used for refining Ontologies and specifying Web Service functionality
- Allow to use the full expressive power of the underlying logic
- First-Order Logic with Frame syntax (F-Logic)
- Specific extensions to capture 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

# Examples

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

# Examples

```
// a simple rule; the brother of someone's parent is that person's
//   uncle
?x[hasUncle hasValue ?z] impliedBy ?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.
```

# Examples

```
// a simple rule; the brother of someone's parent is that person's
//   uncle
?x[hasUncle hasValue ?z] impliedBy ?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]).
```

# Examples

```
// a simple rule; the brother of someone's parent is that person's
//    uncle
?x[hasUncle hasValue ?z] impliedBy ?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]).

// a person is either a Man or a Woman
?x memberOf Person implies ?x memberOf Man or ?x memberOf Woman.
```

# WSML Variants vs. Features

| Feature | Core | DL | Flight | Rule | Full |
|---|---|---|---|---|---|
| Classical Negation (**neg**) | - | X | - | - | X |
| Existential Quantification | - | X | - | - | X |
| Disjunction | - | X | - | - | X |
| Meta Modeling | - | - | X | X | X |
| Default Negation (**naf**) | - | - | X | X | X |
| LP implication | - | - | X | X | X |
| Integrity Constraints | - | - | X | X | X |
| Function Symbols | - | - | - | X | X |
| Unsafe Rules | - | - | - | X | X |

# Outline

# Outline

# 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>
```

# Outline

# 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
```

# Outline

# Conclusions

- WSML is a language for modeling of Semantic Web Services
- Based on the Web Service Modeling Ontology WSMO
- WSML is a Web language:
  - IRIs for object identification
  - XML datatypes
- WSML is based on well-known logical formalisms:
  - Description Logics
  - Logic Programming
  - Frame Logic
- Syntax has two parts:
  - Conceptual modeling
  - Arbitrary logical expressions
- XML and RDF syntaxes for exchange over the Web

# Ongoing and Future Work

- Integration of LP and DL
  - Incorporation in WSML framework
- WSML-Full semantics
  - First-Order Autoepistemic Logic
- RDF Representation of WSML
- Semantics of Functional Description
- Language for Behavioral Description
- Uses of Non-Functional Properties
- Grounding to existing Web Service Standards

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

# Questions?