The Web Service Modeling Language WSML

An Overview

Jos de Bruijn
jos.debruijn@deri.org

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

June 15, 2005

Outline

Outline

Introduction

Recap of WSMO

WSML Language Variants

WSML Syntax

WSML Exchange Syntaxes

Conclusions

The World Wide Web

What is the WWW?

- Largest document repository ever ($>$ 8 billion Web pages indexed by Google)
- Highly distributed
  - Millions of publishers
  - No control over consistency of published content
- Web Technologies
  - HTTP for transferring documents
  - HTML for marking up documents
  - URI for addressing documents
- Most content on the Web is in natural language (HTML)
  - Natural language not suitable for machine reading
  - Current Web is "syntactic"
  - Problems in automatically:
    - Retrieving documents
    - Extracting relevant information from retrieved documents
    - Combining information from different sources

The Semantic Web

- Making the Web machine-readable
- Publishing data in machine-readable format
- Relating data on the Web to established vocabularies (ontologies)
- Ontologies specified in formal language to allow reasoning
- Ontologies enable automation in:
  - Retrieval of relevant information
  - Extracting relevant information from retrieved document
  - Combination of information from different sources (as long as they are related to the same ontology)

Web Services

- Next step in software engineering:
  - 1960s: Procedural
  - 1980s: Object Orientation
  - 1990s: Component-based
  - 2000s: Web Services
- Loosely coupled, reusable components
- Add new level of functionality to the Web
- Web Service Technologies
  - SOAP for accessing Web Services
  - WSDL for describing Web Services
  - UDDI for publishing and looking up Web Services

Web Services are not enough

- Like the current Web, Web Services are “syntactic”
- No automation in:
  - Finding services
  - Selecting services
  - Negotiation with service provider
  - Composing services
  - Executing services

Combining Semantic Web and Web Services
Semantic Web Services

- Semantic Web + Web Services = Semantic Web Services
- Using Semantic Web technologies to describe Web Services
- Enable automation in:
  - Publication
  - Discovery
  - Selection
  - Composition
  - Mediation
  - Execution

The Web Service Modeling Language WSML

1. A language for the Semantic description of Web Services

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

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

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

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

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

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

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

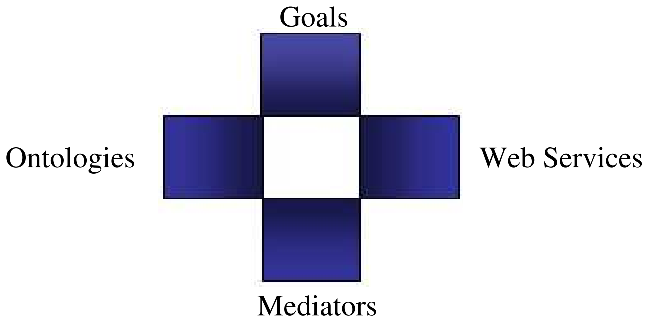
Outline

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)

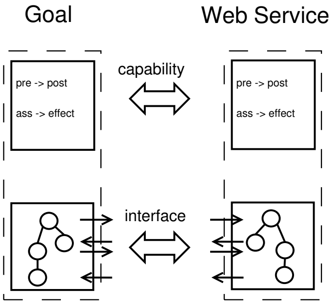
- Behavioral description, in the form of an *interface*:
  - Choreography
    - How to interact with the service
  - Orchestration
    - Use of external Web Service to realize the functionality
  - Both choreography and orchestration are decompositions of the capability

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

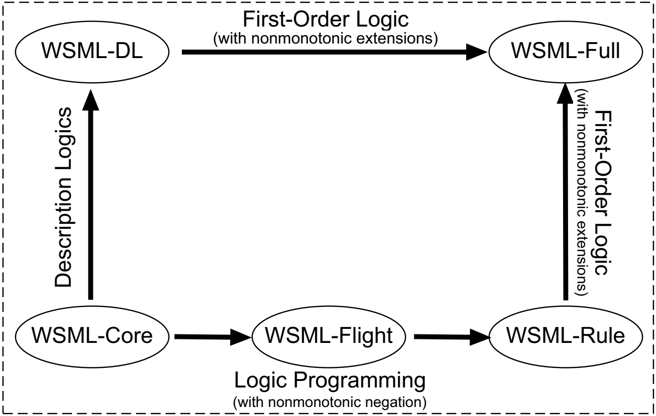
The Web Service Modeling Ontology WSMO

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

WSML Language Variants



First Order Logic - Syntax
Symbols

Constants	$a, b, john, \ldots$
Function symbols	$f, g, +, married - to, \ldots$
Predicate Symbols	$p, q, >, marriage, \ldots$
Variables	$x, y, \ldots$
Connectives	$\neg, \wedge, \vee, \leftarrow, \rightarrow, \leftrightarrow$
Quantifiers	$\forall, \exists$
(Equality)	$=$

Terms

- Every constant is a term
  - $a, b, john$
- Every variable is a term
  - x, y
- If f is an n-place function symbol and $t_1, ..., t_n$ are terms, then $f(t_1, ..., t_n)$ is a term
  - $f(x), f(a), f(g(a))$
  - $father - of(john), married - to(mary)$

Atomic formulas

- If p is an n-place predicate symbol and $t_1, ..., t_n$ are terms, then $p(t_1, ..., t_n)$ is an atomic formula
  - $p(x), q(f(a), y)$
  - $marriage(father - of(john), mary, date(2005, 4, 6))$
- If t_1, t_2 are terms, then $t_1 = t_2$ is an atomic formula
  - $f(x) = a, married - to(mary) = father - of(john)$

Formulas

- Any atomic formula is a formula
- If A, B are formulas and $x_1, ..., x_n$ are variables then:
  - $\neg A$ is a formula
  - $A \wedge B$ is a formula
  - $A \vee B$ is a formula
  - $A \leftarrow B$ is a formula
  - $A \rightarrow B$ is a formula
  - $A \leftrightarrow B$ is a formula
  - $\forall x_1, ..., x_n : A$ is a formula
  - $\exists x_1, ..., x_n : A$ is a formula
- Examples:
  - $\forall x, y, d : marriage(x, y, d) \rightarrow married - to(x) = y \wedge married - to(y) = x$
  - $\forall x : number(x) \rightarrow \exists y : y > x$

Horn subset

- A Horn formula is a disjunction of literals with one positive literal, with all variables universally quantified:
  - $(\forall)\neg B_1 \vee ... \vee \neg B_n \vee H$
- Can be written as an implication:
  - $(\forall)B_1 \wedge ... \wedge B_n \rightarrow H$
- Horn formulas are the basis for Logic Programming

First-Order Logic - Semantics
Interpretation

- The meaning of a First-Order formula is assigned using an *interpretation*
- An interpretation $\mathcal{I}$ consists of:
  - Domain Δ: a set of objects
  - A set of relations R: $\Delta^1 \times ... \times \Delta^n$
  - A set of functions F: $\Delta^1 \times ... \times \Delta^n \mapsto \Delta$
  - A mapping function $\cdot$ which:
    - Maps constants to objects
    - Maps predicate symbols to relations
    - Maps function symbols to functions
- An interpretation is a *model* of a formula A if it makes the formula *true*:
  - $\mathcal{I} \models A$

Truth of a formula

A (atomic formula)	is true iff	$A^{\mathcal{I}}$ is in the model
$\neg A$	is true iff	$A^{\mathcal{I}}$ is *not* true
$A \wedge B$	is true iff	$A^{\mathcal{I}}$ *and* $B^{\mathcal{I}}$ are true
$A \vee B$	is true iff	$A^{\mathcal{I}}$ *or* $B^{\mathcal{I}}$ is true (or both)
$A \rightarrow B$	is true iff	in every case where $A^{\mathcal{I}}$ is true, $B^{\mathcal{I}}$ is true

What about variables?

- We have not discussed semantics of variables

What about variables?

- We have not discussed semantics of variables
- Variables *have no semantics*

What about variables?

- We have not discussed semantics of variables
- Variables *have no semantics*
- What to do with variables?

What about variables?

- We have not discussed semantics of variables
- Variables *have no semantics*
- What to do with variables?
- *Assign* values to variables using an assignment B
  - e.g., $\{x \mapsto a, y \mapsto john\}$
- An interpretation $\mathcal{I}$ makes a formula A *true* under a variable assignment B:
  - $\mathcal{I} \models_B A$

What about variables?

- We have not discussed semantics of variables
- Variables *have no semantics*
- What to do with variables?
- *Assign* values to variables using an assignment B
  - e.g., $\{x \mapsto a, y \mapsto john\}$
- An interpretation $\mathcal{I}$ makes a formula A *true* under a variable assignment B:
  - $\mathcal{I} \models_B A$
- Quantifiers:
  - $\exists x A$: there exists an assignment for x which makes A true
  - $\forall x A$: for all possible assignments of x, A is true

Logic Programming - Syntax

- Any FOL term is a term in LP
- Any FOL atomic formula is an atomic formula in LP
- Any Horn formula is a rule in LP (quantification usually omitted)
  - $H \leftarrow B_1 \wedge \ldots \wedge B_n$
- Logic programming is a syntactic subset of FOL

Logic Programming - Syntax

- Any FOL term is a term in LP
- Any FOL atomic formula is an atomic formula in LP
- Any Horn formula is a rule in LP (quantification usually omitted)
  - $H \leftarrow B_1 \wedge \ldots \wedge B_n$
- Logic programming is a syntactic subset of FOL
- Note! Negation-as-failure in LP is an *extension* of Horn rules
  - $\neg \neq \mathit{not}$

Logic Programming - Semantics

Herbrand Universe and Herbrand Base

- The Herbrand Universe U_P is the set of all ground terms which can be formed from constants and function symbols in program P.
  Example:
  `a,b,f(a),f(b),f(f(a)),f(f(b)),f(f(f(a))),...`
- The Herbrand Base B_P is the set of all ground atoms which can be built from predicate symbols in P, using ground terms from U_P as arguments.
  Example: `p(a),p(b),q(a),q(b),p(f(a)),q(f(a)),...`

Logic Programming - Semantics

Herbrand Interpretation and Least Herbrand Model

- A Herbrand Interpretation IP is a subset of the Herbrand Base BP.
- A Herbrand Model MP is a Herbrand Interpretation which makes every formula true, i.e.:
  - Every fact in P is in MP, and
  - For every rule R in P holds: if every positive literal in the body is in MP, then also the head literal is in MP.

  Note: this only works for positive programs, i.e., programs without negation!
- The semantics of a program P is characterized in terms of the least Herbrand Model, which is the intersection of all possible Herbrand Models.
- Each positive program has one unique least Herbrand Model.

Relationship between FOL and LP

- Semantics LP defined in terms of minimal Herbrand model
  - Only one minimal model
- Semantics FOL defined in terms of First-Order models
  - Typically, infinitely many First-Order models
- The minimal Herbrand model is a First-Order model
- In fact, every Herbrand model is a First-Order model
- There exist First-Order models which are not Herbrand models

Entailment in FOL and LP

Entailment in FOL and LP

- General First-Order entailment:
  - $\phi \models \psi$ iff for every interpretation $\mathcal{I}$: if $\mathcal{I} \models \phi$ then $\mathcal{I} \models \psi$
  - Thus, the set of models of ϕ $M(\phi)$ is a subset of $M(\psi)$: $M(\phi) \subseteq M(\psi)$
  - e.g., $p(x) \land q(x) \models p(x)$

Entailment in FOL and LP

- General First-Order entailment:
  - $\phi \models \psi$ iff for every interpretation $\mathcal{I}$: if $\mathcal{I} \models \phi$ then $\mathcal{I} \models \psi$
  - Thus, the set of models of ϕ $M(\phi)$ is a subset of $M(\psi)$: $M(\phi) \subseteq M(\psi)$
  - e.g., $p(x) \wedge q(x) \models p(x)$
- Ground entailment:
  - $\phi \models \psi_{ground}$ iff for every interpretation $\mathcal{I}$: if $\mathcal{I} \models \phi$ then $\mathcal{I} \models \psi_{ground}$ and $\psi ground$ **does not** contain variables
  - e.g., $(p(x) \rightarrow q(x)) \wedge p(a) \models q(a)$

Entailment in FOL and LP

- General First-Order entailment:
  - $\phi \models \psi$ iff for every interpretation $\mathcal{I}$: if $\mathcal{I} \models \phi$ then $\mathcal{I} \models \psi$
  - Thus, the set of models of ϕ $M(\phi)$ is a subset of $M(\psi)$: $M(\phi) \subseteq M(\psi)$
  - e.g., $p(x) \wedge q(x) \models p(x)$
- Ground entailment:
  - $\phi \models \psi_{ground}$ iff for every interpretation $\mathcal{I}$: if $\mathcal{I} \models \phi$ then $\mathcal{I} \models \psi_{ground}$ and $\psi ground$ **does not** contain variables
  - e.g., $(p(x) \rightarrow q(x)) \wedge p(a) \models q(a)$
- Logic Programming only defines ground entailment

Entailment in FOL and LP

- General First-Order entailment:
  - $\phi \models \psi$ iff for every interpretation $\mathcal{I}$: if $\mathcal{I} \models \phi$ then $\mathcal{I} \models \psi$
  - Thus, the set of models of ϕ $M(\phi)$ is a subset of $M(\psi)$: $M(\phi) \subseteq M(\psi)$
  - e.g., $p(x) \wedge q(x) \models p(x)$
- Ground entailment:
  - $\phi \models \psi_{ground}$ iff for every interpretation $\mathcal{I}$: if $\mathcal{I} \models \phi$ then $\mathcal{I} \models \psi_{ground}$ and $\psi ground$ **does not** contain variables
  - e.g., $(p(x) \rightarrow q(x)) \wedge p(a) \models q(a)$
- Logic Programming only defines ground entailment
- Horn Logic (i.e., Horn subset of FOL) is equivalent to Logic Programming wrt. ground entailment
  - For any set of Horn formulas ϕ and a ground Horn formula ψ_{ground}: $\phi \models_{FOL} \psi_{ground}$ iff $\phi \models_{LP} \psi_{ground}$
  - $\models_{FOL}$ is classical First-Order entailment; $\models_{LP}$ is LP entailment

Description Logics

- Most DLs similar to 2-variable fragment of FOL
  - No more than 2 variables under the scope of a quantifier
    - *Exception: transitive properties*
  - Classes correspond to unary predicates
  - Properties correspond to binary predicates
  - No function symbols
- Most DLs are *decidable*
- We focus on $\mathcal{SHIQ}$ DL (close to the DL underlying OWL DL), and disregard concrete domains (e.g., int, string) for now
- $\mathcal{SHIQ} =$
  - Concept hierarchies
  - Concept conjunction, disjunction, negation
  - Rule hierarchies
  - Existential, universal quantification
  - Qualified number restrictions (minimal, maximal cardinality)
  - Symmetric, inverse, transitive properties

Description Logics

- Most DLs similar to 2-variable fragment of FOL
  - No more than 2 variables under the scope of a quantifier
    - *Exception: transitive properties*
  - Classes correspond to unary predicates
  - Properties correspond to binary predicates
  - No function symbols
- Most DLs are *decidable*
- We focus on $\mathcal{SHIQ}$ DL (close to the DL underlying OWL DL), and disregard concrete domains (e.g., int, string) for now
- $\mathcal{SHIQ} =$
  - Concept hierarchies
  - Concept conjunction, disjunction, negation
  - Rule hierarchies
  - Existential, universal quantification
  - Qualified number restrictions (minimal, maximal cardinality)
  - Symmetric, inverse, transitive properties

$\mathcal{SHIQ}$ - Syntax
Concept descriptions

$C, D \longrightarrow A \mid$	(atomic concept)
$\top \mid$	(universal concept)
$\bot \mid$	(bottom concept)
$C \sqcap D \mid$	(intersection)
$C \sqcup D \mid$	(disjunction)
$\neg C \mid$	(negation)
$\forall R.C \mid$	(value restriction)
$\exists R.C \mid$	(existential quantification)
$\geqslant nR.C \mid$	(minimal cardinality)
$\leqslant nR.C \mid$	(maximal cardinality)

$\mathcal{SHIQ}$ - Syntax
Individual assertions

$a \in C$
$\langle a, b \rangle \in R$

$\mathcal{SHIQ}$ - Syntax
Axioms

$C \sqsubseteq D$	(class subsumption)
$C \equiv D$	(equivalence)
$Q \sqsubseteq R$	(property subsumption)
$R \equiv Q^-$	(inverse roles)
$R \equiv R^-$	(symmetric roles)
$R^+ \sqsubseteq R$	(transitive properties)

$\mathcal{SHIQ}$ Examples

- $Human \sqsubseteq \forall hasChild.Human \sqcap = 2 hasParent.Human$
- $Parent \sqsubseteq \exists hasChild.\top$
- $HumanParent \equiv Human \sqcap Parent$
- $hasChild \equiv hasParent^{-}$

$\mathcal{SHIQ}$ Examples

- $Human \sqsubseteq \forall hasChild.Human \sqcap = 2 hasParent.Human$
- $Parent \sqsubseteq \exists hasChild.\top$
- $HumanParent \equiv Human \sqcap Parent$
- $hasChild \equiv hasParent^{-}$

if $\langle john, mary \rangle \in hasChild$

$\mathcal{SHIQ}$ Examples

- $Human \sqsubseteq \forall hasChild.Human \sqcap = 2 hasParent.Human$
- $Parent \sqsubseteq \exists hasChild.\top$
- $HumanParent \equiv Human \sqcap Parent$
- $hasChild \equiv hasParent^{-}$

if $\langle john, mary \rangle \in hasChild$ then $\langle mary, john \rangle \in hasParent$

Mapping $\mathcal{SHIQ}$ to FOL

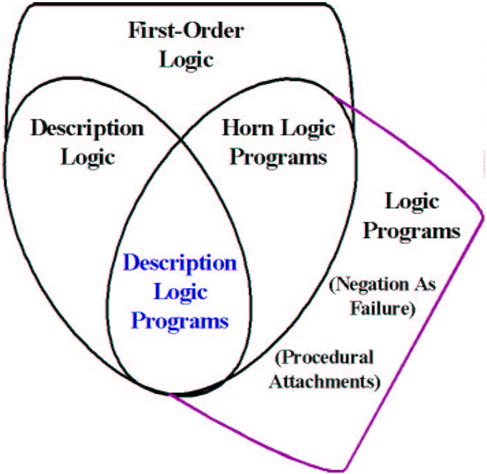
A (atomic concept)	$A(x)$
$\top$	$\top$
$\bot$	$\bot$
$C \sqcap D$	$tr(C) \wedge tr(D)$
$C \sqcup D$	$tr(C) \vee tr(C)$
$\neg C$	$\neg tr(C)$
$\forall R.C$	$\forall y : R(x,y) \rightarrow tr(C,y)$
$\exists R.C$	$\exists y : R(x,y) \wedge tr(C,y)$
$\geqslant nR.C$	$\exists y_1, \ldots, y_n : \bigwedge R(X, y_i) \wedge \bigwedge tr(C, y_i) \wedge \bigwedge y_i \neq y_j$
$\leqslant nR.C$	$\forall y_1, \ldots, y_{n+1} : \bigwedge R(X, y_i) \bigwedge tr(C, y_i) \wedge \rightarrow \bigvee y_i =$

Mapping $\mathcal{SHIQ}$ to FOL

$a \in A$	$A(a)$
$\langle a, b \rangle \in R$	$R(a, b)$

$C \sqsubseteq D$	$\forall x : tr(C, x) \rightarrow tr(D, x)$
$C \equiv D$	$\forall x : tr(C, x) \leftrightarrow tr(D, x)$
$Q \sqsubseteq R$	$\forall x, y : Q(r, y) \rightarrow R(x, y)$
$R \equiv Q^{-}$	$\forall x, y : R(x, y) \leftrightarrow Q(y, x)$
$R^{+} \sqsubseteq R$	$\forall x, y, z : R(x, y) \wedge R(y, z) \rightarrow R(x, z)$

Relation between DL and LP



Description Logic Programs

- "Intersection" of Description Logics and Logic Programming
- That part of Description Logics (OWL in particular) which can be translated to a Logic Program
- Horn Logic subset of $\mathcal{SHIQ}$, *reduced* to a Logic Program: Description Logic Program: DLP

Description Logic Programs

- "Intersection" of Description Logics and Logic Programming
- That part of Description Logics (OWL in particular) which can be translated to a Logic Program
- Horn Logic subset of $\mathcal{SHIQ}$, *reduced* to a Logic Program: Description Logic Program: DLP
- General idea:
  1. Translate $\mathcal{SHIQ}$ axiom to First-Order Logic
  2. Rewrite to Horn Logic
     - If rewriting not possible: formula not in DLP
  3. Reduce to Logic Program

WSML-Core

WSML-Core

- Basic interoperability layer between Description Logics and Logic Programming paradigms

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

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

WSML-Core Logical Expressions

- Limitations in logical expressions
  - From Description Logic point-of-view, there is a lack of:
    - Existentials
    - Disjunction
    - (Classical) negation
    - Equality

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

WSML-DL

- Extension of WSML-Core

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

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

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

WSML-Flight

- Extension of WSML-Core

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

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

WSML-Flight Logical Expressions

- Syntax based on Datalog fragment of F-Logic, extended with negation-as-failure

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

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

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

WSML-Rule

- Extension of WSML-Flight

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

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

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

WSML-Full

- Extension of WSML-Rule *and* WSML-DL

WSML-Full

- Extension of WSML-Rule *and* WSML-DL
- Based on First Order Logic with nonmonotonic extensions
  - Entailment is undecidable
  - Very expressive

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

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

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

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"

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

Table by Holger Lausen

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>
```

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

Conclusions

- WSML is a language for modeling of Semantic Web Services
- Based on the Web Service Modeling Ontology WSMO

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

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

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

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

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

Questions?