Reasoning with WSML

Barry Bishop
STI - Innsbruck

SOA4All Kick-off -Madrid, 25th-27th March 2008

SOA4All contributing to NESSI

Introduction

- What is WSML?
- Layering
- Examples
- Reasoning software

WSML

The Web Services Modelling Language[1] (WSML) is based on WSMO

- Ontologies
- Web Services
- Goals
- Mediators

[1] http://www.wsmo.org/TR/d16/d16.1/v0.3/

Ontologies

Ontologies capture domain specific knowledge:

- Concepts – hierarchy, attributes, instances
- Relations
- Axioms

Example - Concepts

```
concept Person subConceptOf {Animal, LegalEntity}
          name ofType _string

          // parent is the inverse of child
          child inverseOf( parent ) ofType Person

          parent ofType Person
          sibling ofType Person

instance john memberOf Person
          name hasValue "John Smith"
```

Example - Relations and Axioms

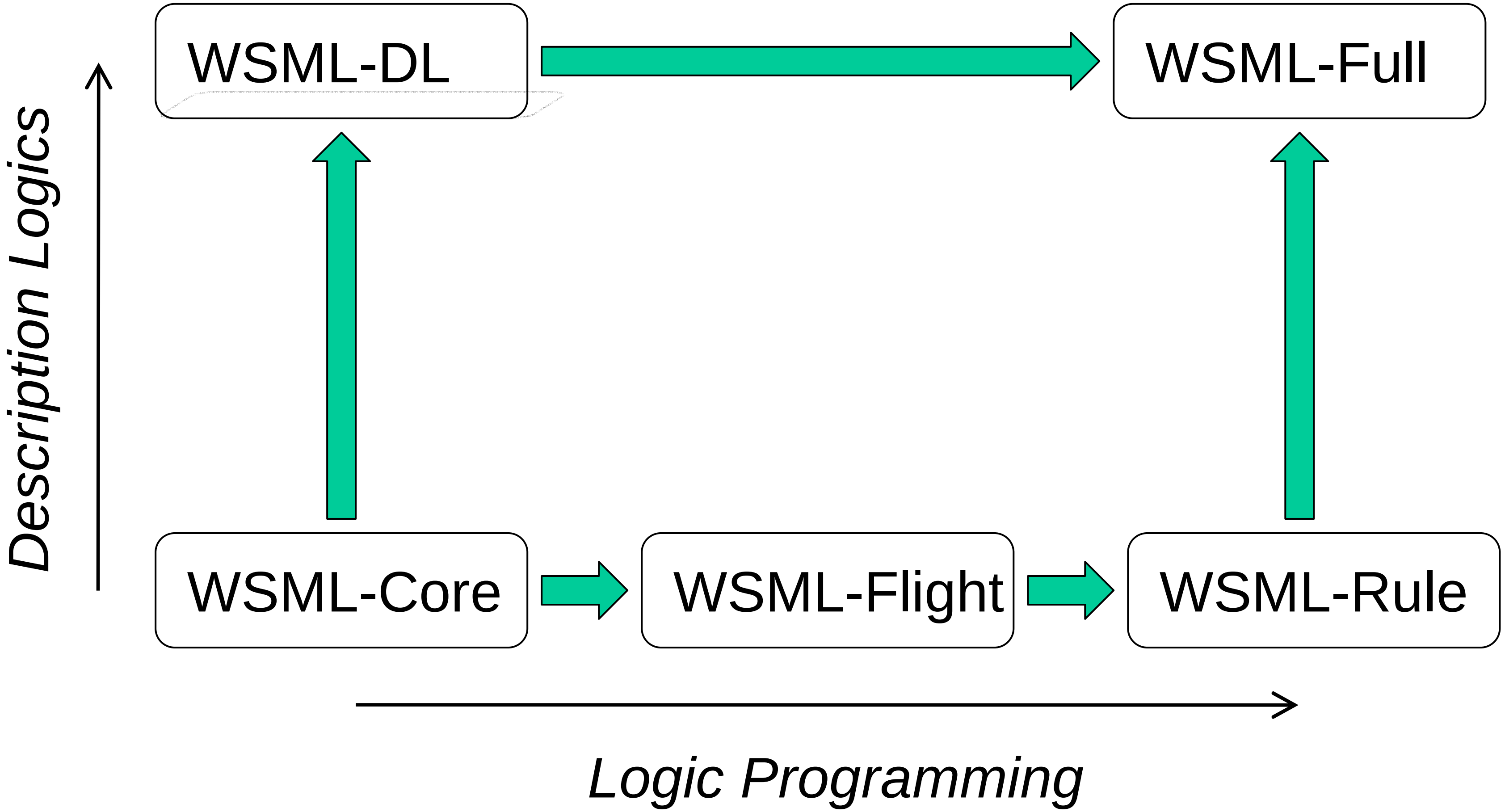
```
relation Marriage (ofType Person, ofType Person, ofType date)

relationInstance Marriage( john, mary, date(2005,03,03) )

axiom personUncle definedBy
            ?x[uncle hasValue ?z]
            impliedBy
            ?x[ parent hasValue ?y ] and
            ?y[ sibling hasValue ?z ] and
            ?z memberOf Male.
```

WSML Variants

WSML is a framework for a set of layered languages

- All variants share the same conceptual syntax
- Vary on the expressiveness of their logical syntax
- Limitations on expressivity related to:
  - Knowledge representation paradigm
  - Complexity of reasoning
- Logical expressions are used to:
  - Refine ontologies (axioms)
  - Describe web services and goals (capabilities)

WSML Variants



WSML-Core

Intersection of logic programming and description logics

- Concepts
- Attributes
- Binary relations
- Instances
- Concept and relation hierarchies
- Data types (XML schema)
- Decidable!

WSML-Core Example

```
wsmlVariant _"http://www.wsmo.org/../wsml-core"
namespace { _"http://www.audio.org/player#" }

ontology MusicPlayers

concept AudioSystem
       isBroken ofType _string

concept iPod subConceptOf AudioSystem

instance myIPod memberOf iPod
       isBroken hasValue "yes"

instance yourIPod memberOf iPod
```

WSML-Core Example

Axiom:

```
axiom fixing definedBy
      ?x[isBroken hasValue ?y]
      implies
      ?x memberOf NeedsFixing.
```

Query:

```
?x memberOf AudioSystem and
      ?x memberOf NeedsFixing.
```

WSML-DL

Extends the expressivity of WSML-Core

- Captures the Description Logic SHIQ(D)
  - Classical negation
  - Disjunction and existential quantification in rule heads
- WSML-DL ontology can directly import OWL-DL
- Reasoning tasks include:
  - Consistency
  - Satisfiability
  - Checking entailment
  - Schema reasoning
- Decidable!

WSML-DL Example

Human is defined as the disjunction between Man and Woman:

```
axiom human definedBy
       ?x memberOf Human equivalent
       ?x memberOf Woman or ?x memberOf Man.
```

Every person has a father:

```
axiom father definedBy
       ?x memberOf Person implies
       exists ?y (?x[ father hasValue ?y]).
```

WSML-Flight

Extension of WSML-Core, equivalent to Datalog with inequality and (locally) stratified default negation

- Powerful rule-based language
- Constraints (integrity, attribute cardinality/range)
- Negation as failure
- Meta-modelling
- Decidable!

WSML-Flight Example

Axiom:

```
axiom isSingle definedBy
      ?x[family_status hasValue single]
      :-
      ?x memberOf Human and
      naf ( ?x[married_to hasValue ?y] ).
```

Integrity constraint:

```
!− ?x memberOf Man and ?x memberOf Woman.
```

WSML-Flight Example

Cardinality constraint:

```
concept Person subConceptOf {Animal, LegalEntity}
            // A functional attribute (maximal cardinality=1)
            name ofType (0 1) _string
```

WSML-Rule

Extends WSML-Flight with:

- Function symbols
- Unrestricted use of variables (unsafe rules)
- Unstratified rules

WSML-Rule Example

Extends WSML-Flight with:

- function symbols

```
?x[father hasValue f(?x)] :- ?x memberOf Person.
```

- unrestricted use of variables (unsafe rules)

```
?x[knowsAbout hasValue ?y] :- ?x memberOf Expert.
```

- unstratified rules

```
?x memberOf Political :- naf ?x memberOf Hippy.
?x memberOf Hippy :- ?x memberOf GreenPeace and
      ?x memberOf Political.
```

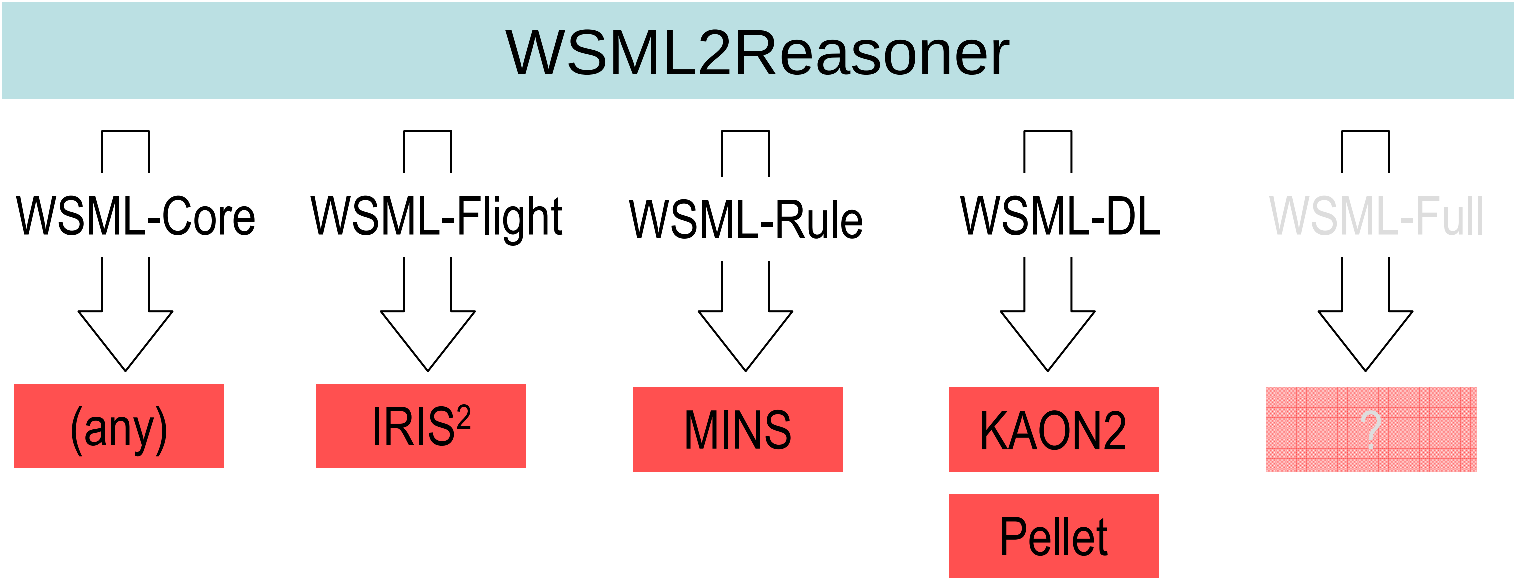
WSML-Full

- Unifies WSML-DL and WSML-Rule in a common first order framework
- Semantics of WSML-Full still an open research issue

Software Framework

WSML2Reasoner[1] is intended as:

- a framework for reasoning with all WSML language variants



[1] http://www.wsml2reasoner.org/

[2] http://www.iris-reasoner.org/

IRIS reasoner

IRIS is a Datalog reasoner

- Extended with:
  - Locally stratified default negation
  - XML Schema data types
  - Bottom-up evaluation
  - (In)equality built-in predicates
- => Fully supports WSML-Flight
- Open source (GNU lesser GPL)

MINS reasoner

MINS is a:

- Datalog reasoner with:
  - Well-founded semantics
  - Function symbols
  - Only 2 data types (int and string)
- => (Kind of) supports WSML-Rule

However, it is:

- no longer supported
- can not be used as a platform for the future

IRIS reasoner (future)

- For IRIS to support WSML-Rule:
  - Function symbols (already supported)
  - Optimised well-founded semantics implementation
  - Top-down evaluation
  - To be well-behaved in the undecidable cases
- => Will support WSML-Rule

KAON2 reasoner

KAON2:

- Is an OWL-DL reasoner that converts OWL-DL to disjunctive Datalog
- Is better suited for instance oriented reasoning
- Is not open-source
- Can be used to support WSML-DL in this framework

Pellet reasoner

Pellet is:

- An OWL-DL reasoner that uses tableau reasoning
- Better suited for schema oriented reasoning
- Open-source
- Used to support WSML-DL in this framework

Challenges

A single coherent software framework that:

- Is open source
- Supports WSML-DL and WSML-Rule
- Combines a suite of complementary techniques
- Is robust, fast and scalable
- Can be adapted for approximate reasoning