OWL & Rules

Introduction to the Semantic Web

Birte Glimm

November 7, 2010

OUTLINE

1 RULE-BASED FORMALISMS

2 RDF, RDFS & OWL 2

3 OWL 2 PROFILES

4 OWL 2 RL & RULES

5 IMPLEMENTATIONS & CONCLUSIONS

Rule-Based Formalisms

- Rules provide a natural of modelling "if-then" knowledge
- General form of a rule

  Body $\rightarrow$ Head   alternative writing: Head :- Body

- Body: (possibly empty) conjunction of atoms
- Head: at most one atom (Horn) or a disjunction of atoms

Example

$\forall x\ \forall y$ (hasSister(x, y) $\rightarrow$ hasSibling(x, y))
$\forall x$ (Male(x) $\wedge$ Female(x) $\rightarrow \bot$)

$\Rightarrow$ We use short names (hasSister) instead of fully qualified (http://example.org/myExample#hasSister) or abbreviated IRIs (ex:hasSister) throughout

$\Rightarrow$ We use x, y, and z as variables throughout

THE SEMANTICS OF RULES

- Syntactically, the rules we consider are just FOL formulae
- Can be interpreted under standard FOL semantics
- Other (non-monotonic) interpretations are possible
  - well-founded semantics
  - stable model semantics
  - answer set semantics
- For Horn rules, these interpretations coincide (unless negation of atoms is allowed)
- Here, we only consider the FOL (=open world) semantics
- Production rules consider the consequence to be an action "If-then do" $\Rightarrow$ not considered here

Rule Interchange Format – RIF

- RIF is a W3C standard for expressing rules
- RIF has several dialects and features
  - Basic Logic Dialect ⇒ RIF BLD (declarative)
    - RIF Core, function-free subset of RIF BLD
  - Production Rules ⇒ RIF PRD
  - Datatypes and Built-in functions ⇒ RIF DTB
  - Framework for Logic Dialects ⇒ RIF FLD (a general framework for logic-based rule languages, covering RIF BLD and RIF Core)
- Can be used in combination with RDF or OWL documents

RIF (Presentation) Syntax Example

```
Forall ?x ?y (
  hasSibling(?x, ?y) :- hasSister(?x, ?y)
)
```

What We Cannot Say with Rules

- With rules, one cannot require the existence of individuals with certain properties except by explicitly naming them
- We can express that there are two persons are married by giving them names
- We cannot express something like:“Every twin has some sibling”
- Requires function symbols (in RIF BLD, not RIF Core)

Example

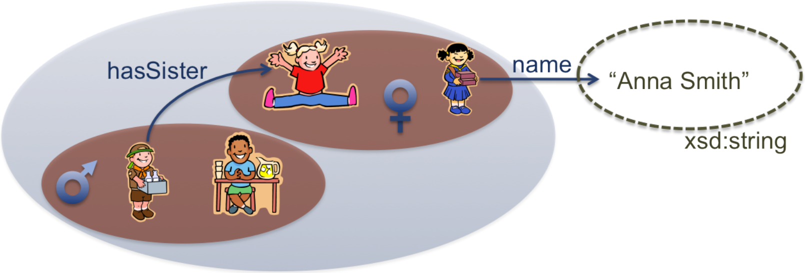
hasSibling(Mary, Peter)
Twin(x) $\rightarrow$ hasSibling(x, Somebody) $\lightning$
Twin(x) $\rightarrow$ hasSibling(x, f(x))

$\Rightarrow$ Program evaluation might not terminate ☹

$\Rightarrow$ One can/has to explicitly specify the desired inference steps. OWL has a large set of predefined modeling constructs.

What OWL Talks About (Semantics)

- No customizable rule set, but modeling constructs with pre-defined semantics
- OWL ontologies talk about worlds that contain
  - Individuals (constants) such as Mary, Peter
  - Classes ⇝ unary predicates: Male(_), Female(_)
  - Properties ⇝ binary predicates: hasSister(_, _)
    - Object properties linking a pair of individuals
    - Data properties linking an individual with a concrete value (string, integer, . . .)



RDF-Based versus Direct Semantics

- The OWL RDF-Based Semantics (aka OWL Full) is an extension of the RDFS Semantics
  - Individuals, Classes, and Properties are interpreted as elements of the domain
  - Classes have an extension that is a subset of the domain
  - Properties have an extension of pairs of elements from the domain
- The OLW Direct Semantics (aka OWL DL) is directly model-theoretic
  - Based on Description Logics
  - Classes are interpreted as subsets of the domain
  - Properties are interpreted as sets of pairs of elements from the domain
  - $\Rightarrow$ Syntactic restrictions on well-formed sets of RDF triples

Stating Assertional Knowledge

Asserts information about concrete, named individuals

Class Assertion Example

OWL Functional Style Syntax

ClassAssertion(Male Peter)

Turtle Syntax

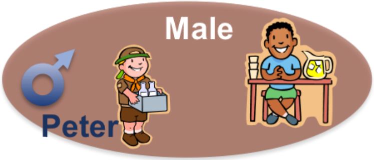
Peter rdf:type Male

RDF/XML Syntax

<Male rdf:about="Peter"/>

Rules Syntax

$\rightarrow$ Male(Peter)



Stating Assertional Knowledge

Asserts information about concrete, named individuals

Object Property Assertion Example

OWL Functional Style Syntax

ObjectPropertyAssertion(hasSister Peter Mary)

Turtle Syntax

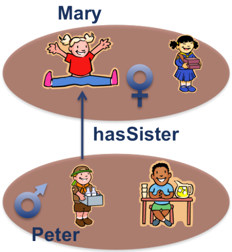
Peter hasSister Mary

RDF/XML Syntax

```
<rdf:Description rdf:about="Peter">
    <hasSister rdf:resource="Mary"/>
</rdf:Description>
```

Rules Syntax

$\rightarrow$ hasSister(Peter, Mary)



$\Rightarrow$ That is all that can be said in plain RDF

Stating Terminological Knowledge

Information about how classes and properties relate in general

Subclass Axiom Example

OWL Functional Style Syntax

SubClassOf(Male Person)

Turtle Syntax

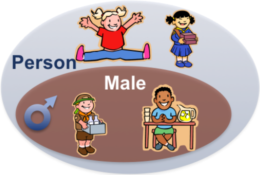
Male rdfs:subClassOf Person

RDF/XML Syntax

```
<owl:Class rdf:ID="Person"/>
<owl:Class rdf:ID="Male">
   <rdfs:subClassOf rdf:resource="Person"/>
</owl:Class>
```

Rules Syntax

Male(x) $\rightarrow$ Person(x)



Stating Terminological Knowledge

Information about how classes and properties relate in general

Subproperty Axiom Example

OWL Functional Style Syntax

SubObjectPropertyOf(hasSister hasSibling)

Turtle Syntax

hasSister rdfs:subPropertyOf hasSibling

RDF/XML Syntax

<owl:ObjectProperty rdf:ID="hasSibling"/>
<owl:ObjectProperty rdf:ID="hasSister">
<rdfs:subPropertyOf rdf:resource="hasSibling"/>
</owl:ObjectProperty>

Rules Syntax

hasSister(x, y) $\rightarrow$ hasSibling(x, y)

Stating Terminological Knowledge

- RDFS can further specify domain and range classes for properties.
- For example, the domain and range of hasSibling could be specified as Person.

$\Rightarrow$ Careful, this is not a constraint, but an implication.

Range Example

hasSister rdfs:range Female
Peter hasSister Mary
Mary rdf:type Male

$\Rightarrow$ Mary rdf:type Female

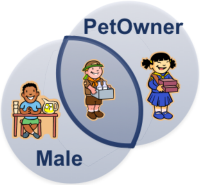
Contradiction only when Male and Female are defined as disjoint!

OWL Class Constructors

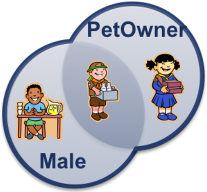
Complex classes can be build by means of constructors

Intersection and Union

ObjectIntersectionOf(Male PetOwner)



ObjectUnionOf(Male PetOwner)



SubClassOf(ObjectIntersectionOf(Male PetOwner) Friendly)

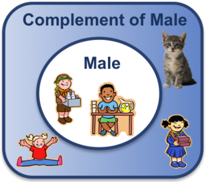
Male(x) $\wedge$ PetOwner(x) $\rightarrow$ Friendly(x)

OWL Class Constructors

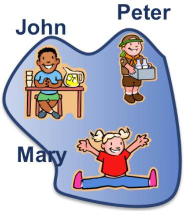
Complex classes can be build by means of constructors

Complement & Closed Classes

ObjectComplementOf(Male)



ObjectOneOf(Peter Mary John)



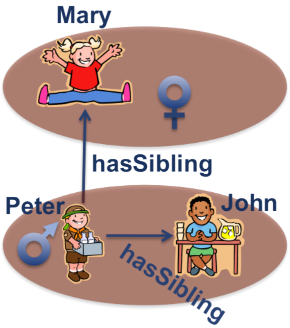
OWL Class Constructors

Complex classes can be build by means of constructors

Restrictions

ObjectSomeValuesFrom(
hasSibling Female)
⇒ Peter

ObjectAllValuesFrom(
hasSibling Female)
⇒ Mary, John

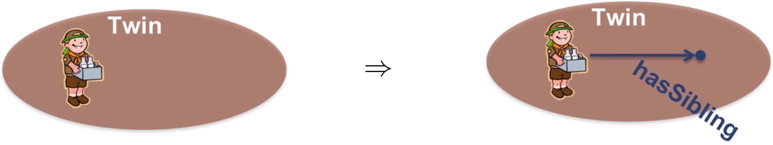


OWL Class Constructors in Axioms

Restrictions in Axioms

ClassAssertion(Twin Peter)
SubClassOf(Twin ObjectSomeValuesFrom(hasSibling Twin))

Twin(Peter)
Twin(x) $\rightarrow$ hasSibling(x, f(x))
Twin(x) $\rightarrow$ Twin(f(x))



OWL Class Constructors

There are more constructors available:

- At least, at most and exact cardinality restrictions
- Self restriction
- Restrictions on data ranges
- Constructors for data ranges

OWL Built-ins

Special Classes

- owl:Thing ⇒ contains all individuals of the domain
- owl:Nothing ⇒ the empty class containing no individuals

Special Object Properties

- owl:topObjectProperty ⇒ connects all possible pairs of individuals
- owl:bottomObjectProperty ⇒ does not connect any pair of individuals

Special Data Properties

- owl:topDataProperty ⇒ connects all possible individuals with all literals
- owl:bottomDataProperty ⇒ does not connect any individual with a literal

Property Axioms

Can define characteristics of properties

Inverses & Functionality

InverseObjectProperties(
hasSister isSisterOf)

hasSister(x, y) $\rightarrow$ isSisterOf(y, x)

FunctionalObjectProperty(hasMother)

hasMother(x, y_1) $\wedge$ hasMother(x, y_2)
$\rightarrow y_1 = y_2$

Property Axioms

Can define characteristics of properties

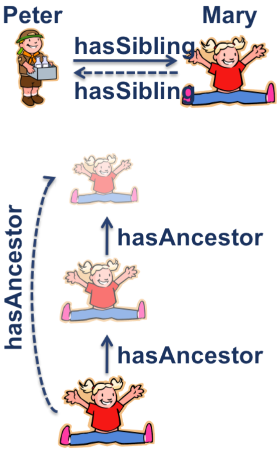
Symmetry & Transitivity

SymmetricObjectProperty(hasSibling)

hasSibling(x, y) $\rightarrow$ hasSibling(y, x)

TransitiveObjectProperty(hasAncestor)

hasAncestor(x, y) $\wedge$ hasAncestor(y, z) $\rightarrow$ hasAncestor(x, z)



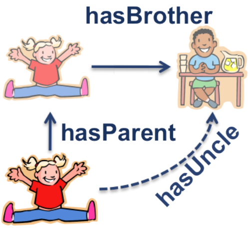
Property Chain Axioms

Allow for inferring the existence of a property from a chain of properties

Property Chains

SubObjectPropertyOf(
ObjectPropertyChain(hasParent hasBrother)
hasUncle)

$hasParent(x, y) \wedge hasBrother(y, z) \rightarrow hasUncle(x, z)$



Property Axioms

More property axioms available

- Reflexive, irreflexive, asymmetric, inverse functional, and disjoint object properties
- Functional and disjoint data properties

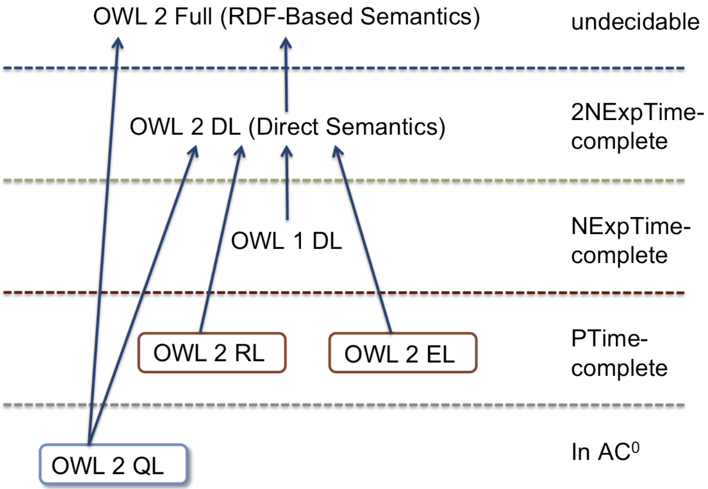
OWL SEMANTICS

- OWL RDF-Based Semantics (OWL Full)
  - All constructors can be used in an unrestricted way
  - Reasoning works with any RDF document
  - Depending on the input, reasoning might not terminate
- OWL Direct Semantics (OWL DL)
  - Based on Description Logics
  - Accepts only certain well-formed RDF documents as input
  - Makes restrictions one the usage of constructors (e.g., regularity restrictions on role chains)
  - Guarantees termination

OWL 2 Profiles

- OWL 2 DL is decidable, but computationally hard
  - ⇒ not scalable enough for many applications
- OWL Full is not even decidable
  - ⇒ not many implementations that support all of OWL Full are available
- Idea: identify subsets of OWL 2 which are
  - sufficiently expressive, but
  - of lower complexity (tractable)
- Profiles tailored to specific reasoning services
  - Terminological/schema reasoning:
    ⇒ OWL EL
  - Query Answering via database engines:
    ⇒ OWL QL
  - Assertional/data reasoning with rule engines:
    ⇒ OWL RL

OWL 2 Profiles



OWL 2 EL

- A (near maximal) fragment of OWL 2 such that
  - Satisfiability checking is in PTime (PTime-Complete)
  - Data complexity of query answering also PTime-Complete
- Class hierarchy (all subclass relations between classes) can be computed in "one pass"
- Exploits saturation-based techniques developed for $\mathcal{EL}$ description logics
  - ⇒ Can be extended to the Horn (non-disjunctive) fragment of OWL DL [Kazakov 2009]
- Allowed:
  - SubClassOf axioms with intersection, someValuesFrom, owl:Thing, owl:Nothing, closed classes with one member (nominal)
  - Property chain axioms, range restrictions (under certain conditions)
- Disallowed:
  - Negation (complement), disjunction (union), allValuesFrom, inverse properties

OWL 2 QL

- A (near maximal) fragment of OWL 2 such that
  - Data complexity of conjunctive query answering is in AC^0
- Can exploit query rewriting based reasoning technique
  - ⇒ Data storage and query evaluation can be delegated to standard RDBMS
- Benefits from research in DL-Lite description logics
  - ⇒ Novel technique to prevent exponential blowup from rewritings [Kontchakov et al. 2010, Rosati & Almatelli 2010]
  - ⇒ Can be extended to more expressive languages by using a Datalog engine [Perez-Urbina et al. 2009]
- Allowed:
  - Subproperties, Domain, Range
  - SubClassOf axioms with left hand side: class name or SomeValuesFrom(op owl:Thing), right hand side: intersection of class names, SomeValuesFrom(op c), and negations of lhs expressions

OWL 2 RL

- A (near maximal) fragment of OWL 2 such that
  - Reasoning is PTime-complete (ontology consistency, class expression satisfiability, class expression subsumption, instance checking, and conjunctive query answering)
  - Reasoning is sound and complete when the input RDF graph has certain properties, and sound on arbitrary RDF graphs
- Can work directly on RDF triples to enrich instance data (materialize schema inferences for facts)
- Reasoning can be implemented in a rule engine (with equality support)

♡ In OWL RL RIF and OWL meet since any RIF (Core) rule engine can be used to implement OWL RL

⇒ W3C Working Group Note: “OWL 2 RL in RIF” at http://www.w3.org/TR/rif-owl-rl/

OWL 2 RL Inferences via Rules

- OWL 2 RL specification provides complete rule set
- Each RDF triple is encoded via a ternary predicate T(_, _, _)

Example Rule for Subproperty Reasoning

prp-spo1 T(?p1, rdfs:subPropertyOf, ?p2) $\wedge$ T(?x, ?p1, ?y)
$\rightarrow$ T(?x, ?p2, ?y)

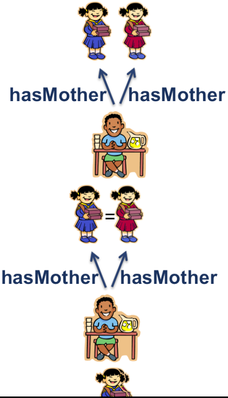
hasSister rdfs:subPropertyOf hasSibling
Peter hasSister Mary
$\Rightarrow$ Peter hasSibling Mary

OWL 2 RL Inferences via Rules

Example Rule for Functionality Reasoning

prp-fp T(?p, rdf:type, owl:FunctionalProperty) $\wedge$
T(?x, ?p, ?y_1) $\wedge$ T(?x, ?p, ?y_2)
$\rightarrow$ T(?y1, owl:sameAs, ?y2)

hasMother rdf:type owl:FunctionalProperty
John hasMother Anna
John hasMother Ann
$\Rightarrow$ Anna owl:SameAs Ann



OWL 2 RL Inferences via Rules

Person rdfs:subClassOf _:c
_:c rdf:type owl:Restriction
_:c owl:allValuesFrom Person
_:c owl:onProperty hasChild
Anna hasChild Mary
Anna rdf:type Person

SubClassOf(Person
    ObjectAllValuesFrom(hasChild
        Person))
ObjectPropertyAssertion(hasChild
    Anna Mary)
ClassAssertion(Person Anna)

Class Expression & Axiom Reasoning

cax-sco   T(?c1, rdfs:subClassOf, ?c2) $\wedge$
    T(?x, rdf:type, ?c1)
        $\rightarrow$ T(?x, rdf:type, ?c2)

cls-avf   T(?x, owl:allValuesFrom, ?y) $\wedge$
    T(?x, owl:onProperty, ?p) $\wedge$
    T(?u, rdf:type, ?x) $\wedge$
    T(?u, ?p, ?v)
        $\rightarrow$ T(?v, rdf:type, ?y)

OWL 2 RL Inferences via Rules

Person rdfs:subClassOf _:c ⇒ Anna rdf:type _:c
_:c rdf:type owl:Restriction ⇒ Mary rdf:type Person
_:c owl:allValuesFrom Person
_:c owl:onProperty hasChild
Anna hasChild Mary
Anna rdf:type Person

Class Expression & Axiom Reasoning

cax-sco T(Person, rdfs:subClassOf, _:c) ∧
T(Anna, rdf:type, Person)
→ T(Anna, rdf:type, _:c)

cls-avf T(_:c, owl:allValuesFrom, Person) ∧
T(_:c, owl:onProperty, hasChild) ∧
T(Anna, rdf:type, _:c) ∧
T(Anna, hasChild, Mary)
→ T(Mary, rdf:type, Person)

OWL 2 RL IN RIF

- More optimized implementation than via the fixed OWL 2 RL rule set possible
- The OWL 2 RL rules can be implemented in the RIF Core dialect
  - ⇒ Either as fixed or ontology-specific rule set
- W3C Working Group Note: "OWL 2 RL in RIF" outlines different algorithms for OWL RL reasoning in RIF
  - ⇒ http://www.w3.org/TR/rif-owl-rl/

RIF Implementations

- RIF BLD
  - Eye, IBM DB2 XML, IRIS, OntoBroker (partial), riftr, Silk, VampirePrime
- RIF Core
  - all above plus fuxi, IBM Websphere ILOG JRules, RIFle
- RIF PRD
  - IBM Websphere ILOG JRules, OBR , RIFle
- RIF DTB
  - Eye, IRIS, OBR (partial), RIFle, riftr

See http://www.w3.org/2005/rules/wiki/Implementations

OWL Implementations

- OWL 2 DL
  - FaCT++, HermiT, Pellet, RacerPro (partial)
- OWL 2 RL
  - ELLY, Jena, Oracle, OWLIM, OWLRL
  - Essentially any rule engine
  - E.g., via RIF Rules in the RIF Core dialect
- OWL 2 QL
  - Owlgres, Quill, QuOnto, REQUIEM
  - Essentially any SQL engine (with query rewriting on top)
- OWL 2 EL
  - CB, CEL, ELLY, JCEL, Pellet, SHER, snorocket

See http://www.w3.org/2007/OWL/wiki/Implementations

Conclusions

- OWL 2 defines several modeling constructs for which OWL reasoners provide automated inference services
  - OWL Direct Semantics: set-theoretic semantics, based on description logics
  - OWL RDF-Based Semantics: extension of RDFS, works directly on triples
  - OWL 2 Profiles for efficient and scalable reasoning
- Rules allow for customizable inferences
  - RIF W3C standard for applying rules to semantic web data
  - RIF dialects (Core, BLD, PRD) for different purposes
  - Further RIF FLD dialects: RIF Core Answer Set Programming Dialect, RIF Core Logic Programming Dialect, RIF Uncertainty Rule Dialect

⇒ OWL RL can be implemented via RIF Core rules

⇒ Also OWL EL can be implemented in a rule engine

⇒ SPARQL Entailment Regimes lift SPARQL to RDF(S), OWL, and RIF reasoning: http://www.w3.org/TR/sparql11-entailment/

References

- OWL 2: http://www.w3.org/TR/owl2-overview/
- RIF: http://www.w3.org/TR/rif-overview/
- SPARQL Entailment Regimes: http://www.w3.org/TR/sparql11-entailment/
- Book "Foundations of Semantic Web Technologies". P. Hitzler, M. Krötzsch, S. Rudolph. CRC Press, 2009

[Kazakov 2009] Y. Kazakov. Consequence-Driven Reasoning for Horn SHIQ Ontologies. IJCAI, 2009
[Kontchakov et al. 2010] R. Kontchakov, C. Lutz, D. Toman, F. Wolter and M. Zakharyaschev. The Combined Approach to Query Answering in DL-Lite. KR, 2010
[Rosati & Almatelli 2010] R. Rosati, A. Almatelli. Improving Query Answering over DL-Lite Ontologies. KR, 2010
[Perez-Urbina et al. 2009] H. Pérez-Urbina, I. Horrocks, B. Motik. Efficient Query Answering for OWL 2. ISWC, 2009