OWL 2 Rules (Part 1)

Tutorial at ESWC2009
May 31, 2009

Pascal Hitzler
Markus Krötzsch
Sebastian Rudolph

AIFB, Universität Karlsruhe (TH)
http://www.pascal-hitzler.de
http://korrekt.org
http://www.sebastian-rudolph.de
Outline Part 1

- The Early Days of KR: Rule-Based Formalisms
- OWL 2 DL – the new DL-based Web Ontology Language
- Semantics of OWL DL
- Tractable Fragments
The Early Days of KR: Rule-Based Formalisms

- rules provide a natural way of modelling “if-then“ knowledge
- general form of a (Horn) rule:

\[ \text{Body} \rightarrow \text{Head} \]

- body: (possibly empty) conjunction of atoms, head: at most one atom
- Examples:

\[
\begin{align*}
\text{married}(x,y) \land \text{Woman}(x) & \rightarrow \text{Man}(y) \\
\text{Man}(x) \land \text{Woman}(x) & \rightarrow \\
& \rightarrow \text{married}((\text{pascal}, \text{anne}))
\end{align*}
\]
The Early Days of KR: Rule-Based Formalisms

- rules provide a natural way of modelling “if-then” knowledge
- general form of a (Horn) rule:

\[ \text{Body} \rightarrow \text{Head} \]

- body: (possibly empty) conjunction of atoms, head: at most one atom
- Examples:

\[ \forall x \forall y (\text{married}(x,y) \land \text{Woman}(x) \rightarrow \text{Man}(y)) \]

\[ \forall x (\text{Man}(x) \land \text{Woman}(x) \rightarrow \text{false}) \]

true → married(pascal,anne)
On the Semantics of Rules

- syntactically, rules are just FOL formulae
- hence they can be interpreted under FOL standard semantics
- other (non-monotonic) interpretations are possible:
  - well-founded semantics
  - stable model semantics
  - answer set semantics
- in the case of Horn rules, they all coincide (differences if negation of atoms is allowed)
- in this tutorial, we strictly adhere to FOL (=open-world) semantics
What We Cannot Say with Rules

- with rules, one cannot require the existence of individuals with certain properties except by explicitly naming them.

- i.e., we can express that there are two persons that are married by giving them names (say, person1 and person2):

\[
\text{true } \rightarrow \text{married(person1,person2)}
\]

- but we cannot express something like:
  “every husband is married to somebody“

\[
\text{wrong: } \text{husband(x) } \rightarrow \text{married(x,person)}
\]

That's where OWL comes in!
What OWL Talks About (Semantics)

- both OWL 1 DL and OWL 2 DL are based on description logics
- here, we will treat OWL from the “description logic viewpoint“:
  - we use DL syntax
  - we won’t talk about datatypes and non-semantic features of OWL
- OWL (DL) ontologies talk about worlds that contain
  
  **individuals**
  
  constants: pascal, anne

  **classes / concepts**
  
  unary predicates: male(_), female(_)

  **properties / roles**
  
  binary predicates: married(_,_)
Assertional Knowledge

- asserts information about concrete named individuals
  - class membership: Male(pascal)
    - rule version: → Male(pascal)
  - property membership: married(anne,pascal)
    - rule version: → married (anne,pascal)

That's all what can be said with RDF!
Terminological Knowledge – Subclasses and Subproperties

- information about how classes and properties relate in general

- subclass: Child $\subseteq$ Person

  `<owl:Class rdf:about="Child">`  
  `<rdfs:subClassOf rdf:resource="Person"/>`  
  `</owl:Class>`

  rule version: Child(x) $\rightarrow$ Person(x)

- subproperty: hasHusband $\subseteq$ married

  `<owl:ObjectProperty rdf:about="hasHusband">`  
  `<rdfs:subPropertyOf rdf:resource="married"/>`  
  `</owl:ObjectProperty>`

  rule version: hasHusband(x,y) $\rightarrow$ married (x,y)
Class Constructors

- build new classes from class, property and individual names
  - union: Actor ▐ Politician
    
    ```xml
    <owl:unionOf rdf:parseType="Collection">  
    <owl:Class rdf:about="Actor"/>  
    <owl:Class rdf:about="Politician"/>  
    </owl:unionOf>
    ```
  - intersection: Actor ▴ Politician
    
    ```xml
    <owl:intersectionOf rdf:parseType="Collection">  
    <owl:Class rdf:about="Actor"/>  
    <owl:Class rdf:about="Politician"/>  
    </owl:intersectionOf>
    ```
Class Constructors

- build new classes from class, property and individual names

- complement: $\neg$Politician

  - $\langle$owl:complementOf
    rdf:resource="Politician$\rangle$

- closed classes: $\{\text{anne, merula, pascal}\}$

  - $\langle$owl:oneOf rdf:parseType="Collection"
    $\langle$rdf:Description rdf:about="anne"/>  
    $\langle$rdf:Description rdf:about="merula"/>  
    $\langle$rdf:Description rdf:about="pascal"/>  
  $\rangle$
Class Constructors

- build new classes from class, property and individual names
- existential quantification: $\exists\text{hasChild.Female}$

```xml
<owl:Restriction>
  <owl:onProperty rdf:resource="hasChild"/>
  <owl:someValuesFrom rdf:resource="Female"/>
</owl:Restriction>
```
Class Constructors

- build new classes from class, property and individual names
- universal quantification: $\forall \text{hasChild.Female}$

```xml
<owl:Restriction>
  <owl:onProperty rdf:resource="hasChild"/>
  <owl:allValuesFrom rdf:resource="Female"/>
</owl:Restriction>
```
Class Constructors

- build new classes from class, property and individual names
- cardinality restriction: ≥2hasChild.Female

```xml
<owl:Restriction>
  <owl:minQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:minQualifiedCardinality>
  <owl:onProperty rdf:about="hasChild"/>
  <owl:onClass rdf:about="Female"/>
</owl:Restriction>
```
Class Constructors

- build new classes from class, property and individual names
  - Self-restriction: $\exists$ killed.Self

```xml
<owl:Restriction>
  <owl:onProperty rdf:resource="killed"/>
  <owl:hasSelf rdf:datatype="&xsd:boolean"/>
  true
</owl:hasSelf>
</owl:Restriction>
```
Special Classes and Properties

- special classes
  - top class: $\top$
    ...class containing all individuals of the domain
    owl:Thing
  - bottom class: $\bot$
    ...“empty“ class containing no individuals
    owl:Nothing
- universal property: $U$
  ...property linking every individual to every individual
  owl;topObjectProperty
Property Chain Axioms

- allow to infer the existence of a property from a chain of properties:

- \( \text{hasParent} \circ \text{hasParent} \sqsubset \text{hasGrandparent} \)

  rule version: \( \text{hasParent}(x,y) \land \text{hasParent}(y,z) \rightarrow \text{hasGrandparent}(x,z) \)

```
<rdf:Description rdf:about="hasGrandparent">
  <owl:propertyChainAxiom rdf:parseType="Collection">
    <owl:ObjectProperty rdf:about="hasParent"/>
    <owl:ObjectProperty rdf:about="hasParent"/>
  </owl:propertyChainAxiom>
</rdf:Description>
```
Property Chain Axioms

- allow to infer the existence of a property from a chain of properties:

  - hasEnemy ∘ hasFriend ⊆ hasEnemy
  - rule version: hasEnemy(x,y) ∧ hasFriend(y,z) → hasEnemy(x,z)

```
<rdf:Description rdf:about="hasEnemy">
  <owl:propertyChainAxiom rdf:parseType="Collection">
    <owl:ObjectProperty rdf:about="hasEnemy"/>
    <owl:ObjectProperty rdf:about="hasFriend"/>
  </owl:propertyChainAxiom>
</rdf:Description>
```
arbitrary property chain axioms lead to undecidability

restriction: set of property chain axioms has to be regular

there must be a strict linear order $\prec$ on the properties

every property chain axiom has to have one of the following forms:

$$R \circ R \sqsubseteq R \quad S \sqsubseteq R \quad S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$$

thereby, $S_i \prec R$ for all $i=1, 2, \ldots, n$.

Example 1:  

$$R \circ S \sqsubseteq R \quad S \circ S \sqsubseteq S \quad R \circ S \circ R \sqsubseteq T$$

$\rightarrow$ regular with order $S \prec R \prec T$

Example 2:  

$$R \circ T \circ S \sqsubseteq T$$

$\rightarrow$ not regular because form not admissible

Example 3:  

$$R \circ S \sqsubseteq S \quad S \circ R \sqsubseteq R$$

$\rightarrow$ not regular because no adequate order exists
combining property chain axioms and cardinality constraints may lead to undecidability

restriction: use only *simple* properties in cardinality expressions (i.e. those which cannot be – directly or indirectly – inferred from property chains)

technically:
- for any property chain axiom \( S_1 \circ S_2 \circ ... \circ S_n \sqsubseteq R \) with \( n>1 \), \( R \) is non-simple
- for any subproperty axiom \( S \sqsubseteq R \) with \( S \) non-simple, \( R \) is non-simple
- all other properties are simple

Example: \( Q \circ P \sqsubseteq R \quad R \circ P \sqsubseteq R \quad R \sqsubseteq S \quad P \sqsubseteq R \quad Q \sqsubseteq S \)

non-simple: \( R, S \)  
simple: \( P, Q \)
Property Characteristics

- a property can be
  - the inverse of another property: hasParent ≡ parentOf
    - rule version:
      - hasParent(x,y) → parentOf(y,x)
      - parentOf(x,y) → hasParent(y,x)
  
- disjoint with another property: Disj(hasParent, parentOf)
  - rule version:
    - hasParent(x,y), parentOf(x,y) →

- other property characteristics that can be expressed:
  (inverse) functionality, transitivity, symmetry, asymmetry, reflexivity, irreflexivity
OWL 2 DL – Semantics

- model-theoretic semantics
- starts with interpretations
- an interpretation maps
  individual names, class names and property names...

...into a domain
OWL 2 DL – Semantics

- mapping is extended to complex class expressions:
  - $\top^I = \Delta^I$
  - $\bot^I = \emptyset$
  - $(C \cap D)^I = C^I \cap D^I$
  - $(C \cup D)^I = C^I \cup D^I$
  - $(\neg C)^I = \Delta^I \setminus C^I$
  - $\forall R.C = \{ x \mid \forall (x,y) \in R^I \rightarrow y \in C^I \}$
  - $\exists R.C = \{ x \mid \exists (x,y) \in R^I \land y \in C^I \}$
  - $\geq n R.C = \{ x \mid \#\{ y \mid (x,y) \in R^I \land y \in C^I \} \geq n \}$
  - $\leq n R.C = \{ x \mid \#\{ y \mid (x,y) \in R^I \land y \in C^I \} \leq n \}$

- ...and to role expressions:
  - $U^I = \Delta^I \times \Delta^I$
  - $(R^-)^I = \{ (y,x) \mid (x,y) \in R^I \}$

- ...and to axioms:
  - $C(a)$ holds, if $a^I \in C^I$
  - $R(a,b)$ holds, if $(a^I,b^I) \in R^I$
  - $C \subseteq D$ holds, if $C^I \subseteq D^I$
  - $R \subseteq S$ holds, if $R^I \subseteq S^I$
  - Disj($R,S$) holds if $R^I \cap S^I = \emptyset$
  - $S_1 \circ S_2 \circ \ldots \circ S_n \subseteq R$ holds if $S_1^I \circ S_2^I \circ \ldots \circ S_n^I \subseteq R^I$
but often OWL 2 DL is said to be a fragment of FOL...

- yes, there is a translation of OWL 2 DL into FOL

\[ \pi(C \sqsubseteq D) = (\forall x)(\pi_x(C) \rightarrow \pi_x(D)) \]
\[ \pi_x(A) = A(x) \]
\[ \pi_x(\neg C) = \neg \pi_x(C) \]
\[ \pi_x(C \sqcap D) = \pi_x(C) \land \pi_x(D) \]
\[ \pi_x(C \sqcup D) = \pi_x(C) \lor \pi_x(D) \]
\[ \pi_x(\forall R.C) = (\forall x_1)(R(x, x_1) \rightarrow \pi_{x_1}(C)) \]
\[ \pi_x(\exists R.C) = (\exists x_1)(R(x, x_1) \land \pi_{x_1}(C)) \]
\[ \pi_x(\geq n S.C) = (\exists x_1) \ldots (\exists x_n) \left( \bigwedge_{i \neq j} (x_i \neq x_j) \land \bigwedge_i (S(x, x_i) \land \pi_{x_i}(C)) \right) \]
\[ \pi_x(\leq n S.C) = (\exists x_1) \ldots (\exists x_{n+1}) \left( \bigwedge_{i \neq j} (x_i \neq x_j) \land \bigwedge_i (S(x, x_i) \land \pi_{x_i}(C)) \right) \]
\[ \pi_x(\{a\}) = (x = a) \]
\[ \pi_x(\exists S.\text{Self}) = S(x, x) \]

\[ \pi_x(\exists S.\text{Ref}) = (\exists x y)(\pi_{x,y}(R_1) \rightarrow \pi_{x,y}(R_2)) \]
\[ \pi_x(\exists S.\text{Asy}(R)) = (\exists x y)(\pi_{x,y}(R) \rightarrow \neg \pi_{y,x}(R)) \]
\[ \pi_x(\exists S.\text{Dis}(R_1, R_2)) = \neg (\exists x)(\exists y)(\pi_{x,y}(R_1) \land \pi_{x,y}(R_2)) \]

...which (interpreted under FOL semantics) coincides with the definition just given.
OWL 2 Profiles

- OWL 2 DL is very expressive (although decidable)
  - tool support for full OWL 2 DL difficult to achieve
- complexity for standard reasoning tasks: N2ExpTime
  - scalability cannot be guaranteed
- idea: identify subsets of OWL 2 DL which are
  - still sufficiently expressive
  - of lower complexity (preferably in PTime)
  - computationally easier to handle

- OWL 2 Profiles:
  - OWL EL
  - OWL RL
  - OWL QL
OWL 2 EL

- allowed:
  - subclass axioms with intersection, existential quantification, top, bottom
    - closed classes must have only one member
  - property chain axioms, range restrictions (under certain conditions)
- disallowed:
  - negation, disjunction, arbitrary universal quantification, role inverses

Reasoning is PTime complete

Examples:

- $\exists \text{has.Sorrow} \subseteq \exists \text{has.Liqueur}$
- $\top \subseteq \exists \text{hasParent.Person}$
- $\exists \text{married.} \top \cap \text{CatholicPriest} \subseteq \perp$
- $\text{German} \subseteq \exists \text{knows.} \{\text{angela}\}$
- hasParent $\circ$ hasParent $\subseteq$ hasGrandparent
OWL 2 RL

— motivated by the question: what fraction of OWL 2 DL can be expressed by rules (with equality)?

— examples:

- $\exists\text{parentOf.}\exists\text{parentOf.} \top \sqsubseteq \text{Grandfather}$
  - rule version: $\text{parentOf}(x, y) \land \text{parentOf}(y, z) \rightarrow \text{Grandfather}(x)$

- $\text{Orphan} \sqsubseteq \forall\text{hasParent.}\text{Dead}$
  - rule version: $\text{Orphan}(x) \land \text{hasParent}(x, y) \rightarrow \text{Dead}(y)$

- $\text{Monogamous} \sqsubseteq \leq 1\text{married.}\text{Alive}$
  - rule version:
    $\text{Monogamous}(x) \land \text{married}(x, y) \land \text{Alive}(y) \land \text{married}(x, z) \land \text{Alive}(z) \rightarrow y = z$

- $\text{childOf} \circ \text{childOf} \sqsubseteq \text{grandchildOf}$
  - rule version: $\text{childOf}(x, y) \land \text{childOf}(y, z) \rightarrow \text{grandchildOf}(x, z)$

- $\text{Disj}(\text{childOf}, \text{parentOf})$
  - rule version: $\text{childOf}(x, y) \land \text{parentOf}(x, y)$
OWL 2 RL

- syntactic characterization:
  - essentially, all axiom types are allowed
  - disallow certain constructors on lhs and rhs of subclass statements

- cardinality restrictions: only on rhs and only \( \leq 1 \) and \( \leq 0 \) allowed
- closed classes: only with one member

- Reasoning is PTime complete
- Example Ontology: SWRC
**OWL 2 QL**

- motivated by the question: what fraction of OWL 2 DL can be captured by standard database technology?
- formally: query answering LOGSPACE w.r.t. data (via translation into SQL)
- allowed:
  - subproperties, domain, range
  - subclass statements with
    - left hand side: class name or expression of type $\exists r.T$
    - right hand side: intersection of class names, expressions of type $\exists r.C$ and negations of lhs expressions
    - no closed classes!
- Example:
  $\exists \text{married.}$ $T \sqsubseteq \neg \text{Free} \sqcap \exists \text{has.Sorrow}$
OWL 2 Reasoner

- OWL 2 DL:
  - Pellet http://clarkparsia.com/pellet/
  - HermiT http://www.hermit-reasoner.com/
- OWL 2 EL:
  - CEL http://code.google.com/p/cel/
- OWL 2 RL:
  - essentially any rule engine
- OWL 2 QL:
  - essentially any SQL engine (with a bit of query rewriting on top)
References

- OWL 2 W3C Documentation
  - http://www.w3.org/TR/owl2-overview/


  (Grab a flyer from us.)
Thanks!

http://semantic-web-grundlagen.de/wiki/ESWC09_Tutorial