Semantic Web Modelling Languages (Part 2)

Tutorial at IJCAI-09
July 13, 2009

... actually moving to Wright State University, Dayton, OH, Sept. 2009

Pascal Hitzler

Markus Krötzsch

Sebastian Rudolph

AIFB, Universität Karlsruhe (TH)
Germany

http://www.pascal-hitzler.de
http://korrekt.org
http://www.sebastian-rudolph.de
Full set of slides available from

http://semantic-web-grundlagen.de/wiki/IJCAI-09_Tutorial
OWL – Overview

- Web Ontology Language
  - W3C Recommendation for the Semantic Web, 2004
  - OWL 2 (revised W3C Recommendation) forthcoming in 2009
    - we already present this here

- Semantic Web KR language based on description logics (DLs)
  - OWL DL is essentially DL SROIQ(D)
  - KR for web resources, using URIs.
  - Using web-enabled syntaxes, e.g. based on XML or RDF.
    - We present
      - DL syntax (used in research – not part of the W3C recommendation)
      - (some) RDF Turtle syntax
References


- Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter Patel-Schneider, Sebastian Rudolph, OWL 2 Web Ontology Language: Primer. http://www.w3.org/TR/owl2-primer/

References – Textbooks


Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- Expressivity Examples: Rules in OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools
- Outlook
- References
Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- Expressivity Examples: Rules in OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools
- Outlook
- References
Rationale behind OWL

- Open World Assumption
- Favourable trade-off between expressivity and scalability
- Integrates with RDFS
- Purely declarative semantics

Features:
- Fragment of first-order predicate logic (FOL)
- Decidable
- Known complexity classes (N2ExpTime for OWL 2 DL)
- Reasonably efficient for real KBs
OWL Building Blocks

- individuals (written as URIs)
  - also: constants (FOL), ressources (RDF)
  - http://example.org/sebastianRudolph
  - we write these lowercase and abbreviated, e.g. "sebastianRudolph"

- classes (also written as URIs!)
  - also: concepts, unary predicates (FOL)
  - we write these uppercase, e.g. "Father"

- properties (also written as URIs!)
  - also: roles (DL), binary predicates (FOL)
  - we write these lowercase, e.g. "hasDaughter"
DL syntax

- Person(mary)
- Woman ⊆ Person
  - Person ≡ HumanBeing
- hasWife(john,mary)
- hasWife ⊆ hasSpouse
  - hasSpouse ≡ marriedWith

FOL syntax

- Person(mary)
- ∀x (Woman(x) → Person(x))
- hasWife(john,mary)
- ∀x ∀y (hasWife(x,y) → hasSpouse(x,y))

ABox statements

TBox statements
DL syntax                    RDFS syntax

- Person(mary)                    :mary rdf:type :Person .

- Woman ⊆ Person
  - Person ≡ HumanBeing

- hasWife(john, mary)              :john :hasWife :mary .

- hasWife ⊆ hasSpouse
  - hasSpouse ≡ marriedWith

- :Woman rdfs:subClassOf :Person .
Special classes and properties

- **owl:Thing** (RDF syntax)
  - DL-syntax: $\top$
  - contains everything

- **owl:Nothing** (RDF syntax)
  - DL-syntax: $\bot$
  - empty class

- **owl:topProperty** (RDF syntax)
  - DL-syntax: $U$
  - every pair is in $U$

- **owl:bottomProperty** (RDF syntax)
  - empty property
### Class constructors

**conjunction**
- Mother $\equiv$ Woman $\cap$ Parent
- $\text{:Mother owl:equivalentClass } _\cdot _x$ .
  $_:x$ rdf:type owl:Class .
  $_:x$ owl:intersectionOf ( :Woman :Parent ) .

**disjunction**
- Parent $\equiv$ Mother $\sqcup$ Father
- $\text{:Parent owl:equivalentClass } _\cdot _x$ .
  $_:x$ rdf:type owl:Class .
  $_:x$ owl:unionOf ( :Mother :Father ) .

**negation**
- ChildlessPerson $\equiv$ Person $\cap$ $\neg$Parent
- $\text{:ChildlessPerson owl:equivalentClass } _\cdot _x$ .
  $_:x$ rdf:type owl:Class .
  $_:x$ owl:intersectionOf ( :Person $_:y$ ) .
  $_:y$ owl:complementOf :Parent .

$\forall x \ ( \text{Mother}(x) \leftrightarrow \text{Woman}(x) \land \text{Parent}(x))$

$\forall x \ ( \text{Parent}(x) \leftrightarrow \text{Mother}(x) \land \text{Father}(x))$

$\forall x \ ( \text{ChildlessPerson}(x) \leftrightarrow \text{Person}(x) \land \neg\text{Parent}(x))$
Class constructors

- **existential quantification**
  - only to be used with a role – also called a *property restriction*
  - Parent $\equiv \exists_{\text{hasChild}}.\text{Person}$
  - $:^{\text{Parent}}\equiv\exists_{\text{hasChild}}.^{\text{Person}}$ 
    - $^{_x}\equiv\exists_{\text{hasChild}}.\exists_{\text{Person}}$
    - $^{_x}\equiv\exists_{\text{hasChild}}.\exists_{\text{Person}}$
  - $^{_x}\equiv\exists_{\text{hasChild}}.\exists_{\text{Person}}$
  - $^{_x}\equiv\exists_{\text{hasChild}}.\exists_{\text{Person}}$

- **universal quantification**
  - only to be used with a role – also called a *property restriction*
  - Person $\cap$ Happy $\equiv \forall_{\text{hasChild}}.\text{Happy}$
  - $^{_x}\equiv\forall_{\text{hasChild}}.\text{Happy}$
    - $^{_x}\equiv\forall_{\text{hasChild}}.\text{Happy}$
    - $^{_x}\equiv\forall_{\text{hasChild}}.\text{Happy}$
    - $^{_x}\equiv\forall_{\text{hasChild}}.\text{Happy}$

Class constructors can be nested arbitrarily

\[ \forall x (\text{Parent}(x) \iff \exists y (\text{hasChild}(x,y) \land \text{Person}(y))) \]

\[ \forall x (\text{Person}(x) \land \text{Happy}(x) \iff \forall y (\text{hasChild}(x,y) \to \text{Happy}(y))) \]
Understanding SROIQ(D)

The description logic ALC

- **ABox expressions:**
  - Individual assignments: Father(john)
  - Property assignments: hasWife(john, mary)

- **TBox expressions:**
  - Subclass relationships
  - Conjunction
  - Disjunction
  - Negation
  - Property restrictions

Complexity: ExpTime

Also: $\top$, $\bot$
Understanding SROIQ(D)

ALC + role chains = SR

- hasParent o hasBrother ⊆ hasUncle
  - :hasUncle owl:propertyChainAxiom (:hasParent :hasBrother ) .
  \[ \forall x \forall y (\exists z ((\text{hasParent}(x,z) \land \text{hasBrother}(z,y)) \rightarrow \text{hasUncle}(x,y))) \]

- includes top property and bottom property

- includes S = ALC + transitivity
  - hasAncestor o hasAncestor ⊆ hasAncestor

- includes SH = S + role hierarchies
  - hasFather ⊆ hasParent

I'll skip RDF syntax in the following.
Understanding SROIQ(D)

- O – nominals (closed classes)
  - MyBirthdayGuests ≡ \{bill, john, mary\}
  - Note the difference to
    MyBirthdayGuests(bill)
    MyBirthdayGuests(john)
    MyBirthdayGuests(mary)

- Individual equality and inequality (no unique name assumption!)
  - bill = john
    - \{bill\} ≡ \{john\}
  - bill ≠ john
    - \{bill\} ⊖ \{john\} ≡ ⊥
Understanding SROIQ(D)

- **I** – inverse roles
  - hasParent $\equiv$ hasChild$^-$
  - Orphan $\equiv$ $\forall$ hasChild$^-$ . Dead

- **Q** – qualified cardinality restrictions
  - $\leq$4 hasChild . Parent(john)
  - HappyFather $\equiv$ $\geq$2 hasChild . Female
  - Car $\subseteq$ $=$4 hasTyre . $\top$

- Complexity SHIQ, SHOQ, SHIO: ExpTime.
  Complexity SHOIQ: NExpTime
  Complexity SROIQ: N2ExpTime
Understanding SROIQ(D)

Properties can be declared to be

- Transitive hasAncestor
- Symmetric hasSpouse
- Asymmetric hasChild
- Reflexive hasRelative
- Irreflexive parentOf
- Functional hasHusband
- InverseFunctional hasHusband

called property characteristics
Understanding SROIQ(D)

(D) – datatypes

- so far, we have only seen properties with individuals in second argument, called *object properties or abstract roles* (DL)

- properties with datatype literals in second argument are called *data properties or concrete roles* (DL)

- allowed are many XML Schema datatypes, including xsd:integer, xsd:string, xsd:float, xsd:boolean, xsd:anyURI, xsd:dateTime

  and also e.g. owl:real
Understanding SROIQ(D)

(D) – datatypes

- hasAge(john, "51"^^xsd:integer)

- additional use of constraining facets (from XML Schema)
  - e.g. Teenager ⊑ Person ⊓ ∃ hasAge.(xsd:integer: ≥ 12 and ≤ 19)
    note: this is not standard DL notation!
  - :Teenager rdfs:subClassOf _:x .
    _:x rdf:type owl:Restriction .
    _:x owl:onProperty :hasAge .
    _:x owl:someValuesFrom _:y .
    _:y rdf:type rdfs:Datatype .
    _:y owl:onDatatype xsd:integer .
Understanding SROIQ(D)

further expressive features

- **Self**
  - PersonCommittingSuicide ≡ \(\exists\)kills.Self

- **Keys (not really in SROIQ(D), but in OWL)**
  - set of (object or data) properties whose values uniquely identify an object

- **disjoint properties**
  - Disjoint(hasParent,hasChild)

- **explicit anonymous individuals**
  - as in RDF: can be used instead of named individuals
SROIQ(D) constructors – overview

- ABox assignments of individuals to classes or properties
- ALC: ⊆, ≡ for classes
  ∩, ∪, ¬, ∃, ∀
  ⊤, ⊥
- SR: + property chains, property characteristics, role hierarchies ⊆
- SRO: + nominals {o}
- SROI: + inverse properties
- SROIQ: + qualified cardinality constraints
- SROIQ(D): + datatypes (including facets)

- + top and bottom roles (for objects and datatypes)
- + disjoint properties
- + Self
- + Keys (not in SROIQ(D), but in OWL)
Some Syntactic Sugar in OWL

This applies to the non-DL syntaxes (e.g. RDF syntax).

- disjoint classes
  - Apple ∩ Pear ⊆ ⊥

- disjoint union
  - Parent ≡ Mother ∪ Father
    Mother ∩ Father ⊆ ⊥

- negative property assignments (also for datatypes)
  - ¬hasAge(jack,"53"^^xsd:integer)
Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- Expressivity Examples: Rules in OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools
- Outlook
- References
OWL – Extralogical Features

- OWL ontologies have URIs and can be referenced by others via:
  - import statements
- Namespace declarations
- Entity declarations (must be done)
- Versioning information etc.

- Annotations
  - Entities and axioms (statements) can be endowed with annotations, e.g. using rdfs:comment.
  - OWL syntax provides annotation properties for this purpose.
The modal logic perspective

- Description logics can be understood from a modal logic perspective.

- Each pair of $\forall R$ and $\exists R$ statements give rise to a pair of modalities.

- Essentially, some description logics are multi-modal logics.

- See [The Description Logic Handbook].
The RDFS perspective

RDFS semantics is weaker

- :mary rdf:type :Person .
- :Mother rdfs:subClassOf :Woman .
- :john :hasWife :Mary .
- :hasWife rdfs:subPropertyOf :hasSpouse
- :hasWife rdfs:range :Woman .
- :hasWife rdfs:domain :Man .
- Person(mary)
- Mother ⊆ Woman
- hasWife(john,mary)
- hasWife ⊆ hasSpouse
- T ⊆ ∀hasWife.Woman
- T ⊆ ∀hasWife⁻.Man or ∃hasWife. T ⊆ Man

RDFS also allows to
- make statements about statements → only possible through annotations in OWL
- mix class names, individual names, property names (they are all URIs) → punning in OWL
Description logics impose *type separation*, i.e. names of individuals, classes, and properties must be disjoint.

In OWL 2 Full, type separation does not apply.

In OWL 2 DL, type separation is relaxed, but a class X and an individual X are interpreted semantically as if they were different.

Father(john)
SocialRole(Father)

See further below on the two different semantics for OWL.
Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- Expressivity Examples: Rules in OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools
- Outlook
- References
Expressivity Examples: Rules in OWL

- \(\text{Man}(x) \land \text{hasBrother}(x,y) \land \text{hasChild}(y,z) \rightarrow \text{Uncle}(x)\)
  - \(\text{Man} \equiv \exists \text{hasBrother}. \exists \text{hasChild}. \top \subseteq \text{Uncle}\)

- \(\text{kills}(x,x) \rightarrow \text{suicide}(x)\)
  - \(\exists \text{kills}. \text{Self} \subseteq \text{suicide}\)
  - \(\text{suicide} \subseteq \exists \text{kills}. \text{Self}\)

Note: with these two axioms,

\text{suicide} is basically the same as \text{kills}

- \(\text{NutAllergic}(x) \land \text{NutProduct}(y) \rightarrow \text{dislikes}(x,y)\)
  - \(\text{NutAllergic} \equiv \exists \text{nutAllergic}. \text{Self}\)
  - \(\text{NutProduct} \equiv \exists \text{nutProduct}. \text{Self}\)
  - \(\text{nutAllergic} \cup \text{nutProduct} \subseteq \text{dislikes}\)
Expressivity Examples: Rules in OWL

- dislikes(x, z) \land Dish(y) \land contains(y, z) \rightarrow dislikes(x, y)
  
  - Dish \equiv \exists\text{dish.Self}
  
  dislikes \circ contains \circ \neg \text{dislikes}

- worksAt(x, y) \land University(y) \land supervises(x, z) \land \text{PhDStudent(z)} \rightarrow \text{professorOf}(x, z)
  
  - \exists\text{worksAt.University} \equiv \exists\text{worksAtUniversity.Self}
  
  \text{PhDStudent} \equiv \exists\text{phDStudent.Self}
  
  worksAtUniversity \circ supervises \circ \text{phDStudent} \subseteq \text{professorOf}

- Basic requirement for expressibility of rules in OWL 2:
  tree-shapedness of rule bodies

- For more on this, see
  [Description Logic Rules] and [ELP].
Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- Expressivity Examples: Rules in OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools
- Outlook
- References
OWL Semantics

- There are two semantics for OWL.

1. Description Logic Semantics
   also: Direct Semantics; FOL Semantics
   Can be obtained by translation to FOL.
   Syntax restrictions apply! (see next slide)

2. RDF-based Semantics
   No syntax restrictions apply.
   Extends the direct semantics with RDFS-reasoning features.

In the following, we will deal with the direct semantics only.
OWL Direct Semantics

To obtain decidability, syntactic restrictions apply.

- Type separation / punning
- No cycles in property chains.
- No transitive properties in cardinality restrictions.
OWL Direct Semantics: Restrictions

- 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:
    \[
    \begin{align*}
    R \circ R \subset R \\
    S^- \subset R \\
    S_1 \circ S_2 \circ \ldots \circ S_n \subset R \\
    S_1 \circ S_2 \circ \ldots \circ S_n \circ R \subset R
    \end{align*}
    \]
  - thereby, $S_i \prec R$ for all $i = 1, 2, \ldots, n$.

- Example 1: $R \circ S \subset R \quad S \circ S \subset S \quad R \circ S \circ R \subset T$
  \[\rightarrow\ \text{regular with order } S \prec R \prec T\]
- Example 2: $R \circ T \circ S \subset T$
  \[\rightarrow\ \text{not regular because form not admissible}\]
- Example 3: $R \circ S \subset S \quad S \circ R \subset R$
  \[\rightarrow\ \text{not regular because no adequate order exists}\]
OWL Direct Semantics: Restrictions

- 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 $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$ 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: $Q \circ P \sqsubseteq R$  $R \circ P \sqsubseteq R$  $R \sqsubseteq S$  $P \sqsubseteq R$  $Q \sqsubseteq S$
  non-simple: $R, S$  simple: $P, Q$
OWL Direct Semantics

- model-theoretic semantics
- starts with interpretations
- an interpretation maps

individual names, class names and property names...

...into a domain
OWL Direct Semantics

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

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

- ...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 $R^I \subseteq S^I$
  - Dis$(R,S)$ 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 (with equality)...

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

...which (interpreted under FOL semantics) coincides with the definition just given.
Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- Expressivity Examples: Rules in OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools
- Outlook
- References
OWL Profiles

- OWL Full – using the RDFS-based semantics
- OWL DL – using the FOL semantics

The OWL 2 documents describe further profiles, which are of polynomial complexity:

- OWL EL  (EL++)
- OWL QL  (DL Lite$_R$)
- OWL RL  (DLP)
OWL 2 EL

- **allowed:**
  - subclass axioms with intersection, existential quantification, top, bottom
  - closed classs must have only one member
  - property chain axioms, range restrictions (under certain conditions)

- **disallowed:**
  - negation, disjunction, arbitrary universal quantification, role inverses

- Examples:
  - $\exists \text{has.Sorrow} \sqsubseteq \exists \text{has.Liqueur}$; $\top \sqsubseteq \exists \text{hasParent}.\text{Person}$
  - $\exists \text{married.}\top \cap \text{CatholicPriest} \sqsubseteq \bot$; $\text{German} \sqsubseteq \exists \text{knows.}\{\text{angela}\}$
  - $\exists \text{hasParent} \pm \exists \text{hasParent} \sqsubseteq \exists \text{hasGrandparent}$
OWL 2 RL

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

examples:

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

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

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

- \( \text{childOf} \pm \text{childOf} \subseteq \text{grandchildOf} \)
  
  rule version: \( \text{childOf}(x,y) \land \text{childOf}(y,z) \rightarrow \text{grandchildOf}(x,z) \)

- \( \text{Disj} (\text{childOf}, \text{parentOf}) \)
  
  rule version: \( \text{childOf}(x,y) \land \text{parentOf}(x,y) \rightarrow \)
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

- Reasoner conformance requires only soundness.
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:
  $\exists married. T \sqsubseteq \neg \text{Free} \cap \exists has. Sorrow$
Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- Expressivity Examples: Rules in OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools
- Outlook
- References
Proof Theory

- Traditionally using tableaux algorithms (see below)

Alternatives:
- Transformation to disjunctive datalog using basic superposition done for SHIQ
- Naive mapping to Datalog for OWL RL
- Mapping to SQL for OWL QL
- Special-purpose algorithms for OWL EL e.g. transformation to Datalog
Proof theory Via Tableaux

- Adaptation of FOL tableaux algorithms.

- Problem: OWL is decidable, but FOL tableaux algorithms do not guarantee termination.

- Solution: blocking.
DL Tableaux Termination Problem

TBox: $\exists R. T$
ABox: $T(a_1)$

- Is satisfiable:
  Model $M$ contains elements $a_1^M, a_2^M, \ldots$ and $R^M(a_i^M, a_{i+1}^M)$ for all $i \geq 1$.
- But naive tableau does not terminate!
Nothing essentially new happens.  
Idea: y does not need to be expanded, because it is basically a copy of x.

⇒ Blocking
Blocking (in ALC)

- y is *blocked* (by x) if
  - y is not an individual (but a variable),
  - y is a successor of x and \( L(y) \subseteq L(x) \),
  - or an ancestor of y is blocked.

\[ \begin{array}{c}
    a \xrightarrow{R} x \xrightarrow{R} y \\
    \text{R} \quad \text{R} \\
    \text{∃R.T} \quad \text{∃R.T} \quad \text{∃R.T}
  \end{array} \]

y blocked by x in this example.

Blocking conditions for more expressive DLs are more involved; the idea is the same.
Show that

\( C(a) \quad C(c) \)

\( R(a,b) \quad R(a,c) \)

\( S(a,a) \quad S(c,b) \)

\( C \subseteq \forall S.A \)

\( A \subseteq \exists R.\exists S.A \)

\( A \subseteq \exists R.C \)

implies \( \exists R.\exists R.\exists S.A(a) \).
**ALC Tableau Example**

TBox:
- \( \neg C \sqcup \forall S.A \)
- \( \neg A \sqcup \exists R.\exists S.A \)
- \( \neg A \sqcup \exists R.C \)

ABox
- \( C(a) \)
- \( C(c) \)
- \( R(a,b) \)
- \( R(a,c) \)
- \( S(a,a) \)
- \( S(c,b) \)

\(-\exists R.\exists S.A(a) \text{ is } \forall R.\forall S.\neg A(a)\)

Diagram: A network of relations and entities, including:
- **C**: \( \forall R.\forall S.\neg A \)
- **A**: \( \forall R.\forall S.\neg A \)
- **∃S.A**: \( \forall S.\neg A \)
- **A**: \( \neg A \)
- **R**: Relations between entities, such as \( R(a,b) \) and \( R(a,c) \)
- **S**: Relations between entities, such as \( S(a,a) \) and \( S(c,b) \)
- **a**, **b**, **c**, **x**, **y**: Entities connecting through relations
Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- Expressivity Examples: Rules in OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools
- Outlook
- References
OWL tools (incomplete listing)

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)

Editors:
- Protégé
- NeOn Toolkit
- TopBraid Composer
Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- Expressivity Examples: Rules in OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools
- Outlook
- References
Some Current Research Questions

- Integrating OWL and Rules
- Inconsistency handling / paraconsistent reasoning
- Local closed world reasoning
- Uncertainty handling (fuzzy / probabilistic)
- Modularization
- Distributedness
- Belief Revision (Ontology Evolution)
- Abduction/Explanation/Justification
- Approximate Reasoning
- Ontology Learning
- Modelling / Design Patterns
- Ontology Engineering (Modelling Processes)
- Interfaces (GUIs, Constrained Natural Language, etc.)
Further remarks

- Several major conferences on Semantic Web:
  - ISWC (>600), ESWC (>300), WWW Semantic Web track, IJCAI Semantic Web track, etc.
- Semantic Web languages taught in many university courses world-wide.
  - Becomes established topic.
- Industrial uptake currently happening
  - e.g. OWL reasoners by IBM, ORACLE
  - many application studies by major IT companies
  - considerable number of spin-offs
  - venture capital (e.g. VULCAN Inc.)
- Considerable uptake in the life sciences
- Substantial project funding (EU, NIH, etc.)
Suggestions for OWL?

- Annual Workshop OWL: Experiences and Directions
- Co-located with ISWC09 (just beforehand), October 2009.
- Usually >80 people, most of them doing applications. Major OWL language designers are there.
- Past discussions had major impact on OWL 2 → state your opinion there!
- Low paper barrier, position statements and experience reports welcome. Deadline July 24th.

Contents

- OWL – Basic Ideas
- OWL As the Description Logic SROIQ(D)
- Different Perspectives on OWL
- Expressivity Examples: Rules in OWL
- OWL Semantics
- OWL Profiles
- Proof Theory
- Tools
- Outlook
- References
Main References


- Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter Patel-Schneider, Sebastian Rudolph, OWL 2 Web Ontology Language: Primer. http://www.w3.org/TR/owl2-primer/

Main References – Textbooks


Further References

- DL complexity calculator: http://www.cs.man.ac.uk/~ezolin/dl/


Thanks!

http://semantic-web-grundlagen.de/wiki/IJCAI-09_Tutorial