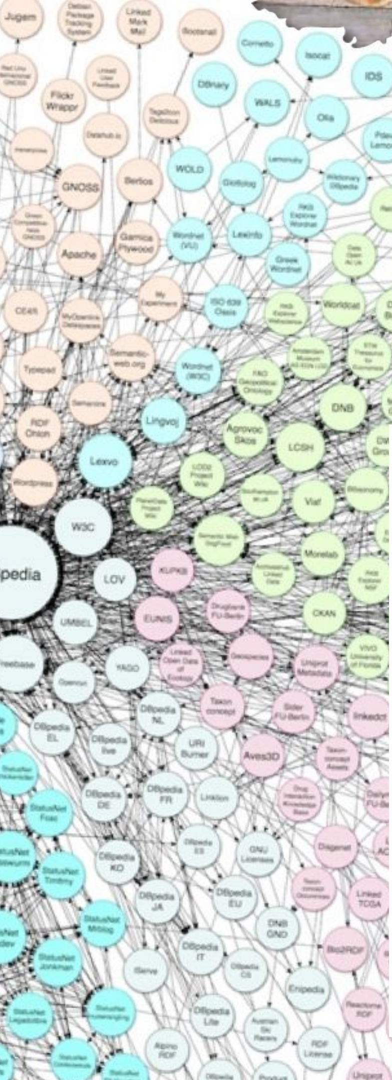

Knowledge Engineering with Semantic Web Technologies

Lecture 4: OWL, Rules and Reasoning

4.7 Rules - Is this more than Description Logics?

Dr. Harald Sack
Hasso Plattner Institute for IT Systems Engineering
University of Potsdam

Autumn 2015

Are Description Logics really enough?

- More expressivity also means more complexity
  - this might lead to undecidability (as for FOL)
- Do we really need more expressivity than OWL DL offers?
- Consider the following example:
  - *„A squanderer is a person whose expenses are higher than his income“*

```
● Squanderer ⊑ Person
● Squanderer ⊑ hasExpenses.⊤
● Squanderer ⊑ hasIncome.⊤
```

?

- We need a constructor to combine Classes and Properties
- **Problem:** Mixing of TBox and ABox

Rules and the Semantic Web

- The **Semantic Web** concentrates on **declarative forms of knowledge representation**
  - Description Logics, OWL, RDFS
- **Rules** are a common form of **procedural knowledge representation** in Knowledge Engineering
  - Expert Systems
  - Prolog, CLIPS, JESS, OPS, …
- Knowledge representation formalisms of the Semantic Web have **expressive limitations** which can be overcome by rule-based knowledge
  - e.g. composition of complex classes from classes and properties

What are Rules?

```
IF   A ...  THEN   B ...
     A       →     B
```

- Interpretation of a rule depends on context
  - General Inference:
    Premise → Conclusion
  - Hypothesis:
    Cause → Effect
  - Production:
    Condition → Action

What are Rules?

- **Logical Rules** (FOL implication):
  - $F \rightarrow G$ is equivalent with $\neg F \vee G$
  - Logical extension of the KB (static)
  - Open World, declarative
- **Procedural Rules** (e.g. Production Rules):
  - `If X then Y else Z`
  - executable machine instructions (dynamic)
  - operational (semantics = effect at application)
- **Logic Programming Rules** (e.g. Prolog, F-Logic):
  - „`woman(X) <- person(X) AND NOT man(X)`“
  - Approximation of logical semantics with operational aspects
  - Closed World (mostly), semi-declarative

FOL as Rule Language

- Rules as FOL implications (Horn Clause)

$$\underbrace{A_1 \wedge A_2 \wedge \dots \wedge A_n}_{\text{Body}} \rightarrow \underbrace{H}_{\text{Head}}$$

- semantically equivalent with $\neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_n \vee H$
- where A_i, H are atomic formulas
- Quantification most times omitted,
  free variables are considered to be universally quantified
  - i.e. the rule holds for all possible assignments

FOL as Rule Language

- Rules as FOL implications (Horn Clause)

$$\underbrace{H}_{\text{Head}} \leftarrow \underbrace{A_1 \wedge A_2 \wedge \ldots \wedge A_n}_{\text{Body}}$$

*often written also from right to left (← or **:-**)*

Variants of FOL Rules

- **Lloyd-Topor Transformation**
  - Several atoms in the head are usually considered as conjunction

$$A_1 \quad A_2 \quad \ldots \quad A_n \quad \rightarrow \quad H_1 \quad H_2 \quad \ldots \quad H_m$$

- is equivalent to

$$A_1 \quad A_2 \quad \ldots \quad A_n \quad \rightarrow \quad H_1$$
$$A_1 \quad A_2 \quad \ldots \quad A_n \quad \rightarrow \quad H_2$$
$$\ldots$$
$$A_1 \quad A_2 \quad \ldots \quad A_n \quad \rightarrow \quad H_m$$

Variants of FOL Rules

- **Disjunctive Rules**
  - Disjunction of several non-negated Atoms

$$\underbrace{A_1 \land A_2 \land \ldots \land A_n}_{\text{Body}} \rightarrow \underbrace{H_1 \lor H_2 \lor \ldots \lor H_m}_{\text{Head}}$$

- reverse implication, as e.g.
  „if I see someting, then the light is on or the sun is shining“

Variants of FOL Rules

- **FOL Rules**
  - **Clause:** Disjunction of atomic formulas or negated atomic formulas
  - **Horn Clause:** Clause with **at most one** not negated atom

    $\neg p \lor \neg q \lor ... \lor \neg t \lor u$ can be written as

    $p \land q \land ... \land t \rightarrow u$
- **Definite Clause:** Clause with **exactly one** not negated atom
- **Fact:** Clause of **a single** not negated atom

Variants of FOL Rules

- **Examples**
  - Person(x) $\rightarrow$ Woman(x) $\vee$ Man(x) [clause]
  - Man(x) $\wedge$ hasChild(x,y) $\rightarrow$ Father(x) [definite clause]
  - hasBrother(mother(x),y) $\rightarrow$ isUncle(x,y) [with function symbol]
  - Man(x) $\wedge$ Woman(x) $\rightarrow$ [horn clause]
  - Woman(Anja) [fact]

Example Revisited

- Consider the following example:
  - *„A squanderer is a person whose expenses are higher than his income“*

```
Person(x) ∧ hasIncome(x,y)
          ∧ hasExpenses(x,z)
          ∧ (z > y)
          → Squanderer(x)
```

- Arithmetics can be part of rules and modeled like a predicate
  - $(z > y) \triangleq \text{greaterThan}(z,y)$

Description Logics vs. Rules

- Rules are usually considered to apply only to **known** constants.
- No possibility to „create“ new things „on the flight“ by using existential quantification $\exists$

$$\text{Human} \sqsubseteq \exists \text{hasParent.Human}$$

- If rules are considered FOL formulas, then combining rules with $\mathcal{ALC}$ leads to undecidability.
- What about decidable FOL-Rules....?
  - ▶ DATALOG

DATALOG

- is a **logical rule language** that consists of
  - **horn clauses without function symbols**
  - conjunction, constants, universally quantified variables, predicate symbols
  - no disjunction, no negation, no existential quantification, no function symbols
- originally developed as foundation of deductive databases (Gallaire, Minkers, 1978)
- Knowledge Bases (Datalog Programs)
  are sets of horn clauses (without function symbols)
- DATALOG is decidable and computationally efficient, ExpTime

DATALOG Syntax

- DATALOG **Term**: constant **c** or variable **v**
- DATALOG **Atom**: $\mathbf{p(t_1,\ldots,t_n)}$
  with predicate p, and terms $t_1,\ldots,t_n$
- DATALOG **Rule**: $\forall \mathbf{x_1} \ldots \forall \mathbf{x_n}\ (\mathbf{B_1} \land \ldots \land \mathbf{B_n} \rightarrow \mathbf{H})$
  with $B_1,\ldots,B_n$, H atoms and $x_1,\ldots,x_n$ variables
- DATALOG **Program**: set of DATALOG rules

DATALOG Examples

- `Vegetarian(x)∧FishProduct(y) → dislikes(x,y)`
- `orderedDish(x,y)∧dislikes(x,y) → Unhappy(x)`
- `orderedDish(x,y) → Dish(y)`
- `dislikes(x,z)∧Dish(y)∧contains(y,z)→ dislikes(x,y)`
- `→ Vegetarian(Matthias)`
- `Happy(x) ∧ Unhappy(x) →`

- DATALOG Rules allow mixing classes and relations (i.e. unary and binary predicates)
- therefore it can be more expressive than DL
- A combination of DATALOG and OWL is the SWRL Language

08 Putting Everything Together

OpenHPI - Course Knowledge Engineering with Semantic Web Technologies

Lecture 4: OWL, Rules, and Reasoning