UNIVERSITY OF
Southampton

School of Electronics
and Computer Science

Rules

Dr Nicholas Gibbins
nmg@ecs.soton.ac.uk

32/3019

The Role of Rules

- Semantic Web concentrates on declarative forms of knowledge representation
    - OWL, RDF Schema
- Rules are a common form of knowledge representation elsewhere in Knowledge Engineering
    - Expert Systems – CLIPS, JESS, etc

The Role of Rules

- The KR formalisms of the Semantic Web have expressive limitations which can be overcome by rule-based knowledge
- For example, we cannot express the fact that a person’s parent’s brother is the person’s uncle in either RDFS or OWL (including OWL Full)
    - No role composition in OWL 1.0

The Role of Rules

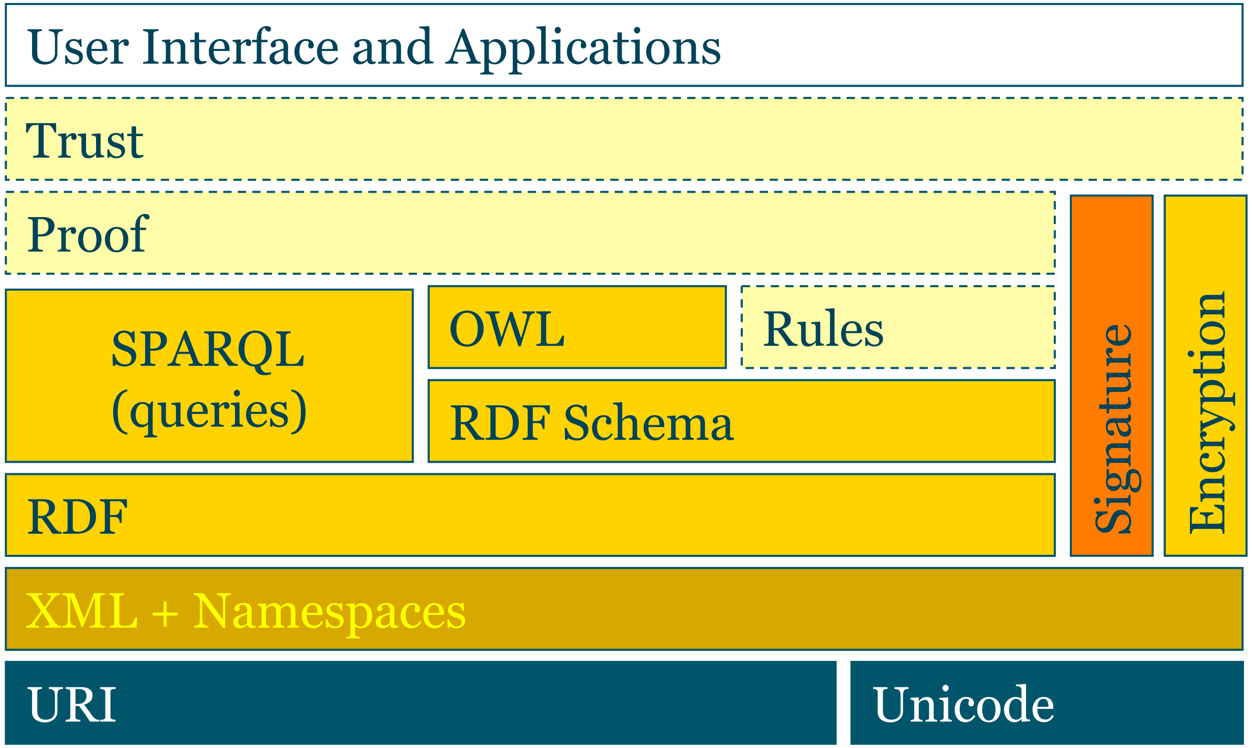
- Trivial to express in a language like Prolog:
-

```
          hasUncle(X,Y)          :-          hasParent(X,Z),
                                             hasBrother(Z,Y).

hasBrother(X,Y)    :-                        isMale(Y),
                                             hasParent(X,Z),
                                             hasParent(Y,Z).
```

The Semantic Web layer cake

UNIVERSITY OF Southampton
School of Electronics
and Computer Science



User Interface and Applications

Trust

Proof

SPARQL (queries)

OWL

Rules

RDF Schema

RDF

Signature

Encryption

XML + Namespaces

URI

Unicode

Attribution

Explanation

Ontologies + Inference

Metadata

Standard syntax

Identity

Rules and the Semantic Web

- Several proposed rule languages for use with the SW
  - RuleML
  - (N3 Rules)
  - (Jena Rules)
  - Semantic Web Rule Language (SWRL)
  - Rule Interchange Format (RIF)

Rule Format

- The majority of rules in rule-based systems are of the form:

  $A \Leftarrow B1 \wedge B2 \wedge \ldots \wedge Bn$

- A is known as the consequent or head of the rule
- B1…Bn are known as the antecedents or body of the rule

- Also known as Horn Clauses (disjunction with at most one positive literal)

Description Logics and Rules

- Some work on designing DLs which include trigger rules of the form:

  $C \Rightarrow D$

  (if an individual is a member of C, then it must be a member of D

Description Logics and Rules

- $C \Rightarrow D$ is not the same as saying $C \sqsubseteq D$
  (every instance of C is an instance of D)
- $C \sqsubseteq D$ is equivalent to saying $\neg D \sqsubseteq \neg C$ (contrapositive)
- The trigger rule $C \Rightarrow D$ is not equivalent to $\neg D \Rightarrow \neg C$
- DLs with rules include an epistemic (modal) operator K:
  - KC can be read as “the class of things which are **known** to be of class C”
  - $C \Rightarrow D$ is equivalent to $KC \sqsubseteq D$
  - Used as a foundation for SWRL, etc

N3 Rules

- Defines log: namespace for logical operators
- Not widely implemented (cwm + ?)
- log: namespace puts ontology into OWL Full

```
{?x ont:parent ?y. ?y ont:brother ?z. } log:implies {?x ont:uncle ?z. }.
```

Jena Rules

- Jena RDF/OWL library contains support for forward- and backward-chaining rules:

```
# Example rule file
@prefix ont: <http://example.org/ontology#>.
@include <RDFS>.

[rule1: (?f ont:parent ?a) (?u ont:brother ?f) -> (?u ont:uncle ?a)]
```

- Only implemented in Jena

SWRL

- Submitted to W3C in 2004
- Based on RuleML subset and OWL
- XML and RDF-based serialisations (also, human-readable abstract syntax)
- Obeys constraints put on OWL re: disjointness of instances and datatype values
- Two types of variable in expressions
  - I-variable – matches class instances
  - D-variable – matches datatype values

SWRL Rule Example

hasParent(?x1,?x2) ∧ hasBrother(?x2,?x3) ⇒ hasUncle(?x1,?x3)

- In abstract syntax:

```
Implies(Antecedent(hasParent(I-variable(x1) I-variable(x2))
                   hasBrother(I-variable(x2) I-variable(x3)))
        Consequent(hasUncle(I-variable(x1) I-variable(x3))))
```

SWRL Rule Example

Artist(?x) ∧ artistStyle(?x,?y) ∧ Style(?y) ∧ creator(?z,?x) ⇒
style/period(?z,?y)

```
Implies(Antecedent(Artist(I-variable(x))
                   artistStyle(I-variable(x) I-variable(y))
                   Style(I-variable(y))
                   creator(I-variable(z) I-variable(x)))
     Consequent(style/period(I-variable(z) I-variable(y))))
```

SWRL Rule Example

Artist(?x) ∧ (≤1 artistStyle)(?x) ∧ creator(?z,?x) ⇒
(≤1 style/period)(?z)

```
Implies(Antecedent(Artist(I-variable(x))
                   (restriction(artistStyle maxCardinality(1)))
                           (I-variable(x))
                   Style(I-variable(y))
                   creator(I-variable(z) I-variable(x)))
        Consequent((restriction(style/period maxCardinality(1))
                           (I-variable(z))))
```

SWRL XML Syntax

- Based on OWL XML Presentation Syntax (with RuleML)

```
<ruleml:imp>
        <ruleml:_rlab ruleml:href="#example1"/>
        <ruleml:_body>
                <swrlx:individualPropertyAtom  swrlx:property="hasParent">
                        <ruleml:var>x1</ruleml:var>
                        <ruleml:var>x2</ruleml:var>
                </swrlx:individualPropertyAtom>
                <swrlx:individualPropertyAtom  swrlx:property="hasBrother">
                        <ruleml:var>x2</ruleml:var>
                        <ruleml:var>x3</ruleml:var>
                </swrlx:individualPropertyAtom>
        </ruleml:_body>
        <ruleml:_head>
                <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
                        <ruleml:var>x1</ruleml:var>
                        <ruleml:var>x3</ruleml:var>
                </swrlx:individualPropertyAtom>
        </ruleml:_head>
</ruleml:imp>
```

SWRL RDF Syntax

```
<swrl:Variable rdf:ID="x1"/>
<swrl:Variable rdf:ID="x2"/>
<swrl:Variable rdf:ID="x3"/>
<ruleml:Imp>
        <ruleml:body rdf:parseType="Collection”>
                <swrl:IndividualPropertyAtom>
                        <swrl:propertyPredicate rdf:resource="&eg;hasParent"/>
                        <swrl:argument1 rdf:resource="#x1" />
                        <swrl:argument2 rdf:resource="#x2" />
                </swrl:IndividualPropertyAtom>
                <swrl:IndividualPropertyAtom>
                        <swrl:propertyPredicate rdf:resource="&eg;hasSibling"/>
                        <swrl:argument1 rdf:resource="#x2" />
                        <swrl:argument2 rdf:resource="#x3" />
                </swrl:IndividualPropertyAtom>
        </ruleml:body>
        …
```

Rule Interchange Format

- W3C Working Group chartered in late 2005
- More expressive language than SWRL
  - Common core with extensions
- Two phases of standardisation:
  1. Core language (due May 2007)
  2. Standard extensions (due June 2008, June 2009)
- Some delays in first phase
  - Basic Logic Dialect still at Working Draft (July 2008)
  - Next release due May 2009 (mostly LC and CR)

Rule Interchange Format

- Defines XML syntax and non-XML presentation syntax (c.f. OWL)

- Latest version from:
  http://www.w3.org/2005/rules/wiki/RIF_Working_Group

RIF Presentation Syntax

```
Document(
    Prefix(cpt http://example.com/concepts#)
    Prefix(ppl http://example.com/people#)
    Prefix(bks http://example.com/books#)
    Group (
            Forall ?Buyer ?Item ?Seller (
                        cpt:buy(?Buyer ?Item ?Seller) :-
                                    cpt:sell(?Seller ?Item ?Buyer)
            )
            cpt:sell(ppl:John bks:LeRif ppl:Mary)
    )
)
```