The Semantic Web

Lecture 7

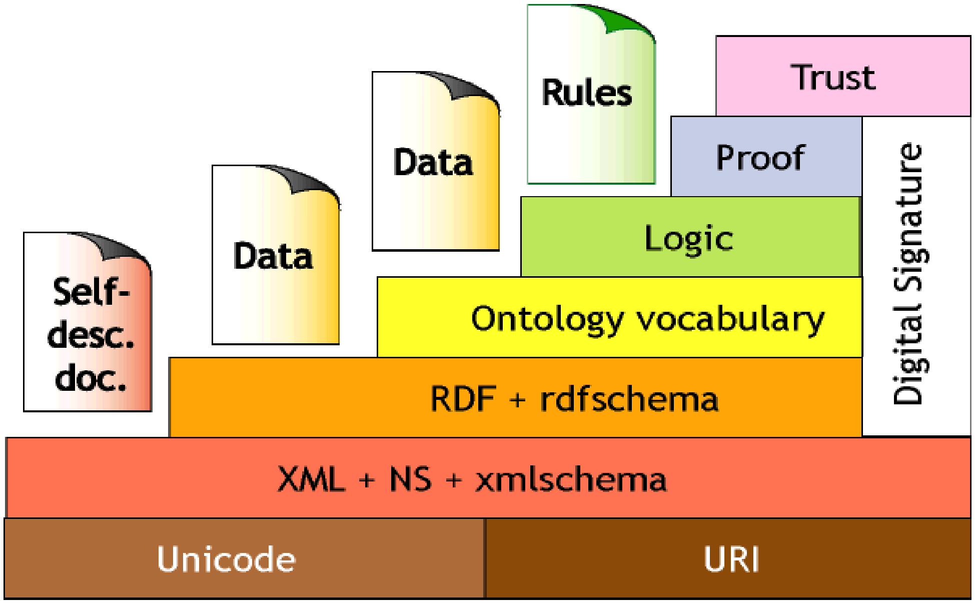
The upper layers

Riccardo Rosati

Dottorato in Ingegneria Informatica
Sapienza Università di Roma

a.a. 2006/07

The Semantic Web Tower



Trust

Rules

Data

Proof

Data

Logic

Self-
desc.
doc.

Ontology vocabulary

Digital Signature

RDF + rdfschema

XML + NS + xmlschema

Unicode

URI

The Logic/Proof/Trust layers

SW Stack upper layers:

- the Logic layer enables the writing of **rules**
- the Proof layer executes the rules
- the Trust layer decides whether to trust the given proof or not

technology for these layer at a very early stage:

- no standards yet
- open architectural issues

The notion of rule

- rule = “if-then” statement
- a rule can be static or dynamic
  - static rule (implication): if condition C1 is true then conclude that also condition C2 is true
  - dynamic rule: if event E occurs and condition C holds then execute action A
- semantics of rules:
  - procedural (operational)
  - declarative

Rule bases as knowledge bases

- static rules may be considered as statements expressing knowledge
- rule base = knowledge base
- interpretation of a rule similar (but not equal) to the boolean implication operator
- “constructive” (one-way) implication (contrapositive does not hold)
- more generally, semantics of rules based on (various notions of) **closed-world assumption**

Expressive limitations of DLs and OWL

- the typical expressiveness of DLs does not allow for addressing the following aspects:
- defining **predicates of arbitrary arity** (not just unary and binary) using **variable quantification** beyond the tree-like structure of DL concepts (many DLs are subsets of the two-variable fragment of FOL)
- formulating **expressive queries** over DL knowledge bases (beyond concept subsumption and instance checking)
- formalizing various **forms of closed-world reasoning** over DL KBs
- more generally, expressing forms of **nonmonotonic knowledge**, like default rules

Rule formalisms

- static rules:
  - logic programming languages:
    - Prolog
    - answer set programs
    - nonmonotonic Datalog
- dynamic rules:
  - ECA rules
  - production rules
  - ...

Logic programming: Prolog

Prolog rule: statement of the form

- a **:** – b1, b2, ... , bn
- intuitive reading: “if a then b1 and b2 and ... and bn”
- a = rule head
- b1, b2, ... , bn = rule body
- a and all bi’s are first-order atoms
- some bi may be negated

Logic programming

examples:

- uncle(x,y) :- father(x,z), brother(z,y).
- grandparent(x,y) :- parent(x,y), parent(y,z).

recusive rules:

- ancestor(x,y) :- parent(x,y).
- ancestor(x,y) :- parent(x,z), ancestor(z,y).

use of negation:

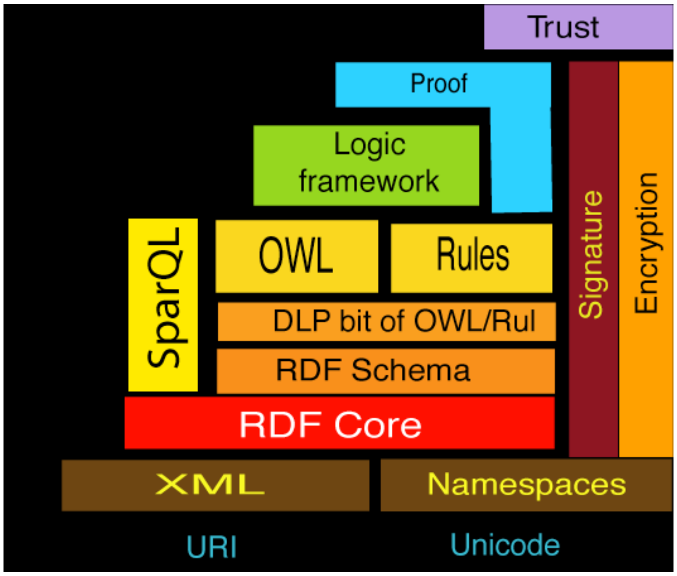
- innocent(x) :- person(x), not guilty(x),

Rules as an alternative ontology language

general idea: use rules as an ontology language

- first proposal: use rule-based languages **instead** of OWL
  - change of the Semantic Web Stack
- second proposal: use rule-based languages AND OWL as ontology languages
  - different change of the Stack (two-stack)
  - rules are not **on top** of OWL anymore, they are **besides** OWL

One-stack vs. two-stack architecture



Trends

- Trust layer: research at a very early stage
- strictly depends on the choices concerning the lower layers
- some preliminary results:
  - provenance / pinpointing in Description Logics: finding the explanation for an answer
  - techniques for authorization (not yet specific for DLs)
  - quality of the answers / ranking (top-k answers)