The Semantic Web

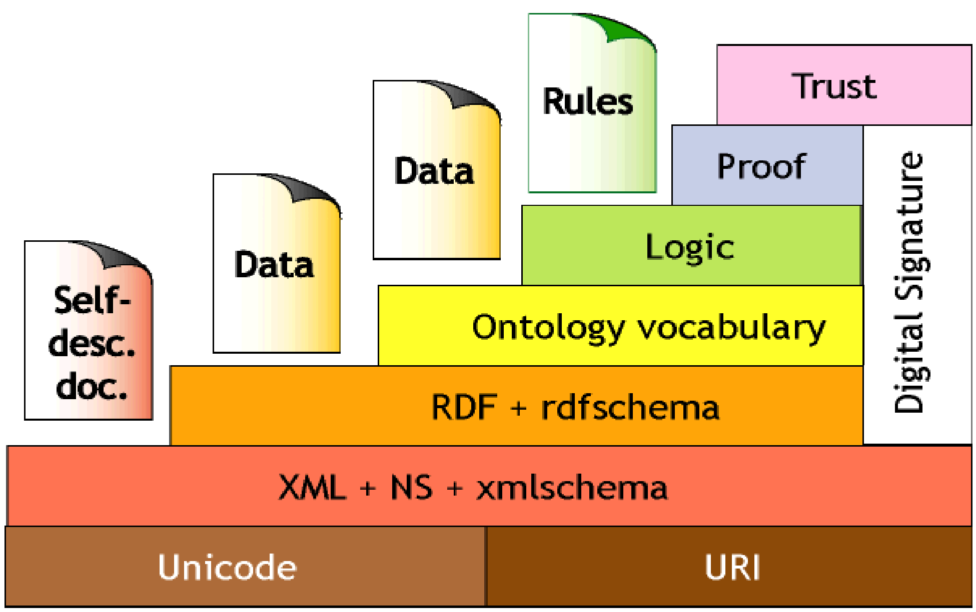
Lecture 5

The ontology layer 2: Reasoning in OWL, tractable fragments of OWL

Riccardo Rosati

Dottorato in Ingegneria Informatica
Sapienza Università di Roma
a.a. 2006/07

The Semantic Web Tower



OWL language

- Three species of OWL
  - OWL full is union of OWL syntax and RDF
  - OWL DL restricted to FOL fragment
  - OWL Lite is "easier to implement" subset of OWL DL
- OWL DL based on SHIQ Description Logic
  - In fact it is equivalent to SHOIN(D_n) DL
- OWL DL Benefits from many years of DL research
  - Well defined semantics
  - Formal properties well understood (complexity, decidability)
  - Known reasoning algorithms
  - Implemented systems (highly optimised)

OWL class constructors

Constructor	DL Syntax	Example	Modal Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1 \wedge \ldots \wedge C_n$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1 \vee \ldots \vee C_n$
complementOf	$\neg C$	¬Male	$\neg C$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x_1 \vee \ldots \vee x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$[P]C$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\langle P \rangle C$
maxCardinality	$\leqslant nP$	⩽1hasChild	$[P]_{n+1}$
minCardinality	$\geqslant nP$	⩾2hasChild	$\langle P \rangle_n$

Arbitrarily complex nesting of constructors:

- E.g., Person ⊓ ∀hasChild.Doctor ⊔∃hasChild.Doctor

DL knowledge bases (ontologies)

- An OWL ontology maps to a DL Knowledge Base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
  - $\mathcal{T}$(Tbox) is a set of axioms of the form:
    - $C \sqsubseteq D$ (concept inclusion)
    - $C \equiv D$ (concept equivalence)
    - $R \sqsubseteq S$ (role inclusion)
    - $R \equiv S$ (role equivalence)
    - $R^+ \sqsubseteq R$ (role transitivity)
  - $\mathcal{A}$(Abox) is a set of axioms of the form
    - $x \in D$ (concept instantiation)
    - $\langle x,y \rangle \in R$ (role instantiation)

DL vs. First-Order Logic

- in general, DLs correspond to decidable subclasses of first-order logic (FOL)
- DL KB = first-order theory
- OWL Full is NOT a FOL fragment!
  - reasoning in OWL Full is undecidable
- OWL-DL and OWL-Lite are decidable fragments of FOL

DL vs. First-Order Logic

let $\mathcal{K} = \langle \mathcal{T},\mathcal{A} \rangle$ be an ontology about persons where:

- $\mathcal{T}$ contains the following inclusion assertions:

  MALE $\sqsubseteq$ PERSON

  FEMALE $\sqsubseteq$ PERSON

  MALE $\sqsubseteq \neg$ FEMALE

  PERSON $\sqsubseteq \exists$Father^{-}.MALE

- $\mathcal{A}$ contains the following instance assertions:

  MALE(Bob)

  PERSON (Mary)

  PERSON(Paul)

DL vs. First-Order Logic

- $\mathcal{T}$ corresponds to the following FOL sentences:

  $\forall$ x. MALE(x) $\rightarrow$ PERSON(x)

  $\forall$ x. FEMALE(x) $\rightarrow$ PERSON(x)

  $\forall$ x. MALE(x) $\rightarrow$ $\neg$FEMALE(x)

  $\forall$ x. PERSON(x) $\rightarrow$ $\exists$y. Father(y,x) and MALE(y)

- $\mathcal{A}$ corresponds to the following FOL ground atoms:

  MALE(Bob)

  PERSON (Mary)

  PERSON(Paul)

Inference tasks

- Knowledge is correct (captures intuitions)
  - C subsumes D w.r.t. $\mathcal{K}$ iff for *every* model $\mathcal{I}$ of $\mathcal{K}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- Knowledge is minimally redundant (no unintended synonyms)
  - C is equivalent to D w.r.t. $\mathcal{K}$ iff for *every* model $\mathcal{I}$ of $\mathcal{K}$, $C^{\mathcal{I}} = D^{\mathcal{I}}$
- Knowledge is meaningful (classes can have instances)
  - C is satisfiable w.r.t. $\mathcal{K}$ iff there exists *some* model $\mathcal{I}$ of $\mathcal{K}$ s.t. $C^{\mathcal{I}} \neq \emptyset$
- Querying knowledge
  - x is an instance of C w.r.t. $\mathcal{K}$ iff for *every* model $\mathcal{I}$ of $\mathcal{K}$, $x^{\mathcal{I}} \in C^{\mathcal{I}}$
  - $\langle x,y \rangle$ is an instance of R w.r.t. $\mathcal{K}$ iff for, *every* model $\mathcal{I}$ of $\mathcal{K}$, $(x^{\mathcal{I}},y^{\mathcal{I}}) \in R^{\mathcal{I}}$
- Knowledge base consistency
  - A KB $\mathcal{K}$ is consistent iff there exists *some* model $\mathcal{I}$ of $\mathcal{K}$

Inference tasks

- OWL-DL ontology = first-order logical theory
- verifying the formal properties of the ontology corresponds to **reasoning** over a first-order theory

Consistency of the ontology

- Is the ontology K=(T,A) consistent (non-self-contradictory)?
- i.e., is there at least a model for K?
- intensional + extensional reasoning task
- fundamental formal property:
- inconsistent ontology => there is a semantic problem in K!
- K must be repaired

Consistency of the ontology

Example TBox:

MALE ⊑ PERSON

FEMALE ⊑ PERSON

MALE ⊑¬ FEMALE

PERSON ⊑∃hasFather.MALE

PERSON ⊑∃hasMother.FEMALE

hasMother ⊑ hasParent

hasFather ⊑ hasParent

∃hasParent.BLACK-EYES ⊑ BLACK-EYES

Consistency of the ontology

Example ABox:

MALE(Bob)
MALE(Paul)
FEMALE(Ann)
hasFather(Paul,Ann)
hasMother(Mary,Paul)
BLACK-EYES(Mary)
$\neg$ BLACK-EYES(Ann)

$\Rightarrow$ TBox + ABox **inconsistent** (Ann should have black eyes)

Concept consistency

- is a concept definition C consistent in a TBox T?
- i.e., is there a model of T in which C has a non-empty extension?
- intensional (schema) reasoning task
- detects a fundamental modeling problem in T:
  - if a concept is not consistent, then it can never be populated!

Concept subsumption

- is a concept C subsumed by another concept D in T?
- i.e., is the extension of C contained in the extension of D in every model of T?
- intensional (schema) reasoning task
- allows to do classification of concepts (i.e., to construct the concept ISA hierarchy)

Instance checking

- is an individual a a member of concept C in K?
- i.e., is the fact C(a) satisfied by every interpretation of K?
- intensional + extensional reasoning task
- basic “instance-level query” (tell me if object a is in class C)

Instance retrieval

- find all members of concept C in K
- i.e., compute all individuals a such that C(a) is satisfied by every interpretation of K
- intensional + extensional reasoning task
- (slight) generalization of instance checking

Conjunctive query answering

- compute the answers to a conjunctive query q in K
- i.e., compute all tuples of individuals t such that q(t) is entailed by K (= q(t) is satisfied by every interpretation of K)
- extensional + extensional reasoning task
- generalization of instance checking and instance retrieval
- i.e., database queries over ontologies

Inference tasks

- reasoning in OWL-DL is decidable (and the complexity is characterized)
- however: high computational complexity (EXPTIME)
- (optimized) reasoning algorithms developed
- OWL-DL reasoning tools implemented

Current OWL technology

two kinds of tools:

- OWL editors ("environments")
- OWL reasoners

OWL editors

- allow for visualizing/browsing/editing OWL ontologies
- able to connect to an external OWL reasoner

  => OWL "environments"
- main current tools:
  - Protege
  - SWOOP
  - OWLed2

OWL reasoning tools

two categories:

- OWL-DL reasoners
  - Racer, RacerPro
  - Pellet
  - Fact++
  - KAON2
- reasoners for "tractable fragments" of OWL-DL
  - QuOnto
  - OntoSearch2

OWL-DL reasoning tools

- all tools support "standard" reasoning tasks, i.e.:
  - consistency of the ontology
  - concept consistency
  - concept subsumption and classification
  - instance checking and retrieval
- they do not fully support conjunctive queries
- problem: the "official" query language for OWL has not been defined yet

Limits of current OWL-DL reasoners

- performance of OWL-DL reasoners:
- "practically good" for the intensional level
  - the size of a TBox is not likely to scale up too much
- not good for the extensional level
  - unable to handle instances (ABoxes) of large size (or even medium size)...
  - ...even for the basic extensional service (instance checking)

Limits of current OWL-DL reasoners

- why are these tools so bad with (large) ABoxes?
- two main reasons:
- current algorithms are mainly derived by algorithms defined for purely intensional tasks
  - no real optimization for ABox services
- these algorithms work in main memory => bottleneck for very large instances

OWL-DL technology vs. large instances

- the current limits of OWL-DL reasoners make it impossible to use these tools for real data integration on the web
- web sources are likely to be data intensive sources
- e.g., relational databases accessed through a web interface
- on the other hand, data integration is the prominent (future) application for Semantic Web technology! [Berners-Lee et al., IEEE Intelligent Systems, May 2006]

A solution: tractable OWL fragments

- how to overcome these limitations if we want to build data-intensive Semantic Web applications?
- solution 1: limit the expressive power of the ontology language

  => **tractable fragments** of OWL
- solution 2: wait for more efficient OWL-DL reasoners
- to arrive at solution 2, we may benefit from the new technology developed for OWL tractable fragments

Tractable OWL fragments

- idea: sacrifice part of the expressiveness of the ontology language...
- ...to have more efficient ontology tools
- OWL Lite is a standardized fragment of OWL-DL
- is OWL Lite OK?
- NO! it is still too expressive for ABox reasoning
- OWL Lite is not really "lite"!

Tractable OWL fragments

- other fragments of OWL-DL have been proposed
- open problem (no standard yet)
- main current proposals:
  - DL-Lite
  - EL
  - Horn-SHIQ
  - DLP

DL-Lite

- DL-Lite is a tractable OWL-DL fragment
- defined by the DIS-Sapienza DASI research group
- main objectives:
  - allow for very efficient treatment of large ABoxes...
  - ...even for very expressive queries (conjunctive queries)

DL-Lite syntax

- concept expressions:
  - atomic concept
  - role domain
  - role range
- DL-Lite TBox = set of
  - concept inclusions
  - functional assertions (stating that a role is functional
- DL-Lite ABox = set of ground atoms, i.e., assertions A(a), R(a,b) A = concept name, R = role name

DL-Lite abilities

tractability of TBox reasoning:

- all TBox reasoning tasks in DL-Lite are tractable, i.e., solvable in polynomial time

tractability of ABox+TBox reasoning:

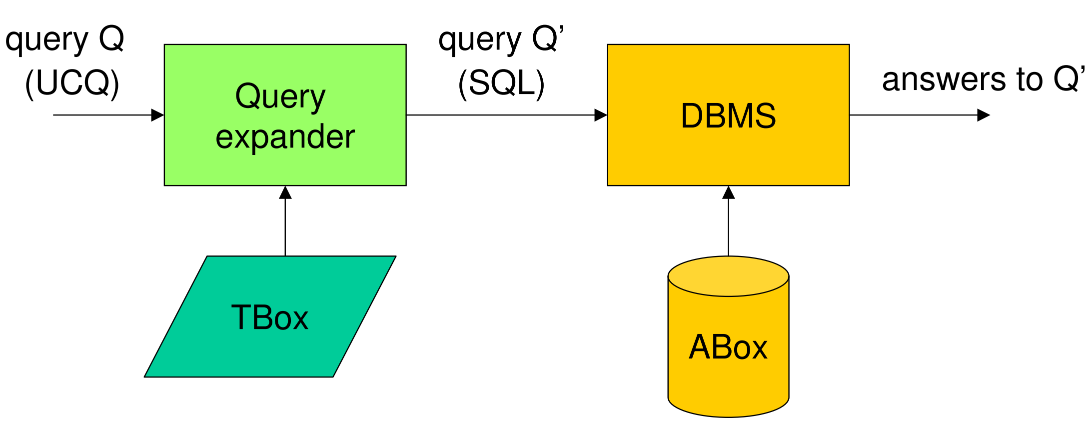
- instance checking and instance retrieval in DL-Lite are solvable in polynomial time
- conjunctive queries over DL-Lite ontologies can be answered in polynomial time (actually in LogSpace) with respect to *data complexity* (i.e., the size of the ABox)

Query answering in DL-Lite

a glimpse on the query answering algorithm:

- query answering in DL-Lite can be reduced to evaluation of an SQL query over a relational database
- query answering by query rewriting + relational database evaluation:
  1. the ABox is stored in a relational database (set of unary and binary tables)
  2. the conjunctive query Q is rewritten with respect to the TBox, obtaining an SQL query Q’
  3. query Q’ is passed to the DBMS which returns the answers

Query answering in DL-Lite



Example

TBox:

MALE $\sqsubseteq$ PERSON
MALE $\sqsubseteq \neg$FEMALE
$\exists$hasFather$^-$ $\sqsubseteq$ MALE
$\exists$hasMother$^-$ $\sqsubseteq$ FEMALE
FEMALE $\sqsubseteq$ PERSON
PERSON $\sqsubseteq$ $\exists$hasFather
PERSON $\sqsubseteq$ $\exists$hasMother

input query:
q(x) $\leftarrow$ PERSON(x)

rewritten query:
q'(x) $\leftarrow$ PERSON(x) $\vee$
FEMALE(x) $\vee$
MALE(x) $\vee$
hasFather(y,x) $\vee$
hasMother(y,x)

Example

rewritten query:
q'(x) $\leftarrow$ PERSON(x) $\vee$
FEMALE(x) $\vee$
MALE(x) $\vee$
hasFather(y,x) $\vee$
hasMother(y,x)

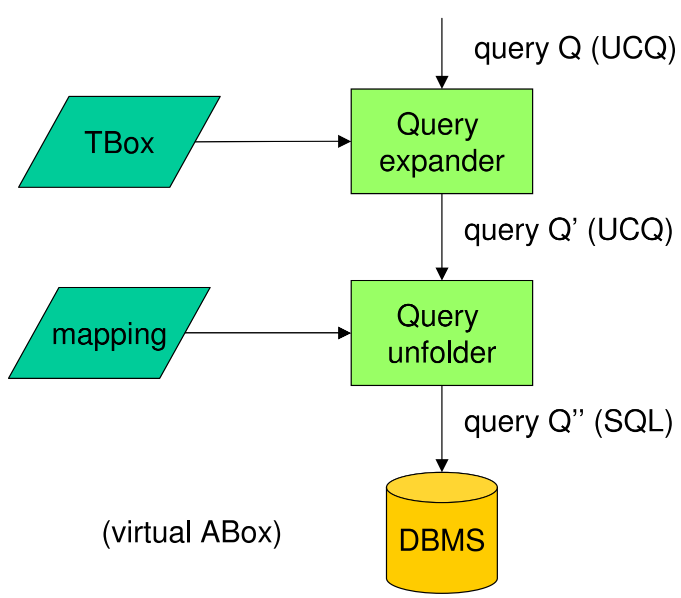
ABox:
MALE(Bob)
MALE(Paul)
FEMALE(Ann)
hasFather(Paul,Ann)
hasMother(Mary,Paul)

answers to query:
{ Bob, Paul, Ann, Mary }

QuOnto

- QuOnto is a reasoner for DL-Lite
- developed by DASI lab at DIS-Sapienza
- implements the above answering technique for conjunctive queries
- able to deal with very large instances (comparable to standard relational databases!)
- currently used in MASTRO, a system for ontology-based data integration

MASTRO (single database)



MASTRO-I (data integration)