The Semantic Web

Lecture 10

Reasoning techniques for ontologies

Riccardo Rosati

Dottorato in Ingegneria Informatica
Sapienza Università di Roma

a.a. 2006/07

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)
¬ BLACK-EYES(Ann)

⇒ 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

Reduction of reasoning tasks

- can reasoning task T1 be reduced to task T2?
- e.g., concept consistency can be reduced to KB consistency:
  - C consistent with respect to TBox T iff the KB (T,{C(a)}) is consistent
  - e.g., concept subsumption $C \sqsubseteq D$ w.r.t. T can be reduced to instance checking (T, {C(a)}) $\models$ D(a) (where a does not occur in T)

Classes of reasoning tasks

The reasoning tasks can be divided in:

- purely intensional tasks (concept consistency, concept subsumption)
- basic extensional tasks (KB consistency, instance checking)
- advanced extensional tasks (conjunctive query answering)

Reasoning techniques for ontologies

- Tableaux: this is the most mature technique, used in systems such FACT/Racer/Pellet
- Automata on infinite tree: the most powerful technique, but not implemented
- Structural analysis: simple, but works only for the weakest languages
- Other: e.g., specialized chase-based techniques for conjunctive queries, used for example in QuOnto

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)

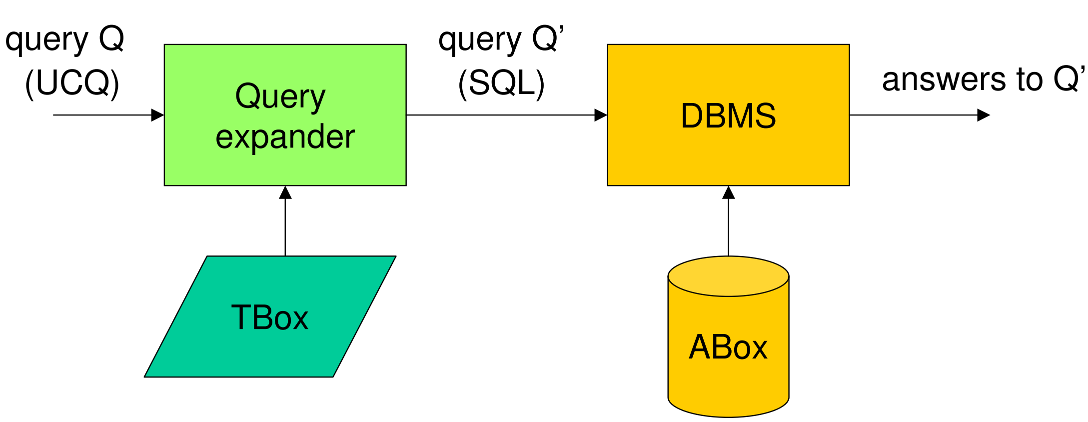
Query answering in DL-Lite

- 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 query 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)

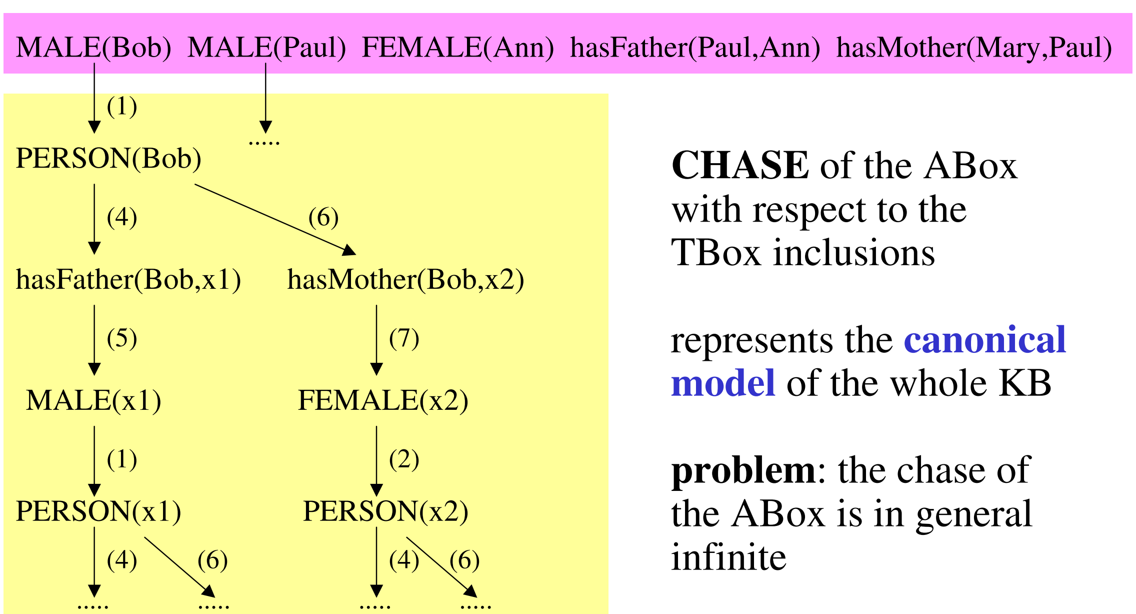
answers to query:
{ Bob, Paul, Ann, Mary }

Query rewriting algorithm for DL-Lite

TBox:

MALE $\sqsubseteq$ PERSON (1)
FEMALE $\sqsubseteq$ PERSON (2)
MALE $\sqsubseteq \neg$FEMALE (3)
PERSON $\sqsubseteq \exists$hasFather (4)
$\exists$hasFather^{-} $\sqsubseteq$ MALE (5)
PERSON $\sqsubseteq \exists$hasMother (6)
$\exists$hasMother^{-} $\sqsubseteq$ FEMALE (7)

Answering queries: chasing the ABox

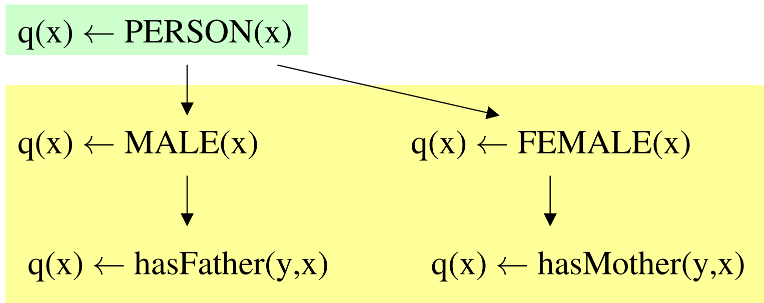


CHASE of the ABox with respect to the TBox inclusions

represents the **canonical model** of the whole KB

problem: the chase of the ABox is in general infinite

Query rewriting algorithm for DL-Lite



CHASE of the query:

- inclusions are applied “from right to left”
- this chase always terminates
- this chase is computed independently of the ABox

Query rewriting algorithm for DL-Lite

the rewriting algorithm iteratively applies two rewriting rules:

- **atom-rewrite**: takes an atom of the conjunctive query and rewrites it applying a TBox inclusion
  - the inclusion is used as a rewriting rule (right-to-left)
- **reduce**: takes two unifiable atoms of the conjunctive query and merges (unifies) them

Query rewriting algorithm for DL-Lite

```
Algorithm PerfectRef (q; T)
Input: conjunctive query q, DL-Lite TBox T
Output: union of conjunctive queries PR
PR := {q};
repeat
    PR0 := PR;
    for each q ∈PR0 do
    (a) for each g in q do
            for each positive inclusion I in T do
              if I is applicable to g then PR := PR ∪{q[g/gr(g,I)]};
    (b) for each g1, g2 in q do
          if g1 and g2 unify then PR := PR ∪{f (reduce(q,g1,g2))}
until PR0 = PR;
return PR
```


Reasoning in DL-Lite

- this query answering technique is in LOGSPACE with respect to data (ABox) complexity
- polynomial technique for deciding KB consistency in DL-Lite
- all main reasoning tasks in DL-Lite can be reduced to either KB consistency or query answering

  => all main reasoning tasks in DL-Lite are tractable

Complexity of concept consistency