Semantic Web

Part 5

The ontology layer 3: OWL 2 profiles and DL-Lite

Riccardo Rosati

Corso di laurea magistrale in Ingegneria Informatica
Sapienza Università di Roma

2012/2013

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 : do not reason over ontologies
- solution 2: limit the expressive power of the ontology language

  => **tractable fragments** of OWL
- solution 3: 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

- The second version of OWL (called OWL2) became a W3C recommendation on October 2009
- Besides the OWL2 Full language and the OWL2 DL language, this recommendation contains three fragments of OWL2 DL called **OWL 2 PROFILES:**
  - **OWL 2 QL** based on the DL **DL-Lite**
  - **OWL 2 EL** based on the DL **EL**
  - **OWL 2 RL** based on the DL **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)

The DL-Lite family

- DL-Lite is a family of Description Logics
- **DL-Lite$_{core}$** = basic DL-Lite language
- main DL-Lite dialects:
  - **DL-Lite$_F$** (DL-Lite$_{core}$ + role functionality)
  - **DL-Lite$_R$** (DL-Lite$_{core}$ + role hierarchies)
  - **DL-Lite$_A$** (DL-Lite$_F$ + DL-Lite$_R$ + attributes + domains)
- the current OWL 2 QL proposal is based on DL-Lite$_R$

DL-Lite$_F$ syntax

concept expressions:

- atomic concept A
- role domain ∃R
- role range $\exists R^-$

role expressions:

- atomic role R
- inverse atomic role R^-

- DL-Lite$_F$ **TBox** = set of
  - concept inclusions
  - concept disjointness assertions
  - functional assertions (stating that a role is functional)
- DL-Lite$_F$ **ABox** = set of ground atoms, i.e., assertions
  - A(a) with A concept name
  - R(a,b) with R role name

Example

TBox:

MALE $\sqsubseteq$ PERSON	concept inclusion
FEMALE $\sqsubseteq$ PERSON	concept inclusion
PERSON $\sqsubseteq$ $\exists$hasFather	concept inclusion
$\exists$hasFather$^-$ $\sqsubseteq$ MALE	concept inclusion
PERSON $\sqsubseteq$ $\exists$hasMother	concept inclusion
$\exists$hasMother$^-$ $\sqsubseteq$ FEMALE	concept inclusion
MALE $\sqsubseteq$ $\neg$FEMALE	concept disjointness
funct(hasMother)	role functionality

ABox:

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

Expressiveness of DL-Lite vs. OWL-DL

main expressive limitations of DL-Lite w.r.t. OWL-DL:

1. **restricted disjunction:**
   - no explicit disjunction
   - binary Horn implications (concept and role inclusions)
2. **restricted negation:**
   - no explicit negation
   - concept (and role) disjointness
3. **restricted existential quantification:**
   - e.g., no qualified existential concepts
4. **limited role cardinality restrictions:**
   - only role functionality allowed
   - not a "real" problem

Expressiveness of DL-Lite vs. RDF/RDFS

DL-Lite captures RDFS...

- RDFS classes = concepts
- RDFS properties = roles
- rdfs:subClassOf = concept inclusion
- rdfs:subPropertyOf = role inclusion
- rdfs:domain = role domain
- rdfs:range = role range

but: DL-Lite does not allow for meta-predicates

DL-Lite extends RDFS:

- “exact” role domain and range
- concept and role disjointness
- inverse roles
- functional roles

DL-Lite vs. conceptual data models

- DL-Lite captures a very large subset of the constructs of conceptual data modeling languages (UML class diagrams, E-R)
- e.g., DL-Lite$_A$ captures almost all the E-R model:
  - entities = concepts
  - binary relationships = roles
  - entity attributes = concept attributes
  - relationship attributes = role attributes
  - cardinality constraints (0,1) = concept inclusions and role functionalities
  - ...

⇒ DL-Lite = a simple yet powerful ontology language

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:

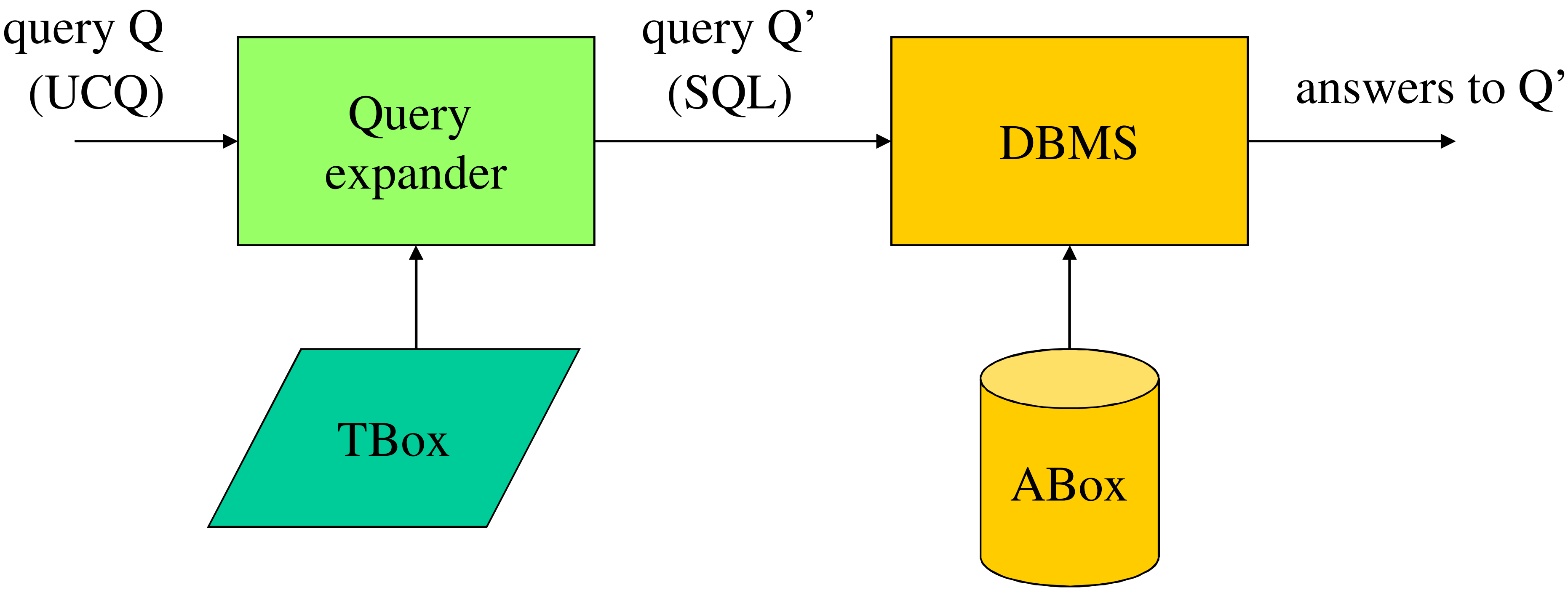
- instance checking and instance retrieval in DL-Lite are solvable in polynomial time
- conjunctive queries over DL-Lite ontologies can be answered in polynomial time (actually in LogSpace) with respect to *data complexity* (i.e., the size of the ABox)

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) ← PERSON(x) ∨
FEMALE(x) ∨
MALE(x) ∨
hasFather(y,x) ∨
hasMother(y,x)

ABox:

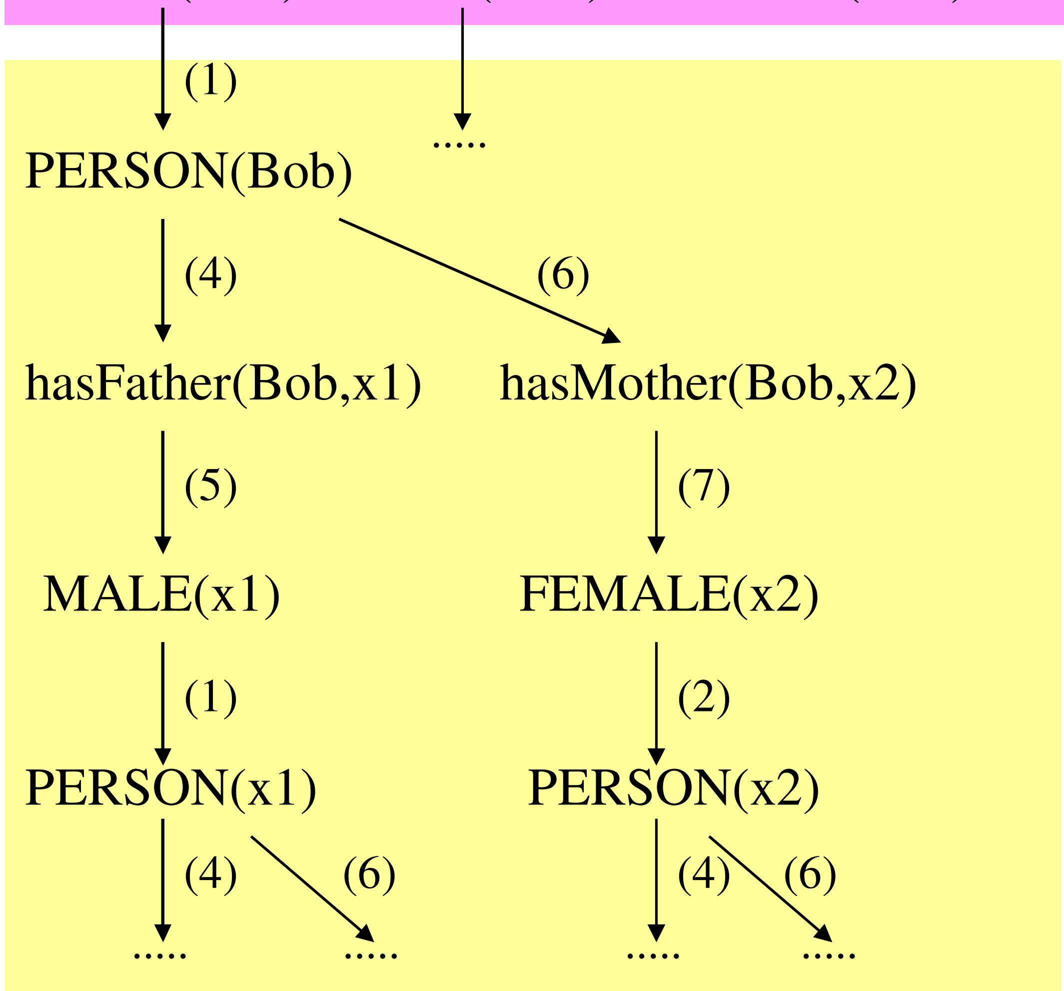
MALE(Bob)
MALE(Paul)
FEMALE(Ann)
hasFather(Paul,Ann)
hasMother(Mary,Paul)

answers to query:

{ Bob, Paul, Ann, Mary }

Answering queries: chasing the ABox

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



CHASE of the ABox with respect to the TBox = adding to the ABox all instance assertions that are logical consequences of the TBox

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

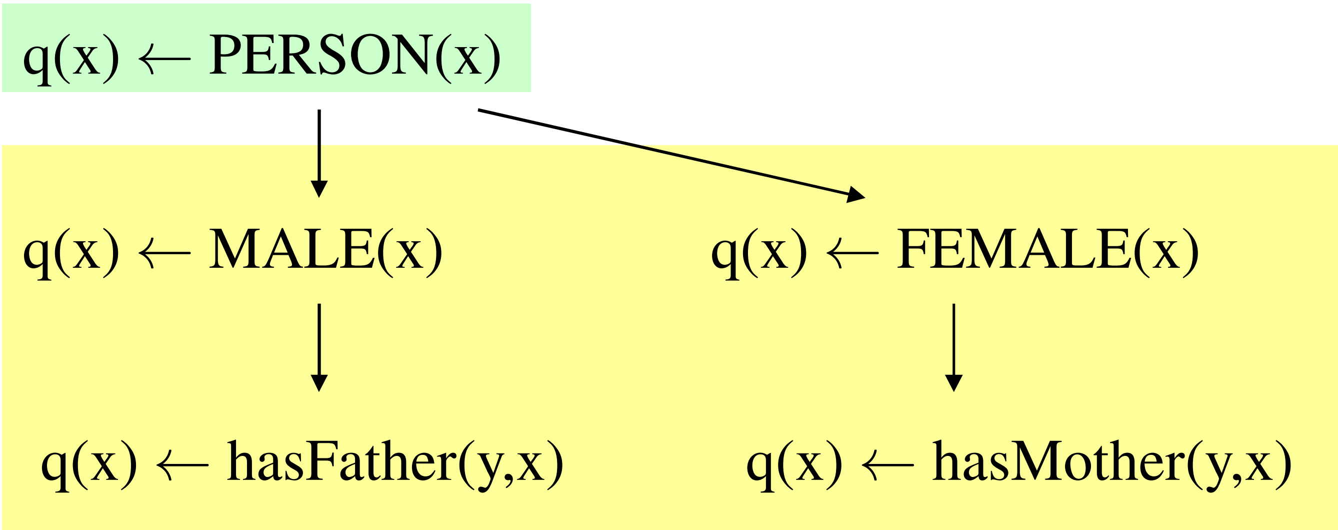
problem: the chase of the ABox is in general infinite

Query rewriting algorithm for DL-Lite

how to avoid the infinite chase of the ABox?

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**
- **reduce**

Atom-rewrite

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)

Example:

- $T = \{ D \sqsubseteq C \}$
- q :- C(x), R(x,y), D(y)
- atom-rewrite(q, C(x), $D \sqsubseteq C$) = q :- D(x), R(x,y), D(y)

Reduce

reduce takes two unifiable atoms of the conjunctive query and merges (unifies) them

Example:

- q :- C(x), R(x,y), R(y,z), D(z)
- reduce(q, R(x,y), R(y,z)) = q:- C(x), R(x,x), D(x)
  (the unification of R(x,y) and R(y,z) implies x=y=z)

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 ∪{atom-rewrite(q,g,I)};
    (b) for each g1, g2 in q  do
          if g1 and g2 unify then PR := PR ∪{reduce(q,g1,g2)}
until PR0 = PR;
return PR
```

Reasoning in DL-Lite

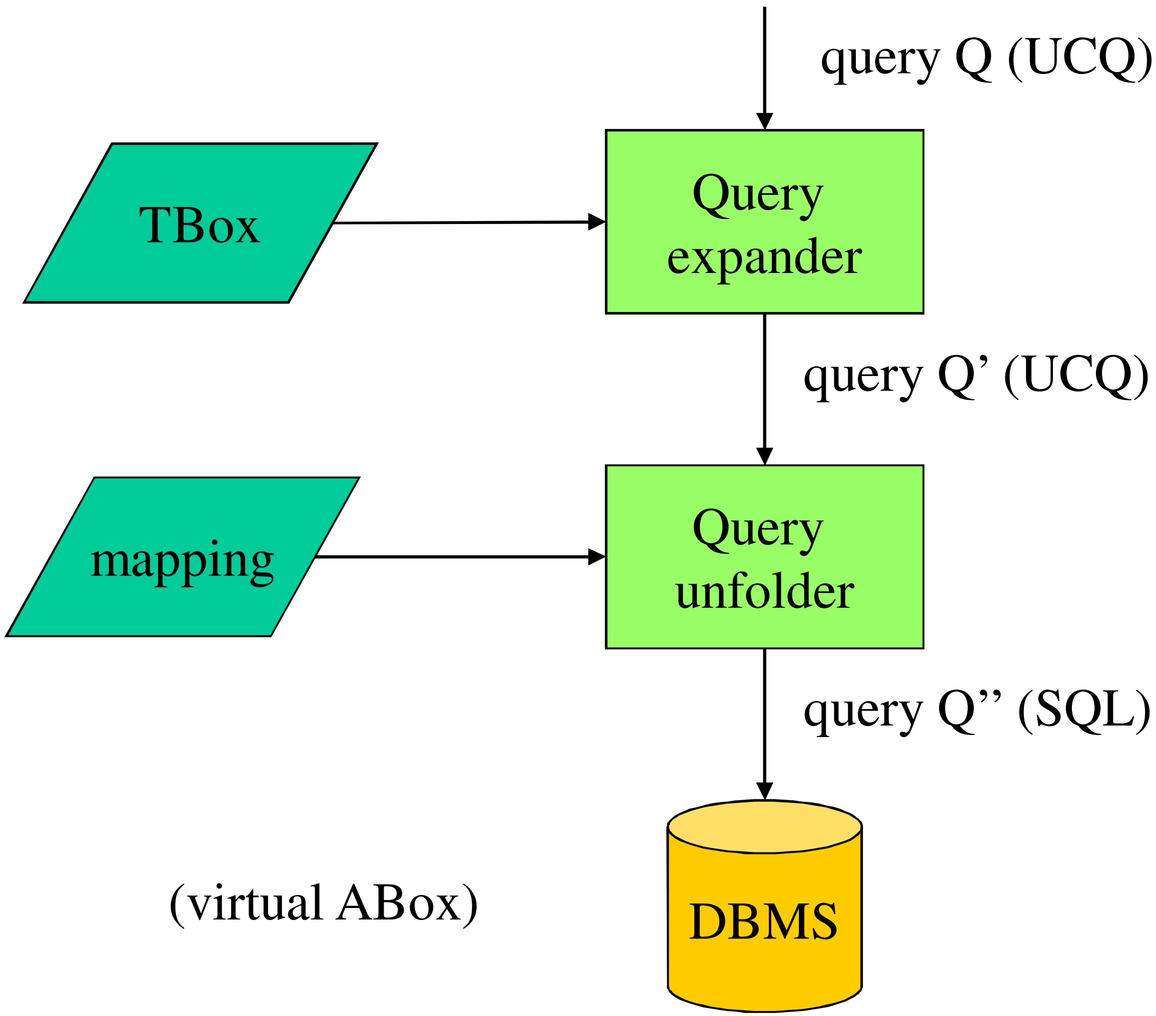
- this query answering technique is in LOGSPACE with respect to data (ABox) complexity
- polynomial technique for deciding KB consistency in DL-Lite
- all main reasoning tasks in DL-Lite can be reduced to either KB consistency or query answering

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

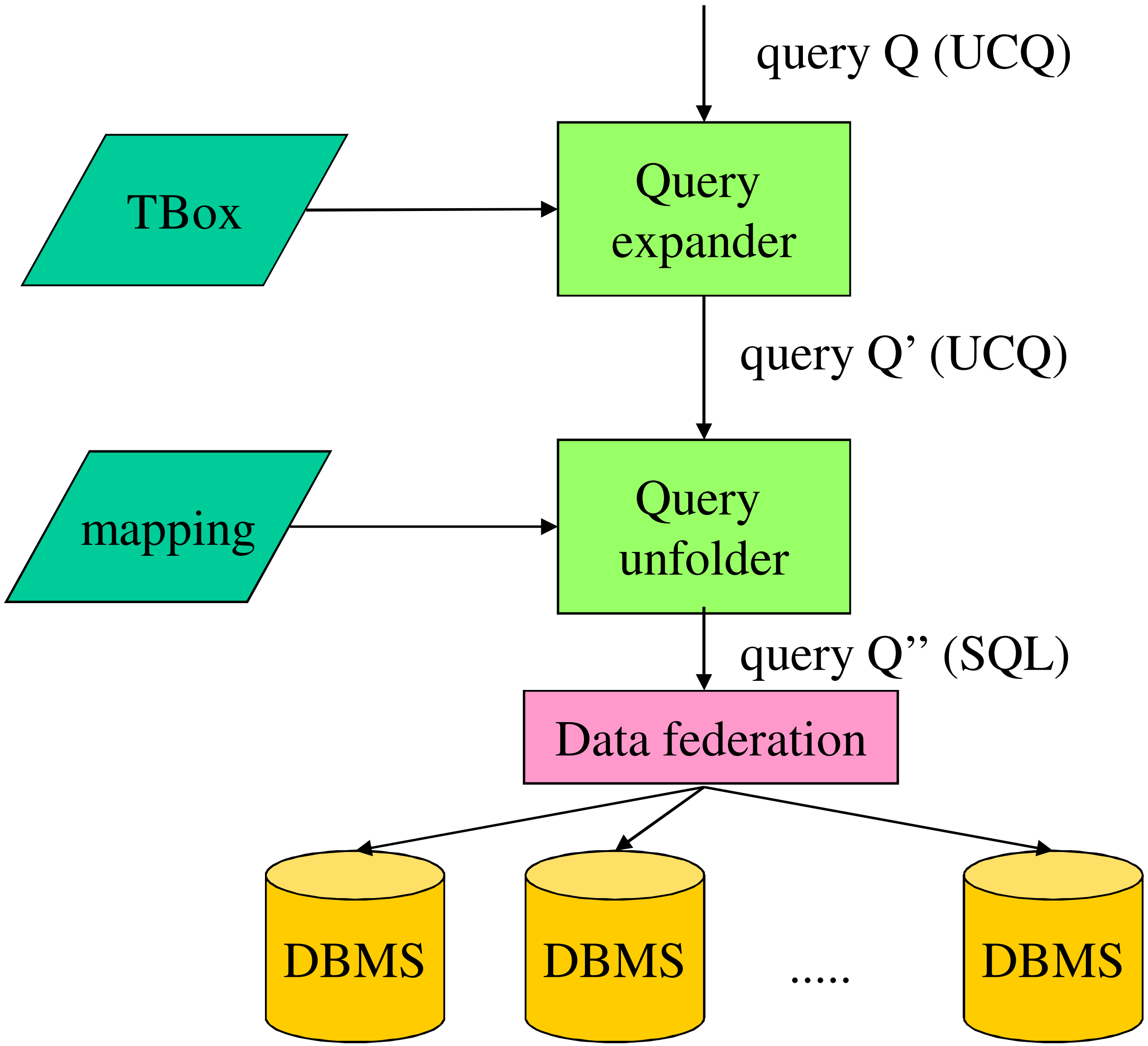
QuOnto

- 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)



DL-Lite: open issues

- scalable reasoning with respect to the TBox
  - handling (very) large TBoxes
- provide support for
  - provenance/justification/explanation
  - annotations/probabilities
- provide full RDFS compliance
  - meta-modeling

References

- OWL 2 profiles:

  `http://www.w3.org/TR/owl2-profiles/`

- DL-Lite papers: see, e.g.,

  `http://www.dis.uniroma1.it/quonto/?q=node/14`

  `http://www.dis.uniroma1.it/~rosati/publications/`

- Quonto system (DL-Lite tool):

  `http://www.dis.uniroma1.it/~quonto/`