

Knowledge Representation and Semantic Technologies

OWL 2

Riccardo Rosati

Corso di Laurea Magistrale in Ingegneria Informatica

Sapienza Università di Roma

2020/2021

Weak sides of OWL 1

- OWL 1 = first release of OWL (2004)
- Three versions of OWL 1:
 - OWL Full: undecidable
 - OWL-DL: reasoning is exponential
 - OWL-Lite: almost same complexity as OWL-DL
- Main criticism: processing OWL is computationally too expensive (exponential)
- especially in Semantic Web applications, scalability (or at least tractability) of processing/reasoning is a crucial property

Limits of 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 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 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: OWL profiles

- 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 (**OWL profiles**)
- 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 **RL**

OWL 2 profiles: Overview

- Each of the three OWL 2 profiles has **tractable** complexity of reasoning: i.e., the main reasoning tasks (ontology consistency, instance checking, etc.) can be computed in polynomial time
- These polynomial reasoning techniques are very different from the general tableau algorithm for Description Logics (ALC)
- In particular, for query answering:
 - OWL 2 QL/DL-Lite: algorithm based on **query rewriting**
 - OWL 2 RL: algorithm based on **ABox materialization**
 - OWL 2 EL: algorithm based on a combination of query rewriting and ABox materialization

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 $\exists R$
- 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:

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

ABox:

$\text{MALE}(\text{Bob}), \text{MALE}(\text{Paul}), \text{FEMALE}(\text{Ann}),$
 $\text{hasFather}(\text{Paul}, \text{Ann}), \text{hasMother}(\text{Mary}, \text{Paul})$

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:

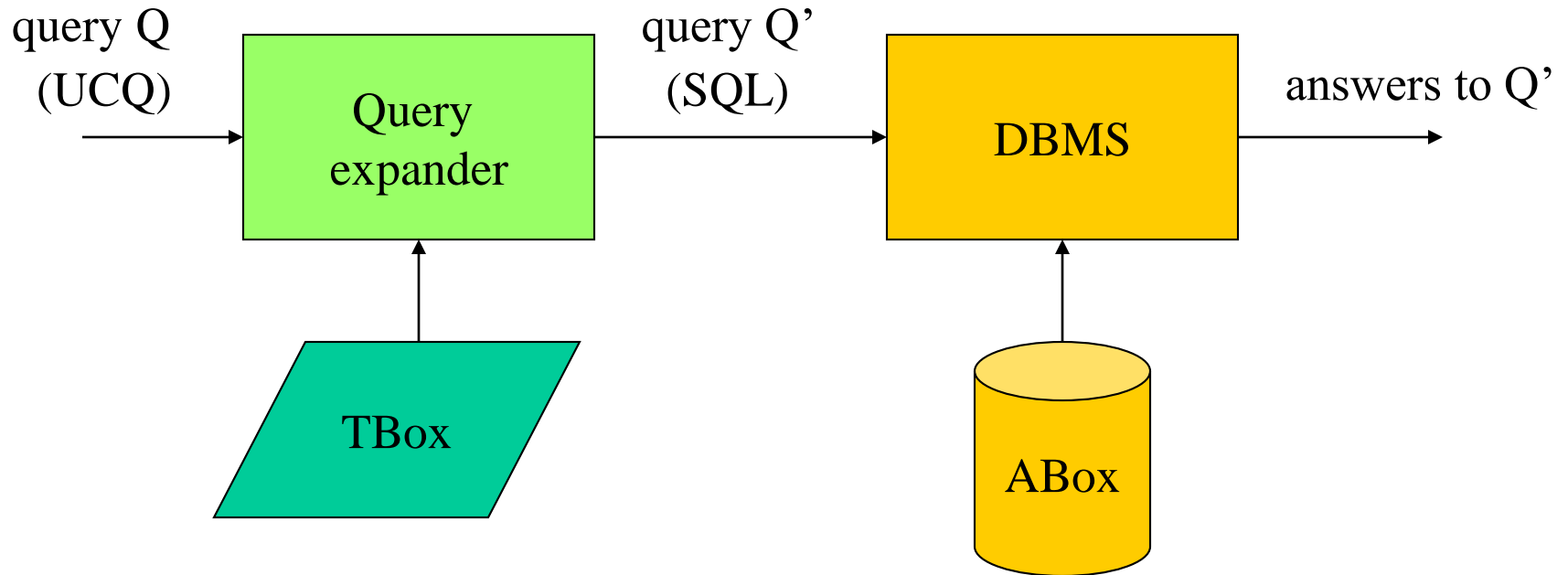
- instance checking and instance retrieval in DL-Lite are solvable in polynomial time
- conjunctive queries over DL-Lite ontologies can be answered in polynomial time (actually in LogSpace) with respect to *data complexity* (i.e., the size of the ABox)

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** (this is the **first-order rewritability** property)
- 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:

$\text{MALE} \sqsubseteq \text{PERSON}$

$\text{MALE} \sqsubseteq \neg \text{FEMALE}$

$\exists \text{hasFather}^- \sqsubseteq \text{MALE}$

$\exists \text{hasMother}^- \sqsubseteq \text{FEMALE}$

$\text{FEMALE} \sqsubseteq \text{PERSON}$

$\text{PERSON} \sqsubseteq \exists \text{hasFather}$

$\text{PERSON} \sqsubseteq \exists \text{hasMother}$

input query:

$q(x) \text{ :- PERSON}(x)$

rewritten query:

$q'(x) \text{ :- PERSON}(x) \vee$
 $\text{FEMALE}(x) \vee$
 $\text{MALE}(x) \vee$
 $\text{hasFather}(y,x) \vee$
 $\text{hasMother}(y,x)$

Example

rewritten query:

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

ABox:

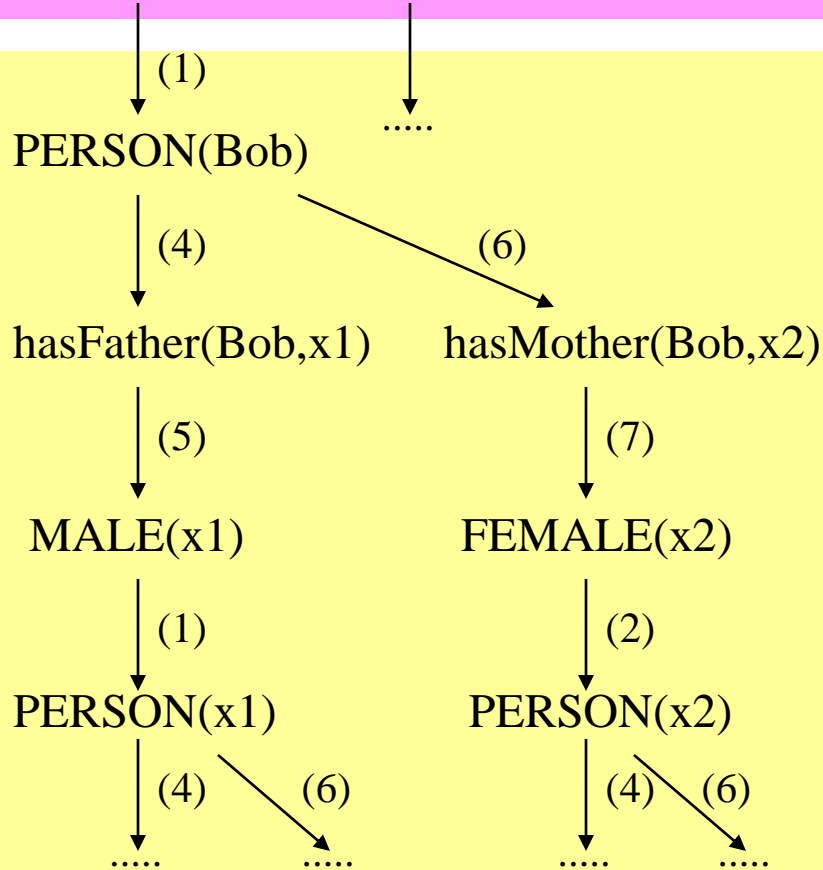
MALE(Bob)
MALE(Paul)
FEMALE(Ann)
hasFather(Ann, Paul)
hasMother(Paul, Mary)

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

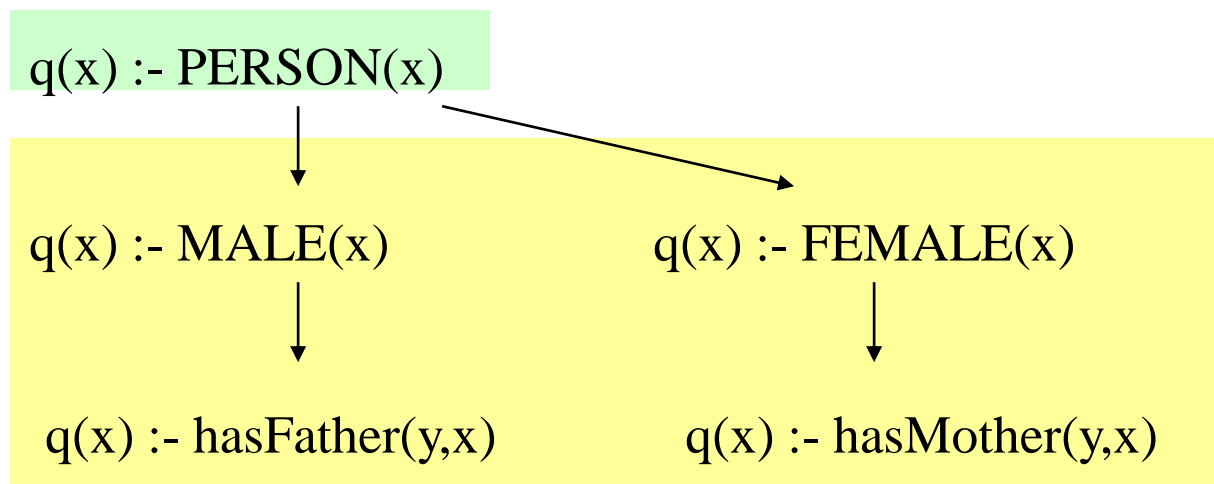
problem: the chase of the ABox is in general infinite

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:

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

Reduce

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

Example:

- $q \text{ :- } C(x), R(x,y), R(y,z), D(z)$
- $\text{reduce}(q, R(x,y), R(y,z)) = q \text{ :- } C(x), R(x,x), D(x)$
(the unification of $R(x,y)$ and $R(y,z)$ implies $x=y=z$)

Query rewriting algorithm for DL-Lite

Algorithm PerfectRef (q, \mathcal{T})

Input: conjunctive query q , DL-Lite TBox \mathcal{T}

Output: union of conjunctive queries PR

PR := { q };

repeat

 PR0 := PR;

 for each $q \in \text{PR0}$ do

 (a) for each g in q do

 for each positive inclusion I in \mathcal{T} do

 if I is applicable to g

 then PR := PR \cup {atom-rewrite(q, g, I)};

 (b) for each $g1, g2$ in q do

 if $g1$ and $g2$ unify then PR := PR \cup {reduce($q, g1, g2$)}

until PR0 = PR;

return PR

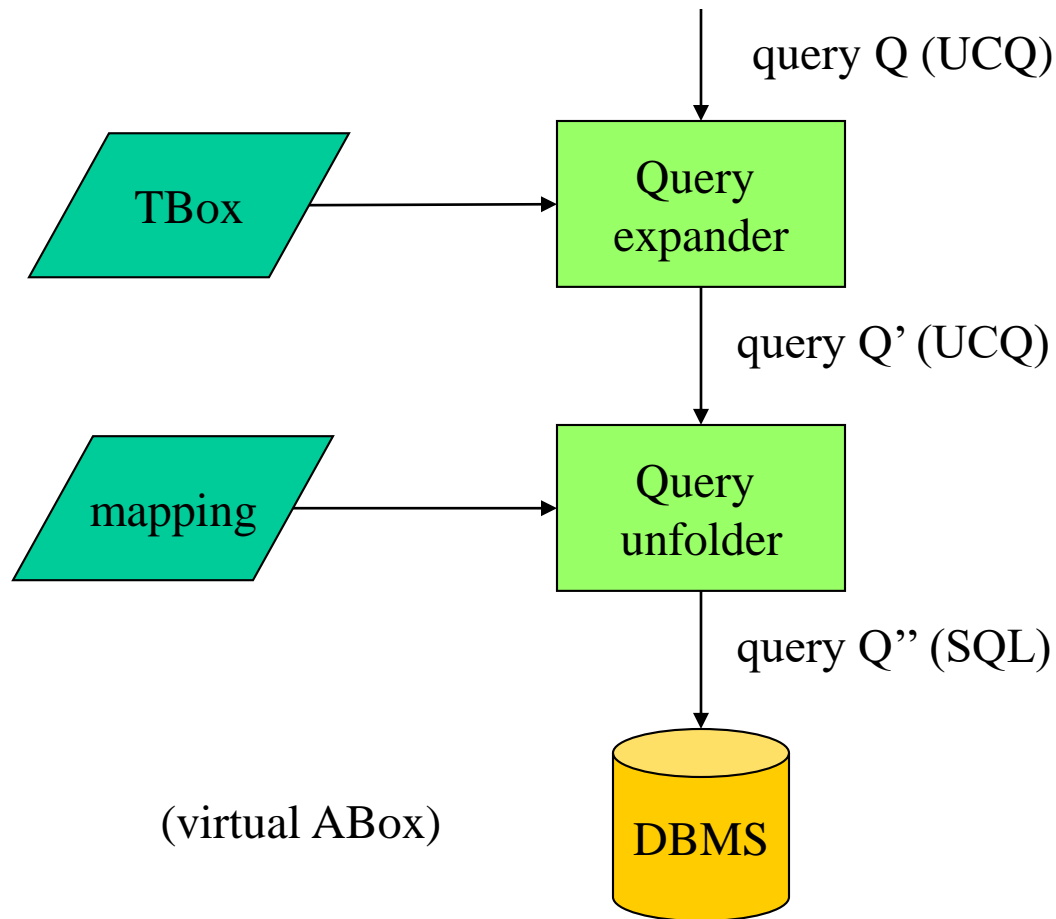
Reasoning in DL-Lite

- 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

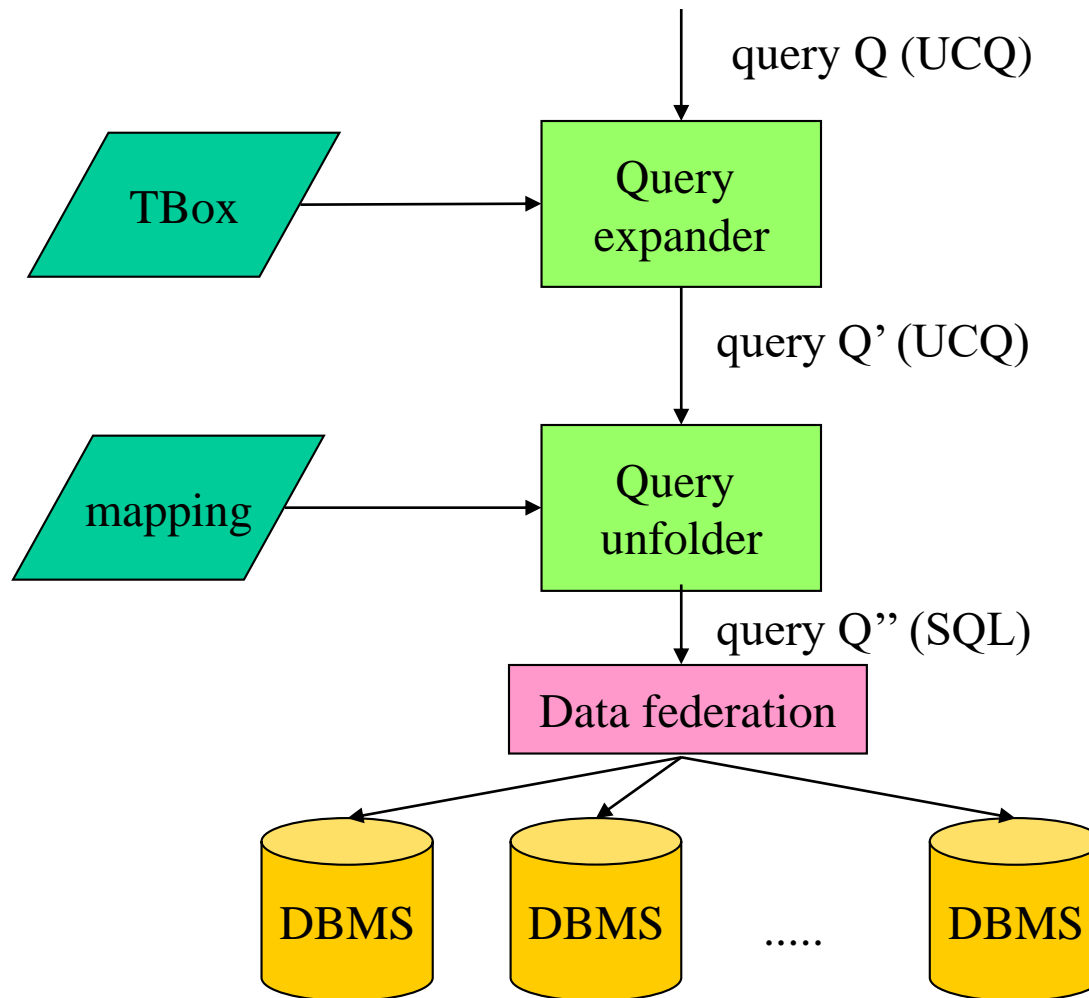
QuOnto

- 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)



The Description Logic RL: Syntax

concept expressions:

- atomic concept A
- concept conjunction $C_1 \sqcap C_2$
- qualified existential $\exists R.C$
- qualified existential $\exists R.\perp$

role expressions:

- atomic role R
- inverse role R^-

- RL **TBox** =
 - set of concept inclusions of the form $C \sqsubseteq A$ or $C \sqsubseteq \perp$
 - set of role inclusions $R_1 \sqsubseteq R_2$
- RL **ABox** = set of ground atoms, i.e., assertions
 - $A(a)$ with A concept name
 - $R(a,b)$ with R role name

RL ontology: Example

TBox:

$\text{MALE} \sqsubseteq \text{PERSON}$

$\text{FEMALE} \sqsubseteq \text{PERSON}$

$\text{hasMother} \sqsubseteq \text{hasParent}$

$\text{hasFather} \sqsubseteq \text{hasParent}$

$\text{MALE} \sqcap \text{FEMALE} \sqsubseteq \perp$

$\text{STUDENT} \sqcap \text{EMPLOYEE} \sqsubseteq \text{WORKING-STUDENT}$

$\exists \text{hasParent.HAPPY} \sqsubseteq \text{HAPPY}$

ABox:

$\text{MALE}(\text{Bob}), \text{MALE}(\text{Paul}), \text{FEMALE}(\text{Ann}),$

$\text{hasFather}(\text{Paul}, \text{Ann}), \text{hasMother}(\text{Mary}, \text{Paul}),$

$\text{HAPPY}(\text{Ann}), \text{EMPLOYEE}(\text{Paul}), \text{STUDENT}(\text{Paul})$

Computational properties of RL

Complexity of reasoning in RL:

- Intensional (TBox) reasoning is PTIME-complete (i.e., tractable)
- Instance checking is PTIME-complete
- Conjunctive query answering is PTIME-complete with respect to data complexity
 - This implies that first-order rewritability does NOT hold for RL
- Conjunctive query answering is NP-complete with respect to combined complexity
- Reasoning in RL can be reduced to reasoning in positive Datalog

Reasoning in RL (and RDFS)

ABox reasoning and query answering in RL (and RDFS) can be done through **forward chaining** (a.k.a. **materialization**), which corresponds to the **chase** procedure mentioned above.

- Chase of the ABox with respect to the TBox = adding to the ABox all instance assertions that are logical consequences of the TBox
- In the case of RL (and RDFS) no new individual is introduced by the chase, so this procedure always terminates (and requires polynomial time)
- After this materialization step, the TBox can be discarded and conjunctive queries can be answered by evaluating them on the materialized ABox

Reasoning in RL: Example

TBox:

MALE \sqsubseteq PERSON
FEMALE \sqsubseteq PERSON
hasMother \sqsubseteq hasParent
hasFather \sqsubseteq hasParent
MALE \sqcap FEMALE $\sqsubseteq \perp$
STUDENT \sqcap EMPLOYEE \sqsubseteq
 WORKING-STUDENT
 \exists hasParent.HAPPY \sqsubseteq HAPPY

ABox:

MALE(Bob), MALE(Paul),
FEMALE(Ann),
hasFather(Paul,Ann),
hasMother(Mary,Paul),
HAPPY(Ann), EMPLOYEE(Paul),
STUDENT(Paul)

Materialization

TBox:

MALE \sqsubseteq PERSON
FEMALE \sqsubseteq PERSON
hasMother \sqsubseteq hasParent
hasFather \sqsubseteq hasParent
MALE \sqcap FEMALE $\sqsubseteq \perp$
STUDENT \sqcap EMPLOYEE \sqsubseteq
 WORKING-STUDENT
 \exists hasParent.HAPPY \sqsubseteq HAPPY

Materialized ABox (chase):

MALE(Bob), MALE(Paul),
FEMALE(Ann),
hasFather(Paul,Ann),
hasMother(Mary,Paul),
HAPPY(Ann), EMPLOYEE(Paul),
STUDENT(Paul),

PERSON(Bob), PERSON(Paul),
PERSON(Ann),
hasParent(Paul,Ann),
hasParent(Mary,Paul),
HAPPY(Paul), HAPPY(Mary),
WORKING-STUDENT(Paul)

Query answering

TBox:

MALE \sqsubseteq PERSON
FEMALE \sqsubseteq PERSON
hasMother \sqsubseteq hasParent
hasFather \sqsubseteq hasParent
MALE \sqcap FEMALE $\sqsubseteq \perp$
STUDENT \sqcap EMPLOYEE \sqsubseteq
 WORKING-STUDENT
 \exists hasParent.HAPPY \sqsubseteq HAPPY

Query: (happy grandchildren)

$q(x) :-$ HAPPY(x), hasParent(x,y),
 hasParent(y,z).

Answer = { Mary }

Materialized ABox:

MALE(Bob), MALE(Paul),
FEMALE(Ann),
hasFather(Paul,Ann),
hasMother(Mary,Paul),
HAPPY(Ann), EMPLOYEE(Paul),
STUDENT(Paul),

PERSON(Bob), PERSON(Paul),
PERSON(Ann),
hasParent(Paul,Ann),
hasParent(Mary,Paul),
HAPPY(Paul), HAPPY(Mary),
WORKING-STUDENT(Paul)

The EL family of DLs

- The EL family of description logics underlies the OWL 2 EL profile
- Several members:
 - EL (core language)
 - EL_{\perp}
 - ELH
 - EL_{++}
 - ...

Syntax of EL

concept expressions:

- atomic concept A
- concept conjunction $C_1 \sqcap C_2$
- qualified existential $\exists R.C$

role expressions:

- atomic role R

- EL **TBox** = set of concept inclusions
- EL **ABox** = set of ground atoms, i.e., assertions
 - $A(a)$ with A concept name
 - $R(a,b)$ with R role name

EL ontology: Example

TBox:

MALE \sqsubseteq PERSON

FEMALE \sqsubseteq PERSON

PERSON $\sqsubseteq \exists \text{hasFather.MALE}$

PERSON $\sqsubseteq \exists \text{hasMother.FEMALE}$

STUDENT \sqcap EMPLOYEE \sqsubseteq WORKING-STUDENT

ABox:

MALE(Bob), MALE(Paul), FEMALE(Ann),

hasFather(Paul,Ann), hasMother(Mary,Paul),

HAPPY(Ann), EMPLOYEE(Paul), STUDENT(Paul)

Computational properties of EL

Complexity of reasoning in EL (and in other languages of this family):

- Intensional (TBox) reasoning is PTIME-complete (i.e., tractable)
- Instance checking is PTIME-complete
- Conjunctive query answering is PTIME-complete with respect to data complexity
 - This implies that first-order rewritability does NOT hold for EL
- Conjunctive query answering is NP-complete with respect to combined complexity

Reasoning in EL

- The chase for an EL ontology may be infinite (for instance, the EL ontology of the previous example has an infinite chase), so ABox materialization is not applicable to EL
- On the other hand, first-order rewritability does not hold for EL, so the query rewriting approach (in SQL) does not work too
- One possible solution is a **hybrid approach** that combines ABox materialization and query rewriting: the ABox is **partially** expanded with respect to the TBox and the query is **partially** rewritten with respect to the TBox (see [Kontchakov et al., 2011])

Reasoning in OWL 2: Summary

Reasoning technique	OWL 2 QL	OWL 2 RL	OWL 2 EL	OWL 2 DL
ABox materialization (forward chaining)	NO	YES	NO (*)	NO
Query rewriting (backward chaining)	YES	NO	NO (*)	NO
Tableau method	YES	YES	YES	YES

(*) A combination of partial ABox materialization and partial query rewriting can be applied to EL ontologies (see [Kontchakov et al., 2011])

References

- OWL W3C Web site:
<http://www.w3.org/2004/OWL/>
- OWL 2 overview:
<http://www.w3.org/TR/owl2-overview/>
- OWL 2 primer:
<http://www.w3.org/TR/2012/REC-owl2-primer-20121211/>
- OWL 2 profiles:
<http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/>
- OWL 2 quick reference guide:
<https://www.w3.org/TR/owl2-quick-reference/>

References

- **SPARQL 1.1:**
<http://www.w3.org/TR/sparql11-query/>
- **SPARQL 1.1 entailment regimes:**
<http://www.w3.org/TR/2013/REC-sparql11-entailment-20130321/>
- **Web page on Description Logic reasoners:**
<http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/>
- **Protege (OWL ontology editor):**
<http://protege.stanford.edu/>

References

- Hermit (OWL reasoning tool):
<http://hermit-reasoner.com/>
- Konclude (OWL 2 DL reasoner):
<http://derivo.de/produkte/konclude/>
- ELK (OWL 2 EL ontology reasoner):
<http://www.cs.ox.ac.uk/isg/tools/ELK/>
- Stardog (OWL 2 profiles and OWL2 DL reasoner):
<http://stardog.com/>
- RacerPro (OWL reasoning tool):
<http://franz.com/agraph/racer/>
- Mastro (DL-Lite ontology-based data access system)
<http://www.dis.uniroma1.it/~mastro/>

References

- Paper describing the combined approach to query answering in EL:
Carsten Lutz, David Toman, Frank Wolter: Conjunctive Query Answering in the Description Logic EL Using a Relational Database System. IJCAI 2009: 2070-2075