SQL, DLs, Datalog, and ASP: comparison

Riccardo Rosati

Knowledge Representation and Semantic Technologies

Corso di Laurea in Ingegneria informatica
Sapienza Università di Roma
2015/2016

CWA vs. OWA

- DLs are based on the semantics of classical logic
- Open-world assumption (OWA):
  - The knowledge expressed by the theory is not complete
  - Many possible worlds (models)
- Databases are based on a closed-world assumption (CWA):
  - The knowledge expressed by the database is complete
  - Only one world is possible (the one completely described by the database)
- Datalog and ASP inherit CWA from databases
- But: ASP is able to deal with incomplete knowledge (disjunction)

Complete vs. incomplete knowledge

- DL ABoxes are **incomplete specifications** of the domain of interest
- TBox can be used to infer implicit facts about the world
  - E.g., from

    ABox = { student(Bob) } and

    TBox = { subClassOf(student, person) }

    we conclude person(Bob) (which is NOT stated in the ABox)
  - The knowledge expressed by the database is complete
  - Only one world is possible (the one completely described by the database)

Complete vs. incomplete knowledge

- In relational DBs, the database instance is a **complete specification**:
  - Schema constraints in databases are **integrity constraints**: they MUST be satisfied by the database instance (otherwise the database is in an illegal state)
  - For this reason, such constraints cannot derive new facts (not appearing in the database instance)

Recursion

- DLs are able to deal with recursive statements
  - E.g., cyclic definitions of concepts in the TBox
- DLs do not allow for recursive queries
- SQL does not allow for recursive queries
  - only a limited form of recursion in views is allowed
- Datalog allows for expressing recursive queries over databases
- ASP allows for expressing recursive queries over databases

Negation

- Several DLs allow for expressing negation in the TBox (and in the ABox) and in queries
- But, DLs do not allow for recursive negation in queries
- SQL allows for negation in queries, but does not allow for recursive negation
- Datalog does not allow for expressing negation
- ASP allows for expressing negation, and even recursive negation

Value invention

- DLs allow for expressing so-called value invention (or mandatory participation to roles)
- Example: inclusion axiom $\text{Person} \subseteq \exists \text{hasMother} \,.\, \top$
- The above axiom cannot be represented in Datalog
- E.g., the rule

  hasMother(x,y) :- Person(x).

  is not range-restricted (so it is not allowed in Datalog)
- Datalog does not allow for talking about elements outside the Herbrand Universe of the program (namely, the constants appearing in the program)
- Same restriction holds for ASP (and SQL too)
- Value invention is a form of incomplete knowledge

Existential rules

- To overcome the above limitation, an extension of Datalog called **existential rules**, or **Datalog +/-**, has been proposed
- Existential rules do not have the range restriction on variables: a variable can appear in the head of the rule
- Such variables are interpreted as existentially quantified variables
- E.g.:

  hasMother(x,y) :- Person(x).

  is an existential rule, whose meaning is: if x is a person, then **there exists** y such that x has mother y
- In this way, value invention can be expressed by rules

Existential rules

- Extending Datalog with existential variables in the head makes the language more expressive
- On the other hand, such an extension makes reasoning harder: in general, reasoning with existential rules is **undecidable**
- Recent studies have defined restricted classes of existential rules in which reasoning is decidable (and even tractable in data complexity)
- Interesting relationship between some of these classes and Description Logics

Comparison: expressiveness

	Semantics	Recursion	NOT (stratif.)	NOT (recursive)	OR	Value invention
SQL	CWA	no	yes	no	yes	no
DL	OWA	yes/no	yes	no	yes	yes
Datalog	CWA	yes	no	no	no	no
Datalog + stratif. negation	CWA	yes	yes	no	no	no
ASP	CWA	yes	yes	yes	yes	no

Complexity of SQL

- Complexity of evaluating queries in SQL is NP-complete
- A query may have an exponential number of answers
- Example:

table T:

A
0
1

query:

```
select R1.A, R2.A, R3.A,…,Rn.A
from T AS R1, T AS R2, T AS R3, …, T AS Rn
```

answers:
(2^n tuples)

R1.A	R2.A	R3.A	…	Rn-1.A	Rn.A
0	0	0	…	0	0
0	0	0	…	0	1
…	…	…	…	…	…
1	1	1	…	1	0
1	1	1	…	1	1

Complexity of SQL (cont.)

Previous query SQL:
select R1.A, R2.A, R3.A,…,Rn.A
from T AS R1, T AS R2, T AS R3, …, T AS Rn

Same query expressed in Datalog:
Q(X1, X2, …,Xn):- T(x1), T(x2), …, T(Xn).

(which shows that Datalog can build IDB predicates with an exponential number of tuples in the minimal model)

Comparison: complexity

	Reasoning task	Complexity
SQL	Query evaluation	NP-complete
ALC, unfoldable TBoxes	Concept consistency, KB satisfiability, instance checking	PSPACE-complete
ALC, GCIs	Concept consistency, KB satisfiability, instance checking	EXPTIME-complete

Comparison: complexity

	Reasoning task	Complexity
Ground positive Datalog	Building the minimal model	PTIME-complete
Ground Datalog with stratified negation	Building the minimal model	PTIME-complete
Ground ASP (no disjunction)	Consistency, brave reasoning	NP-complete
Ground ASP	Consistency, brave reasoning	NP^{NP}-complete

Comparison: complexity

	Reasoning task	Complexity
Positive Datalog	Building the minimal model	EXPTIME-complete
Datalog with stratified negation	Building the minimal model	EXPTIME-complete
ASP (no disjunction)	Consistency, brave reasoning	NEXPTIME-complete
ASP	Consistency, brave reasoning	$\text{NEXPTIME}^{\text{NP}}$-complete