Knowledge Representation and Reasoning

Jef Wijsen

UMONS

February 6, 2025

Course Content

$$\text{AI} \simeq \text{GOFAI} + \text{AML}$$

- AI: Artificial Intelligence
- GOFAI: Good Old-Fashioned Artificial Intelligence ($\simeq$ symbolic AI)
- AML: Adaptive Machine Learning (reinforcement learning, big data...)

$$\text{KRR} \subseteq \text{GOFAI}$$

- KRR: Knowledge Representation and [Automated] Reasoning

Course Content

Logic-based KRR

- Answer Set Programming (ASP): an expressive logic for specifying and solving problems in NP (including NP-complete problems).

  Task: Solving
  Input: A set Σ of logic formulas (also called constraints, rules...).
  (e.g., the rules of Sudoku + a partially filled grid)
  Question: Is there a solution (also called model, answer set...) that satisfies every formula in Σ?

- Web Ontology Language (OWL): less expressive logics that allow automated reasoning about data on the Web.

  Task: Automated reasoning
  Input: A set of logic formulas Σ; a logic formula σ.
  Question: Is σ a logical consequence of Σ?

Note: Automated reasoning is computationally impossible for expressive logics.

Data Complexity

- The data complexity of a query $\{x_1, x_2, \ldots, x_n \mid q(x_1, x_2, \ldots, x_n)\}$ is the complexity of the following problem:

  INPUT: A database instance I; constants $c_1, c_2, \ldots, c_n$.
  QUESTION: Does I satisfy $q(c_1, c_2, \ldots, c_n)$?

- In many contexts, when we talk about a query language (e.g., Datalog), we implicitly refer to the set of all queries expressible in that language.
- For example, we say, "Datalog is in P (for data complexity)", meaning that every query expressible in Datalog has data complexity in P.

Recall from Bases de Données I and II

For data complexity, the following inclusions hold true:

$$\text{FO} \subsetneq \text{Datalog}^{\neg} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{Prolog}$$

where

- FO denotes the class of problems that take as input a relational database instance and can be solved by a query in relational calculus; and
- $\text{Datalog}^{\neg}$ denotes the class of problems that take as input a relational database instance and can be solved by a program in Datalog with stratified negation.

Recall:

- NP-complete problems cannot be programmed in $\text{Datalog}^{\neg}$.
- Automated reasoning is already computationally impossible for FO.

Query Complexity

- The query complexity of a query language $\mathcal{L}$ is the complexity of the following problem, relative to a fixed database instance I:

  INPUT: A query $q(x_1, x_2, \ldots, x_n)$ in $\mathcal{L}$; constants $c_1, c_2, \ldots, c_n$.
  QUESTION: Does I satisfy $q(c_1, c_2, \ldots, c_n)$?

Recall:

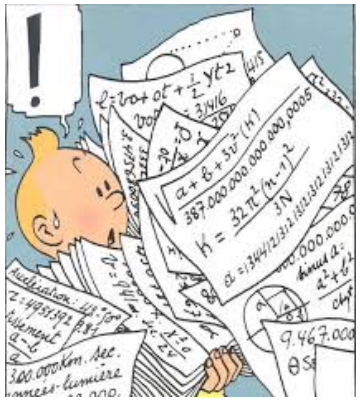
- The query complexity of the class of conjunctive queries is already NP-complete.

Datalog¬

See “A Datalog Primer.”

`https://web.umons.ac.be/app/uploads/sites/84/2024/06/primerDatalog.pdf`

Course Methodology

	Classical course	$\rightsquigarrow$	This course
language	French	$\rightsquigarrow$	English
teacher's role	teaching	$\rightsquigarrow$	guiding
students' role	   being taught	$\rightsquigarrow$	scientific discovery tour
evaluation	exam	$\rightsquigarrow$	project + homeworks + written exam

Course Schedule

Just a screenshot (the full schedule is online):

This document may be updated during the course.

Wednesday, Feb. 5 (15H45)	Meeting in room B4.233 + organization (14')
Thursday, Feb. 6 (15H45)	motivation (72')
Tuesday, Feb. 11 (15H45)	introduction (170')
Wednesday, Feb. 12 (16H15)	Meeting in B4.233; start Homework 1 (due on Feb. 24)
Wednesday, Feb. 19 (15H45)	Meeting in P3E11; start Homework 2 (due on Mar. 3)
Thursday, Feb. 20 (15H45)	modeling (106')
Wednesday, Feb. 26 (15H45)	Meeting in B4.233; discuss Homework 1
Thursday, Feb. 27 (15H45)	language (128')
Tuesday, Mar. 4 (15H45)	
Wednesday, Mar. 5 (15H45)	Meeting in B4.233; discuss Homework 2, start Homework 3 (due on Mar. 23)
Wednesday, Mar. 12 (15H45)	Meeting in B4.233; start Project work
Thursday, Mar. 13 (15H45)	grounding (119')

Datalog¬ by Example

```
red(a,b). red(b,c). red(c,a).
blue(a,c). blue(c,d). blue(d,a).
redTrans(X,Y) :- red(X,Y).
redTrans(X,Z) :- redTrans(X,Y), red(Y,Z).
blueMonopoly(X,Y) :- blue(X,Y), not redTrans(X,Y).
```

- `redTrans` and `blueMonopoly` are IDB predicates (because they occur in rule heads); the other predicates are EDB predicates (= stored database relations).
- The PDG (Program Dependence Graph) has a (non-negated) edge from `redTrans` to `redTrans`, and a negated edge from `blueMonopoly` to `redTrans`.
- Stratified semantics: execute the rules for `redTrans` until no more `redTrans`-facts can be derived; only then can rules with "`not redTrans`" be evaluated.

ASP by Example

```
person(john).
happy(X) :- person(X), not unhappy(X).
unhappy(X) :- person(X), not happy(X).
```

Not stratified: the 1st rule should be executed before the 2nd rule (because of "`not happy`"), but the 2nd should be executed before the 1st (because of "`not unhappy`").

An ASP solver will find two models:

```
clingo version 4.5.4
Solving...
Answer: 1
person(john) happy(john)
Answer: 2
person(john) unhappy(john)
SATISFIABLE

Models       : 2
```