Knowledge Representation and Reasoning

Jef Wijsen

UMONS

February 3, 2026

Table of Contents

1 Content and Methodology of the Course

2 Illustration of Automated Reasoning

3 Illustration of Solving

4 Recalls from Bases de Données I & II

KRR in AI

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

- 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

Logic-Based KRR

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

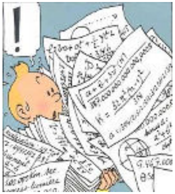
Task:	Automated reasoning
Input:	A set of logic formulas Σ; a logic formula σ.
Question:	Is σ a logical consequence of Σ?

  Since automated reasoning is computationally impossible for full first-order logic, one uses less expressive logics (a.k.a. Description Logics).

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

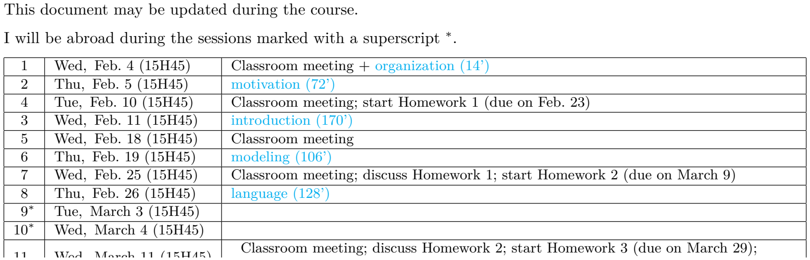
Task:	Solving
Input:	A set Σ of logic formulas (also called constraints, rules...). (e.g., the rules of Sudoku + a partially filled grid)
Question:	Is there a solution (also called model, answer set...) that satisfies every formula in Σ?

Course Methodology

	Classical course		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.

I will be abroad during the sessions marked with a superscript *.

1	Wed, Feb. 4 (15H45)	Classroom meeting + organization (14')
2	Thu, Feb. 5 (15H45)	motivation (72')
4	Tue, Feb. 10 (15H45)	Classroom meeting; start Homework 1 (due on Feb. 23)
3	Wed, Feb. 11 (15H45)	introduction (170')
5	Wed, Feb. 18 (15H45)	Classroom meeting
6	Thu, Feb. 19 (15H45)	modeling (106')
7	Wed, Feb. 25 (15H45)	Classroom meeting; discuss Homework 1; start Homework 2 (due on March 9)
8	Thu, Feb. 26 (15H45)	language (128')
9^*	Tue, March 3 (15H45)	
10^*	Wed, March 4 (15H45)	
11	Wed, March 11 (15H45)	Classroom meeting; discuss Homework 2; start Homework 3 (due on March 29);

Table of Contents

1 Content and Methodology of the Course

2 Illustration of Automated Reasoning

3 Illustration of Solving

4 Recalls from Bases de Données I & II

Automated Reasoning with OWL / Description Logic

Knowledge Base

A Description Logic (DL) knowledge base is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where

- $\mathcal{T}$ is the TBox: terminological axioms
- $\mathcal{A}$ is the ABox: assertions about individuals

TBox:

$$\begin{array}{rcl} \mathsf{Man} & \sqsubseteq & \mathsf{Person} \\ \mathsf{Woman} & \sqsubseteq & \mathsf{Person} \\ \mathsf{Man} \sqcap \mathsf{Woman} & \sqsubseteq & \bot \\ \mathsf{Parent} & \equiv & \mathsf{Person} \sqcap \exists\, \mathsf{hasChild}.\mathsf{Person} \\ \mathsf{Father} & \equiv & \mathsf{Man} \sqcap \mathsf{Parent} \\ \mathsf{Mother} & \equiv & \mathsf{Woman} \sqcap \mathsf{Parent} \end{array}$$

ABox:

$$\begin{array}{l} \mathsf{Man}(\mathsf{john}) \\ \mathsf{Woman}(\mathsf{mary}) \\ \mathsf{hasChild}(\mathsf{john}, \mathsf{mary}) \end{array}$$

Translation in first-order logic

$$\forall x\,(\mathrm{Man}(x) \rightarrow \mathrm{Person}(x))$$
$$\forall x\,(\mathrm{Woman}(x) \rightarrow \mathrm{Person}(x))$$
$$\forall x\,\neg(\mathrm{Man}(x) \wedge \mathrm{Woman}(x))$$
$$\forall x\,\Big(\mathrm{Parent}(x) \leftrightarrow (\mathrm{Person}(x) \wedge \exists y\,(\mathrm{hasChild}(x,y) \wedge \mathrm{Person}(y)))\Big)$$
$$\forall x\,(\mathrm{Father}(x) \leftrightarrow (\mathrm{Man}(x) \wedge \mathrm{Parent}(x)))$$
$$\forall x\,(\mathrm{Mother}(x) \leftrightarrow (\mathrm{Woman}(x) \wedge \mathrm{Parent}(x)))$$

$$\mathrm{Man}(\mathrm{john})$$
$$\mathrm{Woman}(\mathrm{mary})$$
$$\mathrm{hasChild}(\mathrm{john}, \mathrm{mary})$$

We only need two variables, x and y. It is known that automated reasoning is possible in first-order logic restricted to two variables. This two-variable fragment forms the backbone of many Description Logics.

Reasoning Tasks

Automated Reasoning

Given a knowledge base $\mathcal{K}$ and an assertion α, decide whether

$$\mathcal{K} \models \alpha.$$

For example,

- **Consistency checking:** Does $\mathcal{K} \not\models \bot$?
- **Instance checking:** Does $\mathcal{K} \models \text{Father(john)}$?
- **Classification:** Does $\mathcal{K} \models \text{Father} \sqsubseteq \text{Parent}$?
- **Query answering:** Find all individuals x s.t. $\mathcal{K} \models \text{Parent}(x)$

Table of Contents

1 Content and Methodology of the Course

2 Illustration of Automated Reasoning

3 Illustration of Solving

4 Recalls from Bases de Données I & II

Solving Problems in NP

We are given a finite domain of vertices and a binary relation E of edges.

The following formula asks whether there exist three unary relations (i.e., three sets) A, B, C satisfying a first-order condition:

$$\exists A \exists B \exists C \left(\begin{array}{l} \forall x \left[\begin{array}{r} (A(x) \wedge \neg B(x) \wedge \neg C(x)) \\ \vee (\neg A(x) \wedge B(x) \wedge \neg C(x)) \\ \vee (\neg A(x) \wedge \neg B(x) \wedge C(x)) \end{array} \right] \\ \wedge \\ \forall x \forall y \left(E(x,y) \rightarrow \neg \left[\begin{array}{r} (A(x) \wedge A(y)) \\ \vee (B(x) \wedge B(y)) \\ \vee (C(x) \wedge C(y)) \end{array} \right] \right) \end{array} \right)$$

This is similar to querying relational databases, except that the logic used here is more expressive than relational calculus, which cannot express quantification over relations such as $\exists A \exists B \exists C$.

ASP Formulation

Assume a finite set of edge-facts, for example:

```
edge(1,2). edge(1,3). edge(2,3).
```

The following program finds all 3-colorings:

```
adjacent(X,Y) :- edge(X,Y).
adjacent(X,Y) :- edge(Y,X).
vertex(X) :- adjacent(X,Y).

green(X) :- vertex(X), not red(X), not blue(X).
blue(X)  :- vertex(X), not red(X), not green(X).
red(X)   :- vertex(X), not blue(X), not green(X).

:- adjacent(X,Y), red(X), red(Y).
:- adjacent(X,Y), blue(X), blue(Y).
:- adjacent(X,Y), green(X), green(Y).
```

Table of Contents

1 Content and Methodology of the Course

2 Illustration of Automated Reasoning

3 Illustration of Solving

4 Recalls from Bases de Données I & II

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.

Data Complexity Classes

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
- 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 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.

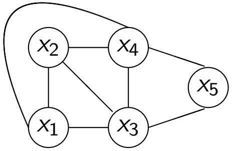
Query Complexity is NP-Complete for Conjunctive Queries

Fixed database

b, g, r are three distinct constants, representing three colors.

r	C_1	C_2
	b	g
	g	b
	b	r
	r	b
	g	r
	r	g

Query



$$q : Answer() \leftarrow R(x_1, x_2), R(x_2, x_1), \ldots, R(x_4, x_5), R(x_5, x_4)$$

Whenever there is an edge between x_i and x_j in the graph, the body of q contains $R(x_i, x_j)$ and $R(x_j, x_i)$.

Claim 1

$q(r) = \{Answer()\} \iff$ *the graph encoded by q is 3-colorable.*

Datalog$^\neg$

See "A Datalog Primer."

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

Datalog$^\neg$ 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
```