Représentation de connaissances

II – Règles ontologiques

et interrogation de bases de connaissances

Marie-Laure Mugnier

KNOWLEDGE BASED SYSTEMS

Knowledge Base (KB)

Reasoning Services

- **General knowledge on the application domain**
  « Cats are Mammals »

  Ontology (TBox in DL)

- **Factual Knowledge**
  Description of specific individuals, situations, ...

  Félix is a Cat

  Factbase, Database
  (Abox in DL)

Knowledge expressed in a KR language

Fundamental tasks

- **Analysing the ontology:** satisfiability of a concept, classification of concepts,...
- **Computing answers** to a query over the KB

...

Reasoning algorithms associated with the KR language

What kinds of Languages to Express Ontologies?

Very light languages

Hierarchies of classes
Hierarchies of binary relations (called « roles » or « properties »)
Signatures of these relations (« domain » and « range »)
→ **RDF Schema**

More expressive fragments of first-order logics

Description Logics
Rule-based languages Datalog, existential rules,
RDF deductive rules, Answer Set Programming ...

From a logical viewpoint: an ontology is composed of

a **finite set of predicates** (to express concepts and relations)
a **finite set of (closed) formulas** over these predicates
of the form **$\forall X$ (condition[X,...] → conclusion[X,...])**

Description Logics: standard reasoning tasks

Standard reasoning tasks on a KB ($\mathcal{T}$, $\mathcal{A}$)

w.r.t. = « with respect to »

- Concept subsumption — $\mathcal{T} \models C \sqsubseteq D$?
- Concept satisfiability — is C satisfiable w.r.t. $\mathcal{T}$?
- KB satisfiability — is ($\mathcal{T}$,$\mathcal{A}$) satisfiable ?
- Instance checking — ($\mathcal{T}$,$\mathcal{A}$) $\models$ C(b), where b is a constant?

All these tasks can be expressed in terms of KB (un)satisfiability provided that the constructors in the considered DL allow for it

Concept subsumption	$\mathcal{T} \models C \sqsubseteq D$ iff ($\mathcal{T}$, {C(a),¬D(a)}) unsatisfiable
Concept satisfiability	C satisfiable w.r.t. $\mathcal{T}$ iff ($\mathcal{T}$, {C(a)}) satisfiable
Instance checking	($\mathcal{T}$,$\mathcal{A}$) $\models$ C(b) iff ($\mathcal{T}$,$\mathcal{A}$∪{¬C(b)}) unsatisfiable

Query answering **beyond** instance checking?

Query answering beyond instance checking?

Standard expressive DL $\mathcal{ALC}$

- Concepts: $$C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists R.C \mid \neg C \mid C_1 \sqcup C_2 \mid \forall R.C$$
- TBox axioms: only concept inclusions

Query answering beyond instance checking?

Instance checking : ∃childOf.T (a) ? « Does *a* have a parent ? »

How to answer a more complex query (« conjunctive query »):

Q: ∃x (childOf(a,x) ∧ childOf(b,x)) ?

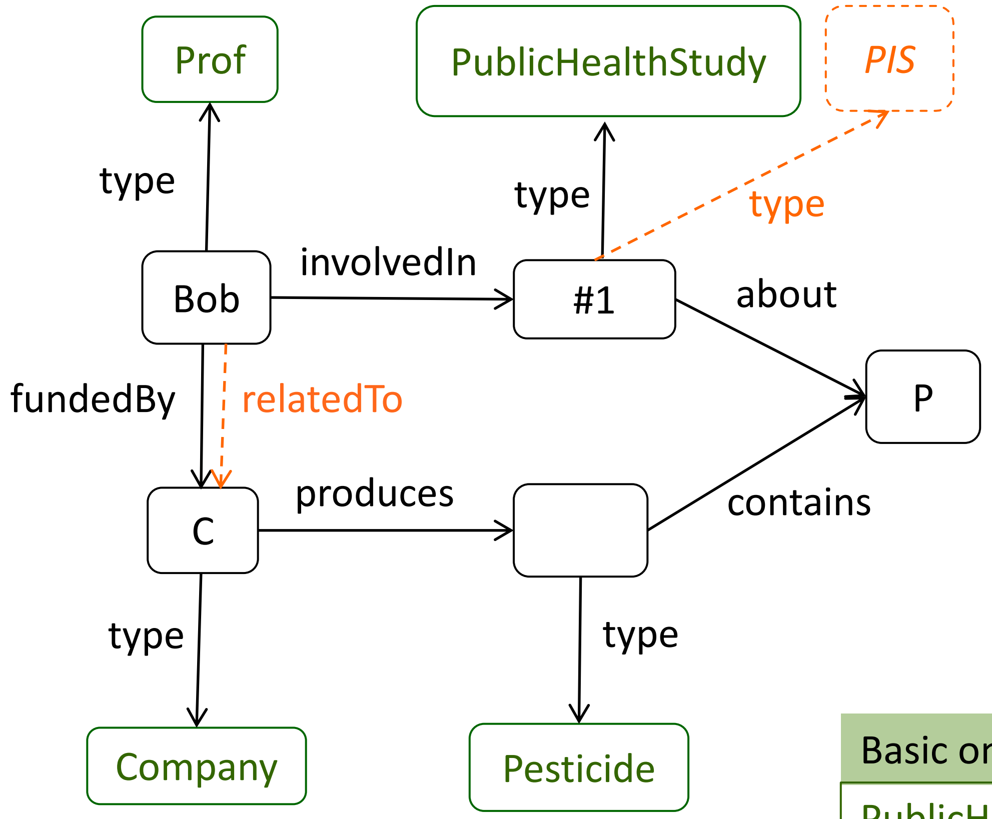
« Do *a* and *b* have a common parent? »

($\mathcal{T}$,$\mathcal{A}$) ⊨ Q ? cannot be reduced to a standard reasoning task

basically because Q cannot be turned into a concept

Query answering with expressive DLs requires other techniques than « tableaux ».
It has very high complexity and may even undecidable

ONTOLOGICAL KNOWLEDGE DESCRIBED BY RULES



Knowledge Graph
(could be seen as RDF triples)

Logical factbase

$\exists x$ (
Prof(Bob) $\wedge$
PHS(#1) $\wedge$
Comp(C) $\wedge$
Pest(x) $\wedge$
involvedIn(Bob,#1) $\wedge$
fundedBy(Bob,C) $\wedge$
about(#1,P) $\wedge$
produces(C,x) $\wedge$
contains(x,P))

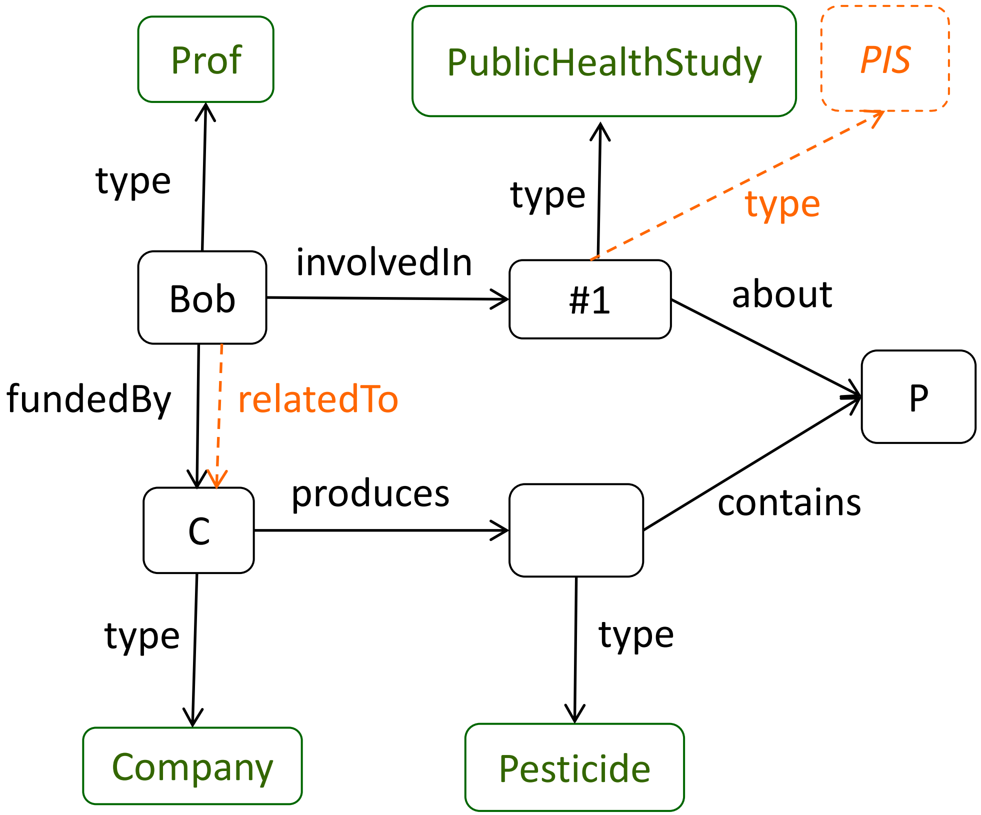
Basic ontological knowledge

PublicHealthStudy **subclass of** PublicInterestStudy
fundedBy **subproperty of** relatedTo

$\forall x\ (PHS(x) \rightarrow PIS(x))$
$\forall x \forall y\ (fundedBy(x,y) \rightarrow relatedTo(x,y))$

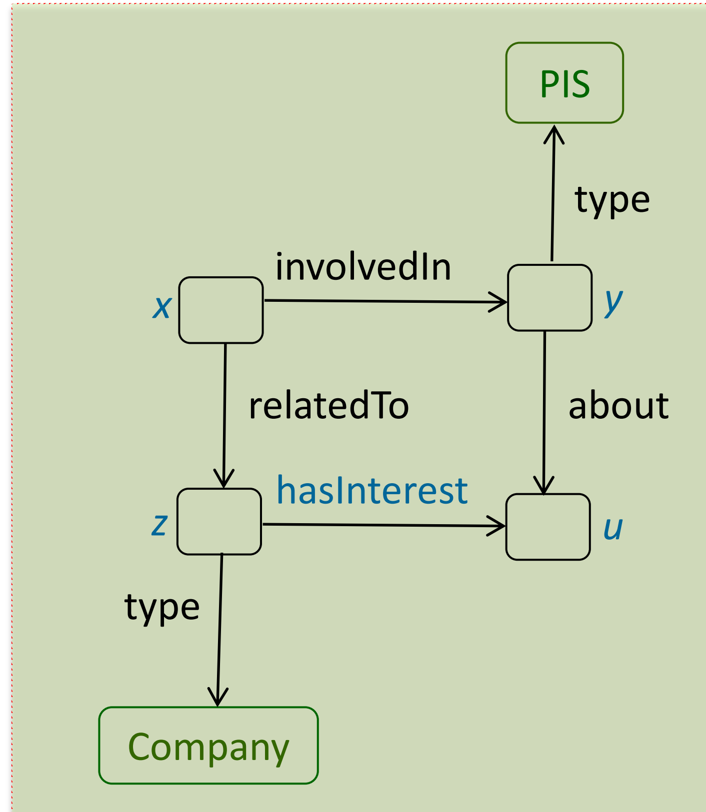
allows to infer: PIS(#1) , relatedTo(Bob,C)

How to Infer Conflicts of Interest (CoI) ?



Query: ''Find all **x, y, z** such that
x has a conflict for study **y**
because of its relationships with company **z**''

q(x,y,z) = ConflictOfInterest(x,y,z)

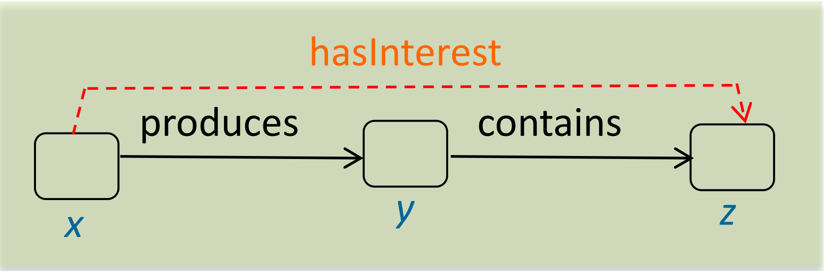


CoI pattern

What kind of **ontological knowledge** would allow to represent the notion of « conflict of interest »?

Defining Conflicts of Interest

R_1: ∀x∀y∀z (produces(x,y) ∧ contains(y,z)
→ hasInterest(x,z))



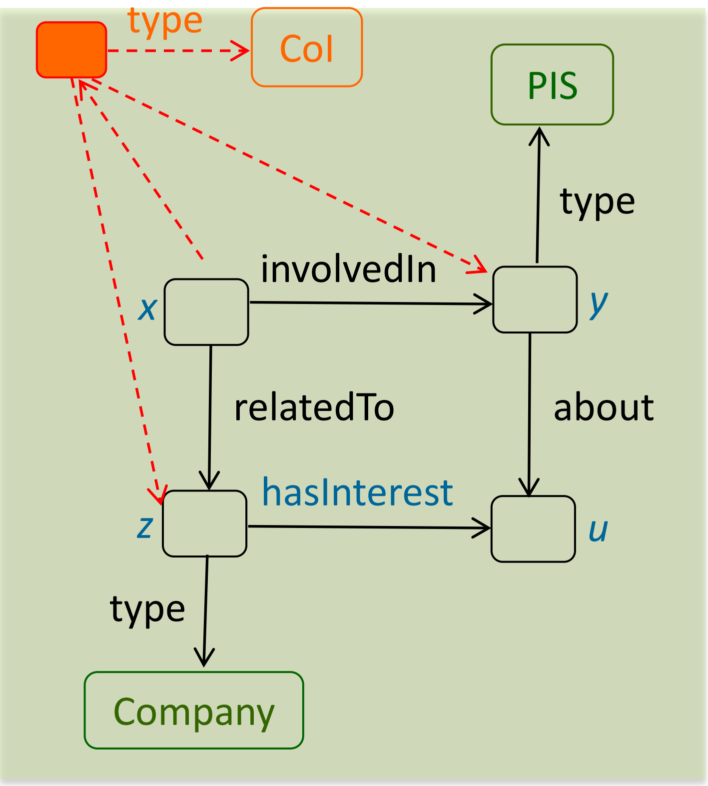
R_2: ∀x∀y∀z∀u (involvedIn(x,y) ∧ PIS(y) ∧ about(y,u) ∧ relatedTo(x,z) ∧ Company(z) ∧ hasInterest(z,u)
→ CoI(x,y,z))

We use here a ternary predicate.
What if we only have unary and binary predicates ?

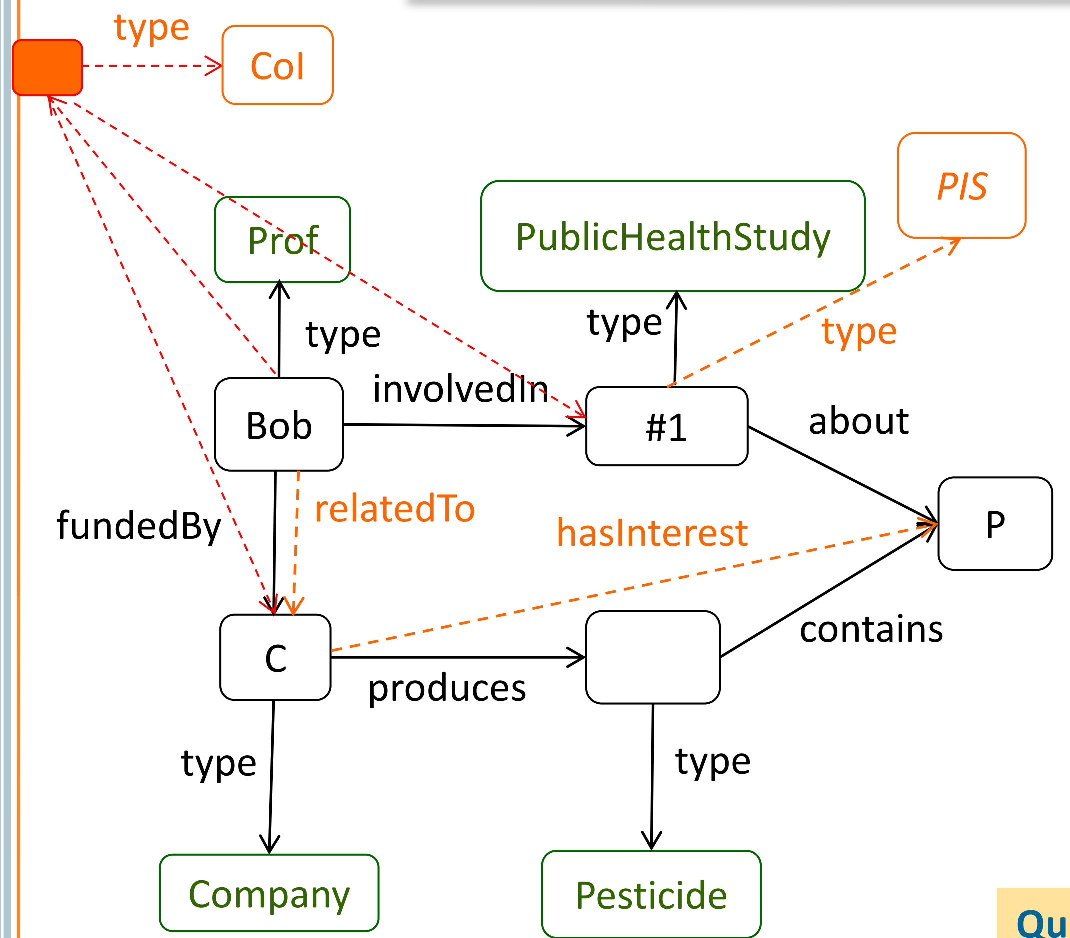
Reification: new object of type CoI

R_2: ∀x∀y∀z∀u (*body[x,y,z,u]* → ∃o
(CoI(o) ∧ in(x,o) ∧ on(o,y) ∧ with(o,z))

where body[x,y,z,u] is the CoI pattern



INFERRING CONFLICTS OF INTEREST



Rules (universal quantifiers omitted)

PHS(x) → PIS(x)
fundedBy(x,y) → relatedTo(x,y)

R_1: produces(x,y) ∧ contains(y,z)
→ hasInterest(x,z)

R_2: involvedIn(x,y) ∧ PIS(y) ∧ about(y,u) ∧
relatedTo(x,z) ∧ Company(z) ∧
hasInterest(z,u)
→ ∃o CoI(o) ∧ in(x,o) ∧ on(o,y) ∧ with(o,z)

Inferred facts

PIS(#1), relatedTo(Bob,C), hasInterest(C,P)
CoI(o_1), in(Bob,o_1), on(o_1,#1), with(o_1,C)

Facts

Prof(Bob), PHS(#1), Comp(C), Pest(x)
involvedIn(Bob,#1), fundedBy(Bob,C)
about(#1,P), produces(C,x), contains(x,P)

Query:
q(x,y,z) =
∃o CoI(o) ∧ in(x,o) ∧ on(o,y) ∧ with(o,z)

Answer: (Bob,#1,C)

Cadre étudié dans ce cours

- **Base de connaissances (KB)** composée :
  - d'une **base de faits**
    (qu'on peut voir comme une base de données relationnelle)
  - d'une **base de règles** positives et conjonctives
    (Datalog)
- **Requêtes conjonctives**
  (correspondant à des requêtes de base en SQL / SPARQl)
- **Problème fondamental : interrogation de la KB**
  (calculer toutes les réponses à une requête conjonctive sur la KB)

Extensions

- **Contraintes négatives**
- (on évoquera les règles existentielles qui généralisent Datalog)
- **Mappings** pour extraire une partie d'une base de données relationnelle et la traduire en une base de faits

FACTBASE

Vocabulary : $(\mathcal{P},\mathcal{C})$ where $\mathcal{P}$ is a finite set of predicates

$\mathcal{C}$ is a possibly infinite set of constants

[Arity of a predicate = its number of arguments]

$\mathcal{P}$ = { Prof/1, PHS/1, involvedIn/2, ...}

$\mathcal{C}$ = {Bob, #1, 456, ...}

Fact : a **ground** atom p(e1 ... ek) with p ∈ $\mathcal{P}$ and ei ∈ $\mathcal{C}$ [ground = no variables]

involvedIn(Bob,#1)

Factbase : usually a set of **ground atoms** on the vocabulary

F = { Prof(Bob), PHS(#1), involvedIn(Bob,#1) }

logically seen as the conjunction of these atoms

BD relationnelle vue comme une base de faits

- **Schéma** de BD : ensemble de relations (avec leurs attributs)

ex: Film [titre, directeur, acteur]
Pariscope [salle, titre, horaire]
Coordonnées [salle, adresse, téléphone]

On peut remplacer les attributs par une numérotation : 1,2,3

→ Vue logique : Film, Pariscope, Coordonnées sont des relations (prédicats) ternaires

- **Instance d'une relation (« table »)** : ensemble de k-uplets (où k est l'**arité** de la relation)

→ Vue logique :

valeurs : constantes

instance de relation : ensemble d'atomes

- **Instance de BD** : ensemble des instances de relation

Une instance de la relation Film

titre	*directeur*	*acteur*
The trouble	Hitchcock	Green
The trouble	Hitchcock	Forsythe
The trouble	Hitchcock	MacLaine
The trouble	Hitchcock	Hitchcock
Cries and Whispers	Bergman	Anderson

Vue logique :

{ film(t,h,g), film(t,h,f), film(t,h,m), film(t,h,h), film(c,b,a) }

Relational Database seen as a Factbase

A **relational database** may naturally be viewed as a factbase

Relational **schema** : finite set R of relations → predicates
infinite domain of values → constants

Instance of a relation r ∈ R : finite set of tuples on r → atoms on r

r

attr1	attr2
a1	a2
a2	a3
a1	a1

{ r(a1,a2), r(a2,a3), r(a1,a1) }

Database instance = { instance for each r in R } → factbase

FACTBASES CAN BE EXTENDED TO UNKNOWN VALUES

Relational database

Movie

m_id	
m1	…
m2	…
?x	

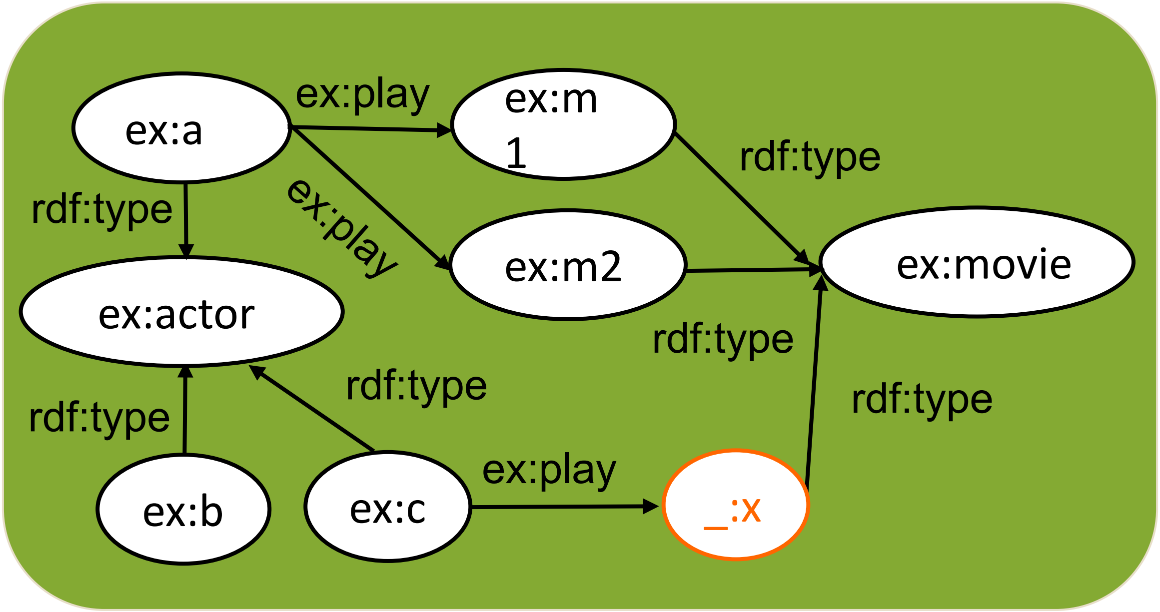
Actor

a_id	
a	…
b	…
c	

Play

m_id	a_id
a	m1
a	m2
c	?x

RDF



Etc.

Abstraction in first-order logic (FOL)

$\exists x$ (movie(m1) $\wedge$ movie(m2) $\wedge$ movie(x)
actor(a) $\wedge$ actor(b) $\wedge$ actor(c)
play(a,m1) $\wedge$ play(a,m2) $\wedge$ play(c,x))

We generalize here the classical notion of a fact by existential variables

factbase = existentially closed conjunction of atoms

LABELLED HYPERGRAPH / GRAPH REPRESENTATION

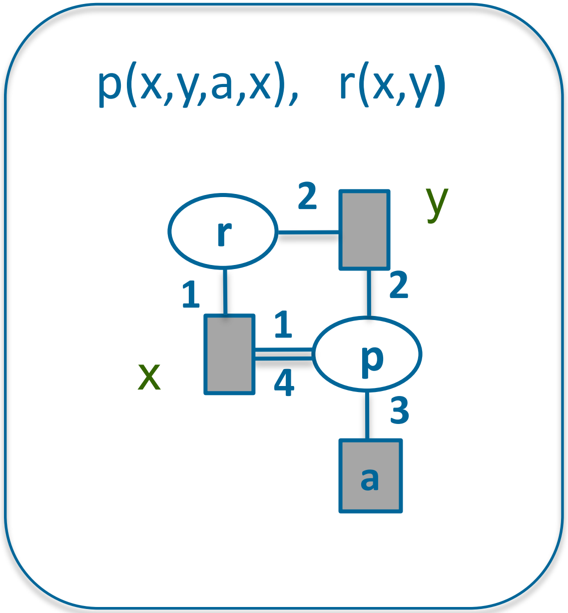
- A fact or a set of facts can be seen as a **set of atoms**

movie(m1), movie(m2), movie(x), actor(a), actor(b), actor(c), play(a,m1), play(a,m2), play(c,x)

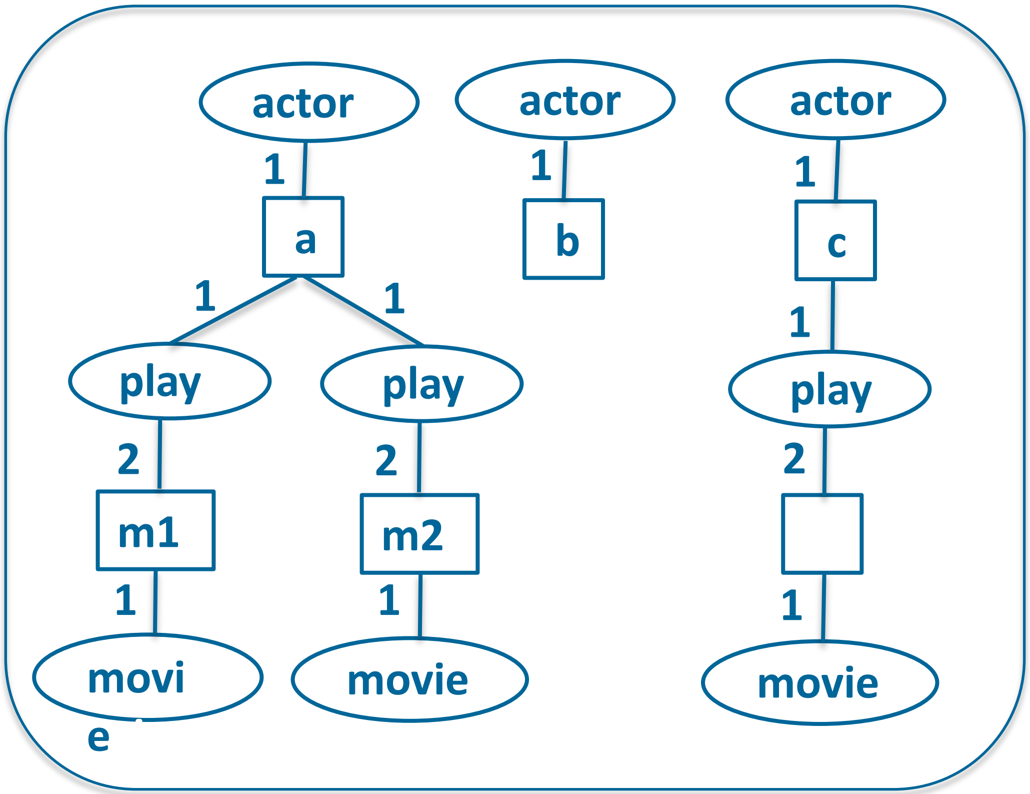
→ hence a **hypergraph**

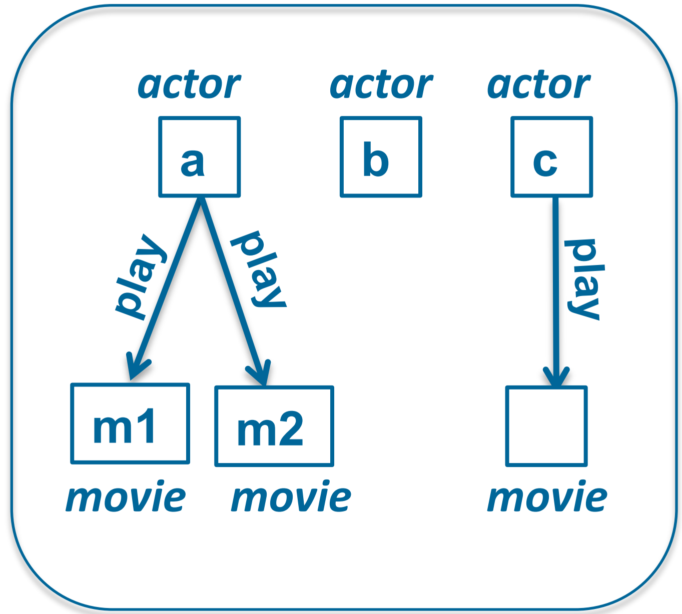
or its associated **bipartite (multi-)graph**

- one (labelled) node per term
- one (labelled) node per atom (~ hyperedge)
- totally ordered edges



movie(m1), movie(m2), movie(x), actor(a), actor(b), actor(c), play(a,m1), play(a,m2), play(c,x)





If predicates are at most binary:
atom nodes can be replaced by **labels** and **directed edges**

CONJUNCTIVE QUERIES (CQ)

$q(x) = \exists y\,(\text{movie}(y) \land \text{play}(x, y))$ *« find all those who play in a movie »*

$q() \;= \exists y\,(\text{movie}(y) \land \text{play}(a, y))$ *« does a play in a movie ? »*

A **CQ** is an **existentially quantified conjunction of atoms**

The **free variables** are the **answer variables**

If closed formula: **Boolean CQ**

Simplified notation

$q(x) = \{\ \text{movie}(y), \text{play}(x,y)\ \}$

Rule notation

$\text{ans}(x) \leftarrow \text{movie}(y), \text{play}(x, y)$ classical Datalog notation

$\text{movie}(y), \text{play}(x, y) \rightarrow \text{ans}(x)$ alternative notation

Basic SQL queries (on relational databases)

SELECT ... FROM ... WHERE *<equalities: restrictions and joins>*

Basic SPARQL (on RDF triples)

SELECT ... WHERE *<basic graph pattern>*

REQUÊTES CONJONCTIVES EN SQL

***En SQL:* «** SELECT ... FROM ... WHERE conditions de jointure »

Film [titre, directeur, acteur]
Pariscope [salle, titre, horaire]
Coordonnées [salle, adresse, tel]

« trouver les noms des films où Hitchcock joue »

SELECT Film.Titre FROM Film
WHERE Film.Acteur = « Hitchcock »

Vue logique ?

$q(x) = \exists y \, \text{Film}(x, y, \text{Hitchcock})$

« trouver les noms des salles dans lesquelles on joue un film de Bergman »

- Requête SQL ?
- Vue logique ?

« trouver les noms des salles dans lesquelles on joue un film de Bergman »

```
SELECT Pariscope.Salle
FROM Film, Pariscope
WHERE
        Film.Directeur = « Bergman »
        AND Film.Titre=Pariscope.Titre
```

Vue logique :

$$q(z) = \exists x\ \exists y\ \exists t\ (Film(x,Bergman,y) \wedge Pariscope(z,x,t))$$

KEY NOTION: HOMOMORPHISM

$q(x) = \exists\, y\, (movie(y) \wedge play(x, y))$

movie(y)
play(x, y)

F

movie(m1)
movie(m2)
movie(m3)
actor(a)
actor(b)
actor(c)
play(a,m1)
play(a,m2)
play(c,m3)

Homomorphism *h* from *q* to *F*:
substitution of var(*q*) *by terms(F)*
such that $h(q) \subseteq F$

$h1 : x \rightarrow a$, $y \rightarrow m1$ $\quad h1(q) = movie(m1) \wedge play(a, m1)$

$h2 : x \rightarrow a$, $y \rightarrow m2$ $\quad h2(q) = movie(m2) \wedge play(a, m2)$

$h3 : x \rightarrow c$, $y \rightarrow m3$ $\quad h3(q) = movie(m3) \wedge play(c, m3)$

Answers: obtained by restricting the domains of homomorphisms to answer variables

$x = a$
$x = c$

ANSWERS TO A CONJUNCTIVE QUERY

Let F be a factbase

- The **answer** to a Boolean CQ q in F is *yes* if $F \vDash q$ *yes = ()*
- Let the CQ $q(x_1,\ldots,x_k)$. A tuple $(a_1, \ldots, a_k)$ of *constants* is an answer to q on a factbase F if $F \vDash q[a_1,\ldots,a_k]$, where $q[a_1,\ldots,a_k]$ is the Boolean CQ obtained from $q(x_1,\ldots,x_k)$ by replacing each x_i by a_i
- Let F and q be seen as sets of atoms. A **homomorphism** h from q to F is a mapping from *variables(q)* to *terms(F)* such that $h(q) \subseteq F$

$F \vDash q()$ iff q can be mapped by homomorphism to F

$(a_1, \ldots, a_k)$ is an answer to $q(x_1,\ldots,x_k)$ on F iff there is a homomorphism from q to F that maps each x_i to a_i

EXEMPLE : INTERROGATION D'UNE BASE DE FAITS

BF F

p(a,b)
p(b,a)
p(a,c)
q(b,b)
q(a,c)
q(c,b)

$Q_1() \;\; = \{ p(x,y), p(y,z), q(z,x) \}$
$Q_2(x) = \{ \; p(x,y), p(y,z), q(z,x) \}$

Homomorphismes de Q_1 et Q_2 dans F ?

$x \mapsto b$
$y \mapsto a$
$z \mapsto c$

$x \mapsto b$
$y \mapsto a$
$z \mapsto b$

Donc ensembles des réponses à Q_1 et Q_2 dans F :

$Q_1(F) = \{()\}$ « yes »
$Q_2(F) = \{ (b) \}$

Ne pas confondre $Q_1(F) = \{()\}$ avec $Q_1(F) = \{\}$

RÈGLES POSITIVES A LA DATALOG (« RANGE-RESTRICTED »)

$\forall x_1 \ldots \forall x_n$ (B → H) **B for Body, H for Head**

Pour les DECOL : attention, en module IA, H était l'hypothèse, ici c'est la conclusion !

où :

- *B* est une conjonction d'atomes (hypothèse, prémisses, condition, *corps*)
- *H* est un atome (conclusion, *tête*)
- $x_1 \ldots x_n$ sont les variables du corps B
- **toutes** les variables de la tête **H** apparaissent dans le corps **B**

R_1: ∀x∀y∀z (produces(x,y) ∧ contains(y,z) → hasInterest(x,z)) — Datalog

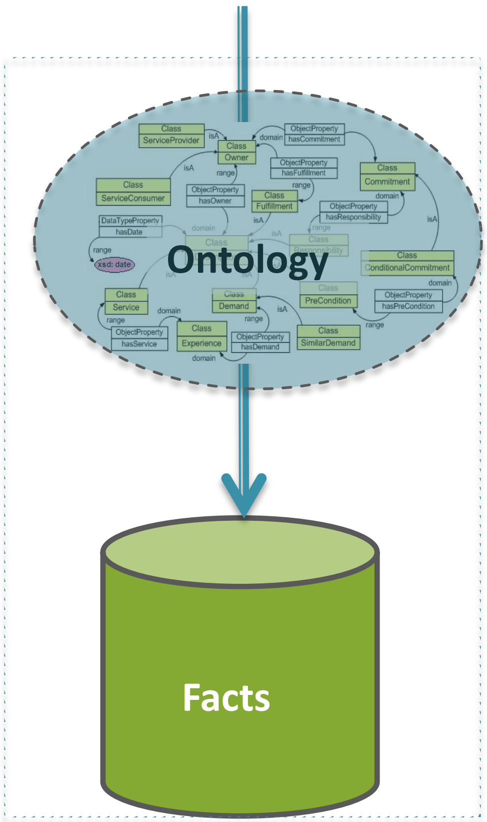
R_2: ∀x∀y∀z∀u (involvedIn(x,y) ∧ PIS(y) ∧ about(y,u) ∧ relatedTo(x,z) ∧ Company(z) ∧ hasInterest(z,u) → CoI(x,y,z)) — Datalog

R'_2: ∀x∀y∀z∀u (involvedIn(x,y) ∧ PIS(y) ∧ about(y,u) ∧ relatedTo(x,z) ∧ Company(z) ∧ hasInterest(z,u) → ∃o (CoI(o) ∧ in(x,o) ∧ on(o,y) ∧ with(o,z)) — pas Datalog (« règle existentielle »)

Notation simplifiée : sans ∀ et des virgules à la place des ∧

Query Answering on a KB

Query



Knowledge Base

The answer to a Boolean CQ Q in K is *yes* if $K \models Q$

A tuple $(a_1, \ldots, a_k)$ of *constants* is an answer to $Q(x_1,\ldots,x_k)$ with respect to K if $K \models Q[a_1,\ldots,a_k]$,

where $Q[a_1,\ldots,a_k]$ is obtained from $Q(x_1,\ldots,x_k)$ by replacing each x_i by a_i.

In our framework: $K = (F, \mathcal{R})$ where:

F is a (ground) factbase
$\mathcal{R}$ is a set of rules

K is logically seen as the conjunction of F and all rules in $\mathcal{R}$

HOW TO ACTUALLY COMPUTE THE ANSWERS TO A QUERY ON A KB?

Forward chaining : starting from F, we iteratively compute all the facts that are consequences of the current factbase and the rules.

$F = \{ \text{fundedBy}(Bob,C), \text{Company}(C) \}$
$R = \forall x \forall y\ (\text{fundedBy}(x,y) \rightarrow \text{relatedTo}(x,y))$
$F,R \models \text{relatedTo}(Bob,C)$

A rule $R: B \rightarrow H$ is applicable to a factbase F if there is a homomorphism h from B to F

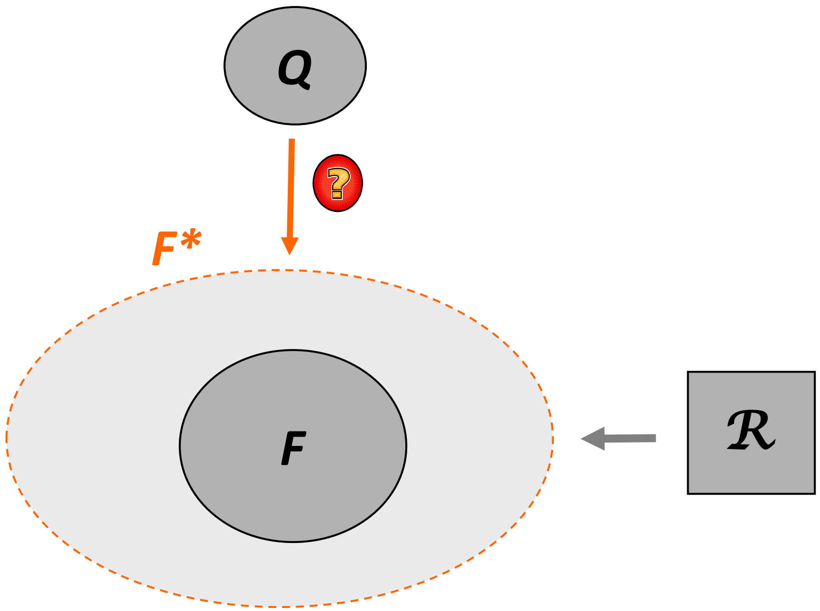
Applying R to F according to h consists of adding $h(H)$ to F

$h : \text{body}(R) \rightarrow F$
$x \mapsto Bob$
$y \mapsto C$

PROPERTIES OF DATALOG RULES

- $K = (F, \mathcal{R})$ where F is a set of (ground) facts
  $\mathcal{R}$ is a set of Datalog rules

By applying rules from $\mathcal{R}$ starting from F, a unique result is obtained: the **saturation** of F by $\mathcal{R}$ (denoted here by ***F****)



F^* is **finite** since no new variable is created

F^* allows to compute the answers to a CQ on K:

$(a_1, \ldots, a_k)$ is an answer to $q(x_1,\ldots,x_k)$ on K iff there is a homomorphism from q to K that maps each x_i to a_i

If k=0: () is an answer means « yes »

EXEMPLE (PISTES CYCLABLES)

F

Direct(A,B)
Direct(B,C)
Direct(C,D)
Direct(D,B)

$\mathcal{R}$

Direct(x,y) → Chemin(x,y)
Direct(x,y) ∧ Chemin(y,z) → Chemin(x,z)

Q(x) = Chemin(A,x) ∧ Chemin(x,D)

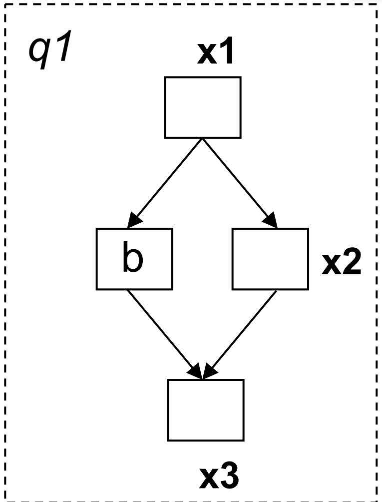
« trouver tous les x qui sont sur un chemin de A à D »

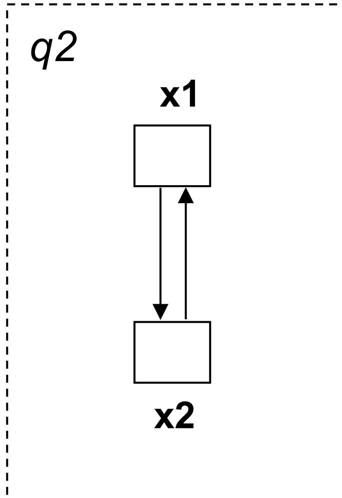
On cherche les homomorphismes de Q dans F*

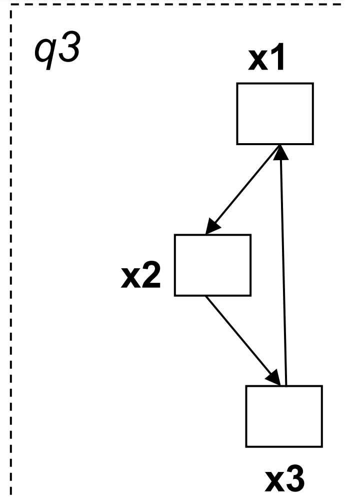
$x \mapsto B$ $x \mapsto C$ $x \mapsto D$

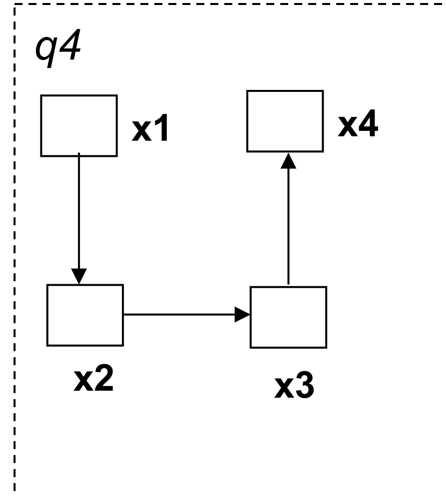
Q(F*) = { (B), (C), (D) }

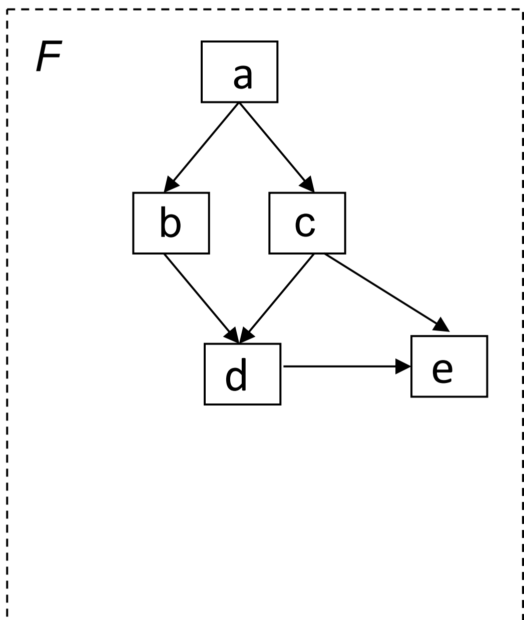
Exercice : Homomorphismes











Ces graphes représentent des requêtes (q_i) où les variables réponses sont x1 et x2 et une base de faits (F).

Il y a un seul prédicat binaire p.

Trouver tous les homomorphismes des q_i dans F.

En déduire les différents ensembles de réponses.