Théorie des bases de données et de connaissances
HAI933I

Jean-François BAGET (Inria)
David CARRAL (Inria)
Marie-Laure MUGNIER (Univ. Montpellier)

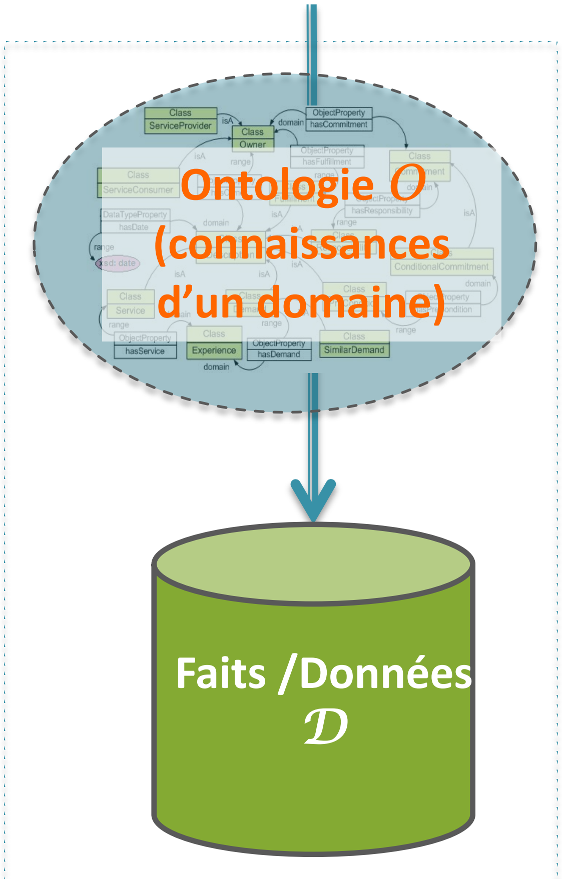
Equipe GraphIK (LIRMM & Inria)
https://team.inria.fr/graphik/

Problème central : Interrogation de bases de connaissances

Exploiter les connaissances du domaine modélisé lors de l'accès aux données

Formalisation en logique du premier ordre

Requête *Q*



Base de connaissances

Ontologie décrite par un langage plus ou moins expressif (logiques de description, règles, ...)

Recherche de « bons » compromis
entre l'**expressivité** du langage
et la **complexité** des raisonnements

Problèmes fondamentaux

- **Consistance / Satisfiabilité : ($\mathcal{D}$,$\mathcal{O}$) a-t-elle un modèle ?**
- **Interrogation : trouver toutes les réponses à *Q* qui sont conséquences de ($\mathcal{D}$,$\mathcal{O}$)**

« ontology-mediated query answering »

Contenu du module

Prérequis : module « représentation de connaissances » de M1
(ou module équivalent pour les étudiants extérieurs)

- **Rappels : Notions fondamentales** (conséquence logique, problème d'interrogation de bases de connaissances, requêtes conjonctives, règles datalog, contraintes négatives)
- **Règles existentielles** (alias **Datalog+**)
- **Liens entre règles existentielles et logiques de description**
- **Datalog avec négation**
- **Answer Set Programming (ASP)**
- **[Interrogation tolérante aux inconsistances]**

Etude des **fondations théoriques** de ces langages, **pas de TP!**

1. Brefs rappels de logique

INTERPRÉTATIONS

- **Vocabulaire:** $\mathcal{V} = (\mathcal{P}, \mathcal{C})$, où $\mathcal{P}$ = ensemble **fini** de **prédicats (**ou **relations)** chacun ayant une **arité** (nombre d'arguments)
  $\mathcal{C}$ = ensemble de constantes (peut être **infini**)

  ⇒ les formules sont construites sur ce vocabulaire

  ⇒ un **atome** est de la forme **$p(e_1, \ldots, e_k)$**
  où $p \in \mathcal{P}$ et chaque e_i (**terme**) est une constante de $\mathcal{C}$ ou une variable

- **Interprétation de $\mathcal{V}$:** $\mathcal{I} = (D_\mathcal{I}, .^\mathcal{I})$, où

  $D_\mathcal{I} \neq \emptyset$ (le **domaine** de l'interprétation)
  pour tout $c \in \mathcal{C}$, $c^\mathcal{I} \in D_\mathcal{I}$
  pour tout $p \in \mathcal{P}$ d'arité k, $p^\mathcal{I} \subseteq D_\mathcal{I}^{\,k}$

$\mathcal{V} = (\{p_{/2}, r_{/3}\}, \{a, b\})$

$\mathcal{I}$: $D_\mathcal{I} = \{d_1, d_2, d_3\}$

$a^\mathcal{I} = d_1$, $b^\mathcal{I} = d_2$
$p^\mathcal{I} = \{ (d_2, d_1), (d_2, d_3), (d_3, d_2) \}$
$r^\mathcal{I} = \{ (d_3, d_3, d_1) \}$

HYPOTHÈSE SIMPLIFICATRICE SUR LES INTERPRÉTATIONS

On va adopter une hypothèse couramment faite qui simplifiera nos notations :

- **hypothèse du nom unique** (**U**nique **N**ame **A**ssumption) :
  deux constantes différentes désignent forcément des objets différents

⇒ dans toute interprétation, deux constantes différentes s'interprètent par deux éléments de domaine différents

⇒ on peut donc simplier les notations en appelant par le même nom une constante et l'élément du domaine qui l'interprète
(« toute constante s'interprète par elle-même »)

$\mathcal{V} = (\{p_{/2}, r_{/3} \}, \{a, b\})$

$\mathcal{I}$:

$D_{\mathcal{I}} = \{d_1, d_2, d_3\}$
$a^{\mathcal{I}} = d_1$, $b^{\mathcal{I}} = d_2$
$p^{\mathcal{I}} = \{ (d_2, d_1), (d_2, d_3), (d_3, d_2) \}$
$r^{\mathcal{I}} = \{ (d_3, d_3, d_1) \}$

$D_{\mathcal{I}} = \{a, b, d_3\}$

$p^{\mathcal{I}} = \{ (b, a), (b, d_3), (d_3, b) \}$
$r^{\mathcal{I}} = \{ (d_3, d_3, a) \}$

INTERPRÉTATIONS (AVEC HYPOTHÈSE UNA)

- **Vocabulaire:** $\mathcal{V} = (\mathcal{P}, \mathcal{C})$, où
  $\mathcal{P}$ = ensemble fini de prédicats
  $\mathcal{C}$ = ensemble de constantes (peut être infini)

- **Interprétation de** $\mathcal{V}$: $\mathcal{I} = (D_{\mathcal{I}}, .^{\mathcal{I}})$, où
  $D_{\mathcal{I}} \neq \emptyset$ (le domaine de l'interprétation)
  $\mathcal{C} \subseteq D_{\mathcal{I}}$ (et pour tout $c \in \mathcal{C}$, $c^{\mathcal{I}} = c$)
  pour tout $p \in \mathcal{P}$ d'arité k, $p^{\mathcal{I}} \subseteq D_{\mathcal{I}}^{k}$

- $\mathcal{I}$ est un **modèle** d'une formule close f (construite sur $\mathcal{V}$) si f est vraie pour $\mathcal{I}$

$\mathcal{V} = (\{p_{/2}, r_{/3}\}, \{a, b\})$

$\mathcal{I}$: $D_{\mathcal{I}} = \{a, b, d_3\}$

$p^{\mathcal{I}} = \{ (b, a), (b, d_3), (d_3, b) \}$
$r^{\mathcal{I}} = \{ (d_3, d_3, a) \}$

$f_1 = \exists x\, \exists y\, (\, p(b,x) \wedge r(x,x,y)\,)$ oui

$f_2 = p(a,b) \wedge p(b,a)$ non

$f_3 = \exists x\, p(x,y)$

pas close

On n'interprétera que des formules closes

CONSÉQUENCE LOGIQUE

Etant données deux formules (closes) *f* et *g*,

$f \vDash g$ (*g* est conséquence de *f*)

signifie que

tout modèle de *f* est un modèle de *g*

(« dans toute situation où *f* est vraie, *g* est forcément vraie aussi »)

$f_1 : p(a) \land \forall x\,(p(x) \rightarrow q(x))$

$f_2 : q(a)$

$f_3 : p(a) \land \exists x\, q(x)$

$f_4 : p(a) \land \neg q(a) \land \forall x\,(p(x) \rightarrow q(x))$

$f_1 \vDash f_2, f_3$

$f_4 \vDash f_1, f_2, f_3$

2. Notions fondamentales sur les bases de connaissances

Cadre étudié en M1

- **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

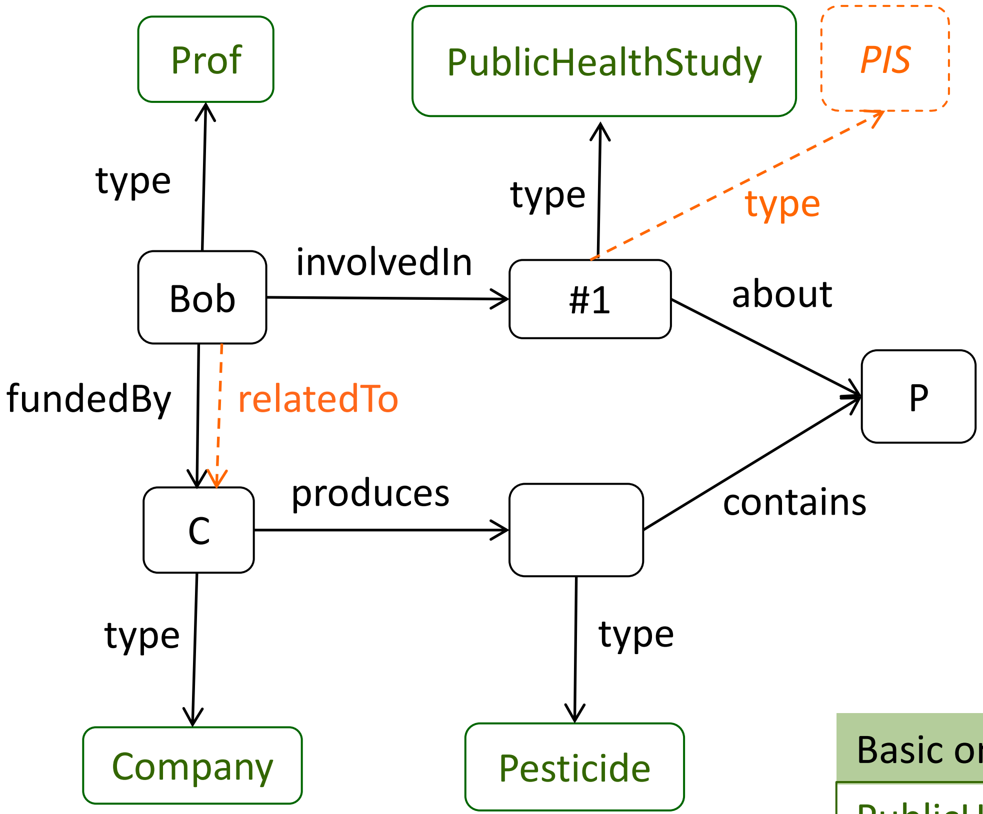
    (Datalog)
- **Requêtes conjonctives**

  (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)
- **Techniques** : chaînage avant, chaînage arrière, réécriture de requête

Extensions

- **Contraintes négatives**
- **Mappings** pour extraire une partie d'une base de données relationnelle et la traduire en une base de faits

EXAMPLE



Knowledge Graph
(could be seen as RDF triples)

Factbase

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

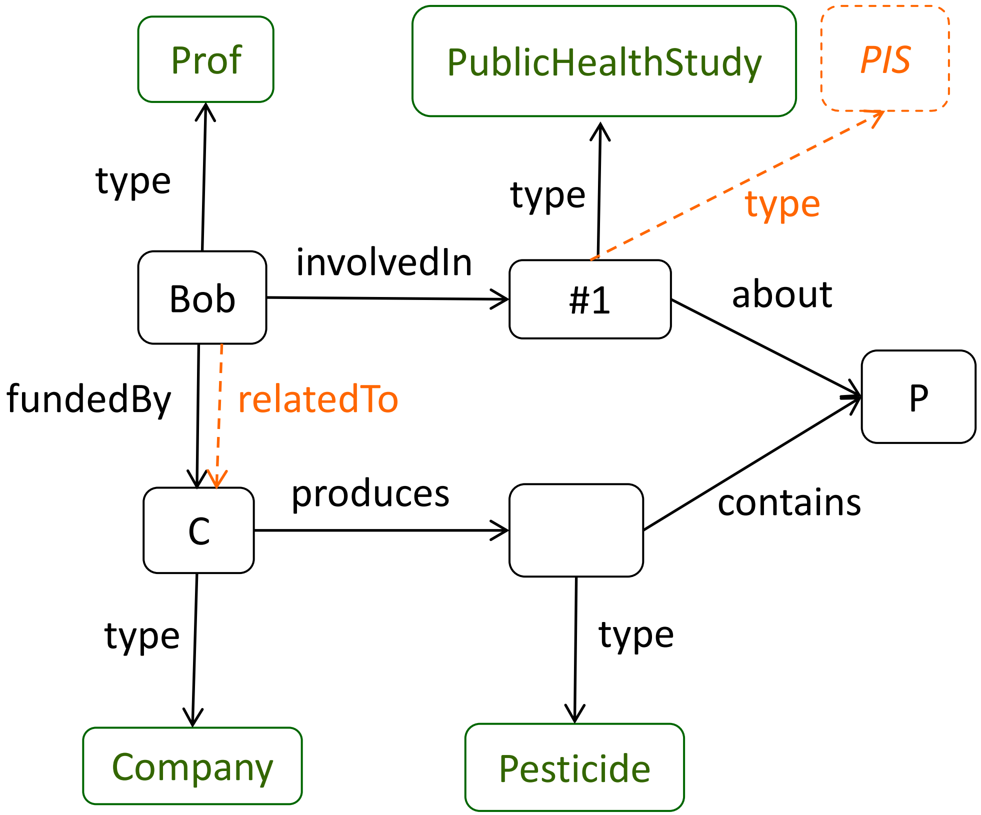
Basic ontological knowledge

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

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

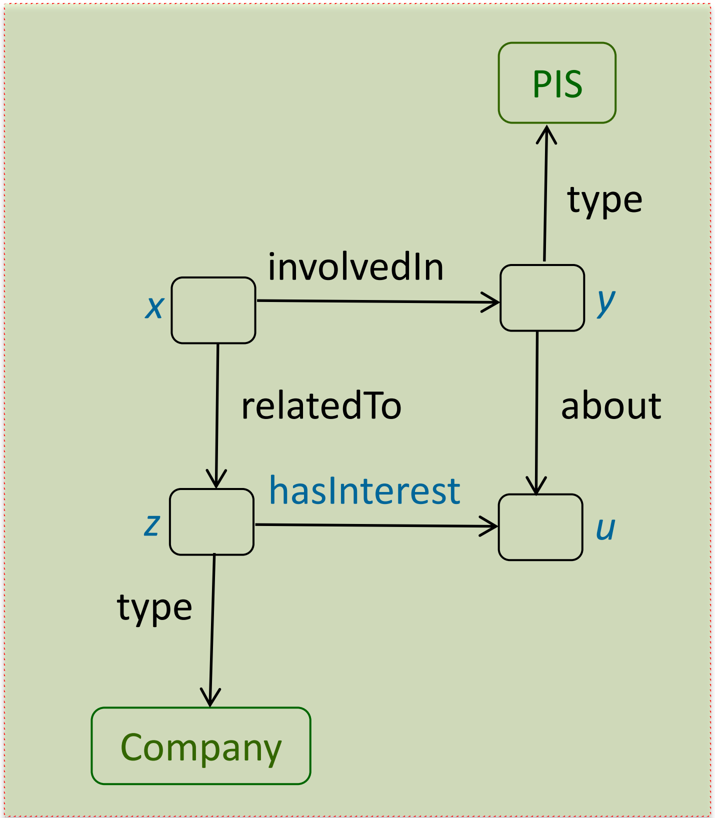
allows to infer: PIS(#1) , relatedTo(Bob,C)

Example: 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 »?

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

Prof(Bob) ∧ PHS(#1) ∧ involvedIn(Bob,#1)

Relational Database seen as a Factbase, and vice versa

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

Relational **schema** :	finite set R of k-ary relations	→ k-ary predicates
	infinite domain of values	→ constants
Instance of a relation r ∈ R :	finite set of k-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

And reciprocally: a **factbase** can be seen as (stored in) a relational database

Factbases can be extended to unknown values

An unknown value is logically seen as an **existentially quantified variable**
Then a factbase is logically seen as the existential closure of the conjunction of its atoms

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

Factbase

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

Logical formula assigned to the factbase

$$\exists x\ (\ \mathrm{movie}(m1) \land \mathrm{movie}(m2) \land \mathrm{movie}(x)$$
$$\mathrm{actor}(a) \land \mathrm{actor}(b) \land \mathrm{actor}(c)$$
$$\mathrm{play}(a,m1) \land \mathrm{play}(a,m2) \land \mathrm{play}(c,x)\ )$$

Conjunctive Queries (CQ)

q(x) = ∃y (movie(y) ∧ play(x, y)) *« find all those who play in a movie »*
q() = ∃y (movie(y) ∧ play(b, y)) *« does* b *play in a movie ? » (*b *is a constant)*

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) = { movie(y), play(x,y) }

Rule notation

ans(x) ← movie(y), play(x, y) classical Datalog notation
movie(y), play(x, y) → 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>*

ANSWERING CQS ON A FACTBASE

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

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 : $x \rightarrow a$
$y \rightarrow m1$

$h1(q) = movie(m1) \wedge play(a, m1)$

h2 : $x \rightarrow a$
$y \rightarrow m2$

$h2(q) = movie(m2) \wedge play(a, m2)$

h3 : $x \rightarrow c$
$y \rightarrow m3$

$h3(q) = movie(m3) \wedge play(c, m3)$

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 $F \models q$ *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 $F \models q[a_1,\ldots,a_k]$,
  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 $h(q) \subseteq F$

$F \models q()$ 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

Homomorphisme et Conséquence logique

Etant données deux formules *f* et *g*,

f $\models$ ***g*** (*g* est conséquence de *f*)
signifie que
tout modèle de *f* est un modèle de *g*

Base de faits *F* — CQ booléenne q() — *vues comme des ensembles d'atomes*

F $\models$ *q()* **ssi** il existe un homomorphisme de *q* dans *F*

Pourquoi ?

MODÈLES D'UNE BASE DE FAITS (SANS VARIABLES)

F = { p(a,b), p(b,c),q(c) }

Si une interprétation *I* est un modèle de *F*, que contient-elle *forcément* ?

$p^{\mathcal{I}}$ contient forcément (a,b) et (b,c)
$q^{\mathcal{I}}$ contient forcément c

Qu'y a-t-il de commun à *tous* les modèles de *F* ?

$p^{\mathcal{I}} = \{ (a,b), (b,c) \}$
$q^{\mathcal{I}} = \{ c \}$

Un **plus petit modèle** d'une formule f est un modèle de f qui n'est plus un modèle si on enlève un élément de l'interprétation d'un prédicat

Une base de faits (sans variables) a un **unique plus petit modèle**

$\mathcal{I}$:

$D_{\mathcal{I}} = \{a,b,c, ...\} = C$
$p^{\mathcal{I}} = \{ (a,b), (b,c) \}$
$q^{\mathcal{I}} = \{ c \}$

Et toute constante du vocabulaire s'interprète par elle-même

Modèle canonique d'une base de faits (sans variables)

Vocabulaire $\mathcal{V} = (\mathcal{P}, \mathcal{C})$
Base de faits F (sans variables) sur $\mathcal{V}$

Modèle **canonique** de F

$\mathcal{M}$: $D^{\mathcal{M}} = \mathcal{C}$
pour tout $p \in \mathcal{P}$ d'arité k, $p^{\mathcal{M}} = \{ (c_1, \ldots, c_k) \mid p(c_1, \ldots, c_k) \in F \}$

Le modèle canonique de F correspond à l'**intersection** de tous les modèles de F

$\mathcal{V} = (\{r_{/3}, p_{/2}, q_{/1} \}, \{a, b, c, d, e\})$

$F = \{ p(a,b), p(b,c), q(c) \}$

$\mathcal{M}$: $D_{\mathcal{M}} = \{a,b,c,d,e\}$
$p^{\mathcal{M}} = \{ (a,b), (b,c) \}$
$q^{\mathcal{M}} = \{ c \}$
$r^{\mathcal{M}} = \emptyset$

Ces formules (sans variables) ont-elles un unique plus petit modèle?

$f = p(a) \vee p(b)$ — 2 plus petits modèles M_1 et M_2 avec $p^{M1} = \{a\}$
$p^{M2} = \{b\}$

$f = p(a) \rightarrow p(b)$
$\equiv$
$\neg p(a) \vee p(b)$ — 1 plus petit modèle M avec $p^M = \emptyset$

$f = p(a) \wedge (p(a) \rightarrow p(b))$ — 1 plus petit modèle M avec $p^M = \{a,b\}$

$f = p(a) \rightarrow \neg p(a)$ — 1 plus petit modèle M avec $p^M = \emptyset$

$f = p(a) \wedge \neg p(a)$ — pas de modèle (insatisfiable)

Qu'est-ce qu'un modèle d'une CQ Booléenne ?

$q()$ = ∃x ∃y ∃z (p(x,y) ∧ p(y,z) ∧ r(x,z,a))

$\mathcal{I}$:

$D_{\mathcal{I}} = \{a,b,c\}$
$p^{\mathcal{I}} = \{ (a,b), (b,c) \}$
$r^{\mathcal{I}} = \{ (a,b,c), (b,c,a) \}$

$\mathcal{I}$ n'est pas un modèle de q

Une interprétation I est un modèle de q si :

il existe une application f des termes de $q()$ dans D_I telle que :

1. $f(c) = c$ pour toute constante c

2. pour tout atome $p(e_1,\ldots,e_k)$ de q, on a $(f(e_1), \ldots f(e_k)) \in p^I$

HOMOMORPHISME ET CONSÉQUENCE LOGIQUE

Base de faits *F*
CQ booléenne q()
$F \models q()$ **ssi** il existe un homomorphisme de *q* dans *F*

Pourquoi ?

- (⇒) Supposons que $F \models q$, c'est-à-dire « tout modèle de *F* est un modèle de *q* »
  Prenons en particulier le modèle de canonique de *F* (soit *M*)
  M est un modèle de *q*
  Il existe donc une application *f* des termes de *q* dans D_M
  telle que :
  1. $f(c) = c$ pour toute constante *c*
  2. pour tout atome $p(e_1,\ldots,e_k)$ de q, $(f(e_1), \ldots f(e_k)) \in p^M$

  f définit un homomorphisme de *q* dans *F*

- (⇐) Soit *h* un homomorphisme de *q* dans *F*
  h montre que le modèle canonique de *F* est un modèle de *q*
  donc tout modèle de *F* est un modèle de *q*
  c'est-à-dire $F \models q$

Règles positives a la Datalog (« range-restricted »)

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

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 **H** apparaissent dans **B**

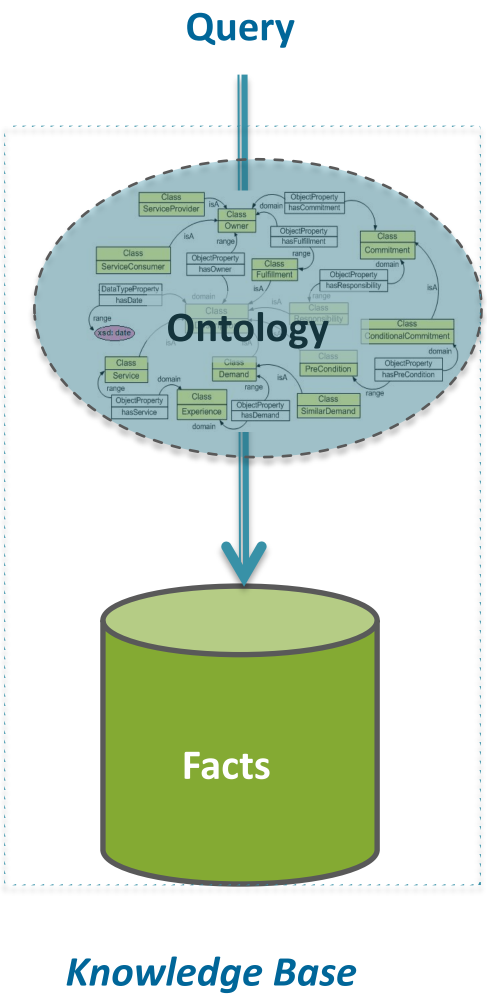
R_1: ∀x∀y∀z (produces(x,y) ∧ contains(y,z) → hasInterest(x,z)) Datalog

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 answering problem: given a KB K and a query Q, find the set of all answers to Q in K (notation: $Q(K)$)

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 $\mathcal{R}$

MODÈLES D'UNE KB (BASE DE FAITS, RÈGLES DATALOG)

K = (F, $\mathcal{R}$) est vue d'un point de vue logique comme la conjonction de F et de toutes les règles de $\mathcal{R}$

donc : un modèle de *K* est un modèle de chaque fait de *F* et chaque règle de $\mathcal{R}$

K = (F, $\mathcal{R}$)
F = {p(a,b), p(b,c) }
$\mathcal{R}$ = {R_1,R_2} avec R_1 : p(x,y) → q(y)
R_2 : q(x), p(x,y) → r(y) }

I est un modèle de K **ssi** :

- I modèle de F : (a,b) ∈ pI et (b,c) ∈ p^I
- I modèle de R_1 : pour tout couple $(d_1,d_2) \in p^I$, on a $d_2 \in q^I$
- I modèle de R_2 : pour tout $d_1 \in q^I$ et $(d_1,d_2) \in p^I$, on a $d_2 \in r^I$

Exemple

$K = (F, \mathcal{R})$

$F = \{p(a,b), p(b,c)\}$

$\mathcal{R} = \{R_1, R_2\}$ avec $R_1 : p(x,y) \rightarrow q(y)$

$R_2 : q(x), p(x,y) \rightarrow r(y)\}$

$I = (D, .^I)$ avec $D = \{a, b, c, e\}$ Rappel : *a, b et c désignent en fait a^I, b^I et c^I*

$p^I = \{(a,b), (b,c), (e,c)\}$

$q^I = \{b, c\}$

$r^I = \{a\}$

I est-elle un modèle de K ?

$I = (D, .^I)$ avec $D = \{a, b, c, e\}$

$p^I = \{(a,b), (b,c)\}$

$q^I = \{b, c\}$

$r^I = \{a,c\}$

I est-elle un modèle de K ?

$I = (D, .^I)$ avec $D = \{a, b, c, e\}$

$p^I = \{(a,b), (b,c)\}$

$q^I = \{b, c\}$

$r^I = \{c\}$

I est-elle un modèle de K ?

PROPRIÉTÉ DU PLUS PETIT MODÈLE UNIQUE

Toute base de connaissances K =(F,$\mathcal{R}$) où $\mathcal{R}$ est un ensemble de règles Datalog possède un unique plus petit modèle M:

pour tout modèle I de K, pour tout prédicat p, on a $p^M \subseteq p^I$

F = {p(a,b), p(b,c) }
$\mathcal{R}$ = {R_1,R_2} avec R_1 : p(x,y) → q(y)
R_2 : q(x), p(x,y) → r(y) }

Quel est son plus petit modèle ?

M = (D, $.^M$) avec D = { a, b, c }
p^M = { (a,b),(b,c) }
q^M = { b, c }
r^M = { c }

Etant donnée une CQ booléenne q, pour déterminer si K ⊨ q il suffit donc de vérifier si le plus petit modèle de K est un modèle de q :

- si oui, tout modèle de K contient ce modèle, c'est donc un modèle de q
- si non, on a un modèle de K qui n'est pas un modèle de q

How to actually compute this model?

Idea : starting from the canonical model of F,
we add all the tuples that are mandatory to satisfy the rules

⇒ Forward chaining : starting from F, iteratively add the atoms that are consequences of F and the rules

F = { fundedBy(Bob,C), Company(C) }
R = ∀x∀y (fundedBy(x,y) → relatedTo(x,y))
F,R ⊨ relatedTo(Bob,C)

A rule *R: B → 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 : body(R) ⟶ F
x ↦ Bob
y ↦ 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

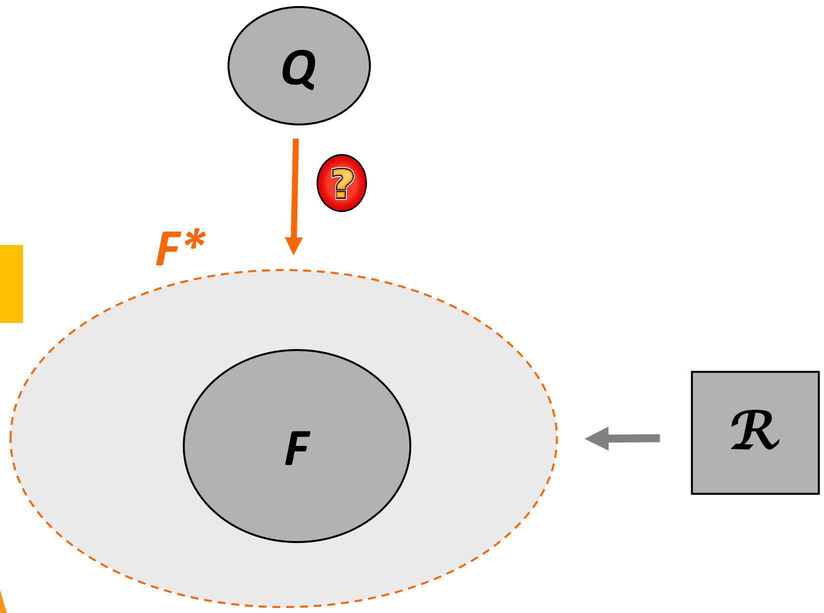
By applying rules from $\mathcal{R}$ starting from F, a unique result is obtained: the **saturation** of F by $\mathcal{R}$ (denoted here by ***F****)

*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 F^* that maps each x_i to a_i

If k=0: () is an answer means « yes »



Why? Because the canonical model of the saturated factbase F* is the **unique smallest model** of *K*

Si on ajoute des contraintes négatives

- Def: Une **contrainte négative** est de la forme

$$\forall X \,(\, \text{Condition}[X] \rightarrow \bot\,) \qquad \equiv \quad \neg \exists X \;\text{Condition}[X]$$

où *Condition* est une conjonction d'atomes et $\bot$ le symbole absurde

$$\forall x\, (\text{Film}(x) \wedge \text{Personne}\,(x) \rightarrow \bot)$$

- Def: Une base de faits *F* **satisfait une contrainte négative** *C* s'il n'y a pas d'homomorphisme de la condition de *C* dans *F* (autrement dit, *C* vue comme une règle n'est pas applicable)

Remarque : $F \cup \{C\}$ est consistante (satisfiable) **ssi** *F* satisfait *C*

- Prop: Une base de connaissances $K = (F, \mathcal{R}, \mathcal{C})$ où $\mathcal{C}$ est un ensemble de contraintes négatives est consistante (satisfiable) **ssi** F^* (la saturation de *F* par $\mathcal{R}$) satisfait toutes les contraintes de $\mathcal{C}$

EXERCICE (APPLICATION DIRECTE DU COURS)

Soit la KB $\mathcal{K} = (F, \mathcal{R}, \mathcal{C})$

F = { r(a,b), r(b,c), r(c,a) }

$\mathcal{R}$ = {r(x,y) → s(x,y) ; s(x,y) ∧ s(y,z) → s(x,z) }

$\mathcal{C}$ = {s(x,y) ∧ s(y,x) → ⊥}

- L'interprétation I :

  D = {a,b,c, d,e}

  r^I = {(a,b), (b,c), (c,a), (d,e)}

  s^I = D x D

  est-elle un modèle de $(F,\mathcal{R})$?
- Quel est le plus petit modèle de $(F,\mathcal{R})$?
- F satisfait-elle $\mathcal{C}$? $\mathcal{K}$ satisfait-elle $\mathcal{C}$?
- $\mathcal{K}$ est-elle consistante (satisfiable) ?
- Soit q() = ∃x lapin(x). $\mathcal{K}$ répond-t-elle oui à q?

INTERROGATION DE KBS AVEC CONTRAINTES NÉGATIVES

Soit une base de connaissances *K = (F, R, C)*

1. *K* est-elle satisfiable ?

On calcule *F** la saturation de *F* par *R*
Puis on teste si *F** satisfait *C*

2. Interrogation de *K*

Si K n'est pas satisfiable, le problème d'interrogation « trivialise »

Sinon, les réponses à une CQ *q* sont données par
les homomorphismes de *q* dans *F**