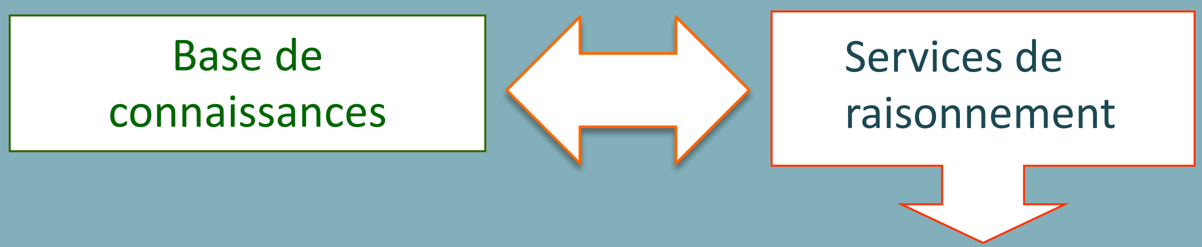
Théorie des bases de données et de connaissances
HAI933I

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

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

DOMAINE : REPRÉSENTATION DE CONNAISSANCES ET RAISONNEMENTS

- Domaine de recherche historiquement au coeur de l'Intelligence Artificielle :
  étude de formalismes de représentation de connaissances et de raisonnement
- Les résultats ont largement essaimé hors de l'IA :
  web sémantique, gestion de données, recherche d'information, etc.



Test de consistance de la base
Réponses à une requête
Explication d'une inférence **etc. etc.**

Conférence internationale emblématique : **KR** https://kr.org/KR2023/

Dans ce module, lien fort avec la théorie des bases de données

Contenu du module

Prérequis : module « **traitement sémantique des données** » de M1

- **Notions de base**
  - **Rappels** (logique, cours de M1)
  - Les objets de base vus comme des **graphes** et hypergraphes
- **Requêtes du premier ordre et algèbre relationnel** (fondement théorique de SQL)
- **Règles existentielles** (alias **Datalog+**)
- **Datalog avec négation**
- **Answer Set Programming / règles existentielles avec négation**

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

Contrôle des connaissances : contrôle continu 40%, contrôle terminal 60%

1. Rappels de logique du premier ordre

Syntaxe : formules construites sur un vocabulaire

- **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**)

  $\Rightarrow$ les formules sont construites sur un vocabulaire

- **Terme** sur $\mathcal{V}$: constante $c \in \mathcal{C}$ ou variable « objet »
  (on ne considère pas les symboles de fonction d'arité > 0 ici)

- **Atome** sur $\mathcal{V}$: de la forme $\mathbf{p(t_1, ..., t_k)}$
  où $p \in \mathcal{P}$ et chaque t_i est un terme sur $\mathcal{V}$

- **Formule** sur $\mathcal{V}$: se définit par induction
  - base : atome sur $\mathcal{V}$: $p(t_1 \ldots t_k)$ « formule atomique »
  - règle de construction : $\neg A$, « formule complexe »
    $(A \land B)$, $(A \lor B)$, $(A \longrightarrow B)$, $(A \longleftrightarrow B)$,
    $\exists x\, A$, $\forall x\, A$
    où A et B sont des formules sur $\mathcal{V}$

- **Variable libre** : l'une de ces occurrences hors de la portée d'un quantificateur
- **Formule close (fermée)** : sans variable libre

Sémantique : Interprétations

- **Interprétation de** $\mathcal{V} = (\mathcal{P}, \mathcal{C})$:

  $\mathcal{I} = (D_\mathcal{I}, .^\mathcal{I})$ 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** (Unique Name Assumption) :
  deux constantes différentes désignent forcément des objets différents

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

⇒ on peut donc simplifier 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 $\mathcal{I}$

  On dit aussi que $\mathcal{I}$ satisfait f

  - Une formule est satisfiable si elle a un modèle. Sinon, elle est insatisfiable.

EXEMPLE

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

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

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

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

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

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

$f_3 = \forall x \, \neg p(x,x)$ oui

$f_4 = \exists x \, p(x,y)$ 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 tout monde où *f* est vraie, *g* est forcément vraie aussi »)

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

$f_1 \vDash f_2, f_3$

$f_2 : q(a)$

$f_4 \vDash f_1, f_2, f_3$

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

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

f_4 est insatisfiable (n'a pas de modèle)
Donc tout est conséquence de f_4

2. Rappels

sur les bases de données
et bases de connaissances

(cours de M1)

Cadre étudié en M1

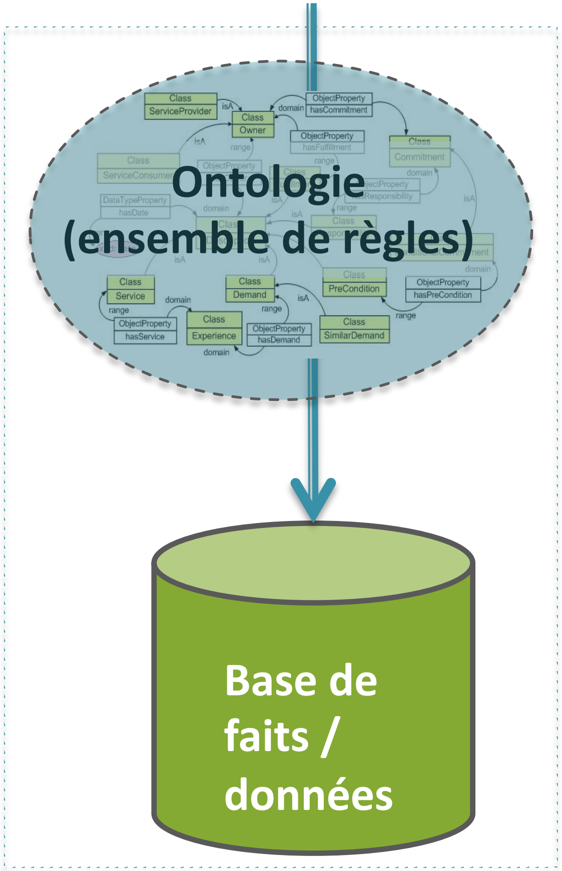
- **Base de connaissances (KB)** composée :
  - d'une **base de faits** (ou base de données relationnelle)
  - d'une **base de règles** positives et conjonctives (Datalog)
- **Requêtes conjonctives**

  (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 sélectionner une partie d'une BD relationnelle
  et la traduire en une base de faits

Bases de connaissances



Base de connaissances

- Ontologie : ensemble fini de règles
- Atome instancié (*ground*) : sans variables
- Fait = atome instancié
- Base de faits = ensemble fini de faits

Nous allons voir que :

- toute BD relationnelle peut être vue comme une base de faits
- toute base de faits peut être vue comme une BD relationnelle

BD RELATIONNELLE = ENSEMBLE DE TABLES

Film

Titre	Directeur	Acteur
Pulp fiction	Q. Tarantino	J. Travolta
Pulp fiction	Q. Tarantino	Q. Tarantino
Pulp fiction	Q. Tarantino	U. Thurman
Grease	R. Kleiser	J. Travolta

Programme

Cinéma	Titre	Horaire
Diagonal	Pulp Fiction	01/03/2022 à 20h

Lieu

Cinéma	Adresse	Site web
Diagonal	...	...

Toute table obéit à un **schéma**

BD RELATIONNELLE (VUE ABSTRAITE)

- Schéma de relation $R[A_1 \ldots A_k]$: R est le nom de la relation d'arité k
  $A_1 \ldots A_k$ est une liste de k attributs (distincts)

- Schéma *S* d'une BD : ensemble fini de schémas de relations

  ex: Film [titre, directeur, acteur]
  Programme [cinéma, titre, horaire]
  Lieu [cinéma, adresse, site web]

- Domaine (noté *dom*): ensemble (fini ou infini)
  c'est l'ensemble des valeurs possibles dans les tables

- Table (ou instance de schéma de relation) pour $R[A_1 \ldots A_k]$ sur *dom* :
  ensemble fini de k-uplets sur *dom*

- Base de données sur *(S, dom)* :
  ensemble fini de tables sur *dom,* comportant exactement une table par schéma de relation de *S*

BD relationnelle (vue logique)

On abstrait encore en remplaçant les attributs *par une* numérotation *:* 1,2,3

schéma de relation (d'arité k) <=> **prédicat** (d'arité k)

ex:	Film [titre, directeur, acteur]	Film/3
	Programme [cinéma, titre, horaire]	Programme/3
	Lieu [cinéma, adresse, téléphone]	Lieu/3

schéma de BD *S* <=> ensemble de prédicats $\mathcal{P}$

domaine *dom* <=> ensemble de constantes C

- Ainsi, *(S, dom)* est vu comme un vocabulaire logique, et réciproquement
- Une BD sur *(S,dom)* est vue comme une base de faits sur le vocabulaire *(S,dom)*, et réciproquement

  1 ligne $(v_1 \ldots v_k)$ d'une table de schéma $R[A_1 \ldots A_k]$ <=> un fait $R(v_1 \ldots v_k)$

D'une BD à une base de faits, et réciproquement

Film

Titre	Directeur	Acteur
Pulp fiction	Q. Tarantino	J. Travolta
Pulp fiction	Q. Tarantino	Q. Tarantino
Pulp fiction	Q. Tarantino	U. Thurman
Grease	R. Kleiser	J. Travolta

Base de faits

{
Film(pf,qt,jt),
Film(pf,qt,qt),
Film(pf,qt,ut),
Film(g,rk,jt)
}

REQUÊTES DANS LE MODÈLE RELATIONNEL

- L'algèbre relationnel est un langage de requête basé sur un ensemble d'opérations : sélection, projection, union, différence, produit cartésien (et opérations dérivées : jointure, ...)
- SQL repose sur l'algèbre relationnel
- Formellement, une requête associe à une BD une table (qui liste les réponses à la requête)

« Trouver les films dans lesquels joue J. Travolta »

```
SELECT DISTINCT Film.titre
FROM Film
WHERE Film.Acteur = « J. travolta »
```

BD →

Titre
Pulp fiction
Grease

Requêtes en logique

- Toute requête de l'algèbre relationnel peut être vue comme une formule de la logique du premier ordre (« first-order query »)
- Les deux langages de requête ont la même expressivité (voir cours de M2)

```
SELECT Film.titre
FROM Film
WHERE Film.Acteur = « J. travolta »
```

$Q(x) = \exists y\ Film(x,y, jt)$
où x,y sont des variables
et jt une constante

Une requête (du premier ordre , « first-order query »)
notée $Q(x_1 \ldots x_k)$ est une formule logique
où $x_1 \ldots x_k$ sont exactement les variables libres, appelées variables réponses

Si k=0, Q() est une requête booléenne
« J. Travolta joue-t-il dans un film ?» $Q() = \exists x\ \exists y\ Film(x,y, jt)$

Quand on n'a pas besoin de mentionner explicitement les variables libres, on peut écrire juste Q au lieu de Q(...).

Rappel de l'exemple

Film

Titre	Directeur	Acteur
Pulp fiction	Q. Tarantino	J. Travolta
Pulp fiction	Q. Tarantino	Q. Tarantino
Pulp fiction	Q. Tarantino	U. Thurman
Grease	R. Kleiser	J. Travolta

Programme

Cinéma	Titre	Horaire
Diagonal	Pulp Fiction	01/03/2022 à 20h

Lieu

Cinéma	Adresse	Site web
Diagonal	...	...

Une classe de requêtes fondamentales

« Trouver les cinémas dans lesquels on passe un film de Tarantino et les titres de ces films »

```
SELECT Programme.Cinéma, Programme.Titre
FROM Film, Programme
WHERE
        Film.Directeur = « Q. Tarantino »
        AND
        Film.Titre = Programme.Titre
```

Vue logique :

$$Q(z,x) = \exists y\ \exists t\ (Film(x,qt,y) \wedge Programme(z,x,t))$$

Ces requêtes qui demandent de trouver un certain « motif » sont appelées requêtes conjonctives

Requêtes conjonctives (Conjunctive Queries)

Une requête conjonctive (**CQ**) $Q(x_1 \dots x_k)$ est de la forme $\exists x_{k+1},\dots,x_m\ A_1 \wedge \dots \wedge A_p$ où $A_1,\dots,A_p$ sont des atomes ayant pour variables $x_1,\dots,x_m$

Autrement dit, une requête conjonctive est une conjonction d'atomes quantifiée existentiellement (mais pas forcément close)

Notation simplifiée

$Q(x_1 \dots x_k) = \{ A_1, \dots, A_p \}$

Notation sous forme de règle

$answer(x_1 \dots x_k) \leftarrow A_1, \dots, A_p$ Notation Datalog classique

$A_1 \wedge \dots \wedge A_p \rightarrow answer\ (x_1 \dots x_k)$ Notation alternative

SQL

SELECT ... FROM ... WHERE *<conditions d'égalité>*

Basic SPARQL

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

UNION DE REQUÊTES CONJONCTIVES (UCQ)

Une union de requêtes conjonctives (**UCQ**) $Q(x_1 \ldots x_k)$
est une disjonction de requêtes conjonctives
ayant toutes $(x_1 \ldots x_k)$ comme liste de variables réponses

Remarque : on peut avoir besoin du prédicat = dans les requêtes conjonctives pour assurer qu'elles aient toutes les mêmes variables réponses

« Trouver les cinémas et titres de films au programme de ces cinémas tel que ce soient des films de Tarantino ou avec Travolta ou le film « The Chef »

$$
\begin{aligned}
Q(z,x) = \quad & (\exists y\ \exists t\ (\mathrm{Film}(x,qt,y) \land \mathrm{Programme}(z,x,t))) \lor \\
& (\exists y\ \exists t\ (\mathrm{Film}(x,y,jt) \land \mathrm{Programme}(z,x,t))) \lor \\
& (\exists t\ (\mathrm{Programme}(z,x,t) \land x = \text{« The Chef »}))
\end{aligned}
$$

SELECT Programme.Cinéma, Programme.Titre FROM ... WHERE ...
UNION
SELECT Programme.Cinéma, Programme.Titre FROM ... WHERE ...
UNION
SELECT Programme.Cinéma, Programme.Titre FROM ... WHERE ...

Sémantique des requêtes : qu'est-ce qu'une réponse ?

Commençons par les requêtes booléennes

Idée : une base de faits *F* répond oui à *Q* si *Q* est « vraie » dans *F*

Formellement :
une base de faits F peut-être vue comme une interprétation logique

- domaine : les constantes de F
- chaque constante s'interprète par elle-même
- chaque prédicat p s'interprète par l'ensemble des uplets (c_1 ... c_k) tels que p(c_1 ... c_k) ∈ F

Exemple

F

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

F vue comme une interprétation $\mathcal{I}$

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

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

On voit que $\mathcal{I}$ est un modèle de F

On l'appelera **le** modèle canonique de F

Remarque : $\mathcal{I}$ est une interprétation restreinte au *vocabulaire qui apparait dans F,* mais on peut l'étendre à tout le vocabulaire logique
(chaque prédicat hors de *F* ayant une interprétation vide)

Exemple :
$\mathcal{V} = (\mathcal{P}, \mathcal{C})$ avec $\mathcal{P} = \{ p/2, q/2, r/1, s/3 \}$ et $\mathcal{C} = \{a,b,c, \dots \}$

$D_{\mathcal{I}} = \mathcal{C}, \; r^{\mathcal{I}} = s^{\mathcal{I}} = \emptyset$

EXEMPLE

F

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

F vue comme une interprétation $\mathcal{I}$

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

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

$Q() = \exists x\ \exists y\ \exists z\ (p(x,y) \land p(y,z) \land q(z,x))$

(Le modèle canonique de) F est-il un modèle de Q ?

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

Deux façons de le prouver

Sémantique des requêtes : qu'est-ce qu'une réponse ?

Mais en général une requête a des variables libres (les variables réponses)

Pour obtenir une formule close (autrement dit, une requête booléenne), on remplace chaque variable libre par une constante :

$Q(x_1 \ldots x_k) \Rightarrow Q(x_1/c_1, \ldots, x_k/c_k)$
requête obtenue en remplaçant dans Q
chaque variable x_i par la constante c_i

Une réponse à $Q(x_1 \ldots x_k)$ sur une base de faits *F* est une liste $(c_1 \ldots c_k)$ de constantes telle que *F* est un modèle de $Q(x_1/c_1, \ldots, x_k/c_k)$.

L'ensemble des réponses à *Q* sur *F* est noté *Q(F)*.

Cas des requêtes booléennes

Que vaut *Q(F)* si *Q* est booléenne ?

Soit *F* n'est pas un modèle de *Q* : *Q(F)* = ø = {}

Soit *F* est un modèle de *Q* : *Q(F)* = { () }

On interprète ø comme la valeur faux et { () } comme la valeur vrai

Autrement dit : la réponse à *Q* est faux si *Q(F)* = ø, sinon vrai

Remarque : les requêtes booléennes ne sont pas directement proposées en SQL (à la différence de SPARQL avec ses requêtes ASK)

Réponses à une requête conjonctive

F

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

$Q() = \exists x\, \exists y\, \exists z\, (p(x,y) \wedge p(y,z) \wedge q(z,x))$

$Q() = \{\ p(x,y), p(y,z), q(z,x)\ \}$

F est-elle un modèle de *Q* ?

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

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

Deux façons d'instancier les variables de Q
par des constantes de F
qui prouvent que F satisfait Q

Un **homomorphisme** **h** de *Q* dans *F* est une application
de l'ensemble des variables de *Q* dans l'ensemble des termes de *F*
telle que $h(Q) \subseteq F$

où *h(Q)* désigne l'ensemble d'atomes obtenu à partir de *Q*
en substituant chaque variable *x* par *h(x)*

Réponses à une requête conjonctive

Soit $Q(x_1,\ldots,x_k)$ une requête conjonctive

Le k-uplet de constantes $(c_1 , \ldots, c_k)$ est une réponse à Q dans F

ssi

(Le modèle canonique de) F est un modèle de $Q(x_1/c_1, \ldots, x_k/c_k)$

[par définition de la notion de réponse]

ssi

il existe un homomorphisme de Q dans F qui envoie chaque x_i sur c_i

Si k = 0 :
La réponse à Q dans F est *vrai (oui)*
ssi il existe un homomorphisme de Q dans F

Exemple

F

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

$Q_1() \quad = \{ p(x,y), p(y,z), q(z,x) \}$
$Q_2(x) \quad = \{ p(x,y), p(y,z), q(z,x) \}$
$Q_3(x,y,z) = \{ p(x,y), p(y,z), q(z,x) \}$

Homomorphismes de ces requêtes dans F

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

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

On obtient donc :
$Q_1(F) = \{()\}$
$Q_2(F) = \{ (b) \}$
$Q_3(F) = \{ (b,a,c), (b,a,b)\}$

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

Monde ouvert / monde clos

- Hypothèse du monde clos (bases de données)

  La base de faits décrit un monde complètement connu

  (tous les faits qui ne sont pas présents sont faux)

  cela correspond à la notion d'interprétation logique

  Pour répondre à une requête, on considère *F* seulement

- Hypothèse du monde ouvert (web sémantique, bases de connaissances)

  La base de faits décrit un monde partiellement connu

  (les faits qui ne sont pas présents peuvent être vrais ou faux)

  Pour répondre à une requête, on considère toutes les bases de faits possibles qui *contiennent F* (= toutes les extensions de *F*)
  Cela correspond à la notion de conséquence logique

MONDE OUVERT / MONDE CLOS

- Hypothèse du monde clos

  La réponse à Q (booléenne) sur F est oui si F « *est un modèle » de* Q,

- Hypothèse du monde ouvert

  La réponse à Q (booléenne) sur F est oui si
  tout modèle de F *est un modèle de* Q (notation : $F \vDash Q$)
  ici, on voit F comme une interprétation, donc cela revient à dire :
  toute base de faits qui contient F « est un modèle » de Q

  (hum ... petit bémol :
  les bases de faits correspondent à des interprétations finies
  or, certaines formules n'ont que des modèles infinis)

Monde ouvert / monde clos

Soit $Q(x_1 \ldots x_k)$ une requête, F une base de faits, $(c_1 \ldots c_k)$ une liste de constantes

- Réponse à une requête avec hypothèse du monde clos

  $(c_1 \ldots c_k)$ est une réponse à Q sur F si F est un modèle de $Q(x_1/c_1, \ldots, x_k/c_k)$.

- Hypothèse du monde ouvert

  $(c_1 \ldots c_k)$ est une réponse à Q sur F si $F \models Q(x_1/c_1, \ldots, x_k/c_k)$.

  Pour marquer la différence, on parle ici de « réponse certaine »

Bonne nouvelle : pour les CQ (et UCQ), ça ne fait aucune différence !

EXEMPLE : DIFFÉRENCE MONDE OUVERT/MONDE CLOS

F = { aPourEnfant(Jules,Chloé), Fille(Chloé) }

« Jules n'a-t-il que des filles ? »

$Q() = \forall x\,(aPourEnfant(Jules,x) \rightarrow Fille(x))$

$\equiv \neg \exists x\,(aPourEnfant(Jules,x) \wedge \neg Fille(x))$

Q est satisfaite dans F (F est un modèle de Q) mais
Q n'est pas conséquence de F ($F \nvDash Q$)

« Jules a-t-il un enfant qui n'est pas une fille ?»
$Q() = \exists x\,(aPourEnfant(Jules,x) \wedge \neg Fille(x))$

Ici les deux notions coincident car F n'est pas un modèle de Q

« Jules a-t-il un enfant qui est une fille ? »
$Q() = \exists x\,(aPourEnfant(Jules,x) \wedge Fille(x))$

Ici aussi les deux notions coincident : pour tout F' avec $F \subseteq F'$, il y a un homomorphisme de Q dans F', donc $F \vDash Q$

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 est-ce vrai ?

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^I contient forcément (a,b) et (b,c)
q^I contient forcément c

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

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

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 :**
c'est l'interprétation qui lui est associée, qu'on appelle son « **modèle canonique** »

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

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

$\mathcal{M}$: $D_{\mathcal{M}} = C$
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 q'un modèle d'une CQ Booléenne ?

$q() = \exists x\, \exists y\, \exists z\, (\, p(x,y) \wedge p(y,z) \wedge r(x,z,a)\,)$

$\mathcal{I}$:

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

$\mathcal{I}$ 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

Soit une base de faits F et une **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 et soit M le modèle canonique de F
  h définit une application des termes de q *dans* D_M
  qui prouve que M est un modèle de q
  Comme M est l'intersection de tous les modèles de F,
  tout modèle de F est un modèle de q, c'est-à-dire $F \models q$

En résumé :

- Toute BD relationnelle peut être vue comme une base de faits, et réciproquement
- L'algèbre relationnel (base théorique de SQL) a la même expressivité que les requêtes du premier ordre
- Quelques classes de requêtes fondamentales :
  - requêtes conjonctives (CQ)
  - unions de requêtes conjonctives (UCQ)
- Une réponse à $Q(x_1 \ldots x_k)$ sur une base de faits F est une liste $(c_1 \ldots c_k)$ de constantes telle que F est un modèle de $Q(x_1/c_1, \ldots, x_k/c_k)$.
  On voit ici F comme une interprétation logique
- Ceci correspond à l'hypothèse du monde clos. En monde ouvert, on aurait la notion de réponse certaine : $F \vDash Q(x_1/c_1, \ldots, x_k/c_k)$.
- Pour les (U)CQs, cela ne fait pas de différence !
- Les réponses à une CQ Q sur F correspondent à des homomorphismes de Q dans F

EXEMPLE : PISTES CYCLABLES

BF

Direct(A,B)

Direct(B,C)

Direct(C,D)

Direct(D,B)

Quelques requêtes du premier ordre (des CQs ici)

Q() = Direct(A,C) ?

Q(x,y) = Direct(x,B) ∧ Direct(B,y) ?

Comment demander s'il y a un chemin de A à C ?

Impossible à formuler par une requête du 1er ordre !

Requête Datalog : ensemble de règles

∀x∀y (Direct(x,y) → Chemin(x,y)**)**

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

Chemin(A,C) → answer()

Les faits de prédicat answer donnent les réponses à la requête (ici une seule réponse car la requête est booléenne)

DATALOG: RÈGLES CONJONCTIVES POSITIVES

Règle Datalog : $\forall x_1 \ldots \forall x_n (H \rightarrow C)$ où :

- H est une conjonction d'atomes et C est un atome
- $x_1 \ldots x_n$ sont les variables de H
- **toutes** les variables de **C** apparaissent dans **H**

Hypothèse → Conclusion
Corps → Tête

Tête ← Corps

$\forall x \forall y \forall z$ ((aParent(x,z) $\wedge$ aParent(y,z)) → frèreOuSoeur(x,y))

Notation simplifiée (on omet les quantificateurs) :

aParent(x,z) $\wedge$ aParent(y,z) → frèreOuSoeur(x,y)

CHAÎNAGE AVANT

F

Direct(A,B)

Direct(B,C)

Direct(C,D)

Direct(D,B)

$\mathcal{R}$

R1 : Direct(x,y) → Chemin(x,y)

R2 : Direct(x,y) ∧ Chemin(y,z) → Chemin(x,z)

R3 : Chemin(A,C) → answer()

Une règle R : H → C est applicable à F s'il existe un homomorphisme h de H dans F

- Cette application de R à F est utile si h(C) ∉ F
- Une règle R : H → C est satisfaite dans F si pour tout homomorphisme h de H dans F, on a h(C) ∈ F (autrement dit, aucune application de R à F n'est utile)
- Appliquer R à F consiste à ajouter h(C) à F
- F est saturée (par rapport à $\mathcal{R}$) si toute règle R ∈ $\mathcal{R}$ est satisfaite dans F

Langage de requêtes Datalog (positif)

Idée : Ajouter la **récursivité** aux requêtes du premier ordre

Inspiré par Prolog (ancêtre de nombreux langages de programmation logique)

Datalog positif = UCQ + récursivité

- Une requête (ou « programme ») Datalog a la forme $(\mathcal{R}, ans)$ où :
  - $\mathcal{R}$ est un ensemble de règles positives conjonctives
  - ans est un prédicat spécial apparaissant seulement en tête de règle et n'appartenant pas au vocabulaire de la base de données

Exemple : $(\mathcal{R}$, answer)

$$\text{avec } \mathcal{R} = \left\{ \begin{array}{l} \text{Direct}(x,y) \rightarrow \text{Chemin}(x,y) \\ \text{Direct}(x,y) \land \text{Chemin}(y,z) \rightarrow \text{Chemin}(x,z) \\ \text{Chemin}(A,C) \rightarrow \text{answer}() \end{array} \right.$$

EXEMPLE

Lignes

Ligne	Station 1	Station 2
4	St Germain	Odéon
4	Odéon	St Germain
10	Odéon	Cluny
…	…	…

"Trouver les stations directement connectées à Odéon"

ans(y) ← Lignes(x, Odéon, y)

ans(y) ← Lignes (x, y, Odéon)
ans(y) ← Lignes (x, Odéon, y)

2 solutions
selon que les lignes
soient bidirectionnelles
ou pas

"Trouver les stations atteignables à partir d'Odéon, directement ou indirectement"

connecté (y, z) ← Lignes (x, y, z)
connecté (x, z) ← connecté (x, y), connecté (y, z)
ans(x) ← connecté (x, Odéon)

QUEL RAPPORT AVEC LES CQ ET UCQ ?

Une CQ $\mathbf{q(x_1 \ldots x_k) = \exists x_{k+1},\ldots,x_m\ A_1 \land \ldots \land A_p}$ peut être vue comme une requête Datalog composée d'une seule règle :

$$\mathbf{A_1 \land \ldots \land A_p \rightarrow ans(x_1 \ldots x_k)}$$

« Trouver les cinémas dans lesquels on passe un film de Tarantino »

Q(z) = ∃x ∃y ∃t (Film(x,qt,y) ∧ Programme(z,x,t))

Q(z) = { Film(x,qt,y), Programme(z,x,t) }

En requête Datalog :

Film(x,qt,y) ∧ Programme(z,x,t) → ans(z)

QUEL RAPPORT AVEC LES CQ ET UCQ ?

Une UCQ **$q(x_1 \ldots x_k) = Q_1(x_1 \ldots x_k) \vee \ldots \vee Q_n (x_1 \ldots x_k)$** où chaque Q_i est une CQ peut être vue comme une requête Datalog composé de n règles :

$$
\begin{aligned}
&Q_1 \rightarrow ans(x_1 \ldots x_k)\\
&\ldots\\
&Q_n \rightarrow ans(x_1 \ldots x_k)
\end{aligned}
$$

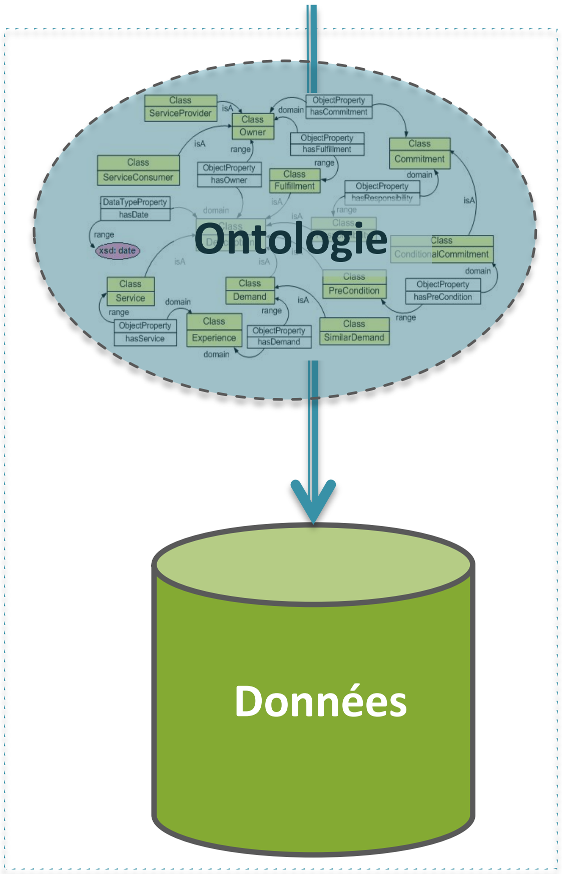
« Trouver les cinémas et titres de films au programme de ces cinémas tel que ce soient des films de Tarantino ou avec Travolta ou le film « The Chef »

$$
\begin{aligned}
Q(z,x) = \; & (\exists y \, \exists t \, (Film(x,qt,y) \wedge Programme(z,x,t))) \vee \\
& (\exists y \, \exists t \, (Film(x,y,jt) \wedge Programme(z,x,t))) \vee \\
& (\exists t \, (Programme(z,x,t) \wedge x = \text{« The Chef »}))
\end{aligned}
$$

$$
\begin{aligned}
&Film(x,qt,y) \wedge Programme(z,x,t) && \rightarrow answer(z,x)\\
&Film(x,y,jt) \wedge Programme(z,x,t) && \rightarrow answer(z,x)\\
&Programme(z, \text{« The Chef »}),t) && \rightarrow answer(z, \text{« The Chef »})
\end{aligned}
$$

QUEL RAPPORT AVEC LES BASES DE CONNAISSANCES ?

Requête



Base de connaissances

Requête du premier ordre
(par exemple une CQ ou une UCQ)

Ensemble de formules logiques
(par exemple des règles conjonctives positives)

Ensemble de faits (atomes instanciés)

Idée :
la réponse a une requête booléenne est oui si cette requête est **conséquence logique de la base de connaissances**

Deux visions des règles Datalog

- **Datalog comme langage de requêtes** :

  Une requête $q = (\mathcal{R}, ans/k)$ est posée sur une BD (ou: base de faits) F

  L'ensemble des réponses à q est l'ensemble des (c1...ck) tels que ans(c1...ck) est conséquence logique de $F \cup \mathcal{R}$

- **Datalog comme langage ontologique :**

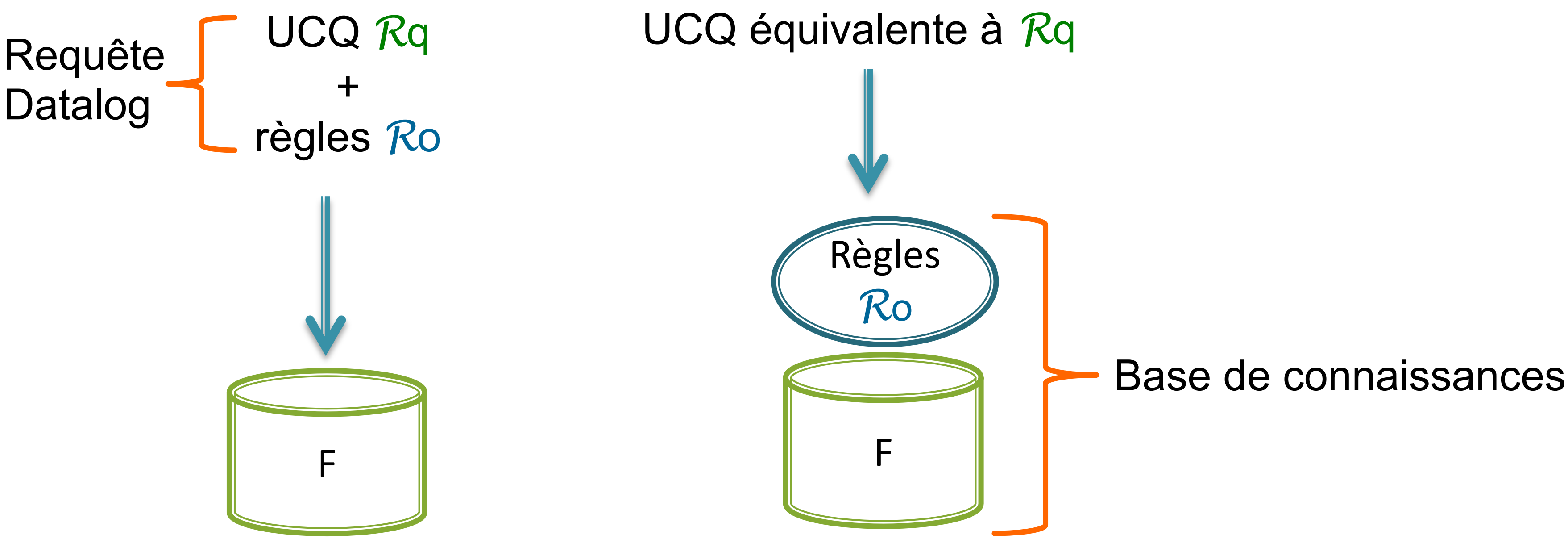
  $\mathcal{R}$ définit une ontologie
  Une requête CQ ou UCQ $q(x_1 \ldots x_k)$ est posée sur une base de connaissances composée d'une base de faits F et de $\mathcal{R}$

  L'ensemble des réponses à q est l'ensemble des (c1...ck) tels que Q(x1/c1...xk/ck) est conséquence logique de $F \cup \mathcal{R}$

Requête Datalog vue comme UCQ + ontologie

Toute requête Datalog ($\mathcal{R}$, ans) peut être décomposée en :

- $\mathcal{R}$q l'ensemble des règles dont la tête est un atome sur ans
- $\mathcal{R}$o : l'ensemble des autres règles



Les réponses à ($\mathcal{R}$q ∪ $\mathcal{R}$o, ans) sur la base de faits F sont exactement les réponses à l'UCQ équivalente à $\mathcal{R}$q sur la base de connaissances (F, $\mathcal{R}$o)

Datalog comme langage ontologique

Avec un ensemble de règles Datalog, on peut décrire entre autres :

- une hiérarchie de classes (ou : concepts)
- une hiérarchie de relations (pas seulement binaires)
- les signatures de ces relations, c'est-à-dire la classe associée à chaque argument de la relation (« domain » et « range » en RDFS)
- les propriétés de ces relations :
  - réflexivité, symétrie, transitivité, ...
- le fait qu'une relation binaire est l'inverse d'une autre

etc.

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 : $(a,b) \in p^I$ et $(b,c) \in p^I$
- 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 ? (non)

$I = (D, .^I)$ 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 ? (oui)

$I = (D, .^I)$ 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 ?

(oui, c'est même un modèle minimal)

PROPRIÉTÉS DE LA BASE DE FAITS SATURÉE

Soit une base de connaissances $K = (F, \mathcal{R})$

- Le chaînage avant calcule une base de faits saturée (F*)
- F* est unique
- F* est le plus petit modèle de $F \cup \mathcal{R}$
  1. C'est un modèle de F car $F \subseteq F^*$
  2. C'est un modèle de $\mathcal{R}$: toute règle $R \in \mathcal{R}$ est satisfaite
     1+2 : c'est un modèle de $F \cup \mathcal{R}$
  3. C'est un plus petit modèle de $F \cup \mathcal{R}$: si on enlève un seul fait, ce n'est plus un modèle de $F \cup \mathcal{R}$
- F* contient exactement l'ensemble des faits qui sont conséquence logique de $F \cup \mathcal{R}$:

  pour tout fait f, $f \in F^*$ ssi $F \cup \mathcal{R} \models f$

- L'ensemble de réponses à une requête Datalog $(\mathcal{R}, ans_{/k})$ sur une base de faits F est l'ensemble des k-uplets $(c_1 \ldots c_k)$ tels que $ans(c_1 \ldots c_k) \in F^*$