Vue logique des bases de données (BD) relationnelles

HAI824 – Traitement sémantique des données

ML Mugnier

Rappels de logique du premier ordre

- Cette logique décrit des **objets** et les **relations** entre ces objets

- Les objets sont appelés des termes : variables ou constantes
  (pas de fonctions ici)

- Les relations sont appelées des prédicats
  Tout prédicat a une arité (nombre d'arguments, qui est fixe)

- Atome $\mathbf{p(t_1 \ldots t_k)}$ ["p-atome" : atome de prédicat p]
  où **p** est un prédicat (ou relation)
  les $\mathbf{t_i}$ sont des termes
- Littéral : atome ou négation d'un atome

- Les variables sont quantifiées universellement $\forall$
  ou existentiellement $\exists$

Formules construites sur un vocabulaire

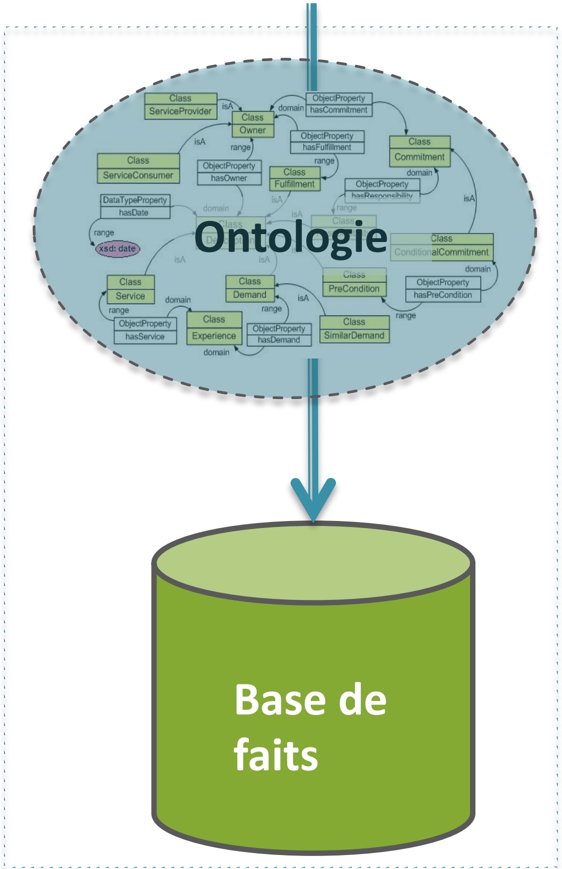
- Vocabulaire logique : *V = (P, C)*
  où *P* est un ensemble de prédicats
  C est un ensemble de constantes
- Terme sur *V* : constante $c \in C$ ou variable
- Formule sur *V* : elle se définit par induction
  - $p(t_1 \ldots t_k)$ où $p \in P$ et chaque t_i est un terme sur *V* — "formule atomique"

  ou
  - $\neg A$, $(A \land B)$, $(A \lor B)$, $(A \longrightarrow B)$, $(A \longleftrightarrow B)$, $\exists x\, A$, $\forall x\, A$ — "formule complexe"
    où A et B sont des formules sur *V*

- Variable libre : l'une de ses occurrences n'est pas dans la portée d'un quantificateur
- Formule close (fermée) : sans variable libre

[si une formule n'est pas close, la notion d'interprétation ne suffit pas à lui donner une valeur de vérité ; il faut en plus assigner à chaque variable libre un élément du domaine de l'interprétation]

Bases de connaissances



Base de connaissances

- Ontologie : ensemble fini de formules closes
- 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 (qui peut être 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 *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

Ensemble de faits :

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

- À toute table on associe un ensemble de faits ayant tous le même prédicat
- À une BD on associe la base de faits formée de l’union des ensembles de faits associés à ses tables

Réciproquement, à toute base de faits on associe une BD, avec une table par prédicat

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

[Cette table des réponses peut être matérialisée sous forme de vue]

REQUÊTES EN LOGIQUE

- Toute requête de l'algèbre relationnel peut être vue comme une formule logique (« 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) est une formule logique $Q(x_1 \ldots x_k)$
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 écrit 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 »

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

Vue logique :

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

Ces requêtes qui demandent de trouver un certain « motif » (« pattern ») sont appelées **requêtes conjonctives**

REQUÊTES CONJONCTIVES (CONJUNCTIVE QUERIES)

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

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

Notation simplifiée

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

Notation sous forme de règle

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

$A_1 \wedge \ldots \wedge A_p \rightarrow answer\ (x_1 \ldots x_k)$ 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 pour variables libres $x_1 \ldots x_k$

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

« 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 »

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

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

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

Formellement : une base de faits est vue comme une interprétation logique

- domaine : les constantes de la base de faits
- 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) est un fait

On peut donc dire qu'une formule close est vraie ou fausse dans *F* autrement dit, que *F* est un modèle ou non de cette formule

(on dit aussi : *F* satisfait cette formule)

$Q = \exists x\ \exists y\ \exists t\ (Film(x,qt,y) \land Programme(Diago,x,t))$
est satisfaite par
F = {Film(pf,qt,jt), Programme(Diago,pf, « 05-03-2023-15h »), …}

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

$Q(z) = \exists x\, \exists y\, \exists t\, (Film(x,qt,y) \wedge Programme(z,x,t))$
Par la substitution z/Diago, on obtient la requête booléenne :
$Q() = \exists x\, \exists y\, \exists t\, (Film(x,qt,y) \wedge Programme(Diago,x,t))$

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)

$Q() = \exists x \exists y \, Film(x,y, jt)$

```
SELECT
CASE WHEN EXISTS
(SELECT * FROM Film WHERE Film.Acteur = « J. Travolta » )
THEN TRUE
   ELSE FALSE
```

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 $(a_1, \ldots, a_k)$ est une réponse à Q dans F

ssi

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 a_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)

$Q_1()$ = { p(x,y), p(y,z), q(z,x) }
$Q_2(x)$ = { 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 : $F \models Q$)

  ici, on voit F comme une interprétation, donc cela revient à dire :
  toute base de faits qui contient F est un modèle de Q

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 alors 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 ? »

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

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

Q est satisfaite dans F (F est un modèle de Q) mais
Q n'est pas conséquence de F ($F \not\models Q$)

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

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

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

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

EXERCICE 1 (VUE LOGIQUE D'UNE BD RELATIONNELLE)

On considère une BD relationnelle qui gère des abonnés, qui peuvent avoir des cartes d'accès, en cours de validité ou pas. Le schéma de la base est le suivant :

Coords [id_abonné, nom, prénom, date_naissance, ville]
Cartes [id_abonné, id_carte, validité]

Pour trouver les dates de naissance de tous les abonnés de Montpellier qui ont une carte d'accès en cours de validité, on fait la requête SQL suivante :

SELECT DISTINCT Coords.date_naissance
FROM Coords, Cartes
WHERE Coords.ville = MPL **AND** Cartes.validité = true
AND Coords.id_abonné = Cartes.id_abonné

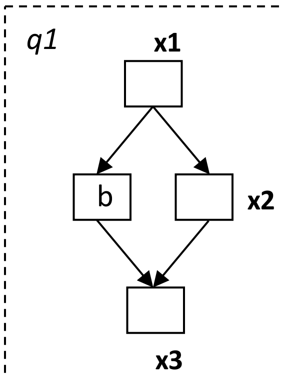
Soit la base de données suivante :

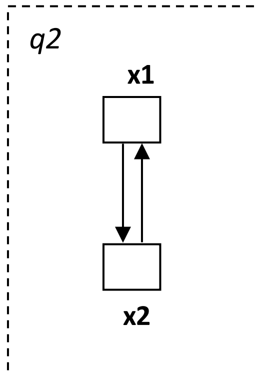
Coords = [[1, N1,P1,D1,MPL],[2,N2,P2,D2,MPL],[3,N3,P3,D3,MPL],[4,N4,P4,D4,MRS]]
Cartes = [[1,401,false],[1,502,true],[1,503,true],[2,404,false],[4,509,true]]

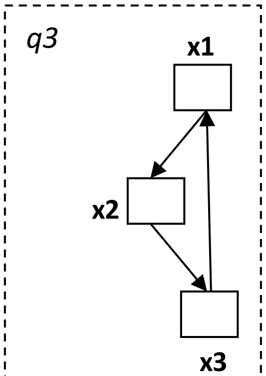
1. Traduire : le schéma en un ensemble de prédicats
la requête en une requête conjonctive
la base de données en une base de faits

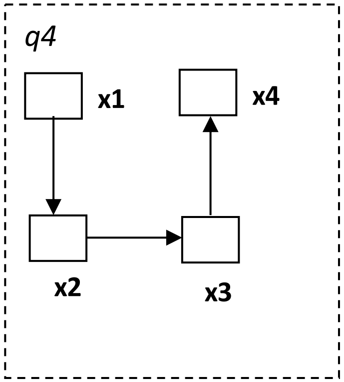
2. Déterminer les réponses à la requête et les justifier par des homomorphismes

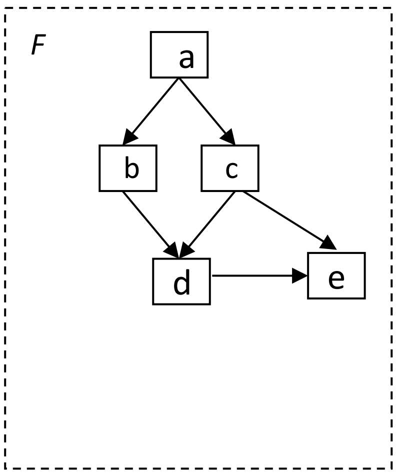
EXERCICE 2 (REQUÊTES CONJONCTIVES)











Ces graphes représentent des CQ (q_i) où les variables réponses sont x_1 et x_2 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.

Exercice 3 (inclusion de requêtes conjonctives)

Etant données deux requêtes conjonctives booléennes, Q1 et Q2, on dit que Q1 est **incluse** dans Q2 (notation Q1 $\sqsubseteq$ Q2) si l'ensemble des bases de faits qui répondent oui à Q1 est inclus dans l'ensemble des bases de faits qui répondent oui à Q2.

Utiliser les notions vues dans ce cours pour prouver que :

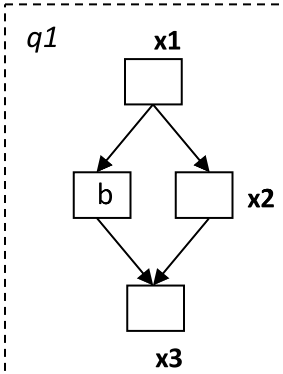
Q1 $\sqsubseteq$ Q2

ssi

il existe un homomorphisme de Q2 dans Q1

Question subsidiaire : pour des requêtes quelconques, Q1 est incluse dans Q2 si pour toute base de faits, l'ensemble des réponses à Q1 est inclus dans l'ensemble des réponses à Q2. Comment étendre le résultat précédent ?

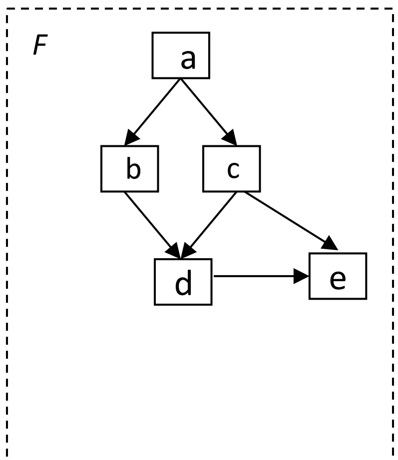
Correction Ex. 2



$q1(x1,x2) = \exists x3\ (p(x1,b) \wedge p(x1,x2) \wedge p(b,x3) \wedge p(x2,x3))$

ce qui correspond à l'ensemble d'atomes :
{ p(x1,b), p(x1,x2), p(b,x3), p(x2,x3) }

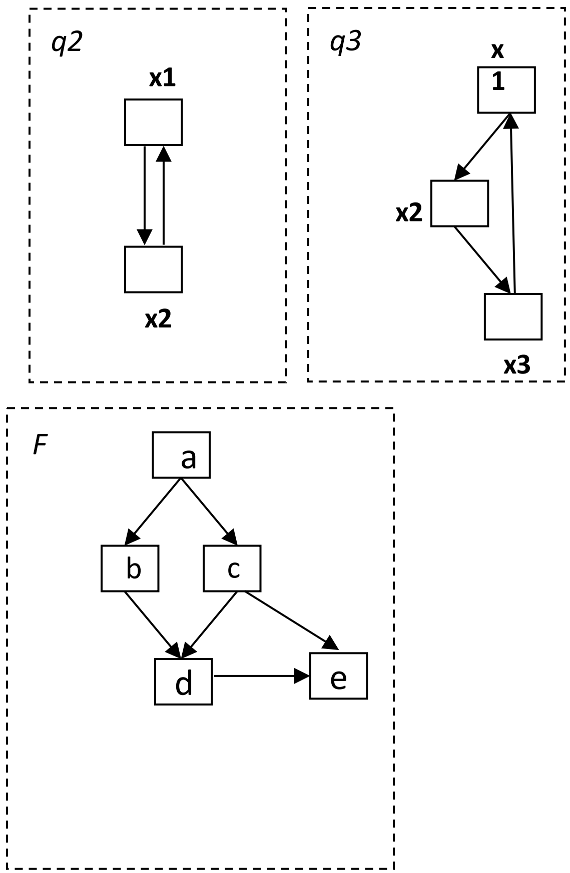
F = { p(a,b), p(a,c), p(b,d), p(c,d), p(c,e), p(d,e) }



h1 : q1 → F	h2 : q1 → F
$x1 \mapsto a$	$x1 \mapsto a$
$x2 \mapsto c$	$x2 \mapsto b$
$x3 \mapsto d$	$x3 \mapsto d$

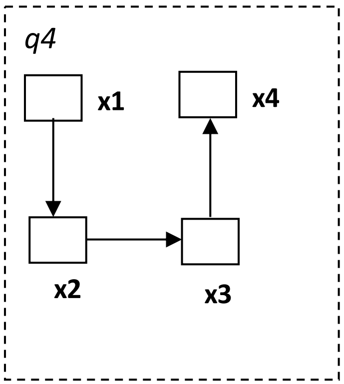
q1(F) = { (a,c), (a,b) }

Correction Ex. 2



$q2(F) = \emptyset$
$q3(F) = \emptyset$

CORRECTION EX. 2



h1 : q4 → F
x1 ↦ a
x2 ↦ b
x3 ↦ d
x4 ↦ e

h2 : q4 → F
x1 ↦ a
x2 ↦ c
x3 ↦ d
x4 ↦ e

q4(F) = { (a,b), (a,c) }