Logique et bases de données

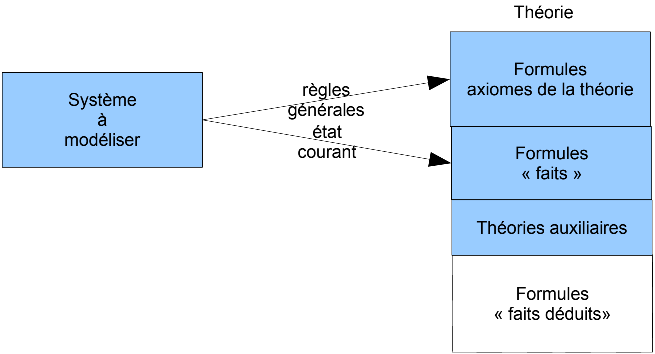
G. Falquet

CUI - UNIGE

21 décembre 2018

Principes de modélisation logique axiomatique

1. Définir les vocabulaire (prédicats, fonctions, constantes)
2. Définir les axiomes



Bases de données et logiques

Voir/interpréter les données à travers la logique

- approche déductive
- approche orientée modèle
  - Une interpréation d'un langage logique (sans fonctions) est
    - un domaine D
    - une fonction d'interprétation des constante
    - pour chaque prédicat n-aire une relation n-aire sur D
  - Principe : les données forment l'interprétation des prédicats.

Données comme interprétations

Il faut représenter des relations

- Tableur
  - Chaque relation est un ensemble de lignes et de colonnes
- Base de donnée relationnelle
  - Chaque table de la base est une relation
- Programme Scala/Swift
  - class P(p1 :T1, p2 :T2, ...)
  - val relP = Set(P(t1,t2), P(u1,u2), P(...), ...)

Le domaine est l'union des valeurs des types de données

Exemple

Vocabulaire : personne(2), parent(2), ...

Base de données B_{gen} :

table Personne

nom	naissance
albert	1888
claudia	1907
john	1927
ida	1934

table Parent

parent	enfant
claudia	john
albert	john
john	anna

Interrogation dans les données

But : retrouver les données satisfaisant un critère

Expression : une formule ouverte sur le vocabulaire de la base de données (+ les prédicats de la théorie des nombres)

Résultat : toutes les valuations des variables libres qui rendent la formule vraie

Exemples

1. trouver les personnes nées en 2001

$$personne(x, 2001)$$

2. les personnes donc tous les parents sont nés avant 1900

$$personne(x, y) \land \forall x' \forall y' \, ((parent(x', x) \land personne(x', y')) \rightarrow y' < 1900)$$

Intégrité des données

Pour que les données représentent bien la réalité, elles doivent satisfaire des conditions appelées contraintes d'intégrité ou invariants du système.
Les contraintes sont représentées par des formules fermées $\phi_1, \ldots, \phi_n$
L'interprétation de chaque ϕ_i selon la base de données B doit être vraie, i.e.

$$B \models \phi_1, \ldots, \phi_n$$

B doit être un modèle des ϕ_i .

Exemple

La base B_{gen} satisfait les contraintes

1. La relation parent doit faire référence à des personnes

$$\forall x \forall x' \left(parent(x, x') \rightarrow \exists n \exists n' \left(personne(x, n) \land personne(x', n') \right) \right)$$

2. Un parent est forcément né avant ses enfants

$$\forall p \forall n \forall p' \forall n' ((personne(p, n) \land parent(p, p') \land personne(p', n')) \rightarrow n < n')$$

3. Une personne n'a qu'une date de naissance

$$\forall p \forall n \forall n' ((personne(p, n) \land personne(p, n')) \rightarrow n = n')$$

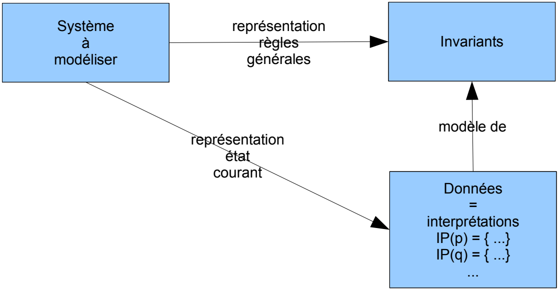
Exemple (suite)

La base B_{gen} ne satisfait pas

toute personne est soit un parent soit un enfant

$$\forall p \forall n \left(personne(p, n) \rightarrow \exists p' (parent(p, p') \vee parent(p', p)) \right)$$

Résumé du principe de représentation



Dynamique de la base de données

- La réalité se modifie au cours du temps (le changement est l'essence même du monde[1])
- Une base de donnée doit donc évoluer, par ajout, suppression ou modification dans les relations pour continuer à représenter la réalité.
- Les modifications acceptables de la base sont celles qui font passer d'un modèle B des invariants à un autre modèle B'.

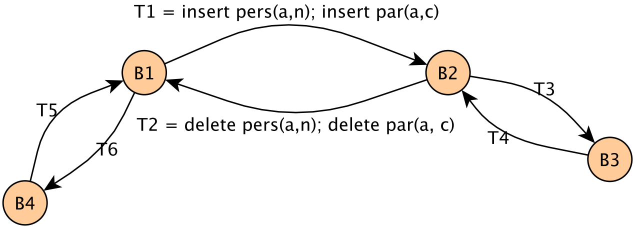
1. Yi Jing (Traité canonique des mutations), env. 750 av. J.C.

Exemple

Pour repecter l'invariant

$$\forall p \forall n \left(personne(p, n) \rightarrow \exists p' (parent(p, p') \vee parent(p', p)\right)$$

on ne peut pas ajouter une personne dans la base sans ajouter en même temps un parent ou un enfant de



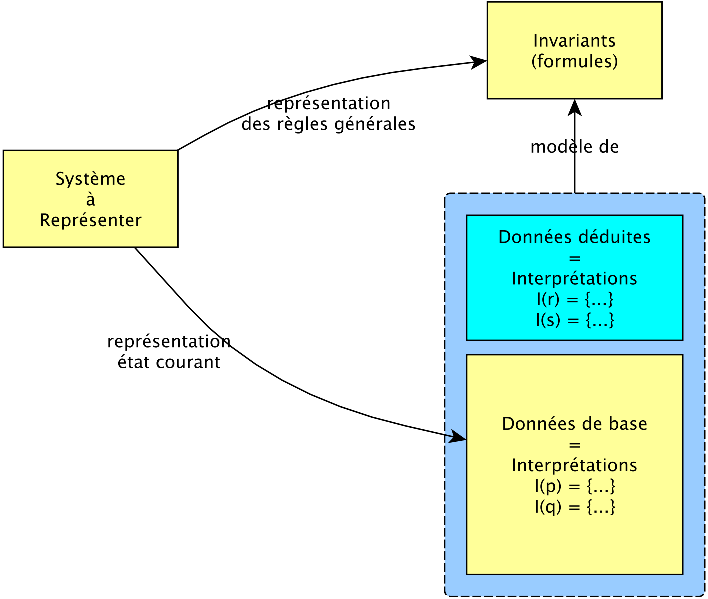
celle-ci

Les transitions d'un modèle à un autre s'appellent des transactions

Approche déductive

Principe :

- Les données forment une interprétation qui n'est pas un modèle
- Les règles générales du domaine servent de règles de déduction
- On cherche un modèle des règles du domaine qui contient l'interprétation de départ



Invariants
(formules)
représentation
des règles générales
modèle de
Système
à
Représenter
Données déduites
=
Interprétations
I(r) = {...}
I(s) = {...}
représentation
état courant
Données de base
=
Interprétations
I(p) = {...}
I(q) = {...}

Datalog

Pour rendre la déduction calculable on limite l'expression des règles à des clauses de Horn sans fonctions

$$tête \leftarrow corps$$

$$q(...) \leftarrow p_1(...) \wedge p_2(...) \wedge \cdots \wedge p_n(...)$$

Toutes les variables sont implicitement quantifiée universellement, il n'y a pas de négations

$$parent(x, y) \leftarrow pere(x, y)$$

$$parent(x, y) \leftarrow mere(x, y)$$

$$grandParent(x, z) \leftarrow parent(x, y) \wedge parent(y, z)$$

Calcul d'un modèle

- Pour chaque règle
  - chercher les valeurs des variables qui rendent vrai le corps
  - remplacer les variables par leur valeur dans la tête
  - ajouter à l'interprétation (relation) de la tête
- Répéter tant que de nouveaux faits sont produits

Propriétés

- Ce processus se termine toujours
  - on ne crée pas de nouvelles constantes
- Il calcule l'unique modèle minimal qui
  - contient les données de départ
  - satisfait les règles
- Sur une base de données relationnelle il peut être réalisé avec des opérations de base
  - sélection, projection, jointure, union

Exemple

```
path(X, Y) :- segment(X, Y).
path(X, Y) :- path(X, Z) , path(Z, Y).
```

Segment :

From	To
a	b
a	c
b	d
k	h
d	k
m	b

Pourquoi un modèle minimal ?

```
grandParent(X,Y) :- parent(X,Z), parent(Z,Y).
parent(a, b). parent(b, c), parent(c, d)
```

Modèle minimal : grandParent $= \{(a, c), (c, d)\}$
Un modèle non-minimal : grandParent $=$
$\{(a, c), (c, d), (c, a), (a, a), (d, a)\}$

- satisfait $\forall x \forall y \forall z(parent(x, z) \wedge parent(z, y) \rightarrow grandParent(x, z)$
- contient des faits qui ne sont pas des conséquences des formules de départ

Limites

- impossible de calculer des fonctions agrégées (somme, moyenne)
  - il faut ajouter la introduire la théorie des ensembles
- syntaxiquement complexe
  - autant de variables que de paramètres pour chaque prédicat
  - notation positionnelle (se souvenir de la signification du k^e paramètre)
  - $\Rightarrow$ invention des langages propres aux bases de données (SQL)