Systèmes à base de règles

en logique du 1er ordre

Cours de HMIN107 (IA)
ML Mugnier

1

Rappels des cours précédents (logique d'ordre 0)

- Base de connaissances : K = (BF, BR)
- Mécanismes de chaînage avant et arrière

- Règles conjonctives positives
  - Les mécanismes de chaînage avant / arrière sont complets
  - Le problème $K \models A$? où A est un atome est polynomial (linéaire)

- Règles conjonctives (avec négation classique)
  - aussi expressives que toute la logique des propositions
  - les mécanismes de chaînage avant / arrière ne sont pas complets

- Règles conjonctives avec négation du monde clos
  - Restriction aux ensembles semi-positifs ou + généralement stratifiés
  - Même complexité du chaînage avant que dans le cas positif

Rappels de logique du premier ordre

- Cette logique décrit des **objets** et les **relations** entre ces objets
- Les objets sont appelés 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 **$p(e_1 \ldots e_k)$**
  où **p** est un prédicat (ou relation)
  les **e_i** sont des termes
- Littéral : atome ou négation d'un atome
- Les variables sont quantifiées universellement ou existentiellement
- On ne raisonne que sur des formules fermées : toute variable est dans la portée d'un quantificateur

Règles (conjonctives) positives et faits

Règle : **$\forall x_1 \ldots \forall x_n$ (H → 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**

∀x∀y ((Pays(x) ∧ FaitPartie(x, UE) ∧ PermisValable (y,x)) → PermisValable (y, F))

Notation simplifiée (on omet les quantificateurs) :

Pays(x) ∧ FaitPartie(x, UE) ∧ PermisValable (y,x) → PermisValable (y, F)

Un fait correspond à une règle à hypothèse vide :
c'est donc un atome instancié (sans variables)

Pays(Danemark), FaitPartie(Danemark, UE), ...

Exemple (permis de conduire)   K = (BF,BR)

F1 : Ville(Copenhague)
F2 : Pays(Danemark)
F3 : FaitPartie(Copenhague, Danemark)
F4 : FaitPartie(Danemark,UE)
F5 : LieuObtentionPermis(Ingrid, Copenhague)
F6 : Pays(F)
F7 : FaitPartie(F, UE)

R1 : Ville(x1) ∧ Pays(y1) ∧ FaitPartie(x1,y1) ∧LieuObtentionPermis(z1,x1) → PermisValable(z1,y1)

"Si z1 obtient un permis (de conduire) dans une ville qui fait partie d'un certain pays, alors z1 a un permis valable dans ce pays"

R2 : Pays(x2) ∧ FaitPartie(x2, UE) ∧ PermisValable (y2,x2) → PermisValable (y2,F)

"Les permis valables dans un pays de l'UE sont valables en France"

R3 : PermisValable(x3,y3) → PeutConduire(x3,y3)

"Si on a un permis valable pour un certain lieu, on peut conduire dans ce lieu »

Requête Q = PeutConduire(Ingrid,F) ?

Interrogation d'une base de connaissances

BR
BF

⟺ Requête Q

Q = PeutConduire(Ingrid,F) ?

Q = PeutConduire (Ingrid,x) ∧ FaitPartie(x, UE) ?

Requête conjonctive : conjonction d'atomes (vue comme un ensemble)

- Si instanciée (sans variables) : réponse oui / non
- Sinon, on veut toutes les valeurs possibles pour les variables dans la base de connaissances

Idée :
1) Calculer la base de faits saturée BF*
2) Interroger BF*

Interrogation d'une base de faits

Oublions les règles pour l'instant

BF

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

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

Réponses à Q dans BF ?

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

Un **homomorphisme *h*** de Q dans BF est une substitution des variables de Q par des constantes de BF telle que :

$$h(Q) \subseteq BF$$

où h(Q) est obtenu à partir de Q en substituant chaque variable x par h(x)

Répondre à une requête conjonctive (Q) dans une BF

- Si Q est sans variable :

  la réponse à Q est oui si $Q \subseteq BF$
  (autrement dit, il existe un homomorphisme « vide » de Q dans BF)

  Traduction logique : BF $\vDash$ Q

- De façon générale :

  tout homomorphisme h de Q dans BF définit une réponse à Q

  Traduction logique : BF $\vDash$ h(Q)
  où h est un homomorphisme de Q dans BF

  Remarque : comme Q n'est pas une formule fermée, BF $\vDash$ Q n'aurait pas de sens ; on considère donc les formules fermées obtenues en appliquant les homomorphismes

Exemple : Base de connaissances (pistes cyclables)

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

Requêtes

Direct(A,C) ?   Direct(x,B) ∧ Direct(B,y) ?

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

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

Requête Q = Chemin(A,C) ?

Pour répondre aux requêtes, on va considérer la base de faits saturée BF*

Chaînage avant (logique d'ordre 1)

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

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

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

- Cette application est utile si h(C) ∉ BF
- Appliquer R à BF consiste à ajouter h(C) dans BF
- BF est saturée (par rapport à BR)
  si aucune application d'une règle de BR à BF n'est utile

Algorithme de chaînage avant (ordre 1)

Algorithme ForwardChaining (K) *// Données : K = (BF, BR)*
Début *// Résultat : BF saturée par BR*

Fin ← faux

Tant que non fin
  nouvFaits ← ∅ *// ensemble des nouveaux faits obtenus à cette étape*
  Pour toute règle R : H → C ∈ BR
    Pour tout (nouvel) homomorphisme S de H dans BF
      Si S(C) ∉ (BF ∪ nouvFaits)
        Ajouter S(C) à nouvFaits
  Si nouvFaits = ∅
    Fin ← vrai
  Sinon Ajouter les éléments de nouvFaits à BF

Fin

Une règle peut s'appliquer plusieurs fois

BF* peut être exponentielle en la taille de BF (l'exposant est l'arité maximale des prédicats)
La complexité de FC(K) n'est plus polynomiale

11

Fondations logiques du chaînage avant

- **Règle d'instanciation universelle** :
  Etant donnée la règle R : $\forall x_1 \ldots x_n$ (H → C)
  en remplaçant les $x_1 \ldots x_n$ par des constantes, on obtient
  une règle instanciée R' : H' → C' telle que R ⊨ R'

- **Modus Ponens généralisé** :
  1. Instancier les variables de R ce qui produit R' = H' → C'
  2. Appliquer le *modus ponens* direct : H', (H' → C') ⊨ C'

BF et R : $\forall x_1 \ldots x_n$ (H → C) sont supposées vraies
Par un homomorphisme h de H dans BF, on obtient :
h(H) ⊆ BF et R' : h(H) → h(C)
On a donc : BF ⊨ h(H) et R ⊨ R' donc BF, R ⊨ h(H), h(H) → h(C)
donc **BF, R ⊨ h(C)**

ADÉQUATION ET COMPLÉTUDE DU CHAÎNAGE AVANT

- Le chaînage avant est **adéquat** pour les règles positives :

  Pour tout atome instancié A, si A ∈ BF* alors BF, BR ⊨ A

- Le chaînage avant est **complet** pour les règles positives :

  Pour tout atome instancié A, si BF, BR ⊨ A alors A ∈ BF*

Soit Q une requête conjonctive
et s une substitution des variables de Q par des constantes

BF, BR ⊨ s(Q) ssi s(Q) ⊆ BF*

autrement dit : ssi s est un homomorphisme de Q dans BF*

EXEMPLE (PISTES CYCLABLES)

BF

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

BR

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

Q = 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 BF*

Réponses : x ↦ B   x ↦ C   x ↦ D

Algorithmes de recherche d'homomorphismes

Q = { p(x,y), p(y,z), q(z,x) }, où x, y et z sont des variables
BF = { p(a,b), p(b,a), p(a,c), q(b,b), q(a,c), q(b,c) }
Quels sont les **homomorphismes** de Q dans BF ?

Pourrait-on résoudre ce problème en utilisant un algorithme résolvant CSP ?

Autrement dit :

Peut-on construire un réseau de contraintes à partir de Q et BF tel que les solutions à ce réseau correspondent exactement aux homomorphismes de Q dans B ?

De HOM à CSP

Problème de décision **HOM**
Données : deux ensembles d'atomes A1, A2 « *instance* de HOM »
Question : existe-t-il un homomorphisme de A1 dans A2 ?

Problème de décision **CSP**
Données : réseau (X,D,C) « *instance* de CSP »
Question : ce réseau est-il (globalement) consistant ?

Résoudre HOM en utilisant un algorithme qui résout CSP ?

Autrement dit, trouver une réduction (polynomiale) de HOM à CSP ?

(si possible préservant les solutions)

DE HOM À CSP

A1 = { p(x,y), p(y,z), q(z,x) }, où x, y, z sont des variables
A2 = { p(a,b), p(b,a), p(a,c), q(b,b), q(a,c), q(b,c) }
Quels sont les **homomorphismes** de A1 dans A2 ?

Représentation sous la forme (X,D,C) ?

X = variables de A1

D = constantes de A2 *(ou termes si A2 peut avoir des variables)*

C = conditions à satisfaire par une assignation ?

Pour tout atome p(x1,...,xk) ∈ A1, on a h(p(x1, ..., xk)) ∈ A2

→ les atomes de A1 fournissent les contraintes
→ les atomes de A2 fournissent les tuples de ces contraintes

HOM → CSP

A1 et A2 → (X,D,C)

Supposons que A1 n'ait pas de constantes
et qu'il n'y ait pas deux fois la même variable dans un atome

- ensemble des variables de A1 → X
- ensemble des termes (constantes) de A2 → D(x) pour tout x ∈ X
  toutes les variables de X ont le même domaine

- atomes de A1 et de A2 → C
  tout atome p(x1, ..., xk) de A1 conduit à une contrainte :
  - de portée (x1, ..., xk)
  - définie par l'ensemble des tuples (a1, ..., ak) tels que p(a1, ..., ak) ∈ A2

HOM → CSP *(suite)*

- transformation de p(x,x) ?
  p(x,y) et x = y où y est une nouvelle varaible

- transformation de p(x,a) où a est une constante ?
  p(x,y) où y est une nouvelle variable
  et domaine(y) = {a}

Cas général :

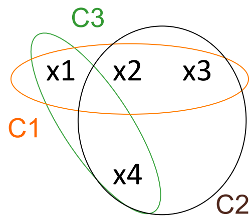
- on remplace les **multi-occurrences** de variables dans **un atome** en introduisant de nouvelles variables et en ajoutant des contraintes d'égalité
- on remplace chaque **constante** par une nouvelle variable dont le domaine est réduit à cette constante

Nous avons construit une réduction (polynomiale) de HOM à CSP (qui, de plus, préserve les solutions)

CSP (Constraint Satisfaction Problem)
Données : un réseau de contraintes P = (X, D, C)
Question : P admet-il une solution? [trouver toutes les solutions]

Exemple de réseau de contraintes

- Ensemble de variables **X**={x1,x2,x3,x4}
- Ensemble de contraintes **C**={C1,C2,C3}
  ⟶ **hypergraphe**



- Domaines des variables **D1=D2=D3=D4**={a,b}
- Définitions des contraintes

C1

x1	x2	x3
a	a	b
a	b	a
b	a	a

C2

x2	x3	x4
a	b	a
b	a	b

C3

x1	x4
a	b
b	b

Réduction (polynomiale) de CSP à HOM (qui, de plus, préserve les solutions) ? Illustrer sur l'exemple

CSP → HOM

(X,D,C) → A1 et A2

X → ensemble des termes (variables) de A1
D → ensemble des termes (constantes) de A2

A1 = { Ci(x1, ..., xk) | Ci ∈ C et porte sur (x1 ... xk) }

A2 = { Ci(a1, ..., ak) | Ci ∈ C et (a1, ..., ak) est dans la définition de Ci }

Retour aux systèmes à base de règles

- Mécanisme de **chaînage arrière** sur les règles positives

- Si on ajoute la **négation classique**, problème d'incomplétude du chaînage avant (et du chaînage arrière)

  BF = { p(a) }
  BR = { q(x) → r(x) ; ¬ q(x) → r(x) }

- On peut considérer la **négation du monde clos**
  - On retrouve les mêmes problèmes qu'en logique des propositions : le résultat du chaînage avant dépend de l'**ordre** dans lequel on considère les règles
  - Même notion d'ensemble de règles **semi-positif** ou **stratifié** qui garantit :
    - une utilisation correcte de la négation
    - un résultat unique

Systèmes à base de connaissances

Base de connaissances *K*

Services de raisonnement

Inférences à partir de *K*

Ontologie
Connaissances générales sur les domaines d'application

Base de faits
Observations particulières sur objets précis

Langages de représentation de connaissances :

- à base de règles
- logiques de description

Inférer de nouvelles connaissances

Tester si la base de connaissances est satisfiable

Répondre à une requête, …

Voir les UE de Représentation des connaissances (M1), Web sémantique (M1), Théorie des bases de connaissances (M2)