Logique classique
Cours 5 : Raisonnement en logique des prédicats

Odile PAPINI

POLYTECH
Université d'Aix-Marseille
odile.papini@univ-amu.fr
http://odile.papini.perso.luminy.univ-amu.fr/sources/LOG.html

Plan du cours

1. Introduction
2. Saturation
3. Unification
4. Résolution
5. Programmation logique

Intoduction

logique et informatique

- représentation des connaissances
- raisonnement sur les connaissances
- mettre en œuvre le raisonnement
- implantation informatique

Raisonnement déductif

(rappel)

$\mathcal{F}$ un ensemble de formules de $\mathcal{L}_{Pr}$ et I une interprétation,

$\mathcal{F}$ est insatisfaisable ou incohérent ssi $\forall I$ telle que $\exists F \in \mathcal{F}$, $I(F) = 0$.

proposition

$F \in \mathcal{L}_{Pr}$, $\quad \mathcal{F} \models F$ ssi $\mathcal{F} \cup \{\neg F\}$ est insatisfaisable ou incohérent.

on ramène le problème de conséquence logique à celui de la cohérence ou de la satisfaisabilité.

réfutation : on cherche à montrer l'incohérence

Saturation

principe

- Mise sous forme prénexe
- Skolemisation
- Construction d'un système de Herbrand

Mise sous forme prénexe

formes prénexes (rappel)

$$Q_1\ x_1\ \cdots\ Q_n\ x_n\ M$$

algorithme

- élimination des connecteurs d'implication et d'équivalence
- renommage des variables (plus de variable libre et liée en même temps)
- suppression des quantificateurs inutiles
- transfert du connecteur de négation immédiatement devant les atomes
- transfert des quantificateurs en tête des formules

Skolémisation

formes de Skolem (rappel)

transformation de A en forme de Skolem S_A

- transformer A en forme prénexe : $Q_1\ x_1\ \cdots\ Q_n\ x_n\ M$
- transformer M en forme conjonctive normale M'
- skolémiser M' :
  1) associer à toute variable quantifiée existentiellement le terme constitué par un symbole fonctionnel ayant pour arguments la liste des variables quantifiées universellement qui précèdent la variable
  2) remplacer chaque occurence de variable quantifiée existentiellement par le terme défini en 1)
  3) supprimer les quantificateurs existentiels

Construction d'un système de Herbrand (rappel)

théorème de Herbrand

on associe à une formule conjonctive normale F l'ensemble C des clauses correspondantes

univers de Herbrand associé à un ensemble de clauses C : ensemble de tous les termes sans variable construit à partir du vocabulaire de C

ensemble de termes sans variable construit à partir des symboles de fonction et des constantes apparaissant dans C

Lorsque le vocabulaire de C ne contient aucune constante on en introduit une arbitrairement

Construction d'un système de Herbrand (rappel)

théorème de Herbrand

C l'ensemble de clauses correspondant à la formule conjonctive normale F

base de Herbrand associé à C : ensemble de tous les atomes sans variable construit à partir des symboles de prédicats appliqués aux termes de l'univers de Herbrand de C

système de Herbrand SH_C associé à C : ensemble des clauses obtenues à partir de C en remplaçant les variables par des éléments de l'univers de Herbrand

théorème de Herbrand

C est satisfaisable ssi SH_C est satisfaisable

Saturation

principe

- Mise sous forme prénexe

  mise sous forme normale

- Skolemisation

  suppression des quantificateurs

- Construction d'un système de Herbrand

  instanciation sur les constantes

- résolution sur les instances

Saturation

exercice

$H_1 = \forall x\, p(x) \rightarrow \forall y\, p(y)$

$H_2 = \forall x\, (p(x) \vee q(x))$

$C = \exists x\, \neg q(x) \rightarrow \forall y\, p(y)$

Utiliser la saturation pour montrer que $\{H_1, H_2\} \models C$

Résolution en calcul des prédicats

résolution R et **saturation S** semi-commutatives :

$$R(S(H_C)) \subseteq S(R(H_C))$$

plus généralement :

$$R^n(S(H_C)) \subseteq S(R^n(H_C))$$

Conséquence

résolution sur clauses prédicatives avant instanciation

étape d'unification préalable

Unification

substitution :

Var : ensemble des variables, T : ensemble des termes
σ fonction de Var dans T, tq
l'ensemble $\{x \in Var, \sigma(x) \neq x\}$ est fini

instance :

t un terme, l : un littéral,

- instance de t (resp. de l), le terme noté $\sigma(t)$ (resp. le littéral $\sigma(l)$), obtenu en remplaçant toutes les occurences des variables x par $\sigma(x)$.
- t_1 et t_2 des termes, t_2 est une instance de t_1 s'il existe une substitution σ telle que $t_2 = \sigma(t_1)$

Unification

unificateur :

2 littéraux l et l' sont **unifiables** s'il existe une substitution σ telle que $\sigma(l) = \sigma(l')$. La substitution est appelée l'unificateur.

unificateur principal :

σ' est plus général que σ, s'il existe σ" tq $\sigma = \sigma" \circ \sigma'$

il existe un unificateur plus général que tous les autres :
unificateur principal

Unification

exemple

On cherche à unifier $G(x, z)$ et $\neg G(y, a)$

unificateur principal $\sigma = \sigma'' \circ \sigma'$

- σ est tel que $\sigma(x) = a$, $\sigma(y) = a$, $\sigma(z) = a$
  On obtient $G(a, a)$ et $\neg G(a, a)$

- $\sigma''(z) = a$ et $\sigma'(x) = y$
  en les composant on obtient $G(y, a)$ et $\neg G(y, a)$ dont les littéraux $G(a, a)$ et $\neg G(a, a)$ sont des instances

Algorithme d'unification

algorithme **unification** (t_1, t_2 : des termes)
début
si l'un des termes (t_1 ou t_2) est une variable x, (l'autre est t)
alors si $x = t$ alors $possible \leftarrow vrai$, $\sigma \leftarrow \emptyset$
sinon
    si x apparaît dans t alors $possible \leftarrow faux$
    sinon $possible \leftarrow vrai$, $\sigma \leftarrow (\sigma(x) = t)$ finsi
fin si
sinon ($t_1 = f(x_1, \cdots, x_n)$ et $t_2 = g(y_1, \cdots, y_m)$)
si $f \neq g$ ou $n \neq m$ alors $possible \leftarrow faux$
sinon $i \leftarrow 0$, $possible \leftarrow vrai$, $\sigma \leftarrow \emptyset$
    tant que $i < n$ et $possible$ faire
        $i \leftarrow i + 1$, $(possible, \sigma') \leftarrow unification(\sigma(x_i), \sigma(y_i))$
        si $possible$ alors $\sigma \leftarrow \sigma' \circ \sigma$ finsi
    fin tant que
fin si fin si
fin

Unification

exercice

- unifier $R(5, x, !(S(y)))$ et $R(z, +(z, w), !(w))$
- unifier $p(x, f(x), a)$ et $p(u, w, w)$

Résolution

proposition (résolution)

S : un ensemble de clauses, c_1, $c_2 \in S$
l_1 apparaît dans c_1 et $\neg l_2$ apparaît dans c_2

θ une substitution de renommage tq $\theta(c_1)$ et c_2 n'ont aucune variable libre en commun
soit σ l'unificateur principal de $\theta(c_1)$ et c_2

$S \models S \cup \{r\}$ et $S \cup \{r\} \models S$

avec $r = \sigma(\theta(c_1 \backslash \{l_1\}) \vee (c_2 \backslash \{\neg l_2\}))$ appelée résolvante

Résolution

Algorithme de résolution

début
tant que $\square \notin S$ faire
choisir l_1, l_2, c_1, c_2 tels que $l \in c_1$ et $l_2 \in c_2$ et
l_1, l_2 unifiables
calculer la résolvante r
à partir de l'unificateur principal
remplacer S par $S \cup \{r\}$
fin tant que
fin

Résolution

exemple de résolution

1) $\neg p(x) \vee p(f(x))$

2) $p(a)$

3) $\neg p(f(z))$

- 1) et 2) avec $\sigma(x) = a$ produit 4) $p(f(a))$
- 3) et 4) avec $\sigma(z) = a$ produit $\square$

Résolution

exercices

- utiliser la résolution pour répondre à la question :
  $\exists x P(x), \forall x \, \forall y (P(x) \rightarrow Q(x, y)) \models \exists x \forall y Q(x, y)$?

- utiliser la résolution pour montrer que $\{H_1, H_2\} \models C$
  avec :
  $H_1 = \forall x \, p(x) \rightarrow \forall y \, p(y)$
  $H_2 = \forall x \, (p(x) \vee q(x))$
  $C = \exists x \, \neg q(x) \rightarrow \forall y \, p(y)$

Résolution

exercice

- Les chevaux sont plus rapides que les chiens
- Il existe un lévrier plus rapide que tout lapin
- Les lévriers sont des chiens
- Harry est un cheval
- Ralph est un lapin

Peut-on déduire que Harry est plus rapide que Ralph ?

Utilisation de la résolution

programmation logique

- programmation déclarative
- premier langage de programmation logique : PROLOG (Luminy, 1973)
- basé sur la logique des prédicats
- résolution mécanisme à la base de l'exécution de programmes PROLOG

Programmation logique

programme logique

un programme logique est un ensemble fini de règles de la forme :

$A.$

ou

$A \leftarrow B_1, \cdots, B_n.$

- A : tête de la règle
- $B_1, \cdots, B_n$: corps de la règle
- A, B_i pour $1 \leq i \leq n$: atomes

Programmation logique

Programmation logique vs programmation impérative

- programmation impérative
  - à l'origine ordinateur conçu pour calculer
  - ne traite pas de situations non spécifiées à l'avance
  - codage d'un algorithme dans un langage de programmation
  - chaque type de question nécessite l'écriture d'une partie ou d'un programme différent
- programmation logique
  - représentation des connaissances sur un sujet dans un formalisme adéquat
  - représentation à partir de laquelle sont déduites des rèponses aux questions de l'utilisateur

Programmation logique vs programmation impérative

programmation logique	programmation impérative
règle	procédure
ensemble de règles	programme
question (but)	appel de procédure
preuve	exécution
substitution, unification	passage de paramètres

Programmation logique

programme logique

P ensemble de règles d'un programme logique : C ensemble de clauses de Horn

Clause de Horn : clause comportant au plus un littéral positif

règle	clause
$A.$	A
$A \leftarrow B_1, \cdots, B_n$	$A \vee \neg B_1 \vee \cdots \vee \neg B_n$

soit P Programme logique, C ensemble de clauses correspondant et Q une Question

$$P \models Q \text{ : résolution sur l'ensemble de clauses } C \cup \{\neg Q\}$$