L o g i q u e

Résumé des épisodes précédents

La démonstration automatique : recherche de démonstrations en calcul des séquents

La Résolution

Une méthode plus « efficace »

I. Clauses et formes clausales

Minimiser le nombre de symboles logiques

Les symboles $\neg$, $\vee$ et $\forall$ suffisent (pour le cas sans quantificateur $\neg$ et $\vee$ suffisent, et même $|$)

Littéral : proposition atomique ou négation proposition atomique
Clause : disjonction de littéraux

Exemple : $\neg P(x) \vee P(S(x))$

Quantification universelle implicite
0 littéral : $\bot$
Disjonction implicitement AC1 (multiensembles)

Toute proposition peut être mise en forme clausale

Transformer un ensemble de propositions Γ en un ensemble de clauses Γ'

(les clôtures universelles de) Γ' cohérent
si et seulement si
(les clôtures universelles de) Γ cohérent

Étape 1 : on remplace $A \Rightarrow B$ par $\neg A \vee B$
Étape 2 : on pousse les négations avec les lois de De Morgan jusqu'aux propositions atomiques

$$\begin{array}{rcl} \neg\top & \longrightarrow & \bot \\ \neg(A \vee B) & \longrightarrow & \neg A \wedge \neg B \\ \neg\forall x\ A & \longrightarrow & \exists x\ \neg A \\ \neg\neg A & \longrightarrow & A \end{array} \qquad \begin{array}{rcl} \neg\bot & \longrightarrow & \top \\ \neg(A \wedge B) & \longrightarrow & \neg A \vee \neg B \\ \neg\exists x\ A & \longrightarrow & \forall x\ \neg A \end{array}$$

Étape 3 :

$$\begin{array}{rcl} \Gamma, C \vee \top & \longrightarrow & \Gamma \\ \Gamma, C \vee \bot & \longrightarrow & \Gamma, C \\ \Gamma, C \vee (A \wedge B) & \longrightarrow & \Gamma, C \vee A, C \vee B \\ \Gamma, C \vee \forall x\ A & \longrightarrow & \Gamma, C \vee A\ (*) \\ \Gamma, C \vee \exists x\ A & \longrightarrow & \Gamma, C \vee (f(y_1, \ldots, y_n)/x)A\ (**) \end{array}$$

(*) $x \notin VL(C)$
(**) f pas dans l'ensemble et $\{y_1, ..., y_n\} = VL(\exists x\ A)$

Exemple

Si Γ est

$$P(0), \quad \forall x\ (P(x) \Rightarrow P(S(x))), \quad \neg P(S(S(0)))$$

alors Γ' est

$$P(0), \quad \neg P(x) \vee P(S(x)), \quad \neg P(S(S(0)))$$

II. La Résolution

Démontrer $\Gamma \vdash A$
Démontrer $\Gamma, \neg A \vdash$
Démontrer $\overline{\forall} C_1, ... \overline{\forall} C_n \vdash$, où $C_1, ..., C_n$ forme clausale de $\Gamma, \neg A$

Recherche d'une contradiction ($\perp$, clause vide) à partir de $C_1, ..., C_n$ avec

$$\frac{P \vee C \quad \neg Q \vee C'}{\sigma(C \vee C')} \sigma = mgu(P, Q) \text{ Résolution}$$

$$\frac{L \vee L' \vee C}{\sigma(L \vee C)} \sigma = mgu(L, L') \text{ Factorisation}$$

Un exemple

$$P(0), \forall x\ (P(x) \Rightarrow P(S(x))) \vdash P(S(S(0)))$$

$$P(0)$$

$$\neg P(x) \vee P(S(x))$$

$$\neg P(S(S(0)))$$

$$\dfrac{\dfrac{\dfrac{P(0) \quad \neg P(x) \vee P(S(x))}{P(S(0))} \quad \neg P(x) \vee P(S(x))}{P(S(S(0)))} \quad \neg P(S(S(0)))}{\bot}$$

Une certaine redondance

$$\dfrac{\dfrac{\dfrac{P(0) \quad \neg P(x) \vee P(S(x))}{P(S(0))} \quad \neg P(x) \vee P(S(x))}{P(S(S(0)))} \quad \neg P(S(S(0)))}{\bot}$$

$$\dfrac{\dfrac{\dfrac{\neg P(S(S(0))) \quad \neg P(x) \vee P(S(x))}{\neg P(S(0))} \quad \neg P(x) \vee P(S(x))}{\neg P(0)} \quad P(0)}{\bot}$$

La nécessité de la règle Factorisation

$$P(w) \vee P(S(x))$$
$$\neg P(y) \vee \neg P(S(z))$$

$$\dfrac{\dfrac{P(w) \vee P(S(x))}{P(S(x))} \quad \dfrac{\neg P(y) \vee \neg P(S(z))}{\neg P(S(z))}}{\bot}$$

Mais la règle Résolution ne donne que des clauses à deux littéraux

III. La correction de la Résolution

Au choix

Si $\perp$ dérivable à partir de $C_1, ..., C_n$, alors $\overline{\forall} C_1, ..., \overline{\forall} C_n \vdash$ démontrable en calcul des séquents

Si $\perp$ dérivable à partir de $C_1, ..., C_n$, alors $\overline{\forall} C_1, ..., \overline{\forall} C_n \vdash$ valide dans tous les modèles

Plus généralement

Si D dérivable à partir de $C_1, ..., C_n$, alors $\overline{\forall} C_1, ..., \overline{\forall} C_n \vdash \overline{\forall} D$ démontrable en calcul des séquents (respectivement valide dans tous les modèles)

Par récurrence sur la structure de la dérivation

Si $\Gamma \vdash \overline{\forall}(P \vee C)$ et $\Gamma \vdash \overline{\forall}(\neg Q \vee C')$ démontrables en calcul des séquents (respectivement valide dans tous les modèles) et $\sigma = mgu(P, Q)$, alors $\Gamma \vdash \overline{\forall}(\sigma(C \vee C'))$ aussi ($\Gamma = \overline{\forall} C_1, ..., \overline{\forall} C_n$)

Repose sur deux lemmes triviaux

$$(\overline{\forall} A) \Rightarrow (\overline{\forall} \sigma A)$$

$$((A \vee B) \wedge (\neg A \vee B')) \Rightarrow (B \vee B')$$

démontrables en calcul des séquents (respectivement valide dans tous les modèles)

Idem pour la règle Factorisation

$$(A \vee A \vee B) \Rightarrow (A \vee B)$$

démontrable en calcul des séquents (respectivement valide dans tous les modèles)

La complétude de la Résolution

Au choix :

Si $\overline{\forall} C_1, ..., \overline{\forall} C_n \vdash$ démontrable en calcul des séquents, alors $\perp$ dérivable à partir de $C_1, ..., C_n$

Si $\overline{\forall} C_1, ..., \overline{\forall} C_n \vdash$ valide dans tous les modèles, alors $\perp$ dérivable à partir de $C_1, ..., C_n$

La prochaine fois

IV. La logique des prédicats comme langage de programmation

Un exemple

$$double(0, 0)$$

$$\forall x \forall y \ (double(x, y) \Rightarrow double(S(x), S(S(y))))$$

Démontrer

$$double(S(S(0)), S(S(S(S(0)))))$$

Un exemple

$$double(0, 0)$$

$$\neg double(x, y) \vee double(S(x), S(S(y)))$$

$$\neg double(S(S(0)), S(S(S(S(0)))))$$

$$\neg double(S(0), S(S(0)))$$

$$\neg double(0, 0)$$

$$\bot$$

Un autre exemple

Démontrer

$$\exists z \; \mathit{double}(S(S(0)), z)$$

$$\neg \mathit{double}(S(S(0)), z)$$

Mieux : trouver un témoin t et une démonstration de

$$\mathit{double}(S(S(0)), t)$$

Identique dans une restriction de la Résolution : la Résolution SLD (*Selective Linear Definite*)

Un autre exemple

$$double(0, 0)$$

$$\neg double(x, y) \vee double(S(x), S(S(y)))$$

$$\neg double(S(S(0)), z)$$

Unification $double(S(S(0)), z) = double(S(x), S(S(y))) : S(0)/x, v/y, S(S(v))/z$

$$\neg double(S(0), v)$$

Unification $double(S(0)), v) = double(S(x), S(S(y))) : 0/x, w/y, S(S(w))/v$

$$\neg double(0, w)$$

Unification $double(0, w) = double(0, 0) : 0/w$

$$\bot$$

au bout du compte $S(S(S(S(0))))/z$

Un programme pour calculer le double

$$double(0, 0)$$

$$\forall x\ \forall y\ (double(S(x), S(S(y))) \Leftarrow double(x, y))$$

$$\neg\exists z\ double(S(S(0)), z)$$

S(S(S(S(0))))/z

Prolog

```
double(o,o).
double(s(X),s(s(Y))) :- double(X,Y).
```

```
?- double(s(s(o)),Z).
```

```
Z = s(s(s(s(o))))
```

Un programme qui calcule aussi la moitié

```
?- double(W,s(s(s(s(o))))).

W = s(s(o))
```

Prolog et les démonstrations constructives

Un prédicat binaire : *double*

Prolog cherche une démonstration (*grosso modo* constructive et sans coupures) de

$$\exists y\ (double(2, y))$$

et extrait le témoin 4 : exécution = recherche de démonstration

Programmer avec des démonstrations consiste à constuire une démonstration de

$$\forall x \exists y\ (double(x, y))$$

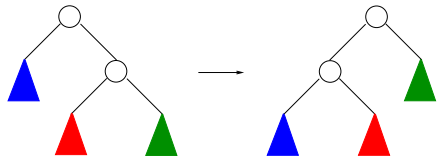
(par exemple par récurrence sur x) pour en déduire

$$\exists y\ (double(2, y))$$

et extraire le témoin : exécution = élimination des coupures

Que Prolog a-t-il de si formidable ?

Une réponse : on accède simplement à un sous-arbre



$$t \rightarrow noeud(noeud(gauche(t), gauche(droite(t))), droite(droite(t)))$$

$$noeud(x, noeud(y, z)) \rightarrow noeud(noeud(x, y), z)$$

Unification (filtrage) entre la donnée et $noeud(x, noeud(y, z))$

Pourquoi pas dans les autres langages ?

En ML (Lisp, Maude...)

```
let f t = match t with
| Noeud(x,Noeud(y,z)) -> Noeud(Noeud(x,y),z)
```

```
let rec double n = match n with
| O -> O
| S(x) -> S(S(double x))
```

V. Le choix de l'unificateur

De nombreuses solutions

$$double(S(S(0)), z) = double(S(x), S(S(y)))$$

$$S(0)/x, S(S(S(S(S(S(S(0)))))))/y, S(S(S(S(S(S(S(S(S(0)))))))))/z$$

$$S(0)/x, v/y, S(S(v))/z$$

Les variables résiduelles

L'algorithme de Gauss appliqué à

$$x + y = 4$$

$$x - y = 2$$

donne $3/x, 1/y$: solution unique
Mais appliqué à

$$x + y = 4$$

$$2x + 2y = 8$$

il donne $(4 - y)/x$, variable résiduelle y : plusieurs solutions
Description paramétrique de l'ensemble des solutions
Mieux : $(4 - v)/x, v/y$

$$double(S(S(0)), z) = double(S(x), S(S(y)))$$

L'algorithme d'unification donne $S(0)/x, S(S(y))/z$
Simplification

$$S(S(0)) = S(x)$$
$$z = S(S(y))$$

Simplification

$$S(0) = x$$
$$z = S(S(y))$$

Solution : $S(0)/x, S(S(y))/z$
ou $S(0)/x, v/y, S(S(v))/z$

L'algorithme d'unification

- $f(t_1, ..., t_n) = f(u_1, ..., u_n)$: on remplace cette équation par $t_1 = u_1, ..., t_n = u_n$
- $f(t_1, ..., t_n) = g(u_1, ..., u_m)$: on échoue
- $X = X$: on supprime cette équation
- $X = t$ (ou $t = X$), X apparaît dans t, t distinct de X : on échoue
- $X = t$ (ou $t = X$), X n'apparaît pas dans t : on substitue X par t dans le reste du système, on résout ($\longrightarrow$ substitution σ), on retourne $\sigma \cup \{\sigma t/X\}$

Solution principale

La substitution σ calculée par l'algorithme d'unification est une solution principale (*most general unifier* : mgu)

Pour toute solution θ il existe η telle que $\theta = \eta \circ \sigma$

Unique à un renomage des variables près

C'est cette substitution que nous devons utiliser dans la règle de Résolution (et de Factorisation)

$$\frac{P \vee C \quad \neg Q \vee C'}{\sigma(C \vee C')}\,\sigma = mgu(P, Q)\ \text{Résolution}$$

L'algorithme de Gauss calcule aussi une solution principale

$$x + y = 4$$

$$2x + 2y = 8$$

L'algorithme de Gauss donne $\sigma = (4 - y)/x$

Pour $\theta = 1/x, 3/y$, il existe $\eta = 3/y$, telle que $\theta = \eta \circ \sigma$, idem pour toute solution réelle

De plus, pour $\theta = w^2/x, (4 - w^2)/y$, il existe $\eta = (4 - w^2)/y$, telle que $\theta = \eta \circ \sigma$, idem pour toute solution paramétrique

La prochaine fois : la complétude de la Résolution