La Résolution

ENS Paris-Saclay

Lundi 4 avril 2022

Simplification des formules

$\neg$, $\vee$ et $\forall$ suffisent à exprimer toutes les propositions.

Definition (littéraux et forme clausale)

- Littéral : proposition atomique ou négation d'une proposition atomique
- Clause : disjonction de littéraux (clause vide : $\bot$)

Simplification des $\forall$: quantification universelle implicite

Simplification des formules

$\neg$, $\vee$ et $\forall$ suffisent à exprimer toutes les propositions.

Definition (littéraux et forme clausale)

- Littéral : proposition atomique ou négation d'une proposition atomique
- Clause : disjonction de littéraux (clause vide : $\bot$)

Simplification des $\forall$: quantification universelle implicite

- $A \Rightarrow B$ devient $\neg A \vee B$.
- Loi de De Morgan pour amener les négations aux propositions atomiques.
- Dans $C_1 \vee ... \vee C_n$, si C_i n'est pas un littéral : on le simplifie.

La Résolution

Si Γ est un ensemble de propositions, on construit l'ensemble Γ' des clauses associées : la clôture universelle de Γ est cohérente ssi celle de Γ' l'est.

La Résolution

Si Γ est un ensemble de propositions, on construit l'ensemble Γ' des clauses associées : la clôture universelle de Γ est cohérente ssi celle de Γ' l'est.

Pour démontrer $\Gamma \vdash A$, on montre $\Gamma, \neg A \vdash \bot$ en mettant $\Gamma, \neg A$ sous forme clausale.

Deux règles :

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

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

Correction de la Résolution

Theorem (Correction de la Résolution)

Si $\Gamma \vdash A$ est démontrable en Résolution, alors $\Gamma \vdash A$ est démontrable en déduction naturelle/calcul des séquents.

On montre (par récurrence sur l'arbre de preuve) que :

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

Preuve de la correction (1/2)

Pour la règle Résolution : $\dfrac{P \vee C \quad \neg Q \vee C'}{\sigma(C \vee C')}$ $(\sigma = mgu(P, Q))$

HR : $\Gamma \vdash \bar{\forall}(P \vee C)$ et $\Gamma \vdash \bar{\forall}(\neg Q \vee C')$ sont démontrables en calcul des séquents (où $\Gamma = \bar{\forall} C_1, ..., \bar{\forall} C_n$).

On veut montrer que $\Gamma \vdash \bar{\forall}\sigma(C \vee C')$ est démontrable en calcul des séquents.

Lemmes :

- $\bar{\forall} A \Rightarrow \bar{\forall}\sigma A$
- $((A \vee B) \wedge (\neg A \vee B')) \Rightarrow (B \vee B')$

Preuve de la correction (2/2)

Pour la règle Factorisation : $\dfrac{L \vee L' \vee C}{\sigma(L \vee C)}$ où $\sigma = mgu(L, L')$

HR : $\Gamma \vdash \bar{\forall}(L \vee L' \vee C)$ est démontrable en calcul des séquents (où $\Gamma = \bar{\forall}C_1, ..., \bar{\forall}C_n$).

On veut montrer que $\Gamma \vdash \bar{\forall}\sigma(L \vee C)$ est démontrable en calcul des séquents.

Lemme :

- $A \vee A \vee B \Rightarrow A \vee B$

Langage de programmation : Prolog

Programme pour calculer le double d'un nombre :

On suppose :

- $double(0, 0)$
- $\forall x \forall y\ double(x, y) \Rightarrow double(S(x), S(S(y)))$

Et on montre par exemple : $\exists z\ double(S(S(0)), z)$

Langage de programmation : Prolog

Programme pour calculer le double d'un nombre :

On suppose :

- $double(0, 0)$
- $\forall x \forall y\ double(x, y) \Rightarrow double(S(x), S(S(y)))$

Et on montre par exemple : $\exists z\ double(S(S(0)), z)$

Pour calculer la moitié : $\exists z\ double(z, S(S(S(S(0)))))$

Langage de programmation : Prolog

Programme pour calculer le double d'un nombre :

On suppose :

- $double(0, 0)$
- $\forall x \forall y\ double(x, y) \Rightarrow double(S(x), S(S(y)))$

Et on montre par exemple : $\exists z\ double(S(S(0)), z)$

Pour calculer la moitié : $\exists z\ double(z, S(S(S(S(0)))))$

Problème : pas de propriété du témoin en Résolution

Langage de programmation : Prolog

Programme pour calculer le double d'un nombre :

On suppose :

- $double(0, 0)$
- $\forall x \forall y\ double(x, y) \Rightarrow double(S(x), S(S(y)))$

Et on montre par exemple : $\exists z\ double(S(S(0)), z)$

Pour calculer la moitié : $\exists z\ double(z, S(S(S(S(0)))))$

Problème : pas de propriété du témoin en Résolution

On utilise donc la Résolution SLD

Prolog :
- Recherche d'une démo constructive sans coupure (avec la Résolution SLD)
- Exécution : extraction du témoin

Prolog :
- Recherche d'une démo constructive sans coupure (avec la Résolution SLD)
- Exécution : extraction du témoin

Démo constructive :
- On fournit une démo constructive de la prop quantifiée universellement
- Exécution : élimination des coupures

Choix de l'unificateur

Problème : plusieurs choix d'unification possibles

Par exemple, pour
$double(S(S(0)), z) = double(S(x), S(S(y)))$:
S(0)/x, 0/y, S(S(0))/z et S(0)/x, S(0)/y, S(S(S(0)))/z
conviennent.

Choix de l'unificateur

Problème : plusieurs choix d'unification possibles

Par exemple, pour
$double(S(S(0)), z) = double(S(x), S(S(y)))$:
S(0)/x, 0/y, S(S(0))/z et S(0)/x, S(0)/y, S(S(S(0)))/z conviennent.

On utilise la "solution principale" σ renvoyée par l'algo d'unification : $S(0)/x, v/y, S(S(v))/z$.
Elle est unique et plus formellement elle vérifie :
Pour toute solution θ, il existe η telle que $\theta = \eta \circ \sigma$.