Cours 11

Calcul des prédicats : résolution

Présentation

La déduction naturelle est un système de déduction correcte et complet mais il ne se prête pas à la preuve automatique.

Dans ce cours, nous intéressons à la preuve par résolution :

- une extension de la preuve par coupure à la logique du 1er ordre,
- qui sert de base à la programmation logique (Prolog).

La preuve par recherche de modèle

Nous chercherons à montrer qu'une théorie (ensemble de formules closes) n'admet pas de modèle, car

Lemme

Une formule close F est conséquence d'une théorie T ($T \models^* F$) *ssi* $T \cup \{\neg F\}$ *n'admet pas de modèle (i.e est contradictoire).*

Plan

1ère étape : mise sous forme de clauses

$$\text{ensemble de formules} \\ \downarrow \\ \text{ensemble de clauses}$$

Mise sous forme de clauses

Littéral : une formule atomique ou sa négation
Clause : une disjonction de littéraux

On passe d'un ensemble Σ de formules à un ensemble S de clôtures universelles de clauses :

1. mise sous forme prénexe normale conjonctive des formules de Σ,
2. mise sous forme de Skolem,
3. distribution des $\forall$ par rapport à $\wedge$ dans chacune des formules,
4. décomposition des conjonctions en ensemble de clôtures universelles de clauses (les $\forall$ sont alors implicites).

Mise sous forme de clauses : exemple

L contient P et Q, des prédicats unaires.

$$\Sigma = \{(\exists x P x \Rightarrow \forall y P y), \forall x (P x \vee Q x), \forall z \neg(\exists x \neg Q x \Rightarrow \forall y P y)\}$$

Mise sous forme de clauses : exemple

L contient P et Q, des prédicats unaires.

$$\begin{aligned} \Sigma &= \{(\exists x Px \Rightarrow \forall y Py), \forall x (Px \vee Qx), \forall z \neg(\exists x \neg Qx \Rightarrow \forall y Py)\} \\ &\xrightarrow{1} \{\forall x \forall y (\neg Px \vee Py), \forall x (Px \vee Qx), \forall z \exists x \exists y (\neg Qx \wedge \neg Py)\} \end{aligned}$$

Mise sous forme de clauses : exemple

L contient P et Q, des prédicats unaires.

$$\begin{aligned}
\Sigma &= \{(\exists x Px \Rightarrow \forall y Py), \forall x(Px \vee Qx), \forall z \neg(\exists x \neg Qx \Rightarrow \forall y Py)\} \\
&\xrightarrow{1} \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), \forall z \exists x \exists y(\neg Qx \wedge \neg Py)\} \\
&\xrightarrow{2} \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), \forall z(\neg Qfz \wedge \neg Pgz)\}
\end{aligned}$$

f, g nouveaux symboles de fonctions d'arité 1

Mise sous forme de clauses : exemple

L contient P et Q, des prédicats unaires.

$$\Sigma = \{(\exists x Px \Rightarrow \forall y Py), \forall x(Px \vee Qx), \forall z \neg(\exists x \neg Qx \Rightarrow \forall y Py)\}$$

$$\xrightarrow{1} \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), \forall z \exists x \exists y(\neg Qx \wedge \neg Py)\}$$

$$\xrightarrow{2} \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), \forall z(\neg Qfz \wedge \neg Pgz)\}$$

f, g nouveaux symboles de fonctions d'arité 1

$$\xrightarrow{3} \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), (\forall z \neg Qfz \wedge \forall z \neg Pgz)\}$$

Mise sous forme de clauses : exemple

L contient P et Q, des prédicats unaires.

$$
\begin{aligned}
\Sigma \quad &= \quad \{(\exists x Px \Rightarrow \forall y Py), \forall x(Px \vee Qx), \forall z \neg(\exists x \neg Qx \Rightarrow \forall y Py)\} \\
&\xrightarrow{1} \quad \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), \forall z \exists x \exists y(\neg Qx \wedge \neg Py)\} \\
&\xrightarrow{2} \quad \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), \forall z(\neg Qfz \wedge \neg Pgz)\} \\
&\qquad\qquad\qquad f,\ g \text{ nouveaux symboles de fonctions d'arité 1} \\
&\xrightarrow{3} \quad \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), (\forall z \neg Qfz \wedge \forall z \neg Pgz)\} \\
&\xrightarrow{4} \quad \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), \forall z \neg Qfz, \forall z \neg Pgz\}
\end{aligned}
$$

Mise sous forme de clauses : exemple

L contient P et Q, des prédicats unaires.

$$\begin{aligned}
\Sigma &= \{(\exists x Px \Rightarrow \forall y Py), \forall x(Px \vee Qx), \forall z \neg(\exists x \neg Qx \Rightarrow \forall y Py)\} \\
&\xrightarrow{1} \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), \forall z \exists x \exists y(\neg Qx \wedge \neg Py)\} \\
&\xrightarrow{2} \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), \forall z(\neg Qfz \wedge \neg Pgz)\} \\
&\qquad\qquad f,\ g \text{ nouveaux symboles de fonctions d'arité 1} \\
&\xrightarrow{3} \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), (\forall z \neg Qfz \wedge \forall z \neg Pgz)\} \\
&\xrightarrow{4} \{\forall x \forall y(\neg Px \vee Py), \forall x(Px \vee Qx), \forall z \neg Qfz, \forall z \neg Pgz\} \\
S &= \{(\neg Px \vee Py), (Px \vee Qx), \neg Qfz, \neg Pgz\}
\end{aligned}$$

Validité

Proposition

S admet un modèle ssi Σ admet un modèle.

Preuve :

- étape 1 : toute forme prénexe d'une formule F est équivalent à F.
- étape 2 : une formule admet un modèle ssi une quelconque de ses formes de Skolem admet un modèle.
- étape 3 : équivalences sémantiques standards
$$\forall x(F \wedge G) \equiv (\forall x F \wedge \forall x G)$$
- étape 4 : $\bigwedge_i F_i$ admet un modèle ssi l'ensemble $\{F_i\}$ admet un modèle.

Une première tentative : la méthode de Herbrand

ensemble de formules

$\downarrow$

ensemble de clauses

$\downarrow$

ensemble de clauses propositionelles + preuve par coupure

La méthode de Herbrand

La recherche exhaustive de modèle est impossible pour une machine, mais on peut se contenter d'une certaine famille de modèles : les *modèles de Herbrand*.

Théorème (de Herbrand)

Une théorie (ensemble de formules closes) admet un modèle ssi elle admet un modèle de Herbrand.

Modèles de Herbrand

Pour un langage du premier ordre L (ayant au moins une constante), on définit

- *Base de Herbrand* : l'ensemble des variables propositionelles $p_{[Rt_1 \cdots t_n]}$ où R un prédicat d'arité n de L et les $t_1, \ldots, t_n$ sont des termes clos de L.
- *Domaine de Herbrand* : l'ensemble des termes clos de L.
- *Modèle de Herbrand* (pour une valuation propositionelle φ sur la base de Herbrand) : le modèle $\mathcal{H}(\varphi)$ définie par
  - domaine : le domaine de Herbrand
  - pour toute constante c, $c^{\mathcal{H}(\varphi)} = c$,
  - pour toute symbole de fonction f d'arité n, $f^{\mathcal{H}(\varphi)} = (t_1, .., t_n) \mapsto ft_1..t_n$,
  - pour toute symbole de prédicat R d'arité n, $R^{\mathcal{H}(\varphi)} = \{(t_1, .., t_n) \,|\varphi(p_{[Rt_1 \cdots t_n]}) = 1\}$

Particularisation

Définition

Une formule *prénexe universelle* est une formule prénexe (close) sans quantificateurs existentiels.

Définition

Une *particularisation* d'une formule prénexe universelle $\forall x_1 \cdot \forall x_n F$ (avec F sans quantificateurs) est une formule de la forme

$$F[t_1/x_1, \ldots, t_n/x_n]$$

avec $t_1, \ldots, t_n$ des termes clos.

Exemple : dans $L = \{a_{(0)}, f_{(1)}, R_{(2)}\}$, $\forall x \forall y Rxfy$ admet comme particularisations

$$Rafa, \quad Rfafa, \quad Rfaffa, \quad \ldots$$

Une réduction au cas propositionnel

La méthode de Herbrand transforme le problème de recherche de modèle en un problème de satisfiabilité de formules propositionelles.

Si F est une formule close sans quantificateurs, nous notons $\phi(F)$ la formule propositionelle obtenue à partir de F en remplaçant chaque formule atomique $Rt_1 \cdots t_n$ par la variable propositionelle $p_{[Rt_1 \cdots t_n]}$.

Exemple :

$$\Phi\left((Rafa \wedge Raa)\right) = (p_{[Rafa]} \wedge p_{[Raa]})$$

Une réduction au cas propositionnel

Remarque : Pour tout valuation φ, $\mathcal{H}(\varphi) \models F$ ssi $\varphi(\Phi(F)) = 1$.

Théorème

Soit S un ensemble de formules prénexes universelles. Soit S_{part} l'ensemble des particularisations des formules de S et $\Phi(S_{part})$ l'image de l'ensemble S_{part} par Φ. Les assertions suivantes sont équivalentes :

- *S admet un modèle*
- *S admet un modèle de Herbrand*
- *S_{part} admet un modèle de Herbrand*
- *$\Phi(S_{part})$ est satisfiable*
- *il n'existe pas de preuve par coupure de $\Phi(S_{part}) \vdash \square$*

Remarque : cette dernière assertion nous fournit une méthode de déduction correcte et complète.

Méthode de Herbrand : exemple

Montrons $\exists x Px,\ \forall x \forall y (Px \Rightarrow Qxy) \models^* \exists x \forall y Qxy$

Méthode de Herbrand : exemple

Montrons $\exists x Px,\ \forall x \forall y (Px \Rightarrow Qxy) \models^* \exists x \forall y Qxy$

1. on montre que $S = \{\exists x Px,\ \forall x \forall y (Px \Rightarrow Qxy), \neg \exists x \forall y Qxy\}$ est contradictoire

Méthode de Herbrand : exemple

Montrons $\exists x Px,\ \forall x \forall y(Px \Rightarrow Qxy) \models^* \exists x \forall y Qxy$

1. on montre que $S = \{\exists x Px,\ \forall x \forall y(Px \Rightarrow Qxy), \neg \exists x \forall y Qxy\}$ est contradictoire
2. mise sous forme de clauses : $\{Pa,\ \forall x \forall y(\neg Px \vee Qxy), \forall x \neg Qxfx\}$

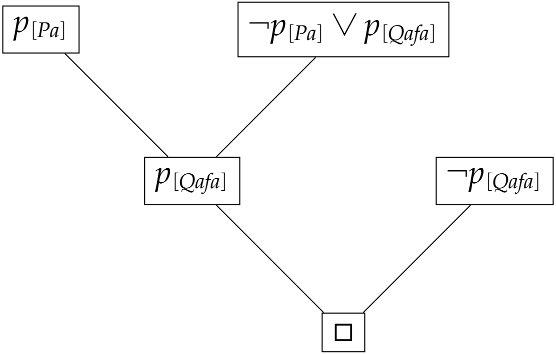
Méthode de Herbrand : exemple

Montrons $\exists x Px,\ \forall x \forall y (Px \Rightarrow Qxy) \models^* \exists x \forall y Qxy$

1. on montre que $S = \{\exists x Px,\ \forall x \forall y (Px \Rightarrow Qxy), \neg \exists x \forall y Qxy\}$ est contradictoire
2. mise sous forme de clauses : $\{Pa,\ \forall x \forall y (\neg Px \vee Qxy), \forall x \neg Qxfx\}$
3. nous cherchons une preuve par coupure de

$$p_{[Pa]}, \bigcup_{n,m} (\neg p_{[Pf^n a]} \vee p_{[Qf^n a f^m a]}), \bigcup_{n} \neg p_{[Qf^n a f^{n+1} a]} \vdash \square$$

Méthode de Herbrand : exemple



Résolution

ensemble de formules

$\downarrow$

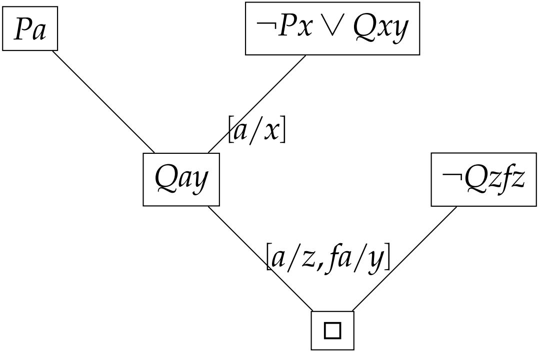
ensemble de clauses

$\downarrow$

résolution (preuve par coupure + unification)

Unification

Nous voulons appliquer la méthode de Herbrand, mais sans avoir à particulariser les formules.



Il faut unifier les formules atomiques, *i.e.* trouver une substitution des variables par des termes pour pouvoir identifier les formules.

Substitution

Définition

Une *substitution* α est une application de $\mathbb{V}$ (ensemble des variables) dans $\mathcal{T}$ (termes). Le domaine d'une substitution α (*i.e.* $\{x \mid \alpha(x) \neq x\}$ est supposé fini.
Si $\mathrm{dom}(\alpha) = \{x_1, .., x_k\}$ et $\alpha(x_i) = t_i$, on note $\alpha = [t_1/x_1, .., t_k/x_k]$ et $\alpha F = F[t_1/x_1, .., t_k/x_k]$.

Exemple : $L = \{R_{(2)}, f_{(1)}, g_{(2)}, h_{(2)}, k_{(3)}, c_{(0)}, d_{(0)}\}$,
$\alpha = [gdz/x, fz/y, d/z]$,
$C = (Rxfy \vee \neg Rgycz)$, alors $\alpha C =$

Substitution

Définition

Une *substitution* α est une application de $\mathbb{V}$ (ensemble des variables) dans $\mathcal{T}$ (termes). Le domaine d'une substitution α (*i.e.* $\{x \mid \alpha(x) \neq x\}$ est supposé fini.
Si $\mathrm{dom}(\alpha) = \{x_1, .., x_k\}$ et $\alpha(x_i) = t_i$, on note $\alpha = [t_1/x_1, .., t_k/x_k]$ et $\alpha F = F[t_1/x_1, .., t_k/x_k]$.

Exemple : $L = \{R_{(2)}, f_{(1)}, g_{(2)}, h_{(2)}, k_{(3)}, c_{(0)}, d_{(0)}\}$,
$\alpha = [gdz/x, fz/y, d/z]$,
$C = (Rxfy \vee \neg Rgycz)$, alors $\alpha C = (Rgdzffz \vee \neg Rgfzcd)$

Unification

Définition

Un ensemble fini de formules atomiques $\{A_1, .., A_n\}$ est *unifiable* s'il existe une substitution α telle que $\alpha A_1 = \alpha A_2 = .. = \alpha A_n$.

Exemple : $L = \{R_{(2)}, f_{(1)}, g_{(2)}, h_{(2)}, k_{(3)}, c_{(0)}, d_{(0)}\}$
$\{Rxfy, Rgycz, Rgtvft\}$ unifiable par

R	x	fy
R	gyc	z
R	gtv	ft

Unification

Définition

Un ensemble fini de formules atomiques $\{A_1, .., A_n\}$ est *unifiable* s'il existe une substitution α telle que $\alpha A_1 = \alpha A_2 = .. = \alpha A_n$.

Exemple : $L = \{R_{(2)}, f_{(1)}, g_{(2)}, h_{(2)}, k_{(3)}, c_{(0)}, d_{(0)}\}$
$\{Rxfy, Rgycz, Rgtvft\}$ unifiable par $[gtc/x, t/y, ft/z, c/v]$,

R	x	fy
R	gyc	z
R	gtv	ft

Unification

Définition

Un ensemble fini de formules atomiques $\{A_1, .., A_n\}$ est *unifiable* s'il existe une substitution α telle que $\alpha A_1 = \alpha A_2 = .. = \alpha A_n$.

Exemple : $L = \{R_{(2)}, f_{(1)}, g_{(2)}, h_{(2)}, k_{(3)}, c_{(0)}, d_{(0)}\}$
$\{Rxfy, Rgycz, Rgtvft\}$ unifiable par $[gtc/x, t/y, ft/z, c/v]$, $[gyc/x, y/t, fy/z, c/v]$,

R	x	fy
R	gyc	z
R	gtv	ft

Unification

Définition

Un ensemble fini de formules atomiques $\{A_1, .., A_n\}$ est *unifiable* s'il existe une substitution α telle que $\alpha A_1 = \alpha A_2 = .. = \alpha A_n$.

Exemple : $L = \{R_{(2)}, f_{(1)}, g_{(2)}, h_{(2)}, k_{(3)}, c_{(0)}, d_{(0)}\}$
$\{Rxfy, Rgycz, Rgtvft\}$ unifiable par $[gtc/x, t/y, ft/z, c/v]$,
$[gyc/x, y/t, fy/z, c/v]$, $[gdc/x, d/t, d/y, fd/z, c/v]$, ...

R	x	fy
R	gyc	z
R	gtv	ft

Unification

On cherche à unifier des ensembles finis de couples de termes.

Définition

S ensemble fini de couples (non ordonnés) de termes
$S = \{< t_i, t'_i > \mid 1 \leqslant i \leqslant m\}$

- une substitution α est une *unificateur* de S si

$$\forall i \in [1, m],\ \alpha t_i = \alpha t'_i$$

- α est un *unificateur principal* si c'est un unificateur de S tel que pour tout σ unificateur de S il existe une substitution β telle que $\sigma = \beta \circ \alpha$

Exemple : on cherche à unifier $\{x, fy\}$

- $[fy/x, t/y]$ est un unificateur pour n'importe quel terme t,
- $[fy, x]$ est un unificateur principal.

Unificateur principal

Proposition

Si σ et σ' sont des unificateurs principaux de $\{< t, t' >\}$, alors ils sont égaux à une permutation des variables près.

Algorithme d'unification

On cherche à unifier les formules atomiques $A = Rt_1, .., t_n$ et $B' = R't'_1, .., t'_n$.

On part de $S = \{<t_i, t'_i>, 1 \leqslant i \leqslant n\}$

Simplification et réduction :

1. pour chaque couple $<t, t'>$ dans S, $t = fu_1..u_n$ et $t' = f'u'_1..u'_n$:
   - si $f \neq f'$, S n'a pas d'unificateur.
   - si $f = f'$, on remplace dans S, $\{<t, t'>\}$ par $\{<u_i, u'_i>, 1 \leqslant i \leqslant n\}$

   On itère le procédé tant que c'est possible. Le nombre d'itérations est fini, hauteur$(S) = max_i(min(hauteur(t_i), hauteur(t'_i)))$. La hauteur de S décroît strictement à chaque itération.

2. on supprime les couples redondants, et les couples $<t, t>$.

On note S' le système simplifié et réduit (après ces 2 étapes).

Algorithme d'unification

Proposition

S' a les mêmes unificateurs que S.
S' est de hauteur nulle : tous ses couples sont de la forme $<x,t>$ ou $<c,t>$.

Définition

On appelle couples *réductibles* ceux de la forme $<x,t>$ avec $x \notin \mathrm{Var}(t)$. Les couples irréductibles sont de la forme $<c,c'>$ avec $c \neq c'$, ou $<c, ft_1..t_n>$ ou $<x,t>$ avec $x \in \mathrm{Var}(t)$.

Proposition

Si S' de hauteur nulle contient un couple irréductible, alors il n'est pas unifiable.

Unification : algorithme

- on part de $\sigma = \emptyset$
- on construit S
- tant que S n'est pas vide :
  - on simplifie $S \rightarrow S'$
  - s'il y a un couple irréductible dans S', échec.
  - on choisit un couple $<x, t>$ dans S'.
    On recommence avec $\sigma = [t/x] \circ \sigma$ et
    $S = \{<\sigma(x'), \sigma(t')> \mid <x', t'> \in S' - \{<x, t>\}\}$.

Unification : validité

Théorème

L'algorithme donne un unificateur principal.

Unification : exemple

Nous cherchons à unifier $Rxgxz$ et $Rfcgfyfw$.

Unification : exemple

Nous cherchons à unifier $Rxgxz$ et $Rfcgfyfw$.

- $\sigma = \emptyset$ et $S = \{<x, fc>, <gxz, gfyfw>\}$

Unification : exemple

Nous cherchons à unifier $Rxgxz$ et $Rfcgfyfw$.

- $\sigma = \emptyset$ et $S = \{< x, fc >, < gxz, gfyfw >\}$
- on simplifie : $S \to \{< x, fc >, < x, fy >, < z, fw >\}$

Unification : exemple

Nous cherchons à unifier $Rxgxz$ et $Rfcgfyfw$.

- $\sigma = \emptyset$ et $S = \{<x, fc>, <gxz, gfyfw>\}$
- on simplifie : $S \to \{<x, fc>, <x, fy>, <z, fw>\}$
- on choisit $<x, fc>$: $\sigma = [fc/x]$ et $S = \{<fc, fy>, <z, fw>\}$

Unification : exemple

Nous cherchons à unifier $Rxgxz$ et $Rfcgfyfw$.

- $\sigma = \emptyset$ et $S = \{<x, fc>, <gxz, gfyfw>\}$
- on simplifie : $S \rightarrow \{<x, fc>, <x, fy>, <z, fw>\}$
- on choisit $<x, fc>$: $\sigma = [fc/x]$ et $S = \{<fc, fy>, <z, fw>\}$
- on simplifie : $S \rightarrow \{<c, y>, <z, fw>\}$

Unification : exemple

Nous cherchons à unifier $Rxgxz$ et $Rfcgfyfw$.

- $\sigma = \emptyset$ et $S = \{<x, fc>, <gxz, gfyfw>\}$
- on simplifie : $S \to \{<x, fc>, <x, fy>, <z, fw>\}$
- on choisit $<x, fc>$: $\sigma = [fc/x]$ et $S = \{<fc, fy>, <z, fw>\}$
- on simplifie : $S \to \{<c, y>, <z, fw>\}$
- on choisit $<c, y>$: $\sigma = [fc/x, c/y]$ et $S = \{<z, fw>\}$

Unification : exemple

Nous cherchons à unifier *Rxgxz* et *Rfcgfyfw*.

- $\sigma = \emptyset$ et $S = \{<x, fc>, <gxz, gfyfw>\}$
- on simplifie : $S \to \{<x, fc>, <x, fy>, <z, fw>\}$
- on choisit $<x, fc>$: $\sigma = [fc/x]$ et $S = \{<fc, fy>, <z, fw>\}$
- on simplifie : $S \to \{<c, y>, <z, fw>\}$
- on choisit $<c, y>$: $\sigma = [fc/x, c/y]$ et $S = \{<z, fw>\}$
- on simplifie : $S \to S$

Unification : exemple

Nous cherchons à unifier $Rxgxz$ et $Rfcgfyfw$.

- $\sigma = \emptyset$ et $S = \{<x, fc>, <gxz, gfyfw>\}$
- on simplifie : $S \to \{<x, fc>, <x, fy>, <z, fw>\}$
- on choisit $<x, fc>$: $\sigma = [fc/x]$ et $S = \{<fc, fy>, <z, fw>\}$
- on simplifie : $S \to \{<c, y>, <z, fw>\}$
- on choisit $<c, y>$: $\sigma = [fc/x, c/y]$ et $S = \{<z, fw>\}$
- on simplifie : $S \to S$
- on choisit $<z, fw>$: $\sigma = [fc/x, c/y, fw/z]$ et $S = \emptyset$

Résolution

Soient $C_1 = (\Gamma_1, \Delta_1), C_2 = (\Gamma_2, \Delta_2)$, des clauses sans variables communes.

Définition

C est un *résolvant* de C_1 et C_2 s'il existe $P_1 \subseteq \Gamma_1$ et $N_2 \subseteq \Delta_2$ tels que $P_1 \cup N_2$ unifiable et si σ est une unificateur principal, alors $C = (\Gamma, \Delta)$ avec $\Gamma = \sigma(\Gamma_1 \setminus P_1) \cup \sigma(\Gamma_2)$ et $\Delta = \sigma(\Delta_2 \setminus N_2) \cup \sigma(\Delta_1)$. On note

$$\frac{C_1 \qquad C_2}{C}$$

De même qu'en calcul propositionnel, on définit une *preuve par résolution* et une *réfutation par résolution*.

Correction de la résolution

On note $\forall C$ la clôture universelle de C.

Lemme

Si C résolvant de C_1 et C_2, alors $\forall C$ est conséquence de $\{\forall C_1, \forall C_2\}$.

Corollaire

S'il existe une réfutation par résolution d'un ensemble de clôtures universelles de clauses S, alors S n'a pas de modèle.

Complétude de la résolution

Théorème

Si un ensemble de clôtures universelles de clauses S n'a pas de modèle, alors il existe une réfutation par résolution de S.

Preuve. On se base sur la complétude de la méthode de Herbrand.

- il existe une preuve par coupure de $\Phi(S_{\text{part}}) \vdash \square$
- toute preuve par coupure de $\Phi(S_{\text{part}}) \vdash \square$ peut être transformée en une réfutation par résolution de S

Exercice

Soient P un prédicat unaire, Q et R des prédicats binaires et g une fonction binaire. Montrer par la méthode de résolution que F est conséquence de $\{F_1, F_2, F_3\}$.

$$
\begin{array}{ll}
F_1 & \forall y\ (\exists z\ \neg Q(z,y) \Rightarrow \forall x\ (R(x,y) \vee P(g(x,y)))) \\
F_2 & \forall x\ \forall y\ ((P(y) \wedge \exists z\ R(x,z)) \Rightarrow Q(y,x)) \\
F_3 & \forall x\ (\exists y\ R(y,x)) \Rightarrow \forall z\ P(g(z,x)) \\
F & \exists y\ \forall z\ \exists x\ (P(g(y,y)) \vee Q(x,z))
\end{array}
$$

Correction de l'exercice

Nous faisons une preuve par réfutation de $\{F_1, F_2, F_3, \neg F\}$.

Mise sous forme prénexe et forme de skolem :

$$\begin{array}{rcl}
F_1 & \equiv & \forall y \forall x \forall z\ (Q(z,y) \vee R(x,y) \vee P(g(x,y))) \\
F_2 & \equiv & \forall x \forall y \forall z\ (\neg P(y) \vee \neg R(x,z) \vee Q(y,x)) \\
F_3 & \equiv & \forall x \forall y\ (\neg R(y,x) \vee P(g(z,x))) \\
\neg F & \equiv & \forall y \forall x\ (\neg P(g(y,y)) \wedge \neg Q(x,f(y)))
\end{array}$$

Correction de l'exercice

Ensemble de clause universellement quantifiées avec renommage de variable pour éviter tout conflit :

$$
\begin{array}{lcl}
C_1 & \equiv & Q(z_1, y_1) \vee R(x_1, y_1) \vee P(g(x_1, y_1)) \\
C_2 & \equiv & \neg P(y_2) \vee \neg R(x_2, z_2) \vee Q(y_2, x_2) \\
C_3 & \equiv & \neg R(y_3, x_3) \vee P(g(z_3, x_3)) \\
C_4 & \equiv & \neg P(g(y_4, y_4)) \\
C_5 & \equiv & \neg Q(x_5, f(y_5))
\end{array}
$$

Correction de l'exercice

Preuve par résolution :

$$\dfrac{\dfrac{C_1 \quad C_4}{Q(z_1, y_4) \vee R(x_4, y_4)}\ {\scriptstyle \begin{matrix} x_1 = y_4 \\ y_1 = y_4 \end{matrix}} \qquad \dfrac{C_2 \quad C_3}{\neg R(x_2, z_2) \vee Q(g(z_3, x_3), x_2) \vee \neg R(y_3, x_3)}\ {\scriptstyle y_2 = g(z_3, x_3)}}{Q(z_1, y_4) \vee Q(g(z_3, y_4), x_4)}\ {\scriptstyle \begin{matrix} x_3 = y_4 \\ x_3 = y_4 \\ y_3 = x_4 \\ x_2 = x_4 \end{matrix}}$$

puis

$$\dfrac{Q(z_1, y_4) \vee Q(g(z_3, y_4), x_4) \quad C_5}{\square}\ {\scriptstyle \begin{matrix} z_1 = g(z_3, y_4), y_4 = f(y_5) \\ x_5 = g(z_3, y_4), x_4 = f(y_5) \end{matrix}}$$

Exercice : Unification

On considère un langage $\mathcal{L}$ comprenant une fonction binaire g, une fonction unaire f, une constante a et des variables $p, r, x, y, z, t, s, u, v, w$. Indiquer pour chacun des ensembles de termes suivant s'ils sont unifiables. Si oui, donner un unificateur principal. Sinon, donner un couple de termes irréductible.

1. $\{ g(y, g(g(g(f(w), f(v)), g(g(t,a), x)), f(s))),$
   $g(f(f(r)), g(g(x, g(g(t,a), g(v,p))), y)) \}$
2. $\{ g(x, g(f(a), g(y,x))), g(g(g(v,w), g(t,s)), g(f(a), x)) \}$

Exercice : Unification

On considère un langage $\mathcal{L}$ comprenant une fonction binaire g, une fonction unaire f, une constante a et des variables $p, r, x, y, z, t, s, u, v, w$. Indiquer pour chacun des ensembles de termes suivant s'ils sont unifiables. Si oui, donner un unificateur principal. Sinon, donner un couple de termes irréductible.

1. $\{ g(y, g(g(g(f(w), f(v)), g(g(t,a), x)), f(s))),$
   $g(f(f(r)), g(g(x, g(g(t,a), g(v,p))), y)) \}$
   oui : $[f(f(r))/y, g(f(w), f(v))/x, f(r)/s, f(w)/v, f(v)/p]$
2. $\{ g(x, g(f(a), g(y,x))), g(g(g(v,w), g(t,s)), g(f(a), x)) \}$

Exercice : Unification

On considère un langage $\mathcal{L}$ comprenant une fonction binaire g, une fonction unaire f, une constante a et des variables $p, r, x, y, z, t, s, u, v, w$. Indiquer pour chacun des ensembles de termes suivant s'ils sont unifiables. Si oui, donner un unificateur principal. Sinon, donner un couple de termes irréductible.

1. $\{ g(y, g(g(g(f(w), f(v)), g(g(t, a), x)), f(s))), g(f(f(r)), g(g(x, g(g(t, a), g(v, p))), y)) \}$
   oui : $[f(f(r))/y, g(f(w), f(v))/x, f(r)/s, f(w)/v, f(v)/p]$
2. $\{ g(x, g(f(a), g(y, x))), g(g(g(v, w), g(t, s)), g(f(a), x)) \}$
   non, car il faudrait unifier $g(y, x)$ avec x

Exercice : preuve par résolution

Monsieur Dupond possède un élevage de lapins dans lequel

- certains lapins sont blancs à grandes oreilles et n'ont que des enfants blancs,
- les lapins à grandes oreilles qui n'ont pas de dents cariées ont toujours au moins un enfant aux yeux bleus,
- un lapin blanc n'a jamais de carie.

Montrer que M. Dupond a bon espoir de trouver dans son élevage un lapin blanc aux yeux bleus. Utiliser pour cela une preuve par réfutation.

Certains lapins sont blancs à grandes oreilles et n'ont que des enfants blancs :

$$F_1 = \exists x\,(\mathrm{blanc}(x) \wedge \mathrm{grandO}(x) \wedge \forall y\,(\mathrm{enfant}(x,y) \Rightarrow \mathrm{blanc}(y)))$$

Les lapins à grandes oreilles qui n'ont pas de dents cariées ont toujours au moins un enfant aux yeux bleus :

$$F_2 = \forall x\,((\mathrm{grandO}(x) \wedge \neg\,\mathrm{dentsC}(x)) \Rightarrow \exists y\,(\mathrm{enfant}(x,y) \wedge \mathrm{yeuxB}(y)$$

Un lapin blanc n'a jamais de carie :

$$F_3 = \forall x\,(\mathrm{blanc}(x) \Rightarrow \neg\,\mathrm{dentsC}(x))$$

M. Dupond a bon espoir de trouver dans son élevage un lapin blanc aux yeux bleus :

$$\neg F_4 = \exists x\,(\mathrm{blanc}(x) \wedge \mathrm{yeuxB}(x))$$

On met toutes ces formules sous une forme prénexe conjonctive :

$$
\begin{aligned}
F_1 &\equiv \exists x \forall y\, ((\mathrm{blanc}(x) \wedge \mathrm{grandO}(x) \wedge (\neg\, \mathrm{enfant}(x,y)) \vee \mathrm{blanc}(y))) \\
F_2 &\equiv \forall x \exists y\ \ ((\neg\, \mathrm{grandO}(x) \vee \mathrm{dentsC}(x) \vee \mathrm{enfant}(x,y)) \\
&\qquad\qquad \wedge (\neg\, \mathrm{grandO}(x) \vee \mathrm{dentsC}(x) \vee \mathrm{yeuxB}(y))) \\
F_3 &\equiv \forall x\, (\neg\, \mathrm{blanc}(x) \vee \neg\, \mathrm{dentsC}(x)) \\
F_4 &\equiv \forall x\, (\neg\, \mathrm{blanc}(x) \vee \neg\, \mathrm{yeuxB}(x))
\end{aligned}
$$

Mises sous forme de Skolem, ces formules donnent les clauses :

$$
\begin{array}{lcc}
F_1 & \rightarrow & \left\{ \begin{array}{c} \mathrm{blanc}(a) \\ \mathrm{grandO}(a) \\ \neg\,\mathrm{enfant}(a,y) \vee \mathrm{blanc}(y) \end{array} \right\} \\
F_2 & \rightarrow & \left\{ \begin{array}{c} \neg\,\mathrm{grandO}(x_1) \vee \mathrm{dentsC}(x_1) \vee \mathrm{enfant}(x_1, f(x_1)) \\ \neg\,\mathrm{grandO}(x_2) \vee \mathrm{dentsC}(x_2) \vee \mathrm{yeuxB}(f(x_2)) \end{array} \right\} \\
F_3 & \rightarrow & \neg\,\mathrm{blanc}(z) \vee \neg\,\mathrm{dentsC}(z) \\
F_4 & \rightarrow & \neg\,\mathrm{blanc}(t) \vee \neg\,\mathrm{yeuxB}(t)
\end{array}
$$

On dérive ensuite la clause vide de cet ensemble de clauses (preuve par réfutation).

$$\frac{\neg\,\text{blanc}(t) \lor \neg\,\text{yeuxB}(t) \qquad \neg\,\text{grandO}(x_2) \lor \text{dentsC}(x_2) \lor \text{yeuxB}(f(x_2))}{\neg\,\text{blanc}(f(x_2)) \lor \neg\,\text{grandO}(x_2) \lor \text{dentsC}(x_2)}$$

$$\frac{\neg\,\text{blanc}(f(x_2)) \lor \neg\,\text{grandO}(x_2) \lor \text{dentsC}(x_2) \qquad \neg\,\text{enfant}(a,y) \lor \text{blanc}(y)}{\neg\,\text{enfant}(a,f(x_2)) \lor \neg\,\text{grandO}(x_2) \lor \text{dentsC}(x_2)}$$

$$\frac{\neg\,\text{enfant}(a,f(x_2)) \lor \neg\,\text{grandO}(x_2) \lor \text{dentsC}(x_2) \qquad \neg\,\text{grandO}(x_1) \lor \text{dentsC}(x_1) \lor \text{enfant}(x_1,f(x_1))}{\neg\,\text{grandO}(a) \lor \text{dentsC}(a)}$$

$$\frac{\text{grandO}(a) \qquad \neg\,\text{grandO}(a) \lor \text{dentsC}(a)}{\text{dentsC}(a)}$$

$$\frac{\text{dentsC}(a) \qquad \neg\,\text{blanc}(z) \lor \neg\,\text{dentsC}(z)}{\neg\,\text{blanc}(a)}$$

$$\frac{\neg\,\text{blanc}(a) \qquad \text{blanc}(a)}{\square}$$

Programmation logique (pure)

Programmation logique : définitions

Une *clause de Horn* est une clause dont au plus une formule atomique est positive (*i.e.* qui apparaît sans négation). Dans le cas où la clause a exactement une formule atomique positive, on parle de *clause définie* et on la note

$$A \leftarrow A_1, A_2, \ldots, A_n$$

au lieu de

$$A \Leftarrow A_1 \wedge A_2 \wedge \ldots \wedge A_n$$

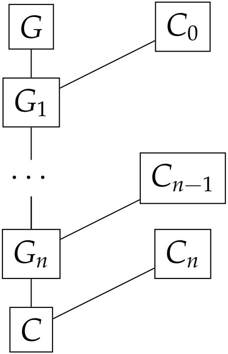
où les $A_1, A_2, \ldots, A_n, A$ sont des formules atomiques. Dans le cas contraire, on parle de *clause négative*.

Programmation logique : définitions

Un *programme logique* est un ensemble fini de clauses définies.

$$P = \{C_0, \ldots, C_n\}$$

Soit P un programme logique, G une clause négative, nommée *but*, et C une clause. Une preuve de C par résolution à partir de P et G est dite *LD* (linéaire définie) si à chaque étape le résolvant est obtenu en utilisant une clause de P, et si la première étape de résolution utilise le but G.



Propriétés de la résolution LD

Proposition

Soit P un programme logique et G une clause négative. $P \cup \{G\}$ est contradictoire ssi il existe une réfutation LD de G à partir de P.

La résolution *LD* apporte plus d'information que la simple contradiction d'un but G à partir d'un programme logique P. On appelle *substitution réponse* la composition successive des différentes substitutions réalisées dans l'arbre de preuve.

Proposition

La substitution réponse σ associée à la réfutation LD de G par P est telle que la clôture universelle de $\sigma(\neg G)$ est conséquence sémantique de P.

PROLOG

Le langage PROLOG est fondé sur ces principes. Un programme PROLOG est un programme logique, donc une liste de clauses définies. On peut considérer l'exemple suivant :

homme(jacques).
homme(julien).
homme(aymeric).
homme(françois).
homme(didier).
femme(brigitte).
femme(martine).
femme(vanessa).
parent(jacques,julien).
parent(jacques,aymeric).
parent(brigitte,julien).
parent(brigitte,aymeric).
parent(martine,françois).

parent(didier,vanessa).
soeur(martine,brigitte).
soeur(brigitte,martine).
soeur(martine,didier).
soeur(brigitte,didier).
fils(x,y) $\leftarrow$ parent(y,x),homme(x)
fille(x,y) $\leftarrow$ parent(y,x),femme(x)
frère(x,y) $\leftarrow$ soeur(y,x),homme(x)
cousin(x,y) $\leftarrow$ fils(x,t),soeur(t,z),parent(z,y)
cousin(x,y) $\leftarrow$ fils(x,t),frère(t,z),parent(z,y)
cousine(x,y) $\leftarrow$ fille(x,t),frère(t,z),parent(z,y)
cousine(x,y) $\leftarrow$ fille(x,t),soeur(t,z),parent(z,y)

Prolog

L'utilisateur peut ensuite poser des questions correspondant à un but $L_1 \wedge \ldots \wedge L_m$ avec la syntaxe

$$> \; L_1, \ldots, L_m$$

Prolog recherche alors toutes les réfutations du but G par résolution LD. S'il y parvient, il affiche la liste des substitutions réponses obtenues. Ainsi à la question

$$> \; \text{cousin(françois,}x\text{)}$$

Prolog répond

$$\{x = \text{julien}\}\{x = \text{aymeric}\}\{x = \text{vanessa}\}$$

La stratégie de résolution de Prolog est la suivante : on cherche à éliminer l'atome le plus à gauche dans le but courant avec l'une des clauses du programme (en suivant l'ordre d'écriture). La recherche s'effectue en profondeur d'abord.

$$\boxed{\neg\mathrm{cousin}(\mathrm{francois}, x)}$$

$\boxed{\neg\text{cousin}(\text{francois}, x)}$ $\boxed{\text{cousin}(x', y), \neg\text{fils}(x', t), \neg\text{soeur}(t, z), \neg\text{parent}(z, y)}$

$\neg\text{cousin}(\text{francois}, x)$

$\text{cousin}(x', y), \neg\text{fils}(x', t), \neg\text{soeur}(t, z), \neg\text{parent}(z, y)$

$[\text{francois}/x', x/y]$

$\neg\text{fils}(\text{francois}, t), \neg\text{soeur}(t, z), \neg\text{parent}(z, x)$

$\neg \text{cousin}(\text{francois}, x)$

$\text{cousin}(x', y), \neg \text{fils}(x', t), \neg \text{soeur}(t, z), \neg \text{parent}(z, y)$

$[\text{francois}/x', x/y]$

$\neg \text{fils}(\text{francois}, t), \neg \text{soeur}(t, z), \neg \text{parent}(z, x)$

$\text{fils}(x', y), \neg \text{parent}(y, x'), \neg \text{homme}(x')$

$\neg\text{cousin}(\text{francois}, x)$

$\text{cousin}(x', y), \neg\text{fils}(x', t), \neg\text{soeur}(t, z), \neg\text{parent}(z, y)$

$[\text{francois}/x', x/y]$

$\neg\text{fils}(\text{francois}, t), \neg\text{soeur}(t, z), \neg\text{parent}(z, x)$

$\text{fils}(x', y), \neg\text{parent}(y, x'), \neg\text{homme}(x')$

$[\text{francois}/x', t/y]$

$\neg\text{parent}(t, \text{francois}), \neg\text{homme}(\text{francois}),$
$\neg\text{soeur}(t, z), \neg\text{parent}(z, x)$

$\neg\text{cousin}(\text{francois}, x)$

$\text{cousin}(x', y), \neg\text{fils}(x', t), \neg\text{soeur}(t, z), \neg\text{parent}(z, y)$

$[\text{francois}/x', x/y]$

$\neg\text{fils}(\text{francois}, t), \neg\text{soeur}(t, z), \neg\text{parent}(z, x)$

$\text{fils}(x', y), \neg\text{parent}(y, x'), \neg\text{homme}(x')$

$[\text{francois}/x', t/y]$

$\neg\text{parent}(t, \text{francois}), \neg\text{homme}(\text{francois}),$
$\neg\text{soeur}(t, z), \neg\text{parent}(z, x)$

$\text{parent}(\text{martine}, \text{francois})$

$\neg\mathrm{cousin}(\mathrm{francois}, x)$

$\mathrm{cousin}(x', y), \neg\mathrm{fils}(x', t), \neg\mathrm{soeur}(t, z), \neg\mathrm{parent}(z, y)$

$[\mathrm{francois}/x', x/y]$

$\neg\mathrm{fils}(\mathrm{francois}, t), \neg\mathrm{soeur}(t, z), \neg\mathrm{parent}(z, x)$

$\mathrm{fils}(x', y), \neg\mathrm{parent}(y, x'), \neg\mathrm{homme}(x')$

$[\mathrm{francois}/x', t/y]$

$\neg\mathrm{parent}(t, \mathrm{francois}), \neg\mathrm{homme}(\mathrm{francois}),$
$\neg\mathrm{soeur}(t, z), \neg\mathrm{parent}(z, x)$

$\mathrm{parent}(\mathrm{martine}, \mathrm{francois})$

$[\mathrm{martine}/t]$

$\neg\mathrm{homme}(\mathrm{francois}),$
$\neg\mathrm{soeur}(\mathrm{martine}, z), \neg\mathrm{parent}(z, x)$

$\neg$cousin(francois, x)

cousin(x', y), $\neg$fils(x', t), $\neg$soeur(t, z), $\neg$parent(z, y)

[francois/x', x/y]

$\neg$fils(francois, t), $\neg$soeur(t, z), $\neg$parent(z, x)

fils(x', y), $\neg$parent(y, x'), $\neg$homme(x')

[francois/x', t/y]

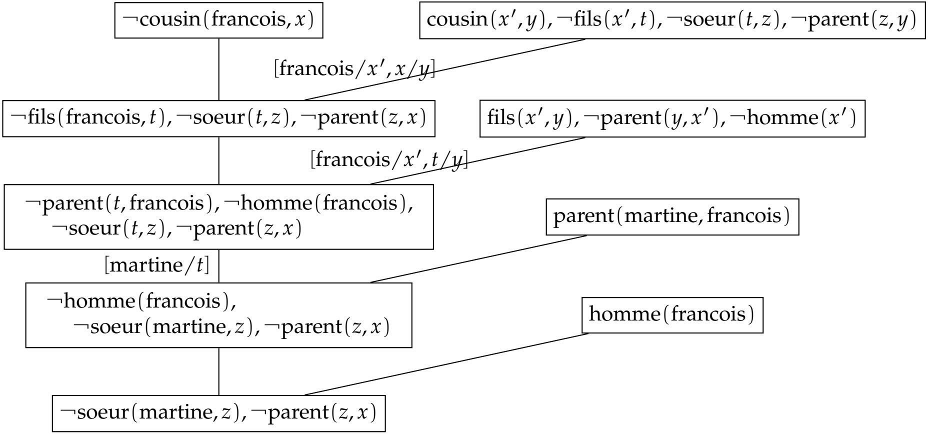
$\neg$parent(t, francois), $\neg$homme(francois),
$\neg$soeur(t, z), $\neg$parent(z, x)

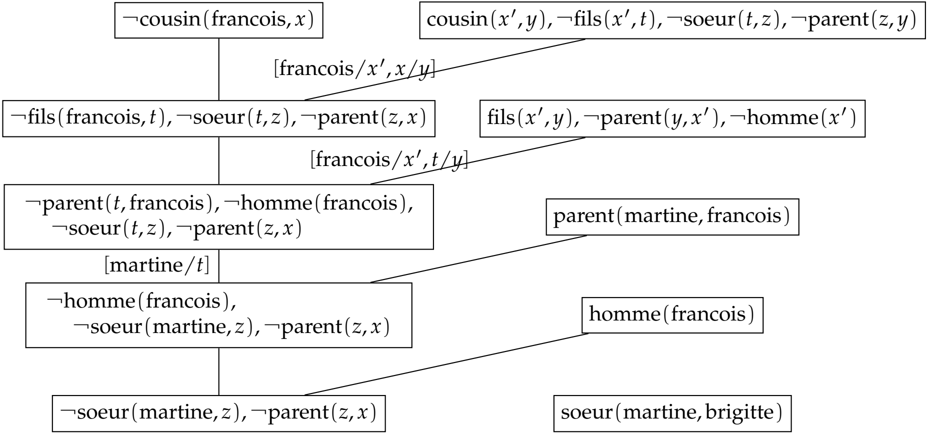
parent(martine, francois)

[martine/t]

$\neg$homme(francois),
$\neg$soeur(martine, z), $\neg$parent(z, x)

homme(francois)


¬cousin(francois, x)
cousin(x′, y), ¬fils(x′, t), ¬soeur(t, z), ¬parent(z, y)
[francois/x′, x/y]
¬fils(francois, t), ¬soeur(t, z), ¬parent(z, x)
fils(x′, y), ¬parent(y, x′), ¬homme(x′)
[francois/x′, t/y]
¬parent(t, francois), ¬homme(francois), ¬soeur(t, z), ¬parent(z, x)
parent(martine, francois)
[martine/t]
¬homme(francois), ¬soeur(martine, z), ¬parent(z, x)
homme(francois)
¬soeur(martine, z), ¬parent(z, x)


¬cousin(francois, x)
cousin(x′, y), ¬fils(x′, t), ¬soeur(t, z), ¬parent(z, y)
[francois/x′, x/y]
¬fils(francois, t), ¬soeur(t, z), ¬parent(z, x)
fils(x′, y), ¬parent(y, x′), ¬homme(x′)
[francois/x′, t/y]
¬parent(t, francois), ¬homme(francois), ¬soeur(t, z), ¬parent(z, x)
parent(martine, francois)
[martine/t]
¬homme(francois), ¬soeur(martine, z), ¬parent(z, x)
homme(francois)
¬soeur(martine, z), ¬parent(z, x)
soeur(martine, brigitte)

$\neg\text{cousin}(\text{francois}, x)$ $\quad$ $\text{cousin}(x', y), \neg\text{fils}(x', t), \neg\text{soeur}(t, z), \neg\text{parent}(z, y)$

$[\text{francois}/x', x/y]$

$\neg\text{fils}(\text{francois}, t), \neg\text{soeur}(t, z), \neg\text{parent}(z, x)$ $\quad$ $\text{fils}(x', y), \neg\text{parent}(y, x'), \neg\text{homme}(x')$

$[\text{francois}/x', t/y]$

$\neg\text{parent}(t, \text{francois}), \neg\text{homme}(\text{francois}), \neg\text{soeur}(t, z), \neg\text{parent}(z, x)$ $\quad$ $\text{parent}(\text{martine}, \text{francois})$

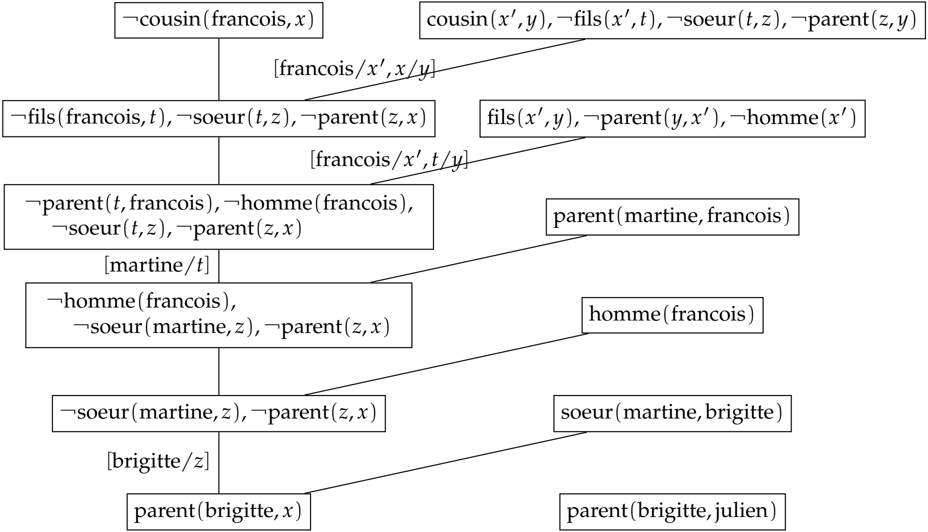
$[\text{martine}/t]$

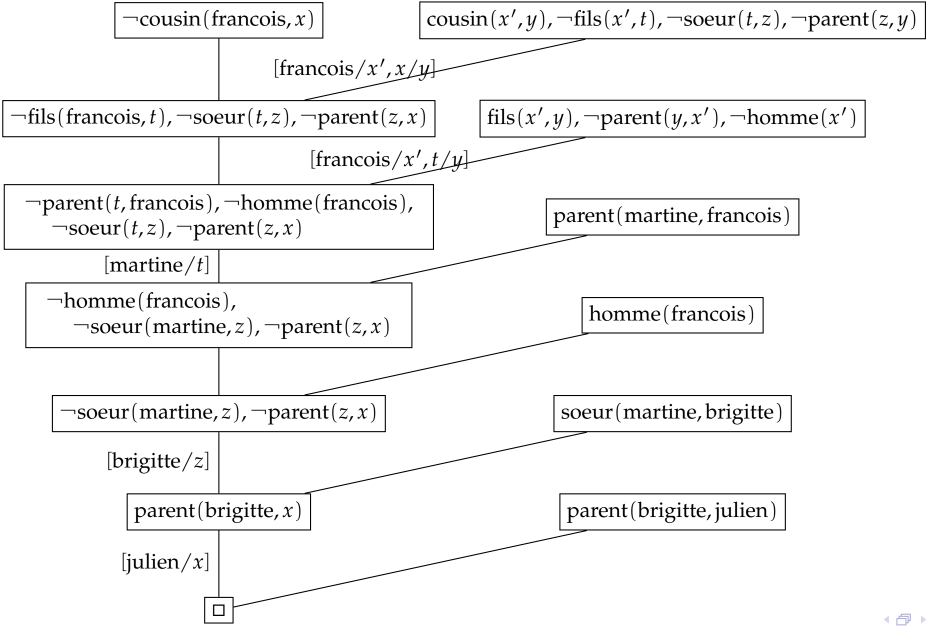
$\neg\text{homme}(\text{francois}), \neg\text{soeur}(\text{martine}, z), \neg\text{parent}(z, x)$ $\quad$ $\text{homme}(\text{francois})$

$\neg\text{soeur}(\text{martine}, z), \neg\text{parent}(z, x)$ $\quad$ $\text{soeur}(\text{martine}, \text{brigitte})$

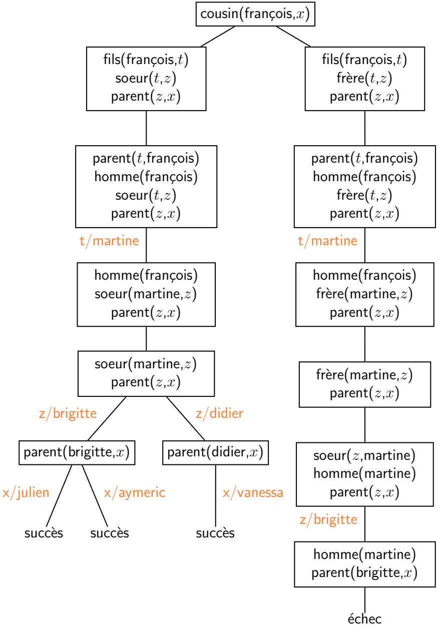
$[\text{brigitte}/z]$

$\text{parent}(\text{brigitte}, x)$


¬cousin(francois, x)
cousin(x′, y), ¬fils(x′, t), ¬soeur(t, z), ¬parent(z, y)
[francois/x′, x/y]
¬fils(francois, t), ¬soeur(t, z), ¬parent(z, x)
fils(x′, y), ¬parent(y, x′), ¬homme(x′)
[francois/x′, t/y]
¬parent(t, francois), ¬homme(francois), ¬soeur(t, z), ¬parent(z, x)
parent(martine, francois)
[martine/t]
¬homme(francois), ¬soeur(martine, z), ¬parent(z, x)
homme(francois)
¬soeur(martine, z), ¬parent(z, x)
soeur(martine, brigitte)
[brigitte/z]
parent(brigitte, x)
parent(brigitte, julien)


¬cousin(francois, x)
cousin(x′, y), ¬fils(x′, t), ¬soeur(t, z), ¬parent(z, y)
[francois/x′, x/y]
¬fils(francois, t), ¬soeur(t, z), ¬parent(z, x)
fils(x′, y), ¬parent(y, x′), ¬homme(x′)
[francois/x′, t/y]
¬parent(t, francois), ¬homme(francois), ¬soeur(t, z), ¬parent(z, x)
parent(martine, francois)
[martine/t]
¬homme(francois), ¬soeur(martine, z), ¬parent(z, x)
homme(francois)
¬soeur(martine, z), ¬parent(z, x)
soeur(martine, brigitte)
[brigitte/z]
parent(brigitte, x)
parent(brigitte, julien)
[julien/x]
□

Exemple


cousin(françois,x)
fils(françois,t) soeur(t,z) parent(z,x)
fils(françois,t) frère(t,z) parent(z,x)
parent(t,françois) homme(françois) soeur(t,z) parent(z,x)
parent(t,françois) homme(françois) frère(t,z) parent(z,x)
t/martine
t/martine
homme(françois) soeur(martine,z) parent(z,x)
homme(françois) frère(martine,z) parent(z,x)
soeur(martine,z) parent(z,x)
frère(martine,z) parent(z,x)
z/brigitte
z/didier
parent(brigitte,x)
parent(didier,x)
soeur(z,martine) homme(martine) parent(z,x)
x/julien
x/aymeric
x/vanessa
succès
succès
succès
z/brigitte
homme(martine) parent(brigitte,x)
échec

Compter avec Prolog

```
int(o).
int(s(X)) <-- int(X).

add(o,X,X) <-- int(X).
add(s(X),Y,s(Z)) <-- add(X,Y,Z).

mult(o,X,o) <-- int(X).
mult(s(X),Y,Z) <-- add(Z1,Y,Z),mult(X,Y,Z1).
```

Compter avec Prolog

```
But ? add(3,4,Z)
  { Z = s(s(s(s(s(s(s(o))))))) }

But ? add(X,Y,4)
  { Y = s(s(s(s(o)))), X = o }
  { Y = s(s(s(o))), X = s(o) }
  { Y = s(s(o)), X = s(s(o)) }
  { Y = s(o), X = s(s(s(o))) }
  { Y = o, X = s(s(s(s(o)))) }
```

Plan