Le calcul de la résolution

Luigi Santocanale

Cours Logique, Déduction, Programmation 2005–2006

1 Termes et substitutions

Définition 1.1 Une *signature* est un couple (Ω, ar), où Ω est un ensemble de symboles de fonction et $ar : \Omega \longrightarrow N$. Pour $f \in \Omega$, $ar(f)$ est l'*arité* de f.

Définition 1.2 Étant donné une signature Ω et un ensemble de variables X, l'ensemble $\mathcal{T}(\Omega, X)$ des termes sur Ω et X est défini par induction comme il suit :
– si $x \in X$, alors $x \in \mathcal{T}(\Omega, X)$,
– si $f \in \Omega$, $ar(f) = n$ et $t_1, \ldots, t_n \in \mathcal{T}(\Omega, X)$, alors $f(t_1, \ldots, t_n) \in \mathcal{T}(\Omega, X)$.

Définition 1.3 Une substitution est une fonction $\sigma : X \longrightarrow \mathcal{T}(\Omega, X)$ de support fini, c.-à-d. telle que l'ensemble

$$\{\, x \in X \,|\, \sigma(x) \neq x \}$$

est fini.

Définition 1.4 Par induction, on étend une substitution $\sigma : X \longrightarrow \mathcal{T}(\Omega, X)$ à une fonction $\tilde{\sigma} : \mathcal{T}(\Omega, X) \longrightarrow \mathcal{T}(\Omega, X)$:
– $\tilde{\sigma}(x) = \sigma(x)$,
– $\tilde{\sigma}(f(t_1, \ldots, t_n)) = f(\tilde{\sigma}(t_1), \ldots, \tilde{\sigma}(t_n))$.

Une fonction de la forme $\tilde{\sigma}$ est univoquement définie par la substitution σ. Nous allons donc faire abus de notation et on écrira σ à la place de $\tilde{\sigma}$ et σt à la place de $\tilde{\sigma}(t)$. Les relations

$$\widetilde{(\tilde{\tau} \circ \sigma)}(t) = \tilde{\tau}(\tilde{\sigma}(t))\,.$$

montrent que cet abus est cohérente avec la composition de fonctions. De même, on a une substitution triviale, $\iota(x) = x$ pour tout $x \in X$, et l'on s'aperçoit que pour tout $t \in \mathcal{T}(\Omega, X)$ $\tilde{\iota}(t) = t$.

L'ensemble des variables dans un terme peut être défini par induction comme il suit :
– $Var(x) = \{\, x \,\}$,
– $Var(f(t_1, \ldots, t_n)) = \bigcup_{i=1,\ldots,n} Var(t_i)$.

2 Unification

Problèmes d'unification.

Une instance du problème d'unification est un suite de couples de termes

$$P = (t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)\,.$$

où $t_i, v_i \in \mathcal{T}(\Omega, X)$.

Le problème d'unification :

Étant donné l'instance

$$(t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)\,,$$

on se demande : existe t'il une substitution σ (si oui, construire σ) telle que, pour $i = 1, \ldots, n$,

$$\sigma t_i = \sigma v_i$$

?

On appelle une substitution σ avec cette propriétés un *unificateur* (anglais : « unifier ») du problème (t_i, v_i), $i = 1, \ldots, n$. Évidemment si σ est un tel unificateur, alors il en est de même pour $\psi \circ \sigma$, où ψ est une substitution quelconque.

Recherche d'une solution à un problème d'unification.

Considérons un problème d'unification

$$(t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)\,.$$

Pour trouver un unificateur on peut procéder, de façon recursive, comme il suit :

Si $n = 0$, alors la substitution ι est bien un unificateur.
Supposons que $n \geq 1$, choisissons une couple (t_i, v_i), nous allons assumer que $i = 1$:

1. Si $t_1 = v_1 = x \in X$, alors retourner un unificateur de $(t_2, v_2), \ldots, (t_n, v_n)$.
2. Si le terme t_1 est une variable $x \notin Var(v_1)$, posons

$$\sigma(z) = \begin{cases} t_1, & \text{si } z = x \\ z, & \text{sinon.} \end{cases}$$

   Soit τ un unificateur de $(\sigma t_2, \sigma v_2) \ldots (\sigma t_n, \sigma v_n)$, on retourne $\tau \circ \sigma$.
3. Le cas symétrique : si $v_1 = x \notin Var(t_1)$, on definit σ comme en haut – avec v_1 à la place de t_1 – et on retourne $\tau \circ \sigma$, τ étant un unificateur de $(\sigma t_2, \sigma v_2) \ldots (\sigma t_n, \sigma v_n)$.
4. Sinon, soit $t_1 = f(a_1, \ldots, a_m)$ et $v_1 = g(b_1, \ldots, b_k)$. Si $f = g$ et $n = m$, alors calculer alors un unificateur de

$$(a_1, b_1), \ldots, (a_m, b_m), (t_2, v_2), \ldots, (t_n, v_n)\,.$$

Exercice.
– Que se passe t-il si une des conditions décrites ci-dessus n'est pas vérifié ?
– On vient de décrire, de façon informelle, un algorithme récursif pour calculer un unificateur. Montrer que cet algorithme se termine. Pour cela il faudra définir la taille d'un instance, et montrer qu'à chaque appel récursif la taille de l'instance est plus petite.

Posons $\sigma \leq \tau$ s'il existe une substitution ψ telle que $\tau = \psi \circ \sigma$: on dira alors que σ est plus générale que τ. On peut démontrer que si $\tau \leq \sigma$ et $\sigma \leq \tau$, c.à-d. $\sigma = \psi_1 \circ \tau$ et $\tau = \psi_2 \circ \sigma$, alors les ψ_i sont des rénommages des variables : $\psi_i(x) \in X$ pour tout $x \in X$.

Proposition 2.1 Soit $(t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)$ un problème d'unification. Si ce problème admet une solution, alors il admet une solution plus générales : il existe une solution σ telle que pour toute autre solution τ on a $\tau = \psi \circ \sigma$.

L'algorithme décrit ci-dessus produit un unificateur plus générale.

3 Résolution au premier ordre

Un langage au premier ordre $\mathcal{L}$ est un couple de signatures $ar_\Omega : \Omega \longrightarrow N$ et $ar_\Pi : \Pi \longrightarrow N$. Les éléments de Ω sont les symboles de fonction, les éléments de Π sont les symboles de prédicat. Si $P \in \Pi$ et $ar_\Pi(P) = n$, on dira que P est un symbole de prédicat n-aire.

Définition 3.1 – Une *formule atomique* est de la forme

$$P(t_1, \ldots, t_n)$$

où P est un symbole de prédicat n-aire et $t_i \in \mathcal{T}(\Omega, X)$ pour $i = 1, \ldots, n$.

– Un *littéral* est ou bien une formule atomique, ou bien la négation d'une formule atomique.
– Si $L_1, \ldots, L_n$ sont des littéraux, alors

$$\forall x_1, \ldots, x_k\ L_1 \vee \ldots \vee L_n \tag{1}$$

est une *clause*.
Ici $x_1, \ldots, x_k$ est la liste des variables occurrentes dans un terme dans littéral L_i. On dit que la (1) est la fermeture universelle de la disjonction $L_1 \vee \ldots \vee L_n$.
– Une *théorie* est un ensemble de clauses.

Remarquons qu'une clause peut être (la fermeture universelle d') une disjonction de 0-littéraux : on parle alors de la clause vide, que l'on dénote par $\perp$.

Exemple. Soit $\Omega = \{0, s, +\}$ où $ar(0) = 0, ar(s) = 1, ar(+) = 2$ et $\Pi = \{P\}$, où P est un symbole de prédicat binaire.

Les suivantes sont des formules atomiques :

$$P(0(), s(x)),\quad P(+(x, s(0())), 0())$$

Les suivantes sont des littéraux :

$$P(x, y),\quad \neg P(s(y), 0())$$

La suivante est une clause :

$$\forall x \forall y (P(x, y) \vee \neg P(s(y), 0()))$$

La suivante est une théorie

$$\{\, \forall x (P(0(), x)),\ \forall x \forall y (P(x, y) \vee P(y, x)) \,\}$$

Nous allons utiliser une notation abrégée et infixe si cela n'engendre des problèmes. Ainsi, nos exemples deviennent :
– Formules atomiques :

$$P(0, s(x)),\quad P(x + s(0), 0)$$

– Littéraux :

$$P(x, y),\quad \neg P(s(y), 0)$$

– Une clause :

$$P(x, y) \vee \neg P(s(y), 0)$$

– Une théorie :

$$\{\, P(0, x),\ P(x, y) \vee P(y, x) \,\}$$

On remarque que, dans la notation abrégée d'une clause, on omet d'écrire les quantificateurs. En plus, les formules

$$\forall x \phi(x) \quad \forall y \phi(y)$$

sont logiquement équivalentes, on peut supposer que dans une théorie de la forme

$$\{ \gamma_1, \ldots, \gamma_n, \ldots \}$$

où les γ_i sont des clauses, on a que les variables de γ_i sont disjointes des variables de γ_j si $i \neq j$.

On étend la définition d'application d'une substitution aux formules atomiques, aux littéraux et aux clauses :
- Formule atomique : $\sigma P(t_1, \ldots, t_n) = P(\sigma t_1, \ldots, \sigma t_n)$,
- Littéral :
  - littéral positif, comme la formule atomique, $\sigma P(t_1, \ldots, t_n) = P(\sigma t_1, \ldots, s\sigma t_n)$,
  - littéral négatif : $\sigma \neg P(t_1, \ldots, t_n) = \neg P(\sigma t_1, \ldots, s\sigma t_n)$.
- Clause : $\sigma(L_1 \vee \ldots \vee L_n) = \sigma L_1 \vee \ldots \vee \sigma L_n$.

En principe, on peut étendre la définition de la substitution à une formule quelconque.

Définition 3.2 Soient A et B deux formules atomiques. On dit que σ est un unificateur principal de A et B ssi
- $A = P(t_1, \ldots, t_n)$, $B = Q(s_1, \ldots, s_m)$,
- $P = Q$ (et donc $n = m$), et
- σ est un unificateur principal du problème

$$(t_1, s_1), \ldots (t_n, s_n) \, .$$

Si σ est un unificateur principal de A et B, alors

$$\sigma A \;\; = \;\; \sigma B \, .$$

3.1 Modèles et validitée

Rappelons la notions de validité dans un modèle du langage $\mathcal{L}$. D'abord un modèle du langage $\mathcal{L}$ est un triplet $\langle \, U \, , \, \| \cdot \|_\Omega \, , \, \| \cdot \|_\Pi \, \rangle$, où
- U est un ensemble,
- $\|f\|_\Omega : U^n \longrightarrow U$ est une fonction n-aire pour tout symbole de fonction $f \in \Omega$ telle que $ar_\Omega(f) = n$,
- $\|P\|_\Pi \subseteq U^n$ pour tout symbole de prédicat $P \in \Pi$ tel que $ar_\Pi(P) = n$.

Étant donné un modèle $\langle U, \| \cdot \|_\Omega, \| \cdot \|_\Pi \rangle$ et une fonction $I : X \longrightarrow U$, on peut définir l'interprétation de tout terme $t \in \mathcal{T}(\Omega, X)$, notée $\|t\|_I$, comme il suit :
- si $t = x$, alors $\|t\|_I = I(x)$,
- si $t = f(t_1, \ldots, t_n)$ alors $\|f(t_1, \ldots, t_n)\|_I = \|f\|_\Omega(\|t_1\|, \ldots, \|t_n\|)$.

Remarquons que $\|t\|_I \in U$.

Nous allons dire qu'une formule atomique $P(t_1, \ldots, t_n)$ est vraie dans un modèle $\mathcal{M} = \langle U, \| \cdot \|_\Omega, \| \cdot \|_\Pi \rangle$, par rapport à une fonction $I : X \longrightarrow U$, si et seulement si

$$(\|t_1\|_I, \ldots, \|t_n\|_I) \in \|P\|_\Pi \, .$$

Nous noterons cette relation par $\mathcal{M} \models_I P(t_1, \ldots, t_n)$. Nous allons dire que la negation d'une formule atomique $\neg P(t_1, \ldots, t_n)$ est vraie dans un modèle $\mathcal{M}$ par rapport à une fonction $I : X \longrightarrow U$ si et seulement si $\mathcal{M} \not\models_I P(t_1, \ldots, t_n)$.

Définition 3.3 Une clause $C = L_1 \vee \ldots \vee L_k$ est vraie dans un modèle $\mathcal{M}$ – relation notée $\mathcal{M} \models C$ – ssi il pour tout $I : X \longrightarrow U$ il existe $j \in \{ 1, \ldots, k \}$ tel que $\mathcal{M} \models_I L_j$. Une théorie T est vraie dans un modele $\mathcal{M}$ – relation notée $\mathcal{M} \models T$ - ssi, pour tout $C \in T$, $\mathcal{M} \models C$.

Rappelons que une théorie T n'est pas cohérente ssi il n'existe pas un modèle $\mathcal{M}$ tel que $\mathcal{M} \models T$. Par la complètude de la logique du premier ordre, cela revient à dire qu'à l'aide des inférences logiques usuelles on peut dériver une contradiction à partir de l'ensemble T.

3.2 Le calcul R

Le calcul de la résolution a été introduit par Robinson en 1965. Son but, à une première approximation, est de pouvoir déduire des nouvelles clauses à partir de clauses données, de façon que l'inférence ainsi faite soit valide. La propriété fondamentale de ce calcul est sa complétude : si une ensemble de clauses n'est pas cohérent, alors on peut dériver la clause vide à l'aide des règles de ce calcul.

Pour présenter ce calcul, on va se représenter les clauses sous une forme particulière. Considérons par exemple la clause :

$$\neg P_1 \vee \ldots \vee \neg P_n \vee Q_1 \vee \ldots \vee Q_m \, .$$

On pose alors $\Gamma = P_1, \ldots, P_n$ et $\Delta = Q_1, \ldots Q_m$ et on dénotera cette clause par $\Gamma \Rightarrow \Delta$.

En général une clause contient des littéraux positifs, pas préfixés par la négation, et des littéraux négatifs, préfixés par la négation. On regroupe à gauche du symbole $\Rightarrow$ la forme positive des littéraux négatifs, et à sa droite les littéraux positifs. On peut donc supposer que Γ et Δ sont deux ensembles de formules atomiques. Avec la notation Δ, A on dénotera l'ensemble union $\Delta \cup \{ A \}$.

Remarquons que si $C = \Gamma \Rightarrow \Delta$, alors $\mathcal{M} \models C$ ssi pour tout $I : X \longrightarrow U$, si pour tout $\gamma \in \Gamma$ $\mathcal{M} \models_I \gamma$, alors il existe $\delta \in \Delta$ tel que $\mathcal{M} \models_I \delta$.

Les règles du calcul sont deux :

Règle de résolution :

$$\frac{\Gamma_1 \Rightarrow \Delta_2, A \qquad B, \Gamma_2 \Rightarrow \Delta_2}{\sigma(\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2)} \ \sigma \text{ un unificateur principal de } A \text{ et } B$$

Règle de factorisation (à droite) :

$$\frac{\Gamma \Rightarrow \Delta, A, B}{\sigma(\Gamma \Rightarrow \Delta, A)} \ \sigma \text{ un unificateur principal de } A \text{ et } B$$

Proposition 3.4 Le calcul de la résolution est valide : si une théorie est vraie dans un modèle $\mathcal{M}$, si $\{ C_1, \ldots, C_n \} \subseteq T$, et si

$$\frac{C_1, \ldots, C_n}{C_0}$$

est une règle d'inférence du calcul, alors C est aussi vraie dans le modèle $\mathcal{M}$.

Proof. En effet, si $\forall x \phi(x)$ est vrai dans un modèle, alors en est de même pour $\phi(t)$ pour n'importe quel terme t. Par conséquent, si $\mathcal{M} \models C$ alors $\mathcal{M} \models \sigma C$.

On peut alors justifier la règle de résolution par

$$\frac{\dfrac{\Gamma_1 \Rightarrow \Delta_2, A}{\sigma\Gamma_1 \Rightarrow \sigma\Delta_2, \sigma A} \quad \dfrac{B, \Gamma_2 \Rightarrow \Delta_2}{\sigma B, \sigma\Gamma_2 \Rightarrow \sigma\Delta_2}}{\sigma\Gamma_1\sigma\Gamma_2 \Rightarrow \sigma\Delta_1, \sigma\Delta_2} \sigma A = \sigma B \text{ , inférence de la logique propositionnelle}$$

et la règle de factorisation par

$$\frac{\dfrac{\Gamma \Rightarrow \Delta, A, B}{\sigma\Gamma \Rightarrow \sigma\Delta, \sigma A, \sigma B}}{\sigma\Gamma \Rightarrow \sigma\Delta, \sigma A} \sigma A = \sigma B \text{ , inférence de la logique propositionnelle}$$

□

Définition 3.5 On dit qu'un calcul est complet si étant donnée une théorie incohérente T, il existe une preuve dans ce calcul de la clause vide $\perp$.

On remarque que la clause vide est représentée dans ce calcul par le symbole $\Rightarrow$ la clause n'ayant aucun littéral positif ni négatif.

Proposition 3.6 Le calcul R est complet.

Cette propriété ne sera pas démontrée ici.

Exemple. Considérons la théorie composée par

$$R(x, s(x)) \Rightarrow Q(f(x)) \quad (1)$$
$$\Rightarrow R(c, y) \quad (2)$$
$$Q(f(y)) \Rightarrow \quad (3)$$

On obtient une preuve de la clause vide comme il suit :

$$R(x, s(x)) \Rightarrow \quad (4, \text{Res } 1.2 ; 3.1)$$
$$\Rightarrow \quad (5, \text{Res } 2.1 ; 4.1)$$

On a obtenu la clause $R(x, s(x)) \Rightarrow$ par application de la règle de résolution à la deuxième formule atomique de la première clause, c.-à-d. à $Q(f(x))$, avec la première formule atomique de la première troisième clause, c.-à-d la formule $Q(f(y))$. Un unificateur principal est évidemment lsa substitution $\sigma : \{ y \to x \}$. On déduit $R(x, s(x)) \Rightarrow$ comme quatrième clause.

On applique une deuxième fois la règle de résolution à $R(c, y)$, la première formule atomique de la deuxième clause, et à $R(x, s(x))$, première formule atomique de la quatrième clause. Un unificateur principal est dans ce cas $\{ x \to c, y \to s(x) \}$. Par élimination de la formule atomique $R(c, s(y))$ on obtient la clause vide.

Exemple. On veut montrer que la formule

$$\forall x \forall y (R(x, f(y)) \vee R(y, f(x)))$$

implique logiquement la formule

$$\exists x \exists y (R(x, f(y)) \wedge R(y, f(x))) .$$

Cela revient à dire que la théorie

$$\{ \forall x \forall y (R(x, f(y)) \vee R(y, f(x))) , \forall z \forall w (\neg R(z, f(w)) \vee \neg R(w, f(z))) \} .$$

est incohérente. On peut donc considérer l'ensemble de clauses :

$$\Rightarrow R(x, f(y)), R(y, f(x)) \quad (1)$$
$$R(z, f(w)), R(w, f(z)) \Rightarrow \quad (2)$$

et dériver dans le calcul R :

$$\Rightarrow R(x, f(y)) \quad (3, \text{Fac } 1.1 ; 1.2)$$
$$R(y, f(x)) \Rightarrow \quad (4, \text{Res } 3.1 ; 2.1)$$
$$\Rightarrow \quad (5, \text{Res } 3.1 ; 4.1)$$

Exemple. Si une théorie est cohérente, on ne peut pas dériver la clause vide. La théorie suivante est cohérente et le processus de saturation ne se termine pas :

$$\Rightarrow P(c) \quad (1)$$
$$P(x) \Rightarrow P(f(x)) \quad (2)$$

et commencer ici à dériver :

$$\Rightarrow P(f(c)) \quad (3, \text{Res } 2.1\,;1.1)$$
$$\Rightarrow P(f(f(c)) \quad (4, \text{Res } 2.1\,;3.1)$$
$$\dots$$

4 La structure d'un démonstrateur automatique

Saturation. Un démonstrateur automatique, utilisant le calcul de la résolution, dérivera toute clause possible à partir d'une théorie donnée : il s'arrêtera seulement si la clause vide a été dérivée, et dans ce cas il annoncera que la théorie est incohérente, où s'il n'est pas possible dériver aucune nouvelle clause (et la clause vide n'a pas été dérivée). Dans ce dernier cas il annoncera que la théorie est cohérente, car le calcul de la résolution est complet.

Simplification. On s'aperçoit que la tache d'engendrer toutes les clauses dérivables à partir d'une théorie donnée engendrera des calculs assez complexes où même irréalisables. On essaye donc d'entrelacer le processus de saturation avec un processus d'élimination des clauses « redondantes ». Un clause est redondante si elle n'est pas nécessaire à fin de dériver la clause vide. Par exemple :

– une tautologie est une clause redondante (cf. la procédure de Davis Putnam) qu'on peut toujours éliminer d'un ensemble de clauses. On pourra donc éliminer les clauses de la forme

$$\Gamma, A \Rightarrow A, \Delta$$

– une sur-clause est redondante (cf. Davis Putnam) : supposons que l'on a $\Gamma_1 \Rightarrow \Delta_1$, $\Gamma_2 \Rightarrow \Delta_2$ tels que, pour une substitution σ, $\sigma\Gamma_1 \subseteq \Gamma_2$ and $\sigma\Delta_1 \subseteq \Delta_2$: on peut alors éliminer $\Gamma_2 \Rightarrow \Delta_2$.

La procédure `saturer` qui suit est un premier pas vers l'implémentation d'un démonstrateur automatique. Elle est analogue aux un algorithme de parcours en largeur de graphes (l'analogie étant avec les graphes infinis).

saturer.code

```
 1 : Procédure saturer(T)
 2 : T : ensemble de clauses
 3 : {
 4 :
 5 :    WO (* worked off : ensemble de clauses déjà elaborées *)
 6 :    Us (* usable  : enseble de clause utilisables *)
 7 :
 8 :    Us := T
 9 :    WO := ensemble vide
10 :
11 :    tant-que( vrai )
12 :    {
13 :       si Us contient la clause vide
14 :         retourner "théorie contradictoire"
15 :
16 :       si Us est vide,
17 :         retourner "ensemble saturé, théorie cohérente"
18 :
19 :       choisir C in Us
20 :       ajouter C à WO
21 :       enlever C de Us
22 :
23 :       New :=  appliquer toutes les règles entre C et WO
24 :
25 :       simplifier l'ensemble New (par rapport à New, Us et WO)
26 :       simplifier WO et US (par rapport à new)
27 :
28 :       Us := union de US et New
29 :    }
30 : }
```

Exercise 4.1 *L'ensemble* `U` *dans le pseudo-code* `saturer` *peut être implémenté par une pile ou par un queue. Quelle est l' implémentation correcte ?*

Exemple. Voici la procédure appliquées à la théorie

$$\Rightarrow P(f(a)) \quad (1)$$
$$P(f(x)) \Rightarrow P(x) \quad (2)$$
$$P(f(a)), P(f(x)) \Rightarrow \quad (3)$$

Au début `WO` est vide, `Us` contient $1, 2, 3$.

1. La clause 1 est sélectionnée et ajoutée à `WO` : on ne peut rien engendrer.
2. La clause 2 est sélectionnée et ajoutée à `WO` : les clause suivantes sont produites :

$$\Rightarrow P(a) \quad (4,, \text{Res } 1.1.\,;2.1)$$
$$P(f(f(x))) \Rightarrow P(x) \quad (5,, \text{Res } 2.1.\,;2.2)$$

Observons que dans la dernière inférence on a prix une copie de la clause 2, avec la variable x remplacée par y. On a donc appliquée la régle

$$\frac{P(f(y)) \Rightarrow P(y) \quad P(f(x)) \Rightarrow P(x)}{P(f(f(x))) \Rightarrow P(x)}\, y \to f(x)$$

3. $\texttt{WO} = \{1, 2\}$ et $\texttt{Us} = \{3, 4, 5\}$: la clause 4 est choisie, il n'est pas possible d'engendrer d'autres clauses.
4. $\texttt{WO} = \{1, 2, 4\}$ et $\texttt{Us} = \{3, 5\}$: la clause 3 est choisie, on engendre :

$$\begin{aligned} P(f(x)) \Rightarrow & \qquad (6, \text{Res } 1.1\,;3.1)\\ P(f(a)) \Rightarrow & \qquad (7, \text{Res } 1.1\,;3.2)\\ P(f(f(a))), P(f(x)) \Rightarrow & \qquad (8, \text{Res } 2.2\,;3.1)\\ P(f(a)), P(f(f(x))) \Rightarrow & \qquad (9, \text{Res } 2.2\,;3.2) \end{aligned}$$

Les clauses $3, 7, 8, 9$ sont des sur-clauses de 6 : elles sont supprimées.

5. $\texttt{WO} = \{1, 2, 4\}$ et $\texttt{Us} = \{5, 6\}$, la clause 6 est choisie et on peut déduire

$$\Rightarrow \qquad (10 \text{ Res } 1.1\,;6.1)$$

et donc l'algorithme s'arrête et annonce qu'il a trouvé la clause vide.

5 Indécidabilité de la logique du premier ordre

Le procédure `saturer` est une procédure de demi-décision : si elle s'arrête, alors on a bien répondu à la question « est ce que la théorie passé en paramètre est cohérente ». Si la procédure ne s'arrête pas, alors la théorie est cohérente. Le problème est évidemment savoir si cette procédure s'arrête après un temps fini.

On peut montrer que si on est capable de décider si un démonstrateur automatique s'arrête sur une théorie passé en entrée, alors on peut décider la logique du premier ordre. Rappelons, que décider un problème veut dire qu'il existe un algorithme qui s'arrête toujours, et réponde à la question si une instance du problème appartient au non aux exemplaire positifs du problème. Pour ce qui concerne logique du premier ordre, on aimerait avoir un algorithme $\mathcal{A}$ qui, sur toute entrée ϕ formule de la logique du premier ordre, s'arrête toujours, et réponde à la question est ce que ϕ est vrai dans toutes les modèles ? Il est un résultat classique de la théorie de la calculabilité qu'un tel algorithme n'existe pas.