919 : Unification : Algorithmes et applications

10/04/2015

L'unification est une méthode de raisonnement syntaxique qui détermine la satisfiabilité d'un système d'équations. Ce procédé est utilisé à de nombreux niveaux des raisonnements informatiques.

I Signature, termes et substitution

1) Termes

<u>Définition 1</u> (Signature) Une signature Σ est un ensemble de <u>symboles de fonctions</u>, associés chacun a une <u>arité</u>.
$\Sigma^{(n)}$ est l'ensemble des symboles d'arité n
$\Sigma^{(0)}$ est l'ensemble des <u>constantes</u>

<u>Exemple</u> : Un groupe peut être représenté par un élément neutre e d'arité 0, une fonction inverse i d'arité 1, et une addition f d'arité 2. $\Sigma_G = \{e, i, f\}$

On considère maintenant une signature Σ fixe, et on se donne un <u>ensemble de variable</u> $V (= \{x, y, x_1, \dots x_n \dots\})$ tel que $V \cap \Sigma = \emptyset$.

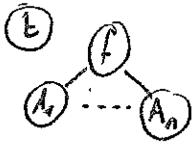
<u>Définition 2</u> (Termes) L'ensemble des termes de signature Σ et de variable V, noté $T(\Sigma, V)$ est le plus petit ensemble vérifiant
- $V \subset T(\Sigma, V)$
- $\forall n \geqslant 0$, $\forall f \in \Sigma^{(n)}$, $\forall (t_1 \dots t_n) \in T(\Sigma, V)^n$, on a $f(t_1, \dots t_n) \in T(\Sigma, V)$

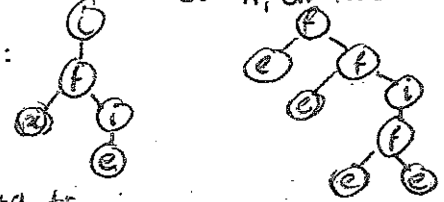
<u>Remarque</u>: C'est un ensemble inductif on pourra appliquer les théorèmes d'induction pour nos preuves.

<u>Définition 3</u> (Terme clos) On dit qu'un terme est clos s'il ne contient aucune occurrence de variables.

<u>Exemples</u> : $t_1 = i(f(x, i(e)))$ est un terme de Σ_G.
$t_2 = f(e, f(e, i(f(e,e))))$ est un terme clos de Σ_G avec x comme variable

2) Arbres

La représentation arborescente découle de la définition inductive de $T(\Sigma, V)$:
- Si $t \in \Sigma^{(0)} \cup V$, l'arbre de t est (t)
- Si $t = f(t_1 \dots t_n)$, l'arbre de t est  où A_i est l'arbre de t_i

<u>exemples</u> :  les arbres associés à t_1 et t_2

3) Substitutions

<u>Définition 4</u> Une substitution est une application $\sigma : T(\Sigma, V) \to T(\Sigma, V)$ telle que
$\forall f \in \Sigma^{(n)}$, $\forall (t_1 \dots t_n) \in T(\Sigma, V)^n$ $\sigma(f(t_1 \dots t_n)) = f(\sigma(t_1) \dots \sigma(t_n))$

<u>Remarque</u>: Une substitution est définie par sa restriction à V. En général, on travaillera sur des substitutions à domaine fini

<u>Exemple</u> : σ définie par $x \mapsto i(e)$
$\sigma(t_1) = i(f(i(e), i(e)))$

<u>Définition 5</u>: Une substitution injective qui remplace toute variable par une variable est appelée <u>renommage</u>

<u>Définition 6</u> (Instance) Un terme s est une instance de t s'il existe une substitution σ telle que $t = \sigma(s)$

<u>Définition 7</u> (substitution plus générale) σ est plus générale que σ' s'il existe une substitution δ telle que $\sigma' = \delta\sigma$. On note $\sigma \leqslant \sigma'$

<u>Proposition 8</u> $\leqslant$ est un préordre sur les substitutions

<u>Proposition 9</u> $t \lesssim s$ et $t \gtrsim s \iff \exists$ un renommage δ tel que $t = \delta s$ $(s = \delta^{-1} t)$
On note $t \sim s$

II Problèmes d'unification

1) Exemple introductif simple

Soit une fonction Caml prenant en entrée un vecteur. Pour vérifier que l'entrée est valide, on doit l'unifier avec le modèle du vecteur, par exemple vecteur(taille, position, type)
où : - vecteur est une fonction d'arité 3 de Σ
- taille, position et type sont des variables qui décrivent la taille, la position dans la mémoire, et le type de donnée du vecteur.

Pour une fonction prenant en entrée un vecteur d'entiers, on devra l'unifier avec vecteur(taille, position, int) où int est une constante de Σ.

2) Système d'équation et unification

Définition 10 : Un problème d'unification est un ensemble de couples $S = \{(s_1, t_1), \dots (s_n, t_n)\}$. Une solution de ce problème est une substitution σ telle que $\forall i \in [\![1,n]\!], \sigma(s_i) = \sigma(t_i)$.
σ est appelé unificateur. On note $U(S)$ l'ensemble des unificateurs de S.
S est unifiable si $U(S) \neq \emptyset$

Définition 11 (Unificateur le plus général) Une substitution σ est un unificateur le plus général d'un problème S ou unificateur principal ssi :
- $\sigma \in U(S)$
- $\forall \sigma' \in U(S), \sigma \lesssim \sigma'$

Exemple $S = \{(x,y)\}$ $\quad \sigma_1 : x \mapsto y$
$\sigma_2 : x \mapsto z, \ y \mapsto z$

σ_2 n'est pas unificateur principal car σ_1 est un unificateur de S mais on n'a pas $\sigma_2 \lesssim \sigma_1$

Théorème 12 : Si un problème S est unifiable, alors il existe un unificateur principal de S idempotent (ie $\sigma^2 = \sigma$)

3) Algorithme d'unification

Définition 13 (forme résolue) S est sous forme résolue si $S = \{(x_1, t_1), \dots (x_n, t_n)\}$ où $\forall x_i \in V$, $x_i \neq x_j$ pour $i \neq j$, et $\forall i, \ x_i \notin Var(t_i)$

Proposition 14 Si S est sous forme résolue, alors $\tilde{S} = \{x_i \mapsto$ … est un unificateur principal

Algorithme : entrée : S
sortie : Un système résolu équivalent à S ou détecter s'il n'y a pas de solution (S n'est pas unifiable) P

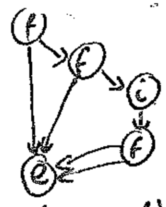
Proposition 15 L'algorithme est correct et termine

Remarque : La complexité est au moins exponentielle en temps et espace

Amélioration de complexité : DAG (graphe orienté acyclique)

On représente un système par un graphe orienté acyclique, en respectant le fait qu'il n'y ait pas deux sous-arbres identiques. En particulier, chaque variable n'apparaît qu'une seule fois

exemple : l'arbre associé à t_2 est :



Le procédé d'unification reste le même, on met des pointeurs entre deux noeuds unifiés. L'unificateur se lit en regardant les pointeurs qui partent des variables. (cf Annexe 1)
En utilisant un procédé de marquage des noeuds, on obtient une complexité quadratique

III Systèmes de réécriture

Définition 16 : Une règle de réécriture est une paire (l, r) où l n'est pas une variable et $Var(r) \subset Var(l)$
On note $l \to r$.
Un système de réécriture est un couple $S = (\Sigma, R)$, Σ est une signature et R un ensemble de règles de réécriture (fini)

On suppose donné dans la suite $S = (\Sigma, R)$.

Définition 17 (Position, greffe, sous terme)
Soit s un terme de Σ. L'ensemble $Pos(s)$ des positions de s est l'ensemble des noeuds de l'arbre représentant s.
On note $s_{|p}$ le sous arbre ayant pour racine la position p
On note $s[r]_p$ le terme dont le sous arbre en position p est remplacé par r.

r. ture)

→ la relation sur les termes définie par

$(\exists \sigma$ substitution, $\exists p \in Pos(s), \exists l \to r \in R$ tele que

$s_{|p} = \sigma(l)$ et $t = s[\sigma(r)]_p)$ Voir Annexe 2

$\xrightarrow{*}$ sa clôture réflexive transitive

$\xrightarrow{\alpha}$ sa clôture réflexive

fication sert à savoir quelle règle de récriture

licable sur un terme

note $s \downarrow t$ si $\exists r$ telque $s \xrightarrow{*} r$, $t \xrightarrow{*} r$.

urch Rosser ssi $s \xleftrightarrow{*} t \Rightarrow s \downarrow t$

nfluent ssi $s \xleftarrow{*} u \xrightarrow{*} t \Rightarrow s \downarrow t$

alement confluent ssi $s \leftarrow u \rightarrow t \Rightarrow s \downarrow t$

inant ssi il n'existe pas de chaîne infinie $s_1 \to s_2 \to \dots$

vergent ssi → est confluent et terminant.

man) Une relation terminante est confluente

: elle est localement confluente

Soit $l_1 \to r_1$, $l_2 \to r_2$ deux règles de récritures.

nommage) que $Var(l_1, r_1) \cap Var(l_2, r_2) = \emptyset$.

(l_1) tel que $l_{1|p} \notin V$ et soit θ un unificateur

de $\{(l_{1|p}, l_2)\}$.

$\langle \theta r_1, (\theta l_1)[\theta r_2]_p \rangle$ est une paire critique :

θr_1 et $\theta l_1 = \theta l_1[\theta(l_2)]_p \longrightarrow \theta(l_1)[\theta(r_2)]_p$

IV Méthode de résolution et programmation logique

1) Méthode de résolution

$\Sigma \models F \Leftrightarrow \Sigma \cup \{\neg F\}$ n'a pas de modèle : propriété fondamentale des méthodes de résolution.

Définition 22 : $C_1 = (\Gamma_1, \Delta_1)$, $C_2 = (\Gamma_2, \Delta_2)$ deux clauses sans variables communes. Si on trouve $P = \{P_1 .. P_n\} \subset \Delta_1$, $N = \{N_1, .. N_k\} \subset \Delta_2$ tels que $S = \{(P_1, N_1), (P_1, P_2), \dots (P_1, P_n), (N_1, N_2), \dots (N_1, N_k)\}$ soit unifiable et σ un unificateur principal de S, alors la clause

$$C = \Big(\sigma(\Gamma_1) \cup \sigma(\Gamma_2) \setminus \sigma(N), \big(\sigma(\Delta_1) \setminus \sigma(P)\big) \cup \sigma(\Delta_2)\Big)$$

est appelé <u>résolvant</u> de C_1 et C_2 noté $\dfrac{C_1 \quad C_2}{C}$

Définition 23 Soit S un ensemble de clauses, C un clause

une preuve de C par résolution à partir de S est une suite finie $C_1, \dots C_n$ telle que $C_n = C$ et $\forall i \in [\![1, n-1]\!]$,

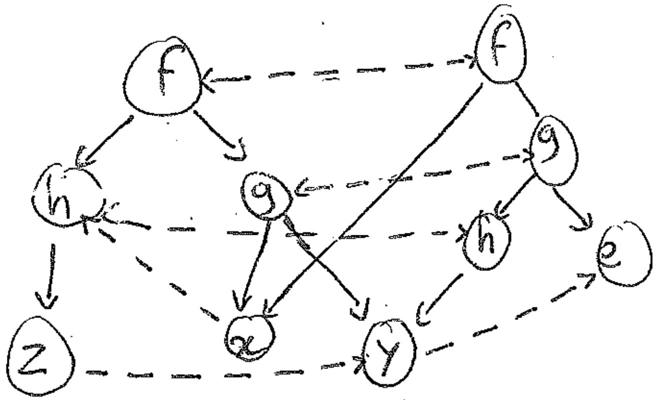
- Soit $C_i \in S$
- Soit $\exists j, k \in [\![1, i-1]\!] \,/\, \dfrac{C_j \quad C_k}{C_i}$

On note $S \vdash_{Res} C$

Proposition 24 Si $S \vdash_{Res} C$ alors $S \models \forall C$

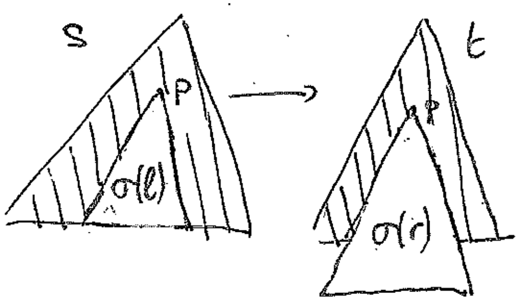
où $\forall C$ est C précédé de quantificateurs universels pour toutes les variables

Annexe 1 : Unification de $\{(f(h(z), g(x,y)), f(x, g(h(y),e)))\}$



$\sigma : \{x \mapsto h(e),\ z \mapsto e,\ y \mapsto e\}$

Annexe 2 : réécriture



améliorer la complexité de l'algo

↳ il faut savoir la complexité ! NP-complet.

Recherche peut-être plus d'algos.

Paires critiques

[Baader, Term Rewriting and All That]

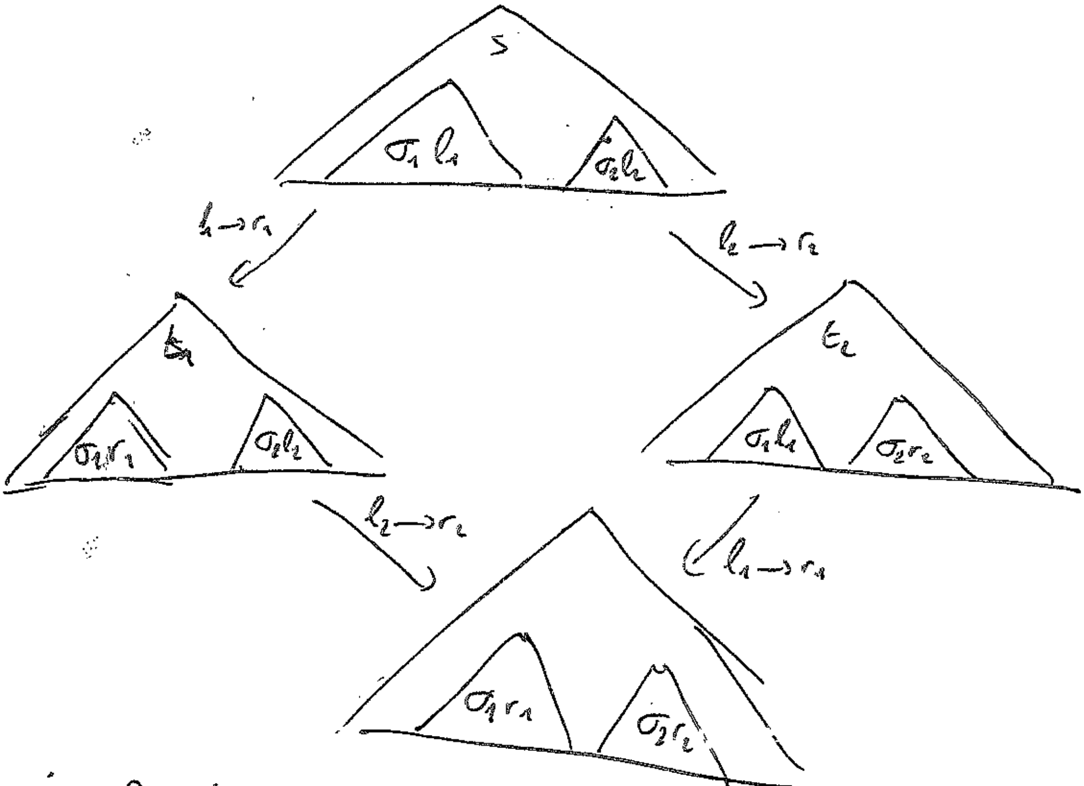
Théorème : Un système de réécriture est localement confluent si et seulement si toutes ses paires critiques sont joignables

Cadre : La réécriture est non deterministe.

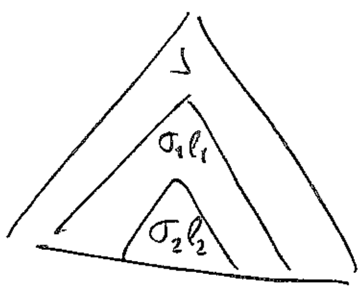
Soit s un terme pouvant se réécrire en t_1 et t_2

$s_{|p_1} = \sigma_1(l_1)$, $s_{|p_2} = \sigma_2(l_2)$, $l_1 \to r_1$, $l_2 \to r_2$

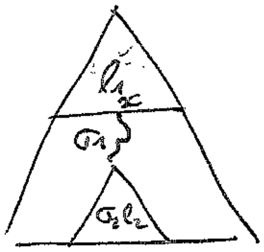
Cas 1 : $\sigma_1(l_1)$ et $\sigma_2(l_2)$ sont dans des sous arbres distincts :



Cas 2 : p_2 est un noeud du sous arbre de racine p_1 $(p_2 = p_1 p)$



On étudie uniquement $s_{|p_1}$ puisque $\to$ est compatible avec le contexte $s[\ ]_{p_1}$.

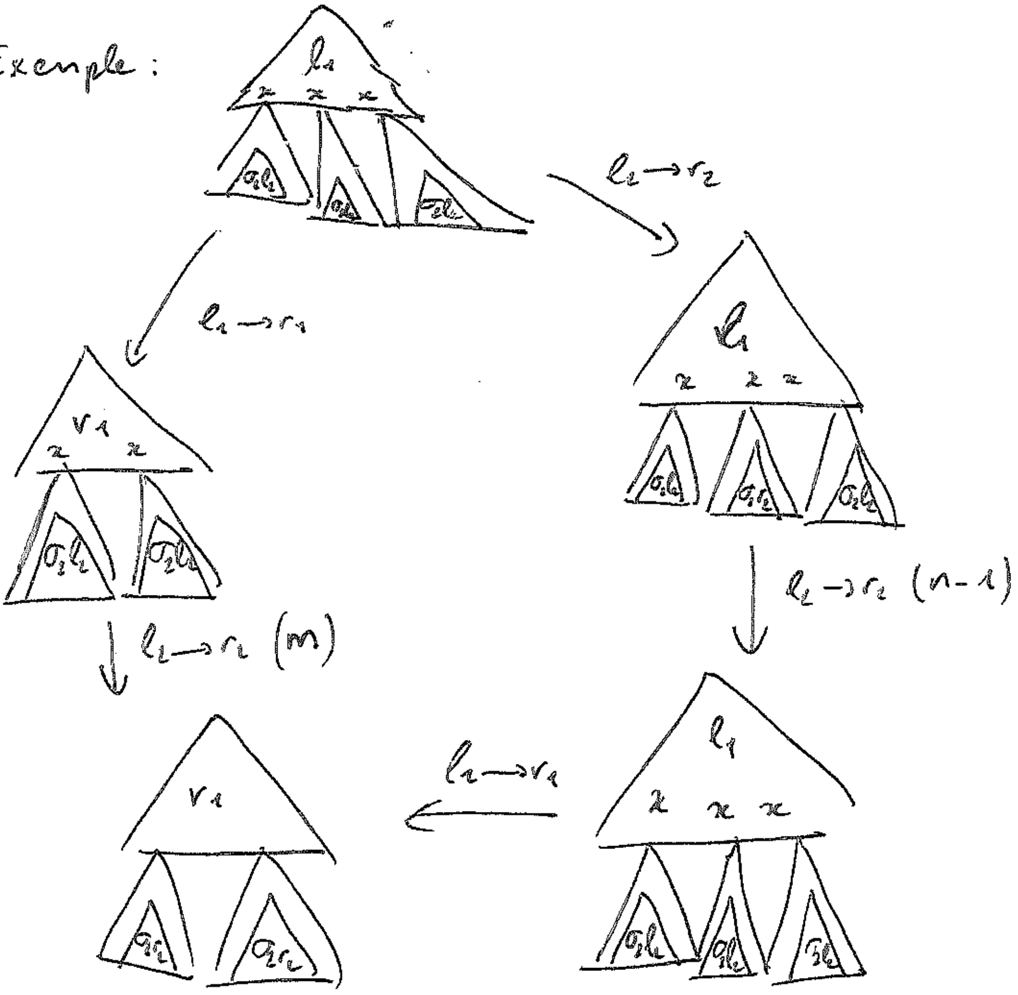
-1 : 

Cas non critique

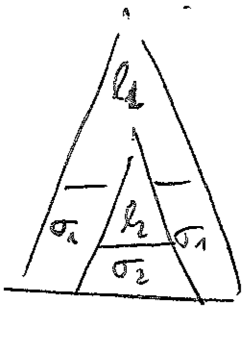
$\sigma_2 l_2$ est un sous terme de $\sigma_1(x)$, $x \in Var(l_1)$

Soit n le nombre de fois où x apparaît dans l_1.
m ______________________ r_1

Exemple :



Cas 2.3 : Paire critique



<u>Définition</u>: Soit $l_1 \to r_1$, $l_2 \to r_2$ deux règles telles que $Var(l_1, r_1) \cap Var(l_2, r_2) = \emptyset$. Soit $p \in Pos(l_1)$ tel que $l_{1|p} \notin V$ et soit θ un unificateur principal de $\{(l_{1|p}, l_2)\}$

Alors $\langle \theta r_1, (\theta l_1)[\theta r_2]_p \rangle$ est une paire critique

<u>Preuve du Théorème</u>

$\Rightarrow$: localement confluent donc $\theta r_1 \leftarrow \theta l_1 \rightarrow (\theta l_1)[\theta r_2]_p$, $\theta r_1 \xrightarrow{*} z \xleftarrow{*} (\theta l_1)[\theta r_2]_p$

$\Leftarrow$ oui, regardez au dessus !

Algorithme d'unification

[Baader, Term Rewriting Systems and All That]

Soit $S = \{(s_1, t_1), \dots (s_n, t_n)\}$ un problème d'unification

<u>Lemme 1</u> Si S contient un couple de termes de la forme $(f(s_1 \dots s_n), g(t_1 \dots t_k))$ alors S n'est pas unifiable

<u>Preuve:</u> $\forall \sigma$ substitution, $\sigma(f(s_1, \dots s_n)) = f(\sigma(s_1), \dots \sigma(s_n)) \neq g(\sigma(t_1), \dots \sigma(t_k)) = \sigma(g(t_1 \dots$

<u>Lemme 2</u> Si S contient un couple de termes de la forme (x, t), où $x \in Var(t)$ et $x \neq t$, alors S n'est pas unifiable

<u>Preuve</u> : t est de la forme $f(t_1 \dots t_n)$, $\exists i \in [\![1,n]\!]$, $x \in Var(t_i)$

Alors $|\sigma(x)| \leq |\sigma(t_i)| < |\sigma(t)|$.

On définit alors deux tests algorithmiques qui déterminent qu'un système n'est pas unifiable

A - Clash $S = \{(f(s_1 \dots s_n), g(t_1 \dots t_k))\} \cup S'$, $f \neq g \dots$

B - Occurs-Check $S = \{(x, t)\} \cup S'$, $x \neq t$, $x \in Var\, t$

$\Rightarrow$ Retourner "S n'est pas unifiable"

Il faut maintenant définir la procédure permettant de transformer un système S pour arriver à un système sous forme résolu, ou montrer que S n'est pas unifiable. On définit pour cela quatre opérations

1- Supprime : $S = \{(t, t)\} \cup S' \Rightarrow S \leftarrow S'$

2- <u>Décompose</u> : $S = \{(f(t_1, \dots t_n), f(s_1, \dots s_n))\} \cup S' \Rightarrow S \leftarrow S' \cup \{(t_1, s_1), (t_2, s_2) \dots (t_n, s_n)\}$

3- Oriente : $S = \{(t, x)\} \cup S'$, $t \notin V$, $x \in V \Rightarrow S \leftarrow S' \cup \{(x, t)\}$

4- Elimine : $S = \{(x, t)\} \cup S'$, $x \in Var(S) \setminus Var(t) \Rightarrow S \leftarrow \{(x, t)\} \cup (x \mapsto t)\, S'$

Algorithme général : entrée: S

sortie: Un système résolu équivalent à S ou "S n'est pas unifiable"

Unifier (S)

Tant qu'on peut appliquer une des règles
- → appliquer la règle
- → tester Clash et Occurs Check

Retourner S

Correction de l'algorithme :

Si l'algorithme termine et retourne S'. Alors S' ne contient pas de couples
- $(f(\ldots), f(\ldots))$ d'après la règle 2
- $(f(\ldots), g(\ldots))$ d'après le test A
- (x, x) d'après la règle 1
- (t, x), $t \notin V$ d'après la règle 3
- (x, t), $x \in Var(t)$ d'après le test B

⟹ les règles sont toutes sous forme résolue.

De plus, d'après la règle 4, $\forall x \in V$, x apparaît au plus une fois dans S

⟹ S est sous forme résolue

Terminaison de l'algorithme

On définit les variants de boucle suivants :

n_1 : nombre de variables non résolues de S

n_2 : taille de S

n_3 : nombre de couples $(t, x) \in S$

On vérifie alors que chacune des opération fait décroître le triplet (n_1, n_2, n_3) avec l'ordre lexicographique

	n_1	n_2	n_3
Supprime :	$\geq$	$>$	
Décompose	$\geq$	$>$	
Oriente	$\geq$	$=$	$>$
Élimine	$>$		

Unifier (S)

Tant qu'on peut appliquer une des règles
- → appliquer la règle
- → tester Clash et Occurs Check

Retourner S

Correction de l'algorithme :

Si l'algorithme termine et retourne S', Alors S' ne contient pas de couples
- $(f(\dots), f(\dots))$ d'après la règle 2
- $(f(\dots), g(\dots))$ d'après le test A
- (x, x) d'après la règle 1
- (t, x), $t \notin V$ d'après la règle 3
- (x, t), $x \in Var(t)$ d'après le test B

$\Rightarrow$ les règles sont toutes sous forme résolue.

De plus, d'après la règle 4, $\forall x \in V$, x apparaît au plus une fois dans S

$\Rightarrow$ S est sous forme résolue

Terminaison de l'algorithme

On définit les variants de boucle suivants :

n_1 : nombre de variables non résolues de S

n_2 : taille de S

n_3 : nombre de couples $(t, x) \in S$

On vérifie alors que chacune des opération fait décroître le triplet (n_1, n_2, n_3) avec l'ordre lexicographique

	n_1	n_2	n_3
Supprime :	$\geqslant$	$>$	
Décompose	$\geqslant$	$>$	
Oriente	$\geqslant$	$=$	$>$
Elimine	$>$		

Définition 17 (réécriture)

- On note $\rightarrow$ la relation sur les termes définie par
$(s \rightarrow t) \Leftrightarrow (\exists \sigma$ substitution, $\exists p \in Pos(s)$, $\exists l \rightarrow r \in R$ tels que $s_{|p} = \sigma(l)$ et $t = s[\sigma(r)]_p)$ Voir Annexe 2
- On note $\xrightarrow{*}$ sa clôture réflexive transitive
- On note $\xleftrightarrow{*}$ sa clôture réflexive

Remarque: L'unification sert à savoir quelle règle de réécriture est applicable sur un terme

Définition 18: On note $s \downarrow t$ si $\exists r$ tel que $s \xrightarrow{*} r$, $t \xrightarrow{*} r$.

- $\rightarrow$ est Church Rosser ssi $s \xleftrightarrow{*} t \Rightarrow s \downarrow t$
- $\rightarrow$ est confluent ssi $s \xleftarrow{*} u \xrightarrow{*} t \Rightarrow s \downarrow t$
- $\rightarrow$ est localement confluent ssi $s \leftarrow u \rightarrow t \Rightarrow s \downarrow t$
- $\rightarrow$ est terminant ssi il n'existe pas de chaîne infinie $s_1 \rightarrow s_2 \rightarrow \dots$
- $\rightarrow$ est convergent ssi $\rightarrow$ est confluent et terminant.

Lemme 19 (Newman) Une relation terminante est confluente si elle est localement confluente

Paires Critiques: Soit $l_1 \rightarrow r_1$, $l_2 \rightarrow r_2$ deux règles de réécritures.
On suppose (renommage) que $Var(l_1, r_1) \cap Var(l_2, r_2) = \emptyset$.
Soit $p \in Pos(l_1)$ tel que $l_{1|p} \notin V$ et soit θ un unificateur principal de $\{(l_{1|p}, l_2)\}$.
Alors $\langle \theta r_1, (\theta l_1)[\theta r_2]_p \rangle$ est une paire critique:
$\theta l_1 \rightarrow \theta r_1$ et $\theta l_1 = \theta l_1[\theta(l_2)]_p \rightarrow \theta(l_1)[\theta(r_2)]_p$

Théorème 20: Un système de réécriture terminant est convergent si et seulement si ses paires critiques sont joignables

Corollaire 21 la convergence d'un système de réécriture terminant est un problème décidable

IV Méthode de résolution et programmation logique

1) Méthode de résolution

$\Sigma \models F \Leftrightarrow \Sigma \cup \{\neg F\}$ n'a pas de modèle : propriété fondamentale des méthodes de résolution.

Définition 22: $C_1 = (\Gamma_1, \Delta_1)$, $C_2 = (\Gamma_2, \Delta_2)$ deux clauses sans variables communes. Si on trouve $P = \{P_1 .. P_n\} \subset \Delta_1$, $N = \{N_1, .. N_n\} \subset \Delta_2$ tels que $S = \{(P_1, N_1), (P_1, P_2), \dots (P_1, P_n), (N_1, N_2), \dots (N_1, N_n)\}$ soit unifiable et σ un unificateur principal de S, alors la clause
$$C = \left(\sigma(\Gamma_1) \cup \sigma(\Gamma_2) \setminus \sigma(N), (\sigma(\Delta_1) \setminus \sigma(P)) \cup \sigma(\Delta_2)\right)$$
est appelé <u>résolvant</u> de C_1 et C_2 noté $\dfrac{C_1 \quad C_2}{C}$

Définition 23 Soit S un ensemble de clauses, C une clause. Une preuve de C par résolution à partir de S est une suite finie $C_1, \dots C_n$ telle que $C_n = C$ et $\forall i \in [\![1, n]\!]$,

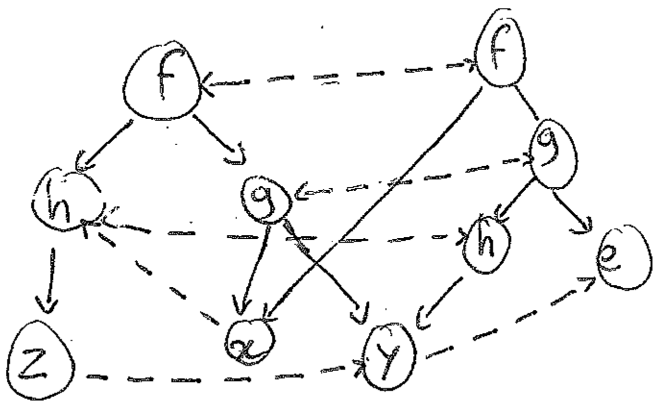
- Soit $C_i \in S$
- Soit $\exists j, k \in [\![1, i-1]\!] \,/\, \dfrac{C_j \quad C_k}{C_i}$

On note $S \vdash_{Res} C$

Proposition 24 Si $S \vdash_{Res} C$ alors $S \models \forall C$ où $\forall C$ est C précédé de quantificateurs universels pour toutes les variables

Ref: Baader

Annexe 1 : Unification de $\{(f(h(z), g(x, y)),\ \text{''}(x, g(h(y), e)))\}$



$\sigma : \{x \mapsto h(e),\ z \mapsto e,\ y \mapsto e\}$

améliorer la complexité de l'algo

↳ il faut savoir la complexité ! NP-complet.

Recherche peut-être plus d'algos.

Annexe 2 : réécriture