L'unification sert à résoudre des équations syntaxiques sans aucune considération sémantique. Elle joue un rôle central pour effectuer des preuves en propageant les contraintes ainsi que pour calculer informatiquement, notamment sous le langage Caml.

I Termes du 1er ordre, unification, filtrage.

1) Termes

Définition 1 : • Une <u>signature</u> Σ est un ensemble de symboles de fonctions avec une certaine arité. L'<u>arité</u> d'une fonction f est l'entier naturel n d'arguments de la fonction. On note $\Sigma^{(n)}$ l'ensemble des fonctions d'arité n.

• Une fonction d'arité nulle est aussi appelée <u>constante</u>.

Exemples 2 : • Une signature pour la théorie des groupes : $\Sigma_1 = \{e ; (\cdot)^{-1} ; *(\cdot,\cdot)\}$

• Une signature pour l'analyse réelle :

$$\Sigma_2 = \{0, 1, \dots, \pi, e^2, \sin(\cdot), \ln(\cdot), \dots, +(\cdot,\cdot), \times(\cdot,\cdot), \dots\}$$

À partir de maintenant, on suppose fixés une signature Σ et un ensemble V dénombrable de variables. On notera souvent $x, y, z, x_1, \dots, x_n \dots$ les variables et on suppose $V \cap \Sigma = \phi$.

Définition 3 : <u>L'ensemble $T(\Sigma,V)$ des termes</u> sur Σ et V est défini inductivement par :

• $V \cup \Sigma^{(0)} \subset T(\Sigma,V)$ (T contient les constantes et les variables)

• Si $f \in \Sigma^{(n)}$ et $(t_1, \dots, t_n) \in T(\Sigma,V)^n$, alors $f(t_1, \dots, t_n) \in T(\Sigma,V)$

($T(\Sigma,V)$ est stable par application des fonctions de Σ.)

Remarque 4 : On peut définir alternativement $T(\Sigma,V)$ comme le plus petit sous-ensemble contenant les constantes et les variables et stable par application des fonctions de Σ.

Définition 5 : Un terme $t \in T(\Sigma,V)$ est <u>clos</u> s'il ne contient aucune variable.

Exemples 6 : • $t_1 = e * x^{-1}$ est un terme sur Σ_1

• $t_2 = \ln(\pi \times \sqrt{2}) + |0|$ est un terme sur Σ_2.

2) <u>Représentation arborescente des termes</u>

En utilisant la définition inductive de $T(\Sigma,V)$, on définit l'arbre de t par :

• si $t \in \Sigma^{(0)} \cup V$, alors l'arbre de t est : t

• si $t = f(t_1, \dots, t_n)$, alors l'arbre de t est :

où A_i désigne l'arbre de t_i.

• La profondeur de t est la profondeur de l'arbre associé à t.

On définit également pour $(t,s) \in T(\Sigma,V)$:

• $Pos(t)$ est <u>l'ensemble des positions</u> de t : – si $t \in \Sigma^{(0)} \cup V$, $Pos(t) = \{\varepsilon = \text{mot vide}\}$

– si $t = f(t_1, \dots, t_n)$, $Pos(t) = \{\varepsilon\} \cup \bigcup_{i=1}^{n} \{i.p \mid p \in Pos(t_i)\}$

• $t_{|p}$ désigne le terme associé à l'arbre dont la racine est à la position p.

• $t[s]_p$ est la greffe <u>du terme s</u> à la position p, ie on remplace $t_{|p}$ par s à la position p.

Exemples 7 : cf annexe

3) <u>Substitutions</u>

Définition 8 : Une <u>substitution</u> est une application $\sigma : T(\Sigma,V) \to T(\Sigma,V)$ telle que :

$$\forall f \in \Sigma^{(n)}, \forall (t_1, \dots, t_n) \in T(\Sigma,V)^n, \sigma(f(t_1, \dots, t_n)) = f(\sigma(t_1), \dots, \sigma(t_n)).$$

Remarques 9 : • Une substitution est entièrement déterminée par sa restriction aux variables.

• Dans la pratique, on travaille avec des substitutions à domaine fini, où $dom(\sigma) = \{x \in V \mid \sigma(x) \neq x\}$

Exemple 10 : Si $\sigma(x) = x * e^{-1}$, alors $\sigma(t_1) = e * (x * e^{-1})^{-1}$

Définition 11 : Une substitution qui remplace toute variable par une variable et qui est injective est appelée <u>renommage</u>.

Définition 12 : • Un terme s est <u>une instance de t</u> (ou <u>t filtre s</u>) si : $\exists \sigma$ substitution, $t = \sigma(s)$

On note $s \approx t$ si σ est un renommage et $s < t$ sinon. En général, on note $s \lesssim t$.

• σ est plus <u>générale que</u> σ' si : $\exists \delta$ substitution, $\sigma = \delta \circ \sigma'$. On note $\sigma' \lesssim \sigma$

Proposition 13 : $\lesssim$ est un pré-ordre sur les substitutions.

Proposition 14 : $(t \approx s)$ si et seulement si $(t \lesssim s$ et $s \lesssim t)$.

4) Systèmes d'équations et unification

Problème : Soit $S=\{s_1=t_1,\dots,s_n=t_n\}$ un système d'équations. Une solution de ce problème est une substitution σ telle que : $\forall i\in[\![1,n]\!],\ \sigma(s_i)=\sigma(t_i)$.

Définition 15 : • Une solution au problème est appelée unificateur. On note $U(S)$ l'ensemble des unificateurs.

• Un système est dit unifiable si $U(S)\neq\emptyset$.

Proposition-définition 16 : Un système S unifiable admet un, unique à renommage près, unificateur minimal pour $\leq$ appelé unificateur principal (noté mgu).

Exemple 17 : $\{g(x,f(x))=y;\ f(y)=f(z)\}$ admet $\sigma:\begin{cases}y\mapsto g(x,f(x))\\ z\mapsto g(x,f(x))\end{cases}$ pour unificateur principal.

Lemme 18 : Soit σ une substitution, (σ est idempotente) si et seulement si ($\mathrm{Im}(\sigma)\cap \mathrm{dom}(\sigma)=\emptyset$)

Proposition 19 : Si S est unifiable, il possède un mgu idempotent.

5) Problème de filtrage

Problème : Pour $(t,s)\in T(\Sigma,V)$, trouver une substitution σ telle que $\sigma(s)=t$; s filtre t ?

Exemple 20 : En Caml, l'appel "match x with $\begin{array}{l}a\to\dots\\ b\to\dots\end{array}$" correspond à deux problèmes de filtrages : d'abord le problème x filtre a ?, puis x filtre b ?.

On peut considérer le problème de filtrage comme un cas particulier de l'unification. Pour cela, on remplace les variables de t par des nouvelles constantes, t sera clos et $\sigma(t)=t$.

Exemple 21 : Le problème $f(x,y)$ filtre $f(g(c),x)$? devient le problème d'unification $\{f(x,y)=f(g(c),c_x)\}$. La solution $\begin{cases}x\mapsto g(c)\\ y\mapsto c_x\end{cases}$ devient $\sigma=\begin{cases}x\mapsto g(c)\\ y\mapsto x\end{cases}$

II Algorithmes d'unification

1) Transformation des systèmes. [DEV]

Principe : On cherche à transformer le système en un système équivalent sous forme résolue ou détecter s'il n'y a pas de solution.

Définition 22 : On dit que S est sous forme résolue $\{x_1=t_1,\dots,x_n=t_n\}$ si (x_i) sont des variables qui n'apparaissent dans aucun (t_i).

Proposition 23 : Sous forme résolue, le système S admet $\vec{S}=(x_i\mapsto t_i)$ comme mgu idempotent.

- décompose : $\{f(\vec{t_n})=f(\vec{u_n})\}\cup S \rightsquigarrow \{t_1=u_1,\dots,t_n=u_n\}\cup S$
- supprime : $\{t=t\}\cup S\rightsquigarrow S$
- oriente : $\{t=x\}\cup S\rightsquigarrow\{x=t\}\cup S$
- élimine : $\{x=t\}\cup S\rightsquigarrow\{x=t\}\cup\{x\mapsto t\}(S)$ si $x\in Var(S)\setminus Var(t)$
- conflit : $\{f(\vec{t_n})=g(\vec{u_n})\}\cup S\rightsquigarrow$ faux
- redondance : $\{x=t\}\cup S\rightsquigarrow$ faux si $x\in Var(t)$ et $x\neq t$

Algorithme : unifier(S) = tant que l'on peut appliquer une étape $S\rightsquigarrow T$ faire $S\leftarrow T$
si S=faux retourner "pas de solution"
retourner $\vec{S}$

Exemple 24 : $S=\{x=f(a),\ g(x,x)=g(x,y)\}$
$\rightsquigarrow\{x=f(a),\ g(f(a),f(a))=g(f(a),y)\}$ élimine $x=f(a)$.
$\rightsquigarrow\{x=f(a),\ f(a)=f(a),\ f(a)=y\}$ decompose $g(f(a),f(a))=g(f(a),y)$
$\rightsquigarrow\{x=f(a),\ f(a)=y\}$ supprime $f(a)=f(a)$
$\rightsquigarrow\{x=f(a),\ y=f(a)\}$ oriente $f(a)=y$

Remarque 25 : Cet algorithme est non déterministe.

Complexité : Au moins exponentielle en temps et en espace : pour $S=\{x_n=f(x_{n-1},x_{n-1})\}$

Proposition 26 : • $\rightsquigarrow$ est bien fondée
• si unifier(S) renvoie faux, le système n'est pas unifiable
• si $S\overset{*}{\rightsquigarrow}S'$ et S' sous forme normale alors S' équivalent à S et $Var(S')=Var(S)$ et $\vec{S'}$ mgu de S.

2) Représentation par graphes orientés acycliques

Principe : On ne créer pas de nouveaux termes mais on utilise des pointeurs.

Exemple 27 : $s_3(x)=f(x_1,\ f(x_2,\ f(x_3$ $)))$
$t_3(x)=f(f(x_0,x_0),\ f(x_1,x_1),\ f(f(x_2,x_2)))$
cf annexe

Complexité : linéaire en espace, mais encore exponentielle en temps.

3) Algorithme quadratique

Principe : • Partager aussi les noeuds. Après "unification" de deux noeuds, on créée un pointeur de l'un à l'autre.
• On marques les sommets lors du parcours, le test devient linéaire
• Avant d'unifier t_i et t_j on créer un lien entre les deux.

Complexité : $O(m^2)$ où m est le nombre d'arêtes du graphe.

Exemple 28 : cf annexe pour $f(x,g(x,x))=f(h(y),g(z),h(b)))$

III Application aux systèmes de réécriture.

1) TRS

Définition 29 : Une règle de réécriture est une paire $(l, r) \in T(\Sigma, V)^2$ telle que :

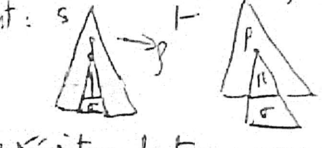
- l n'est pas une variable
- $Var(r) \subseteq Var(l)$

l est appelé redex, r est appelé contraction

Définition 30 :

- Pour une règle de réduction $\rho : l \to r$, on définit la relation $\to_\rho$ par : $(s \to_\rho t)$ ssi ($\exists \sigma$ substitution, $p \in Pos(s)$, $s_{|p} = \sigma(l)$ et $t = s[\sigma(r)]_p$

Schématiquement :



- Un système de réécriture de termes TRS est un couple $R = (\Sigma, R)$ où Σ est une signature et R un ensemble de règles de réduction. On note $\to$ la relation : $(s \to t)$ ssi $(s \to_\rho t$ pour un certain $\rho \in R)$. On note aussi
- $\xrightarrow{*}$ la clôture réflexive transitive de $\to$.
- $\xleftrightarrow{*}$ la clôture réflexive transitive symétrique de $\to$

2) Confluence et terminaison

Définition 31 :

- $\to$ est dite terminante si elle n'admet aucune suite infinie
- $\to$ est dite confluente si $y_1 \xleftarrow{*} x \xrightarrow{*} y_2$ implique $\exists z,\ y_1 \xrightarrow{*} z \xleftarrow{*} y_2$
- $\to$ est dite localement confluente si $y_1 \leftarrow x \to y_2$ implique $\exists z,\ y_1 \xrightarrow{*} z \xleftarrow{*} y_2$

Lemme 32 (Newmann) : Une relation terminante est confluente ssi elle est localement confluente.

Application 33 : Les relations convergentes (confluentes + terminantes) sont telles que
$(x \xleftrightarrow{*} y) \iff (f_n(x) = f_n(y))$ où $f_n(x)$ désigne la forme normale de x.

3) Paires critiques

Proposition 34 : Si R est un TRS fini et convergent alors $\xleftrightarrow{*}_R$ est décidable

Définition 35 : Si $l_i \to r_i$ sont deux règles d'un TRS telles que $Var(l_1) \cap Var(l_2) = \emptyset$, et si $p \in pos(l_1)$ tq $l_{1|p} \notin V$ et $\theta = mgu(l_{1|p}, l_2)$, alors si p n'est pas la racine alors on dit que $< \theta(r_1), \theta(l_1)[\theta(r_2)]_p >$ est une paire critique

Lemme 36 (Paires critiques) : Si $s \to_R t_i$, $i = 1, 2$ alors $t_i = s[u_i]_p$ où $<u_1, u_2>$ ou $<u_2, u_1>$ est une paire critique.
[DEV]

Théorème 37 : Un système TRS terminant est confluant si ses paires critiques sont joignables.

IV Application à la méthode de résolution et à la programmation logique

1) Méthode de résolution

La méthode de résolution repose sur la propriété : $(\Sigma \models F)$ ssi ($\Sigma \cup \{\neg F\}$ n'a pas de modèle)

Définition 38 : Soient $C_1 = (\Gamma_1, \Delta_1)$ et $C_2 = (\Gamma_2, \Delta_2)$ deux clauses sans variables communes. Si on trouve $P_1 \subseteq \Delta_1$, $N_2 \subseteq \Gamma_2$ tels que $P_1 \cup N_2$ soit unifiable et σ un mgu, alors la clause $C = (\Gamma_1 \cup \Gamma_2 \setminus \Delta_2)\sigma$ et $\Delta = (\Delta_1)$ est appelée résolvant de C_1 et C_2 et est notée $\frac{C_1 \quad C_2}{C}$.

Définition 39 : Soient S ensemble de clauses et C une clause. Une preuve par résolution de C à partir de S est une suite finie $C_1, \dots, C_n$ tq $C_n = C$ et $\forall i \in [\![1, n]\!]$

- soit $C_i \in S$
- soit $\exists j, k \in [\![1, j]\!]$, $\frac{C_j \quad C_k}{C_i}$

On note $S \vdash_{Res} C$.

Proposition 40 : Si $S \vdash_{Res} C$ alors $S \models \forall C$

2) Programmation logique

Définition 41 :

- Une clause est dite définie si elle possède un unique littéral positif.
- Un programme logique est un ensemble fini de clauses définies.
- Un but est une formule négative.

Proposition 42 :

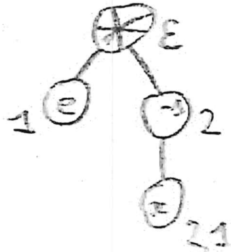
- Tout programme logique possède un modèle.
- Si G est un but, $P \cup \{G\}$ ne possède pas de modèle ssi $P \cup \{G\} \vdash_{Res} \square$

Si on note σ la composée des unificateurs de la preuve, on
$(P \cup \{G\} \vdash_{Res} \square \Rightarrow P \models \forall \sigma(G))$

Application 43 : Cela donne un moyen de trouver un contre-exemple

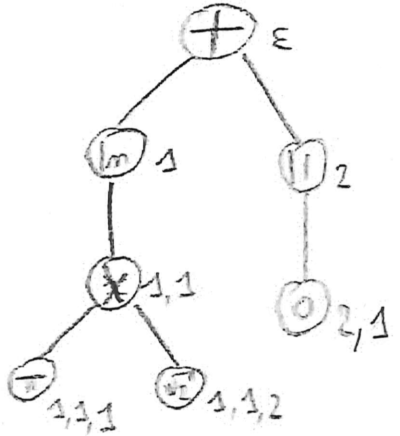
Réf : Bader / Term Rewriting System

• Ex 8 : Représentation de $t_1 = e * x^{-1}$:

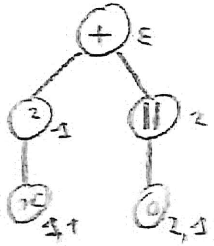


$t_{1|2} =$ 

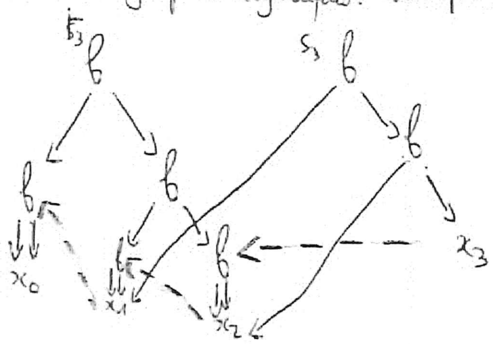
• Représentation de $t_2 = \ln(\pi \times \sqrt{2}) + |0|$:



$t_2[x^2]_1 = x^2 + |0|$ est représenté par :



• Représentation graphes acycliques : exemple 27



• Exemple 28 : $\ell f(x, g(x,x)) = f(h(y), g(3, h(a)))$