Ref: David, Nour, Raffalli
→ Prop 1.4.1

9/10

dirt 1

1/1

Dev 1 : Les involutions sont des bijections

Un peu court.

Pour f un symbole de fonction unaire, on va montrer que sous l'hypothèse que f est une involution, f est une bijection.

Notations :

$Ij[f] := \forall x \forall y (f(x)=f(y) \to x=y)$ ← injection

$S[f] := \forall y \exists x (f(x)=y)$ ← surjection

$B[f] := Ij[f] \wedge S[f]$ ← bijection

$Iv[f] := \forall x (f(f(x))=x)$ ← involution

On va montrer : $Iv[f] \vdash B[f]$

$\boxed{I}$ On suppose que f est une involution.

On cherche à montrer que f est une bijection, c'est-à-dire que f est injective et surjective.

* f est injective :

Soient x et y.

On suppose $f(x)=f(y)$, on cherche alors à montrer que $x=y$

$$x \underset{(i)}{=} f(f(x)) \underset{(ii)}{=} f(f(y)) \underset{(iii)}{=} y$$

(i) et (iii) : car f est une involution

(ii) : on utilise l'égalité $f(x)=f(y)$.

* f est surjective :

Soit y.

On cherche x tel que $f(x)=y$.

$x=f(y)$ convient :

$f(f(y))=y$ car f est une involution.

$\boxed{II}$ On ajoute 2 règles pour pouvoir traiter l'égalité : (Déduction naturelle)

$$\frac{[t=t]}{t=t}\,{=_i} \quad \text{et} \quad \frac{A[x:=t] \quad t=u}{A[x:=u]}\,{=_e}$$

(*) Les seules hypothèses des démonstrations sont $Iv[f]$, dans laquelle ni y ($*_2$) ni x ($*_1$) n'est libre.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f(f(x))=f(f(x))}\,{=_i} \quad f(x)=f(y)}{f(f(x))=f(f(y))}\,{=_e} \quad \dfrac{Iv[f]}{f(f(x))=x}\,\forall_e}{x=f(f(y))}\,{=_e} \quad \dfrac{Iv[f]}{f(f(y))=y}\,\forall_e}{x=y}\,{=_e}}{f(x)=f(y) \to x=y}\,{\to_i}}{\forall y\,(f(x)=f(y) \to x=y)}\,\forall_i *_1}{Ij[f]}\,\forall_i *_2 \quad \dfrac{\dfrac{\dfrac{Iv[f]}{f(f(y))=y}\,\forall_e}{\exists x\,(f(x)=y)}\,\exists_i}{S[f]}\,\forall_i *_1}{B[f]}\,\wedge_i$$

trop compliqué (peut-être ok si plus clairs) + long

Dev 2 : LK <=> NK

Ref : David, Nour, Raffalli
→ Th 5.3.1 (⇒), 5.3.3 (II), 5.3.5 (I), 5.3.6 (⇐) et 5.3.7

(9/8) 1/2

Pour Γ un ensemble fini de formules et A une formule de la logique du 1er ordre :

$$\Gamma \vdash_{NK} A \quad \text{si et seulement si} \quad \Gamma \vdash_{LK} A.$$

on regarde la dernière règle appliquée

$\boxed{\Rightarrow}$ On montre que si $\Gamma \vdash_{NK} A$ alors $\Gamma \vdash_{LK} A$.

On transforme une dérivation de $\Gamma \vdash_{NK} A$ en une dérivation $\Gamma \vdash_{LK} A$.

→ par induction sur la taille de la dérivation de $\Gamma \vdash_{NK} A$.

→ on a juste à traduire (toutes) les règles de NK dans LK.

ax, aff, (intro) ⇝ dte

* Certaines règles de NK sont directement dans LK (ce sont des cas particuliers) :

$ax_{NK} \to ax_{LK}$; $\to_i \to \to_d$; $\wedge_i \to \wedge_d$; $\forall_i \to \forall_d$; $\exists_i \to \exists_d$; $aff \to aff_g$

* On traduit les autres règles (à l'aide de coupures, d'affaiblissements et de contractions) :

(1) règles d'introduction restantes :

$$\frac{A}{A \vee B}\vee_i^g \rightsquigarrow \frac{\dfrac{\Gamma \vdash A}{\Gamma \vdash A, B}\,aff_d}{\Gamma \vdash A \vee B}\vee_d$$

(idem pour $\vee_i^d$)

$$\frac{\begin{array}{c}[A]\\ \vdots \\ \bot\end{array}}{\neg A}\neg_i \rightsquigarrow \frac{\dfrac{\begin{array}{c}\vdots\\ \Gamma, A \vdash \bot\end{array} \quad \dfrac{}{\bot \vdash}\bot_g}{\Gamma, A \vdash}\,coupure}{\Gamma \vdash \neg A}\neg_d$$

(2) Autres règles :

(elim) ⇝ gder (+ coup. et aff)

$$\frac{\begin{array}{c}[\neg A]\\ \vdots \\ \bot\end{array}}{A}\bot_c \rightsquigarrow \frac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma, \neg A \vdash \bot\end{array} \quad \dfrac{}{\bot \vdash}\bot_g}{\Gamma, \neg A \vdash}\,coupure \quad \dfrac{\dfrac{}{A \vdash A}\,ax}{\vdash \neg A, A}\neg_d}{\Gamma \vdash A}\,coupure}{}$$

$$\frac{\begin{array}{c}\vdots\\ \neg A\end{array} \quad \begin{array}{c}\vdots\\ A\end{array}}{\bot}\neg_e \rightsquigarrow \frac{\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash \neg A\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash A\end{array}}{\Gamma, \neg A \vdash}\neg_g}{\Gamma \vdash}\,coupure}{\Gamma \vdash \bot}\,contr_g\ aff_d$$

$$\frac{\begin{array}{c}\vdots\\ A \to B\end{array} \quad \begin{array}{c}\vdots\\ A\end{array}}{B}\to_e \rightsquigarrow \frac{\begin{array}{c}\vdots\\ \Gamma \vdash A \to B\end{array} \quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash A\end{array}}{\Gamma \vdash A, B}\,aff_d \quad \dfrac{}{\Gamma, B \vdash B}\,ax}{\Gamma, A \to B \vdash B}\to_g}{\Gamma \vdash B}\,coupure, contr_g$$

Pour les dernières, il suffit de faire une coupure pour faire apparaître la règle correspondante et la preuve obtenue par hypothèse d'induction. □

$\boxed{\Leftarrow}$ On montre que si $\Gamma \vdash_{LK} \Delta$ alors $\Gamma, \neg\Delta \vdash_{LJ+\bot_c} \bot$ avec Δ un ensemble fini de formules. (I)

On obtient donc que si $\Gamma \vdash_{LK} A$ alors $\Gamma \vdash_{LJ+\bot_c} A$ par application de $\bot_c$.

On montre enfin que si $\Gamma \vdash_{LJ+\bot_c} A$ (resp. $\Gamma \vdash_{LJ+\bot_c}$) alors $\Gamma \vdash_{NK} A$ (resp. $\Gamma \vdash_{NK} \bot$). (II)

$\boxed{I}$ De la même façon qu'avant, on transforme une dérivation de $\Gamma \vdash_{LK} \Delta$ en une dérivation $\Gamma, \neg\Delta \vdash_{LJ+\bot_c} \bot$ par induction sur la taille de la dérivation, en étudiant chaque règle.

* Les règles gauches n'ont pas besoin de traduction
* Affd et Contrd correspondent exactement à Affg et Contrg vu que toutes les formules sont à gauche
* Les règles droites (sauf $\vee_d$) sont juste limitées à une formule à droite dans $LJ+\bot_c$.

On les traduit en commençant par un affd pour ne plus avoir le $\bot$, on fait sortir la formule de gauche à droite, grâce à $\neg_g$, on applique la règle, puis on rebascule tout à gauche grâce à $\bot_c$. Par exemple :

meilleur exple

$$\frac{\begin{array}{c}\vdots\\ \Gamma \vdash A, \Delta\end{array} \quad \begin{array}{c}\vdots\\ \Gamma \vdash B, \Delta\end{array}}{\Gamma \vdash A \wedge B, \Delta}\wedge_d \rightsquigarrow \frac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma, \neg\Delta, \neg A \vdash \bot\end{array}}{\Gamma, \neg\Delta \vdash A}\bot_c \quad \dfrac{\begin{array}{c}\vdots\\ \Gamma, \neg\Delta, \neg B \vdash \bot\end{array}}{\Gamma, \neg\Delta \vdash B}\bot_c}{\Gamma, \neg\Delta \vdash A \wedge B}\wedge_d}{\Gamma, \neg\Delta, \neg(A \wedge B) \vdash}\neg_g}{\Gamma, \neg\Delta, \neg(A \wedge B) \vdash \bot}\,aff_d$$

$A \wedge B$ $\wedge_e^g$ ⇝ A

$\Gamma \vdash A \wedge B$

$\Gamma \vdash A$

$$\frac{\dfrac{A \vdash A}{A, B \vdash A}\,aff}{A \wedge B \vdash A}\wedge_g$$

coup

$\Gamma, \Delta \vdash A$

* La règle $\vee_d$ de LK est partagée en deux parties dans LJ : on doit la traiter directement :

$$\dfrac{\vdots}{\Gamma \vdash A, B, \Delta}\ \vee_d \text{ en } (LK) \qquad \dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee_d^1 (LJ) \quad \dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee_d^2 (LJ) \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma, \neg\Delta, \neg A, \neg B \vdash \bot}\ \bot_c}{\Gamma, \neg\Delta, \neg A \vdash B}\ \vee_d^2}{\Gamma, \neg\Delta, \neg A \vdash A \vee B}\ \neg_g}{\Gamma, \neg\Delta, \neg(A \vee B), \neg A \vdash}\ \text{affd}}{\Gamma, \neg\Delta, \neg(A \vee B), \neg A \vdash \bot}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma, \neg\Delta, \neg(A \vee B) \vdash A}\ \bot_c}{\Gamma, \neg\Delta, \neg(A \vee B) \vdash A \vee B}\ \vee_d^1}{\Gamma, \neg\Delta, \neg(A \vee B), \neg(A \vee B) \vdash}\ \neg_g}{\Gamma, \neg\Delta, \neg(A \vee B) \vdash}\ \text{cont}\neg_g}{\Gamma, \neg\Delta, \neg(A \vee B) \vdash \bot}\ \text{affd}}$$

* Il me reste que la règle de coupure à traiter, ce qui est quasiment immédiat :

$$\dfrac{\vdots \quad \vdots}{\Gamma \vdash A, \Delta \quad \Gamma', A \vdash \Delta'}\ \text{coupure } (LK) \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdots}{\Gamma, \neg\Delta, \neg A \vdash \bot}\ \bot_c}{\Gamma, \neg\Delta \vdash A} \quad \dfrac{\vdots}{\Gamma', \neg\Delta', A \vdash \bot}$$
$$\Gamma, \Gamma', \neg\Delta, \neg\Delta' \vdash \bot \ \text{coupure } (LJ)$$

□

$\boxed{\text{II}}$ Encore une fois, on procède par induction sur la dérivation dans $LJ + \bot_c$.
On étudie donc toutes les règles.

* Les règles ax, affg, affd et les règles droites se traduisent quasiment directement :

$ax_{LJ} \rightsquigarrow ax_{NK}$; $affg \rightsquigarrow aff$; $\wedge_d \rightsquigarrow \wedge_i$; $\vee_d^1 \rightsquigarrow \vee_i^g$; $\vee_d^2 \rightsquigarrow \vee_i^d$; $\rightarrow_d \rightsquigarrow \rightarrow_i$; $\neg_d \rightsquigarrow \neg_i$; $\forall_d \rightsquigarrow \forall_i$; $\exists_d \rightsquigarrow \exists_i$

$\bot_g \rightsquigarrow ax_{NK}$; affd $\rightsquigarrow \bot_c$: $\dfrac{}{\bot \vdash}\ \bot_g \rightsquigarrow \dfrac{}{\bot \vdash \bot}\ ax$ et $\dfrac{\Gamma \vdash}{\Gamma \vdash A}\ \text{affd} \rightsquigarrow \dfrac{\vdots}{A}\ \bot_c$ (pas de décharge de A).

* Les règles $\neg_g$, $\vee_g$, $\exists_g$ s'obtiennent en appliquant les règles correspondantes ($\neg_e$, $\vee_e$, $\exists_e$) :

$$\dfrac{\vdots}{\Gamma, \neg A \vdash}\ \neg_g \rightsquigarrow \dfrac{\neg A \quad \dfrac{A \quad \neg A}{A}\ \text{aff}}{\bot}\ \neg_e$$

$$\dfrac{\vdots \quad \vdots}{\Gamma, A \vdash [C] \quad \Gamma, B \vdash [C]}\ \vee_g \rightsquigarrow \dfrac{A \vee B \quad \dfrac{\overset{[A]}{\vdots}\ C \quad A \vee B}{C}\ \text{aff} \quad \dfrac{\overset{[B]}{\vdots}\ C \quad A \vee B}{C}\ \text{aff}}{C}$$

$C = \bot$ si $[C]$ vide

$x \notin FV(\Gamma)$ et $x \notin FV([C])$:

$$\dfrac{\vdots}{\Gamma, A \vdash [C]}\ \exists_g \rightsquigarrow \dfrac{\exists x A \quad \dfrac{\overset{[A]}{\vdots}\ C \quad \exists x A}{C}\ \text{aff}}{C}\ \exists_e$$

* Pour les dernières règles, on se sert de l'enchaînement $\rightarrow_e$ aff, $\rightarrow_i$ comme d'une coupure :

$$\dfrac{\vdots \quad \vdots}{\Gamma \vdash A \quad \Gamma', A \vdash [B]}\ \text{coupure} \rightsquigarrow \dfrac{\dfrac{\overset{(\Gamma)}{\vdots}\ A \quad \Gamma'}{A}\ \text{aff} \quad \dfrac{\dfrac{\overset{[A]\ (\Gamma')}{\vdots}\ B}{A \to B}\ \to_i \quad \Gamma}{A \to B}\ \text{aff}}{(\Gamma, \Gamma')\ B}\ \to_e$$

$$\dfrac{\vdots}{\Gamma, A \vdash [B]}\ \text{cont}\neg_g \rightsquigarrow \dfrac{\dfrac{\overset{(\Gamma, A, A)}{\vdots}\ B}{A \to B}\ \to_i \quad A}{(\Gamma, A)\ B}\ \to_e$$

$$\dfrac{\Gamma \vdash A \quad \Gamma, B \vdash [C]}{\Gamma, A \to B \vdash [C]}\ \to_g \rightsquigarrow \dfrac{\dfrac{A \to B \quad \dfrac{\overset{[B]}{\vdots}\ C}{B \to C}\ \to_i}{B \to C}\ \text{aff} \quad \dfrac{A \to B \quad \dfrac{A \quad A \to B}{A}\ \text{aff}}{B}\ \to_e}{C}\ \to_e$$

$$\dfrac{\vdots}{\Gamma, A[x := t] \vdash [C]}\ \forall_g \rightsquigarrow \dfrac{\dfrac{\dfrac{\overset{[A(x:=t)]}{\vdots}\ C}{A(x:=t) \to C}\ \to_i \quad \forall x A}{A(x:=t) \to C}\ \text{aff} \quad \dfrac{\forall x A}{A[x := t]}\ \forall_e}{C}\ \to_e$$

$$\dfrac{\Gamma, A, B \vdash [C]}{\Gamma, A \wedge B \vdash [C]}\ \wedge_g \rightsquigarrow \dfrac{\dfrac{\dfrac{\dfrac{\overset{[A], [B]}{\vdots}\ C}{B \to C}\ \to_i}{A \to B \to C}\ \to_i \quad A \wedge B}{A \to B \to C}\ \text{aff} \quad \dfrac{A \wedge B}{A}\ \wedge_e^g}{B \to C}\ \to_e \quad \dfrac{A \wedge B}{B}\ \wedge_e^d}{C}\ \to_e$$

□

On a en plus que si Γ est un ensemble fini de formules et $A_1, \dots, A_m$ sont des formules, alors

si $\Gamma \vdash_{LK} A_1, \dots, A_m$, on peut prouver $\Gamma \vdash_{NK} A_1 \vee \dots \vee A_m$.

Ref : David, Nour, Raffalli
→ Partie 7.2

dvt 3
1/2

Dev 3 : Algorithme d'unification

← trop long

On cherche à unifier un ensemble d'équations

Soit E un ensemble d'équations.

On construit par récurrence la suite $(E_i, \sigma_i)_{i \geq 0}$ avec $\forall i \geq 0$, E_i un ensemble d'équations et σ_i une substitution.

* $E_0 = E$ et $\sigma_0 = Id$
* Soit $m \geq 0$, on suppose qu'on a construit (E_m, σ_m).

(a) Si $E_m = E' \sqcup \{f(u_1, \dots, u_q) \sim g(v_1, \dots, v_p)\}$
  si $f = g$ ($\Rightarrow q = p$)
    ↳ $E_{m+1} = E' \cup \{u_1 \sim v_1, \dots, u_q \sim v_q\}$ et $\sigma_{m+1} = \sigma_m$
  sinon
    ↳ renvoyer "Echec clash"

(b) Si $E_m = E' \sqcup \{x \sim x\}$
  ↳ $E_{m+1} = E'$ et $\sigma_{m+1} = \sigma_m$

(c) Si $E_m = E' \sqcup \{x \sim u\}$ avec $x \neq u$ ou $E_m = E' \sqcup \{u \sim x\}$ avec $x \neq u$
  si x n'apparait pas dans u
    ↳ $E_{m+1} = E'[x := u]$ et $\sigma_{m+1} = [x := u] \circ \sigma_m$
  sinon
    ↳ renvoyer "Echec occur-check"

Sinon
  ↳ renvoyer σ_m

Rq : L'algorithme n'est pas déterministe, mais l'ordre n'a pas d'importance pour le résultat
La complexité est exponentielle.

Prop 1 : L'algorithme termine toujours, soit par un échec, soit car $\exists m \in \mathbb{N}$, $E_m = \emptyset$.

On note, pour $m \in \mathbb{N}$ pour lequel (E_m, σ_m) est défini :
- → a_m = le nombre de variables dans E_m.
- → b_m = le nombre de symboles de fonction dans E_m.
- → c_m = le nombre d'équations dans E_m.
- → $f(m) = (a_m, b_m, c_m)$.

Alors pour $m \in \mathbb{N}$, avec (E_m, σ_m) défini, $E_m \neq \emptyset$ et pas d'échec à l'étape m,
$f(m+1) < f(m)$ pour l'ordre lexicographique :

(a) : $a_{m+1} = a_m$ mais $b_{m+1} = b_m - 1 \Rightarrow f(m+1) < f(m)$

(b) : $a_{m+1} = a_m$ ou $a_{m+1} = a_m - 1 \Rightarrow f(m+1) < f(m)$ (x n'apparait pas dans E')
  ↳ $b_{m+1} = b_m$ et $c_{m+1} = c_m - 1 \Rightarrow f(m+1) < f(m)$ (x apparait dans E')

(c) : $a_{m+1} = a_m - 1 \Rightarrow f(m+1) < f(m)$ (on remplace les x par u). □

Th 2 : Si l'algorithme termine avec $E_m = \emptyset$, alors σ_m est le mgu (most general unifier) de E.
S'il échoue (par clash ou occur-check), alors E n'a pas d'unificateur.

Pour cela, on va montrer la propriété H_m, $\forall m \in \mathbb{N}$ pour lequel (E_m, σ_m) est défini :
H_m = à l'étape m, σ unifie E ssi il existe σ' qui unifie E_m et telle que $\sigma = \sigma' \circ \sigma_m$.

I On note m la dernière étape de l'algorithme.

Si $\forall 0 \leq m \leq m$, H_m est vraie alors le Th 2 est vrai :

1 - Si l'algorithme termine avec $E_m = \emptyset$:
* σ_m unifie E : vu que Id unifie E_m alors par H_m, $\sigma = Id \circ \sigma_m = \sigma_m$ unifie E
* σ_m est un mgu de E : par H_m : σ unifie E ssi il existe σ' (qui unifie $\emptyset$) tq $\sigma = \sigma' \circ \sigma_m$ □1.

2. Si l'algorithme échoue par clash :

On a une équation du type $f(u_1, \dots, u_q) \sim g(v_1, \dots, v_p)$ avec $f \neq g$ dans E_m.

$\Rightarrow E_m$ n'a pas d'unificateur car une substitution ne peut pas changer un symbole de fonction.

$\Rightarrow$ par H_m, E n'a pas d'unificateur. □2.

3. Si l'algorithme échoue par occur-check :

On a une équation du type $x \sim u$ ou $u \sim x$ avec $u \neq x$ et x apparait dans u.

Pour toute substitution σ, la taille de $x[\sigma]$ est plus petite que $u[\sigma]$.

(au moins un symbole de fonction dans u, et x apparait).

$\Rightarrow x \sim u$ (ou $u \sim x$) ne peut pas être unifié

$\Rightarrow E_m$ n'a pas d'unificateur

$\Rightarrow$ par H_m, E n'a pas d'unificateur □3.

□I.

$\boxed{\text{II}}$ Il reste à montrer $H_m, \forall 0 \leq m \leq m$. On le fait par récurrence.

1. $\sigma_0 = Id$ et $E_0 = E$.

H_0 : σ unifie E ssi il existe σ' qui unifie E tq $\sigma = \sigma' \circ Id = \sigma'$

$\rightarrow$ ok. □1.

2. Soit $0 \leq m < m$, on suppose H_m, soit σ une substitution.

$H_{m+1} \Leftrightarrow$ (σ unifie E ssi il existe σ' qui unifie E_{m+1} tq $\sigma = \sigma' \circ \sigma_{m+1}$)

$\Leftrightarrow$ (il existe σ'' qui unifie E_m tq $\sigma = \sigma'' \circ \sigma_m$ ssi il existe σ' qui unifie E_{m+1} tq $\sigma = \sigma' \circ \sigma_{m+1}$)

par H_m

L'algorithme n'échoue pas à l'étape $m+1$ car $m+1 \leq m$, donc on peut définir (E_{m+1}, σ_{m+1}).

(a). $E_m = E' \sqcup \{f(u_1, \dots, u_q) \sim f(v_1, \dots, v_q)\}$

$\Rightarrow E_{m+1} = E' \sqcup \{u_1 \sim v_1, \dots, u_q \sim v_q\}$ et $\sigma_{m+1} = \sigma_m$

$\Rightarrow$ σ' unifie E_m tq $\sigma = \sigma' \circ \sigma_m \Leftrightarrow \sigma'$ unifie E_{m+1} tq $\sigma = \sigma' \circ \sigma_m = \sigma' \circ \sigma_{m+1}$.

(b). $E_m = E' \sqcup \{x \sim x\}$

$\Rightarrow E_{m+1} = E'$ et $\sigma_{m+1} = \sigma_m$

$\Rightarrow$ ok.

(c). $E_m = E' \sqcup \{x \sim u\}$ ou $E_m = E' \sqcup \{u \sim x\}$ avec $u \neq x$ et x n'apparait pas dans u.

$\Rightarrow E_{m+1} = E'[x := u]$ et $\sigma_{m+1} = [x := u] \circ \sigma_m$

* si σ'' unifie E_m tq $\sigma = \sigma' \circ \sigma_m$

$\Rightarrow x[\sigma''] = u[\sigma'']$

$\Rightarrow x[\sigma'' \circ [x := u]] = x[x := u][\sigma''] = u[\sigma''] = x[\sigma'']$

et si $y \neq x$, $y[\sigma'' \circ [x := u]] = y[x := u][\sigma''] = y[\sigma'']$ $\Rightarrow \sigma'' \circ [x := u] = \sigma''$.

$\Rightarrow \sigma = \sigma'' \circ \sigma_m = \sigma'' \circ [x := u] \circ \sigma_m = \sigma'' \circ \sigma_{m+1}$

Etant donné que σ'' unifie $E_m \Rightarrow \sigma''$ unifie E_{m+1}.

* si σ' unifie E_{m+1} tq $\sigma = \sigma' \circ \sigma_{m+1} = (\sigma' \circ [x := u]) \circ \sigma_m$

$\Rightarrow \sigma' \circ [x := u]$ unifie donc E_m. □2.

□II.

□ Th2.

Lorsque l'algorithme de Caprasse