927 Exemples de preuves d'algorithmes : correction, terminaison.

[DNR] : David Nour Raffalli [AJ] : Luc Albert, Cours et ex. d'info

[C] : Cormen [W] : Winskel

Considérons un problème algorithmique donné sous la forme d'une <u>spécification</u> :

<u>Entrée</u> : L'ensemble des objets concernés par le problème

<u>Sortie</u> : Une description de la réponse au problème étant donnée une entrée.

Si A est un algorithme censé résoudre le problème, on peut s'intéresser à deux propriétés de A :

* <u>Terminaison</u> : Si e est une entrée, l'exécution de A sur e s'arrête-t-elle après un nombre fini d'étapes ?
* <u>Correction</u> : Si e est une entrée et l'exécution de A s'arrête en un nombre fini d'étapes, A renvoie-t-il la sortie correcte ?

On pourrait ensuite (mais on ne le fera pas) s'interroger sur la complexité de A (nombre d'étapes avant arrêt).

I | Terminaison :

1) Ensembles bien fondés :

[A] II 1

<u>Déf 1</u> : Un ensemble ordonné $(E, \leq)$ est dit <u>bien fondé</u> s'il n'existe pas de suite strictement décroissante dans E.

*<u>Exple 2</u> : * $(\mathbb{N}, \leq)$ est bien fondé.

* Si $(E, \leq)$ et $(E', \leq')$ sont bien fondés, alors l'ordre lexicographique $\leq \times_{lex} \leq'$ sur $E \times E'$ est bien fondé.

Cette notion va nous permettre de formaliser la notion de variant pour prouver la terminaison.

2) Variants de boucle (algorithmes itératifs)

<u>Déf 3</u> : Un <u>variant de boucle</u> est une fonction des variables de la boucle dans un ensemble bien fondé, dont la valeur décroît à chaque tour de boucle.

<u>Prop 4</u> : Si une boucle admet un variant, elle termine.

*<u>Exple 5</u> : Une boucle for admet comme variant : indice final − indice courant

↳ Un peu tordu - mais pas une mauvaise idée

*<u>Exple 6</u> : <u>Algorithme d'Euclide</u> : <u>Entrée</u> : $(a, b) \in (\mathbb{N}^*)^2$
<u>Sortie</u> : $\mathrm{pgcd}(a, b)$

```
Euclide a b = Tant que a ≠ b
                  Si a ≤ b, b ← b − a
                  Sinon a ← a − b
              Renvoyer a
```

<u>Euclide</u> termine : Variant : $\max(a, b)$.

*<u>Exple 7</u> : <u>Algorithme d'unification</u> : [DNR] 7.2.6 et 7.2.8

<u>Entrée</u> : E ensemble fini de termes sur un langage L.

<u>Sortie</u> : L'unificateur principal σ s'il existe, Erreur sinon.

DÉV 1.1

(x variable, u, u_i, v_i termes, f, g symb. de fonctions)

```
Unif E = σ = id_variables.
         Tant que E ≠ ∅
             Si "x=x" ∈ E,  E ← E \ {"x=x"}
             Si "x=u" ∈ E ou "u=x" ∈ E,
                 Si x est dans u, Erreur.
                 Sinon, σ ← [x↦u]∘σ, E ← E[x↦u]
             Si "f(u_1,…,u_n)=g(u_1,…,u_p)" ∈ E,
                 Si f≠g, Erreur.
                 Sinon, E ← E \ {"f(…)=g(…)"} ∪ {u_1=v_1, …, u_n=v_n}
         Renvoyer σ.
```

<u>Unif</u> termine. Variant : (nb_var(E), nb_symb_fcts(E), |E|) avec l'ordre lexicographique sur $\mathbb{N}^3$.

3) Variants de fonctions récursives :

[A] II 5

<u>Thm 8</u> : Soit f fonction récursive, $\varphi : \{\text{Entrées}\} \to E$ avec E bien fondé, $\text{Base} = \varphi^{-1}(\{\text{élts min de } \mathrm{Im}\,\varphi\})$.
Si $f(b)$ termine pour tout $b \in \text{Base}$, et si pour toute entrée e, $f(e)$ ne fait intervenir que des $f(x)$ avec $\varphi(x) < \varphi(e)$, alors f termine.

*<u>Exple 9</u> : <u>Fct d'Ackermann</u> : $((m,n) \in \mathbb{N}^2)$ termine

$$\begin{cases} Ack(0,n) = n+1 \\ Ack(m,0) = Ack(m-1,1) \\ Ack(m,n) = Ack(m-1, Ack(m, n-1)) \end{cases}$$

Variant : (m, n) avec $\leq_{lex}$

Exple 10: Algorithmes Diviser-pour-Régner :
Si les appels récursifs pour une entrée de taille n se font sur des entrées de taille strictement inférieure (et que les cas de base sont traités), alors l'algorithme termine. Variant : taille de l'entrée.
Exple : Recherche dichotomique dans un tableau trié.

4) Indécidabilité :

Le théorème de l'arrêt montre l'indécidabilité du problème de terminaison des algorithmes :

Thm 11 : Il n'existe pas de programme qui prend en entrée le code d'un programme P et de ses arguments $\bar{x}$ et renvoie Oui si P termine sur l'entrée $\bar{x}$, Non sinon.

Ceci donne une limitation de la preuve automatique de programmes.

II Correction :

On parle de correction partielle pour la correction comme définie en introduction, et de correction totale pour la correction et termin.

conditions d'initialisation et de conservation

1) Invariants de boucle (algorithmes itératifs)

[C] 2.1 **Déf 12 :** Un invariant de boucle est un prédicat des variables de la boucle tel que ① il est vrai en entrée de boucle ; ② s'il est vrai avant un tour de boucle, alors il est vrai après.

DÉV 1.2 [DNR] 72.10 *Exple 13 : L'algorithme d'unification (Exple 7) est totalement correct.
Invariant : σ unifie E_{init} ssi $\exists \sigma'$ qui unifie $E_{courant}$ et $\sigma = \sigma' \circ \sigma_{courant}$

[C] 24.3 *Exple 14 : Algorithme de Dijkstra :
Entrée : Un graphe orienté pondéré sans poids négatifs G, $s \in S(G)$
Sortie : L'ensemble des plus courtes distances de s à x, pour $x \in S(G)$

Dijkstra G s =
- dist(s, s) := 0 ; Pour $x \neq s$, dist(s, x) := ∞ ; Q := S(G).
- Tant que $Q \neq \emptyset$,
  - s_1 := nœud de Q le plus proche de s ; $Q \leftarrow Q \setminus \{s_1\}$
  - Pour x voisin de s_1,
    - Si dist(s, s_2) > dist(s, s_1) + poids(s_1, s_2)
      - dist(s, s_2) = dist(s, s_1) + poids(s_1, s_2)

Correct (totalement)
Invariant : les distances dans $S(G) \setminus Q$ sont les bonnes.

*Exple 15 : Un exemple de correction partielle : non-primalité de Fermat :
Entrée : $n \in \mathbb{N}$
Sortie : Oui si n composé, Non sinon

Fermat n =
- a = 1
- Tant que $a^{n-1} \equiv 1$ [mod n]
  - a := nb aléatoire entre 1 et n−1 ;
- Renvoyer oui

2) Invariants de fonctions récursives

Thm 16 : (Induction bien fondée) Soit (E, <) un ens. bien fondé, et P une propriété des éléments de E. Si pour tout $e \in E$, on a $\forall x \in E,\ x < e \Rightarrow P(x)$, alors P(e) est satisfaite pour tout $e \in E$! [A] Prop II 6

*Exple 17 : Factorielle :
Fact n =
- Si n = 0, renvoyer 1
- Sinon, renvoyer n.(Fact (n−1))

(Invariant : Fact n renvoie n!)

*Exple 18 : Calcul des points les plus proches (Diviser-pour-régner) [C] 33.4
Entrée : Un nuage de points du plan.
Sortie : La distance minimale entre deux de ces points.
Idée de l'algorithme : On sépare les points en deux parties par la droite verticale d'abcisse médiane, on calcule récursivement d_{gauche} et d_{droite}, puis on parcourt, pour chaque point dans la bande de largeur 2δ autour de la médiane, les distances aux 7 points suivants par ordonnée croissante dans la bande.
Invariant : L'algorithme renvoie la distance minimale.

*Exple 19 : Fonction 91 de MacCarthy : $f(n) = \begin{cases} n-10 & \text{si } n > 100 \\ f(f(n+11)) & \text{sinon} \end{cases}$. [A] Ex II 8
Alors $f(n) = 91$ si $n \leq 101$.

Classique de partir de la conjecture de Kolax
[illustre la difficulté de la quest° de terminaison]

III) Logique de Hoare :

La logique de Hoare permet de formaliser les raisonnements de correction partielle de programmes, et d'envisager alors la question de la démonstration automatique.

1) Langage IMP

[W] 2.1 **Déf 20** : Langage IMP — On note V l'ensemble des variables

- Expressions arithmétiques : $a ::= n \mid x \mid a+a \mid a\times a$ où $x \in V$
- Expressions booléennes : $b ::= true \mid false \mid a=a \mid a\leq a \mid \neg b \mid b\wedge b \mid b \vee b$
- Commandes : $c ::= skip \mid x := a \mid c;c \mid \text{if } b \text{ then } c \text{ else } c \mid \text{while } b \text{ do } c$

Déf 21 : On appelle environnement une fonction $\sigma : V \to \mathbb{N}$ et configuration une paire $\langle e, \sigma\rangle$ qui représente le fait que l'on va calculer e dans σ.

Hors Programme

Déf 22 : Sémantique des expressions

$$\frac{}{\langle n, \sigma\rangle \to n} \qquad \frac{}{\langle x, \sigma\rangle \to \sigma(x)} \qquad \frac{\langle a_1, \sigma\rangle \to n_1 \quad \langle a_2, \sigma\rangle \to n_2}{\langle a_1+a_2, \sigma\rangle \to n} \text{ où } n = n_1+n_2$$

On définit de façon analogue la sémantique des autres expressions.

Déf 23 : Sémantique des commandes

$$\frac{}{\langle skip, \sigma\rangle \to \sigma} \qquad \frac{\langle a, \sigma\rangle \to n}{\langle x := a, \sigma\rangle \to \sigma[n/x]}$$

$$\frac{\langle b, \sigma\rangle \to true \quad \langle c_1, \sigma\rangle \to \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma\rangle \to \sigma'} \qquad \frac{\langle c_1, \sigma\rangle \to \sigma'' \quad \langle c_2, \sigma''\rangle \to \sigma'}{\langle c_1; c_2, \sigma\rangle \to \sigma'}$$

$$\frac{\langle b, \sigma\rangle \to false \quad \langle c_2, \sigma\rangle \to \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma\rangle \to \sigma'} \qquad \frac{\langle b, \sigma\rangle \to false}{\langle \text{while } b \text{ do } c, \sigma\rangle \to \sigma}$$

$$\frac{\langle b, \sigma\rangle \to true \quad \langle c, \sigma\rangle \to \sigma'' \quad \langle \text{while } b \text{ do } c, \sigma''\rangle \to \sigma'}{\langle \text{while } b \text{ do } c, \sigma\rangle \to \sigma'}$$

La notation n'est pas nécessaire

Not 24 On notera $[\![P]\!]$ la fonction partielle telle que $[\![P]\!](\sigma) = \sigma'$ ssi $\langle P, \sigma\rangle \to \sigma'$

2) Les règles de Hoare :

Déf 25 : Un triplet de Hoare est une expression de la forme $\{A\}p\{B\}$ avec p un programme IMP, et A et B des formules du premier ordre sur $(+, \times, =)$. [W] 6.3

Déf 26 : On appelle règles de Hoare le système déductif suivant : [W] 6.4

$$\frac{}{\{A\} skip \{A\}}\text{skip} \qquad \frac{\{A\wedge b\} p_1 \{B\} \quad \{A \wedge \neg b\} p_2 \{B\}}{\{A\} \text{if } b \text{ then } p_1 \text{ else } p_2 \{B\}}\text{if}$$

$$\frac{}{\{B[a/x]\}\, x := a \,\{B\}}\text{assign} \qquad \frac{\{A \wedge b\} p \{A\}}{\{A\} \text{while } b \text{ do } p \{A \wedge \neg b\}}\text{while}$$

$$\frac{\{A\} p_1 \{C\} \quad \{C\} p_2 \{B\}}{\{A\} p_1;p_2 \{B\}}\text{seq} \qquad \frac{\models A \Rightarrow A' \quad \{A'\}p\{B'\} \quad \models B' \Rightarrow B}{\{A\}p\{B\}}\text{csq}$$

Déf 27 : On dit que σ satisfait $\{A\}p\{B\}$ si pour toute interprétation I, $(\sigma \models^I A) \Rightarrow ([\![p]\!]\sigma \models^I B)$. On note $\sigma \models \{A\}p\{B\}$. [W] 6.2

- On dit que $\{A\}p\{B\}$ est valide si pour tout σ, $\sigma \models \{A\}p\{B\}$.
- On dit que $\{A\}p\{B\}$ est prouvable s'il existe un arbre de dérivation avec $\{A\}p\{B\}$ à la racine. On note $\vdash \{A\}p\{B\}$.

Thm 28 : Pour tous A, B, p, on a $\vdash \{A\}p\{B\} \Rightarrow \models \{A\}p\{B\}$ [W] 6.5 Thm 6.11

Ceci montre la correction des règles de Hoare. Nous allons à présent montrer leur complétude.

Déf 29 : On dit que A est une plus faible précondition (pfp) de (p, B) si pour tous état σ et interprétation I, $\sigma \models^I A$ ssi $[\![p]\!]\sigma \models^I B$ [W] 7.2

Prop 30 : Pour tous (p, B), il existe une pfp $\varphi(p, B)$ Thm 7.5

Prop 31 : Pour tous (p, B), on a $\vdash \{\varphi(p,B)\}p\{B\}$ Lem. 7.6

Thm 32 : (Complétude) Pour tous A, B, p, on a $\models \{A\}p\{B\} \Rightarrow \vdash \{A\}p\{B\}$ Thm 7.7

DÉV 2

Logique de Hoare = au pgme

MAIS sa correction/complétude ... → pas au pgme (en parler si on est sûr)

⚠ Le langage des assertions n'est pas défini ! (cf déf 25)

⚠ Il faut avoir csce des limites du langage IMP simple [↳ pas de pointeur ; réf ; mémoire]

↳ On peut parler d'automatisation ? [Plus faible précondition] → on peut automatiser mais seult en partie → fournir invariants de boucles + obligations de preuves aux interfaces [WHY 3]

↳ au moins y réfléchir

Complétude de la logique de Hoare

Thm : Pour toutes formules A, B et programme p, on a $\models \{A\} p \{B\} \Longrightarrow \vdash \{A\} p \{B\}$

On va démontrer :

Lemme ① : Si $\models \{A\} p \{B\}$ et A_0 est une pfp de (p,B), alors $\models (A \Rightarrow A_0)$

Lemme ② : $\forall (p,B)$, $\exists \varphi(p,B)$ pfp de (p,B), et on a $\vdash \{\varphi(p,B)\} p \{B\}$

Ceci implique le théorème :

Dém (Thm) : D'après ① et ②, si $\models \{A\} p \{B\}$, alors $\models (A \Rightarrow \varphi(p,B))$.
Donc :

$$\dfrac{\models A \Rightarrow \varphi(p,B) \qquad \overline{\{\varphi(p,B)\} p \{B\}}\ ②}{\{A\} p \{B\}}$$

Dém ① : Soit σ tel que $\sigma \models A$. Comme $\models \{A\} p \{B\}$, on a $[\![p]\!]\sigma \models B$, donc $\sigma \models A_0$, d'où $\models (A \Rightarrow A_0)$

Dém ② : Par induction sur p :

* $p =$ skip : $\varphi(p,B) = B$

$\overset{\text{pfp}}{\rightsquigarrow}$ $\sigma \models \varphi(p,B)$ ssi $[\![p]\!]\sigma = \sigma \models B$.

$\underset{\vdash}{\rightsquigarrow}$ $\overline{\{B\} \text{ skip } \{B\}}\ \text{skip}$

* $p =$ "$x := a$" : $\varphi(p,B) = B[a/x]$

$\overset{\text{pfp}}{\rightsquigarrow}$ $\sigma \models B[a/x]$ ssi $\sigma[x \leftarrow a] \models B$

$\underset{\vdash}{\rightsquigarrow}$ $\overline{\{B[a/x]\} p \{B\}}\ \text{assign}$

* $p = p_1 ; p_2$: $\varphi(p,B) = \varphi(p_1, \varphi(p_2, B))$

$\overset{\text{pfp}}{\rightsquigarrow}$ $\sigma \models \varphi(p,B)$ ssi $[\![p_1]\!]\sigma \models \varphi(p_2,B)$
ssi $[\![p_2]\!]([\![p_1]\!]\sigma) \models B$
ssi $[\![p]\!]\sigma \models B$

Qu'est φ. → Écrit comme ça, ça va pas

$\underset{\vdash}{\rightsquigarrow}$ $$\dfrac{\overline{\{\varphi(p,B)\} p_1 \{\varphi(p_2,B)\}}\ \text{ind} \qquad \overline{\{\varphi(p_2,B)\} p_2 \{B\}}\ \text{ind}}{\{\varphi(p,B)\} p \{B\}}\ \text{seq}$$

* $p =$ if b then p_1 else p_2 : $\varphi(p,B) = (b \wedge \varphi(p_1,B)) \vee (\neg b \wedge \varphi(p_2,B))$

$\overset{\text{pfp}}{\rightsquigarrow}$ $\sigma \models \varphi(p,B)$ ssi $(\sigma \models b$ et $[\![p_1]\!]\sigma \models B)$ ou $(\sigma \models \neg b$ et $[\![p_2]\!]\sigma \models B)$
ssi $[\![p]\!]\sigma \models B$

$\underset{\vdash}{\rightsquigarrow}$ $$\dfrac{\dfrac{\vdots}{\{\varphi(p,B) \wedge b\} p_1 \{B\}}\ \text{csq} \qquad \dfrac{\vdots}{\{\varphi(p,B) \wedge \neg b\} p_2 \{B\}}\ \text{csq}}{\{\varphi(p,B)\} p \{B\}}\ \text{if}$$

(car $\varphi(p,B) \wedge b \equiv \varphi(p_1,B)$ et $\varphi(p,B) \wedge \neg b \equiv \varphi(p_2,B)$)

* $p =$ while b do p_1 : φ

On remarque que $[\![p]\!]\sigma \models B$ si et seulement si il existe $\sigma_0, \dots, \sigma_k$ tels que $\sigma = \sigma_0$ et $\forall i < k, \sigma_i \models b$ et $[\![p_1]\!]\sigma_i = \sigma_{i+1}$, et $\sigma_k \models \neg b \wedge B$. On encode alors les σ_i grâce à des suites d'entiers en utilisant la fonction β de Gödel. On admet que ceci est possible (cf [W] thm 7.5, p 105). Ceci donne donc $\varphi(p,B)$ une pfp de (p,B).

$\underset{\vdash}{\rightsquigarrow}$ On va montrer que : ⓐ $\models \{\varphi(p,B) \wedge b\} p_1 \{\varphi(p,B)\}$
et ⓑ $\models (\varphi(p,B) \wedge \neg b) \Rightarrow B$

ⓐ Si $\sigma \models \varphi(p,B) \wedge b$, alors $[\![p]\!]\sigma \models B$ et $\sigma \models b$, donc* $[\![p]\!]\sigma = [\![p]\!]([\![p_1]\!]\sigma)$, d'où $[\![p_1]\!]\sigma \models \varphi(p,B)$.

ⓑ Si $\sigma \models \varphi(p,B) \wedge \neg b$, alors $[\![p]\!]\sigma \models B$ et $\sigma \models \neg b$, donc* $[\![p]\!]\sigma = \sigma$, d'où $\sigma \models B$.

Conclusion : $$\dfrac{\overline{\{\varphi(p,B) \wedge b\} p_1 \{B\}}\ \text{csq, ⓐ et ⓑ}}{\{\varphi(p,B)\} p \{B\}}\ \text{while}$$

(* : Si $\sigma \models b$, alors $[\![p]\!]\sigma$ revient à faire $[\![p]\!]([\![p_1]\!]\sigma)$
Sinon, cela revient à faire skip)

↳ <u>Poser les défs</u>, comme elles ne sont pas ds le plan (langage ...?).

Algorithme d'unification

Entrée : E ensemble d'équations
Sortie : σ un unificateur principal de E s'il existe, un échec sinon

$\sigma := Id$
Tant que $E \neq \emptyset$
  Choisir $e \in E$
  $E' := E \setminus \{e\}$
  Si $e = (f(u_1, \dots, u_p) \sim g(v_1, \dots, v_q))$
    Si $f = g$ (et donc $p = q$)
      $E := E' \cup \{u_1 \sim v_1, \dots, u_p \sim v_p\}$
    Sinon
      Retourner échec 1
  (1)
  Si $e = (x \sim x)$
    $E := E'$
  (2)
  Si $e = (x \sim u)$ ou $e = (u \sim x)$
    Si x n'apparaît pas dans u
      $\sigma := [x := u] \circ \sigma$
      $E := E'[x := u]$
    Sinon
      Retourner échec 2
  (3)
Retourner σ

<u>Lemme</u> Si $x[\sigma] = u[\sigma]$ alors $\sigma = \sigma \circ [x := u]$

<u>Dém</u> Soit $\sigma' = \sigma \circ [x := u]$

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

Si $y \neq x$,

$$y[\sigma'] = y[x := u][\sigma] = y[\sigma]$$

Donc $\sigma = \sigma'$

On note E_n et σ_n les valeurs de E et σ ~~au début du~~ après le $n^{\text{ème}}$ tour de boucle.

<u>Prop</u> L'algorithme termine.

<u>Dém</u> Soit a_n le nombre de variables distinctes dans E_n

b_n le nombre de symboles de fonction dans E_n

$c_n := |E_n|$

Supposons que E_n et E_{n+1} sont définis. Alors,

	$a_{n+1} \overset{?}{\cdot} a_n$	$b_{n+1} \overset{?}{\cdot} b_n$	$c_{n+1} \overset{?}{\cdot} c_n$
1	$=$	$<$	
2	$\leqslant$	$=$	$<$
3	$<$		

Donc $(a_{n+1}, b_{n+1}, c_{n+1}) < (a_n, b_n, c_n)$ où $<$ est l'ordre lexicographique (strict) sur $\mathbb{N}^3$
Comme $<$ est bien fondé, l'algorithme termine.

<u>Prop</u> L'algorithme est correct.

Soit H_n : "Si E_n et σ_n sont bien définis,
(σ unifie E ssi il existe σ' qui unifie E_n tel que $\sigma = \sigma' \circ \sigma_n$)"
pour $n \in \mathbb{N}$.

- H_0 vraie car $\sigma_0 = Id$ et $E_0 = E$
- Supposons H_n pour un $n \in \mathbb{N}$ et que E_{n+1} et σ_{n+1} sont bien définis

(1) et (2) On a $\sigma_{n+1} = \sigma_n$ donc il suffit de montrer que σ' unifie E_n ssi σ' unifie E_{n+1}. Et c'est clair par définition de la substitution pour les termes.

(3)
- Soit σ'' unifiant E_{n+1}. Alors $\sigma' = \sigma'' \circ [x := u]$ unifie E_n et $\sigma = \sigma'' \circ \sigma_{n+1} = \sigma'' \circ [x := u] \circ \sigma_n = \sigma' \circ \sigma_n$
- Soit σ' unifiant E_n. On a $x[\sigma'] = u[\sigma']$ et donc, par le lemme, $\sigma' = \sigma' \circ [x := u]$.
D'où $\sigma = \sigma' \circ \sigma_n = \sigma' \circ [x := u] \circ \sigma_n = \sigma' \circ \sigma_{n+1}$.

Donc H_{n+1}

Donc par récurrence, $\forall n \in \mathbb{N}$, H_n

Soit n la dernière étape. Comme $E_n = \emptyset$, Id unifie E_n donc, par H_n, σ_n unifie E.
Si σ unifie E, il existe σ' tel que $\sigma = \sigma' \circ \sigma_n$ donc σ_n est un unificateur principal.

Si on a un échec de type 1, E_n contient $f(u_1, \dots, u_r) \sim g(v_1, \dots, v_q)$ avec $f \neq g$ donc E_n n'est pas unifiable et donc, par H_n, E n'est pas unifiable.

Si on a un échec de type 2, E_n contient $x \sim u$ avec $u \neq x$ et x apparaît dans u. Alors pour tout σ, $|x[\sigma]| < |u[\sigma]|$ donc $x[\sigma] \neq u[\sigma]$. Donc E_n et E ne sont pas unifiables.

Réfs

- Théo Pierron
- Introduction à la logique, David, Nour, Raffalli, p. 236

Exs classiques (dvpt).

- KMP
- Hopcroft [automate minimal]
- Dijkstra
- Lièvre et tortue
- Preuve de Hoare d'un pgme (fact, fibo, tri insert°) ⚠ BS

Si on ne fait pas, ne pas parler de sémantique. ← - Complétude ou Correct° Hoare

on peut les mettre ds le plan, même sans les mettre en dvpt.