

Cours 1: Algorithmes. Modèles de calcul.

Olivier Bournez

bournez@lix.polytechnique.fr
LIX, Ecole Polytechnique

Au menu

Objectifs du cours INF561

Quelques algorithmes séquentiels

Forme des algorithmes séquentiels

Un tout petit peu de logique

Qu'est ce qu'un algorithme séquentiel ?

Algorithme sur une structure

Quelques questions

- Quel est l'algorithme le plus rapide pour résoudre un problème donné? Comment cet algorithme dépend t'il du langage et de la machine utilisée?
- Peut-on remplacer une recherche exhaustive par un algorithme plus efficace?
- La complexité dépend t'elle des opérations élémentaires autorisées, et comment?
- L'utilisation d'algorithmes randomisés permet-il d'accélérer les algorithmes? de résoudre plus de problèmes?
- Peut-on résoudre les problèmes plus rapidement si l'on autorise les algorithmes à commettre des erreurs sur une faible fraction de leurs entrées?

Au menu des cours

- Comprendre ce qu'est un algorithme.
- Comprendre comment discuter
 - ▶ de l'existence d'algorithmes
 - ▶ de la complexité d'un algorithme :
 - comparaisons : réduction, complétude
 - bornes inférieures
- Comprendre les algorithmes sur une structure arbitraire.
- Comprendre comment on peut établir des bornes inférieures.
- Comprendre la puissance du
 - ▶ non-déterminisme
 - ▶ des algorithmes randomisés
 - ▶ distribués
 - ▶ ...

Au menu

Objectifs du cours INF561

Quelques algorithmes séquentiels

Forme des algorithmes séquentiels

Un tout petit peu de logique

Qu'est ce qu'un algorithme séquentiel ?

Algorithme sur une structure

Sous menu

Quelques algorithmes séquentiels

L'algorithme d'Euclide

L'algorithme de Syracuse

L'ensemble de Mandelbrot

Résoudre des équations

L'algorithme d'Euclide

```
read < a, b >
repeat
  if b = 0 then out ← a
  else par
    a ← b
    b ← a mod b
  endpar
until out!=undef
write out
```

■ Sur \mathbb{N} :

- ▶ Propriété : $\text{pgcd}(a, b) = \text{pgcd}(b, a \bmod b)$.
- ▶ Propriété : $a \bmod b < b$, pour $b \neq 0$.

L'algorithme d'Euclide

```
read < a, b >
repeat
  if b = 0 then out ← a
  else par
    a ← b
    b ← a mod b
  endpar
until out != undef
write out
```

■ Sur $\mathbb{K}[X]$:

- ▶ Propriété : $\text{pgcd}(a, b) = \text{pgcd}(b, a \bmod b)$.
- ▶ Propriété : $\text{deg}(a \bmod b) < \text{deg}(b)$, pour $b \neq 0$.

L'algorithme d'Euclide

```
read < a, b >
repeat
  if b = 0 then out ← a
  else par
    a ← b
    b ← a mod b
  endpar
until out!=undef
write out
```

- Sur $\mathbb{K}[X]$:
 - ▶ Propriété : $\text{pgcd}(a, b) = \text{pgcd}(b, a \bmod b)$.
 - ▶ Propriété : $\text{deg}(a \bmod b) < \text{deg}(b)$, pour $b \neq 0$.
- plus généralement sur tout anneau euclidien A . $\forall a, b \in A$,
 $\exists q, r \in A$ tel que $a = bq + r$, et il existe d'autre part une fonction
 $\nu : A - \{0\} \rightarrow \mathbb{N}$ telle que soit $r = 0$ ou $\nu(r) < \nu(b)$.

Sous menu

Quelques algorithmes séquentiels

L'algorithme d'Euclide

L'algorithme de Syracuse

L'ensemble de Mandelbrot

Résoudre des équations

Algorithme de Syracuse

```
read  $n$   
repeat  
  if  $n = 1$  then  $\text{out} \leftarrow 1$   
  else if  $n \bmod 2 = 0$  then  $n \leftarrow n \text{ div } 2$   
  else  $n \leftarrow 3 * n + 1$   
until  $\text{out} \neq \text{undef}$   
write  $\text{out}$ 
```

Sur \mathbb{N} :

- On ne connaît aucun entier n pour lequel il ne termine pas.

Sous menu

Quelques algorithmes séquentiels

L'algorithme d'Euclide

L'algorithme de Syracuse

L'ensemble de Mandelbrot

Résoudre des équations

L'ensemble de Mandelbrot

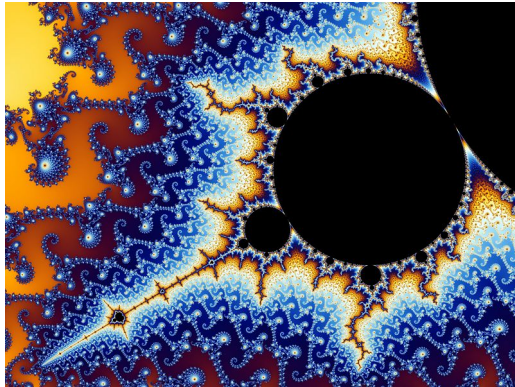
```
read c
z ← 0
repeat
  if |z| > 2 then out ← true
  else z ← z^2 + c
until out != undef
write out
```

Sur \mathbb{C} :

- L'ensemble des complexes c pour lequel il termine correspond à *l'ensemble de Mandelbrot*.
- Il s'agit d'un ensemble fractal, compact, connexe, d'intérieur non vide, conjecturé localement connexe¹

¹Tout voisinage de tout point contient un voisinage connexe.

L'ensemble de Mandelbrot



L'ensemble de Mandelbrot

Sur $(\mathbb{C}, +, -, *, /, =)$:

- L'ensemble de Mandelbrot est récursivement énumérable :
 - ▶ il existe un algorithme qui termine pour c dans l'ensemble : l'algorithme précédent est solution.
- Il n'est pas récursif
 - ▶ on ne peut pas construire d'algorithme qui termine sur son complémentaire.

Sous menu

Quelques algorithmes séquentiels

L'algorithme d'Euclide

L'algorithme de Syracuse

L'ensemble de Mandelbrot

Résoudre des équations

Équations polynomiales à une inconnue

- Le problème : on se donne $P \in \mathbb{R}[X]$, et on veut savoir si P possède une racine réelle.
- Solution :

```
read < a, b, c >
repeat
  if  $b^2 - 4 * a * c > 0$  then out ← vrai
  else out ← faux
until out!=undef
write out
```

- Il existe un algorithme de décision pour $P \in \mathbb{R}[X]$ de degré arbitraire.
- Théorie de Galois : pas de formule explicite par radicaux pour les solutions dans le cas général pour un degré ≥ 5 .

Équations linéaires à n inconnues

- L'algorithme du pivot de Gauss est un grand classique pour résoudre les systèmes d'équations linéaires.

$$\begin{cases} A[1, 1]x_1 + \cdots + A[1, m]x_m = B[1] \\ \cdots \\ A[n, 1]x_1 + \cdots + A[n, m]x_m = B[n] \end{cases}$$

- Il fonctionne sur \mathbb{Q} , sur \mathbb{R} , et en fait sur tout corps \mathbb{K} .
- Il fonctionne en temps $\mathcal{O}(n^3)$.
 - ▶ Strassen a proposé un algorithme en temps de l'ordre de $\mathcal{O}(n^{2.81})$.
 - ▶ Le meilleur algorithme connu fonctionne en temps $\mathcal{O}(n^{2.376})$.
- Sur \mathbb{Z}_2 , un algorithme brutal, moins efficace en $\mathcal{O}(2^n)$: tester tous les x_1, \cdots, x_m .

```

read < A, B >
i ← 1
j ← 1
repeat
  while (i ≤ m and j ≤ n) do
    // Trouver le pivot dans la colonne j,
    //      en partant de la ligne i:
    maxi ← i
    for k ← i+1 to m do
      if abs(A[k,j]) > abs(A[maxi,j]) then maxi ← k
    endfor
    if A[maxi,j] ≠ 0 then
      [ échanger ligne i et maxi de A et B ]
      // A[i,j] contient l'ancienne valeur de A[maxi,j].
      [ diviser chaque coeff. de la ligne i de A et B
        par A[i,j] ]
      // A[i,j] a maintenant la valeur 1.
      for u ← i+1 to m do
        [ soustraire A[u,j]* ligne i de la ligne u de A
          et B ]
        // A[u,j] est maintenant 0
      endfor
      i ← i + 1
    endif
    else out ← faux
      j ← j + 1
  endwhile
out ← vrai

```

Équations polynomiales à n inconnues

- On se donne P_1, \dots, P_n de $\mathbb{K}[X_1, \dots, X_m]$, et l'on veut savoir si le système P_1, \dots, P_n possède une solution, c'est-à-dire si P_1, \dots, P_n possèdent une racine commune dans \mathbb{K}^m .
- Sur \mathbb{Z}_2 :
 - ▶ Il existe un algorithme brutal en $\mathcal{O}(2^n)$ opérations : tester tous les x_1, \dots, x_m .
 - ▶ Aucun algorithme polynomial n'est connu.
 - ▶ Le problème est *NP*-complet.
- Sur \mathbb{R} :
 - ▶ Il existe un algorithme exponentiel (en $\max(n, m)$) pour résoudre ce problème sur le corps \mathbb{R} des réels.
 - ▶ Aucun algorithme polynomial n'est connu.
 - ▶ Le problème est $\text{NP}_{\mathbb{R}}$ complet.

Au menu

Objectifs du cours INF561

Quelques algorithmes séquentiels

Forme des algorithmes séquentiels

Un tout petit peu de logique

Qu'est ce qu'un algorithme séquentiel ?

Algorithme sur une structure

Sous menu

Forme des algorithmes séquentiels
La notion d'étape élémentaire
Résumé

Forme de nos algorithmes

Les algorithmes considérés sont de la forme

```
read <donnée>  
repeat  
  <instructions >  
until out!=undef  
write out
```

- On peut éliminer les boucles **for**, **while**, et les séquences.

La notion d'étape élémentaire : boucle for

```
1 for k ← 1 to m do  
2   <instruction>  
3   ...  
4 endfor  
5 ...
```

est un synonyme de

```
if ctl_state = 1 then {k ← 1 ctl_state ← 2}  
else if ctl_state = 2 and (k ≤ m) then  
  {<instruction> ctl_state ← 3}  
...  
else if ctl_state = 2 and (k > m) then ctl_state ← 5  
else if ctl_state = 4 then {k ← k + 1 ctl_state ← 2}  
...
```

où une variable *ctl_state* code l'état courant, et {...} signifie **par**
... **endpar**, c'est-à-dire la mise en parallèle.

La notion d'étape élémentaire : boucle while

```
1 while <condition>  
2   <instruction>  
3   ...  
4 endwhile  
5 ...
```

est un synonyme de

```
if ctl_state = 1 and not <condition> {ctl_state ← 5}  
else if ctl_state = 1 then  
  {<instruction>  ctl_state ← 2}  
...
```

La notion d'étape élémentaire : séquence

Lorsque l'on écrit

```
1 seq  
2 <instruction>  
3 <instruction '>  
4 endseq
```

on signifie :

```
if ctl_state = 2 then {<instruction> ctl_state ← 3}  
else if ctl_state = 3 then {<instruction '>  
  ctl_state ← 5}  
...
```

Sous menu

Forme des algorithmes séquentiels

La notion d'étape élémentaire

Résumé

Forme de nos algorithmes

```
read <donnée>  
repeat  
  <instructions >  
until out!=undef  
write out
```

avec (peut-être) une variable *ctl_state*.

Intérêt : le temps d'un algorithme correspond alors au nombre d'itérations de la boucle **repeat**.

Au menu

Objectifs du cours INF561

Quelques algorithmes séquentiels

Forme des algorithmes séquentiels

Un tout petit peu de logique

Qu'est ce qu'un algorithme séquentiel ?

Algorithme sur une structure

Sous menu

Un tout petit peu de logique

Cas fonctionnel : Structure du premier ordre

Termes (clos)

Interprétations

Digression : Cas général : Structure du premier ordre

Digression : Termes et formules libres (closes)

Digression : Interprétations

Cas général vs cas fonctionnel

Notion de signature : cas fonctionnel

- Une *signature* Σ (*fonctionnelle d'un langage du premier ordre*) est la donnée d'un ensemble \mathcal{F} de *symboles de fonctions*;
- A chaque symbole de fonction est associé un entier appelé *arité* (un nombre d'arguments).
- Une fonction d'arité 0 (sans argument) s'appelle *une constante*.

Exemple : $\Sigma = (\{0, 1, +, -\})$ est une signature.

Sous menu

Un tout petit peu de logique

Cas fonctionnel : Structure du premier ordre

Termes (clos)

Interprétations

Digression : Cas général : Structure du premier ordre

Digression : Termes et formules libres (closes)

Digression : Interprétations

Cas général vs cas fonctionnel

Notion de terme (clos)

- Un *terme (clos)* est un arbre,
 - ▶ dont les feuilles sont étiquetées par les symboles de constantes,
 - ▶ dont les nœuds internes ayant k fils sont étiquetés par un symbole de fonction ou de relation d'arité k .
- Si l'on préfère, un terme (clos) est de la forme
 - ▶ c où c est une constante.
 - ▶ $f(t_1, t_2, \dots, t_k)$ où f est un symbole de fonction d'arité k , et t_1, t_2, \dots, t_k sont des termes (clos).

Exemples :

- ▶ $+(1, -(0, 1))$ est un terme clos sur la signature $\Sigma = (\{0, 1, +, -\})$.
- ▶ $f(a, b, g(a))$ est un terme clos sur la signature $\Sigma = (\{0, 1, a, b, f, g\})$.

Sous menu

Un tout petit peu de logique

Cas fonctionnel : Structure du premier ordre

Termes (clos)

Interprétations

Digression : Cas général : Structure du premier ordre

Digression : Termes et formules libres (closes)

Digression : Interprétations

Cas général vs cas fonctionnel

Notion de modèle : cas fonctionnel

- Une *réalisation* d'une signature fonctionnelle Σ (encore appelé *modèle* de Σ , ou encore *structure (du premier ordre) sur Σ*) est la donnée
 - ▶ d'un ensemble non vide M , appelé *ensemble de base*,
 - ▶ et pour chaque symbole de fonction $f \in \mathcal{F}$ d'arité n d'une fonction $\llbracket f \rrbracket : M^n \rightarrow M$, appelée *interprétation de f* ,

Exemple : une réalisation de la signature $\Sigma = (\{\mathbf{0}, \mathbf{1}, +, -\})$ correspond à l'ensemble de base \mathbb{N} des entiers naturels, où $\mathbf{0}$ est interprété par l'entier 0, $\mathbf{1}$ par 1, $+$ par l'addition, $-$ par la soustraction.

Interprétation des termes

- Étant donné un terme t l'interprétation du terme t , notée $\llbracket t \rrbracket$ est un élément de M qui se définit inductivement comme on s'y attend :
 - ▶ si t est un symbole de constante c alors $\llbracket t \rrbracket$ est $\llbracket c \rrbracket$;
 - ▶ sinon t est de la forme $f(t_1, \dots, t_k)$, pour un symbole de fonction ou de relation f , et alors $\llbracket f \rrbracket$ est $\llbracket f \rrbracket(\llbracket t_1 \rrbracket, \dots, \llbracket t_k \rrbracket)$.

Exemple : $(\mathbf{1} + \mathbf{1}) + \mathbf{1}$, c'est-à-dire l'arbre $+(+(\mathbf{1}, \mathbf{1}), \mathbf{1})$, s'interprète par l'entier 3.

Sous menu

Un tout petit peu de logique

Cas fonctionnel : Structure du premier ordre

Termes (clos)

Interprétations

Digression : Cas général : Structure du premier ordre

Digression : Termes et formules libres (closes)

Digression : Interprétations

Cas général vs cas fonctionnel

Notion de signature : cas général

- Une *signature* $\Sigma = (\mathcal{F}, \mathcal{R})$ (d'un langage du premier ordre) est la donnée de deux ensembles de symboles :
 - ▶ les éléments du premier ensemble \mathcal{F} sont appelés les *symboles de fonctions* ;
 - ▶ les éléments du second ensemble \mathcal{R} sont appelés les *symboles de prédicats* (aussi appelés *symboles de relations*).
- A chaque symbole de fonction et de prédicat est associé un entier appelé *arité* (un nombre d'arguments).
- Une fonction d'arité 0 (sans argument) s'appelle *une constante*.

Exemple : $\Sigma = (\{\mathbf{0}, \mathbf{1}, +, -\}, \{>, =\})$ est une signature.

Sous menu

Un tout petit peu de logique

Cas fonctionnel : Structure du premier ordre

Termes (clos)

Interprétations

Digression : Cas général : Structure du premier ordre

Digression : Termes et formules libres (closes)

Digression : Interprétations

Cas général vs cas fonctionnel

Termes et formules libres (closes)

- Un *terme (clos)* est ...comme avant.
- Une *formule libre² (close)* ϕ est de la forme
 - ▶ $r_i(t_1, \dots, t_k)$, où $r_i \in \mathcal{R}$ est un symbole de relation d'arité k , t_1, t_2, \dots, t_k sont des termes;
 - ▶ $\neg\psi$ où ψ est une formule (close)
 - ▶ $\psi \vee \chi$ où ψ et χ sont des formules (closes).
 - ▶ $\psi \wedge \chi$ où ψ et χ sont des formules (closes).
- Dans le premier cas, la formule est dite *atomique*.

Exemple : $1 + 1 > 0$ est une formule libre de la signature $\Sigma = (\{\mathbf{0}, \mathbf{1}, +, -\}, \{>, =\})$.

²libre = sans quanteur.

Notion de modèle : cas général

- Une *réalisation* d'une signature $\Sigma = (\mathcal{F}, \mathcal{R})$ (encore appelé *modèle* de Σ , ou encore *structure (du premier ordre) sur Σ*) est la donnée
 - ▶ d'un ensemble non vide M , appelé *ensemble de base*,
 - ▶ et pour chaque symbole de fonction $f \in \mathcal{F}$ d'arité n d'une fonction $\llbracket f \rrbracket : M^n \rightarrow M$, appelée *interprétation de f* ,
 - ▶ et pour chaque symbole de prédicat $r \in \mathcal{R}$ d'arité n d'un sous-ensemble $\llbracket r \rrbracket$ de M^n , appelé *interprétation de r* .

Exemple : une réalisation de la signature $\Sigma = (\{\mathbf{0}, \mathbf{1}, +, -\}, \{>, =\})$ correspond à l'ensemble de base \mathbb{N} des entiers naturels, où $\mathbf{0}$ est interprété par l'entier 0, $\mathbf{1}$ par 1, $+$ par l'addition, $-$ par la soustraction, $=$ par l'égalité sur les entiers et $>$ par l'ordre sur les entiers.

Sous menu

Un tout petit peu de logique

Cas fonctionnel : Structure du premier ordre

Termes (clos)

Interprétations

Digression : Cas général : Structure du premier ordre

Digression : Termes et formules libres (closes)

Digression : Interprétations

Cas général vs cas fonctionnel

Interprétation des termes et formules libres

- L'interprétation des termes est ...comme avant.
- On dit qu'une structure satisfait ϕ si
 - ▶ $(\llbracket t_1 \rrbracket, \dots, \llbracket t_k \rrbracket) \in \llbracket r \rrbracket$ lorsque ϕ est de la forme $r_i(t_1, \dots, t_k)$
 - ▶ ne satisfait pas ψ lorsque ϕ est de la forme $\neg\psi$
 - ▶ satisfait ψ ou satisfait χ lorsque ϕ est de la forme $\psi \vee \chi$
 - ▶ satisfait ψ et satisfait χ lorsque ϕ est de la forme $\psi \wedge \chi$

Exemple : $\neg 1 + 1 = 0$ est satisfaite dans l'interprétation précédente.

Sous menu

Un tout petit peu de logique

Cas fonctionnel : Structure du premier ordre

Termes (clos)

Interprétations

Digression : Cas général : Structure du premier ordre

Digression : Termes et formules libres (closes)

Digression : Interprétations

Cas général vs cas fonctionnel

Vision alternative : par les fonctions caractéristiques

- Convention : \mathcal{F} contient au moins un symbole de constante **0**, et un symbole de constante **1**.
- Notons 1 pour l'interprétation de **1** (*vrai*), et 0 pour l'interprétation de **0** (*faux*), supposés distinctes.
- On peut voir une relation comme une fonction : on voit une relation par sa fonction caractéristique à valeur dans $\{0, 1\}$.

Conséquence : Meilleure définition

- Une *signature* Σ est la donnée d'un ensemble de *symboles de fonctions*; Certaines fonctions sont qualifiées de *relationnelles*.
- On peut écrire $\Sigma = (\mathcal{F}, \mathcal{R})$, où \mathcal{R} sont les fonctions relationnelles, et \mathcal{F} les autres.
- On n'a plus que des signatures fonctionnelles
 - ▶ ...on peut parler de termes (clos)
 - ▶ ...chaque formule atomique devient un terme
 - ▶ ...

Exemple : $\Sigma = (\{\mathbf{0}, \mathbf{1}, +, -\}, \{>, =\})$ est une signature.

Convention d'écriture

On écrira

$$\mathfrak{M} = (M, \mathcal{F}, \mathcal{R})$$

pour la structure sur la signature $\Sigma = (\mathcal{F}, \mathcal{R})$, d'ensemble de base M , lorsque l'interprétation des symboles de \mathcal{F} et de \mathcal{R} est claire.

- Exemple 1 : $(\mathbb{R}, +, -, *, /, >, =)$
- Exemple 2 : $(\mathbb{N}, +, -, =, >)$.

Notion d'isomorphisme

Soient X et Y deux réalisations d'une même signature Σ .

- Un *isomorphisme* est une fonction injective i de l'ensemble de base de X vers l'ensemble de base de Y telle que $[[f]](i(x_1), \dots, i(x_k)) = i(x_0)$ dans Y à chaque fois que $[[f]](x_1, \dots, x_k) = x_0$ dans X , pour chaque symbole de fonction ou de relation f .

Au menu

Objectifs du cours INF561

Quelques algorithmes séquentiels

Forme des algorithmes séquentiels

Un tout petit peu de logique

Qu'est ce qu'un algorithme séquentiel ?

Algorithme sur une structure

Sous menu

Qu'est ce qu'un algorithme séquentiel ?

Approche axiomatique

Premier postulat

Second postulat

Troisième postulat

Un langage de programmation minimal

Forme normale

Approche axiomatique

- On va énoncer trois postulats.
- Un objet qui vérifie les trois postulats correspond à un algorithme.
- On obtient une forme normale :
 - ▶ un algorithme peut s'écrire avec des
 1. des affectations $f(t_1, \dots, t_n) \leftarrow t$
 2. des mises en parallèles **par** \dots **endpar**
 3. des **if** ϕ **then** R

Qu'est ce qu'un algorithme séquentiel ?

Approche axiomatique

Premier postulat

Second postulat

Troisième postulat

Un langage de programmation minimal

Forme normale

Postulat 1

Postulat (Temps séquentiel)

A tout algorithme A est associé un système de transitions, c'est-à-dire,

- *un ensemble $S(A)$, dont les éléments sont appelés des états,*
- *un sous-ensemble $I(A) \subset S(A)$ dont les éléments sont dits initiaux,*
- *et une application $\tau_A : S(A) \rightarrow S(A)$ que l'on appelle évolution en un pas de A .*

Une *exécution* du système de transition est une suite $X_0, X_1, \dots, X_n, \dots$ où

- X_0 est un état initial ($X_0 \in I(A)$),
- chaque X_i est un état ($X_i \in S(A)$),
- $X_{i+1} = \tau_A(X_i)$ pour tout i .

Exemple d'Euclide

```
read < a, b >
repeat
  if b = 0 then out ← a
  else par
    a ← b
    b ← a mod b
  endpar
until out ≠ undef
write out
```

- Les états correspondent à des triplets, qui correspondent aux valeurs de a , b et out .

Vers le postulat 2

- Il nous faut dire que les états correspondent à quelque chose que l'on sait décrire en langage mathématique.

Qu'est ce qu'un algorithme séquentiel ?

Approche axiomatique

Premier postulat

Second postulat

Troisième postulat

Un langage de programmation minimal

Forme normale

Postulat 2

Postulat (Etats abstraits)

- *Les états de A correspondent à des structures du premier ordre.*
- *Les états sont tous sur la même signature.*
- *La fonction d'évolution en un pas τ_A ne modifie pas l'ensemble de base d'aucun état.*
- *$S(A)$ et $I(A)$ sont clos par isomorphisme. De plus, tout isomorphisme de X vers Y est un isomorphisme de $\tau_A(A)$ vers $\tau_A(Y)$.*

Exemple : Pour l'algorithme d'Euclide, les états correspondent à des structures du premier ordre sur la signature $\Sigma = (\{a, b, out, mod\}, \{=\})$. Une interprétation de cette signature fixe un triplet de valeurs pour a, b et out .

Structures versus mémoire

Il est très pratique de voir une structure comme une certaine mémoire :

- si $a \in \Sigma$ est un symbole de constante, on peut voir a comme le nom d'un variable : sa sémantique $\llbracket a \rrbracket$ correspond à sa valeur.
- si $f \in \Sigma$ est un symbole de fonction d'arité k , son interprétation $\llbracket f \rrbracket$ est une fonction de M^k dans M : on peut voir cette fonction comme un tableau de dimension k .

Aussi, étant donné un k -uplet \bar{m} d'éléments de M ,

- une paire (f, \bar{m}) sera appelée *emplacement* ;
- on appelle *contenu* la valeur de $\llbracket f \rrbracket(\bar{m})$, que l'on notera parfois $\llbracket (f, \bar{m}) \rrbracket$.

Mises à jour

- Si (f, \bar{m}) est un emplacement, et si b est un élément de M , alors on appellera³ $(f, \bar{m}) \leftarrow b$ une *mise à jour*.
- Exécuter une mise à jour $(f, \bar{m}) \leftarrow b$ consiste à changer $[[f]](\bar{m})$ en b .
- Pour exécuter un ensemble de mise à jour consistant, on exécute simultanément toutes les mises à jour de l'ensemble.
 - ▶ Le résultat de l'exécution de l'ensemble de mises à jour Δ sur la structure X sera noté $X + \Delta$.

³Autrement dit, $(f, \bar{m}) \leftarrow b$ est une notation du triplet (f, \bar{m}, b) .

Passer d'un état au suivant correspond à des mises à jour

Lemma

Si X et Y sont deux structures de même vocabulaire et de même ensemble de base, il existe un unique ensemble de mises à jour non triviales consistantes tel que $Y = X + \Delta$, noté $Y - X$

Preuve : Δ est l'ensemble des mises à jour $(f, \bar{m}) \leftarrow b$, pour tous les f et \bar{m} tels que $\llbracket f \rrbracket(\bar{m})$ différent dans X et dans Y , avec b choisi comme la valeur de $\llbracket f \rrbracket(\bar{m})$ dans Y .

- En particulier, lorsque X est un état, on notera

$$\Delta(A, X) = \tau_A(X) - X,$$

c'est-à-dire l'ensemble des mises à jour qui correspondent à passer de l'état X à son successeur.

Qu'est ce qu'un algorithme séquentiel ?

Approche axiomatique

Premier postulat

Second postulat

Troisième postulat

Un langage de programmation minimal

Forme normale

Postulat 3

Deux états (donc deux structures) X et Y coïncident sur un ensemble de termes T si $\llbracket t \rrbracket$ possède la même valeur dans la structure X et dans la structure Y pour tout $t \in T$.

Postulat (Exploration bornée)

Il existe un ensemble fini de termes T sur le vocabulaire de l'algorithme A , tel que $\Delta(A, X) = \Delta(A, Y)$ à chaque fois que les états X et Y de A coïncident sur T .

Dans notre algorithme d'Euclide, on peut prendre

$T = \{b = 0, a, b, 0, 1, a \bmod b, \mathbf{out}\}$.

Définition formelle d'un d'algorithme

Definition

Un algorithme (séquentiel déterministe) est un objet A qui satisfait les trois postulats.

Exemple : les algorithmes précédents sont des algorithmes.

Qu'est ce qu'un algorithme séquentiel ?

Approche axiomatique

Premier postulat

Second postulat

Troisième postulat

Un langage de programmation minimal

Forme normale

Un langage de programmation minimal

Soit Σ une signature. Une règle sur Σ est soit :

- une *règle de mise à jour* de la forme

$$f(t_1, \dots, t_k) \leftarrow t_0;$$

- une *règle parallèle* de la forme

$$\mathbf{par} \ R_1 \ R_2 \ \dots \ R_k \ \mathbf{endpar};$$

- une règle de la forme

$$\mathbf{if} \ \phi \ \mathbf{then} \ R_1.$$

où f est un symbole k -aire de la signature Σ , et t_0, \dots, t_k sont des termes sur la signature Σ , R_1, \dots, R_k sont des règles, ϕ est un terme booléen (= une formule atomique).

Rappel : Forme des algorithmes

```
read <donnée>  
repeat  
  <instructions >  
until out!=undef  
write out
```

Qu'est ce qu'un algorithme séquentiel ?

Approche axiomatique

Premier postulat

Second postulat

Troisième postulat

Un langage de programmation minimal

Forme normale

Forme normale

Theorem (Forme normale d'un algorithme)

Soit Σ une signature. Tout algorithme séquentiel A sur la signature Σ peut s'écrire avec $\langle \text{instructions} \rangle$ réduit à une règle de la forme

```
par  
  if  $\phi^{X_1}$  then  $R^{X_1}$   
  if  $\phi^{X_2}$  then  $R^{X_1}$   
  ...  
  if  $\phi^{X_m}$  then  $R^{X_m}$   
endpar
```

chaque R^{X_i} étant de la forme **par** $\langle \text{affectations} \rangle$ **endpar**, où $\langle \text{affectations} \rangle$ est une suite de règles d'affectation.

Exemple

L'algorithme d'Euclide peut aussi s'écrire

```
read < a, b >
repeat
  par
    if  $b=0$  then  $d \leftarrow a$ 
    if  $b \neq 0$  then
      par
         $a \leftarrow b$ 
         $b \leftarrow a \bmod b$ 
      endpar
    endpar
  until out!=undef
write out
```

En fait la forme normale obtenue par le théorème contient beaucoup plus de cas : on distingue toutes les classes d'équivalence des termes modulo leur coïncidence sur T .

Au menu

Objectifs du cours INF561

Quelques algorithmes séquentiels

Forme des algorithmes séquentiels

Un tout petit peu de logique

Qu'est ce qu'un algorithme séquentiel ?

Algorithme sur une structure

Symbole dynamique vs statique

- Un symbole de fonction est *dynamique* si son interprétation peut varier d'un état à son successeur. Il est *statique* dans le cas contraire.
 - ▶ Dans Euclide, *a*, *b* et *out* sont dynamiques, et *mod* est statique

Extension d'une structure

- Une extension d'une structure

$$\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$$

est une structure sur le même ensemble de base mais avec une signature étendue

$$(f_1, \dots, f_u, f'_1, \dots, f'_\ell, r_1, \dots, r_v),$$

où f'_1, \dots, f'_ℓ où chaque f'_j est non-initialisée.

- Formellement :
 - ▶ on a supposé implicitement qu'on a un constante **undef** dont l'interprétation est *undef* (ex : **out** vaut *undef* initialement)).
 - ▶ la fonction $[[f'_j]]$ vaut toujours *undef*.

Algorithme sur une structure \mathfrak{M}

- Nous appellerons *algorithme sur*

$$\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$$

un algorithme sur une extension de \mathfrak{M} .

- C'est à dire :
 - ▶ l'ensemble de base (en particulier de chacun de ses états) est M .
 - ▶ l'algorithme travaille sur une signature $(f_1, \dots, f_u, f'_1, \dots, f'_\ell, r_1, \dots, r_v)$ qui contient éventuellement f'_1, \dots, f'_ℓ , des symboles de fonctions dynamiques.
- Exemple :
 - ▶ L'algorithme d'Euclide sur les réels sera appelé un algorithme sur la structure (\mathbb{R}, mod) .