# Calculabilité & complexité : cours du 24/02/11

⚠ Rappel : on a 3 opérations sur les machines RAM possibles :

- Addition
- Soustraction
- Saut

*Ex :* $7\ mod\ 3 = 1 \rightarrow\ \ soustraction$

$$\left.\begin{matrix} 7-1=6 \\ \frac{6}{3}=2 \end{matrix}\right\} R_2 = R_2 \dot{-}\ 1$$

## 1. Langage while : cf page 30, 3.5.1

Def programme :

- Programme = bloc d'instruction
- Bloc = sequence d'instruction : begin … end {}
- Séquence = une instruction ou une séquence d'instruction prédéfini + une instruction
- Instruction = affectation + fin d'instruction ou fin de while
- Affectation = variable associé à une expression
- Expression = nom ou constantes, avec possibilités opérantes (+ , $\dot{-}$, *, /)

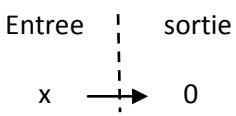
But du cours : pouvoir faire un programme qui peut tout faire avec un nombre d'instruction réduit

Un programme while ne fait que prendre un paramètre en entrée et le rend changé en sortie

⚠ Les variables non définie valent 0 par défaut

Schématisation des exercices du cours :

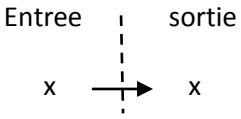
*Exo 46 :*




On ne rentre pas dans le while, donc on obtient la fonction $Z()$

```
Begin
    While le ≠ 0 do
        Begin
            Le = chien + mange
        End
End
```

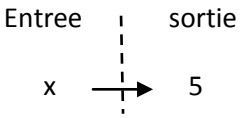
*Exo 47 : Listing 3.6*




On obtient la fonction $Id()$

```
Begin
    Sortie = entrée ;
End
```

*Exo 47 : Listing 3.7*




On obtient la fonction $const = 5$

```
Begin
    Sortie = 5 ;
End
```

## 2. Macro instruction

On va maintenant voir si on peut créer l'instruction IF avec notre schéma while, ainsi savoir si IF es essential ou non

$IF\ nomvar\ \neq 0\ then\ bloc\ 1$

$else\ bloc\ 2$

$tmp = nom\ var;\ \ attention = 1$

$while\ tmp\ \neq 0\ do$

$bloc\ 1$

$tmp = 0; attention = 0$

$while\ attention\ \neq 0\ do$

$bloc\ 2$

$attention = 0$

then

else

Donc IF n'est pas essentiel mais pratique. On va donc imaginer qu'il existe une macro instruction contenant ce code

Rappel : fonction crochet = $< x, y > = \frac{(x+y)(x+y+1)}{2} + y + 1$

C'est une bijection $\mathbb{N} * \mathbb{N} \rightarrow \mathbb{N}$

Elle a donc un inverse qui fait $\mathbb{N} * \mathbb{N} \leftarrow \mathbb{N}$ et qui est $< \prod_g(n), \prod_d(n) > = n$ (partie gauche / droite)

La fonction $\prod_d(n)$ est définie à l'exo 49 (p. 34)

*Exo 50 :*

$n = < x_0 < x_1 < \cdots < x_n, 0 > \cdots > >$

$x_i = \prod_g \prod_d(n)$

## 3. Sémantique (p.38)

La sémantique d'un programme est la fonction de la structure du langage

$(\mathbb{N}^{\mathbb{N}} \rightarrow tab\ infini\ d'entier)$

Def de la sémantique d'un programme

Elle prend en entrée une instruction, le tableau courant avec toutes les valeurs imaginables et changera la variable concerné par l'exécution de l'instruction

En sortie elle donne une tableau d'entier ou $\perp$

$\Longrightarrow$ Définition inductive des langages

- Sémantiques d'instruction :

$$|[\ C]|(x)_i \begin{cases} constante\ si\ i = w(mavar) \\ x_i\ sinon \end{cases} \Longrightarrow PPR$$

- Sémantique de sequence d'instruction

$$|[C_1]|, |[C_2]|, \dots, |[C_n]| \qquad \Longrightarrow PPR$$
$$|[C_n]| \circ \dots \circ |[C_2]| \circ |[C_1]|(x)$$

- Sémantique bloc

On définie d'abord l'instruction while

$C = while\ mavar\ \neq 0\ do$ $\quad |[Bloc]|$ $\quad \Longrightarrow PPR$

$bloc$ $\quad |[C]|(x)$

$\mu_y \prod_n^i |[Bloc]|^y(x) \quad i = w(mavar)$ } minimalisation de y qui rend nulle cette fonction

$\hookrightarrow renvoi\ le\ 1°\ zero$

Le nombre de fois où on exécute la boucle : $|[C]|^{\mu_y \prod_n^i |[Bloc]|^y(x)}(x)$

⚠ La fonction calculée par chaque programme while est PPR, par composition de fonction

## 4. Machines RAM

Def Machines RAM

$RAM \subseteq ou \supseteq ? PPR$

$\| \ ?$

$while \subseteq ou \supseteq ? PPR$

$R_3 = 1$

$IF\ R_3 \neq 0\ then\ GOTOF\ 2$

$comp(3)$

$IF\ R_3 \neq 0\ then\ GOTOB$

Ainsi on sait tout traduire en sémantique

Donc $\left.\begin{matrix} RAM \\ \cup \mid \\ while \end{matrix}\right\} while\ sous\ ensemble\ RAM$

On veut démontrer que pour chaque programme while il existe un programme RAM qui fait la même chose

Différence entre interprète et compilateur

- Un compilateur traduit tout un code d'un coup
- Un interprète traduit ligne par ligne de manière interactive

Définition d'un interprète p. 43 – 44

↪ Simule le fonctionnement d'une machine RAM

Donc $RAM = while \subseteq PPR$