IFT-2002

Informatique Théorique

H14 - cours 9

Julien Marcil - julien.marcil@ift.ulaval.ca

Aujourd'hui

- Deux modèles calculatoires simples
  - Les programmes `RÉPÉTER`
  - Les programmes `TANTQUE`
- La fonction d’Ackermann

Les programmes

RÉPÉTER

Les programmes `RÉPÉTER`

- Un nombre arbitrairement grand de registres est disponible: $r_0, r_1, \ldots$
- Chaque registre contient un entier positif ou nul
- les registres sont implicitement initialisés à 0 avant utilisation

Les instructions d'un programmes RÉPÉTER

- l'instruction $r_i \leftarrow r_j$ remplace le contenu du registre r_i par celui de r_j
- l'instruction $\mathtt{inc}(r_i)$ incrémente de 1 le registre r_i
- l'instruction $\mathtt{répéter}\ r_i\ \mathtt{fois}\ [\langle\mathtt{BLOC}\rangle]$ répète l'éxécution d'un bloc d'instructions r_i fois
    - le nombre d'exécution de $\langle\mathtt{BLOC}\rangle$ est fixe une fois pour toutes avant l'entrée dans la boucle, que r_i y soit modifié ou non

Les programmes RÉPÉTER

Un programme RÉPÉTER implante une fonction

$$\begin{aligned} f\,:\; \mathbb{N}\times\mathbb{N}\times\cdots\times\mathbb{N} &\;\to\; \mathbb{N} \\ (r_1,\; r_2,\; \ldots,\; r_k) &\;\mapsto\; r_0 \end{aligned}$$

Au debut de l'exécution, les registres r_1 à r_k contiennent les arguments de f , et à la fin, r_0 contient $f(r_1, \ldots, r_k)$.

Grammaire

$$
\begin{aligned}
S &\to \langle\texttt{INCRÉMENTATION}\rangle S \mid \lambda \mid \\
&\quad\ \langle\texttt{AFFECTATION}\rangle S \mid \langle\texttt{RÉPÉTER}\rangle S \\
\langle\texttt{INCRÉMENTATION}\rangle &\to \texttt{inc}(V) \\
\langle\texttt{AFFECTATION}\rangle &\to V \leftarrow V \\
\langle\texttt{RÉPÉTER}\rangle &\to \texttt{répéter}\ V\ \texttt{fois}\ [S] \\
V &\to r_N \\
N &\to C \mid CN \\
C &\to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Opérations Arithmétiques

Il est assez facile de faire des programmes `RÉPÉTER` pour des opérations arithmétiques simples.

Addition

$\texttt{PLUS}(r_1, r_2) = r_1 + r_2$

$r_0 \leftarrow r_1$

$\texttt{répéter}\ r_2\ \texttt{fois}$ [

$\quad\texttt{inc}(r_0)$

]

Multiplication

$\texttt{MULT}(r_1, r_2) = r_1 \times r_2$

```
répéter r₁ fois [
    répéter r₂ fois [
        inc(r₀)
    ]
]
```

Exponentiation

$\texttt{EXP}(r_1, r_2) = r_1^{r_2}$

$\texttt{inc}(r_0)$

$\texttt{répéter}\ r_2\ \texttt{fois}\ [$

  $r_3 \leftarrow r_4$

  $\texttt{répéter}\ r_0\ \texttt{fois}\ [$

    $\texttt{répéter}\ r_1\ \texttt{fois}\ [$

      $\texttt{inc}(r_3)$

    $]$

  $]$

  $r_0 \leftarrow r_3$

$]$

Sucre syntaxique

L'instruction $r_i \leftarrow \texttt{PROC}(r_{j_1}, \ldots, r_{j_k})$ signifie que l'on doit substituer à cette ligne un bloc d'instructions qui a pour effet de remplacer le contenu du registre r_i par la valeur calculee par $\texttt{PROC}(r_{j_1}, \ldots, r_{j_k})$, en renommant au besoin les variables qui apparaissent dans le code de la procédure $\texttt{PROC}$.

Les appels recursifs ne sont pas permis.

Sucre syntaxique

L'instruction $r_i \leftarrow k$ signifie que l'on doit substituer à cette ligne k incrémentations, ce qui aura pour effet d'affecter la constante k au registre r_i

Partout, on peut mettre une constante k au lieu d'utiliser une variable auxiliaire qu'on aurait incrémentée k fois.

Exponentiation

$\mathtt{EXP}(r_1, r_2) = r_1^{r_2}$

$r_0 \leftarrow 1$
$\texttt{répéter } r_2 \texttt{ fois [}$
$\quad r_0 \leftarrow \mathtt{MULT}(r_0, r_1)$
]

Opérations Arithmétiques

Est-il possible de faire n'importequel opérations arithmétiques avec des programmes `RÉPÉTER`?

Décrémentation

$\texttt{DEC}(r_1) = \max(0, r_1 - 1)$

```
répéter r₁ fois [
    r₀ ← r₂
    inc(r₂)
]
```

Soustraction

$\mathtt{MOINS}(r_1, r_2) = \max(0, r_1 - r_2)$

$r_0 \leftarrow r_1$

$\mathtt{répéter}\ r_2\ \mathtt{fois}$ [

$\quad r_0 \leftarrow \mathtt{DEC}(r_0)$

]

Factorielle

$\mathtt{FACT}(r_1) = r_1!$

$r_0 \leftarrow 1$

$\texttt{répéter}\ r_1\ \texttt{fois}$ [

  $\texttt{inc}(r_2)$

  $r_0 \leftarrow \mathtt{MULT}(r_0, r_2)$

]

Variables Booléennes

Nous adoptons les conventions syntaxiques suivantes :

- `vrai` pour la constante 1
- `faux` pour la constante 0

Pour évaluer $\langle\texttt{BLOC}\rangle$ conditionnellement à la valeur booléennes r_i on répète $\langle\texttt{BLOC}\rangle$ r_i fois.

L'instruction `si` r_i `alors` $[\langle\texttt{BLOC}\rangle]$ sera mise pour `répéter` r_i `fois` $[\langle\texttt{BLOC}\rangle]$.

Et

$\mathtt{ET}(r_1, r_2) = r_1 \wedge r_2$

$r_0 \leftarrow \mathtt{MULT}(r_1, r_2)$

Negation

$\mathtt{NEG}(r_1) = \neg r_1$

$r_0 \leftarrow \mathtt{MOINS}(1, r_1)$

Ou

$$\mathtt{OU}(r_1, r_2) = r_1 \vee r_2 \;\; = \neg(\neg r_1 \wedge \neg r_2)$$

$$r_1 \leftarrow \mathtt{NEG}(r_1)$$

$$r_2 \leftarrow \mathtt{NEG}(r_2)$$

$$r_0 \leftarrow \mathtt{ET}(r_1, r_2)$$

$$r_0 \leftarrow \mathtt{NEG}(r_0)$$

Plus grand que

$\texttt{PG?}(r_1, r_2) = (r_1 > r_2)$

$r_3 \leftarrow \texttt{MOINS}(r_1, r_2)$

$\texttt{répéter}\ r_3\ \texttt{fois}\ [$

$\quad r_0 \leftarrow \texttt{vrai}$

$]$

Opérations Arithmétiques

Regardons d'autres opérations arithmétiques plus complèxes.

Division

$\mathtt{DIV}(r_1, r_2) = \left\lfloor \frac{r_1}{r_2} \right\rfloor$

$\mathtt{répéter}\ r_1\ \mathtt{fois}$ [
   $r_3 \leftarrow \mathtt{PLUS}(r_3, r_2)$
   $r_4 \leftarrow \mathtt{PG?}(r_3, r_1)$
   $r_4 \leftarrow \mathtt{NEG}(r_4)$
   $\mathtt{si}\ r_4\ \mathtt{alors}$ [
      $\mathtt{inc}(r_0)$
   ]
]

Modulo

$\texttt{MOD}(r_1, r_2) = r_1 \bmod r_2$

$r_0 \leftarrow \texttt{DIV}(r_1, r_2)$

$r_0 \leftarrow \texttt{MULT}(r_0, r_2)$

$r_0 \leftarrow \texttt{MOINS}(r_1, r_0)$

Test de primalité

$\texttt{PREMIER?}(r_1) = (r_1 \in \mathbb{P}\quad)$

$r_0 \leftarrow \texttt{faux}$
$r_5 \leftarrow \texttt{PG?}(r_1, 1)$
$\texttt{si}\ r_5\ \texttt{alors}\ [$
  $r_0 \leftarrow \texttt{vrai}$
  $r_3 \leftarrow 1$
  $r_2 \leftarrow \texttt{MOINS}(r_1, 2)$
  $\texttt{répéter}\ r_2\ \texttt{fois}\ [$
    $\texttt{inc}(r_3)$
    $r_4 \leftarrow \texttt{MOD}(r_1, r_3)$
    $r_5 \leftarrow \texttt{PG?}(1, r_4)$
    $\texttt{si}\ r_5\ \texttt{alors}\ [r_0 \leftarrow \texttt{faux}]$
  $]$
$]$

Prochain nombre premier

$\mathtt{PREMIERSUIV}(r_1) =$ le plus petit nombre premier plus grand que r_1

```
r₂ ← PLUS(r₁, 1)
r₂ ← MULT(r₂, 2)
r₃ ← vrai
répéter r₂ fois [
   inc(r₁)
   r₄ ← PREMIER?(r₁)
   r₄ ← ET(r₃, r₄)
   si r₄ alors [
      r₀ ← r₁
      r₃ ← faux
   ]
]
```

k-ème nombre premier

$\mathtt{PREMIERK}(r_1) =$ le r_1-ème nombre premier

```
répéter r₁ fois [
    r₀ ← PREMIERSUIV(r₀)
]
```

`tableau`

Un tableau d'entiers est un k-tuple $(a_1, a_2, \dots, a_k)$ que nous stockons dans un registre r_j.

Codage de Gödel

Il est possible d'encoder le k-tuple $(a_1, a_2, \ldots, a_k)$ où $a_i \in \mathbb{N}$ dans un entier.

Soit p_n le n-ième nombre premier. Le k-tuple $(a_1, a_2, \ldots, a_k)$ est représenté sant ambiguïté par l'entier

$$p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$$

Exemples

$$
\begin{aligned}
(1,1,1) &= 2^1 3^1 5^1 &&= 30 \\
(2,1,1,1) &= 2^2 3^1 5^1 7^1 &&= 420 \\
(2,0,4,3,0,3) &= 2^2 3^0 5^4 7^3 11^0 13^3 &&= 1883927500
\end{aligned}
$$

Extraction d'un élément d'un `tableau`

$\mathtt{TABLVAL}(r_1, r_2) =$ le r_2-ème élément du `tableau` r_1

$r_3 \leftarrow \mathtt{PREMIERK}(r_2)$
$r_4 \leftarrow r_3$
$\mathtt{répéter}$ r_1 $\mathtt{fois}$ [
  $r_5 \leftarrow \mathtt{MOD}(r_1, r_4)$
  $r_5 \leftarrow \mathtt{PG?}(1, r_5)$
  $\mathtt{si}$ r_5 $\mathtt{alors}$ [
    $\mathtt{inc}(r_0)$
    $r_4 \leftarrow \mathtt{MULT}(r_3, r_4)$
  ]
]

Assignation d'un élément dans un `tableau`

$\mathtt{TABLASS}(r_1, r_2, r_3) =$ le `tableau` r_1 où r_2 -ème élément est remplacé par r_3

$$r_4 \leftarrow \mathtt{TABLVAL}(r_1, r_2)$$
$$r_5 \leftarrow \mathtt{PREMIERK}(r_2)$$
$$r_6 \leftarrow \mathtt{EXP}(r_5, r_4)$$
$$r_0 \leftarrow \mathtt{DIV}(r_1, r_6)$$
$$r_7 \leftarrow \mathtt{EXP}(r_5, r_3)$$
$$r_0 \leftarrow \mathtt{MULT}(r_0, r_7)$$

Puissance des programmes `RÉPÉTER`

Il semble que les programmes `RÉPÉTER` peuvent calculer des fonctions complexes.

Peut-on calculer toutes les fonctions à valeurs entières avec un programme `RÉPÉTER`?

Primitives Récursives

Définition: Les fonctions calculables par un programme `RÉPÉTER` sont appelées **primitives récursives**.

Notation

Pour une fonction $f : \mathbb{N} \to \mathbb{N}$, et un entier n, on note :

$$\begin{aligned}
f^{\langle 0 \rangle}(x) &= x \\
f^{\langle 1 \rangle}(x) &= f(x) \\
f^{\langle 2 \rangle}(x) &= f(f(x)) \\
&\vdots \\
f^{\langle n \rangle}(x) &= \underbrace{f(f(\cdots f(x)\cdots))}_{n \text{ fois}}
\end{aligned}$$

Remarque

Soit $k, n \in \mathbb{N}$ et $k \leq n$

$$f^{\langle n \rangle}(x) = f^{\langle n-k \rangle}\left(f^{\langle k \rangle}(x)\right)$$

Remarque

Soit $n, u \in \mathbb{N}$.

$$\left(f^{\langle n \rangle}\right)^{\langle u \rangle}(x) = f^{\langle nu \rangle}(x)$$

Boucles imbriquées

Définition: Soit la fonction $B_i : \mathbb{N} \to \mathbb{N}$ pour $i \geq 0$

$$B_i(x) = \begin{cases} 1 & \text{si } i = 0, x = 0 \\ 2 & \text{si } i = 0, x = 1 \\ x + 2 & \text{si } i = 0, x > 1 \\ {B_{i-1}}^{\langle x \rangle}(1) & \text{si } i > 0 \end{cases}$$

Exemples

$$
\begin{aligned}
B_0(x) &= x+2 && \text{si } x > 1\\
B_1(x) &= 2x && \text{si } x > 0\\
B_2(x) &= 2^x && \text{si } x \geq 0\\
B_3(x) &= \underbrace{2^{2^{2^{\cdots}}}}_{n \text{ fois}} && \text{si } x > 0
\end{aligned}
$$

Remarques

Il est clair que plus i est grand, plus B_i est une fonction qui croît rapidement.

- La valeur de $B_3(5)$ compte 19729 chiffres.
- La valeur de $B_3(6)$ compte plus de chiffres que le nombre d'atomes dans l'univers.
- La fonction B_3 croît très rapidement, mais ce n'est rien si on la compare à B_4 .
- Le taux de croissance de la fonction B_{100} dépasse l'entendement...

Lemme

Pour tout $i > 0$, B_i est calculables par un programme RÉPÉTER.

Preuve

$\texttt{B[0]}(r_1) = B_0(r_1)$

$r_0 \leftarrow \texttt{PLUS}(r_1, 1)$

$r_2 \leftarrow \texttt{PG?}(r_0, 2)$

$\texttt{si}\ r_2\ \texttt{alors}$ [

$\quad \texttt{inc}(r_0)$

]

Preuve (suite)

Pour un $i > 0$ fixé, $\texttt{B[i]}(r_1) = B_i(r_1)$

$\texttt{inc}(r_0)$

$\texttt{répéter}\ r_1\ \texttt{fois}\ [$

$\quad r_0 \leftarrow \texttt{B[i}-\texttt{1]}(r_0)$

$]$

Remarques

On remarque que le programme `B[i]` compte exactement i boucles `répéter` et la profondeur d'imbrication est aussi i.

Théorème

Propriétés de la famille des fonctions B_i

1. $B_i^{\langle k \rangle}(x)$ est croissant en i, x et k.
2. $B_i(2x) \leq B_i^{\langle 2 \rangle}(x)$ pour $i > 0$, $x \geq 0$.
3. $B_0^{\langle \lceil \frac{y}{2} \rceil + 1 \rangle}(x) \geq y + x$ pour $i, x, y \geq 0$.
4. $B_i^{\langle y \rangle}(x) \leq B_{i+1}(y + x)$ pour $i, x, y \geq 0$.

Preuve: Tous les énoncés du théorème peuvent être facilement prouvés par induction sur i.

Nombre maximal d'imbrications

Définition: Pour tout programme `RÉPÉTER`, $\mathcal{B}(\mathtt{P})$ est le **nombre maximal d'imbrications des boucles** de `P`.

Valeur maximale des registres

Définition: On note $\mathcal{M}(\mathtt{P}, r_1, \ldots, r_k)$ la **valeur maximale des registres** $r_1, \ldots, r_k$ après l'exécution de $\mathtt{P}$.

Théorème

Pour tout programme $\texttt{RÉPÉTER}$, si $\mathcal{B}(\texttt{P}) = i$, alors il existe un entier s tel que

$$\forall_{r_1,\ldots,r_k \in \mathbb{N}} \ \mathcal{M}(\texttt{P}, r_1, \ldots, r_k) \leq {B_i}^{\langle s \rangle}\Big(\max(r_1, \ldots, r_k)\Big)$$

Corollaire

Pour tout $i \geq 0$ il existe une fonction qui n'est pas calculable par un programme `RÉPÉTER` avec une profondeur de boucle i, mais qui est calculable par un programme avec une profondeur de boucle $i + 1$.

La fonction d'Ackermann

Définition: En 1926, Wilhelm Ackermann définit la fonction à deux variables suivante

$$A(i,x) = \begin{cases} 1 & \text{si } x = 0 \\ 2 & \text{si } i = 0, x = 1 \\ x + 2 & \text{si } i = 0, x > 1 \\ A(i-1, A(i, x-1)) & \text{si } i > 0, x > 0 \end{cases}$$

Lemme

$$\forall_{i \in \mathbb{N}} \, \forall_{x \in \mathbb{N}} \, A(i,x) = B_i(x)$$

Preuve. Le lemme sera prouvé par induction d'abord sur i et en suite sur x .

Théorème

La fonction d’Ackermann n’est pas calculable par un programme `RÉPÉTER`.

Donc Ackermann n’est pas primitive récursive.

Remarque

La fonction $F(x) = A(x, x)$ croît plus rapidement que n'importe quelle des fonctions $B_i(x)$.

Remarque

Un programme `RÉPÉTER` ne peut pas entrer dans une boucle infinie, son execution se termine toujours.

Les programmes `TANTQUE`

Les programmes `TANTQUE`

Les programmes `TANTQUE` sont semblables aux programmes `RÉPÉTER`

- Un nombre arbitrairement grand de registres est disponible: $r_0, r_1, \ldots$
- Chaque registre contient un entier positif ou nul
- les registres sont implicitement initialisés à 0 avant utilisation
- l'instruction $r_i \leftarrow r_j$ remplace le contenu du registre r_i par celui de r_j
- l'instruction $\mathtt{inc}(r_i)$ incrémente de 1 le registre r_i

Les instructions d'un programmes `TANTQUE`

Les programmes `TANTQUE` sont semblables aux programmes `RÉPÉTER`, mais les boucles sont differentes:

- l'instruction `tant que` $r_i \neq r_j$ `faire` $[\langle \mathrm{BLOC} \rangle]$ répète l'éxécution d'un bloc d'instructions tant que les valeurs des registres r_i et r_j diffèrent
  - l'inegalité est réévaluée à chaque itération et les valeurs de r_i et r_j peuvent changer

Les programmes TANTQUE

Un programme TANTQUE implante une fonction

$$f : \mathbb{N} \times \mathbb{N} \times \cdots \times \mathbb{N} \;\to\; \mathbb{N} \cup \{\uparrow\}$$

$$(r_1,\ r_2,\ \ldots,\ r_k) \;\mapsto\; \begin{cases} r_0 & \text{si le programme s'arrête} \\ \uparrow & \text{si le programme boucle a l'infini} \end{cases}$$

Au debut de l'exécution, les registres r_1 à r_k contiennent les arguments de f, et à la fin, r_0 contient $f(r_1, \ldots, r_k)$.