Université Bordeaux
M1 Informatique, 2016–2017

TD2 : LOOP/WHILE & Machines de Turing

Exercice 2.1 Soient $m, n \in \mathbb{N}$. On note par $\mathsf{pgcd}(m, n)$ le plus grand diviseur commun de m, n, et par $\mathsf{ppcm}(m, n)$ le plus petit multiple commun de m, n.

Ecrivez des programmes LOOP qui calculent $\mathsf{pgcd}(m, n)$ et $\mathsf{ppcm}(m, n)$.

Exercice 2.2 La racine cubique de 2 est égale à $\sqrt[3]{2} = 1,259921\ldots$. Soit digit : $\mathbb{N} \to \mathbb{N}$ la fonction $\mathrm{digit}(0) = 1, \mathrm{digit}(1) = 12, \mathrm{digit}(2) = 125, \mathrm{digit}(3) = 1259$ etc.

Ecrivez un programme WHILE qui calcule la fonction digit.

Exercice 2.3 Un programme LOOP(1) est un programme LOOP qui ne contient pas d'instructions LOOP imbriquées.
Exemples : l'addition, la sous-traction $x \dot{-} y$, l'instruction IF-THEN-ELSE peuvent être écrites sous forme de programmes LOOP(1).

Montrez que les fonctions suivantes peuvent être écrites sous forme de programme LOOP(1) :

1. $$\mathsf{test}(m, n) = \begin{cases} m & \text{si } n = 0 \\ 0 & \text{si } n > 0 \end{cases}$$
2. $\mathsf{mod}(n, k)$, où mod est le reste entier, et $k > 0$ une constante.
3. $\mathsf{div}(n, k)$, où div est la division entière, et $k > 0$ une constante.

Indication : pour les deux dernières questions, le nombre de registres dépend de k.

Exercice 2.4 Ecrivez l'instruction "IF $x = 0$ THEN P ELSE Q FI" en utilisant seulement l'addition + et la fonction test de l'exercice précédent.

Exercice 2.5 Quelle est la fonction calculée par le programme LOOP(1) suivant (en fonction de x) ?

```
y := 0; z := 0; u := 0;
LOOP(x) DO
    y := z + 2
    z := u
    u := y + 1
OD
```

Commentaire : On peut montrer que les fonctions $f : \mathbb{N}^k \to \mathbb{N}$ calculées par des programems LOOP(1) sont toutes les fonctions qu'on peut obtenir de la fonction constante 0, en utilisant l'incrémentation $s(x) = x + 1$, les projections $\pi_j(x_1, \ldots, x_n) = x_j$, l'addition $S(x, y) = x + y$, la fonction test, et la soustraction $x \dot{-} k$, la division $\mathsf{div}(x, k)$ et le reste $\mathsf{mod}(x, k)$ – où k est une constante.

Exercice 2.6 Écrire une machine de Turing qui calcule la fonction $n \longrightarrow n+1$. Vous pouvez considérer d'abord le cas où les entiers sont codés en unaire, ensuite le cas binaire.

Exercice 2.7

1. Soit $\Sigma = \{a, b, c\}$. Construire une machine de Turing acceptant le langage L des mots $u \in \Sigma^*$ ne contenant pas le facteur aa.
2. De façon générale, étant donné un langage rationnel L, comment construire une machine de Turing reconnaissant les mots de L ?
3. Même question que précédemment pour les langages hors-contexte. Il suffit de décrire la solution de façon informelle.

Exercice 2.8 Ecrire une machine de Turing (à deux bandes) qui teste si l'entrée a la forme $\underbrace{| \cdots |}_{n^2}$, pour $n \in \mathbb{N}$. Vous pouvez utiliser la formule $1 + 3 + \cdots + (2n-1) = n^2$.

Exercice 2.9 Toute machine de Turing peut être simulée par une autre machine dont la tête se déplace (à droite ou à gauche) lors de chacune de ses opérations et ne reste jamais sur la même case.

Exercice 2.10 Simuler une machine de Turing à une bande infinie dans les deux sens par une machine à une bande finie à gauche.

Exercice 2.11

1. Simuler une machine de Turing à deux bandes par une machine à une seule bande.
2. Si on définit la complexité de calcul comme le nombre de pas de calcul, quel est alors le rapport entre les complexités de la machine initiale (à deux bandes) et de la machine qui la simule ?

Exercice 2.12 Montrer que l'on peut simuler toute machine de Turing par une autre machine de Turing qui n'utilise pour son alphabet de travail que les lettres 0, 1 et $\square$.

Exercice 2.13

1. Soit Σ un alphabet, $\# \notin \Sigma$. Construire une machine de Turing qui vérifie que l'entrée a la forme $u\#u$, où u est un mot (arbitraire) sur l'alphabet Σ.
2. Construire une machine de Turing qui recopie le mot d'entrée à la fin de celle-ci.

Exercice 2.14

1. Construire une machine de Turing reconnaissant les mots de Dyck (mots bien parenthésés avec parenthèses $(,), [,]$).
2. Construire une machine de Turing reconnaissant les mots sur $\{a, b\}$ ayant le même nombre d'occurrences de a et de b.
3. Construire une machine de Turing reconnaissant $\{u \in \{a, b, c\}^* \mid |u|_a = |u|_b = |u|_c\}$.