I.U.T. de Nice Côte d'Azur
Département Informatique
DUTINFO S4 – IT

2007-2008

Informatique théorique

TD n° 7.1

Machines à registres et machine Turing

1. Objectif

1.1. Thèse de Church : « Quelques soit le *modèle raisonnable* de machine utilisé, tout ce qui est de manière *informelle effectivement calculable* peut-être *réellement calculé* par une machine de ce modèle »

1.2. Modèles informel et générique de machines de calcul : un dispositif de calcul ayant les propriétés suivantes

- chaque instant, le dispositif se trouve dans un ETAT bien précis et le nombre d'états possibles est fini.
- Le dispositif comporte d'une MEMOIRE qui peut contenir un nombre éventuellement infini D'OBJETS FINIS mais qui à chaque instant n'en contient qu'un NOMBRE FINI
- Le dispositif est muni d'un PROGRAMME qui est OBJET FINI contrôlant les changements d'états possibles et les modifications de la mémoire. A chaque instant, le programme choisit l'ensemble des ACTIONS A EFFECTUER en fonction du CONTENU DE LA MEMOIRE et de L'ETAT dans lequel se trouve la machine

1.3. Quelques définitions

- On appelle MODEL DE MACHINE toute classe de dispositifs de calcul analogues dans le sens qu'ils peuvent être décrits mathématiquement de la même façon. On appelle MACHINE tout dispositif de calcul du modèle choisi.
- Soit M une machine, on appelle CONFIGURATION INSTANTANEE de la machine un objet C qui décrit complètement à un instant donné :
  - L'état dans lequel se trouve la machine
  - Le contenu de la mémoire
  - L'ensemble des actions que le programme peut choisir de déclencher à cet instant.
- Soient C et C' deux configurations instantanées de la machine M, on dit que la machine M calcule C' à partir de C, si dans la configuration C, le programme peut déclencher des actions qui font passer la machine dans la configuration C'. On note alors : C $|M\rightarrow$ C'
- On appelle
  - CONFIGURATION INITIALE de la machine M pour l'entrée x, et noté $C_0(x)$ la configuration instantanée où
    - La mémoire ne contient que la représentation de x
    - La machine est dans l'état initial
  - CONFIGURATION TERMNALE de la machine M, toute configuration instantanée dans la laquelle la machine s'arrête (le programme de M ne peut déclencher aucune actions
- On dit qu'une machine est
  - déterministe si pour toute configuration C, le programme ne peut choisit qu'un seul ensemble d'actions à déclencher ;
  - séquentielle si pour toute configuration C, le programme ne peut déclencher au plus qu'une action. Dans le cas contraire, on dit que la machine est parallèle.
- On appelle CALCUL de la machine M sur l'entrée x, toute suite finie de configuration instantanées C_0, .. , C_n telles que $C_0 |M\rightarrow C_1 |M\rightarrow C_2 |M\rightarrow \ldots C_i |M\rightarrow C_{i+1} \ldots |M\rightarrow C_n$ où $C_0 = C_0(x)$ et Cn est une configuration terminale. On appelle y la donnée contenue dans portion de la mémoire réservé à la sortie et on dit alors que la machine M a CALCULE y partir de x et on note x M→ y la relation de $\mathbf{N}$ dans $\mathbf{N}$ « M calcule y sur l'entrée x ».

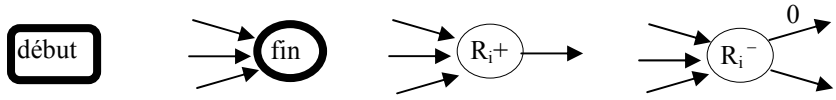
- Soient M et M' deux machines non nécessairement du même modèle. On dit que M' SIMULE la machine M si M' calcule y à partir de x si et seulement si M calcule à à partir de x. Autrement dit : $\forall x,y$ x M→ y ↔ x M'→ y
- On dit que deux modèles de machines sont équivalentes si toute machine M de l'un des deux modèle peut être simulée par une machine M' de l'autre modèle.

Dans la suite du TD on étude 2 modèles de machines équivalentes : Machine à registres et Machine de Turing.

2. Machines à registres

Les machines à registres sont introduites par Shepherdon et Sturgis en 1963. Il existe plusieurs modèles de machines à registre (MR). Nous utilisons un modèle réduit de ces machines :

- la mémoire d'une MR est composé d'un nombre virtuel infini des registres (R_1, R_2, …). Un registre peut contenir un entier naturel quelconque. Si l'entier contenu dans R_i est nul, on dit qu'il est vide. A l'instant donnée, il n'y a qu'un nombre FINI de registres NON vides.
- Les seules opérations possibles sur les registres sont l'*incrémentation* d'un registre donné et la *décrémentation* d'un registre donné si le registre n'est pas vide.
- Le programme est représenté par un graphe orienté fini G (V, E) étiqueté qui contient :
  - deux sommets particuliers qui correspondent respectivement au début et à la fin des calculs :
    - $V_{début}$ de degré entrant nul et de degré sortant 1. Ce sommet est appelé l'état initial de la machine
    - V_{fin} de degré sortant nul. Ce sommet est appelé l'état final de la machine
  - Les autres sommets sont tous étiquetés par un entier n regroupés en 2 catégories :
    - Les sommets d'*incrément* sont de degré sortant 1. Il correspond à l'incrément du registre R_n^+
    - Les sommets de d*écrément* sont de degré sortant 2. Il correspond au décrément du registre R_n^-. De tout sommet de décrément, il part une et une seule arrête étiquetée par 0.



- En entrée les données sont des entiers naturels x1, .., xn qui sont placés dans les registres R_i , … , R_n . Le résultat du calcul est, par convention, placé dans le registre R_1.
- Chaque sommet du graphe correspond à un état de la machine et le programme fait passer la machine dans l'état indiqué par l'arrête après avoir effectué l'opération associé au sommet. Dans le cas du décrément, la machine passe dans l'état indiqué par l'arrête étiquetée par 0 si et seulement si le registre est vde avant l'opération de décrément.
- On appelle $\rho(R)$ le plus grand numéro de registre figurant dna s le graphe de R.

2.1. *Somme de deux entiers* : Dessiner la MR qui calcule la somme de 2 nombres d'entier x et y. Supposons que au début x se trouve dans R_1 et y dans le registre R_2 , le résultat se trouve dans le registre R1 à la fin de l'exécution du programme. Le chargement des registres avec x et y est fait par un mécanisme extérieur à la machine.

2.2. *Remise à zéro du registre R_n* : Dessiner la MR qui calcule la remise à 0 d'un registre R_n. *Copie du registre R_n dans le registre R_m* : Dessiner la MR qui copie du contenu du registre R_n dans le registre R_m au moyen du registre R_p.

2.3. Copie du registre R_n dans les registres R_{m1}, R_{m2}, …, R_{mr}, au moyen du registre R_p.

2.4. Montrer que les fonctions suivantes sont calculables par une machine à registres

- la soustraction tronquée $\lambda xy[x \div y] = \begin{cases} x-y & \text{si } x \geq y \\ 0 & \text{sinon} \end{cases}$

- la multiplication : $\lambda xy[x \,.\, y]$

On utilise la formule de récurrence de x . y :

$$\lambda xy[x \,.\, y] = \begin{cases} 0 & \text{si } x=0 \\ (x-1)y + y & \text{sinon} \end{cases}$$

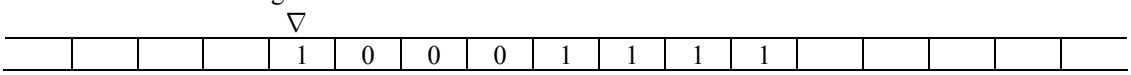
- la fonction : $f(n) = \begin{cases} 0 & \text{si n est pair} \\ 1 & \text{sinon} \end{cases}$

3. Machines de Turing

Les machines de Turing sont introduites par Alain Mathison Turing dans un article célèbre en 1936 qui est considéré comme un des articles fondateurs de l'informatique moderne. Elles servent le prototype des machines dédiées au calcul symbolique. Ce qui fait la force de cette article est d'une part qu'il propose d'un modèle de disposition de calcul capable de simuler n'importe quelle autre machine et d'autre part qu'il établit également les limites de la notion de calcul.

Une machine de Turing (MT) est un dispositif de calcul composé de trois parties :

- Une BANDE décuper en cellules servant la mémoire de la machine. Chacune de ces cellulles peut contenir un caractère pris dans l'alphabet A ou un symbole spécial, le BLANC ou noté # s'il y a ambiguïtéL La bande est infinie dans les deux sens, mains ne contient qu'un nombre fini des symboles NON BLANCS.
- Une TETE DE LECTURE qui est toujours placée sur une cellule et peut remplacer le symbole contenu dans cette cellule par un symbole de l'alphabet ou par un BLANC en se laçant éventuellement d'une cellule vers la gauche ou la droite.



Une UNITE CENTRALE qui contrôle le comportement de la machine . A tout moment, le comportement dépend de l'ETAT INTERNE de la machine.

- Ces états sont en nombre FINI et notés à l'aide d'un ensemble fini Q. Le comportement de la machine dépend alors de la TABLE DE TRANSITIONS qui est un ensemble fini de quintuplets de la forme :
  (q,a,b,D,q') où q,q' Q, a,b A et D = R,L, N
  qui signifie :
  si la machine est dans l'état q et si le symbole sous la tête de lecture-écriture est a
  - la tête de lecture-écriture écrit b dans la cellule puis
  - se déplace à droite si D=R, à gauche si D=L ou ne bouge pas si D=N
  - enfin la machine passe dans l'état q'.
- Si la machine est dans l'état q avec comme symbole lu a et qu'il n'y a pas d'instruction commençant par (q,a), la machine s'arrête de fonctionner. Un tel état est appelé un ETAT D'ARRET.
- La table des transitions est donc le PROGRAMME qu'exécute la machine.
- Deux états jouent un rôle particulier :

        - L'état INITIAL q_0 qui est l'état dans lequel la machine commence ses calculs. La présence de cet état est obligatoire
        - L'état ACCEPTANT q_{acc} qui marque un état d'arrêt de la machine correspondant à une acceptation des données initiales de la bande. Un état d'arrêt qui n'est pas acceptant est dit REJETANT. Il est parfois utile d'avoir plusieurs états acceptants pour simplifier la descripton d'une machine de Turing, mais cette facilité ne change en rien les propriétés fondamentales des MT.
- On suppose toujours qu'au début du calcul, la tête de lecture-écriture est placée sur la lettre de l'alphabet A la plus à GAUCHE de la bande. Si la bande ne contient aucune lettre de A, on considère que la bande contiel le mot vide de A*.

Définition formelle :
Une machine de Turing T est un quintuplet $(Q, A, \boxdot, \delta, q_0)$ tel que

- Q et A sont des ensembles finis disjoints et $\boxdot \notin A \cup Q$;
- δ est une fonction de $Q \times A \cup \{\boxdot\}$ dans $A \cup \{\boxdot\} \times \{R,L,N\} \times Q$

Pour simplifier,on note $Å = A \cup \{\boxdot\}$. Dans le cas, où la machine utilise un état acceptant q_{acc}, on impose que cet état soit un état d'arrêt, c'est-à-dire $\delta(q_{acc},a)$ n'est pas défini quelque soit $a \in Å$.

3.1. Réalisation d'une MT qui calcule la fonction λx[2x] en utilisant le codage unaire : A contient une seule lettre non vide 1 : n = 1^n = 111…1 (n fois de 1).

Exemple : Etat de l'entrée d'une machine de Turing

				∇												
				1	1	1	1									

Etat final :

				∇												
				1	1	1	1	1	1	1	1					

3.2. Réalisation d'une MT qui calcule la fonction λx[2x] en utilisant le codage binaire : A contient deux lettres non vide 0 et 1. Quelle est votre observation en comparaison avec le résultat de l'exercice 3.1 ?

Si l'état au début de l'exécution est :

				∇												
				1	0	1	0	1								

A la fin de l'exécution la machine serta à l'état :

				∇												
				1	0	1	0	1	0							

3.3. Réalisation d'une MT qui calcule la fonction λxy[x+y] en utilisant le codage unaire ?

3.4. Réalisation d'une MT qui calcule la fonction λxy[x+y] en utilisant le codage binaire ?

A = {0,1}
Si l'état au début de l'exécution est :

				∇												
				1	0	1	1		1	1	0	1				

A la fin de l'exécution la machine serta à l'état :

				∇												
					1	1	0	0	0							

On peut distinguer 3 cas possibles dans l'addition binaire :
- 0+0 sans retenu, 0+1 sans retenu,
- 0+0 avec retenu et 1+1 avec retenu
- 1+1 avec retenu.