TD 4: Machines de Turing et compilation

{lionel.rieg,~~paolo.tranquilli~~}@ens-lyon.fr
guest star : marc.lasson

✎ Le *résultat* d'une machine de Turing est le mot inscrit sur son (premier) ruban lorsqu'elle entre dans un état acceptant. Pour faciliter la composition des machines de Turing, on impose que lorsque la machine donne son résultat, sa tête (ses têtes) est sur la première case, et les éventuels autres rubans sont vides.

Exercice 1. *Échauffement*

Construisez les machines de Turing suivantes :

1. M_0 à un ruban sur l'alphabet $\Sigma \cup \{␣\}$ qui efface le mot sur le ruban ;
2. M_1 à un ruban sur l'alphabet $\{0, 1, ␣\}$ qui multiplie par 2 son entrée binaire ;
3. M_2 à un ruban sur l'alphabet $\{0, 1, ␣\}$ qui multiplie par 2 et ajoute 1 à son entrée binaire ;
4. M_3 à un ruban sur l'alphabet $\{0, 1, ␣\}$ qui ajoute 1 à son entrée binaire ;
5. M_4 à un ruban sur l'alphabet $\{0, 1, ␣\}$ qui soustrait 1 à son entrée binaire.

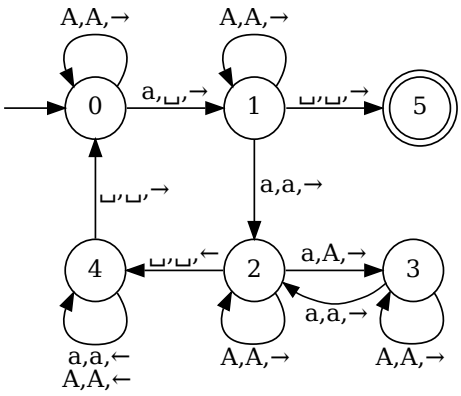
Exercice 2. *Arithmétique binaire*

Construire une machine de Turing qui calcule :

1. la composition de N_1 et N_2, deux machines de Turing fixés.
2. l'addition de deux entiers binaires, séparés par un caractère spécial # ;
3. le miroir du mot sur le ruban ;

Exercice 3. *Correction d'une machine de Turing*

1. Déterminer le langage reconnu par la machine de Turing suivante sur l'alphabet $\{a, A, ␣\}$, avec $\Sigma = \{a\}$.
(indication : faire des exemples)



2. Le prouver en montrant des invariants sur les états de la machine.

Exercice 4. *Compilation vers les machines de Turing*

On va travailler avec un petit langage de programmation impératif qu'on appellera MinImp. Dans MinImp, il y a deux types d'identificateurs (qui peuvent contenir des entiers naturels), les *paramètres* (notés $a, b, \dots$) et les *variables* (notées $x, y, \dots$). On se donne également les constantes entières. On appellera *expressions* $e, f, \dots$ les trois entités syntaxiques données ci-dessus. La grammaire des commandes $C, D, \dots$ de MinImp est donné par :

$$
\begin{array}{rll}
C, D ::= & C; D & \text{(composition séquentielle)} \\
\mid & x \leftarrow e & \text{(affectation)} \\
\mid & x\texttt{++} & \text{(incrément)} \\
\mid & x\texttt{--} & \text{(decrément)} \\
\mid & \texttt{if}\, x\, \texttt{then}\, C\, \texttt{else}\, D\, \texttt{fi} & \text{(conditionel)} \\
\mid & \texttt{while}\, x\, \texttt{do}\, C\, \texttt{done} & \text{(boucle)}
\end{array}
$$

Un programme MinImp P est donné par $Nom(a_1, \dots, a_n) : C; \texttt{return}\, e$, où Nom est un identificateur unique au programme (qu'on utilisera plus tard) et C ne fait référence qu'aux paramètres dans $a_1, \dots, a_n$.

1. Écrire un programme $\texttt{Add}(a, b)$ qui renvoie la somme de a et b.

On se propose maintenant de *compiler* MinImp dans les machines de Turing à plusieurs rubans, c.-à-d. on veut une fonction $(\,.\,)^\circ$ telle que si $P(a_1, \dots, a_n)$ est un programme MinImp, P° est une machine de Turing qui calcule la même chose que P en utilisant comme arguments le contenu de ses n premiers rubans.

2. Étant donné une machine M à k rubans, décrire une machine M' à $n \geq k$ rubans qui fait exactement ce que fait M, mais sur les rubans $i_1, \dots, i_k \leq n$ dans cet ordre, où les indices i_j sont deux à deux distincts.

3. De combien de rubans a-t-on besoin pour traduire un programme P de sorte que chaque ruban contienne tout au long de l'exécution exactement le codage en binaire d'un entier naturel (et sans utiliser des codages $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$) ?

4. Donner la traduction compositionnelle des commandes de MinImp, et puis celle d'un programme.

On étend maintenant la grammaire des expressions avec $P(x_1, \dots, x_n)$, où P est un programme à n paramètres précédemment défini. On suppose que les variables de P lui sont privées.

5. Adapter la traduction $(\,.\,)^\circ$ à ces appels de fonctions. Combien de rubans faut-il pour un programme P donné ?