Corrigé TD 1
Calculabilité

Luc Lapointe
luc.lapointe@ens-paris-saclay.fr
home.lmf.cnrs.fr/LucLapointe/

Si vous trouvez des erreurs dans les solutions, ou des preuves plus élégantes, n'hésitez pas à me les signaler.

Exercice 1 - Machines de Turing

1. Définir une machine de Turing reconnaissant le langage des palindromes.
2. Définir une machine de Turing calculant $n+1$, pour n donné en binaire avec le bit de poids fort à gauche. n peut contenir des 0 non significatifs.
3. Donner explicitement la table d'une machine de Turing qui, étant donné un mot $w \in \{0,1\}^*$, accepte w si w contient au moins autant de 0 que de 1 et rejette sinon. On prendra soin de démontrer que la machine fait bien ce qu'elle est censée faire.

Solution détaillée

Dans toutes les machines illustrées, les transitions non présentes sont des transitions qui rejettent.

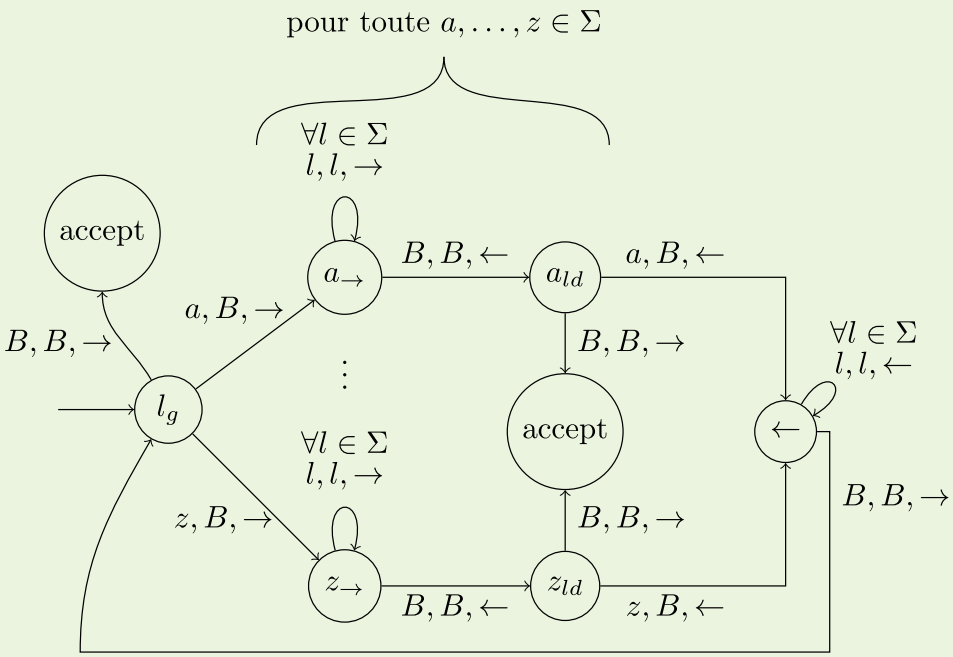

Fig. 1. – Exemple de machine acceptant les palindromes sur l'alphabet $\Sigma = \{a, ..., z\}$. La machine ne dispose que d'un unique état accept, qui est duppliqué sur le schéma pour faciliter sa lecture.

Soit m un mot dont on veut décider s'il est un palindrome avec cette machine. En début d'exécution de la machine, m est le mot qui est écrit sur la bande.

- À l'état l_g, si le curseur pointe sur un B, alors $m = \varepsilon$, qui est un palindrome, et le mot initialement sur le ruban en est aussi un. La machine accepte donc.
- Sinon, notons a la lettre pointée par le curseur. C'est la lettre la plus à gauche de m.

En la lisant, la machine l'efface et part sur une branche de l'automate dont tous les états sont nommés d'après a; une telle branche existe pour chaque lettre de l'alphabet. Tant que l'état courant est $a_{\rightarrow}$, le curseur n'est pas encore sur la lettre la plus à droite de m, et se déplace vers la droite. Une fois l'état a_{lg} atteint, le curseur pointe sur la lettre la plus à droite de m.

- Si cette « lettre » est B, alors m n'avait qu'une seule lettre, et est bien un palindrome. Si non, la suite dépend de la valeur de cette lettre :
- Si c'est un a : on l'efface, et on retourne le plus à gauche possible du mot encore écrit sur le ruban, via l'état $\leftarrow$.
- Si ce n'est ni un B ni un a, alors m n'est pas un palindrome, donc le mot qui était initialement écrit sur le ruban n'en est pas un non plus. La machine rejette.

Une fois l'état $\leftarrow$ quitté, si le mot n'est toujours pas accepté ni rejeté, on sait désormais que :

1. La première et la dernière lettre de m sont les mêmes.
2. Elles ont été effacées.
3. Soit m' le mot m privé de sa première et de sa dernière lettre. Le ruban contient m', entourés de caractères B.
4. Le curseur pointe vers la deuxième lettre de m.

Le mot m privé de sa première et de sa dernière lettre, m est un palindrome si et seulement si m' l'est. La machine réitère tous les calculs décrits avec $m := m'$.

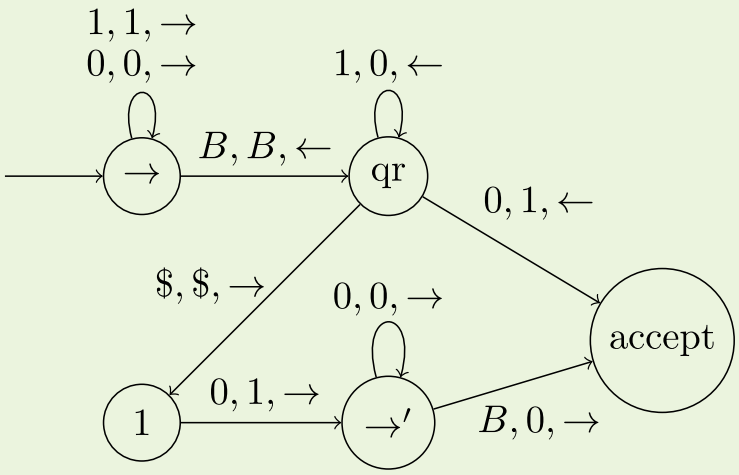

Fig. 2. – Exemple de machine incrémentant un nombre binaire avec bit de poids forts à gauche.

Soit n le nombre binaire avec bit de poids fort à gauche donné en entrée de la machine.

L'état $\rightarrow$ commence par déplacer le curseur vers le bit le plus à droite de n, sans modifier le ruban. Lorsque la machine atteint l'état qr (pour *quarry*) pour la première fois, le curseur pointe sur le bit le plus à droite de n.

L'état qr ajoute 1 au bit qu'il pointe.
- Si ce bit est 0, il est donc changé en 1, et l'addition est terminée.
- Si ce bit est 1, il est changé en 0, et il faut propager la retenue vers le bit immédiatement à gauche.

Le caractère \$ est atteint depuis l'état qr si et seulement si n est initialement un nombre binaire consitué uniquement de 1 *et sans* 0 *significatif*, si et seulement si la machine atteint l'état 1.

Si l'état 1 est atteint, il suffit donc de remplacer le 0 le plus à gauche par un 1, et de rajouter un 0 à droite. C'est ce à quoi servent les trois transitions vers et depuis l'état $\rightarrow'$.

La machine proposée utilise comme alphabet de travail $\Sigma = \{0, 1, X\}$, le X dénotant un caractère « effacé ». La machine efface le caractère le plus à gauche pas encore effacé. Si c'est 1 (respectivement 0), elle efface le 0 (respectivement 1) le plus à gauche pas encore effacé.

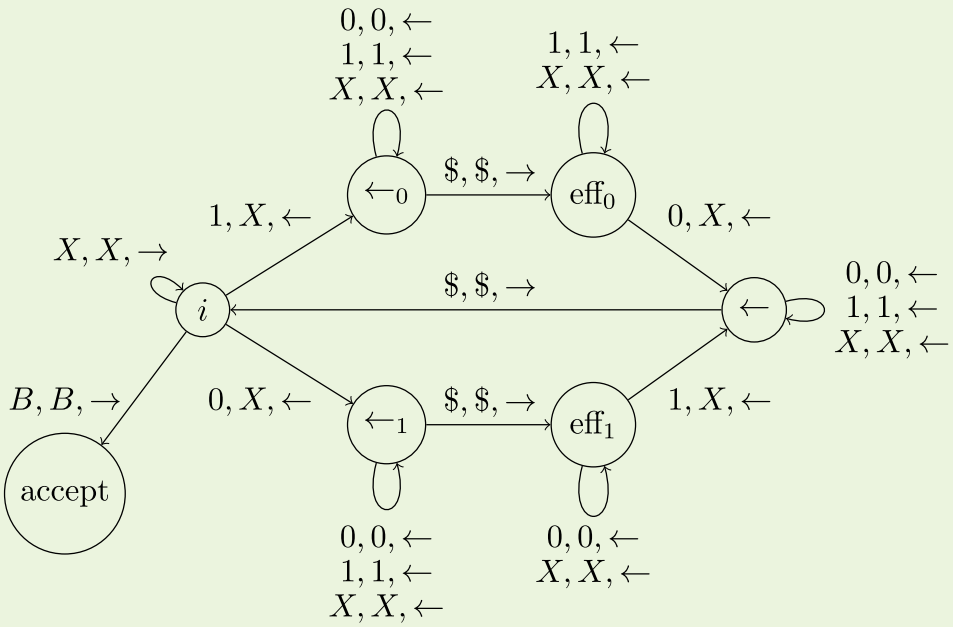

Fig. 3. – Exemple de machine acceptant les mots avec autant de 0 que de 1.

	i	$\leftarrow_0$	eff_0	$\leftarrow_1$	eff_1	$\leftarrow$
0	reject	$\leftarrow_0, 0, \leftarrow$	$\leftarrow, X, \leftarrow$	$\leftarrow_1, 0, \leftarrow$	$\text{eff}_1, 0, \leftarrow$	$\leftarrow, 0, \leftarrow$
1	reject	$\leftarrow_0, 1, \leftarrow$	$\text{eff}_0, 1, \leftarrow$	$\leftarrow_1, 1, \leftarrow$	$\leftarrow, X, \leftarrow$	$\leftarrow, 1, \leftarrow$
X	$i, X, \rightarrow$	$\leftarrow_0, X, \leftarrow$	$\text{eff}_0, X, \leftarrow$	$\leftarrow_1, X, \leftarrow$	$\text{eff}_1, X, \leftarrow$	$\leftarrow, X, \leftarrow$
B	accept	reject	reject	reject	reject	reject
\$	reject	$\text{eff}_0, \$, \rightarrow$	reject	$\text{eff}_1, \$, \rightarrow$	reject	$i, \$, \rightarrow$

Tableau 1. – La table de transitions de la machine en Fig. 3.

Soient n_0 et n_1 le nombre de 0 et de 1 dans le mot sur le ruban. Soit m le mot initialement sur le ruban. La machine illustrée, lorsqu'elle arrive dans l'état q, maintient les invariants illustrés en Tableau 2.

Pour montrer que ces invariants sont toujours vrais, il suffit de constater qu'ils sont vrais avant l'exécution de la machine, et que chaque transition les maintient.

état courant	invariant
$i, \leftarrow$	$n_0 - n_1 = \lvert m\rvert_0 - \lvert m\rvert_1$
$\leftarrow_0, \text{eff}_0$	$n_0 - n_1 - 1 = \lvert m\rvert_0 - \lvert m\rvert_1$
$\leftarrow_1, \text{eff}_1$	$n_0 - n_1 + 1 = \lvert m\rvert_0 - \lvert m\rvert_1$

Tableau 2. – Invariants maintenus par la machine selon l'état courant.

La preuve se repose ensuite sur les constats suivants :

1. i est le premier état visité, puis ne peut être atteint que lorsque le curseur pointait vers \$ dans la configuration précédente. On en déduit que lorsque i

est atteint depuis un autre état, le curseur pointe vers la première case du ruban. C'est également le cas à l'état initial.
2. La machine ne peut donc accepter que si elle atteint l'état i alors que le ruban ne comporte plus ni 1 ni 0.
3. Le constat 2. combiné aux invariants du Tableau 2 permettent d'affirmer qu'**un mot qui n'a pas autant de** 0 **que de** 1 **est nécessairement refusé par la machine**.
4. S'il y a au moins un 1 et un 0 sur le ruban, alors la machine peut sortir de l'état i puis y revenir plus tard, en remplaçant un 1 et un 0 par un X.
5. Le constat 4. combiné aux invariants du Tableau 2 permettent d'affirmer qu'**un mot qui a autant de** 0 **que de** 1 **est nécessairement accepté par la machine**.

Exercice 2 - Questions existentielles

Parmi les trois fonctions suivantes, deux sont calculables, et le problème de la calculabilité de la troisième est un (célèbre) problème ouvert. Lesquelles ?

$$f_1(n) = \begin{cases} 1 \text{ si P} = \text{NP} \\ 0 \text{ sinon.} \end{cases}$$

$$f_2(n) = \begin{cases} 1 \text{ si les décimales de } \pi \text{ contiennent la sous-séquence } 1^n \\ 0 \text{ sinon.} \end{cases}$$

$$f_3(n) = \begin{cases} 1 \text{ si les décimales de } \pi \text{ contiennent} \\ \quad \text{une sous-séquence maximale de 1 de longueur } n \\ 0 \text{ sinon.} \end{cases}$$

Solution

$f_1(n)$ est la fonction constante égale à 0, ou la fonction constante égale à 1. C'est dans les deux cas une fonction calculable !

$f_2(n)$ est soit la fonction constante égale à 1, soit une fonction en palier qui vaut 1 jusqu'à une certaine valeur, puis 0 ensuite. Dans les deux cas, c'est une fonction calculable.

C'est la calculabilité de $f_3(n)$ qui est un problème ouvert !
Ce n'est pas une connaissance attendue, mais en particulier, en septembre 2024, on ne sait pas si π est un nombre univers. S'il l'est, $f_3(n)$ est la fonction constante égale à 1, évidemment calculable.

Exercice 3 - Castors affairés

Pour $n \in \mathbb{N}$, soit $\mathcal{E}_n$ l'ensemble des machines de Turing à ruban bi-infini, sur l'alphabet $\{1, B\}$, à n états (sans compter **accept** ni **reject**), et qui acceptent le mot vide.

Si $M \in \mathcal{E}_n$, on note $f(M)$ le nombre de 1 inscrits sur le ruban quand, sur la donnée ε, M s'arrête en acceptant. On considère la fonction

$$\textsc{Score}(n) = \max\{f(M) \mid M \in \mathcal{E}_n\}.$$

1. Calculer $\textsc{Score}(2)$.
2. Montrer que $\textsc{Score}$ n'est pas calculable.
3. Montrer que $\textsc{Score}(3) \geq 6$.[1]

Solution

Question 1 Il faut énumérer toutes les machines à deux états qui conviennent aux hypothèses, non seulement pour trouver une machine qui atteint la valeur maximale, mais aussi montrer qu'il n'y en a pas d'autres.

On trouve $\textsc{Score}(2) = 4$. En septembre 2024, un exemple d'une machine atteignant cette valeur est disponible sur Wikipédia, sur la page francophone « Castor affairé », section Exemples > 2 états.

Question 2 Supposons que $\textsc{Score}$ soit calculable. Il existe donc une machine $M_{\textsc{Score}}$ qui, à partir de n écrit en unaire sur le ruban, y écrit $\textsc{Score}(n)$ en unaire. Soit S son nombre d'états. L'objectif est de définir, en utilisant $M_{\textsc{Score}}$, une machine ayant un certain nombre d'états K qui permet de contredire la borne $\textsc{Score}(K)$.

Cette nouvelle machine est une composée de plusieurs machines :

- Soit $n \in \mathbb{N}$. Il existe une machine ayant au plus $n + 2$ états[2] qui, à partir d'un ruban vide, écrit n en binaire sur le ruban. On note cette machine M_n.
- Il existe une machine qui double le nombre écrit en binaire inscrit sur le ruban : il suffit de rajouter un 0. On note M_d cette machine, et D son nombre d'états.
- Il existe une machine qui, à partir d'un ruban sur lequel est écrit un entier k en unaire, réécrit le ruban en k caractères 1. Par exemple, en décrémentant petit à petit k, en rajoutant à la droite de k un 1 après chaque décrémentation, puis en redéplaçant la suite de 1 vers le début du ruban lorsque k vaut 0. On note M_u cette machine et U son nombre d'états.

À l'aide de ces machines, on peut définir pour tout $n \in \mathbb{N}$ la machine

$$M_{\textsc{Score}} \circ M_u \circ M_d \circ M_n.$$

[1] En fait, on a égalité.

Cette machine, sur l'entrée vide, commence par écrire n en binaire sur le ruban, le double, le réécrit en unaire, puis renvoie un ruban avec $\textsc{Score}(2n)$ caractères 1 dessus.

Cette machine a au plus $S + U + D + n$ états. On en déduit donc, par définition de $\textsc{Score}$, que

$$\forall n \in \mathbb{N}, \quad \textsc{Score}(S + U + D + n) \geq \textsc{Score}(2n).$$

Par ailleurs, la fonction $\textsc{Score}$ est strictement croissante. On déduit donc, à partir de l'inégalité ci-dessus, que

$$\forall n \in \mathbb{N}, \quad S + U + D + n \geq 2n.$$

S, U et D étant des constantes, on obtient ici notre contradiction.

Question 3 En septembre 2024, un exemple d'une machine atteignant cette valeur est disponible sur Wikipédia, sur la page francophone « Castor affairé », section Exemples > 3 états.

Exercice 4 - Deux états

1. Montrer que toute fonction calculable est calculable par une machine de Turing sur l'alphabet $\{0, 1, B, \$\}$.
2. Montrer que toute fonction calculable est calculable par une machine de Turing à deux états (sans compter **accept** et **reject**).
3. Peut-on réaliser les deux simplifications à la fois ?

[2] Cette majoration est très large. Une majoration logarithmique serait plus fine, mais n'est ici pas nécessaire.

Idée de solution

Question 1 Soit $\Sigma = \{a, ..., z, B, \$\}$ un alphabet. Si Σ possède moins de 2^n caractères, il est possible d'encoder avec n bits les lettres de Σ.

Soit ensuite M une machine sur Σ. Il est possible de définir M', qui essentiellement imite M mais dispose de plusieurs états en plus pour lire et écrire en binaire.

Question 2 *Inspiré des notes de cours de Hubert Comon, enrichi d'explications.*

Soit M une machine de Turing sur l'alphabet Σ et avec des états $Q = \{q_1, ..., q_n\}$. On définit une machine M' qui a pour états Q_0, Q_1. À une configuration

$$a_1 ... a_n q_i a_{n+1} ... a_m$$

de M correspond une configuration

$$(q_O, a_1, \triangleleft)...(q_0, a_n, \triangleleft)(q_i, a_{n+1}, \alpha)(q_0, a_{n+2}, \triangleright)...$$

où α est un caractère insignifiant parmi $\{\triangleleft_d, \triangleright_d, \triangleleft_i, \triangleright_i\}$.

La direction d'un triangle indique de quel côté du curseur on se situe. Le caractère i (respectivement d) indique « Il faut incrémenter l'état courant » (respectivement « Il faut décrémenter l'état courant »).

Soit par exemple une transition $(q_i, a) \mapsto (q_j, a', \rightarrow)$. La principale difficulté est de retranscrire l'état q_j à la position à droite du curseur. Pour ce faire, on peut réaliser un va-et-vient d'incrémentations et décrémentations de la manière suivante :

$$\begin{aligned}
Q_0, (q_i, a, \alpha) &\mapsto Q_1, (q_{j-1}, a', \triangleleft_d), \rightarrow \\
Q_1, (q_k, b, \triangleright) &\mapsto Q_1, (q_{k+1}, b, \triangleright_i), \leftarrow \\
Q_1, (q_k, b, \triangleright_i) &\mapsto Q_1, (q_{k+1}, b, \triangleright_i), \leftarrow \\
Q_1, (q_k, b, \triangleright_d) &\mapsto Q_1, (q_{k-1}, b, \triangleleft_d), \rightarrow \quad \text{Si } k \geq 1 \\
Q_1, (q_0, b, \triangleleft_d) &\mapsto Q_0, (q_0, b, \triangleleft), \rightarrow
\end{aligned}$$

Une transition de la forme $(q_i, a) \mapsto (q_j, a', \leftarrow)$ est décrite par des transitions symétriques, où le sens de toutes les flèches est inversé.

Il faut également une phase d'initialisation qui remplace tous les a sur le ruban de M' par des $(q_0, a, \triangleright)$. Les blancs de M, lorsque rencontrés, sont à remplacer par des $(q_0, B, \triangleright)$.

Question 3 Non. Le nombre de machines à deux états et avec l'alphabet $\{0, 1, B, \$\}$ est fini, mais le nombre de fonctions calculables est dénombrable.