Université de Versailles – Saint-Quentin-en-Yvelines 2019-2020
Licence 2 Informatique

IN406 - Théorie des langages – Machines de Turing [1]

2019-2020

Guillaume Scerri – Guillaume.Scerri@uvsq.fr

Table des matières

1. Tous mes remerciements à Alicia Filipiak pour les illustrations.

1 Introduction

1.1 Quelques rappels

On a vu dans les parties précédentes du cours deux notions de reconnaissance de langages. La reconnaissance par automate, qui correspond à la classe des langages réguliers (ou rationnels). C'est la notion de reconnaissance la plus simple. Intuitivement on n'autorise ici qu'une mémoire finie pour la reconnaissance en codant ce dont on veut se rappeler avec des états (en nombre fini). Le lemme de l'étoile donne un certain nombre de langages qui ne sont pas réguliers, par exemple $\{a^n b^n | n \in \mathbb{N}\}$, l'ensemble des palindromes.

Une deuxième notion est la reconnaissance par automates à pile, qui correspond comme on l'a vu à l'ensemble des langages algébriques. Cette classe de langages est strictement plus grande que la classe des langages réguliers : on peut évidemment reconnaitre tous les lagages réguliers avec des automates à pile, et on peut également par exemple reconnaître les deux langages non réguliers donnés en exemple précédemment. La notion de mémoire pour la reconnaissance est plus forte que pour les automates. On autorise une pile, potentiellement infinie. La mémoire est néanmoins restreinte : on ne peut lire que le symbole sur le dessus de la pile, et empiler/dépiler des symboles. Intuitivement cette contrainte est plus complexe à exprimer que pour les automates. Un intuition imparfaite est que l'on peut se rappeler d'un nombre arbitraire de choses, mais qu'utiliser cette mémoire la consomme. En conséquences certains langages ne sont pas reconnaissables par des automate à pile : il existe un analogue du lemme de l'étoile pour le langages hors contexte [2] qui tire partie de cette structure de mémoire. Par exemple $\{a^n b^n c^n | n \in \mathbb{N}\}$ n'est pas un langage algébrique. Sans en faire de preuve formelle (qui se ferait en utilisant le lemme de la pompe des grammaires hors contexte), intuitivement, pour vérifier que le nombre de b est le même que le nombre de a consomme la mémoire de ce nombre, qui n'est donc plus disponible pour vérifier le nombre de c. De même $\{ww | w \in \Sigma^*\}$ n'est pas algébrique : intuitivement pour accéder à la mémoire de la première lettre quand on lit la deuxième copie de w, on doit dépiler et donc "oublier" le reste de notre mémoire.

On s'intéressera dans cette partie du cours à ce qu'il se passe quand on lève les restriction sur la mémoire. En particulier :

— Comment formaliser un tel modèle de calcul ?

— À quoi correspond la notion de reconnaissance pour ce modèle de calcul ? Peut on reconnaître tous les langages qui seraient reconnaissables par un programme (dans votre lanagage de programmation préféré) ?

— Existe-t-il des langages qui ne sont pas reconnaissables par ce modèle de calcul ?

1.2 Un peu d'histoire

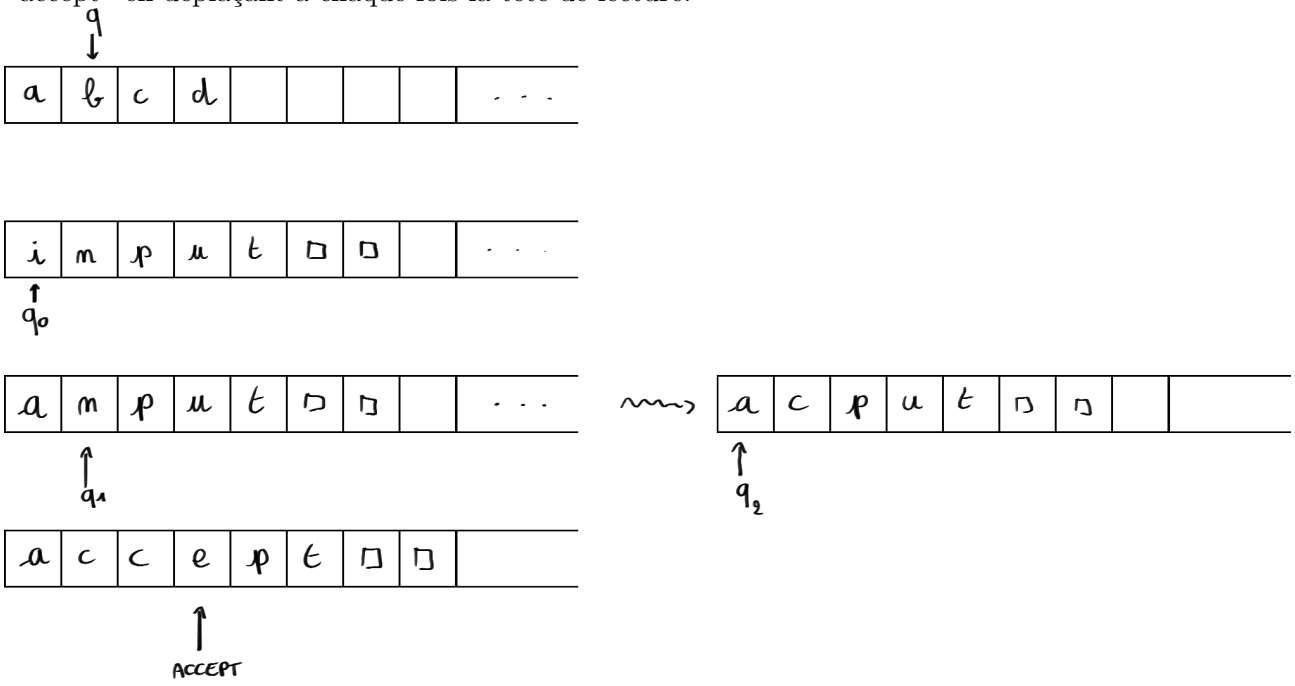
Le modèle de calcul qu'on va présenter dans ce cours, et les questions que l'on se poses ne sont pas nouvelles. Le premier à formaliser le problème fut Alan Turing en 1936 (avant même l'invention de l'ordinateur donc). Le but était de formaliser la façon dont un humain calcule (et non pas comment un humain pense contrairement à la croyance populaire) pour comprendre ce qui est calculable et ce qui ne l'est pas.

Le modèle conceptuel est le suivant : pour calculer un humain à un morceau de papier (qu'on supposera infiniement extensible), et la capacité d'écrire et d'effacer sur ce morceau de papier infini, en fonction de ce qui est écrit sous le stylo et d'une mémoire finie (celle du calculateur) et la capacité de savoir quand le calcul est fini. En particulier on modélisera la mémoire finie du calculateur par un état (dont un état particulier qui est accepteur) et le papier par un ruban infini ainsi qu'une position

2. https://fr.wikipedia.org/wiki/Lemme_d%27it%C3%A9ration_pour_les_langages_alg%C3%A9briques

sur ce ruban qui dénote la position du stylo/regard du calculateur. Une transition consiste à écrire un symbole sous le crayon (qu'on appelera plus tard tête de lecture) en fonction de l'état et du symbole présent sous le crayon, suivi éventuellement d'un déplacement du crayon. On appelle une machine formalisée dans ce modèle une machine de Turing. On a avec ce modèle de calcul deux notions, d'une part on peut reconnaître des langages exactement comme dans le cas des automates ou grammaires (c'est à dire décider certaines proriétés de l'entrée) ; d'autre part on considère également le cas de fonctions calculables, dans ce cas le résultat est ce qui est écrit sur le ruban à la fin du calcul.

A titre d'exemple dessinons la configuration dans laquelle *abcd* est écrit sur le ruban, avec la tête de lecture positionnée au dessus du b, dans l'état q ; et quelques étapes du calcul d'une machine de Turing qui transforme l'entrée *input* en *accept*. Cette machine commence avec sa tête de lecture sur la première lettre du ruban et transforme progressivement les lettres du ruban de pour former le mot "accept" en déplaçant à chaque fois la tête de lecture.



2 Définitions

2.1 Définitions informelles

Une machine de Turing est un modèle de calcul. Elle possède une mémoire linéaire infinie (typiquement seulement vers la droite), typiquement appelé ruban ; une tête de lecture, qui se déplace sur le ruban, et qui lit/écrit sur ce dernier ; ainsi qu'un état similaire à celui d'un automate. Les transitions d'une machine de Turing sont déterminées par son état et la lettre lue sur le ruban. C'est jusque là assez similaire à un automate à pile, avec la différence notable que l'on peut lire des symboles qui ne sont pas nécessairement la tête de pile et se déplacer arbitrairement dans la pile (ou que l'on efface pas la tête de pile pour lire la lettre en dessous). Un machine de Turing possède également deux états particuliers, l'un acceptant, l'autre rejetant. Ceci n'est pas un point essentiel mais un simplification importante par rapport à l'ensemble d'états finaux des automates. Contrairement à un automate, l'entrée d'une machine de Turing est initialement écrite sur le ruban, et non consommée par les transitions. Le ruban étant inifini on n'en utilise qu'un partie finie pour écrire le mot d'entrée, il est complété par une infinité de symboles blanc (typiquement écrits □) distincts des autres symboles de l'alphabet

et que la machine sait donc reconnaître.

Si une machine de Turing atteint l'état acceptant ou rejetant on dit qu'elle s'arrête. Attention, contrairement à un automate, une machine de Turing peut calculer indéfiniement et donc ne pas s'arrêter.

Le langage reconnu par une machine de Turing est l'ensemble des mots pour lesquels elle s'arrête dans l'état acceptant. Si il existe une machine de Turing reconnaissant un langage L, on dit que L est Turing reconnaissable.

Exemple : Construisons une machine de Turing acceptant le langage $\{a^n b^n c^n \mid n \in \mathbb{N}\}$. Sur l'entrée w :

— Vérifier que $w = a^n b^n w' c^n w''$:

1. lire un a sur le ruban et en le remplacer par x,
2. déplacer la tête de lecture vers la droite jusqu'à atteindre un b et le remplacer par y (si on rencontre un blanc on rejette immédiatement),
3. déplacer la tête de lecture vers la droite jusqu'à atteindre un c et le remplacer par z (si on rencontre un blanc on rejette immédiatement),
4. déplacer la tête de lecture vers la gauche jusqu'au premier a restant, si il en existe un recommencer à 1, sinon passer à la suite.

— Vérifier que $w' = w'' = \epsilon$:

1. déplacer la tête de lecture vers la gauche jusqu'au début du ruban,
2. déplacer la tête de lecture vers la droite case par case, à chaque case si la lettre lue n'est pas x_a, x_b ou x_c et que la fin du mot n'est pas atteinte, rejetter,
3. si la fin du mot est atteinte, accepter.

À la fin de la m-ième boucle, le contenu du ruban est

$$\underbrace{x_a \dots x_a}_{m} a \dots a \underbrace{x_b \dots x_b}_{m} b \dots b \underbrace{x_c \dots x_c}_{m} c \dots c$$

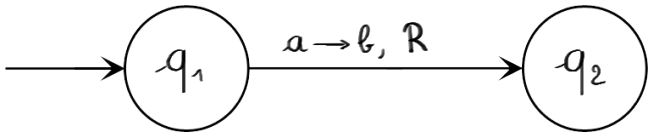
À la fin de la première étape, si le mot est de la forme $a^n b^n c^n$ le contenu du ruban est $x_a^n x_b^n x_c^n$, sinon il restera des de a, b ou c. La deuxième étape consiste simplement à vérifier qu'on est bien dans le premier cas.

Description informelle. On décrira informellement les Machines de Turing d'une façon assez similaires aux automates, avec des diagrames d'état. Supposons que q_1 est l'état initial de la machine. On dénotera comme suit la transition de l'état q_1 à l'état q_2 où l'on

— lit a sous la tête de lecture,

— écrit b à la place du a sous la tête de lecture,

— déplace la tête de lecture vers la droite (R pour droite, L pour gauche).

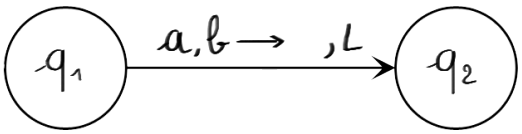


Par abus de notation on dénotera comme suit le couple de transitions où l'on

— lit a ou b sous la tête de lecture,

— ne modifie pas le symbole sous la tête de lecture,

— déplace la tête de lecture vers la gauche (R pour droite, L pour gauche).

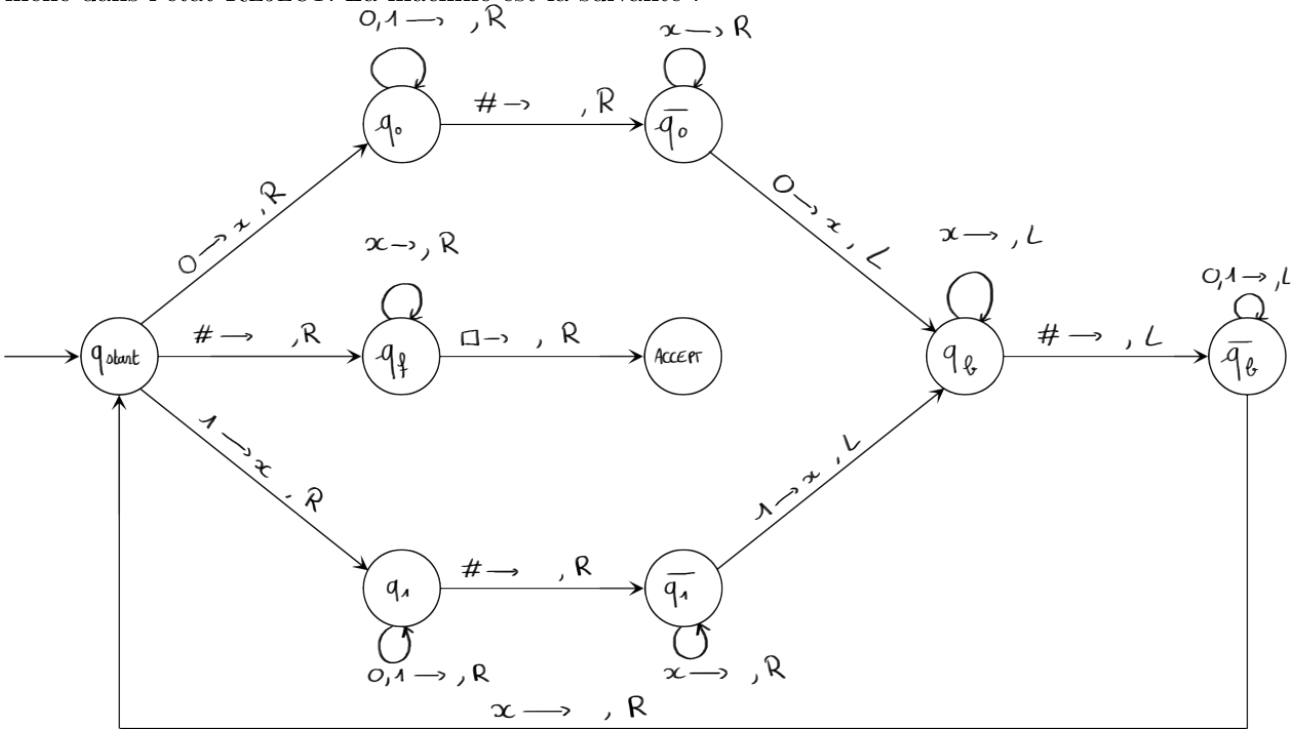


Exemple : Donnons la description informelle d'une machine de Turing reconnaissant le langage

$$\mathcal{L}_{\texttt{EQ}} = \{w\#w \mid w \in \{0,1\}^*\}\,.$$

Le symbole # est simplement un séparateur entre la première et la deuxième copie du mot. Rappelons que ce langage n'est pas algébrique (ni, par conséquent régulier).

L'idée générale de la construction de la machine est de zigzaguer entre la première et la deuxième copie du mot en utilisant l'état de la machine pour se souvenir de la lettre lue dans la première copie et de vérifier que la même lettre est bien présente dans la deuxième copie. On remplacera les lettres déjà vérifiées par des x. La tête de lecture commence sur le premier caractère du ruban (le plus à gauche). On n'écrira volontairement pas les transitions vers l'état Reject pour des raisons de clarté. Si il n'existe aucune transition écrite pour un certain couple (état,lettre sur le ruban), la transition mène dans l'état Reject. La machine est la suivante :



Présentons le calcul de la machine sur 01#01. Chaque ligne représente une configuration, avec la position de la tête de lecture dénotée par la flèche et l'état noté à côté de la flèche.

0 $\uparrow_{q_s}$	1	$\#$	0	1	$\square$	$\ldots$
x	1 $\uparrow_{q_0}$	$\#$	0	1	$\square$	$\ldots$
x	1	$\#$ $\uparrow_{q_0}$	0	1	$\square$	$\ldots$
x	1	$\#$	0 $\uparrow_{\overline{q_0}}$	1	$\square$	$\ldots$
x	1	$\#$ $\uparrow_{q_b}$	x	1	$\square$	$\ldots$
x	1 $\uparrow_{\overline{q_b}}$	$\#$	x	1	$\square$	$\ldots$
x $\uparrow_{\overline{q_b}}$	1	$\#$	x	1	$\square$	$\ldots$
x	1 $\uparrow_{q_s}$	$\#$	x	1	$\square$	$\ldots$
x	x	$\#$ $\uparrow_{q_1}$	x	1	$\square$	$\ldots$
x	x	$\#$	x $\uparrow_{\overline{q_1}}$	1	$\square$	$\ldots$

x	x	$\#$	x	1 $\uparrow_{\overline{q_1}}$	$\square$	$\ldots$
x	x	$\#$	x $\uparrow_{q_b}$	x	$\square$	$\ldots$
x	x	$\#$ $\uparrow_{q_b}$	x	x	$\square$	$\ldots$
x	x $\uparrow_{\overline{q_b}}$	$\#$	x	x	$\square$	$\ldots$
x	x	$\#$ $\uparrow_{q_s}$	x	x	$\square$	$\ldots$
x	x	$\#$	x $\uparrow_{q_f}$	x	$\square$	$\ldots$
x	x	$\#$	x	x $\uparrow_{q_f}$	$\square$	$\ldots$
x	x	$\#$	x	x	$\square$ $\uparrow_{q_f}$	$\ldots$
x	x	$\#$	x	x	$\square$	$\ldots$ $\uparrow_{\textsc{Accept}}$

On notera que la machine atteint l'état ACCEPT, le mot 01#01 est donc bien fait donc bien partie du langage de cette machine de Turing.

2.2 Définitions formelles

Commençons par définir formellement une machine de Turing.

Définition 1. Machine de Turing

Une machine de Turing (abrégé en TM dans la suite) est un septuplet $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ avec

1. Q un ensemble *fini* d'états ;
2. Σ l'alphabet d'entrée, ne contenant pas le symbole blanc $\square$;
3. Γ l'alphabet de ruban, avec $\Sigma \subset \Gamma$ et $\square \in \Gamma$;
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$ la fonction de transition, prenant en entrée un état et la lettre lue sous la tête de lecture et produisant un nouvel état, la lettre écrite sous la tête de lecture et le déplacement de la tête de lecture (R pour droite, L pour gauche, S pour rester sur place) ;
5. $q_0 \in Q$ l'état initial ;
6. $q_a \in Q$ l'état acceptant ;
7. $q_r \in Q$ l'état rejetant, avec $q_a \neq q_r$.

Exemple : Donnons la description formelle de la machine $M_{\texttt{EQ}}$ décrite plus haut :

$$M_{\texttt{EQ}} = (Q_{\texttt{EQ}}, \{0, 1, \#\}, \{0, 1, \#, x, \square\}, \delta_{\texttt{EQ}}, q_{\text{start}}, \textsc{Accept}, \textsc{Reject})$$

avec

— $Q_{\texttt{EQ}} = \{q_{\text{start}}, q_0, \overline{q_0}, q_1, \overline{q_1}, q_b, \overline{q_b}, q_f, \text{ACCEPT}, \text{REJECT}\}$. Notons que l'état REJECT n'est pas précisé dans la description informelle. On supposera toujours que les transitions non précisées dans une description informelle mènent dans l'état REJECT sans les préciser explicitement.

— $\delta_{\texttt{EQ}}$ défini par

$$
\begin{aligned}
\delta_{\texttt{EQ}}(q_{\text{start}}, 0) &= (q_0, x, R)\\
\delta_{\texttt{EQ}}(q_{\text{start}}, 1) &= (q_0, x, R)\\
\delta_{\texttt{EQ}}(q_{\text{start}}, \#) &= (q_f, \#, R)\\
\delta_{\texttt{EQ}}(q_0, 0) &= (q_0, 0, R)\\
\delta_{\texttt{EQ}}(q_0, 1) &= (q_0, 1, R)\\
\delta_{\texttt{EQ}}(q_0, \#) &= (q_0, \#, R)\\
\delta_{\texttt{EQ}}(\overline{q_0}, x) &= (\overline{q_0}, x, R)\\
\delta_{\texttt{EQ}}(\overline{q_0}, 0) &= (q_b, x, L)\\
\delta_{\texttt{EQ}}(q_1, 0) &= (q_1, 0, R)\\
\delta_{\texttt{EQ}}(q_1, 1) &= (q_1, 1, R)\\
\delta_{\texttt{EQ}}(q_1, \#) &= (q_1, \#, R)\\
\delta_{\texttt{EQ}}(\overline{q_1}, x) &= (\overline{q_1}, x, R)\\
\delta_{\texttt{EQ}}(\overline{q_1}, 1) &= (q_b, x, L)\\
\delta_{\texttt{EQ}}(q_b, x) &= (q_b, x, L)\\
\delta_{\texttt{EQ}}(q_b, \#) &= (\overline{q_b}, \#, L)\\
\delta_{\texttt{EQ}}(\overline{q_b}, 0) &= (\overline{q_b}, 0, L)\\
\delta_{\texttt{EQ}}(\overline{q_b}, 1) &= (\overline{q_b}, 1, L)\\
\delta_{\texttt{EQ}}(\overline{q_b}, x) &= (q_{\text{start}}, x, R)\\
\delta_{\texttt{EQ}}(q_f, x) &= (q_f, x, R)\\
\delta_{\texttt{EQ}}(q_f, \square) &= (\text{ACCEPT}, \square, R)\\
\delta_{\texttt{EQ}}(q, x) &= (\text{REJECT}, \square, R) \text{ dans tous les autres cas}
\end{aligned}
$$

Pour définir le calcul d'une machine de Turing nous devons maintenant définir une configuration d'une machine de Turing, avant de définir les transitions entre différentes configurations. Remarquons qu'on considère un ruban qui n'est infini que vers la droite.

Définition 2. Configuration

La configuration d'une TM est donnée par :

1. la position de la tête de lecture sur le ruban,
2. le contenu du ruban,
3. l'état de la machine de Turing.

On dénote uqv la configuration dans laquelle :

— la partie du ruban à gauche de la tête de lecture (strictement) contient le mot $u \in \Gamma^*$,

— la partie du ruban à droite de la tête de lecture (incluant la position sur laquelle est la tête de lecture) contient le mot $v \in \Gamma^*$ suivi par une infinité de symboles blancs ($\square$),

— la machine de Turing est dans l'étant q.

Pour la machine de Turing $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$, une configuration uqv est *acceptante* (resp. *rejetante*) si $q = q_a$ (resp. $q = q_r$). La *configuration initiale* de M sur l'entrée w est $\varepsilon q w$.

Exemple : Reprenons l'exemple de l'une des configurations de la machine $M_{\texttt{EQ}}$ décrite plus haut calculant sur 01#01. Si le ruban contient $xx\#x1$ et que la tête de lecture est sur le # dans l'état q_1, la configration est dénotée par $xxq_1\#x1$.

Une étape de calcul d'une machine de Turing consiste en la lecture de la lettre sous la tête de lecture, suivie par l'application des modifications du ruban (et de l'état de la machine) prescrites pas la fonction de transition.

Définition 3. Transitions

Il y a une *transition* de la configuration C_1 à la configuration C_2 dans la machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ (dénoté $C_1 \to_M C_2$) dans les deux cas suivants :

— $ua\ q\ bv \to_M u\ q'\ acv$ si $\delta(q, b) = q', c, L$,

— $u\ q\ bv \to_M uc\ q'\ v$ si $\delta(q, b) = q', c, R$.

Dans le cas où la tête de lecture est complètement à droite de la partie écrite du ruban (configuration $uq\varepsilon$), on lit des symboles blancs (i.e. $uq\varepsilon \to_M C$ si et seulement si $uq\square \to_M C$).

On dit que M accepte l'entrée w si il existe un suite de configurations $C_0, \ldots, C_k$ telle que :

— $C_0 = q_0 x$,

— pour $0 \leq l < k$, $C_l \to_M C_{l+1}$,

— C_k est acceptante.

Exemple : Reprenons la configuration de l'exemple précédent, et la même machine $M_{\texttt{EQ}}$:

$$xxq_1\#x1 \to_{M_{\texttt{EQ}}} xx\#\overline{q_1}x1$$

Définition 4. Reconnaissance

On dit qu'une machine M *reconnaît* le langage L si $x \in L \Leftrightarrow M$ accepte x. Si il existe une machine M qui reconnaît un langage L, on dit que L est *Turing reconnaissable*.

Exemple : Sur l'entrée 01#01 la machine $M_{\texttt{EQ}}$ termine dans la configuration $xx\#xx\square$ACCEPT qui est une configuration acceptante, elle accepte donc le mot 01#01. Plus généralement, elle accepte le langage $\mathcal{L}_{\texttt{EQ}}$, qui est donc Turing reconnaissable.

2.3 Machines de Turing qui calculent

Il est souvent utile de définir des machines de Turing qui calculent des fonctions (partielles), en particulier parce qu'un certain nombre de problèmes se résolvent naturellement en composant plusieurs machines de Turing qui calculent diverses fonctions afin de faire les calculs nécessaires pour reconnaître le langage. Un définition formelle de cette compositions sera donnée plus tard.

Il est important de noter que les fonctions que calculent les machines de Turing ne sont pas nécessairement définies sur toute entrée. Ce sont des fonctions partielles.

Définition 5. Fonction partielle

Une fonction partielle f de E dans F est constituée d'un ensemble de définition $D_f \subseteq E$ et d'une fonction $f : D_f \to F$. Pour tout $x \in E$ on dit que f est définie pour x si $x \in D_f$. D_f est l'ensemble de définition de f. Si f n'est pas définie pour x, on note $f(x) = \perp$.

Exemple : Considérons la fonction partielle $f_+ : \{0,1,\#\}^* \to \{0,1\}^*$ telle que pour n, m nombres écrits en binaire $f_+(m\#n) = m + n$. Son ensemble de définition est $D_f = L((0+1)^*.\#(0+1)^*)$ (f_+ n'est définie que pour les couples d'entiers). Par exemple f_+ n'est pas définie pour $01\#1\#\#$.

Définition 6. Machine de Turing qui calcule

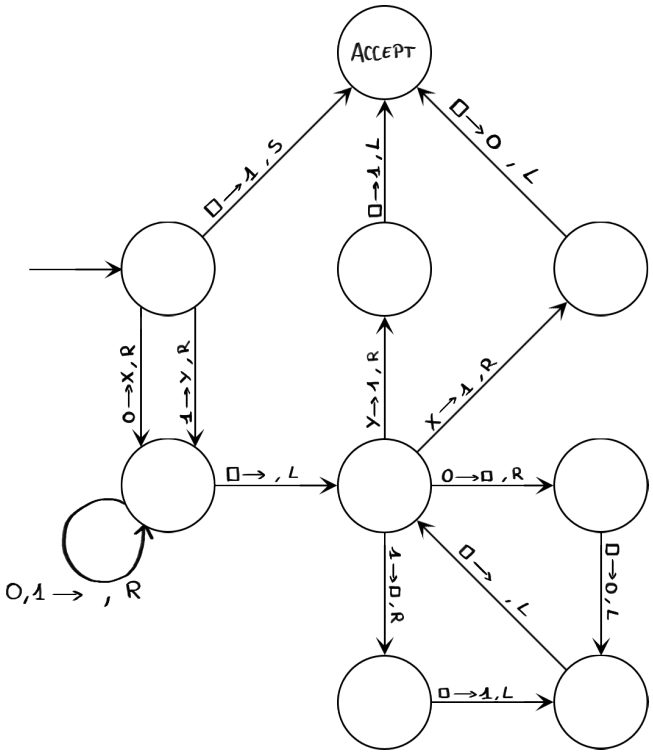
On dit que la machine de Turing M calcule la fonction partielle $f_M : \Sigma^* \to \Sigma^*$ si

1. pour tout x dans le domaine de définition de f_M, si $f_M(x) = y$ alors M accepte x avec la configuration acceptante $q_a y$,
2. pour tout x tel que $f_M(x) = \perp$, M n'accepte pas x (i.e. M ne termine aps ou M termine dans une configuration rejetante).

Exemple : Prenons comme exemple la fonction qui ajoute un \$ au début de son argument. La fonction partielle $f_\$: \{0,1,\$\}^* \to \{0,1,\$\}^*$ a pour ensemble de définition $\{0,1\}^*$ et est définie sur son ensemble de définition par $f_\$(x) = \$.x$. Une machine de Turing calculant cette fonction est la machine $M_\$$ décrite plus bas. Intuitivement, cette machine

1. marque le début du mot,
2. déplace la tête de lecture à la fin du mot,
3. déplace chacune des lettres d'une case vers la droite, de la dernière à la première lettre,
4. quand la tête de lecture arrive à la première lettre, écrit un \$, déplace la lettre d'une case vers la droite, ramène la tête de lecture à gauche et accepte.

Le seul cas particulier est le cas où l'entrée est ε, auquel cas il suffit d'écrire un \$ sur le ruban et d'accepter sans bouger la tête de lecture.



Encodages. Quand on considère des machines de Turing qui calculent, il est rare que les objets sur lesquels nous voulons calcaluer soient naturellement des chaînes de caractères sur une certain alphabet. Étant donné un objet O on notera $\langle O \rangle$ son encodage comme un chaîne de caractère sur un alphabet bien choisi. Cet encodage n'est jamais unique, il s'agit juste d'en trouver un raisonnable. Par exemple pour les entiers on choisit typiquement leur représentation en base 2, pour les graphes on pourrait choisir un représentation ligne par ligne de leur matrice d'adjacence avec chaque ligne séparée par un $\#$, ...

Par abus de notation, quand f n'opère pas sur des chaînes de caractères, on dira que M calcule f si $M(\langle O \rangle) = \langle f(O) \rangle$. Pour reprendre l'exemple de notre fonction f_+ qui calcule la somme de deux entiers soit M_+ une machine calculant cette fonction, on dira par abus de notation que M_+ calcule la fonction $f : \mathbb{N}^2 \to \mathbb{N}$ donnée par $f(x, y) = x + y$ (en encodant implicitement les entiers en base 2 et les paires d'entier séparées par un $\#$).