Références bibliographiques :

- *Introduction à la calculabilité*, Pierre WOLPER, édition Dunod (Chapitres 5 sur les machines de Turing, chapitre 7 sur l'indécidabilité, chapitre 6 sur les fonctions primitives récursives)
- *Langages formels, calculabilité et complexité*, O. Carton, éditions Vuibert

Revenons sur les machines de Turing (= MdT):

On définit une machine de Turing comme un septuplet (Q, Σ, #, Γ, δ, q_0, F) où :

- Q : un ensemble fini d'états
- Σ : l'alphabet d'entrée
- # : un symbole spécial appelé "blanc"
- Γ : l'alphabet du ruban tel que $\Sigma \cup \{\#\} \subseteq \Gamma$
- δ : $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$ les fonctions de transition *(L et R pour Left et Right : sens parcours ruban)*
- q_0 : état initial
- F : ensemble des états acceptants

Pour le fonctionnement d'une telle machine, on se doit de définit le ruban (ce qui sera lu) dans son état de départ. On définit la configuration initiale ainsi :
Le mot $w = w_1 w_2 \ldots w_k$ est placé au début du ruban, il est suivi d'une infinité de blanc et la tête de lecture est sur la première case. La machine est dans l'unique état initial q_0.
Idée du ruban à l'état initial :

w_1	w_2	w_3	…	w_k	#	#	#	#	…

↑

Remarque : De manière générale, la première case de lecture ne contient pas un blanc, sauf si le mot concerné est le mot vide ε

En ce qui concerne l'évolution de la machine de Turing :
A chaque étape/pas, elle suit l'unique transition possible à partir du la configuration dans laquelle elle se trouve (si une telle transition existe). On travaille en effet ici sur des machines de Turing déterministes (c'est l'unicité qui nous le garantit).

Le mot w est accepté si la machine atteint un état final (un état de F).

Remarque : Il est important de différencier "état" et "configuration". Cette dernière est données par le triplet :

- État courant (appartenant à Q)
- Le contenu (infini) du ruban
- La position de la tête de lecture sur le ruban

En particulier, ce qui nous intéresse avec ces deux derniers points (pour appliquer transition), c'est de connaître la position de la tête de lecture et la lettre courante.

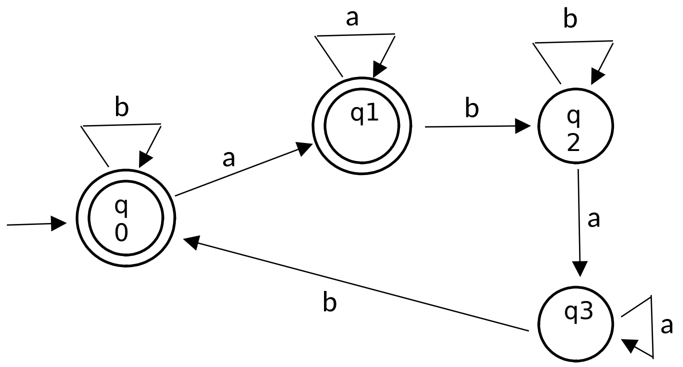
On a vu d'où la machine part, comment elle évolue et ce qui fait qu'un mot est accepté.
Que se passe-t-il alors si le mot w n'est pas accepté ? On a trois cas de figure possible :

1) La machine s'arrête parce qu'elle atteint une configuration dans laquelle aucune transition sortante n'est définie et l'état courant n'est pas un état final.
2) La machine s'arrête parce que l'unique transition à partir de la configuration courante spécifie un mouvement à gauche de la tête de lecture alors que cette tête de lecture est sur la première case du ruban (et l'état courant n'est pas final).
3) La machine passe par une suite infinie de configurations (possiblement deux à deux distinctes) sans jamais atteindre d'état final.

Remarque : en k transitions, on ne peut écrire que sur les cases numérotés entre 1 et k du ruban. Comme on est parti du mot w à une étape d'exécution particulière, seule les cases numérotées entre 1 et max(|w|, k) peuvent contenir autre chose que le blanc #.
De plus, toute case après case max(|w|, k) contient #.
C'est ce point particulier qui fait qu'à chaque étape d'exécution de la machine, il y a un nombre fini de cases qui contiennent autre chose que #.
Alors la contenu du ruban peut être décrit entièrement par un mot $c \in \Gamma^*$ fini : c'est le mot de la première case du ruban jusqu'à la case non blanche la plus à droite.

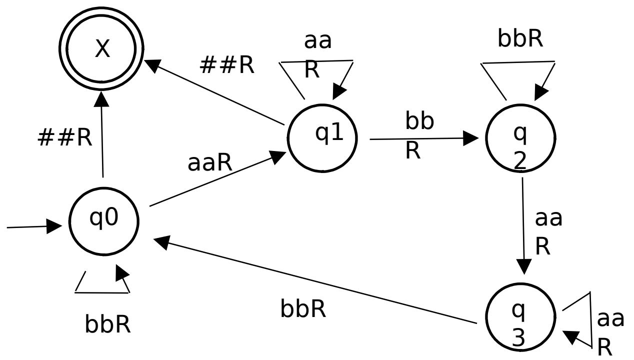
Une configuration de machine de Turing est bien descriptible par une information finie :
- Un entier entre 1 et |Q|
- Un mot fini de Γ^*
- Un entier

Exercice : Ecrire une machine de Turing qui reconnaît les mots $w \in \{a,b\}^*$ qui contiennent un nombre pair d'occurrences du motif ab
Exemple : aaabbaabbba $\in$ L
ababaababaaabaaa $\notin$ L



On peut modéliser ce problème sous la forme d'un automate

On en déduit alors facilement une machine de Turing qui suit le même fonctionnement :



Machine de Turing permettant de répondre au problème

On pose :

- q0 = j’ai vu un nombre impair de ab et j’attends un b comme prochaine occurrence
- q1 = j’ai vu un nombre impair de ab et j’attends un a comme prochaine occurrence
- q2 = j’ai vu un nombre pair de ab et j’attends un b comme prochaine occurrence
- q3 = j’ai vu un nombre pair de ab et j’attends un a comme prochaine occurrence
- X = j’ai vu un nombre pair de ab et j’ai lu un blanc

État initial : q0
Etat final : X

On peut dresser la table de transitions :

Q	**Γ**	**Q**	**Γ**	**{L, R}**
q0	b	q0	b	R
q0	a	q1	a	R
q1	b	q2	b	R
q1	a	q1	a	R
q2	b	q2	b	R
q2	a	q3	a	R
q3	b	q0	b	R
q3	a	q3	a	R
q0	#	X	b	R
q1	#	X	a	R

Remarque : au niveau des notations pour les transitions, on peut utiliser :

- $\delta(q, x) = (q', y, L \text{ or } R)$
- $(q, x, q', y, L \text{ or } R) \in \delta$

Définition : (langage reconnu par une MdT)
Le langage L reconnu/accepté par une machine de Turing M est l'ensemble des mots acceptés par M.

Proposition :
Si un langage $L \subseteq \Sigma^*$ est accepté par une machine de Turing M, alors il existe M' une machine de Turing qui accepte (exactement) L et qui remplit les deux conditions suivantes :

- Aucun état final de M' n'a de transition sortante définie (i.e. : la machine s'arrête dès lors qu'elle atteint un état final).
- Il n'existe pas de $w \in \Sigma^*$ tel que le cas de rejet (2) vu plus haut arrive (i.e. l'unique transition définie à partir de la configuration courante spécifie un mouvement à gauche de la tête de lecture alors qu'on est dans première case).

Preuve de la proposition :

- Pour satisfaire la première condition :

Il suffit de retirer de M (la machine de Turing dont il est question) toutes les transitions sortantes des états finaux.

- Pour satisfaire la deuxième condition :

Exercice : faire une machine de Turing qui, à partir de sa configuration initiale :

a) Décale le mot d'une case à droite
b) Place une ✵ sur la première case du ruban
c) Positionne sa tête de lecture sur la deuxième case du ruban
d) Se trouve dans l'état q0

La configuration initiale étant : $w \in \Sigma^*$, q'0 est l'état initial et sa tête de lecture est mise sur la première case.
Indication : on peut écrire les transitions ainsi $T^* = \{ (q1, a, q2, suiv(a), R) \mid a \in \Sigma \}$

On pose :
$T^* = \{ (q'_0, a, q_a, ✵, R),\ a \in \Sigma \}$
$T^R = \{ (q_b, a, q_a, b, R),\ a \in \Sigma,\ q_b \in Q \backslash \{q_0, q^L\} \}$
$T^\# = \{ (q_b, \#, q^L, b, L),\ q_b \in Q \backslash \{q_0, q^L\} \}$
$T^L = \{ (q^L, a, q^L, a, L),\ a \in \Sigma \} \cup \{ (q^L, ✵, q_0, ✵, R) \}$
$T^2 = \{ (q^L, ✵, q0, ✵, R) \}$

$Q_{décale} = \{ \{q_a,\ a \in \Sigma\} \cup \{ \# \} \} \cup \{ q_0 \} \cup \{ q^L \}$
$\Gamma = \Sigma \cup \{ \# \} \cup \{ ✵ \}$

Commentaires/ explications :

- T^* *permet de : “écrire ✵ dans la première case et retenir la lettre qui y était avant”*
- T^R *permet de : “remplacer la lettre courante par la lettre lue à l’étape précédente et retenir la lettre qui y était avant “*
- $T^\#$ *permet de : “trouver le premier blanc après avoir écrit l’étoile”*
- T^L *permet de : “revenir vers le début du ruban, jusqu’à ✵”*
- T^2 *permet de : “on a trouvé ✵, on se positionne sur la 2ème case”*
- q_a *= état qui retient la lettre lue (et écrasée) à l’étape d’avant*
- q^L *= état quand on est en train de revenir à gauche du ruban*

Finalement : on pose $\delta = T^* \cup T^R \cup T^\# \cup T^L \cup T^2$

En pratique :
On pose M = $(Q, \Sigma, \#, \Gamma, \delta, q_0, F)$ une machine de Turing qui ne satisfait pas le cas de rejet (2).
D’après la proposition, on peut donc transformer M en une machine M’ qui la satisfera.
Alors on aura M’ = $(Q', \Sigma, \#, \Gamma, \delta', q_0', F)$ avec :

- $Q' = Q_{\text{décale}} \cup Q \cup q_{perdu}$
- $\Gamma = \Sigma \cup \{\#\} \cup \{✵\}$
- $\delta' = T^* \cup T^R \cup T^\# \cup T^{perdu} \cup T^2 \cup \delta$ où $T^{perdu} = \{ (q, ✵, q_{perdu}, ✵, R) \mid q \in Q \}$

Théorème :
Si un langage L est reconnu par une machine de Turing, alors L est reconnu par une machine de Turing “gentille” qui satisfait les cas de rejet (1) et (2).

Dans la suite, on ne considèrera que des machines de Turing satisfaisant les cas de rejet (1) et (2).

Corollaire :
Dans ces machines, il n’y a que trois possibilités pour un mot w :

La machine s’arrête car aucune transition n’est définie	Dans un état final	W est accepté
	Dans un état non final	W n’est pas accepté
La machine ne s’arrête pas		W n’est pas accepté

Les trois façons de parler suivantes sont équivalentes :

- La machine M accepte le langage L
- Le langage L est accepté par la machine M
- L'ensemble des mots $w \in \Sigma^*$ acceptés par M est exactement L

Définition : (langage décidé)
Un langage L est décidé par une machine de Turing si :

- M accepte L
- M s’arrête pour tous les mots $w \in \Sigma^*$

La notion de “langage décidé” convient pour essayer de capturer ce qu’est une MCE.

Thèse de Church-Turing :
Un langage L peut être reconnu par une méthode de calcul effective (= MCE) si et seulement si L est décidé par une machine de Turing.

Remarque : La bonne notion est bien celle de langage “décidé” et non pas de langage “accepté”.