Institut Galilée
Année 2007-2008

Calculabilité
L3 INFO 1

TD MT

# Machines de Turing

**Exercice 1** (Quelques exemples de machines de Turing)**.**

1. *Construire une machine de Turing acceptant le langage* $\{uc\bar{u} \mid u \in \{a,b\}^*\}$.
2. *Construire une machine de Turing acceptant le langage* $\{u \in \{a,b\}^* \mid |u|_a = |u|_b\}$.
3. *construire une machine de Turing calculant* $n+1$ *pour* $n$ *donné en binaire sur le ruban d'entrée.*
4. *En utilisant ce qui précède, décrire une méthode qui permettrait de calculer* $n+m$ *en binaire (*$n$ *et* $m$ *donnés sur le ruban).*

**Correction 2.**

1. On utilise la méthode suivante :
   - lire le premier caractère $a$ ou $b$ du ruban l'effacer (avec un blanc) et conserver sa valeur dans l'état (états $q_a$ et $q_b$)
   - aller à la dernière case non vide (blanc) du ruban
   - vérifier que le caractère qui s'y trouve est bien le même et l'effacer.
   - revenir (état $q_d$) au premier caractère non blanc du ruban et recommencer tant que ce caractère est $a$ ou $b$.
   - Lorsque le premier caractère est $c$ (état $q_c$) vérifier que la case suivante est bien vide et à cette condition entrer dans l'état final.

   L'automate correspondant à la machine de Turing est donné figure 1.
2. On utilise la méthode suivante. Tant qu'on est pas dans l'état final, lire le caractère courant :
   - si c'est $a$ ou $b$ conserver sa valeur dans l'état, le remplacer par un $c$ puis parcourir le ruban vers la droite jusqu'à trouver le caractère complémentaire, le remplacer par un $c$ et revenir au début du ruban.
   - Si c'est un $c$ passer au caractère suivant à droite.
   - si c'est un blanc passer dans l'état final.

   L'automate correspondant à la machine de Turing est donné figure 2.
3. On suppose que le nombre en binaire est écrit de gauche à droite des bits de poids faibles aux bits de poids forts. Pour ajouter 1 il suffit soit de faire passer le bit de poids faible à un s'il était à zéro (ou s'il n'y en avait pas), soit de le faire passer à zéro s'il était à un et dans ce cas puisqu'il y a une retenue de un de l'ajouter au nombre binaire formé par les autres bits. La solution est dans l'automate de la figure 3. On ne remet pas la tête au début.
4. Il y a plusieurs solutions possible, en voici une qui exploite l'opération successeur vue précédemment. On pourrait tout aussi bien imiter la manière dont on pose habituellement une addition de deux nombres.

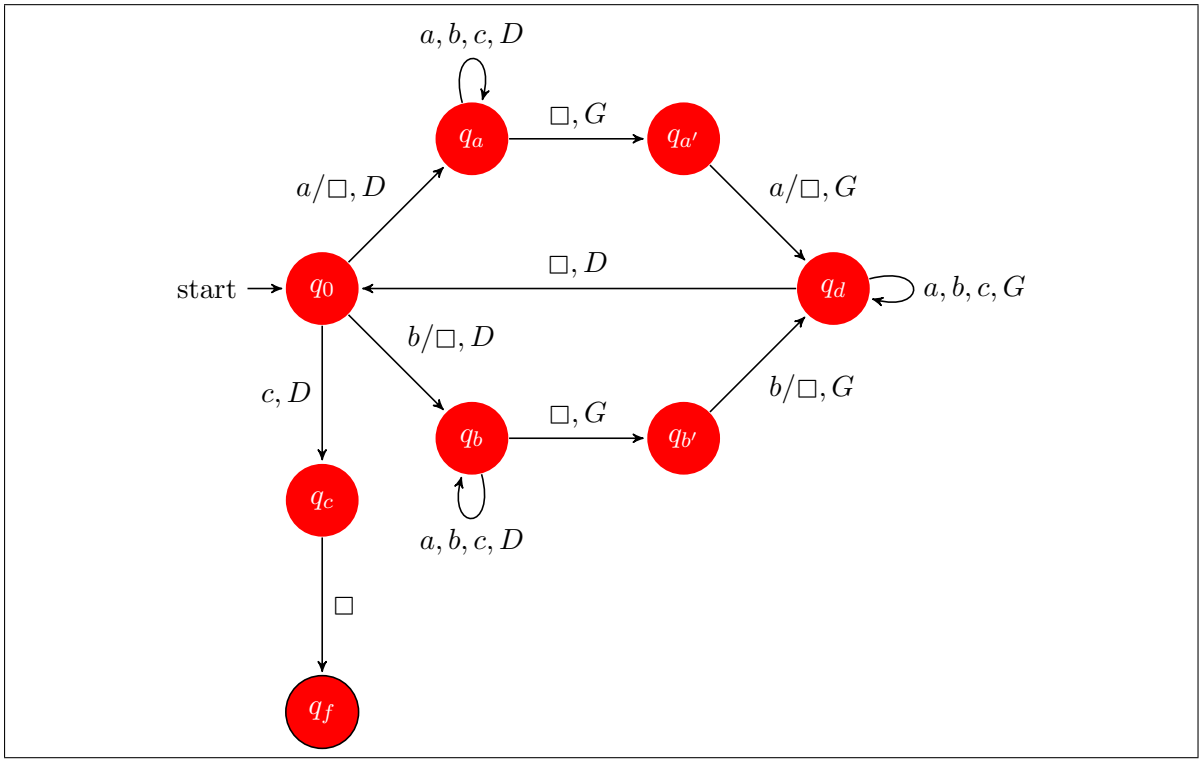


**Fig. 1:** Un mot, un séparateur ($c$), un mot miroir

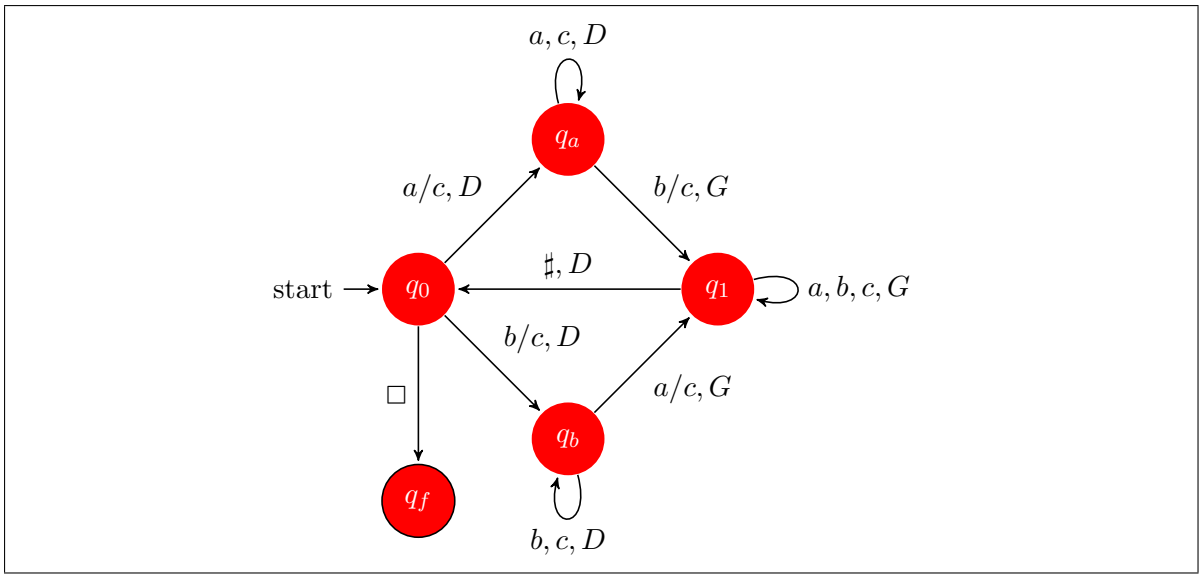


**Fig. 2:** Autant de $a$ que de $b$

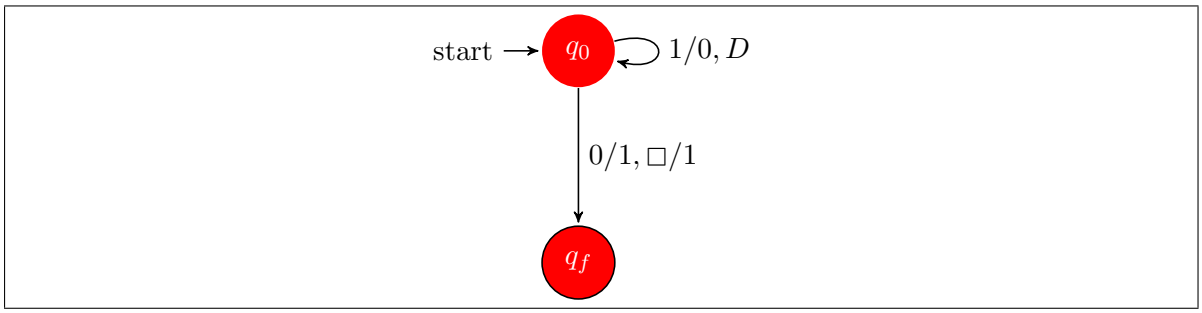


**Fig. 3:** Successeur binaire

Comme précédemment, on utilise la représentation des nombres binaires de gauche à droite par bits de poids croissants. On commence par remarquer qu'ajouter un nombre binaire dans lequel seul le dernier bit est à un à un autre nombre est une généralisation simple du cas du successeur que nous venons de traiter. Par exemple, pour ajouter $m = 00001$ à $n = 10111001$, il suffit de sauter les quatre premiers bits (1011) de $n$ – pour ces bits nous noterons $z$ pour 0, $u$ pour 1 – et de prendre le successeur du nombre binaire formé des bits restants 1001. Ainsi $10111001 + 00001$ se ré-écrit $uzuu1001 + 1$ ce qui donne $uzuu0101$, qui se ré-écrit enfin 10110101. Maintenant si $m$ vaut 1101, par décomposition de $B$ en $1 + 01 + 0001$, il suffira de faire $n + 1 + 01 + 0001$. Mais on se rend compte que cette décomposition de $m$ peut être traitée à la volée par la traduction des premiers bits de $n$ sous la forme d'un mot de $\{z, u\}^*$. Ainsi on peut effectuer $10111001 + 1101$ comme ceci (on représente explicitement les espaces libérés en *poussant* le signe +) :

```
n + m  = 10111001+1101
       = z1111001 +101
       = zz000101  +01
       = zzz00101   +1
       = zzzu0101    +
       = 00010101
```

Autre exemple, avec $n = 01011111$ et $m = 1111$.

```
n + m  = 01011111+1111
       = u1011111 +111
       = uz111111  +11
       = uzz000001  +1
       = uzz100001   +
       = 100100001
```

Il faut toutefois être attentif à la taille des représentation binaires qui peut augmenter au cours de l'addition : en plaçant le nombre à la représentation la plus longue à gauche et en réservant une place vide avant le signe plus on se garantit un espace suffisant pour réaliser toute l'opération. Car effectuer successeur ne peut qu'occuper une case de plus à chaque étape (en réalité un seul blanc en tout et pour tout est nécessaire).

On suppose que, au départ, le nombre à la représentation la plus longue est écrit au début du ruban, c'est $n$, suivi d'un blanc (le signe plus est inutile) puis du second nombre, $m$ et d'un symbole de fin ♯.(Au besoin on peut effectuer des opérations de préparation du ruban pour que celui-ci soit dans la configuration que l'on vient de décrire).

Tant qu'on n'atteint pas l'état $q_n$ ($n$ pour nettoyage) on fait en boucle les opération suivantes. On lit le premier bit $(0,1)$ de $m$. on remplace ce bit par un blanc et on conserve sa valeur $b$ dans l'état. On va au premier bit $a$ de $n$ (ce bit est 0 ou 1, les $z$ et $u$ survenant par la suite ne comptent pas). Si $b$ vaut 0 on traduit le bit $a$ en lettre : $z$ si $a = 0$, $u$ si $a = 1$ et on recommence la boucle. Si $b$ vaut un on lance l'opération successeur à partir du bit $a$ en prenant bien soin de traduire le bit qui prendra la place de $a$ en lettre (les autres bits résultants de l'opération sont écrits avec 0 et 1). Une fois que c'est fait on recommence la boucle. Si $B$ n'a plus de bits (on lit $\sharp$) on passe dans l'état nettoyage et on va à gauche.

Dans l'état nettoyage ($q_n$), on se trouve à gauche du $\sharp$ de fin. En parcourant le ruban vers la gauche on remplace $u$ par 1 et $z$ par 0 jusqu'à aller butter dans le $\sharp$ de début. (On pourrait également gérer l'effacement ou le déplacement du $\sharp$ de fin.)

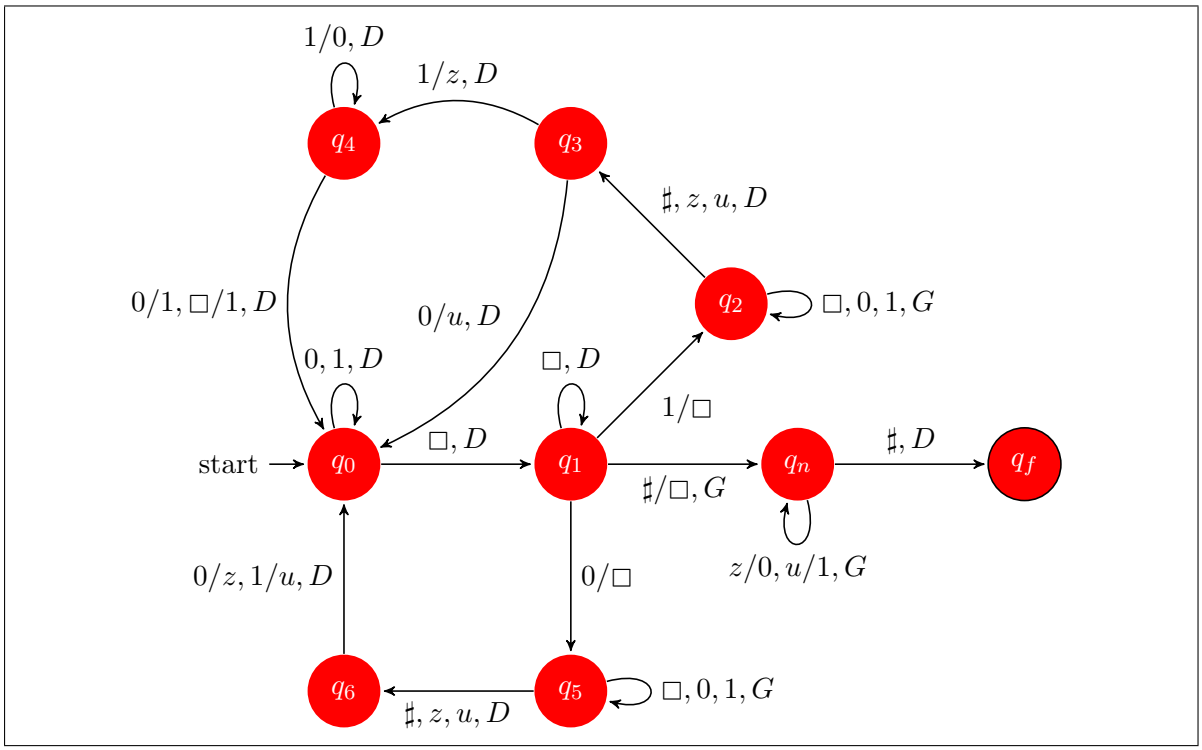


**Fig. 4:** Somme binaire