Langages formels (11X003) - Automne 2025

11. Machines de Turing non-déterministes (et autres)

Enseignant: Arnaud Casteigts

Assistants: A.-Q. Berger & M. Marseloo
Moniteurs: N. Beghdadi & E. Bussod

Dans ce cours, nous allons voir différentes notions liées aux machines de Turing, sans lien particulier. Nous mentionnons d'abord différents modèles de machines de Turing, tous équivalents. Nous discutons ensuite des machines de Turing non-déterministes. Puis, nous parlons de machines de Turing qui effectuent des traitements autres que d'accepter et rejeter des mots. Enfin, nous discutons brièvement de machines à plusieurs bandes et de machines qui potentiellement ne s'arrêtent jamais.

11.1 Différents modèles équivalents

- La tête de lecture ne peut aller qu'à gauche ou à droite (mais pas rester sur place) :

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

- Existence de plusieurs bandes (exemples plus loin)

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}$$

- Une bande infinie des deux côtés : Même fonction de transition que normalement. La seule différence est qu'on peut se déplacer à gauche du point de départ.
- Machines non-déterministes (exemples plus loin)

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R, S\})$$

C'est un fait assez surprenant que tous ces modèles s'avèrent être équivalents en termes de langages. Cela se démontre par **simulation** : chacun de ces modèles est capable d'imiter le comportement des autres. Dans certain cas, cette simulation est très "lente" (les opérations du modèle simulé requièrent de nombreuses opérations natives), mais hormis la rapidité, les capacités sont les mêmes. On dit que chacun de ces modèles est **Turing-complet**.

De toutes ces équivalences, la plus surprenante est peut-être celle entre les machines de Turing déterministes et non-déterministes. Pour simuler une machine non-déterminisme, une machine déterministe peut en effet explorer méthodiquement toutes ses exécutions possibles (nous verrons cela au second semestre).

Une autre nouveauté des machines de Turing par rapport aux autres modèles que l'on connaît est qu'elles peuvent effectuer d'autres traitements que d'accepter et rejeter des mots. Nous allons voir quelques exemples.

11.2 Machines de Turing non-déterministes

D'une certaine manière, les machines de Turing non-déterministes n'existent pas (nos ordinateurs sont déterministes). On pourrait donc être tentés d'ignorer leur étude. Mais ces machines s'avèrent centrales pour plusieurs questions en informatique fondamentale. Elles sont notamment à l'origine de la fameuse question P vs. NP, dont vous entendrez à nouveau parler. Un autre intérêt du non-déterminisme, comme pour les AFN, est de permettre parfois d'exprimer des traitements de manière plus simple.

Lors d'un branchement non-déterministe, c'est l'ensemble de la machine qui est dupliquée dans plusieurs univers : l'état, la bande et la position de la tête de lecture. Autrement dit, grâce au non-déterminisme, une *configuration* peut donner naissance à différentes configurations dans des univers parallèles. Dans ce cas, on représente les différentes exécutions possibles sous forme d'un arbre de configurations.

Comme pour les AFN et les APN, une telle machine accepte un mot si et seulement si au moins une des branches d'exécution l'accepte.

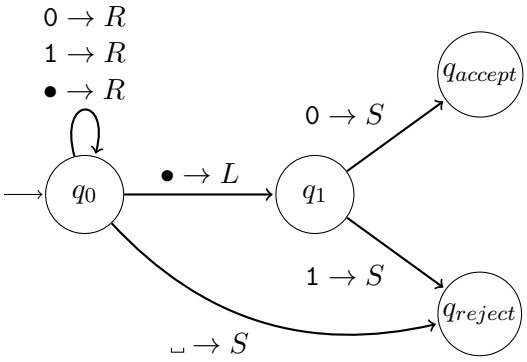
Exemple

Voyons un exemple d'une telle machine : on reçoit en entrée une liste de mots binaires séparés par des • et on veut déterminer si au moins l'un de ces mots représente un nombre pair (termine par 0). Par simplicité, on supposera qu'un point supplémentaire est présent après le dernier mot, par exemple, voici une entrée possible :

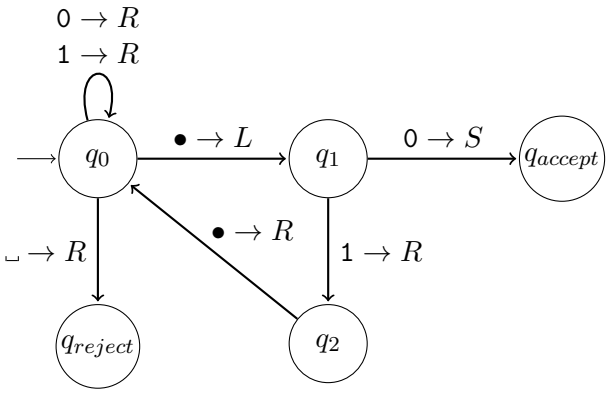
$$01101 \bullet 0110 \bullet 1010101 \bullet 100100 \bullet$$

Une machine déterministe pour ce problème examinerait chaque mot séquentiellement. Avec le non-déterminisme, on peut créer différentes branches de calcul qui examinent chacune un mot en parallèle. Plus précisément, lorsque le premier point est atteint, une branche non-déterministe va examiner ce mot, tandis qu'une autre va se diriger directement vers le point suivant (et ainsi de suite). La machine acceptera à condition qu'*au moins une* branche atteigne l'état q_{accept}.

Voici la machine correspondante :

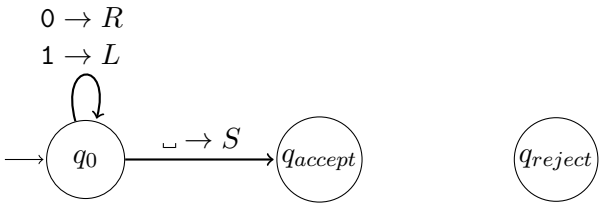


Et la version déterministe, pour comparaison :



11.3 Des machines qui ne terminent pas !

Soit la machine suivante, dont l'alphabet d'entrée est $\Sigma = \{0, 1\}$:



Que fait cette machine ? Quel langage reconnaît-elle ? Bizarrement, on découvre ici que certains mots, par exemple `01`, ne seront ni acceptés, ni rejetés, car cette machine ne termine pas toujours, elle peut boucler à l'infini sur certains mots ! C'est une nouveauté pour nous.

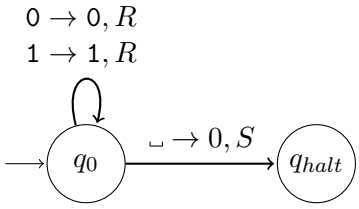
Nous reviendrons la semaine prochaine sur ce problème, qui bouscule la notion habituelle de langage reconnu par une machine. Nous distinguerons notamment deux notions : les langages Turing-reconnaissables, et les langages Turing-décidables. Affaire à suivre...

11.4 Effectuer des traitements autres qu'accepter ou rejeter

Les machines de Turing sont capables de faire d'autres choses que d'accepter ou rejeter des mots. Elles peuvent effectuer des traitements et produire des choses en sortie, comme nos ordinateurs. Dans ces cas là, on considère généralement une autre définition où les états q_{accept} et q_{reject} sont remplacés par un seul état spécial q_{halt}, qui arrête l'exécution de la machine sans accepter ni rejecter spécifiquement. On considère alors que la sortie de la machine correspond au contenu de la bande à la fin de l'exécution. Dans le modèle à plusieurs bandes, on suppose en général que le mot d'entrée est sur la première bande et qu'il faut écrire la sortie sur la dernière bande (voir l'exemple 3 ci-dessous).

Exemple 1 : Multiplier un nombre binaire par 2 **(modèle à une bande)**

Description de la stratégie : aller au bout du mot, rajouter un `0`, puis terminer.



Exemple 2 : Concaténer deux mots **(modèle à une bande)**

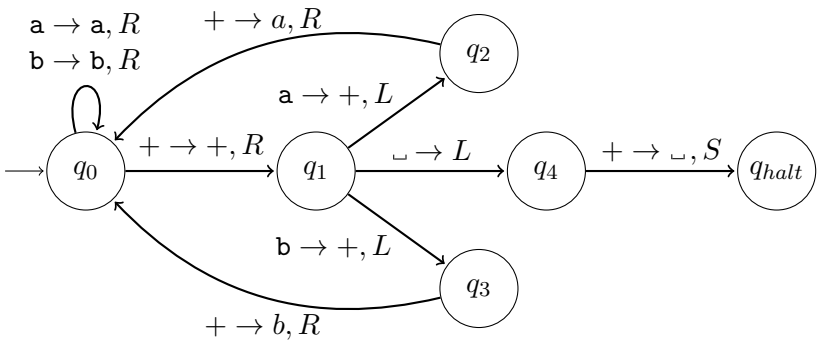
On souhaite concaténer deux mots en entrée, disons sur l'alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, +\}$, où $+$ indique où effectuer la concaténation.

Par exemple : `abba`+`bab` doit produire `abbabab`.

Difficulté : une machine de Turing ne peut pas copier plusieurs symboles d'un coup. Il faut décomposer le traitement en opérations élémentaires.

Idée générale : échanger le symbole $+$ avec le symbole à sa droite. Répéter jusqu'à avoir un espace à droite de $+$. Supprimer alors le $+$ et terminer.

Description détaillée : on commence par se déplacer jusqu'au $+$. Ensuite, on va à droite et on répète : remplacer le symbole courant par un $+$, en mémorisant ce symbole via un état de l'automate et en reculant à gauche, remplacer l'ancien $+$ par ce symbole en ré-avançant d'un cran à droite pour recommencer. Si un espace est rencontré, on efface le $+$ et on termine.



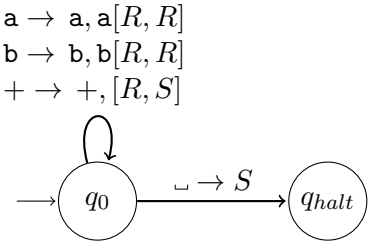
État de la bande lors des passages sur q_1 :

$$\texttt{abba+bab} \to \texttt{abbab+ab} \to \texttt{abbaba+b} \to \texttt{abbabab+}$$

Exemple 3 : Concaténer deux mots **(modèle à deux bandes)**

Rappelons que dans le cas général, pour une machine à k bandes, la fonction de transition devient $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$. On peut alors adopter la convention graphique suivante pour dessiner les transitions sur cette machine (par exemple ici pour une machine à deux bandes) : $\texttt{a}, \texttt{b} \to \texttt{c}, \texttt{d}[R, L]$, avec la signification suivante : si on lit `a` sur la première bande et `b` sur la deuxième bande, alors on écrit `c` sur la première bande, `d` sur la deuxième bande, on déplace la première tête de lecture vers la droite et la deuxième vers la gauche. Les crochets servent à séparer les déplacement du reste, pour éviter toute ambiguité. Cette notation peut être généralisée à n'importe quel nombre de bandes.

Revisitons l'exemple précédent (concaténation) en supposant que la machine a deux bandes : une d'entrée (contenant les deux mots à concaténer avec un + entre eux) et une bande de sortie, qui devra contenir le résultat de la concaténation. Le traitement à effectuer devient plus simple, car il suffit de recopier les deux mots en s'abstenant simplement de copier le séparateur. Cela donne la machine suivante :



Exemple 4 : additionner deux nombres binaires

→ En séance d'exercices. La principale difficulté sera de gérer la retenue.