Machine de Turing

Un peu d'histoire

Vous avez peut-être déjà entendu parler d'Alan Turing, à la tête de l'équipe qui a réussi à déchiffrer des messages de la machine Enigma pendant la deuxième guerre mondiale, donnant ainsi un net avantage aux alliés ce qui a nettement réduit la durée du conflit.
Tout juste avant l'arrivée des premiers ordinateurs, ce même Alan Turing a eu l'idée en 1936 d'une machine abstraite (donc pas une vraie qu'on peut toucher, mais juste une description théorique), assez rudimentaire, composée d'un état interne, d'un ruban sur lequel on peut écrire/lire des lettres et de règles (le programme) disant comment faire évoluer cette machine.
A priori rien de bien révolutionnaire, sauf qu'en fait si ! Les fondements que cette machine de Turing a contribué à mettre en place ont pu aboutir quelques années plus tard au concept d'ordinateurs comme vous les connaissez.

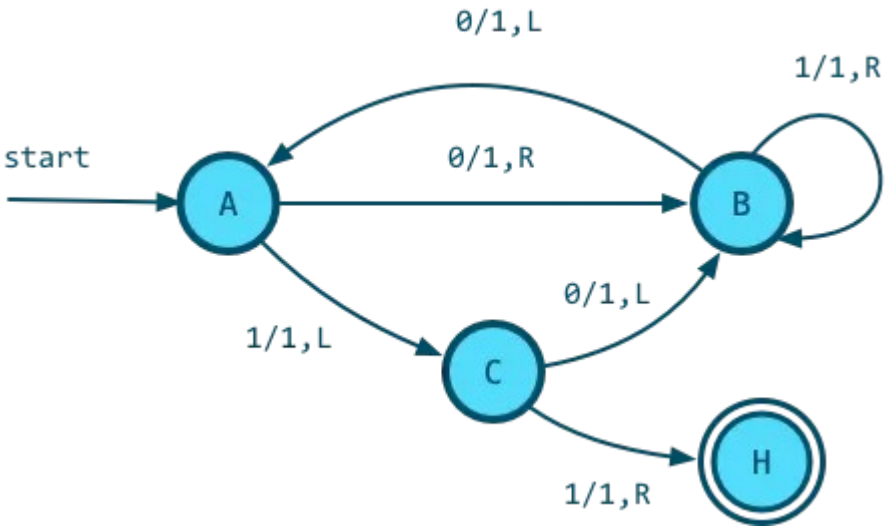
Composition de la machine de Turing

La machine de Turing est composée des éléments suivants :

- d'un ruban infini divisé en cases, dans lesquelles la machine peut écrire des symboles d'un alphabet.
- d'une tête de lecture et d'écriture
- d'une table de transition, dont chaque ligne :
  - est associée à un état
  - spécifie les actions à effectuer quand la machine est dans cet état en fonction du symbole lu par la tête de lecture
- ayant comme actions possibles :
  - Écrire un symbole (choisi dans un alphabet, 0 ou 1 souvent)
  - Se déplacer d'une case sur la gauche ou sur la droite
  - Changer d'état
- S'arrêtant quand on atteint un état désigné comme « terminal »

Exemple :

La table de transition peut être représenté par un Automate fini déterministe :



Ici, la représentation comme automate fini déterministe du castor affairé à trois états produisant le plus de 1.
L'alphabet est compris des caractères 0 et 1. On a 4 états A, B, C, H.
A est l'état initial (flèche start), H (pour HALT) un état terminal (il aurait pu y en avoir plusieurs)
Chaque cercle correspond à un état, chaque flèche à une transition. Le label de chaque flèche donne le symbole lu et le résultat ; par exemple, « 0/1,R » indique que le symbole « 0 » est lu, qu'on écrit 1 sur le ruban et qu'on se déplace à droite (right, « R »).

Définition formelle

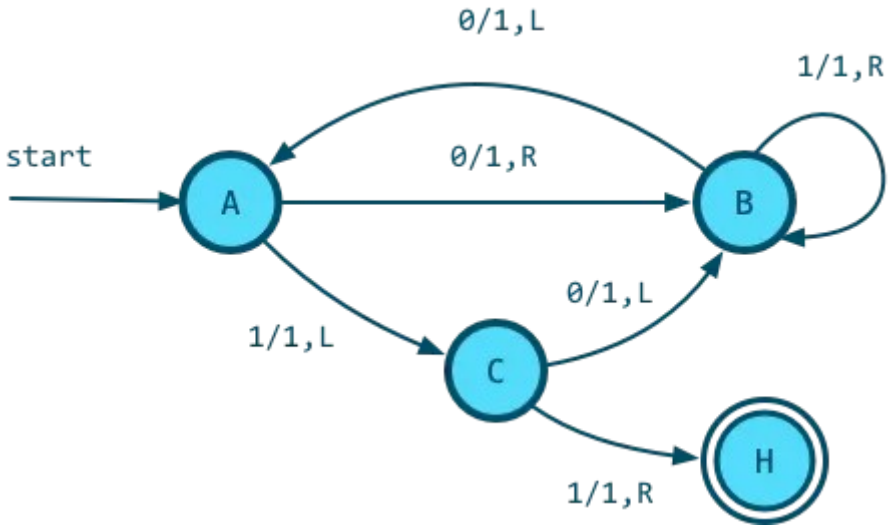
Plusieurs définitions formelles proches les unes des autres peuvent être données d'une machine de Turing. L'une d'elles[1], relativement courante, est choisie ici. Une machine de Turing est un quintuplet $(Q, \Gamma, q_0, \delta, F)$ où :

- Q est un ensemble fini d'*états* ;
- Γ est l'*alphabet de travail* des symboles de la bande, contenant B un symbole particulier (dit *blanc*), $B \in \Gamma$;
- q_0 est l'*état initial*, $q_0 \in Q$;
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ est la *fonction de transition* ;
- F est l'ensemble des *états acceptants* (ou finals[2], terminaux), $F \subseteq Q$.

Il s'agit d'un modèle de machine de Turing complète et déterministe ; i.e $\forall q \in Q, \forall \gamma \in \Gamma, \delta(q, \gamma)$ est définie et unique .

Exercice 1 : Exécuter la machine de Turing de l'exemple sur un ruban contenant uniquement des 0.

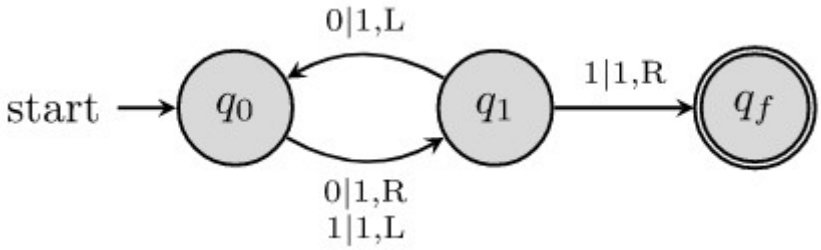
```
          A
0 0 0 0 0 0 0 0 0 0
            B
0 0 0 0 0 1 0 0 0 0
          A
0 0 0 0 0 1 1 0 0 0
        C
0 0 0 0 0 1 1 0 0 0
      B
0 0 0 0 1 1 1 0 0 0
    A
0 0 0 1 1 1 1 0 0 0
      B
0 0 1 1 1 1 1 0 0 0
            B
0 0 1 1 1 1 1 0 0 0
          A
0 0 1 1 1 1 1 1 0 0
        C
0 0 1 1 1 1 1 1 0 0
          H
0 0 1 1 1 1 1 1 0 0
```



Exercice 2 : Dessiner l'automate correspondant aux transitions suivantes puis tester la machine sur un ruban avec des 0.

Pour une machine à deux états (**A** et **B**), le castor affairé correspond à la table de transition suivante

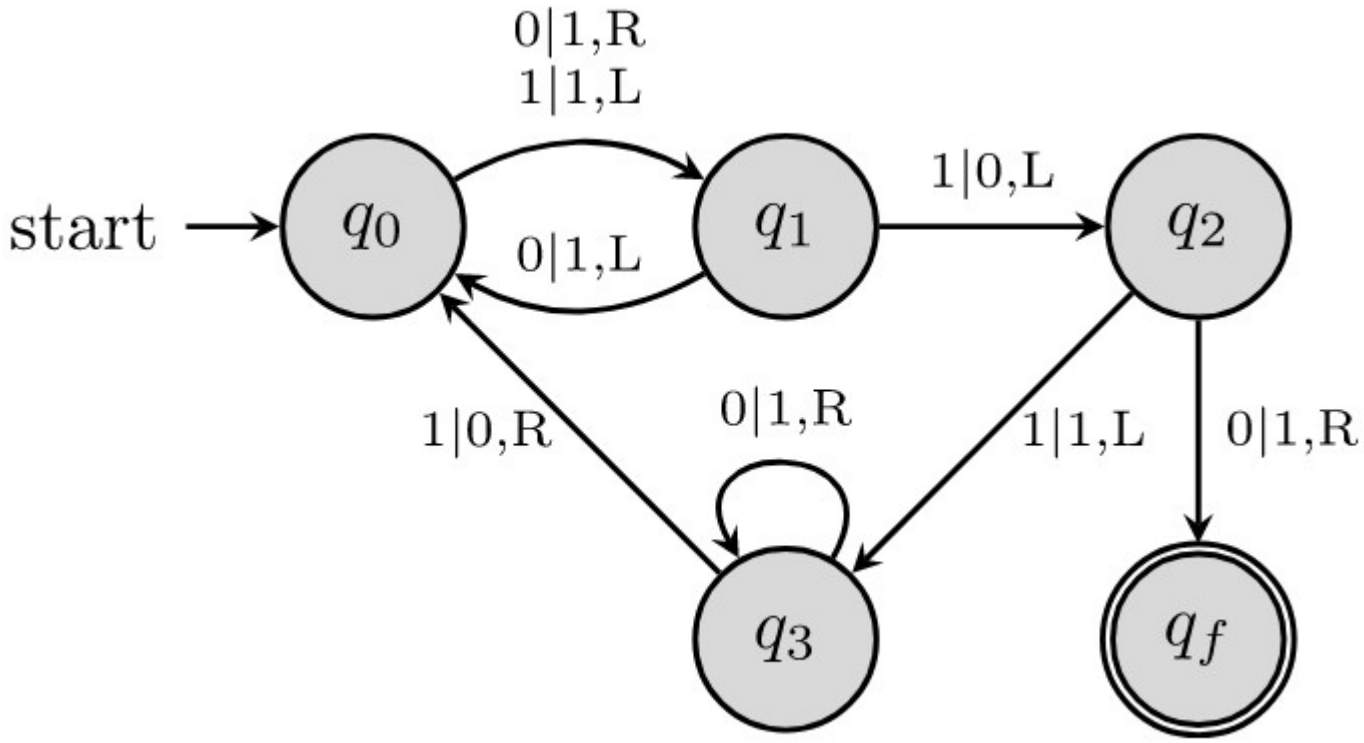
État	Symbole	
	0	**1**
A	• Écrire le symbole **1** • Déplacer le ruban à **droite** • Passer à l'état **B**	• Écrire le symbole **1** • Déplacer le ruban à **gauche** • Passer à l'état **B**
B	• Écrire le symbole **1** • Déplacer le ruban à **gauche** • Passer à l'état **A**	• Écrire le symbole **1** • Déplacer le ruban à **droite** • Passer à l'état **STOP**



Exercice 3 : Dessiner l'automate correspondant aux transitions suivantes puis tester la machine sur un ruban avec des 0.

Pour une machine à quatre états, le castor affairé correspond à la table de transition suivante

État	Symbole	
	0	**1**
A	• Écrire le symbole **1** • Déplacer le ruban à **droite** • Passer à l'état **B**	• Écrire le symbole **1** • Déplacer le ruban à **gauche** • Passer à l'état **B**
B	• Écrire le symbole **1** • Déplacer le ruban à **gauche** • Passer à l'état **A**	• Écrire le symbole **0** • Déplacer le ruban à **gauche** • Passer à l'état **C**
C	• Écrire le symbole **1** • Déplacer le ruban à **droite** • Passer à l'état **STOP**	• Écrire le symbole **1** • Déplacer le ruban à **gauche** • Passer à l'état **D**
D	• Écrire le symbole **1** • Déplacer le ruban à **droite** • Passer à l'état **D**	• Écrire le symbole **0** • Déplacer le ruban à **droite** • Passer à l'état **A**



Tester cette automate sur http://turingmachine.vassar.edu/
Combien de 1 apparaissent sur le ruban ?

Défis

A vous maintenant de concevoir des machines de Turing, c'est-à-dire me donner la table d'actions pour les machines dont le fonctionnement est décrit ci-après.
Vous travaillerez sur les 3 défis de votre choix, la difficulté étant bien entendu valorisée.

1. détecteur d'orange :
Entrée : un mot composé de couleurs parmi orange, jaune et bleu
Objectif : ajoute une brique verte en fin de mot (= après la dernière brique) si le mot ne contient pas de brique orange, ne change rien sinon

2. détecteur d'orange (2) :
Entrée : un mot composé de couleurs parmi orange, jaune et bleu
Objectif : ajoute une brique verte en fin de mot (= après la dernière brique) si le mot contient une brique orange, ne change rien sinon

3. détecteur d'orange (3) :
Entrée : un mot non vide composé de couleurs parmi orange, jaune et bleu
Objectif : remplace la dernière brique par une verte s'il y a une brique orange dans le mot, ne change rien sinon

4. détecteur d'orange (4) :
Entrée : un mot composé de couleurs parmi orange, jaune et bleu
Objectif : ajoute une brique verte en début de mot (= avant la première brique) s'il y a une brique orange dans le mot, ne change rien sinon.
Attention, dans les machines de Turing il y a toujours de l'espace libre (illimité) avant et après le mot. Il faut donc bien mettre sa tête de lecture sur la première brique du mot mais laisser de l'espace à gauche et à droite du mot sur votre ruban.

5. parité :
Entrée : un mot composé de couleurs parmi orange, vert, jaune et bleu
Objectif : l'état final doit être différent selon si le nombre de briques du mot est pair ou impair.

6. parité verte :
Entrée : un mot composé de couleurs parmi orange, vert, jaune et bleu
Objectif : idem que ci-dessus mais on doit avec l'état de contrôle à la fin savoir si le nombre de briques vertes du mot est pair ou non (les autres couleurs importent peu)

7. pas deux de suite :
Entrée : un mot composé de couleurs parmi orange, jaune et bleu
Objectif : si le mot contient deux briques de suite de la même couleur, la machine ajoute une brique verte en fin de mot

8. pas deux de suite (2) :
Entrée : un mot composé de couleurs parmi orange, jaune et bleu
Objectif : si le mot contient deux briques de suite de la même couleur, la machine remplace la deuxième brique par une verte et continue

9. pas deux de suite (3) :
Entrée : un mot composé de couleurs parmi orange, vert, jaune et bleu
Objectif : avoir en fin d'exécution un mot qui ressemble au maximum au mot d'entrée mais n'a plus deux briques de suite de la même couleur
indice : quand on remplace une brique par une autre, attention à la nouvelle couleur choisie

10. incrémenteur :
Entrée : un mot composé de couleurs parmi jaune et bleu
Objectif : on considère qu'une brique jaune vaut 0 et une brique bleue vaut 1.
Un mot sur le ruban est ainsi l'écriture binaire d'un nombre, par exemple 0110 est l'écriture binaire de 6 et 10111 l'écriture binaire de 23.
Réaliser un incrémenteur (qui ajoute 1 à un nombre donné). L'algorithme est comme en décimal, sauf qu'on n'a que deux chiffres : 0 + 1 = 1 et 1 + 1 = 0 et je retiens 1. Par exemple 0100 + 1 = 0101, et 0101 + 1 = 0110 (on a propagé une fois la retenue).

Castor Affairé

Le castor affairé de n est une machine de Turing à (n+1) états qui en prenant en entrée un bandeau contenant uniquement des 0 doit écrire le maximum de 1 possible avant de s'arrêter.
Vous avez vu dans les question précédentes les castor affairé de 2, 3 et 4 états.

A partir de 6 états, Le castor affairé est **incalculable**. Le nombre d'étapes pour que l'algorithme finisse est supérieur à $10 \uparrow\uparrow 15$, autrement dit le nombre d'étapes qu'elle effectue avant de s'arrêter est de $10^{10^{10^{\cdot^{\cdot^{\cdot^{10}}}}}}$ avec 15 itérations de puissance de 10, soit bien plus que le nombre d'atomes dans l'univers…

Conclusion

Avec cette activité le lien avec l'informatique est clair : on exécute et on écrit des programmes, OK.

Mais la question qui reste est de savoir quel est l'intérêt de continuer à utiliser ce modèle rudimentaire inventé au milieu des années 30. Pourquoi s'en souvenir plus de 80 ans plus tard alors que nous avons des ordinateurs très puissants que nous programmons dans des langages dits "évolués", bien plus proche du langage naturel ? Parce que ce modèle, si rudimentaire soit-il, a la même capacité de calcul que n'importe quel ordinateur.

Oui nos ordinateurs modernes sont bien plus rapides, efficaces, agréables à programmer, mais ce qui est incroyable c'est que pour tout problème qu'un ordinateur sait résoudre, si complexe soit-il, on peut concevoir une machine de Turing (sans doute énorme et peu efficace) qui fait la même chose. Si vous ne voyez toujours pas l'intérêt cela va venir.

Imaginez maintenant que vous trouvez un problème pour lequel vous réussissez à prouver qu'aucune machine de Turing n'est capable de le résoudre. Avec cette preuve et le fait que la machine de Turing sait faire la même chose que les ordinateurs, on obtient automatiquement le résultat suivant : aucun ordinateur, même si dans 100 ans ils sont bien plus puissants, ne pourra résoudre ce problème.

Et faire cette preuve sur les machines de Turing est rendu plus simple car le modèle de ces machines est très précis et les "instructions" ou règles sont d'un type très restreint.