DM5 : Théorème de Rice

Ce devoir est facultatif. Il est à rendre pour le 24/01 à 10h.

Partie 1 *Machines de Turing*

L'objectif de cette partie est d'introduire le concept de *machine de Turing*. Intuitivement parlant, une telle machine M est la donnée d'un automate fini A et d'un ruban consitué d'un nombre infini dénombrable de cases sur lequel se déplace une tête de lecture. Initialement, on écrit un mot u lettre à lettre sur le ruban de façon à ce que sa première lettre soit sous la tête de lecture et on se place dans l'état initial de l'automate. Partant de là, l'exécution de M sur u consiste à effectuer une suite de transitions consistant chacune à :

- Lire la lettre ℓ sous la tête de lecture.
- En fonction de ℓ et de l'état de A, changer éventuellement d'état dans A.
- Ecrire un symbole dans la case du ruban qui est sous la tête de lecture (et qui remplace donc celui qui y était).
- Déplacer la tête de lecture à gauche ou à droite d'une case.

On procède ainsi tant qu'on n'atteint pas d'état final de l'automate A. Plus formellement :

Définition : machine de Turing

Une *machine de Turing* est un tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r, \square)$ tel que :

- Q est un ensemble fini d'états.
- Σ est un alphabet appelé *alphabet d'entrée*.
- Γ est un alphabet appelé *alphabet de ruban*. Il vérifie $\Sigma \subset \Gamma$.
- $\delta : Q \setminus \{q_a, q_r\} \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ est la *fonction de transition*. Cette fonction prend en entrée (q, X) tel que q est l'état du système et X la lettre lue dans la case c sous la tête de lecture sur le ruban avant la transition. Elle lui associe (q', Y, d) tel que q' est l'état du système après la transition, Y est la lettre écrite dans la case c et d est la direction dans laquelle s'est déplacée la tête de lecture.
- $q_0 \in Q$ est *l'état initial* du système, q_a *l'état acceptant* et q_r *l'état rejetant*. Les états q_a et q_r sont finaux.
- $\square$ est un symbole de $\Gamma \setminus \Sigma$ appelé *symbole blanc*.

Le ruban infini nécessaire à l'exécution d'une machine de Turing ne fait pas partie de sa description : on l'indice loisiblement par $\mathbb{Z}$ et on peut dès lors repérer la position de la tête de lecture par l'indice de la case qu'elle pointe.

Définition : configuration et calcul

Si M est une machine de Turing, une *configuration* C de M est un élément de $Q \times \Gamma^{\mathbb{Z}} \times \mathbb{Z}$ décrivant l'état dans lequel se trouve le système, le contenu du ruban et la position de la tête de lecture. Une transition de M permet de passer d'une configuration à la suivante.

Le *calcul* de M sur un mot $u = u_0...u_{n-1} \in \Sigma^\star$ est la suite (potentiellement infinie) des configurations par lesquelles passe M au fil de ses transitions en partant de la configuration initiale $(q_0, (u_i)_{i\in\mathbb{Z}}, 0)$ où pour tout $i \notin [\![0, n-1]\!]$, $u_i = \square$. Autrement dit, initialement, on écrit le mot u sur le ruban à partir de la position 0 et toutes les autres cases du ruban contiennent le symbole blanc.

Il y a trois comportements possibles pour une machine de Turing M sur une entrée u :

- Le calcul de M sur u termine en temps fini (c'est-à-dire, le nombre de configurations visitées à partir de l'état intial est fini) et la configuration finale est $(q_a, \_, \_)$. On dit alors que M *accepte* u. Le *langage accepté* par M, noté $L(M)$, est le langage des mots de $\Sigma^\star$ qui sont acceptés par M.
- Le calcul de M sur u termine en temps fini sur la configuration finale $(q_r, \_, \_)$. On dit alors que M *refuse* u.
- Le calcul de M ne termine pas en temps fini. On dit que l'exécution de M sur u *boucle*.

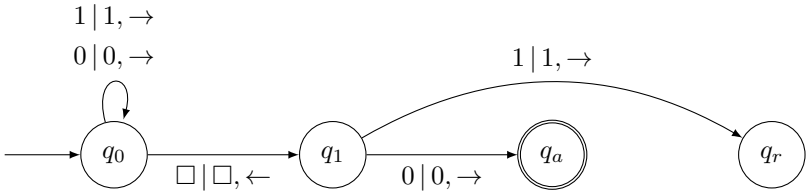
Définition : langages (semi-)décidables

Un langage $L \subset \Sigma^\star$ est :

- *semi-décidable* s'il existe une machine de Turing M telle que $L = L(M)$. On note RE l'ensemble des langages semi-décidables (sous entendu, sur Σ).
- *décidable* s'il existe une machine de Turing M telle que $L = L(M)$ et telle que tout calcul de M termine en temps fini. On note R l'ensemble des langages décidables (sous entendu, sur Σ).

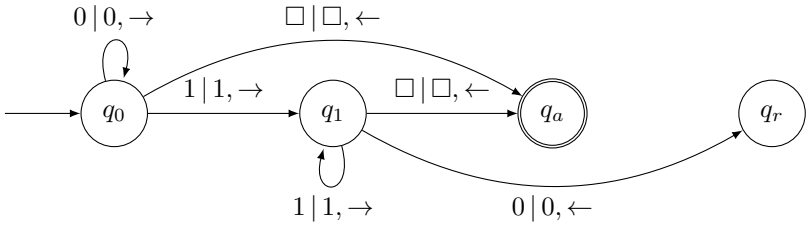
On dit par ailleurs qu'un problème est (semi-)décidable si le langage formé par les encodages des instances positives de ce problème est (semi-)décidable au sens défini ci-dessus. Remarquons par ailleurs qu'une machine de Turing M peut être représentée (encodée) par une chaîne de caractères finie unique (englobant la description de ses états, transitions, alphabets...), qu'on notera $\langle M \rangle$ et donc qu'on peut tout à fait considérer des langages dont les éléments sont des encodages de machines de Turing vérifiant telle ou telle propriété.

Tout comme avec les automates finis, on peut condenser les informations du tuple décrivant une machine de Turing via un graphe étiqueté : ses sommets sont les états de Q et on place un arc de q vers q' étiqueté par $X \,|\, Y, d$ s'il existe une transition $(q, X) \to (q', Y, d)$ dans la machine. Par exemple, on peut représenter comme suit une machine de Turing M_{pair} permettant de décider le langage sur $\{0, 1\}$ des écritures binaires d'entiers pairs :



Le principe de fonctionnement de cette machine est le suivant : on se déplace vers la droite jusqu'à trouver la fin du mot, c'est-à-dire jusqu'à trouver le premier symbole blanc. On revient alors sur nos pas d'un cran pour observer le dernier bit de l'écriture binaire du mot écrit sur le ruban : on l'accepte si et seulement si ce dernier vaut 0.

1. Indiquer la suite de configurations par lesquelles passe la machine M ci-dessous lors du calcul sur le mot 001101. Déterminer en justifiant brièvement le langage $L(M)$:



2. Proposer une machine de Turing décidant le même langage que M_{pair} mais n'autorisant que des déplacements de la tête de lecture vers la droite. Expliquer son principe.
3. Construire une machine de Turing permettant de tester si un élément de $\{0, 1\}^\star$ appartient au langage $\sum_{n \in \mathbb{N}} 0^n 1^n$ en expliquant la démarche.
4. Montrer que si L est un langage rationnel alors il existe une machine de Turing M tel que $L(M) = L$.
5. Montrer qu'il existe des langages n'appartenant pas à RE.

Interlude *Problème de l'arrêt version machines de Turing*

Dans cette partie, on réécrit la preuve de l'indécidabilité du problème de l'arrêt vue en cours mais avec les machines de Turing comme modèle de calcul. On étudie le problème :

$$\text{ARRET} : \begin{cases} \textbf{Entrée} : \text{Une machine de Turing } M \text{ et un mot } u. \\ \textbf{Sortie} : \text{Oui si le calcul de } M \text{ sur } u \text{ termine en temps fini, non sinon.} \end{cases}$$

Supposons par l'absurde qu'il existe une machine de Turing A permettant de décider ARRET, c'est-à-dire qui prend en entrée l'encodage $\langle M, u\rangle$ d'un couple (M, u) où M est une machine de Turing et u un mot, telle que A termine sur toute entrée et $L(A) = \{\langle M, u\rangle \,|\, \text{le calcul de la machine de Turing } M \text{ sur } u \text{ termine}\}$. On peut par exemple considérer que $\langle M, u\rangle$ est formé de la concaténation de l'encodage $\langle M\rangle$ de M suivi d'un caractère \$ n'appartenant à aucun des alphabets en présence puis suivi des lettres de u.

Alors on peut construire une machine de Turing P ayant le comportement suivant : P prend en entrée l'encodage d'une machine de Turing M. Si A accepte $\langle M, \langle M\rangle\rangle$ alors P boucle indéfiniment, sinon P accepte son entrée. Observons le comportement de la machine P sur son propre encodage :

Le calcul de P sur $\langle P\rangle$ ne termine pas ssi A accepte $\langle P, \langle P\rangle\rangle$ (par construction de P)
ssi $\langle P, \langle P\rangle\rangle \in L(A)$
ssi le calcul de P sur $\langle P\rangle$ termine (par définition de $L(A)$).

On aboutit à une contradiction qui assure de l'indécidabilité du langage $\{\langle M, u\rangle \,|\, \text{le calcul de la machine de Turing } M \text{ sur } u \text{ termine}\}$ ce qui équivaut à dire que le problème ARRET est indécidable.

Remarque : Pour décrire P, on se place à plus haut niveau que celui qui consisterait à décrire exactement les transitions de P. C'est légitime : pour construire P il suffit d'ajouter un nombre fini d'états et transitions à A comme suit.

- *On ajoute quelques états et transitions en amont de l'état initial de A permettant de transformer le ruban (qui contient initialement l'encodage de M) de sorte à ce qu'il contienne l'encodage de $(M, \langle M\rangle)$.*
- *On ajoute une transition depuis l'état acceptant de A vers un nouvel état p dont on fait un puits : quel que soit le symbole X lu sur le ruban, on ajoute la transition $(p, X) \to (p, X, \to)$.*
- *On ajoute une transition depuis l'état refusant de A vers l'état acceptant de P.*

On se permettra dans la partie 2 de décrire les machines de Turing comme dans cette partie plutôt qu'à partir de leurs transitions afin de clarifier les idées des preuves et s'épargner une description hautement rébarbative.

Partie 2 *Théorème de Rice*

Dans cette partie, on fixe un alphabet Σ. L'objectif est de démontrer le :

Théorème de Rice

Pour toute partie $\mathcal{P}$ telle que $\emptyset \subsetneq \mathcal{P} \subsetneq \text{RE}$, le langage $L_{\mathcal{P}}$ est indéciadable avec :

$$L_{\mathcal{P}} = \{\langle M\rangle \,|\, M \text{ est une machine de Turing telle que } L(M) \in \mathcal{P}\}$$

Dit autrement, pour toute propriété P non triviale sur les langages récursivement énumérables (non triviale au sens où il y a au moins un langage qui vérifie P et un qui ne la vérifie pas), le problème consistant à savoir si le langage $L(M)$ d'une machine de Turing vérifie P est indécidable.

Remarquez bien qu'on parle de propriété P portant sur les langages reconnus par machine de Turing et non sur les machines de Turing elles-même : P est une propriété sémantique et non syntaxique. Par exemple, on ne peut pas appliquer le théorème de Rice au langage $\{\langle M\rangle \,|\, M \text{ a au plus } k \text{ états}\}$ (qui est d'ailleurs décidable).

On fixe les notations suivantes :

- On note $\mathcal{M} = \{M_i \,|\, i \in \mathbb{N}\}$ l'ensemble des machines de Turing ayant Σ comme alphabet d'entrée (on peut d'après la question 5), et on note
$$N : \begin{cases} \Sigma^\star & \longrightarrow \mathbb{N} \\ u & \longmapsto N(u) \end{cases}$$
une fonction bijective de numérotation des mots de $\Sigma^\star$.
- On note $L_D = \{u \in \Sigma^\star \,|\, u \notin L(M_{N(u)})\}$.
- On note $L_\in = \{\langle M, u\rangle \,|\, M \in \mathcal{M}, u \in \Sigma^\star \text{ et } u \in L(M)\}$. Toute machine acceptant $L_\in$ s'appelle une machine de Turing universelle et on admet qu'il existe une telle machine. Ainsi, $L_\in \in \text{RE}$.

6. Montrer que L_D n'appartient pas à RE. En déduire que son complémentaire $\overline{L_D}$ n'est pas décidable.

Définition : Fonction calculable

Si M est une machine de Turing sur un alphabet d'entrée Σ et un alphabet de ruban Γ, et u est une entrée sur laquelle M s'arrête, on note $M(u) \in \Gamma^\star$ le mot écrit sur le ruban dans la configuration finale atteinte (en supprimant à gauche et à droite les symboles blancs inutiles).

La fonction $f : \Sigma^\star \to \Gamma^\star$ est dite *calculable* s'il existe une machine de Turing M tel que pour toute entrée $u \in \Sigma^\star$, le calcul de M sur u s'arrête avec $f(u)$ écrit sur le ruban.

De manière similaire à ce que nous avons vu en cours, on dit qu'un langage L_A sur Σ_A se réduit à un langage L_B sur Σ_B, et on note $L_A \leq L_B$, s'il existe une fonction calculable $f : \Sigma_A \to \Sigma_B$ tel que $u \in L_A \Leftrightarrow f(u) \in L_B$.

7. Expliquer pourquoi, si $L_A \leq L_B$ et L_A est indécidable alors L_B l'est aussi.

8. On considère la fonction f ayant le comportement suivant sur $u \in \Sigma^\star$:

   ```
   Déterminer N(u)
   Déterminer M_{N(u)}
   Renvoyer ⟨M_{N(u)}, u⟩
   ```

   Expliquer pourquoi f est calculable. On ne demande pas d'exhiber explicitement une machine calculant f.

9. Déduire de la question précédente que $\overline{L_D} \leq L_\in$.

10. On fixe à présent $\mathcal{P}$ telle que $\emptyset \subsetneq \mathcal{P} \subsetneq \text{RE}$ et on cherche à montrer que $L_\mathcal{P}$ est indécidable par réduction depuis $L_\in$.

    a) Pourquoi ne perd-t-on pas en généralité à supposer que le langage vide n'est pas dans $\mathcal{P}$?

    b) Comme $\mathcal{P} \neq \emptyset$, il existe un langage L de RE dans $\mathcal{P}$. Par définition, ce langage est accepté par une machine de Turing M_L. Si M est une machine de Turing et u un mot, on considère la fonction $f : \langle M, u\rangle \mapsto \langle M'\rangle$ où M' est une machine de Turing dont le fonctionnement sur une entrée $x \in \Sigma^\star$ est :

    ```
    Simuler M sur u à l'aide d'une machine de Turing universelle
    Si M accepte u
        Effectuer le calcul de M_L sur x
        Si M_L accepte x, accepter
    Refuser
    ```

    Montrer que $\langle M, u\rangle \in L_\in$ si et seulement si $L(M') \in \mathcal{P}$.

    c) En déduire le théorème de Rice.