Corrigé TD 2
Calculabilité

Luc Lapointe
luc.lapointe@ens-paris-saclay.fr
home.lmf.cnrs.fr/LucLapointe/

Exercice 1 - Palindromes sur 2 rubans

Montrer qu'il existe une machine de Turing sur 2 rubans qui reconnaît le langage des palindromes en temps linéaire.

Comparer au temps que met une machine à 1 ruban.

Solution

1. Écrire sur le deuxième ruban le mot pris en entrée.
2. Placer le curseur du premier ruban à gauche du mot, et le curseur du deuxième ruban à droite du mot.
3. Lire sur le premier ruban de gauche à droite, en même temps que sur le deuxième ruban de droite à gauche.
   - Si les lettres ne correspondent pas, rejeter.
   - Si l'intégralité du mot est lue, accepter.

En comparaison, la machine à un ruban présentée dans le TD 1 fonctionne en temps quadratique.

Exercice 2 - Clôtures

1. Montrer que la classe des langages récursifs est close par intersection, union et complémentaire.
2. Montrer que la classe des langages récursivement énumérables est close par intersection et union.
3. Montrer que la classe des langages récursivement énumérables n'est pas close par complémentaire.

Solution

Soient L_1 et L_2 deux langages récursifs, reconnus par les machines M_1 et M_2.

On construit une machine-produit à deux rubans $M_1 \times M_2$, qui exécute M_1 sur le premier ruban et exécute M_2 sur le deuxième.

Intersection Exécuter la machine-produit $M_1 \times M_2$.
- Si un état rejetant est atteint dans M_1 ou M_2, alors $M_1 \times M_2$ rejette.

- Si un état acceptant est atteint dans M_1 ou M_2, il devient un puits qui ne modifie pas le ruban. Si le deuxième état acceptant est atteint, le mot est accepté.

Union Exécuter la machine-produit $M_1 \times M_2$.

- Si un état acceptant est atteint dans M_1 ou M_2, alors $M_1 \times M_2$ accepte.
- Si un état rejetant est atteint dans M_1 ou M_2, il devient un puits qui ne modifie pas le ruban. Si le deuxième état rejetant est atteint, le mot est rejeté.

Complémentaire Comme M_1 termine, permuter **accept** et **reject** dans M_1 donne une machine qui accepte le complémentaire de L_1.

Soient L_1 et L_2 deux langages récursivement énumérables, reconnus par les machines M_1 et M_2.

Les constructions pour l'union et l'intersection sont exactement les mêmes que pour des langages récursifs. Voici en plus la preuve qu'elles sont correctes :

Intersection Soit $m \in L_1 \cap L_2$. Alors m est accepté par M_1 et par M_2, donc $M_1 \times M_2$ accepte bien m.

Soit $m \notin L_1 \cap L_2$. Supposons sans perte de généralité que $m \notin L_1$. Alors soit m est rejeté par M_1, soit M_1 ne termine par sur m. Dans le premier cas, $M_1 \times M_2$ rejette m et dans le deuxième cas, soit elle rejette car M_2 rejette m, soit elle ne termine pas car M_2 ne termine pas sur m.

Union Soit $m \in L_1 \cup L_2$. Supposons sans perte de généralité que $m \in L_1$. Alors m est accepté par M_1, donc $M_1 \times M_2$ accepte m.

Soit $m \notin L_1 \cup L_2$. Alors, pour $i \in \{1, 2\}$ soit m est rejeté par M_i, soit M_i ne termine par sur m. Si m est rejeté par les deux, alors $M_1 \times M_2$ rejette M. Si une des deux ne termine pas sur m, alors $M_1 \times M_2$ ne termine pas sur m.

- *(Admis si vous n'avez pas encore vu la machine universelle.)* À encodage fixé, le langage des encodages de machines de Turing qui s'arrêtent sur le mot vide est récursivement énumérable. **Preuve :** Une machine universelle qui, sur une entrée qui correspond à un encodage de machine, l'exécute sur ε, et sinon rejette, convient. ■ Ce n'est par contre pas un langage récursif : sinon le problème de l'arrêt serait décidable.
- *(Sinon, Propriété de cours)* Il existe un langage récursivement énumérable mais non récursif.
- Propriété (admise) : Si un langage est à la fois récursivement énumérable et co-récursivement énumérable, il est récursif.

- Donc les langages récursivement énumérables ne sont pas clos par complémentaires.

Exercice 3 - Fonctions calculables et langages récursivement énumérables

1. Soit f une fonction calculable. Montrer que l'image de f est un langage récursivement énumérable.
2. Montrer que l'image d'une fonction quelconque n'est pas forcément un langage récursivement énumérable.

Ébauche de solution

1. Soit f une fonction calculable sur Σ. Il est possible d'énumérer les mots de Σ^*. On peut donc définir par exemple une machine M' à deux rubans qui, sur l'entrée m :
   - Sur le premier ruban énumère les mots de Σ^*, selon l'ordre lexicographique par exemple.
   - Sur le deuxième ruban, entre chaque énumération du premier ruban, recopier le premier ruban sur le deuxième, puis exécuter f dessus. Si le résultat est m, la machine accepte.

   Cette machine est constructible car f est calculable.
2. Attention, dire « la preuve précédente ne fonctionne plus car f n'est pas calculable » n'est pas une preuve !

   Pour cette question, on peut par exemple considérer une fonction f qui renvoie son entrée si ladite entrée est le code d'une machine de Turing qui s'arrête sur ε, et qui renvoie ε sinon. Son image est l'ensemble des codes de machines de Turing qui s'arrêtent sur ε, ainsi que ε. Ce langage n'est pas récursivement énumérable.

Exercice 4 - Image de langages récursivement énumérables

1. Soit f une fonction calculable. Montrer que si L est récursivement énumérable, alors $f(L)$ aussi.
2. Montrer que l'implication devient fausse avec L quelconque ou f quelconque.

Ébauche de solution

1. *(Admis)* Pour L un langage récursivement énumérable, il est possible d'énumérer les mots de L.
   Une fois cette propriété admise, la preuve est similaire à l'exercice précédent, mais en énumérant les mots de L plutôt que ceux de Σ^*.
2. f quelconque : idem exercice précédent.

   L quelconque : On considère f la fonction identité, évidemment calculable, et L l'ensemble des codes de machines de Turing qui s'arrêtent sur le mot vide. $f(L)$ n'est pas récursivement énumérable.

Exercice 5 - MTF

Une machine de Turing avec états finaux (MTF) est presque comme une machine de Turing du cours, mais il n'y a pas d'état acceptant ou rejetant.

Un mot en entrée est *accepté* s'il existe un calcul qui termine dans un état final, et *rejeté* sinon. Un langage est *MTF-semi-décidable* s'il existe une MTF qui accepte exactement ses mots. Si en plus cette MTF ne permet de calcul infini sur aucune entrée, le langage est *MTF-décidable*.

La fonction associée à une MTF a pour *domaine* le langage reconnu et pour *valeur* le plus grand préfixe sans blanc sur le ruban au moment de l'arrêt. Cette fonction est dite *MTF-semi-calculable*. Si son domaine est MTF-décidable, elle est dite *MTF-calculable*.

Montrer les propriétés suivantes :

1. Un langage est semi-décidable ssi il est MTF-semi-décidable.
2. Un langage est décidable ssi il est MTF-décidable.
3. Une fonction est semi-calculable ssi elle est MTF-semi-calculable.
4. Une fonction est calculable ssi elle est MTF-calculable.

Idée de la construction

Ce corrigé n'est pas détaillé, et donne juste des idées pour passer d'un sens à l'autre.

Passer de MT à MTF On remplace l'état acceptant par deux états finaux f_g et f_d. Le premier (resp. second), sur tout caractère lu, déplace le curseur à gauche (resp. droite) sans modifier le ruban, et transitionne vers f_d (resp. f_g).

Pour les états rejetant : idem, sauf que les deux états rajoutés ne sont pas finaux.

Passer de MTF à MT On commence par copier toute la MTF en oubliant quel état est final, et on rajoute un état acceptant. Pour chaque transition dans la MTF qui pointe vers un état final, on ajoute une transition identique dans la MT, mais qui pointe vers l'état acceptant. Ceci crée à chaque fois une transition non-déterministe.

Une fois ces transitions exhibées, il faut encore prouver qu'elles fonctionnent. N'hésitez pas à me contacter si vous n'y arrivez pas.