IFT2105-A19 : Solutions du Devoir 5

Problème 1

On peut utiliser le théorème de Rice, avec $S = \mathsf{P}$. Il suffit de montrer que P est non-trivial au sens du théorème de Rice, donc qu'il existe une MT M^* telle que $L(M^*) \in \mathsf{P}$ et une autre MT $M^\dagger$ telle que $L(M^\dagger) \notin \mathsf{P}$. Pour M^*, on peut choisir une MT qui accepte tout les mots (on peut le faire en temps constant, c'est donc clairement polynomial). Pour $M^\dagger$, on peut choisir une MT qui accepte A_{MT}, donc une machine qui prend en entrée $\langle M, w\rangle$, qui simule M sur w et qui accepte si M accepte. Vu que A_{MT} est indécidable, $A_{\text{MT}} \notin \mathsf{P}$. Évidemment, d'autres choix sont possibles pour ces deux machines, et on peut également prouver l'énoncé sans passer par le théorème de Rice.

Problème 2

On commence par montrer que $L \notin \mathsf{REC}$. Pour ce faire, on montre que $\overline{A_{\text{MT}}} \leq L$; puisque $\overline{A_{\text{MT}}} \notin \mathsf{REC}$, ceci démontrera que $L \notin \mathsf{REC}$. La réduction suivante fonctionne :

$$f(y) = \begin{cases} \langle M_{\text{oui}}, 0\rangle & \text{si } y \text{ n'est pas un encodage valide de type } \langle M, w\rangle \\ \langle M', 1\rangle & \text{si } y = \langle M, w\rangle \end{cases}$$

où M_{oui} est une MT qui accepte dès le début du calcul, et où M' est la MT suivante :

1. Simule M sur w
2. Si M accepte w, alors accepte.
3. Si M rejette w, alors boucle.

On voit que si $\langle M, w\rangle \in \overline{A_{\text{MT}}}$, alors soit M rejette ou boucle sur w, et alors M' boucle sur entrée vide, et donc $\langle M', 1\rangle \in L$. Si $\langle M, w\rangle \notin \overline{A_{\text{MT}}}$, alors M accepte w, et alors $\langle M', 1\rangle \notin L$. De plus, si y n'est pas de la bonne forme, alors $y \in \overline{A_{\text{MT}}}$, et alors la réduction nous donne $\langle M_{\text{oui}}, 0\rangle \in L$.

Pour prouver que $\overline{L}$ n'est pas reconnaissable, on prouve la réduction $\overline{A_{\text{MT}}} \leq \overline{L}$, ce qui est équivalent à prouver que $A_{\text{MT}} \leq L$. On utilise alors la réduction suivante :

$$g(y) = \begin{cases} \varepsilon & \text{si } y \text{ n'est pas un encodage valide de type } \langle M, w\rangle \\ \langle M', 0\rangle & \text{si } y = \langle M, w\rangle \end{cases}$$

où M' est la même MT que dans l'autre réduction. On voit que si $\langle M, w\rangle \in A_{\text{MT}}$, alors M accepte w, et alors M' accepte sur entrée vide, et donc $\langle M', 0\rangle \in L$. Si $\langle M, w\rangle \notin A_{\text{MT}}$, alors M rejette ou boucle sur w, et alors M' boucle sur entrée vide, et donc $\langle M', 0\rangle \notin L$. De plus, si y n'est pas de la bonne forme, alors $y \notin A_{\text{MT}}$, et alors la réduction nous donne $\varepsilon \notin L$.

Problème 3

(a) Supposons que L_1 et L_2 sont deux langages décidables. Étant donné un mot $w = w_1 \cdots w_n$, on peut alors décider si $w \in L_1 \circ L_2$ par la procédure suivante :

Pour $i = 0$ à n:
    Si $w_1 \cdots w_i \in L_1$ et $w_{i+1} \cdots w_n \in L_2$, accepte
Rejette

Autrement dit, on teste toute les coupures possibles de w en deux mots, et on accepte si une des coupures est telle que le premier mot est dans L_1 et le deuxième mot est dans L_2.

(b) Supposons que L_1 et L_2 sont deux langages reconnus par des MT M_1 et M_2 respectivement. Alors, on peut construire une MT qui reconnaît $L_1 \cap L_2$ en simulant d'abord M_1 sur l'input, si celle-ci accepte, alors on simule M_2 sur w, et si celle-ci accepte aussi, on accepte. Si une des machines rejette, on rejette, et clairement si une des machines boucle, on boucle. Cette machine accepte w ssi w est accepté par M_1 et M_2, ce qui veut dire que $w \in L_1 \cap L_2$.

(c) Pour l'union, on fait comme pour l'intersection, mais il faut simuler M_1 et M_2 en parallèle, car on doit accepter si une des deux machines accepte. Or, si on simule d'abord M_1 puis ensuite M_2 et que M_1 boucle, on ne pourra jamais tester si M_2 accepte.

Problème 4

Pour prouver que $f(n)$ n'est pas calculable par un programme `TANTQUE`, on suppose qu'au contraire il existe un programme `TANTQUE` `F` qui calcule f, et on montre que ceci mène à une contradiction, un peu comme on l'a fait pour prouver que A_{MT} est indécidable. L'idée sera de créer un programme à N lignes de code qui produit en sortie $f(N) + 1$, ce qui contredit la définition de f. On veut donc un programme qui finit par

$$r_0 \leftarrow \texttt{F}(r_1)$$
$$\texttt{inc}(r_0)$$

où on aura réussi à mettre le nombre de lignes de code de notre programme dans r_1 avant d'arriver à ces deux dernières lignes. Pour commencer le programme, supposons qu'on initialise r_2 à un nombre n_0 qu'on déterminera plus tard. Ceci coûte n_0 lignes de code, via une série d'instructions `inc`. Maintenant, pour pouvoir obtenir un nombre plus grand que le nombre de lignes de code, on ajoute un routine qui va mettre $2n_0$ dans le registre r_1. On peut faire ceci avec une boucle :

```
tant que r₂ ≠ r₃ [
      inc(r₃)
      inc(r₁)
      inc(r₁)
]
```

On a donc le programme suivant :

```
inc(r₂)
inc(r₂)
⋮
inc(r₂)
tant que r₂ ≠ r₃ [
      inc(r₃)
      inc(r₁)
      inc(r₁)
]
r₀ ← F(r₁)
inc(r₀)
```

Si `F` a n_F lignes de code, ce programme a $N = n_0 + n_F + 5$ lignes. Pour que notre preuve marche, il faut donc que $N = 2n_0$. On peut alors résoudre l'équation $2n_0 = n_0 + n_F + 5$ et on obtient $n_0 = n_F + 5$. Il suffit donc de mettre $n_F + 5$ incrémentations au début pour obtenir notre contradiction.

Note : La version plus connue de la fonction $f(n)$ s'appelle la « Busy Beaver function », qui est plutôt définie en terme de machines de Turing :

$$BB(n) = \{x \in \mathbb{N} \mid \text{il existe une MT à } n \text{ états ou moins qui calcule } x \text{ et s'arrête}\}.$$

Pour bien spécifier la fonction, on spécifie que l'alphabet de ruban est $\{1, \sqcup\}$, et la MT doit calculer x en unaire. Évidemment, $BB(n)$ est également incalculable.