TD 12

Exercice 1. *Canard analphabète*

On dit qu'une machine **énumère** un langage L si elle écrit tous les $w \in L$ sur son ruban. (Si L est infini la machine ne termine pas). On rappelle qu'un langage est récursivement énumérable s'il existe une machine de Turing qui accepte exactement les mots du langage.

1. Montrer qu'un langage est récursivement énumérable si et seulement si il existe une machine de Turing qui l'énumère.
2. Soit $\prec$ la relation tel que $u \prec v$ si et seulement si $|u| < |v|$ ou $|u| = |v|$ et $u <_{lex} v$. Montrer que s'il existe une machine de Turing qui énumère L dans l'ordre $\prec$, alors L est récursif.
3. Soit L un langage récursivement énumérable infini. Montrer qu'il existe un sous-ensemble infini de L qui est récursif.

Exercice 2. *La poste par correspondance*

Σ est un alphabet fini et P un ensemble fini de paires de mots sur Σ. Le Problème de Correspondance de Post associé à Σ, P est l'existence d'une suite non vide $(v_i, w_i)_i$ d'éléments de P telle que la concaténation des v_i soit égale à la concaténation des w_i. Le Problème de Correspondance de Post Modifié est celui de l'existence d'une telle suite lorsque le premier terme est fixé.

1. Résoudre PCP pour les instances suivantes :
   1. $P = (aab, ab), (bab, ba), (aab, abab)$
   2. $P = (a, ab), (ba, aba), (b, aba), (bba, b)$
   3. $P = (ab, bb), (aa, ba), (ab, abb), (bb, bab)$
   4. $P = (a, abb), (aab, b), (b, aa), (bb, bba)$
2. Montrer que si Σ ne contient qu'une lettre le problème est décidable.
3. Montrer l'équivalence entre PCP et PCPM, c'est à dire qu'à partir d'un algorithme résolvant toute instance de PCP, vous pouvez créer un algorithme résolvant toute instance de $PCPM$, et inversement.
   Indication : Dans le cas PCP permet de résoudre PCPM, vous pourrez ajouter à l'alphabet deux lettres $*$ et \$ et utiliser les deux morphismes p et s suivant :
   $\forall a_1 \dots a_k \in \Sigma^*$, $p(a_1 \dots a_k) = *a_1 * \dots * a_k$ *et* $s(a_1 \dots a_k) = a_1 * \dots a_k *$.
4. Peut-on se passer des couples de la forme (w, w) dans PCPM ?
5. Montrer que PCP est indécidable.
   Indication : Pour montrer cela, vous pouvez montrer que PCP permet de résoudre l'arrêt : à une machine M et une entrée x on peut créer une instance de PCP qui est acceptée si et seulement si la machine M s'arrête sur l'entrée x.

Exercice 3. *Moins de place*

Une machine de Turing est dite *linéairement bornée* si elle n'écrit pas en dehors de l'espace utilisé par la donnée.

Comme on peut toujours supposer qu'une machine de Turing n'écrit pas le symbole blanc $\sqcup$ (en le dupliquant éventuellement en un autre symbole $\sqcup_2$), on peut aussi dire qu'une machine linéairement bornée est une machine telle que si $\delta(p, \sqcup) \ni (q, b, x)$ est une transition alors $b = \sqcup$ et $x = \leftarrow$. Ceci empêche la machine de modifier les symboles blancs présents sur la bande.

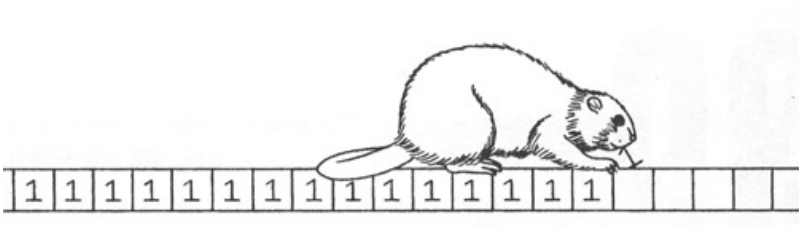
1. Etant donné un mot w et une machine linéairement bornée M, peut-on décider si M accepte w ?
2. Etant donné une machine linéairement bornée M, peut-on décider si M n'accepte aucun mot, c'est-à-dire $L(M) = \emptyset$?

   Indice : On pourra commencer par montrer que ce problème est indécidable pour les machines de Turing en général.

Exercice 4. *Castor*

On considère des machines de Turing à une bande bi-infinie et à n états, plus un état d'arrêt noté H. L'alphabet est $\Sigma = \{0, 1\}$, où 0 fait également office de symbole blanc. La tête de lecture ne peut pas rester sur place.

On appelle *Castor Affairé*, ou *Busy Beaver* [TIBOR RADÓ, 1962], la fonction qui à n associe le nombre maximal de 1 qu'il est possible d'obtenir par une machine de Turing à n états dont la bande ne contient initialement que des 0 et **qui s'arrête** (et telle que définie ci-dessus). On note $\mathbf{BB}(n)$ cette fonction. On définit de la même façon la fonction *Max Shift* telle que $\mathbf{MS}(n)$ est le nombre maximal possible de transitions pour n états.



1. Combien existe-t-il de machines à 2 symboles et n états (plus l'état d'arrêt H) ?
2. Combien existe-t-il de machines à 2 symboles et à 2 états ? Construire une (ou des) machines qui atteignent les scores $\mathbf{BB}(2)$ et $\mathbf{MS}(2)$.

   Actuellement, nous connaissons les valeurs exactes de $\mathbf{BB}(n)$ et $\mathbf{MS}(n)$ uniquement pour quelques petites valeurs de n. On peut constater (voir Tab. 1) que ces fonctions croissent très vite : en fait elles croissent plus vite que n'importe quelle fonction calculable.

n	$\mathbf{BB}(n)$	$\mathbf{MS}(n)$
2	4	6
3	6	21
4	13	107
5	$\geq$ 4098	$\geq$ 47 176 870
6	$\geq 3,5 \times 10^{18276}$	$\geq 7,4 \times 10^{36534}$

TABLE 1 – Les premières valeurs de **BB** et **MS**.

3. Montrer que la fonction **BB** n'est pas calculable par une machine de Turing.

   Indice : on pourra considérer la composition de machines de Turing, dont une qui double le nombre de 1 sur le ruban.
4. Montrer que la fonction **MS** n'est pas calculable.