Série 10

Exercice 1

Oui, il est possible de décider si cette machine va s'arrêter. En effet, si une MT se retrouve deux fois dans la même configuration, alors cette MT boucle à l'infini, car la séquence de configurations qui a mené à la configuration déjà visitée va se répéter indéfiniment.

Soit M, un automate linéairement borné partant sur l'input w de longueur k, avec un alphabet de bande de cardinalité y et un nombre l d'états.

On peut décider si cet automate s'arrête en le simulant au moyen d'une MT A :

On compte les pas de calcul de M.

Si M s'arrête (accepte ou rejette) avant les premiers $y^k lk$ pas de calcul, alors A accepte.

Si M ne s'arrête pas avant les premiers $y^k lk$ pas de calcul, alors A rejette.

Exercice 2

Montrer que la collection des langages Turing-reconnaissables est close par les opérations union, concaténation, star et intersection.
Considérons L_1, L_2 deux langages Turing-reconnaissables et $M_1(L_1), M_2(L_2)$ les machines qui les reconnaissent. On peut ensuite constuire différentes machines pour démontrer ces opérations. Notons qu'il faut faire attention à bien tester les deux machines en parallèle et pas de manière séquentielle au cas où la première ne s'arrêterait pas.

1. union
   On peut construire une MT qui reconnait $L_1 \cup L_2$. Avec une machine à plusieurs bande qui simule M_1 sur une bande et M_2 sur l'autre. Pour chaque pas de notre MT on effectue un pas de M_1 et de M_2. On accepte si une des deux machines simulées accepte, et on rejette si les deux machines rejetent.
2. concaténation
   Il y a $n+1$ moyen de décomposer l'input w de longueur n en $w = y \cdot z$. Pour chaque combinaison il faut tester si $y \epsilon L_1$, $z \epsilon L_2$. Il ne faut pas oublier de les tester en parallèle à cause du problème de l'arrêt ! Si l'on trouve au moins une combinaison pour laquelle $y \epsilon L_1$, $z \epsilon L_2$ est vrai, alors on accepte, autrement on rejette ou on boucle.
3. star
   Il y a 2^{n-1} combinaisons pour partager l'input w en sous-parties $w_1, .., w_l$ telles que $w = w_1 \cdot ... \cdot w_l$. Il faut tester pour toutes ces combinaisons (toujours de manière simultanée, en parallèle) s'il existe un cas où $w = w_1 \cdot ... \cdot w_l$ est vrai pour tous les w_i, auquel cas on accepte. S'il n'existe pas de telle combinaison, on rejette.
4. intersection
   On peut construire une MT qui reconnait $L_1 \cap L_2$ avec une machine à plusieurs bandes qui simule M_1 sur une bande et M_2 sur l'autre. Pour chaque pas de notre MT on effectue un pas de M_1 et de M_2. On accepte si les deux machines simulées acceptent, et on rejette si une des deux machines rejette.

Exercice 3

Montrer que la collection des langages Turing-décidables est close par les opérations...
Considérons L_1, L_2 deux langages décidables et $M_1(L_1), M_2(L_2)$ les machines qui les décident. Contrairement à l'exercice 2, on n'a pas besoin de se préoccuper de vérifier les machines en parallèle, car il s'agit de décideurs qui, par définition, s'arrêtent de toute façon.

1. union
   On peut construire une MT qui décide $L_1 \cup L_2$. On simule en premier la machine M_1, puis la machine M_2. On accepte dès que une des deux machine accepte et on rejette si les deux rejettent.
2. concaténation
   On peut construire une MT qui décide $L_1 \cdot L_2$. On simule en premier la machine M_1, puis la machine M_2 pour tous les cas possibles de découpage de notre input ($n+1$ possibilités, voir ex.2). On accepte dès que l'on trouve une solution où les deux machines acceptent, sinon, on rejette.
3. star
   On peut construire une MT qui décide L_1^*. On simule toutes les combinaisons (2^{n+1} possibilités, voir ex.2) de découpage de l'input et on accepte si on trouve une combinaison où $w = w_1 \cdot ... \cdot w_l$ est vrai pour tous les w_i. Sinon, on rejette.
4. complémentation
   On peut construire une machine qui reconnait le complément de L_1, uniquement car L_1 est décidable. Si M_1 accepte alors notre MT rejette et si M_1 rejette alors MT accepte. On remarque que L_1^C est décidable ssi L_1 l'est aussi.
5. intersection
   On peut construire une MT qui décide $L_1 \cap L_2$. On simule en premier la machine M_1, puis la machine M_2. On rejette dès qu'une des deux machines rejette et on accepte si les deux acceptent.

Exercice 4

La collection des langages Turing-reconnaissables n'est pas close par complémentation.
Pour un Langage L Turing-reconnaissable, son complément L^C ne l'est pas, car si L et L^C étaient Turing-reconnaissables L serait décidable. Ce qui est une contradiction !