**Magistère de Mathématiques et d'Informatique** **Pierre-Alain Reynier**
**ENS de Cachan** *9 octobre 2006*
**TD de Calculabilité et de Complexité**

# Indécidabilité, Réductions

Dans cette feuille d'exercices, nous utiliserons l'abréviation $R(\Sigma)$ (resp. $RE(\Sigma)$) pour dénoter l'ensemble des langages récursifs (resp. récursivement énumérables) sur l'alphabet $\Sigma$. Quand le contexte le permettra, nous omettrons $\Sigma$.

**Exercice 1**
Fixons un alphabet $\Sigma$. Donner un algorithme qui, étant données une entrée $x \in \Sigma^*$, et la description d'une machine de Turing $M$ acceptant le langage $\{x\#y \in \Sigma^*\#\Sigma^* \mid (x,y) \in \Delta\}$ où $\Delta$ est une relation ($\Delta \subseteq \Sigma^* \times \Sigma^*$), construit la description d'une machine de Turing $M_x$ qui accepte le langage $\{y \in \Sigma^* \mid (x,y) \in \Delta\}$.

**Exercice 2**
Soit $J = \{w \in \{0,1\}^* \mid w = 0x$ pour un $x \in LU$ ou $w = 1y$ pour un $y \notin LU\}$.
Montrer que ni $J$ ni son complémentaire ne sont récursivement énumérables.

**Exercice 3**
On considère l'ensemble des propriétés suivantes :

$$Prop = \{= \emptyset,\ \neq \emptyset,\ \in R,\ \notin R,\ \in RE,\ \notin RE,\ \text{fini},\ \text{infini}\}$$

Étant donné un élément $\mathcal{P} \in Prop$, on définit le langage $L_{\mathcal{P}} = \{< M > \in \{0,1\}^* \mid L(M) \models \mathcal{P}\}$.
Par exemple, $L_{=\emptyset}$ est l'ensemble des codes des machines de Turing dont le langage est vide.

1. Que pouvez-vous dire de $L_{\in RE}$ et $L_{\notin RE}$ ?
2. Montrer que $L_{\neq\emptyset}$ est récursivement énumérable mais non récursif. Montrer que $L_{=\emptyset}$ n'est pas récursivement énumérable.
3. Qu'en est-il de $L_{\in R}$ et $L_{\notin R}$ ?
4. Montrer enfin que les langages $L_{\text{fini}}$ et $L_{\text{infini}}$ ne sont pas récursivement énumérables.

**Exercice 4**
On considère

$EGALITE = \{< M_1, M_2 > \in \Sigma^* \mid L(M_1) = L(M_2)$ où $M_1$ et $M_2$ sont des machines de Turing$\}$.

Montrer que ni $EGALITE$ ni son complémentaire ne sont récursivement énumérables.

**Exercice 5**
Montrer qu'un langage $X \subseteq \Sigma^*$ est récursivement énumérable si et seulement s'il existe un langage récursif $D \subseteq \Sigma^*\#\Sigma^*$ tel que : $X = \{x \in \Sigma^* \mid$ il existe $y \in \Sigma^*, x\#y \in D\}$.

**Exercice 6 (Théorème du point fixe)**
Soit $f$ est une fonction récursive qui envoie les codes de machines de Turing sur des codes de machine de Turing. Montrer qu'il existe une machine $M_0$ telle que, sur toute donnée $x$, $f(< M_0 >)(x) = M_0(x)$. (Autrement dit $M_0$ et $f(< M_0 >)$ sont extensionnellement égales – par abus de notation, on confond la machine et son code).
*Indication* : considérer la fonction $g$ qui à $< M >$ associe le code de la machine $M(M)$ qui à $x$ associe $M(M)(x)$ quand $M(M)$ est le code d'une machine de Turing et ne termine pas sinon. Condidérer ensuite la machine $g(M_{f(g)})$.

**Exercice 7 (Automates linéairement bornés)**
Un *Automate linéairement borné* est une machine de Turing dont l'espace de travail est limité à droite par l'extrémité du mot d'entrée. Montrer que le problème suivant est indécidable :

**Donnée :** Un automate linéairement borné $M$.

**Question :** Est-ce que, pour tout $w$, $M$ s'arrête sur $w$ ?

**Exercice 8 (Générateurs de langages RE)**
Montrer qu'il est possible de générer grâce à une machine de Turing tout langage récursivement énumérable.

**Exercice 9 (Générateurs de langages récursifs)**
Montrer que si un langage peut être énuméré par une machine de Turing en ordre lexicographique, il est décidable.