Théorème 6.5.1 *Le langage* universel

$L_U = \{< M, w > \mid w \in L(M), M \textit{machine de Turing sur un alphabet } \Sigma, w \in \Sigma^*\}$

est récursivement énumérable.

Preuve:
On ne donne que les grandes lignes.

On utilise trois rubans sans perdre de généralité d'après le théorème 6.4.2.

On utilisera l'alphabet $\Sigma_U = \{0\ 1\ ;\ ,\ d_\Sigma\ d_\delta\ f_\delta\ \mapsto \to \downarrow \leftarrow\ B\ \$\}$.

Le codage $c(\gamma)$ d'une configuration de la machine $M = (Q, \Sigma, q_0, \delta, B, \$)$ est défini par $c(w, q, w') = \$ < w >, c(q), < w' >$. On définit M_U de manière à ce que, pour toute machine M, si γ est une configuration de M alors $\gamma \vdash_M \gamma'$ si et seulement si $(q_s, (\epsilon, \$ < M >), (\epsilon, c(\gamma)), (\epsilon, \$)) \vdash^*_{M_U} (q_s, (\epsilon, \$ < M >), (\epsilon, c(\gamma')), (\epsilon, \$))$ pour un certain état q_s de la machine M_U.

La machine universelle commence par écrire la configuration initiale de M sur le deuxième ruban : à partir de $(q_0^U, (\epsilon, < M, w >), (\epsilon, \$))$ on définit (ce que nous ne détaillons pas ici) des transitions permettant d'atteindre la configuration $(q_s, (\epsilon, < M >), (\epsilon, \$c(\gamma_0))$ où γ_0 est la configuration initiale de M sur w. Il s'agit essentiellement d'avancer la première tête de lecture jusqu'à la partie de la donnée correspondant à $< w >$, puis de déplacer $< w >$ sur le deuxième ruban. Notons aussi que copier q_0 s'effectue en se plaçant au début de $< Q >$ sur le premier ruban, recopiant $< Q >$, puis en changeant le dernier bit à 1.

Puis, si $\gamma \mapsto_M \gamma'$, on procède comme suit :

1. Se placer au début (du codage) de la première transition sur le premier ruban et au début (du codage) de l'état de M sur le second ruban.
2. On progresse simultanément sur les deux rubans tant que les symboles sont identiques. Comme f_δ apparaît à droite de la tête de lecture sur le premier ruban et pas sur le deuxième ruban, on atteint une situation dans laquelle les deux symboles ne sont pas identiques. On distingue alors suivant les cas :
   (a) si les symboles différents sur les deux rubans sont dans $\{0, 1\}$, alors on avance la tête de lecture sur le premier ruban jusqu'au début de la prochaine transition (repérée par ";") s'il y en a une. Sinon, la machine s'arrête (aucune transition n'est possible depuis la configuration de M codée sur le deuxième ruban). Sur le deuxième ruban, on revient à la position du début du codage de l'état de M (repéré par ",").
   (b) Sinon : on a trouvé une transition possible. Soit $(q_1, (\alpha c(q)", "c(a) \mapsto , c(q')", "c(b)\beta), (< w > ", ", c(q)", "c(a) < w' >))$ la configuration de M_U. Sur le premier ruban, on se déplace jusqu'au prochain symbole de déplacement m, que l'on mémorise dans l'état. Selon ce symbole,
      — Si $m =\downarrow$, on recopie $c(q')", "c(b)$ sur le deuxième ruban
      — Si $m =\to$ on recopie $c(b)$ sur le deuxième ruban (en écrasant $", "c(q)$) puis on recopie $", "c(q')", "$ sur le deuxième ruban. Comme tous les états et tous les symboles ont même longueur, on se retrouve avec la configuration $(< w > c(b)", "c(q')", ", < w' >)$

sur le deuxième ruban. Si $< w' >= \epsilon$ (i.e., le symbole pointé par la tête du deuxième ruban est un blanc), on copie $c(B)$ sur le deuxième ruban. Recopier $c(B)$ s'effectue comme la copie de $c(q_0)$: on se place au début de $< \Sigma >$ sur le premier ruban, on recopie $< \Sigma >$, puis on change le dernier bit à 1.

— Si $m =\leftarrow$, soit $w = w_1 a'$. On recule de $| < \Sigma > |+1$ symboles sur le deuxième ruban (ce qui est possible en utilisant $< \Sigma >$ sur le premier ruban), puis on recopie $| < \Sigma > |$ symboles sur le troisième ruban (autrement dit, on recopie $c(a')$ sur le troisième ruban), on revient à nouveau $| < \Sigma > |$ symboles en arrière sur le deuxième ruban, puis on recopie $","c(q')","$ sur le deuxième ruban (ceci écrase, au moins en partie, $c(a')$, qui a auparavant été sauvegardé sur le troisième ruban) . Puis on recopie le troisième ruban sur le deuxième. Enfin, on recopie $c(b)$ sur le deuxième ruban. On arrive ainsi dans la configuration $(< w_1 > ","c(q')","c(a')c(b), < w' >)$ su second ruban.

3. On revient au début des deux rubans et on efface le troisième ruban.

On se retrouve alors dans le codage de la configuration suivante de M.

Remarquons que les codages sont calculables. Par exemple :

Lemme 6.5.1 *La fonction qui à $< M >$ associe $< M, < M >>$ est calculable.*

En effet, il suffit de recoder $< M >$ une deuxième fois en utilisant les codages des symboles 0 et 1.

Exercice 160
Expliciter le codage

Exercice 161
Montrer que $\{< M > \mid M$ Machine de Turing $\}$ est récursif.

Théorème 6.5.2 *L_u n'est pas co-récursivement énumérable (et donc pas récursif).*

$\overline{L_u} = \{< M, w > \mid w \notin L(M)\}$. Si ce langage était r.é, soit M_N une machine de Turing telle que $L(M_N) = \overline{L_u}$. On construit alors une machine de Turing D telle que $L(D) = \{< M > \mid < M >\notin L(M)\}$: sur la donnée $< M >$, D simplement recopie le code de M pour obtenir $< M, < M >>$ puis applique M_N. Mais alors,

$$< D >\in L(D) \Longleftrightarrow < D >\notin L(D)$$

Ce qui est absurde.

6.6 Problème de l'arrêt

Soit $L_{arret} = \{< M, x > \mid M(x) \neq \perp\}$.

Théorème 6.6.1 *L_{arret} est récursivement énumérable et pas co-récursivement énumérable (donc indécidable).*

Tout d'abord, L_{arret} est récursivement énumérable : considérons la machine M_{arret} qui, étant donné $< M, x >$, simule la machine universelle et, quand celle-ci est sur le point d'accepter ou de rejecter, accepte. M_{arret} accepte L_{arret}.

Si L_{arret} était co-récursivement énumérable, soit $\overline{M_{arret}}$ une machine qui accepte $\overline{L_{arret}}$. On construit alors la machine H qui accepte $\{< M > \mid M(< M >) = \perp\}$ comme suit : H commence par dupliquer $< M >$, écrivant $< M, < M >>$, puis applique $\overline{M_{arret}}$. Si $\overline{M_{arret}}$ est sur le point de s'arrêter en **reject**, H entre dans un état spécial, où H se déplace indéfiniment vers la droite sans rien faire (et donc ne s'arrête pas).

$< H > \in L(H)$ si et seulement si $H(< H >) = \perp$. Mais, comme H ne rejette aucun mot, $H(x) = \perp$ si et seulement si $x \notin L(H)$. Donc $H(< H >) = \perp$ si et seulement si $< H > \notin L(H)$. Absurde. Il n'existe donc aucune machine de Turing qui accepte $\overline{L_{arret}}$.

6.7 Réductions

D'une manière générale, on posera les problèmes de la façon suivante :

Donnée : $D \in S$

Question : Q

où $Q \subseteq S \subseteq \Sigma^*$. Il s'agit d'une façon d'écrire l'ensemble $\{D \in S \mid Q\}$ où Q est cette fois vu comme un prédicat. On se passe le plus souvent de parler du codage de la donnée. Par exemple, on écrit

Donnée : M, w où M est une machine de Turing et $w \in \{0,1\}^*$

Question : M s'arrête-t-elle sur w ?

est indécidable.

Cet énoncé est une façon de dire que $\{< M, w > \mid M(w) \neq \perp\}$ n'est pas récursif.

Pour montrer qu'un problème, donné sous cette forme, est indécidable, on procède souvent par *réduction* d'un problème connu pour être indécidable : Si P_1 est le problème

Donnée : $D_1 \in S_1$

Question : Q_1

est indécidable, et P_2 est le problème

Donnée : $D_2 \in S_2$

Question : Q_2

Pour prouver que P_2 est indécidable, il suffit d'exhiber une fonction récursive $f : S_1 \to S_2$ telle que, pour toute donnée $D_1 \in S_1$, $D_1 \in Q_1 \iff f(D_1) \in Q_2$. Voici un premier exemple

Proposition 6.7.1 *Le problème suivant est indécidable :*

L'arrêt universel :

Donnée *: une machine de Turing M*

Question *: M s'arrête-t-elle sur toutes les données ?*

On réduit le problème de l'arrêt : on considère la fonction f qui à $< M, x >$ associe $< M_x >$ où M_x est la machine qui écrit x puis simule M sur x (sans tenir compte de l'entrée). f est une fonction calculable : il suffit en effet d'ajouter$|x|+2$ états à M, qui permettent d'effacer le ruban et d'écrire x. De plus, M_x s'arrête sur toute entrée si et seulement si M s'arrête sur x.