Corrigé de Calculabilité

Cours

1. La « thèse de Church », qui est une conjecture, exprime le fait que tous les modèles de calcul sont équivalents deux à deux. On fait référence ici aux modèles de calcul « suffisamment » puissants, permettant de réaliser des boucles non bornées. Cette conjecture permet de choisir un modèle de calcul particulier quand on veut montrer un résultat de calculabilité. On peut choisir le modèle le plus adapté au problème.

   Cette conjecture semble être infirmée par un résultat récent de Tien Kieu qui a proposé un algorithme quantique pour résoudre le dixième problème de Hilbert (équivalent au problème de l'arrêt). Voir `arxiv.org/abs/quant-ph/0110136`. Ce résultat donne lieu à de nombreuses controverses (chercher `kieu hilbert` avec un moteur de recherche).
2. Une propriété indécidable est un calcul, à résultat vrai ou faux, non faisable par une machine. L'exemple le plus classique est le problème de l'arrêt d'une machine de Turing : il n'existe pas de programme qui prendrait en argument une machine de Turing et qui répondrait *oui* si la machine s'arrête (pour une entrée fixée) et *non* si la machine ne s'arrête pas.
3. Quatre exemples de modèles de calcul :
   (a) Les machines de Turing
   (b) Le lambda-calcul
   (c) Les fonctions récursives (constantes, projections, composition, récurrence, schéma μ-borné)
   (d) Un langage de programmation généraliste (on peut préférer un langage sémantiquement simple comme un langage fonctionnel à un langage *touffu* comme C ou Ada).

λ-calcul

1. On peut représenter un entier positif n par la paire $(n, 0)$. On obtient donc le combinateur suivant

$$\mathcal{Z} \quad = \quad \lambda n.\lambda p.(p\ n\ \lambda f x.x)$$

2. Il « suffit » d'écrire :

$$\begin{aligned}(n_1, n_2) \times (n'_1, n'_2) &= (n_1 - n_2)(n'_1 - n'_2) \\ &= n_1 n'_1 + n_2 n'_2 - n_2 n'_1 - n_1 n'_2 \\ &= (n_1 n'_1 + n_2 n'_2, n_2 n'_1 + n_1 n'_2)\end{aligned}$$

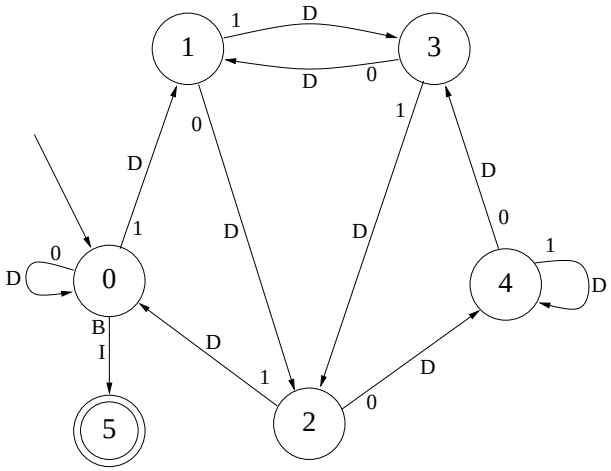
   D'où le combinateur de multiplication pour les entiers de Church relatifs :

$$\bar{\times} \quad = \quad \lambda p_1 p_2.(\lambda n_1 n_2 n'_1 n'_2.\ <\bar{+}\ (\bar{*}\ n_1\ n'_1)\ (\bar{*}\ n_2\ n'_2), \bar{+}\ (\bar{*}\ n_1\ n'_2)\ (\bar{*}\ n_2\ n'_1)>\ (fst\ p_1)\ (snd\ p_1)\ (fst\ p_2)\ (snd\ p_2))$$

   où $\bar{+}$ et $\bar{*}$ sont les combinateurs d'addition et de multiplication sur les entiers de Church naturels, $<,>$ est le constructeur de paire et fst et snd les accesseurs aux éléments d'une paire.

Machines de Turing

L'indication donne l'algorithme à suivre : parcourir le nombre de gauche à droite tout en *mémorisant* le reste modulo 5 de la partie haute du nombre. Cette mémorisation peut s'effectuer simplement avec l'automate : on associe un état à chaque reste possible. On obtient une machine de Turing à 5 états numérotés par les valeurs des restes possibles dont les transitions sont données par les propriétés sur les modulos : si n est la partie gauche à k bits d'un nombre N, quand on regarde le bit $k+1$, si c'est un zéro alors la partie gauche à $k+1$ bits de N vaut $2n$ sinon elle vaut $2n+1$.



Décidabilité

1. La propriété de normalisabilité (!) en temps fini est décidable. On peut en donner une preuve constructive : considérons un programme qui cherche un redex (une application dont le membre gauche est une abstraction) dans un lambda-terme et qui effectue la bêta-réduction et qui laisse le terme inchangé si ce dernier ne contient pas de redex. Ce programme s'exécute en temps fini. On appelle ensuite ce programme 42 fois (avec une boucle bornée) et on observe le résultat : s'il contient encore un redex, alors on retourne *non* sinon on retourne *oui*.

   De façon générale, toutes les propriétés *finies* sont décidables : il suffit d'essayer tous les cas.