Numéro anonymat : **← CE N'EST PAS votre numéro d'étudiant**

Calculabilité/Complexité – 22/01/2021

Lire ATTENTIVEMENT les questions. Il est possible d'admettre des réponses pour ne pas rester bloqué dans un problème. Répondre dans le cadre. Écrire au stylo (pas de crayon). Seule une feuille A4 manuscrite de documents est autorisée.

1 Fonctions récursives

On se propose de regarder partiellement le codage du fonctionnement d'une machine de Turing simple (une bande, alphabet binaire `0`, `1`) à l'aide de fonctions récursives.

Question 1. On suppose connues l'addition *add*, la soustraction tronquée *minus* et une comparaison $<$ qui retourne 0 quand elle est vérifiée. Proposer des fonctions récursives pour les opérations suivantes :

1. La multiplication par deux : *times2*,
2. Le quotient de la division entière par deux : *div2*,
3. Le reste de la division entière par deux : *mod2*.

-
-
-

On suppose récursives les constructions de triplets et couples (qui sont donc des entiers), elles seront notées dans la suite à l'aide de crochets. On se donne également les projections sur le premier, le deuxième et le troisième élément des n-uplets ainsi construits : respectivement $P_1,\ P_2,\ P_3$.
Par exemple : $P_2([a, b, c]) = b,\ P_3([a, b, c]) = c,\ P_1([a, b]) = a$.

On associe naturellement la valeur 0 au caractère `0` et la valeur 1 au caractère `1`. La configuration d'une machine simple dont la bande contient le mot w_1w_2, dont la tête pointe sur le caractère à l'extrême gauche de w_2 et dans l'état q est notée (w_1, q, w_2). On peut représenter cette configuration par un entier $[n_1, n_q, n_2]$ où n_q est un entier représentant l'état q, n_1 est un entier représentant w_1 et n_2 est un entier représentant w_2 lu de droite à gauche (penser à une façon de voir la bande proche de la partie 3 du contrôle de décembre).

Question 2. Proposer une fonction récursive qui sur la donnée d'une configuration retourne le caractère pointé.

On suppose qu'une règle de transition $R = (\alpha, q) \to (\mathtt{1}, q', \blacktriangleright)$ s'applique à une configuration (w_1, q, w_2) codée par $[n_1, n_q, n_2]$ (on ne s'occupe pas des tests d'applicabilité). On cherche une fonction récursive $\mathcal{R}$ codant l'action de R.

Question 3. Plus précisément on a dans ce cas : $\mathcal{R}([n_1, n_q, n_2]) = [f_1([n_1, n_q, n_2]), f_2([n_1, n_q, n_2]), f_3([n_1, n_q, n_2])]$. Proposer les expressions des fonctions f_1 et f_3.

-
-

2 λ-calcul pur

On rappelle l'opérateur de point fixe de Turing : $(Z\ Z)$ où $Z \equiv \lambda y.\lambda x.(x\ ((y\ y)\ x))$. On suppose un codage arbitraire des entiers (on les notera en base 10) et d'une fonction successeur S. On donne dans ce cadre un test à 0, T_z, tel que $(T_z\ n)$ se β-réduit en *vrai* (la première projection) si $n = 0$ et en *faux* (la deuxième projection) sinon.

Question 4. Vérifier que (ZZ) est un opérateur de point fixe puis proposer deux λ-termes codant respectivement l'addition et la multiplication des entiers (attention on **n'est pas** en train de travailler avec des entiers de Church).

Point fixe :

ADD

MULT

Question 5. Calculer $((\text{ADD}\ 2)\ 1)$ en soulignant à chaque étape le rédexe choisi [1].

3 λ-calculs typés

3.1 Types simples

Question 6. Montrer par induction que pour tous types simples t et t', et tous termes U et V, si $U : t$ et $V : t'$ alors pour x variable de type t' on a $U[x \mapsto V] : t$.

-
-
-
-

3.2 Système T

Question 7. (Bonus) On a vu en cours que toutes les fonctions exprimables dans le système T de Gödel étaient totales (les calculs s'arrêtent toujours). On a aussi énoncé que des fonctions non primitives pouvaient y être représentées. On est donc plus expressif avec T qu'avec des fonctions récursives primitives. Expliquer rapidement pourquoi **on ne peut pas** représenter *toutes* les fonctions récursives totales dans le système T.

1. On pourra choisir $((\text{ADD}\ 1)\ 2)$ si c'est plus court pour l'addition codée à la question 4.

Numéro anonymat : ← **CE N'EST *toujours* PAS votre numéro d'étudiant**

4 NP-complétude

Question 8. Soit P_1 un problème NP-complet. Proposer une méthode faisant intervenir P_1 pour montrer qu'un autre problème P_2 est NP-complet.

-
-

Question 9. TRIPARTITE MATCHING est un problème **NP-complet** qui a pour données :

— Trois ensembles[2] finis B, G et H de même cardinal m,

— Un ensemble $T \subseteq B \times G \times H$

et pour question :

— Existe-t-il un ensemble $D \subseteq T$ tel que

  — Les triplets dans D n'ont pas de composantes communes deux à deux et

  — Chacun des éléments de B, G et H apparaît dans un triplet ?

On considère le problème EXACT COVER BY 3-SETS qui a pour données :

— Un ensemble fini U de cardinal $3m$,

— Une famille $F = \{S_1, \ldots, S_k\}$ telle que pour tout i : $S_i \subseteq U$ et est de cardinal 3

et pour question :

— Existe-t-il m ensembles *disjoints* dans F dont l'union est U ?

Ce problème est dans NP, montrer qu'il est NP-complet.

2. Historiquement on les note B, G et H pour *Boys*, *Girls* et *Homes*.

Constantes

$$C_{k,c} \in \mathcal{F}_k, \quad C_{k,c}(x_1, \ldots, x_k) = c$$

Successeur

$$S \in \mathcal{F}_1, S(x) = x + 1$$

Projections

$$\pi_{k,i} \in \mathcal{F}_k, \quad \pi_{k,i}(x_1, \ldots, x_i, \ldots, x_k) = x_i$$

Schéma de *Composition* :
— f à n arguments
— $g_1, \ldots, g_n$ à m arguments
$\rightsquigarrow$ h à m arguments

$$h(x_1, \ldots, x_m) = f(g_1(x_1, \ldots, x_m), \ldots, g_n(x_1, \ldots, x_m))$$

$h = \mathit{Comp}_{n,m}(f, g_1, \ldots, g_n)$

Schéma de *récursion primitive* :
— b à n arguments
— h à $n+2$ arguments
$\rightsquigarrow$ f à $n+1$ arguments

$$\begin{array}{rcl} f(0, x_1, \ldots, x_n) & = & b(x_1, \ldots, x_n) \\ f(k+1, x_1, \ldots, x_n) & = & h(k, x_1, \ldots, x_n, \underbrace{f(k, x_1, \ldots, x_n)}) \end{array}$$

$f = \mathit{Rec}(b, h)$

Schéma de *minimisation* :
— g à $n+1$ arguments
$\rightsquigarrow$ f à n arguments

$$f(x_1, \ldots, x_n) = \min\{k \mid g(k, x_1, \ldots, x_n) = 0\}$$

$f = \mathit{Min}(g)$

Règles de typage simple du λ-calcul

Variable	$\dfrac{x : t \in \Gamma}{\Gamma \vdash x : t}$	(première occurrence)
Abstraction	$\dfrac{\{x : t_1\} \cup \Gamma \vdash E : t_2}{\Gamma \vdash \lambda x.E : t_1 \to t_2}$	
Application	$\dfrac{\Gamma \vdash E_1 : t_1 \to t_2 \quad \Gamma \vdash E_2 : t_1}{\Gamma \vdash (E_1\ E_2) : t_2}$	