Numéro anonymat :

Calculabilité/Complexité – 21/01/2022

Lire les questions. Répondre dans le cadre. Écrire au stylo (pas de crayon). Une feuille A4 manuscrite autorisée de documents.

Fonctions récursives de Gödel

Question 1. On suppose connues l'addition *add*, la soustraction tronquée *minus*. On décide que le booléen `false` est codé par la constante 0 et que `true` est codé par 1.

Proposer des fonctions récursives de Gödel pour les opérations suivantes :

1. La fonction *norm* qui retourne 0 si elle a 0 en paramètre et 1 sinon.
2. La négation logique *not*,
3. La conjonction logique *and*,
4. Le test d'égalité de deux entiers *equal*. *Exemple :* $\mathrm{equal}(2,3)$ *et* $\mathrm{equal}(3,2)$ *valent tous deux* `false`.

Question 2. En supposant que la multiplication et la comparaison $>$ (retournant 0 pour "vrai") de deux entiers sont récursives primitives, montrer que la fonction retournant la racine cubique entière d'un entier est récursive.

λ-calcul pur, listes

On se donne dans toute cette partie des entiers natifs (pas de Church) en notation décimale, munis de l'addition add et de la multiplication $mult$.

Question 3. On rappelle l'opérateur de point fixe de Turing : $(Z\ Z)$ où $Z \equiv \lambda y.\lambda x.(x\ ((y\ y)\ x))$.
Vérifier que $(Z\ Z)$ est un opérateur de point fixe.

Question 4. Proposer une fonction `sq` qui à x associe x^2.

Question 5. On rappelle qu'on peut coder le constructeur de paire en λ-calcul par une fonction qui prend les deux éléments à stocker et attend la projection qu'on voudra en faire. Donner l'expression en λ-calcul pur du constructeur de paire et des première et deuxième projections (appelées respectivement VRAI et FAUX).

Si une paire est caractérisée par l'attente d'une fonction d'accès à son contenu, une liste est différente d'une composition de paires en ce qu'elle peut être vide. Son utilisation attend en particulier deux choses :

1. Une valeur à retourner quand elle est vide et
2. Une fonction accédant à son contenu lorsqu'elle n'est pas vide (sur le modèle des paires).

On peut donc définir une liste comme une fonction à *deux* paramètres.

La liste vide nil est définie comme VRAI (et on remarque que son deuxième paramètre est inutilisé).

Le constructeur de liste non vide $cons$ attend, lui, deux valeurs à stocker (un élément et une liste) et retourne une fonction attendant un paramètre inutilisé (on se moque dans ce cas de la valeur à vide) et une fonction de projection généralisée utilisant l'élément et la liste stockés : $cons \equiv \lambda t.\lambda q.\lambda d.\lambda p.((p\ t)\ q)$.

Le premier paramètre formel correspond à la tête, c'est-à-dire à l'élément qu'on ajoute, le deuxième à la queue, c'est-à-dire la liste à laquelle on ajoute l'élément. Il reste bien une fonction à deux paramètres formels : d qui ne sert pas et p qui est appliqué à l'élément et la queue (et peut donc les utiliser).

La liste comportant les éléments 3, 2 et 1 rentrés dans cet ordre (et donc avec 1 en tête) peut ainsi être construite par : $((cons\ 1)\ ((cons\ 2)\ ((cons\ 3)\ nil)))$.

On remarque qu'une liste se comporte comme son propre test de vacuité : si elle est vide la valeur retournée est son premier paramètre, sinon elle applique son deuxième paramètre à sa tête et à sa queue.

Question 6. Proposer une fonction phd qui sur la donnée d'une liste comportant des entiers retourne le premier élément (en tête) de celle-ci ou 0 si cette liste est vide. **Ne PAS utiliser cette fonction dans la suite.**

Numéro anonymat :

Question 7. Proposer une fonction $suml$ qui somme les carrés des éléments d'une liste dont tous les éléments sont des entiers. En particulier $(suml\ nil)$ se réduit en 0 et $(suml\ ((cons\ 2)\ ((cons\ 3)\ nil)))$ se réduit en 13.

Question 8. Proposer une fonction $long$ qui calcule la longueur d'une liste.
Détailler la réduction de $(long\ ((cons\ 3)\ nil))$.

Il reste une page d'énoncé…

NP-complétude

Question 9. 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 10. Le problème EXACT COVER BY 3-SETS a pour données :
— Un ensemble fini U de cardinal $3m$,
— Une famille $F = \{S_1, \dots, S_k\}$ 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 ?
Il est **NP-complet**.

On considère le problème SET PACKING qui a pour données :
— Un ensemble fini V,
— Une famille $G = \{T_1, \dots, T_l\}$ telle que pour tout j : $T_j \subseteq V$,
— Un entier k
et pour question :
— Existe-t-il k ensembles *disjoints* dans G ?
Ce problème est dans NP, montrer qu'il est NP-complet.

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

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

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