DM 1 – à rendre le 8 octobre

Le soin accordé à la rédaction sera pris en compte dans l'évaluation de la copie.

Exercice 1.
Justifier de manière détaillée que la fonction qui à n associe le $(n+1)$-ème nombre pair s'écrivant comme somme de deux nombres premiers est récursive primitive.

Exercice 2.
Soit $\mathcal{C}$ le plus petit sous-ensemble de $\mathcal{F}$ qui contient les projections, les fonctions constantes, la fonction successeur, est stable par composition et par *itération*, c'est-à-dire que si $f_1, ..., f_p \in \mathcal{C} : \mathbb{N}^p \to \mathbb{N}$ alors les fonction $g_1, ..., g_p : \mathbb{N}^{p+1} \to \mathbb{N}$ définies par

$$\begin{cases} g_i(x_1, ..., x_p, 0) &= x_i \\ g_i(x_1, ..., x_p, y+1) &= f_i(g_1(x_1, ..., x_p, y), ..., g_p(x_1, ..., x_p, y)) \end{cases}$$

sont dans $\mathcal{C}$. Montrer que $\mathcal{C}$ est égal à l'ensemble des fonctions récursives primitives.

Problème 3.
On va définir une variante des machines RAM, les machines PRAM (Primitive Random Access Memory) et chercher à comprendre ce qu'elles calculent. Une machine PRAM a un nombre $p \in \mathbb{N}$ de registres et prend un nombre $q \in \mathbb{N}$ d'arguments. Elle possède également un accumulateur où les calculs seront effectués.
Une *configuration* d'une machine PRAM à p registres et q arguments est la donnée d'un quadruplet

$$((x_1, ..., x_q), ((r_{i,n})_{n\in\mathbb{N}})_{i=1}^p, a, s)$$

où $x_1, ..., x_q, r_{i,n}, a, s \in \mathbb{N}$ et tous les $r_{i,n}$ sont nuls sauf un nombre fini d'entre eux.
Les x_k sont les arguments, les $r_{i,n}$ sont le contenu des registres, a est le contenu de l'accumulateur (où les calculs sont effectués) et enfin s est le contenu de la sortie. La *configuration initiale* d'une machine PRAM d'arguments $(x_1, ..., x_p)$ est $((x_1, ..., x_p), ((0)_{n\in\mathbb{N}})_{i=1}^p, 0, 0)$.
Une *instruction* est une chaîne de caractères et d'entiers de la forme suivante

- `load`$\#k$ (mettre l'entier k dans l'accumulateur)
- `load`$_j k$ (mettre le contenu de $r_{j,k}$ dans l'accumulateur)
- `load`$_j(k)$ (mettre le contenu de $r_{j,r_{j,k}}$ dans l'accumulateur)
- `store`$_j k$ (enregistrer le contenu de l'accumulateur dans $r_{j,k}$)
- `store`$_j(k)$ (enregistrer le contenu de l'accumulateur dans $r_{j,r_{j,k}}$)
- `incr` (incrémente l'accumulateur)
- `decr` (décrémente l'accumulateur ; s'il est nul, ne fait rien)
- `do` (début de boucle effectuée autant de fois que le contenu de l'accumulateur)
- `enddo` (fin de la boucle `do`)
- `read`$_k$ (mettre la k-ème entrée x_k dans l'accumulateur)
- `write` (mettre le contenu de l'accumulateur dans la sortie)

On définit par induction un *programme* comme une suite d'instructions :

- La suite vide est un programme.
- Si P est un programme, alors P suivi d'une instruction de la forme `load`$_j\#n$, `load`$_j n$, `load`$_j(n)$, `store`$_j n$, `store`$_j(n)$, `incr`, `decr`, `read`$_k$, `write` est un programme.
- Si P et Q sont des programmes, alors P suivi de `do`, puis de Q puis de `enddo` est un programme.

On définit par induction le résultat d'un programme sur une configuration, qui est une nouvelle configuration.

- Le programme vide ne modifie pas la configuration
- Si P est un programme dont le résultat est $((x_1, ..., x_p), ((r_{j,n})_{n\in\mathbb{N}})_{j=1}^p, a, s)$ et P est suivi de
  - `load#`k la configuration résultante est $((x_1, ..., x_p), ((r_{i,n})_{n\in\mathbb{N}})_{i=1}^p, k, s)$
  - `load`$_j k$ la configuration résultante est $((x_1, ..., x_p), ((r_{i,n})_{n\in\mathbb{N}})_{i=1}^p, r_{j,k}, s)$
  - `load`$_j(k)$ la configuration résultante est $((x_1, ..., x_p), ((r_{i,n})_{n\in\mathbb{N}})_{i=1}^p, r_{j,r_{j,k}}, s)$
  - `store`$_j k$ la configuration résultante est $((x_1, ..., x_p), ((r'_{i,n})_{n\in\mathbb{N}})_{j=1}^p, a, s)$ avec $r'_{i,n} = r_{i,n}$ sauf pour $i = j$ et $n = k$ auquel cas $r'_{j,k} = a$
  - `store`$_j(k)$ la configuration résultante est $((x_1, ..., x_p), ((r'_{i,n})_{n\in\mathbb{N}})_{j=1}^p, a, s)$ avec $r'_{i,n} = r_{i,n}$ sauf pour $i = j$ et $n = r_{j,k}$ auquel cas $r'_{j,r_{j,k}} = a$
  - `incr` la configuration résultante est $((x_1, ..., x_p), ((r_{i,n})_{n\in\mathbb{N}})_{i=1}^p, a + 1, s)$
  - `decr` la configuration résultante est $((x_1, ..., x_p), ((r_{i,n})_{n\in\mathbb{N}})_{i=1}^p, a \dot{-} 1, s)$
  - `read`$_k$ la configuration résultante est $((x_1, ..., x_p), ((r_{i,n})_{n\in\mathbb{N}})_{i=1}^p, x_k, s)$
  - `write` la configuration résultante est $((x_1, ..., x_p), ((r_{i,n})_{n\in\mathbb{N}})_{i=1}^p, a, a)$
- Si le programme est de la forme P, `do`, Q, `enddo`, la configuration résultante est obtenue en appliquant une fois le programme P à la configuration $((x_1, ..., x_p), ((r_{j,n})_{n\in\mathbb{N}})_{j=1}^p, a, s)$ puis, en notant $((x'_1, ..., x'_p), (r'_{j,n})_{n\in\mathbb{N}})_{j=1}^p, a', s')$ la configuration obtenue, en itérant a' fois le programme P sur cette dernière (en particulier si $a' = 0$, on ne modifie pas la configuration obtenue).

Chaque programme P définit une fonction $f_P : \mathbb{N}^p \to \mathbb{N}$ donnée par $f(x_1, ..., x_p) = s$ où $((x_1, ..., x_p), ((r_{i,n})_{n\in\mathbb{N}})_{i=1}^p, a, s)$ est le résultat de l'exécution du programme P sur la configuration initiale d'arguments $(x_1, ..., x_p)$. Une telle fonction est dite *PRAM-exécutable*.

1. Montrer que les fonctions projection, successeur et constantes sont PRAM-exécutables (on se contentera d'écrire les programmes correspondants).
2. Montrer que la fonction somme est PRAM-exécutable.
3. Montrer que la fonction $(j, n) \mapsto pn + j + 1$ est PRAM-exécutable.
4. Montrer que toute fonction PRAM-exécutable est PRAM-exécutable par une machine PRAM avec un seul registre.
5. Montrer que toute fonction récursive primitive est PRAM-exécutable.
6. On travaille désormais avec un seul registre (ce qui ne change pas l'ensemble des fonctions PRAM-exécutables d'après la question 4). Donner un codage récursif primitif des configurations.
7. Montrer par induction que pour tout programme P, la fonction qui au code d'une configuration C associe le code de la configuration C' résultante du programme P appliqué à C est récursive primitive.
8. Conclure que toute fonction est PRAM-exécutable ssi elle est récursive primitive.

Questions bonus

9. Coder les programmes par des entiers de sorte que l'ensemble C des entiers correspondant à un programme à un argument soit récursif primitif. (On ne justifiera pas que C est effectivement récursif primitif).
10. Montrer que la fonction f qui à (i, x) associe le résultat de l'exécution du programme de code i sur x si $i \in C$, sinon 0 est récursive.
11. En utilisant un argument diagonal, montrer que f n'est pas récursive primitive.
12. Montrer que la fonction qui à un code de programme associe son temps d'exécution n'est pas récursive primitive.