IFT2105–Introduction à l'informatique théorique
Été 2024, (Devoir #1)

Louis Salvail

Université de Montréal (DIRO), QC, Canada
salvail@iro.umontreal.ca
Bureau: Pavillon André-Aisenstadt, #3369

1 Remise

Il s'agit du premier devoir pour le cours. La date de remise est:

Vendredi, 28 mai 2024, 9h30, aucun retard ne sera toléré.

Vous pouvez faire votre devoir en équipe de deux au maximum. Votre démo apprécie les devoirs remis en LaTeX. Un boni de 10% est donné si vous remettez un devoir produit par un traitement de texte. LaTeX est de loin le meilleur pour écrire des textes scientifiques (note: LaTeX a été écrit par Donald Knuth, le même qui a introduit les flèches qui portent son nom). Vous remettez votre devoir sur Studium en un seul fichier PDF. Une seule remise par équipe.

2 Questions

1. Donnez les programmes `RÉPÉTER` qui permettent de calculer les opérations suivantes sur des registres qui contiennent des nombres rationnels (positifs ou négatifs) plutôt que des entiers naturels. Vous pouvez supposer que vous disposez de la procédure (ou marco) $\mathsf{pgcd}(r_1, r_2)$ qui retourne le PGCD des entiers rangés dans les registres r_1 et r_2. Avant de donner votre code, expliquez comment vous représentez les nombres rationnels. Ensuite, donnez un programme `RÉPÉTER` pour chacune des macros suivantes. Vous devez expliquer (brièvement) comment chaque programme fonctionne. Vous pouvez donner les macros dans l'ordre que vous voulez pour pouvoir les réutiliser pour les prochaines macros. Cet ensemble de macros permet d'opérer sur les nombres rationnels d'une façon complète.

   $\mathsf{Q0}$: retourne dans r_0 le nombre rationnel 0.

   $\mathsf{Q}(r_1, r_2) = r_1/r_2$: retourne dans r_0 le nombre rationnel r_1/r_2 où r_1 et r_2 sont interprétés comme des entiers naturels. Notez que tel que défini, les nombres rationnels construits avec $\mathsf{Q}(r_1, r_2)$ sont toujours positifs.

   $\mathsf{pm}(r_1) = -r_1$: retourne ans r_0, le nombre rationnel r_1 de signe inversé. Cette procédure permet de changer le signe du nombre rationnel rangé dans r_1. Lorsqu'elle est utilisée à la suite $\mathsf{Q}(\cdot, \cdot)$, elle permet de produire des nombres rationnels négatifs.

   $\mathsf{inv}(r_1) = 1/r_1$: pour r_1 contenant un nombre rationnel, retourne dans r_0 le nombre rationnel $1/r_1$.

num(r_1): retourne dans r_0 la valeur absolue du numérateur du nombre rationnel rangé dans r_1.

dénom(r_1): retourne dans r_0 la valeur absolue du dénominateur du nombre rationnel rangé dans r_1.

neg?$(r_1) = (r_1 < 0)$: retourne `vrai` dans r_0 si le nombre rationnel dans r_1 est tel que $r_1 < 0$ et retourne `faux` dans r_0 sinon.

égal?$(r_1, r_2) = (r_1{=}r_2)$: retourne $r_0 =$ `vrai` si le nombre rationnel rangé dans r_1 est égal au nombre rationnel rangé dans r_2. Sinon, la macro retourne $r_0 =$ `faux`.

pg?$(r_1, r_2) = (r_1{>}r_2)$: retourne $r_0 =$ `vrai` si le nombre rationnel rangé dans r_1 est plus grand que le nombre rationnel rangé dans r_2. Sinon, la macro retourne $r_0 =$ `faux`.

pge?$(r_1, r_2) = (r_1{\geq}r_2)$: retourne $r_0 =$ `vrai` si le nombre rationnel rangé dans r_1 est plus grand ou égal au nombre rationnel rangé dans r_2. Sinon, la macro retourne $r_0 =$ `faux`.

add$(r_1, r_2) = r_1 + r_2$: retourne $r_0 = r_1 + r_2$, le nombre rationnel obtenu en additionnant les nombres rationnels rangés dans r_1 et r_2.

mult$(r_1, r_2) = r_1 \cdot r_2$: retourne $r_0 = r_1 \cdot r_2$, le nombre rationnel obtenu en multipliant les nombres rationnels rangés dans r_1 et r_2.

plaf$(r_1) = \lceil r_1 \rceil$: retourne dans r_0 le plus petit entier naturel plus grand ou égal au nombre rationnel rangé dans r_1.

planc$(r_1) = \lfloor r_1 \rfloor$: retourne dans r_0 le plus grand entier naturel plus petit ou égal au nombre rationnel rangé dans r_1.

2. L'exercice précédent semble indiquer que nous aurions pu définir un langage comme le langage RÉPÉTER, mais où les registres ne contiennent que des nombres rationnels. Croyez-vous qu'il soit possible de définir le langage RÉPÉTER$_{\mathbb{Q}}$ dont les registres ne contiennent que des nombres rationnels. Comme pour le langage RÉPÉTER, les registres contiennent initialement la valeur 0. L'instruction $r_i \leftarrow r_j$ devrait également être une instruction RÉPÉTER$_{\mathbb{Q}}$. Les boucles `répéter` du langage RÉPÉTER$_{\mathbb{Q}}$ peuvent être facilement adaptées au cas où les registres contiennent des nombres rationnels. Il suffit de définir l'instruction

   `répéterplanc` r_i `fois` [⟨BLOC⟩]

   qui indique que le bloc d'instructions est exécuté $||\lfloor r_i \rfloor||$ fois (la valeur absolue du plancher de r_i fois). Nous pourrions également en avoir une autre qui répète le bloc $||\lceil r_i \rceil||$ fois. Maintenant, pouvez-vous ajouter quelques instructions qui permettent au langage RÉPÉTER$_{\mathbb{Q}}$ d'opérer complètement et seulement avec des nombres rationnels rangés dans les registres? En particulier, pour chaque nombre rationnel $q \in \mathbb{Q}$, il doit exister un programme RÉPÉTER$_{\mathbb{Q}}$ qui retourne q dans r_0 de la même façon que pour chaque $n \in \mathbb{N}$, il y a un programme RÉPÉTER qui produit n dans r_0. Si vous répondez oui alors les instructions supplémentaires que vous donnez devraient être peu nombreuses. Si vous répondez non alors expliquez clairement pourquoi il n'est pas possible de définir un ensemble d'instructions capable d'opérer complètement dans $\mathbb{Q}$. Contrairement à l'exercice précédent, les entiers naturels ne sont pas utilisés pour produire des nombres rationnels, mais plutôt des instructions du langage (comme les entiers naturels en RÉPÉTER qui sont générables par l'instruction `inc`(r_i), une conséquence de l'axiomatisation de l'arithmétique de Peano).

3. La fonction $B_i(x)$ augmente à très grande vitesse en fonction de x, même pour de petites valeurs de i. Pour $i = 4$, la croissance est déjà monstrueuse. En effet, montrez que pour $x \geq 1$,

$$B_4(x) \geq \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{x+3 \text{ fois}} - 3 \ ,$$

où le nombre de 2 empilés est $x + 3$. Votre démonstration doit procéder par induction mathématique. L'inégalité est en fait une égalité.

4. Définissons un nouveau type de programme, les programmes `RÉPÉTERPASTROP`. Les programmes `RÉPÉTERPASTROP` sont identiques aux programmes `RÉPÉTER` à l'exception des boucles `RÉPÉTER` qui ne peuvent pas être imbriquées dans un programme `RÉPÉTERPASTROP`. Trouvez le plus petit $k \in \mathbb{N}$ tel que n^k ne peut pas être calculé par un programme `RÉPÉTERPASTROP` à partir de n'importe quel input n. Prouvez votre réponse à l'aide de ce qui a été vu en classe et en démo. Pour y parvenir, montrez que pour chaque programme `RÉPÉTERPASTROP`, il existe une valeur n_0 (qui dépend de la taille du programme) telle que pour chaque $n > n_0$, n^k ne peut pas être calculé par le programme. Ainsi, vous pouvez conclure qu'il n'existe pas de programme `RÉPÉTERPASTROP` qui puisse calculer la fonction n^k pour chaque $n \in \mathbb{N}$ donné en input.
5. Les programme `RÉPÈTEIMBRIQUEPASTROP` sont comme les programmes `RÉPÉTER`, mais la façon d'imbriquer les boucles `répéter` dans les programmes `RÉPÈTEIMBRIQUEPASTROP` est limitée. Intuitivement, vous pouvez imbriqués les boucles les unes dans les autres, mais les bornes de chacune des boucles `répéter` demeurent statiques pendant toute l'exécution de la boucle extérieure. Répondez à la question suivante Montrez que les programmes `RÉPÈTEIMBRIQUEPASTROP` ne calculent pas toutes les fonctions calculables par un programme `RÉPÉTER`, mais peuvent calculer des fonctions qui ne sont pas calculables par les programmes `RÉPÉTERPASTROP`.