Série d'exercices #$11\frac{1}{2}$

IFT-1215

April 3, 2015

6.15

Prendre le point le vue d'un compilateur (pour un langage de style Lisp).

Pour traduire une *expression* e telle que $(\geq\ x\ (+\ y\ 1))$, qui teste si la variable x est plus grande ou égale à $y + 1$, en un morceau de code LMC, un compilateur va définir une fonction qui prend une *sous-expression* et renvoie sa traduction en code LMC de sorte que l'exécution du code laisse le résultat dans l'accumulateur.

1. Montrer quel code pourrait être généré pour la sous-expression 1.
2. Montrer quel code pourrait être généré pour la sous-expression x.
3. Montrer les instructions LMC que devrait générer un compilateur pour traduire l'expression entière $(+\ e_1\ e_2)$ qui renvoie la somme des expressions entières e_1 et e_2. On présume pour cela que le reste du compilateur a déjà converti e_1 en une séquence d'instructions I_1 qui laisse le résultat dans l'accumulateur, et de même avec e_2 qui a été compilé en I_2.
4. Même chose pour $(\geq\ e_1\ e_2)$ qui est une expression booléenne qui renvoie vrai ssi e_1 est plus grand ou égal à e_2.
5. Même chose pour l'expression conditionnelle (`if` $e\ i_1\ i_2$) où e est une expression booléenne selon laquelle l'une ou l'autre des séquences d'instructions i_1 ou i_2 est exécutée.
6. Même chose pour (`while` $e\ i$) qui exécute les instructions i tant que l'expression booléenne e est vraie.
7. Qu'en est-il d'un appel de fonction $(f\ e_1\ ...\ e_n)$?

Décrire les hypothèses sur lesquelles ces traductions se basent (e.g. quelles contraintes doivent être satisfaites par la traduction de e_1, e_2, ...).