Projet : Seconde Partie

⟨ ⟩ ⟹ ⟨ , ⟩

13 mai 2015

Maximilien Colange

Date de rendu : 2 juin 2015 à 23h55

Fichiers à rendre : deux fichiers :
- le rapport au format PDF `rapport_projet_2_`*nom*`.pdf`
- un fichier contenant le code Prolog `source_projet_2_`*nom*`.pl`

Comme leur nom l'indique, les travaux personnels sont *personnels*. Il est strictement interdit de copier. Des similitudes flagrantes entre deux rendus seront sanctionnées par une note de 0.

Tout dépassement de la date de rendu sera pénalisé. Cette date est donnée en heure locale de Genève.

Le non respect du format de rendu entraînera une pénalité sur la note. La récidive est plus sévèrement punie.

N'oubliez pas d'indiquer vos noms et prénoms dans les fichiers du rendu !

Ce projet comporte deux parties notées séparément. Il s'agit dans cette seconde partie d'étendre le langage défini en première partie. Vous pouvez soit partir de votre première partie, soit vous appuyer sur la correction de cette première partie.

On se réfèrera à l'énoncé de la première partie pour la définition première du langage.

On souhaite étendre ce langage afin de le rendre plus puissant (en fait, plus compact, car il était déjà Turing-complet). On souhaite ajouter :
- la définition de portée via des blocs. Un registre déclaré dans un bloc n'est pas visible à l'extérieur du bloc. Il peut porter le même nom qu'un registre à l'extérieur du bloc : on dit qu'il le masque.
- la déclaration automatique de registre : si un nom de registre est utilisé sans avoir été déclaré, on le suppose déclaré implicitement (avec une valeur par défaut) dans le bloc de portée global (i.e. le plus extérieur). Cette déclaration implicite a lieu au moment de la première utilisation du registre.
- l'ajout d'opérations arithmétiques (on se contentera de l'addition et de la soustraction) : `add(`r_1`, `r_2`, `r_3`)` affecte à r_3 la somme des valeurs de r_1 et r_2 ; `sub(`r_1`, `r_2`, `r_3`)` affecte à r_3 la différence entre la valeur de r_1 et celle de r_2.
- une instruction conditionnelle, sous la forme d'un `if (`$r_1 == r_2$`) then` ... `else` .... Se pose le problème de la gestion correcte du pointeur d'instructions.
- le saut est généralisé : le numéro de ligne peut être spécifié soit par un entier (comme en première partie), soit par (la valeur d') un registre.

```
x = 3
a = 1
begin block
  x = 0
  b = 2
  add(x, a, x) // x vaut alors 1
  begin block
    incr(c)
    jump(x,c,5)
  end block
end block
```

Les lignes `begin block` et `end block` marquent la limite des blocs. Libre à vous d'utiliser une notation alternative. Vous pouvez aussi considérer ces lignes comme non-numérotées (ce ne sont en effet pas des vraies instructions, juste des délimiteurs).

Exercise 1 : Sur papier

Adaptez la syntaxe et la sémantique définie en première partie. Pensez à l'impact éventuel des nouvelles règles sur les anciennes. Pour la gestion des blocs, réfléchissez à une structure de la mémoire adéquate.

N'hésitez pas, quand c'est possible, à traduire les nouvelles constructions avec les anciennes.
Exemple : en C, `for (`a `;` b `;` c`) { `d `}` est en fait synonyme de a`; while (`b`) { `d `; `b `}`. Ainsi, on n'a besoin de décrire que la sémantique du `while`, qui sert aussi à exprimer la sémantique du `for`. Ce procédé permet de simplifier l'écriture de compilateurs : il n'y a qu'une structure de boucle à implémenter, qui gère les deux structures de boucle offertes à l'utilisateur.

Exercise 2 : En Prolog

Implémentez en Prolog la nouvelle sémantique que vous avez décrite. Votre code Prolog prend en entrée un contexte de départ dont la mémoire est vide et le pointeur d'instruction est 1 (on commence à la première ligne). Il doit retourner un autre contexte dont la mémoire associe une valeur à chacune des variables définies dans le programme en entrée, cette valeur devant correspondre à celle obtenue à la fin du programme. Le compteur d'instruction de retour doit alors égal au nombre d'instructions du programme d'entrée plus un.

Dans votre code et votre rapport doit figurer au moins un petit programme de test qui fonctionne (notez que cela implique qu'il doit terminer).