Sémantique des langages de programmation
Mastère PLMT
examen

Yves Bertot

Décembre 2004

1 Définition de langage

Votre travail est de donner l'ensemble des règles d'inférence pour décrire le comportement des différentes constructions d'un langage.

Il s'agit de définir un langage de programmation impératif avec les constructions suivantes (les instructions sont dénotées par des variables $i_1, \ldots, i_n$).

- `if` e `then` i_1 `else` i_2,
- x `:=` e,
- i_1`;`i_2, (séquence)
- l`:loop` i `end`,
- `exit` l,
- `skip`.

Le comportement des trois premières constructions est habituel et correspond aux comportements décrits dans le langage impératif utilisé dans le cours.

Les deux dernières constructions permettent de disposer de boucles avec des sorties multiples. L'exécution d'une construction l`:loop` i `end` répète l'exécution de l'instruction i jusqu'à ce que l'une des instructions exécutées à l'intérieur de i soit une instruction `exit` l.

Bien sûr, dans l'exécution du fragment i_1`;` i_2, il est possible que l'exécution de i_1 contienne l´exécution d'une instruction `exit`, dans ce cas l'instruction i_2 n'est pas exécutée.

Il est possible d'utiliser une instruction `exit` pour sortir d'une boucle plus loin que la boucle immédiatement englobante.

Enfin, si l'on considère l'instruction suivante, elle doit s'exécuter normalement dans l'état $(x, 0) \cdot (y, 0) \cdot (z, 0) \cdot \emptyset$, et terminer normalement dans l'état $(x, 2) \cdot (y, 12) \cdot (z, 4) \cdot \emptyset$.

```
main : loop
  x := x + 1;
  z := 0;
 inner : loop
   z := z + 1;
   y := y + x;
   if 10 < y then
   exit main;
   else if 3 < z then
    exit inner;
   else
     skip;
 loop;
loop;
```

Your work is to give the semantic rules that describe the behavior of a language's constructs. The objective is to define an imperative programming language with the following constructs (variables i_1, ..., i_n range over instructions):

- `if` e `then` i_1 `else` i_2,
- x `:=` e,
- i_1`;`i_2, (sequence)
- l`:loop` i `end`,
- `exit` l,
- `skip`.

The behavior of the first three instructions is usual and corresponds to the one described in the course.

The last two constructs make it possible to have loops with multiple exits. Executing a construct l`:loop` i `end` repeats the instruction i until an `exit` l is encountered.

Of course, when executing a construct i_1`;` i_2, it is possible that executing i_1 reaches an `exit`. In this case, the instruction i_2 is not executed.

It is possible to use an `exit` statement to exit a loop that is further than the closest loop.

Last, executing the program given above from the state $(x, 0) \cdot (y, 0) \cdot (z, 0) \cdot \emptyset$ should terminate normally and return the state $(x, 2) \cdot (y, 12) \cdot (z, 4) \cdot \emptyset$.