Université Bordeaux 1
Licence, module Compilation, 2017/2018

Langage IMP

1 Langage source : syntaxe

Le langage IMP [1], est un langage de programmation impératif très simple, où toutes les variables sont de type entier. Voici une grammaire de ce langage (au format de Bison) :

```
E: E Pl T
 | E Mo T
 | T


T: T Mu F
 | F


F: '(' E ')'
 | I
 | V


C : V Af E
  | Sk
  | '(' C ')'
  | If E Th C El C
  | Wh E Do C
  |  C Se C
```

où les symboles `I V Af Sk Se If Th El Wh Do Pl Mo Mu` sont des unités lexicales, correpondant aux ensembles de lexèmes suivants :
`I` : une suite de chiffres, non-vide, commençant par un chiffre non-nul
`V` : un identificateur de variable

symbole	`Af`	`Sk`	`Se`	`If`	`Th`	`El`	`Wh`	`Do`	`Pl`	`Mo`	`Mu`
lexeme	`:=`	`Skip`	`;`	`if`	`then`	`else`	`while`	`do`	`+`	`-`	`*`

(i.e. ces unités lexicales ne comportent qu'un lexème, qui est donné dans le tableau). On appelle *commande* un mot engendré par le non-terminal C ; une *commande atomique* est une commande qui n'est pas décomposable sous la forme c_1 `Se` c_2 pour des commandes c_1, c_2 ; une *expression* est un mot e engendré par le non-terminal E.

1. qui est dû à J. Goubault-Larrecq, cours de sémantique et compilation, licence 1, ENS Cachan, 2013.

2 Langage source : sémantique

La sémantique de IMP est définie ci-dessous. Le procédé de définition employé s'appelle une sémantique *opérationnelle à petits pas*. Notons V un ensemble de variables [2]. On appelle *environnement* sur V toute application $\rho : V \to \mathbb{Z}$. Pour tout mot w sur l'alphabet $\{0, 1, \ldots, 9\}$, on note $\nu(w) \in \mathbb{N}$ l'entier dénoté par w en base 10. La *valeur* d'une expression e dans un environnement ρ est définie par :

$$[\![x]\!]\rho = \rho(x) \quad \text{pour toute variable } x \in V,$$

$$[\![w]\!]\rho = \nu(w) \text{ pour tout mot } w \in \{1, \ldots, 9\}\{0, 1, \ldots, 9\}^* \cup \{0\},$$

$$[\![e_1 \texttt{ Pl } e_2]\!]\rho = [\![e_1]\!]\rho + [\![e_2]\!]\rho, \quad [\![e_1 \texttt{ Mo } e_2]\!]\rho = [\![e_1]\!]\rho - [\![e_2]\!]\rho, \quad [\![e_1 \texttt{ Mu } e_2]\!]\rho = ([\![e_1]\!]\rho) * ([\![e_2]\!]\rho), \quad [\![(e)]\!]\rho = [\![e]\!],$$

pour toutes expressions e, e_1, e_2.
Une suite de commandes est un mot sur l'alphabet des commandes. On note $\cdot$ le produit de concaténation des suites de commandes. Pour toute variable $x \in V$, expression e, commande atomique c_0, commande c, suite de commandes C, on pose :

$$\begin{array}{rcll}
((c) \cdot C, \rho) & \to & (c \cdot C, \rho) & \\
(c \texttt{ Se } c_0 \cdot C, \rho) & \to & (c \cdot c_0 \cdot C, \rho) & \\
(\texttt{Sk } \cdot C, \rho) & \to & (C, \rho) & \\
(x \texttt{ Af } e \cdot C, \rho) & \to & (C, \rho[x \mapsto [\![e]\!]\rho]) & \\
(\texttt{If } e \texttt{ Th } c \texttt{ El } c_0 \cdot C, \rho) & \to & (c \cdot C, \rho) & \text{si } [\![e]\!]\rho \neq 0 \\
(\texttt{If } e \texttt{ Th } c \texttt{ El } c_0 \cdot C, \rho) & \to & (c_0 \cdot C, \rho) & \text{si } [\![e]\!]\rho = 0 \\
(\texttt{Wh } e \texttt{ Do } c_0 \cdot C, \rho) & \to & (c_0 \cdot \texttt{Wh } e \texttt{ Do } c_0 \cdot C, \rho) & \text{si } [\![e]\!]\rho \neq 0 \\
(\texttt{Wh } e \texttt{ Do } c_0 \cdot C, \rho) & \to & (C, \rho) & \text{si } [\![e]\!]\rho = 0.
\end{array}$$

La sémantique d'une commande c est définie par : $[\![c]\!]\rho = \rho'$ si et seulement si

$$(c, \rho) \to^* (\varepsilon, \rho').$$

2. On assimile chaque identificateur, qui est un mot concret, à une seule variable.