Semántica formal de un lenguaje de programación

2 – Expresiones booleanas

Maximiliano Cristiá
Universidad Nacional de Rosario
Argentina

2019

Este documento describe el modelo matemático que formaliza la semántica de un lenguaje de programación muy simple. Es una extensión del lenguaje y modelo presentado en el vídeo `https://youtu.be/3ybrhTCU6BQ`.

La extensión consiste en agregar algunas expresiones booleanas que serán las condiciones para las sentencias condicionales if y while. El vídeo que presenta esta extensión lo pueden ver acá `https://youtu.be/oiXaJ0ClUyE`.

La gramática del lenguaje de programación

En esta sección presentamos la gramática del lenguaje de programación en BNF.

Sea $\mathbb{Z}$ el conjunto de los números enteros y Var el conjunto de los nombres de variables que se pueden utilizar en el lenguaje de programación.

$ExprE ::= \mathbb{Z} \mid Var \mid ExprE + ExprE$

$ExprB ::= \mathsf{true} \mid \mathsf{false} \mid ExprE == ExprE \mid ExprE <= ExprE \mid \sim ExprB$

$Sentencia ::= Var = ExprE \mid Estructura$

$Estructura ::=$
$\quad \mathsf{if}\ ExprB\ \mathsf{then}\ Programa\ \mathsf{fi}$
$\quad \mid \mathsf{while}\ ExprB\ \mathsf{do}\ Programa\ \mathsf{done}$

$Programa ::= Sentencia \mid Programa\ ;\ Sentencia$

La semántica formal del lenguaje de programación

Ahora definimos la función *evalb* que nos permite evaluar las expresiones booleanas de nuestro lenguaje. *evalb* depende de una expresión booleana (*ExprB*) y de

una memoria. Definimos $\mathbb{B}$ como el conjunto $\{\mathsf{true}, \mathsf{false}\}$.

$$evalb : (Var \to \mathbb{Z}) \times ExprB \to \mathbb{B}$$

$$evalb(m, b) = b, \text{ si } b \in \mathbb{B}$$

$$evalb(m, e_1 == e_2) = \begin{cases} \mathsf{true} & \text{si } m[\![e_1]\!] = m[\![e_2]\!] \\ \mathsf{false} & \text{si } m[\![e_1]\!] \neq m[\![e_2]\!] \end{cases}$$

$$evalb(m, e_1 <= e_2) = \begin{cases} \mathsf{true} & \text{si } m[\![e_1]\!] \leq m[\![e_2]\!] \\ \mathsf{false} & \text{si } m[\![e_1]\!] > m[\![e_2]\!] \end{cases}$$

$$evalb(m, \sim e) = \begin{cases} \mathsf{false} & \text{si } e = \mathsf{true} \\ \mathsf{true} & \text{si } e = \mathsf{false} \\ evalb(m, \sim evalb(m, e)) & \text{en cualquier otro caso} \end{cases}$$

De igual forma a como hicimos con *eval* y *exec* definimos un sinónimo para *evalb* con la intención de simplificar la notación:

$$evalb(m, e) = m[\![e]\!]$$

Aunque usamos la misma notación que para el sinónimo de *eval* y *exec* no debería haber confusión porque aplican sobre elementos gramaticales distintos. En particular el sinónimo de *evalb* aplica sobre expresiones booleanas.

Ahora tenemos que redefinir la función *exec* solo para las *Estructura*. Recordar que hemos definido un sinónimo para *exec*:

$$exec(m, P) = m[\![P]\!]$$

Informalmente, queremos que la semántica de las estructuras if y while sea que el interior se ejecuta si la condición evalúa a true.

- Para la estructura if.

$$m[\![\mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{fi}]\!] = \begin{cases} m[\![P]\!] & \text{si } m[\![e]\!] = \mathsf{true} \\ m & \text{si } m[\![e]\!] = \mathsf{false} \end{cases}$$

- Para la estructura while:

$$m[\![\mathsf{while}\ e\ \mathsf{do}\ P\ \mathsf{done}]\!] = \begin{cases} m[\![P\ ;\ \mathsf{while}\ e\ \mathsf{do}\ P\ \mathsf{done}]\!] & \text{si } m[\![e]\!] = \mathsf{true} \\ m & \text{si } m[\![e]\!] = \mathsf{false} \end{cases}$$

Ejercicios

1. Extienda la gramática y la semántica de las expresiones booleanas con la conjunción.