Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Licenciatura en Ciencias de la Computación
Estructuras de datos y algoritmos II

Lenguaje Interpretado Simple

Alumnos:
CRESPO, Lisandro (C-6165/4)
MISTA, Agustín (M-6105/1)
.
.

Docentes:
JASKELIOFF, Mauro
RABASEDAS, Juan Manuel
SIMICH, Eugenia
MANZINO, Cecilia

28 de Agosto de 2015

Ejercicio 2.2.1. *Extendemos la sintaxis abstracta y concreta de las expresiones enteras del LIS a modo de incluir el operador de asignación ternario del lenguaje C.*

Sintaxis abstracta

$$\begin{aligned}
\langle intexp \rangle \ ::= & \langle nat \rangle \mid \langle var \rangle \mid -u \ \langle intexp \rangle \\
& \mid \langle intexp \rangle \ + \ \langle intexp \rangle \\
& \mid \langle intexp \rangle \ -b \ \langle intexp \rangle \\
& \mid \langle intexp \rangle \ \times \ \langle intexp \rangle \\
& \mid \langle intexp \rangle \ \div \ \langle intexp \rangle \\
& \mid \langle boolexp \rangle \ ? \ \langle intexp \rangle \ : \ \langle intexp \rangle
\end{aligned}$$

Sintaxis concreta

$$\begin{aligned}
\langle intexp \rangle \ ::= & \langle nat \rangle \\
& \mid \ \langle var \rangle \\
& \mid \ \text{`}-\text{`} \ \langle intexp \rangle \\
& \mid \ \langle intexp \rangle \ \text{`}+\text{`} \ \langle intexp \rangle \\
& \mid \ \langle intexp \rangle \ \text{`}-\text{`} \ \langle intexp \rangle \\
& \mid \ \langle intexp \rangle \ \text{`}*\text{`} \ \langle intexp \rangle \\
& \mid \ \langle intexp \rangle \ \text{`}/\text{`} \ \langle intexp \rangle \\
& \mid \ \text{`(`} \ \langle intexp \rangle \ \text{`)`} \\
& \mid \ \langle boolexp \rangle \ \text{`?`} \ \langle intexp \rangle \ \text{` : `} \ \langle intexp \rangle
\end{aligned}$$

Ejercicio 2.3.1. *Extendemos la sintaxis abstracta de las expresiones enteras en Haskell para incluir el operador de asignación ternario descripto en el Ejercicio 2.2.1.*

$$\begin{aligned}
dataIntExp = & \ Const \ \ Int \\
& \mid Var \ \ Variable \\
& \mid UMinus \ \ IntExp \\
& \mid Plus \ \ IntExp \ \ IntExp \\
& \mid Minus \ \ IntExp \ \ IntExp \\
& \mid Times \ \ IntExp \ \ IntExp \\
& \mid Div \ \ IntExp \ \ IntExp \\
& \mid IfAss \ \ BoolExp \ \ IntExp \ \ IntExp
\end{aligned}$$

Ejercicio 2.4.1. *Extendemos la semántica denotacional de las expresiones enteras para incluir el operador ternario descripto en el Ejercicio 2.2.1*

$$[\![cond \ ? \ expT \ : \ expF]\!]_{intexp}\sigma = \begin{cases} [\![expT]\!]_{intexp}\sigma & \text{si } [\![cond]\!]_{boolexp}\sigma = \textbf{true} \\ [\![expF]\!]_{intexp}\sigma & \text{si } [\![cond]\!]_{boolexp}\sigma = \textbf{false} \end{cases}$$

Ejercicio 2.5.1. *Demostrar que la relación de evaluación de un paso $\rightsquigarrow$ es determinista*

Blah, blah, blah..

Ejercicio 2.5.2. *Construimos un árbol de prueba para demostrar que:*

$$\langle x := x + 1;\ \textbf{if } x > 0 \textbf{ then skip else } x := x - 1, [\sigma|x : 0]\rangle \rightsquigarrow^* [\sigma|x : 1]$$

$$\frac{(1) \qquad\qquad (2)}{\langle x := x + 1;\ \textbf{if } x > 0 \textbf{ then skip else } x := x - 1, [\sigma|x : 0]\rangle \rightsquigarrow^* [\sigma|x : 1]}\ TR2$$

$$\frac{\dfrac{\dfrac{}{\langle x := x + 1, [\sigma|x : 0]\rangle \rightsquigarrow [\sigma|x : 1]}\ ASS^{(3)}}{\langle x := x + 1;\ \textbf{if } x > 0 \textbf{ then skip else } x := x - 1, [\sigma|x : 0]\rangle \rightsquigarrow \langle \textbf{if } x > 0 \textbf{ then skip else } x := x - 1, [\sigma|x : 1]\rangle}\ SEQ1}{\langle x := x + 1;\ \textbf{if } x > 0 \textbf{ then skip else } x := x - 1, [\sigma|x : 0]\rangle \rightsquigarrow^* \langle \textbf{if } x > 0 \textbf{ then skip else } x := x - 1, [\sigma|x : 1]\rangle}\ TR1$$

(1)

$$\frac{\dfrac{\dfrac{[\![x > 0]\!][\sigma|x : 1] = \textbf{true}^{(4)}}{\langle \textbf{if } x > 0 \textbf{ then skip else } x := x - 1, [\sigma|x : 1]\rangle \rightsquigarrow \langle \textbf{skip}, [\sigma|x : 1]\rangle}\ IF1}{\langle \textbf{if } x > 0 \textbf{ then skip else } x := x - 1, [\sigma|x : 1]\rangle \rightsquigarrow^* \langle \textbf{skip}, [\sigma|x : 1]\rangle}\ TR1 \qquad \dfrac{\dfrac{}{\langle \textbf{skip}, [\sigma|x : 1]\rangle \rightsquigarrow [\sigma|x : 1]}\ SKIP}{\langle \textbf{skip}, [\sigma|x : 1]\rangle \rightsquigarrow^* [\sigma|x : 1]}\ TR1}{\langle \textbf{if } x > 0 \textbf{ then skip else } x := x - 1, [\sigma|x : 1]\rangle \rightsquigarrow^* [\sigma|x : 1]}\ TR2$$

(2)

Además, probamos que:

$$[\![x + 1]\!][\sigma|x : 0] = [\![x]\!][\sigma|x : 0] + [\![1]\!][\sigma|x : 1] = 0 + 1 = \textbf{1} \quad (3)$$

$$[\![x > 0]\!][\sigma|x : 1] = [\![x]\!][\sigma|x : 1] > [\![0]\!] = 1 > 0 = \textbf{true} \quad (4)$$

Ejercicio 2.5.6. *Agregamos una producción a la gramática abstracta de los comandos del LIS para el comando* **repeat**

$$\begin{aligned}\langle comm\rangle\ ::= &\ \textbf{skip}\\ &\mid \langle var\rangle := \langle intexp\rangle\\ &\mid \langle comm\rangle\ ;\ \langle comm\rangle\\ &\mid \textbf{if } \langle boolexp\rangle \textbf{ then } \langle comm\rangle \textbf{ else } \langle comm\rangle\\ &\mid \textbf{while } \langle boolexp\rangle \textbf{ do } \langle comm\rangle\\ &\mid \textbf{repeat } \langle comm\rangle \textbf{ until } \langle boolexp\rangle\end{aligned}$$

Extendemos la semántica operacional del LIS con reglas de inferencia para el comando **repeat**

$$\frac{\langle c, \sigma\rangle \rightsquigarrow \sigma' \qquad [\![b]\!]_{boolexp}\sigma' = \textbf{false}}{\langle \textbf{repeat } c \textbf{ until } b,\ \sigma\rangle \rightsquigarrow \langle c; \textbf{repeat } c \textbf{ until } b,\ \sigma'\rangle}\ REP1$$

$$\frac{\langle c, \sigma\rangle \rightsquigarrow \sigma' \qquad [\![b]\!]_{boolexp}\sigma' = \textbf{true}}{\langle \textbf{repeat } c \textbf{ until } b,\ \sigma\rangle \rightsquigarrow \sigma'}\ REP2$$

PREGUNTAR!!!