Análisis de Lenguajes de Programación

Trabajo práctico 1

Ignacio Litmanovich Aldana Zarate

Octubre 2020

Ejercicio 1

Sintaxis abstracta de LIS

intexp ::= nat
| var
|$-_u$ intexp
|intexp $-_b$ intexp
|intexp x intexp
|intexp ÷ intexp
|var = intexp
|intexp , intexp

El resto de las reglas quedan iguales a las dadas en el enunciado.

Sintaxis concreta de LIS

intexp ::= nat
| var
| '-' intexp
|intexp '+' intexp
|intexp '-' intexp
|intexp '*' intexp
|'('intexp')'
|var '=' intext
|intexp ',' intexp

El resto de las reglas quedan iguales a las dadas en el enunciado.

Pero, podemos observar que esta gramática es ambigua. No denota los órdenes de precedencia de los operadores (por ejemplo, que la multiplicación o división tiene mayor precedencia que la suma o resta) por lo que nos podría conllevar a resultados erróneos.

Resolución de ambigüedad de la gramática

Sintaxis abstracta de LIS desambigüada

intseq ::= intseq , intass
| intass
intass ::= intass = intexp
| intexp
intexp ::= intexp + intterm
| intexp - intterm
| interm
interm ::= interm * factor
| interm / factor
| factor
factor::= nat | var | -factor | (intseq)
boolexp ::= boolor
boolor ::= boolor ∨ booland | booland
booland ::= booland ∧ boolnot | boolnot
boolnot ::= ¬ boolterm | boolterm
boolterm ::= true

| false
| (boolexp)
| intseq == intseq
| intseq $\neq$ intseq
| intseq < intseq
| intseq > intseq
comm ::= comm ; commin
| commin
commin ::= skip
| var = intseq
| **if** boolexp **then** comm **else** comm
| **while** boolexp **do** comm

Sintaxis concreta de LIS desambigüada

digit::= ’0’ | ’1’ | . . . | ’9’
letter::= ’a’ | . . . | ’Z’
nat::= digit | digit nat
var::= letter | letter var
intseq ::= intseq ’,’ intass
| intass
intass ::= intass ’=’ intexp
| intexp
intexp ::= intexp ’+’ term
| intexp ’-’ interm
| interm
interm ::= interm’*’ factor
| interm ’/’ factor
| factor
factor::= nat | var | ’-’factor | ’(’intseq’)’
boolexp ::= **boolor**
boolor ::= boolor ’||’ booland | booland
booland ::= booland ’&&’ boolnot | boolnot
boolnot ::= ’!’boolterm | boolterm
boolterm = intseq ’==’ intseq
| intseq ’!=’ intseq
| intseq ’<’ intseq
| intseq ’>’ intseq
| true
| false
| ’(’boolexp’)’
comm ::= comm ’;’ commin
| commin
commin ::= **skip**
| var ’=’ intseq
| ’if’ boolexp ’\{’ comm ’\}’
| ’if’ boolexp ’{’ comm ’}’ **else** ’{’ comm ’}’
| ’while’ boolexp **do** comm

Ejercicio 2 y 3

Resueltos en los archivos AST.hs y Parser.hs.

Ejercicio 4

$$\frac{\langle e_0,\ \sigma\rangle \Downarrow_{\text{exp}} \langle n_0,\ \sigma'\rangle \quad \langle e_1,\ \sigma'\rangle \Downarrow_{\text{exp}} \langle n_1,\ \sigma''\rangle}{\langle e_0, e_1,\ \sigma\rangle \Downarrow_{\text{exp}} \langle n_1,\ \sigma''\rangle}\ \textsc{Comma}$$

$$\frac{\langle e,\ \sigma\rangle \Downarrow_{\text{exp}} \quad \langle n,\ \sigma'\rangle}{\langle v := e,\ \sigma\rangle \Downarrow_{\text{exp}} \langle n,\ [\sigma'|v : e]\rangle}\ \text{E}\textsc{Assign}$$

Ejercicio 5

Queremos ver que si $t \to t'$ y $t \to t''$ entonces $t' = t''$

Demostración

Procedemos por inducción sobre la derivación $t \rightsquigarrow t'$.

Tenemos como hipótesis inductiva que todas las subderivaciones pertenecientes a t son deterministas.

Si la última regla de $t \rightsquigarrow t'$ fue ASS:

Entonces sabemos que t es de la forma $\langle v = e, \sigma\rangle$. Como $\Downarrow$ es determinista, e *solo* evalúa a n (no podemos tener $\langle e, \sigma\rangle \Downarrow \langle n, \sigma'\rangle$ y $\langle e, \sigma\rangle \Downarrow \langle n_1, \sigma'\rangle$ con $n \neq n_1$). Además, por la forma de t, ninguna de las otras reglas presentes puede ser utilizada. Por lo tanto, la última regla en la segunda derivación puede ser solamente ASS y el único valor posible a asignar por σ' es n. Por lo tanto, $t' = t''$.

Si la última regla de $t \rightsquigarrow t'$ fue SEQ_1:

Entonces sabemos que t es de la forma $\langle \textbf{skip}; t_1, \sigma\rangle$. Entonces, la premisa para aplicar SEQ_2 no puede ser posible, ya que toda ejecución que termina lo hace en **skip**, para algún estado σ. Por lo tanto, la última regla aplicada en la segunda derivación no puede ser SEQ_2 ya que la derivación $\langle \textbf{skip}, \sigma\rangle \rightsquigarrow \langle c_0', \sigma'\rangle$ no es posible. Además, por la forma de t, ninguna de las otras reglas presentes puede ser utilizada. Por ende $t' = t''$.

Si la última regla de $t \to t'$ fue SEQ_2:

Entonces sabemos que t es de la forma $\langle t_1; t_2, \sigma\rangle$, donde $\langle t_1, \sigma\rangle \rightsquigarrow \langle t_1', \sigma'\rangle$. Debido a esto, la última regla aplicada en la segunda derivación no puede ser SEQ_1, porque no podemos tener $t1 = \langle \textbf{skip}, \sigma\rangle$ ya que toda ejecución que termina lo hace en **skip**. Esto es un absurdo ya que tenemos como premisa $\langle t_1, \sigma\rangle \rightsquigarrow \langle t_1', \sigma'\rangle$.

Por **HI**, en la segunda derivación tendremos también que $\langle t_1, \sigma\rangle \rightsquigarrow \langle t_1', \sigma'\rangle$ donde los comandos y los estados derivados coinciden con los de la primera derivación, por ser subderivaciones de t.

Además, por la forma de t, ninguna de las otras reglas presentes puede ser utilizada. Por ende $t' = t''$.

Si la última regla de $t \rightsquigarrow t'$ fue IF_1: Entonces sabemos que t es de la forma $\langle \textbf{if}\ t_1\ \textbf{then}\ t_2\ \textbf{else}\ t_3, \sigma\rangle$ donde $\langle t_1, \sigma\rangle \Downarrow \langle \textbf{true}, \sigma'\rangle$. Debido a que $\Downarrow$ es determinista, entonces no puede suceder que la última regla usada en $t \rightsquigarrow t''$ sea IF_2. (No podemos tener $\langle t_1, \sigma\rangle \Downarrow \langle \mathbf{true}, \sigma'\rangle$ y $\langle t_1, \sigma\rangle \Downarrow \langle \mathbf{false}, \sigma'\rangle$). Además, por la forma de t, ninguna de las otras reglas presentes puede ser utilizada.

Por lo tanto, la última regla en la segunda derivación puede ser solamente IF_1. Por lo tanto $t' = t''$.

Si la última regla de $t \rightsquigarrow t'$ fue IF_2, entonces sabemos que t es de la forma $\langle \mathbf{if}\ t_1\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3, \sigma\rangle$ donde $\langle t_1, \sigma\rangle \Downarrow \langle \mathbf{false}, \sigma'\rangle$. Debido a que $\Downarrow$ es determinista, entonces no puede suceder que la última regla utilizada en $t \rightsquigarrow t''$ sea IF_1 (No podemos tener $\langle t_1, \sigma\rangle \Downarrow \langle \mathbf{true}, \sigma'\rangle$ y $\langle t_1, \sigma\rangle \Downarrow \langle \mathbf{false}, \sigma'\rangle$). Además, por la forma de t, ninguna de las otras reglas presentes puede ser utilizada.

Por lo tanto, la última regla en la segunda derivación puede ser solamente IF_2. Por lo tanto, $t' = t''$.

Si la última regla de $t \rightsquigarrow t'$ fue WHILE_1:

Entonces sabemos que t es de la forma $\langle \mathbf{while}\ t_1\ \mathbf{do}\ t_2, \sigma\rangle$ donde $\langle t_2, \sigma\rangle \Downarrow \langle \mathbf{true}, \sigma'\rangle$. Debido a que $\Downarrow$ es determinista, entonces no puede suceder que la última regla usada en $t \rightsquigarrow t''$ sea WHILE_2. (No podemos tener $\langle t_2, \sigma\rangle \Downarrow \langle \mathbf{true}, \sigma'\rangle$ y $\langle t_2, \sigma\rangle \Downarrow \langle \mathbf{false}, \sigma'\rangle$. Además, por la forma de t, ninguna de las otras reglas presentes puede ser utilizada. Por lo tanto, la última regla en la segunda derivación puede ser solamente WHILE_1. Por lo tanto, $t' = t''$.

Si la última regla de $t \rightsquigarrow t'$ fue WHILE_2: Entonces sabemos que t es de la forma $\langle \mathbf{while}\ t_1\ \mathbf{do}\ t_2, \sigma\rangle$ donde $\langle t_2, \sigma\rangle \Downarrow \langle \mathbf{false}, \sigma'\rangle$. Debido a que $\Downarrow$ es determinista, entonces no puede suceder que la última regla usada en $t \rightsquigarrow t''$ sea WHILE_1. (No podemos tener $\langle t_2, \sigma\rangle \Downarrow \langle \mathbf{true}, \sigma'\rangle$ y $\langle t_2, \sigma\rangle \Downarrow \langle \mathbf{false}, \sigma'\rangle$. Además, por la forma de t, ninguna de las otras reglas presentes puede ser utilizada. Por lo tanto, la última regla en la segunda derivación puede ser solamente WHILE_1. Por lo tanto, $t' = t''$.

Hemos demostrado determinismo para todas las comandos posibles existentes en *comm*.

$$\therefore \text{ La relación de evaluación en un paso } \rightsquigarrow \text{ es determinista.}$$

$\blacksquare$

Ejercicio 6

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle 1,\ [[\sigma|\ x:\ 2] \mid y:\ 2]\rangle \Downarrow_{\text{exp}} \langle 1,\ [[\sigma|\ x:\ 2] \mid y:\ 2]\rangle}\ \textsc{NVal}}{\langle y = 1,\ [[\sigma|\ x:\ 2] \mid y:\ 2]\rangle \Downarrow_{\text{exp}} \langle 1,\ [[\sigma|\ x:\ 2] \mid y:\ 1]\rangle}\ \textsc{EAssign}}{\langle x = y = 1,\ [[\sigma|\ x:\ 2] \mid y:\ 2]\rangle \rightsquigarrow \langle \mathbf{skip},\ [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle}\ \textsc{Ass}}{\langle x = y = 1;\quad \mathbf{while}\ x > 0\ \mathbf{do}\ x = x - y,\ [[\sigma|\ x:\ 2] \mid y:\ 2]\rangle \rightsquigarrow \langle \mathbf{skip}, \mathbf{while}\ x > 0\ \mathbf{do}\ x = x - y,\ [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle}\ \textsc{Seq}_2$$

Figura 1: Paso 1

$$\dfrac{}{\langle \mathbf{skip}; \mathbf{while}\ x > 0\ \mathbf{do}\ x = x - y, \quad [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle \rightsquigarrow \langle \mathbf{while}\ x > 0\ \mathbf{do}\ x = x - y,\ [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle}\ \textsc{Seq}_1$$

Figura 2: Paso 2

$$\dfrac{\dfrac{\dfrac{}{\langle x,\ [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle \Downarrow_{\text{exp}} \langle [[\sigma|\ x:\ 1] \mid y:\ 1] \quad x,\ [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle}\ \textsc{Var} \qquad \dfrac{}{\langle 0,\ [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle \Downarrow_{\text{exp}} \langle \mathbf{0},\ [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle}\ \textsc{NVal}}{\langle x > 0,\ [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle \Downarrow_{\text{exp}} \langle \mathbf{1 > 0},\ [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle}\ \textsc{Gt}}{\langle \mathbf{while}\ x > 0\ \mathbf{do}\ x = x -_b y,\ [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle \rightsquigarrow \langle x = x - y; \mathbf{while}\ x > 0\ \mathbf{do}\ ,\ [[\sigma|\ x:\ 1] \mid y:\ 1]\rangle}\ \textsc{While}_1$$

Figura 3: Paso 3.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle x,\ [[\sigma|\ x:\ 1]\,|\ y:\ 1]\rangle \Downarrow_{\text{exp}} \langle 1,\ [[\sigma|\ x:\ 1]\,|\ y:\ 1]\rangle}\,\textsc{Var} \qquad \dfrac{}{\langle y,\ [[\sigma|\ x:\ 1]\,|\ y:\ 1]\rangle \Downarrow_{\text{exp}} \langle 1,\ [[\sigma|\ x:\ 1]\,|\ y:\ 1]\rangle}\,\textsc{Var}}{\langle x-y,\ [[\sigma|\ x:\ 1]\,|\ y:\ 1]\rangle \Downarrow_{\text{exp}} \langle \mathbf{1-1},\ [[\sigma|\ x:\ 1]\,|\ y:\ 1]\rangle}\,\textsc{Minus}}{\langle x=x-y,\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle \rightsquigarrow \langle \mathit{skip},\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle}\,\textsc{Ass}}{\langle x=x-y,\ \textbf{while}\ x>0\ \textbf{do}\ x=x-y\rangle\, [[\sigma|\ x:\ 0]\,|\ y:\ 1] \rightsquigarrow \langle \textbf{skip}; \textbf{while}\ x>0\ \textbf{do}\ x=x-y,\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle}\,\textsc{Seq}_2$$

Figura 4: Paso 4.

$$\dfrac{}{\langle \textbf{skip}; \textbf{while}\ x>0\ \textbf{do}\ x=x-y,\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle \rightsquigarrow \langle \textbf{while}\ x>0\ \textbf{do}\ x=x-y,\quad [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle}\,\textsc{Seq}_1$$

Figura 5: Paso 5

$$\dfrac{\dfrac{\dfrac{}{\langle x,\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle \Downarrow_{\text{exp}} \langle [[\sigma|\ x:\ 0]\,|\ y:\ 1]\quad x,\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle}\,\textsc{Var} \qquad \dfrac{}{\langle 0,\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle \Downarrow_{\text{exp}} \langle \mathbf{0},\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle}\,\text{N}\textsc{Val}}{\langle x>0,\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle \Downarrow_{\text{exp}} \langle \mathbf{0>0},\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle}\,\textsc{Gt}}{\langle \textbf{while}\ x>0\ \textbf{do}\ x=x-_b y,\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle \rightsquigarrow \langle \textbf{skip},\ [[\sigma|\ x:\ 0]\,|\ y:\ 1]\rangle}\,\textsc{While}_2$$

Figura 6: Paso 6.

Ejercicio 10

Modificación de sintaxis abstracta de LIS:
comm ::= comm ; commin
| commin
commin ::= skip
| var = intseq
| **if** boolexp **then** comm **else** comm
| **while** boolexp **do** comm
| **for** (intexp;boolexp;intexp) **do** comm
A la semántica operacional de comandos se le agrega la siguiente regla:

$$\dfrac{\langle \boldsymbol{e_1},\ \sigma\rangle \Downarrow_{\text{exp}} \langle \boldsymbol{n},\ \sigma'\rangle}{\langle \textbf{for}\ (e1;b;e2)\ \textbf{do}\ c,\ \sigma\rangle \rightsquigarrow \langle \textbf{while}\ b\ \textbf{do}\ (c; \textbf{if}\ e2>0\ \textbf{then skip else skip}),\ \sigma'\rangle}\,\textsc{For}$$