Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Análisis de lenguajes de programación

Trabajo Práctico 1

Autor:
Tomás Castro Rojas
Blas Barbagelata

22 de septiembre de 2021

1. Ejercicio 1

Extendemos las sintaxis abstracta y concreta de LIS para incluir asignaciones de variables como expresiones enteras, de la forma x = e y el operador **,** para escribir una secuencia de expresiones enteras. Solo incluimos las partes de cada sintaxis donde se hacen modificaciones.

1.1. Sintaxis Abstracta

$$
\begin{aligned}
\textbf{intexp} ::= & \ nat \mid var \mid \ -_u \ intexp \\
& \mid \ intexp \ + \ intexp \\
& \mid \ intexp \ -_b \ intexp \\
& \mid \ intexp \ \times \ intexp \\
& \mid \ intexp \ \div \ intexp \\
& \mid \ var = intexp \\
& \mid \ intexp, \ intexp
\end{aligned}
$$

1.2. Sintaxis Concreta

$$
\begin{aligned}
\textbf{intexp} ::= & \ nat \\
& \mid var \\
& \mid \text{'-'} \ intexp \\
& \mid \ intexp \ \text{'+'} \ intexp \\
& \mid \ intexp \ \text{'-'} \ intexp \\
& \mid \ intexp \ \text{'*'} \ intexp \\
& \mid \ intexp \ \text{'/'} \ intexp \\
& \mid \ \text{'('} \ intexp \ \text{')'} \\
& \mid \ var \ \text{'='} \ intexp \\
& \mid \ intexp \ \text{','} \ intexp
\end{aligned}
$$

2. Ejercicio 2

```
data Exp a where
  Const  ::Int -> Exp Int
  Var    ::Variable -> Exp Int
  UMinus ::Exp Int -> Exp Int
  Plus   ::Exp Int -> Exp Int -> Exp Int
  Minus  ::Exp Int -> Exp Int -> Exp Int
  Times  ::Exp Int -> Exp Int -> Exp Int
  Div    ::Exp Int -> Exp Int -> Exp Int
  EAssgn ::Variable -> Exp Int -> Exp Int
  ESeq   ::Exp Int -> Exp Int -> Exp Int
```

3. Ejercicio 3

Para la implementación del Parser para LIS primero decidimos desambiguar la gramática que tenemos para dejar en claro el orden de precedencia de los distintos operadores y además para tener como guía de como debería funcionar el Parser.

3.1. Sintaxis Abstracta

$$
\begin{aligned}
\textbf{intseq} &::= \mathit{intseq},\ \mathit{intassgn} \mid \mathit{intassgn}\\
\textbf{intassgn} &::= \mathit{var}\ =\ \mathit{intexp} \mid \mathit{intexp}\\
\textbf{intexp} &::= \mathit{intexp}\ +\ \mathit{intterm} \mid \mathit{intexp}\ -_b\ \mathit{intterm} \mid \mathit{intterm}\\
\textbf{intterm} &::= \mathit{intterm}\ \times\ \mathit{factor} \mid \mathit{intterm}\ \div\ \mathit{factor} \mid \mathit{factor}\\
\textbf{intfactor} &::= \mathit{nat} \mid \mathit{var} \mid\ -_u\, intfactor \mid (\mathit{intseq})
\end{aligned}
$$

$$
\begin{aligned}
\textbf{boolexp} &::= \mathit{boolexp}\ \vee\ \mathit{booland} \mid \mathit{booland}\\
\textbf{booland} &::= \mathit{booland}\ \wedge\ \mathit{boolnot} \mid \mathit{boolnot}\\
\textbf{boolnot} &::= \neg\, \mathit{boolterm} \mid \mathit{boolterm}\\
\textbf{boolterm} &::= \textbf{true} \mid \textbf{false}\\
&\mid \mathit{intseq}\ ==\ \mathit{intseq}\\
&\mid \mathit{intseq} \neq \mathit{intseq}\\
&\mid \mathit{intseq}\ <\ \mathit{intseq}\\
&\mid \mathit{intseq}\ >\ \mathit{intseq}\\
&\mid (\mathit{boolexp})
\end{aligned}
$$

$$
\begin{aligned}
\textbf{commseq} &::= \textit{commseq}\ ;\ \textit{comm} \mid \textit{comm} \\
\textbf{comm} &::= \textbf{skip} \\
&\mid \textit{var}\ =\ \textit{intseq} \\
&\mid \textbf{if}\ \textit{boolexp}\ \textbf{then}\ \textit{commseq}\ \textbf{else}\ \textit{commseq} \\
&\mid \textbf{repeat}\ \textit{commseq}\ \textbf{until}\ \textit{boolexp}
\end{aligned}
$$

3.2. Sintaxis Concreta

$$
\begin{aligned}
\textbf{digit} &::= \text{'0'} \mid \text{'1'} \mid \ldots \mid \text{'9'} \\
\textbf{letter} &::= \text{'a'} \mid \text{'b'} \mid \ldots \mid \text{'Z'} \\
\textbf{nat} &::= \textit{digit} \mid \textit{digit nat} \\
\textbf{var} &::= \textit{letter} \mid \textit{letter var}
\end{aligned}
$$

$$
\begin{aligned}
\textbf{intseq} &::= \textit{intseq}\text{','}\ \textit{intassgn} \mid \textit{intassgn} \\
\textbf{intassgn} &::= \textit{var}\ \text{'='}\ \textit{intexp} \mid \textit{intexp} \\
\textbf{intexp} &::= \textit{intexp}\ \text{'+'}\ \textit{intterm} \mid \textit{intexp}\ \text{'-'}\ \textit{intterm} \mid \textit{intterm} \\
\textbf{intterm} &::= \textit{intterm}\ \text{'*'}\ factor \mid \textit{intterm}\ \text{'/'}\ factor \mid factor \\
\textbf{intfactor} &::= \textit{nat} \mid \textit{var} \mid \text{'-'}\ intfactor \mid \text{'('}\textit{intseq}\text{')'}
\end{aligned}
$$

$$
\begin{aligned}
\textbf{boolexp} &::= \textit{boolexp}\ \text{'||'}\ \textit{booland} \mid \textit{booland} \\
\textbf{booland} &::= \textit{booland}\ \text{'\&\&'}\ \textit{boolnot} \mid \textit{boolnot} \\
\textbf{boolnot} &::= \text{'!'}\ \textit{boolterm} \mid \textit{boolterm} \\
\textbf{boolterm} &::= \textit{true} \mid false \\
&\mid \textit{intseq}\ \text{'=='}\ \textit{intseq} \\
&\mid \textit{intseq}\ \text{'!='}\ \textit{intseq} \\
&\mid \textit{intseq}\ \text{'<'}\ \textit{intseq} \\
&\mid \textit{intseq}\ \text{'>'}\ \textit{intseq} \\
&\mid \text{'('}\textit{boolexp}\text{')'}
\end{aligned}
$$

$$
\begin{aligned}
\textbf{commseq} &::= \textit{commseq}\ \text{';'}\ \textit{comm} \mid \textit{comm} \\
\textbf{comm} &::= \textbf{skip} \\
&\mid \textit{var}\ \text{'='}\ \textit{intseq} \\
&\mid \text{'if'}\ \textit{boolexp}\ \text{'then'}\ \text{'\{'}\ \textit{commseq}\ \text{'\}'} \\
&\mid \text{'if'}\ \textit{boolexp}\ \text{'then'}\ \text{'\{'}\ \textit{commseq}\ \text{'\}'}\ \text{'else'}\ \text{'\{'}\ \textit{commseq}\ \text{'\}'} \\
&\mid \text{'repeat'}\ \textit{commseq}\ \text{'until'}\ \textit{boolexp}
\end{aligned}
$$

4. Ejercicio 4

Extendemos la semantica big-step para incluir la asignación de expresiones como expresiones y el operador **,** para secuencias de expresiones.

$$\frac{\langle e,\ \sigma\rangle \Downarrow_{\text{exp}} \langle n,\ \sigma'\rangle}{\langle var = e,\ \sigma\rangle \Downarrow_{\text{exp}} \langle n,\ [\sigma'\ |var : n]\rangle}\ \textsc{EAssgn}$$

$$\frac{\langle e_1,\ \sigma\rangle \Downarrow_{\text{exp}} \langle n_1,\ \sigma'\rangle \quad \langle e_2,\ \sigma'\rangle \Downarrow_{\text{exp}} \langle n_2,\ \sigma''\rangle}{\langle e_1, e_2,\ \sigma\rangle \Downarrow_{\text{exp}} \langle n_2,\ \sigma''\rangle}\ \textsc{ESeq}$$

5. Ejercicio 5

Vamos a demostrar que la relación de evaluación en un paso $\rightsquigarrow$ es determinista. Formalmente, si $t \rightsquigarrow t'$ y $t \rightsquigarrow t''$ entonces $t' = t''$.
Suponiendo que la relación $\Downarrow$ es determinista, vamos a demostrar el determinismo realizando inducción sobre la derivación $t \rightsquigarrow t'$. Analizamos caso por caso:

- Si $t \rightsquigarrow t'$ usando como ultima regla Ass

Entonces el arbol de derivación debe tener la forma

$$\frac{\langle e,\ \sigma\rangle \Downarrow_{\text{exp}} \langle n,\ \sigma'\rangle}{\underbrace{\langle var = e,\ \sigma\rangle}_{t} \rightsquigarrow \underbrace{\langle skip,\ [\sigma'|var : n]\rangle}_{t'}}\ \textsc{Ass}$$

Ahora, supongamos que $t' \neq t''$. Dado la forma de t la única regla que se puede aplicar es Ass. Entonces, t'' debe tener la forma $\langle skip,\ [\sigma''|var : n']\rangle$ con la premisa $\langle e,\ \sigma\rangle \Downarrow_{\text{exp}} \langle n',\ \sigma''\rangle$. Como $t' \neq t''$ debe suceder que $\sigma' \neq \sigma''$ o $n' \neq n''$. Pero esto implicaría que la relación $\Downarrow$ no es determinista, lo cual contradice la hipótesis. Absurdo.
$\therefore t' = t''$

- Si $t \rightsquigarrow t'$ usando como ultima regla Seq1

Entonces el arbol de derivación debe tener la forma

$$\frac{}{\underbrace{\langle \textbf{skip}; c,\ \sigma\rangle}_{t} \rightsquigarrow \underbrace{\langle c,\ \sigma\rangle}_{t'}}\ \textsc{Seq1}$$

Dado la forma de t la unica regla que se puede aplicar es Seq1. Entonces debe ocurrir que t'' tiene la forma $\langle c,\ \sigma\rangle$.
$\therefore t' = t''$

- Si $t \rightsquigarrow t'$ usando como ultima regla Seq2

Entonces el arbol de derivación debe tener la forma

$$\frac{\langle c_0,\ \sigma\rangle \rightsquigarrow \langle c_0',\ \sigma'\rangle}{\underbrace{\langle c_0; c_1,\ \sigma\rangle}_{t} \rightsquigarrow \underbrace{\langle c_0'; c_1,\ \sigma'\rangle}_{t'}}\ \textsc{Seq2}$$

Dada la forma de t y que se aplico la regla Seq2, podemos asegurar que $c_0 \neq \textbf{skip}$ ya que de lo contrario la premisa no tendria sentido, la ejecución finaliza en un skip. Entonces la unica regla que se puede aplicar dada la forma de t es Seq2. Luego, la premisa debe tener forma $\langle c_0,\ \sigma\rangle \rightsquigarrow \langle c_0'',\ \sigma''\rangle$ y t'' tiene la forma $\langle c_0''; c_1,\ \sigma''\rangle$. Por Hipotesis Inductiva, las subderivaciones de $\rightsquigarrow$ son deterministas. Por lo tanto, $\langle c_0',\ \sigma'\rangle = \langle c_0'',\ \sigma''\rangle$.
$\therefore t' = t''$

- Si $t \rightsquigarrow t'$ usando como ultima regla IF1

Entonces el arbol de derivación debe tener la forma

$$\frac{\langle bexp,\ \sigma\rangle \Downarrow_{\text{exp}} \langle \textbf{true},\ \sigma'\rangle}{\underbrace{\langle\ \textbf{if}\ bexp\ \textbf{then}\ c_0\ \textbf{else}\ c_1,\ \sigma\rangle}_{t} \rightsquigarrow \underbrace{\langle c_0,\ \sigma'\rangle}_{t'}}\ \text{If1}$$

Por la forma de t, solo es posible aplicar alguna de las reglas If. Luego, por hipotesis, $\Downarrow$ es determinista. Entonces $\langle bexp,\ \sigma\rangle$ solo puede evaluar a $\langle \textbf{true},\ \sigma'\rangle$. Que evalue a $\langle \textbf{false},\ \sigma'\rangle$ seria un absurdo. Ergo, en $t \rightsquigarrow t''$ solo es posible aplicar If1.
$\therefore t' = t''$

- Si $t \rightsquigarrow t'$ usando como ultima regla IF2

Entonces el arbol de derivación debe tener la forma

$$\frac{\langle bexp,\ \sigma\rangle \Downarrow_{\text{exp}} \langle \textbf{false},\ \sigma'\rangle}{\underbrace{\langle\ \textbf{if}\ bexp\ \textbf{then}\ c_0\ \textbf{else}\ c_1,\ \sigma\rangle}_{t} \rightsquigarrow \underbrace{\langle c_1,\ \sigma'\rangle}_{t'}}\ \text{If2}$$

Analogo a If1 $\therefore t' = t''$

- Si $t \rightsquigarrow t'$ usando como ultima regla Repeat

$$\frac{}{\underbrace{\langle \textbf{repeat}\ c\ \textbf{until}\ b,\ \sigma\rangle}_{t} \rightsquigarrow \underbrace{\langle c;\ \textbf{if}\ b\ \textbf{then}\ \textbf{skip}\ \textbf{else}\ \textbf{repeat}\ c\ \textbf{until}\ b,\ \sigma\rangle}_{t'}}\ \text{Repeat}$$

Dada la forma del termino t, solo es posible aplicar la regla Repeat ya que las demas necesitan un comando distinto a **repeat**. Entonces, en $t \rightsquigarrow t''$ tambien solo se puede aplicar la regla de inferencia Repeat.
$\therefore t' = t''$

Concluimos en cada caso de derivación posible que $t' = t''$.
Por lo tanto, la relación $\rightsquigarrow$ es determinista. $\square$

6. Ejercicio 6

Queremos demostrar que el siguiente juicio es válido:

$$\langle x = y = 1; \textbf{repeat}\ x = x - y\ \textbf{until}\ x == 0,\ [[\sigma|x:2]|y:2]\rangle \rightsquigarrow^* \langle \textbf{skip},\ [[\sigma|x:0]|y:1]\rangle$$

Para ello debemos encontrar una traza finita de ejecución $t_1 \rightsquigarrow t_2 \rightsquigarrow \cdots \rightsquigarrow t_n$ donde t_1 es $\langle x = y = 1; \textbf{repeat}\ x = x - y\ \textbf{until}\ x == 0,\ [[\sigma|x:2]|y:2]\rangle$ y t_n $\langle \textbf{skip},\ [[\sigma|x:0]|y:1]\rangle$

Comencemos con $t_1 \rightsquigarrow t_2$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle 1,\ [[\sigma|x:2]|y:2]\rangle \Downarrow_{\exp} \langle 1,\ [[\sigma|x:2]|y:2]\rangle}\ \text{NVal}}{\langle y=1,\ [[\sigma|x:2]|y:2]\rangle \Downarrow_{\exp} \langle 1,\ [[\sigma|x:2]|y:1]\rangle}\ \text{EAssgn}}{\langle x=y=1,\ [[\sigma|x:2]|y:2]\rangle \leadsto \langle \mathit{skip},\ [[\sigma|x:1]|y:1]\rangle}\ \text{Ass}}{\left\langle x=y=1;\underbrace{\textbf{repeat}\ x=x-y\ \textbf{until}\ x==0}_{c_1},\ [[\sigma|x:2]|y:2]\right\rangle \leadsto \underbrace{\langle \textbf{skip}; c_1,\ [[\sigma|x:1]|y:1]\rangle}_{t_2}}\ \text{Seq2}$$

- $t_2 \leadsto t_3$

$$\dfrac{}{\left\langle \textbf{skip};\textbf{repeat}\ \underbrace{x=x-y}_{comm}\ \textbf{until}\ \underbrace{x==0}_{bexp},\ [[\sigma|x:1]|y:1]\right\rangle \leadsto \underbrace{\left\langle c_1,\ \underbrace{[[\sigma|x:1]|y:1]}_{\sigma'}\right\rangle}_{t_3}}\ \text{Seq1}$$

- $t_3 \leadsto t_4$

$$\dfrac{}{\langle \textbf{repeat}\ x=x-y\ \textbf{until}\ x==0,\ \sigma'\rangle \leadsto \underbrace{\left\langle comm;\ \underbrace{\textbf{if}\ bexp\ \textbf{then skip else}\ c_1}_{c_2},\ \sigma'\right\rangle}_{t_4}}\ \text{Repeat}$$

- $t_4 \leadsto t_5$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle x,\ \sigma'\rangle \Downarrow_{\exp} \langle 1,\ \sigma'\rangle}\ \text{Var} \qquad \dfrac{}{\langle y,\ \sigma'\rangle \Downarrow_{\exp} \langle 1,\ \sigma'\rangle}\ \text{Var}}{\langle x -_b y,\ \sigma'\rangle \Downarrow_{\exp} \langle 0,\ \sigma'\rangle}\ \text{Minus}}{\langle x = x -_b y,\ \sigma'\rangle \leadsto \langle \textbf{skip},\ [\sigma|x:0]|y:1]\rangle}\ \text{Ass}}{\langle comm;\ \textbf{if}\ bexp\ \textbf{then skip else}\ c_1,\ \sigma'\rangle \leadsto \underbrace{\langle \textbf{skip}; c_2,\ [\sigma|x:0]|y:1]\rangle}_{t_5}}\ \text{Seq2}$$

- $t_5 \leadsto t_6$

$$\dfrac{}{\langle \textbf{skip};\ \textbf{if}\ bexp\ \textbf{then skip else}\ c_1,\ [\sigma|x:0]|y:1]\rangle \leadsto \underbrace{\left\langle c_2,\ \underbrace{[\sigma|x:0]|y:1]}_{\sigma''}\right\rangle}_{t_6}}\ \text{Seq}_1$$

- $t_6 \leadsto t_7$

$$\dfrac{\dfrac{\dfrac{}{\langle x,\ \sigma''\rangle \Downarrow_{\exp} \langle 0,\ \sigma''\rangle}\ \text{Var} \qquad \dfrac{}{\langle 0,\ \sigma''\rangle \Downarrow_{\exp} \langle 0,\ \sigma''\rangle}\ \text{NVal}}{\langle x==0,\ \sigma''\rangle \Downarrow_{\exp} \langle True,\ \sigma''\rangle}\ \text{Eq}}{\langle\ \textbf{if}\ bexp\ \textbf{then skip else}\ c_1,\ \sigma''\rangle \leadsto \underbrace{\langle \textbf{skip},\ [\sigma|x:0]|y:1]\rangle}_{t_7}}\ \text{If}_1$$

Con esto encontramos una traza finita de ejecución de comandos en un paso. La secuencia quedaria asi:

$$t_1 \leadsto t_2 \leadsto t_3 \leadsto t_4 \leadsto t_5 \leadsto t_6 \leadsto t_7$$

Por ser $\leadsto^*$ la clausura reflexivo transitiva de $\leadsto$ concluimos que:

$$t_1 \leadsto^* t_7$$

$$\langle x=y=1;\textbf{repeat}\ x=x-y\ \textbf{until}\ x==0,\ [[\sigma|x:2]|y:2]\rangle \leadsto^* \langle \textbf{skip},\ [\sigma|x:0]|y:1]\rangle$$

7. Ejercicio 10

Agregamos una producción en la gramática abstracta de LIS para el comando **for**

$$\begin{aligned}\textbf{comm} ::= {} & \textbf{skip} \\ & \mid \textit{var} \ = \ \textit{intexp} \\ & \mid \textbf{if } \textit{boolexp} \textbf{ then } \textit{comm} \textbf{ else } \textit{comm} \\ & \mid \textbf{repeat } \textit{comm} \textbf{ until } \textit{boolexp} \\ & \mid \textbf{for } (\textit{intexp};\ \textit{boolexp};\ \textit{intexp})\ \textit{comm}\end{aligned}$$

Extendemos la semántica operacional de comandos para incluir el comando **FOR**

$$\frac{\langle e_1,\ \sigma\rangle \Downarrow_{\text{exp}} \langle n,\ \sigma'\rangle}{\langle \textbf{for}(e_1; e_2; e_3)\ c,\ \sigma\rangle \rightsquigarrow \langle \textbf{if } e_2 \textbf{ then repeat } (c; \text{eval3}) \textbf{ until } \neg e_2 \textbf{ else skip},\ \sigma'\rangle}\ \text{FOR}$$

Donde eval3 es **if** $e_3 > 0$ **then skip else skip**