Trabajo Práctico N° 1

Análisis de Lenguajes de Programación

Mellino, Natalia   Farizano, Juan Ignacio

Ejercicio 1

A continuación extendemos las sintaxis abstracta y concreta para incluir asignaciones de variables como expresiones enteras y para escribir una secuencia de expresiones enteras.

Sintaxis Abstracta

$$
\begin{aligned}
intexp ::= &\ nat \mid var \mid -_u\ intexp \\
\mid &\ intexp\ +\ intexp \\
\mid &\ intexp\ -_b\ intexp \\
\mid &\ intexp\ \times\ intexp \\
\mid &\ intexp\ \div\ intexp \\
\mid &\ var = intexp \\
\mid &\ intexp, \ intexp \\
boolesxp ::= &\ \mathbf{true} \mid \mathbf{false} \\
\mid &\ intexp\ ==\ intexp \\
\mid &\ intexp\ \neq\ intexp \\
\mid &\ intexp\ <\ intexp \\
\mid &\ intexp\ >\ intexp \\
\mid &\ boolexp\ \vee\ boolexp \\
\mid &\ boolexp\ \wedge\ boolexp \\
\mid &\ \neg\ boolexp \\
comm ::= &\ \mathbf{skip} \\
\mid &\ var = intexp \\
\mid &\ comm;\ comm \\
\mid &\ \mathbf{if}\ boolexp\ \mathbf{then}\ comm\ \mathbf{else}\ comm \\
\mid &\ \mathbf{while}\ boolexp\ \mathbf{do}\ comm
\end{aligned}
$$

Sintaxis Concreta

$$
\begin{aligned}
digit &::= \text{'0'} \mid \text{'1'} \mid \ \ldots \ \mid \text{'9'} \\
letter &::= \text{'a'} \mid \ \ldots \ \mid \text{'Z'} \\
nat &::= digit \mid digit\ nat \\
var &::= letter \mid letter\ var \\
intexp &::= nat \\
&\mid var \\
&\mid \text{'-'}\ intexp \\
&\mid \ intexp\ \text{'+'}\ intexp \\
&\mid \ intexp\ \text{'-'}\ intexp \\
&\mid \ intexp\ \text{'*'}\ intexp \\
&\mid \ intexp\ \text{'/'}\ intexp \\
&\mid \ \text{'('}\ intexp\ \text{')'} \\
&\mid \ var\ \text{'='}\ intexp \\
&\mid \ intexp\ \text{','}\ intexp \\
boolesxp &::= \text{'true'} \mid \text{'false'} \\
&\mid \ intexp\ \text{'=='}\ intexp \\
&\mid \ intexp\ \text{'!='}\ intexp \\
&\mid \ intexp\ \text{'<'}\ intexp \\
&\mid \ intexp\ \text{'>'}\ intexp \\
&\mid \ boolexp\ \text{'\&\&'}\ boolexp \\
&\mid \ boolexp\ \text{'||'}\ boolexp \\
&\mid \ \text{'!'}\ boolexp \\
&\mid \ \text{'('}\ boolexp\ \text{')'} \\
com &::= \textbf{skip} \\
&\mid \ var\ \text{'='}\ intexp \\
&\mid \ comm\ \text{';'}\ comm \\
&\mid \ \text{'if'}\ \ boolexp\ \ \text{'\{'}\ \ comm\ \ \text{'\}'} \\
&\mid \ \text{'if'}\ \ boolexp\ \ \text{'\{'}\ \ comm\ \ \text{'\}'}\ \ \text{'else'}\ \ \text{'\{'}\ \ comm\ \ \text{'\}'} \\
&\mid \ \text{'while'}\ \ boolexp\ \ \text{'\{'}\ \ comm\ \ \text{'\}'}
\end{aligned}
$$

Ejercicio 2:

Para extender la realización de la sistaxis abstracta en Haskell, incluimos estos contructores en el tipo de datos parametrizado Exp a.

$$
\begin{aligned}
\text{EAssgn} &:: \text{Variable} \to \text{Exp Int} \to \text{Exp Int} \\
\text{ESeq} &:: \text{Exp Int} \to \text{Exp Int} \to \text{Exp Int}
\end{aligned}
$$

En el archivo *src/AST.hs* se encuentra reflejado este cambio.

Ejercicio 3:

Para implementar el parser, modificamos la gramática extendida en el ejercicio 1 en una que no presente ambigüedad.

Sintaxis Abstracta

$$
\begin{aligned}
intexp &::= intexp,\ intexp1 \mid intexp1 \\
intexp1 &::= var = intexp1 \mid intexp2 \\
intexp2 &::= intexp2\ +\ intexp3 \mid intexp2\ -_b\ intexp3 \mid intexp2 \\
intexp3 &::= intexp3\ \times\ intexp4 \mid intexp3\ \div\ intexp4 \mid intexp4 \\
intexp4 &::= \ -_u\ intexp4 \mid nat \mid var \mid (intexp)
\end{aligned}
$$

$$
\begin{aligned}
boolexp &::= boolexp\ \vee\ boolexp1 \mid boolexp1 \\
boolexp1 &::= boolexp1\ \wedge\ boolexp2 \mid boolexp2 \\
boolexp2 &::= \neg\, boolexp2 \mid boolexp3 \\
boolexp3 &::= \mathbf{true} \mid \mathbf{false} \\
&\mid\ intexp == intexp \\
&\mid\ intexp \neq intexp \\
&\mid\ intexp > intexp \\
&\mid\ intexp < intexp \\
&\mid\ (boolexp)
\end{aligned}
$$

$$
\begin{aligned}
comm &::= comm;\ comm1 \mid comm1 \\
comm1 &::= \mathbf{skip} \\
&\mid\ var = intexp \\
&\mid\ \mathbf{if}\ boolexp\ \mathbf{then}\ comm\ \mathbf{else}\ comm \\
&\mid\ \mathbf{while}\ boolexp\ \mathbf{do}\ comm
\end{aligned}
$$

Sintaxis Concreta

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

$$
\begin{aligned}
\mathit{intexp} &::= \mathit{intexp}\ \text{','}\ \mathit{intexp1} \mid \mathit{intexp1} \\
\mathit{intexp1} &::= \mathit{var}\ \text{'='}\ \mathit{intexp1} \mid \mathit{intexp2} \\
\mathit{intexp2} &::= \mathit{intexp2}\ \text{'+'}\ \mathit{intexp3} \mid \mathit{intexp2}\ \text{'-'}\ \mathit{intexp3} \mid \mathit{intexp2} \\
\mathit{intexp3} &::= \mathit{intexp3}\ \text{'*'}\ \mathit{intexp4} \mid \mathit{intexp3}\ \text{'/'}\ \mathit{intexp4} \mid \mathit{intexp4} \\
\mathit{intexp4} &::= \text{'-'}\ \mathit{intexp4} \mid \mathit{nat} \mid \mathit{var} \mid \text{'('}\mathit{intexp}\text{')'}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{boolexp} &::= \mathit{boolexp}\ \text{'||'}\ \mathit{boolexp1} \mid \mathit{boolexp1} \\
\mathit{boolexp1} &::= \mathit{boolexp1}\ \text{'\&\&'}\ \mathit{boolexp2} \mid \mathit{boolexp2} \\
\mathit{boolexp2} &::= \text{'!='}\ \mathit{boolexp2} \mid \mathit{boolexp3} \\
\mathit{boolexp3} &::= \mathbf{true} \mid \mathbf{false} \\
&\quad \mid\ \mathit{intexp}\ \text{'=='}\ \mathit{intexp} \\
&\quad \mid\ \mathit{intexp}\ \text{'!='}\ \mathit{intexp} \\
&\quad \mid\ \mathit{intexp}\ \text{'<'}\ \mathit{intexp} \\
&\quad \mid\ \mathit{intexp}\ \text{'>'}\ \mathit{intexp} \\
&\quad \mid\ (\mathit{boolexp})
\end{aligned}
$$

$$
\begin{aligned}
\mathit{comm} &::= \mathit{comm}\ \text{';'}\ \mathit{comm1} \mid \mathit{comm1} \\
\mathit{comm1} &::= \mathbf{skip} \\
&\quad \mid\ \mathit{var}\ \text{'='}\ \mathit{intexp} \\
&\quad \mid\ \text{'if'}\ \ \mathit{boolexp}\ \ \text{'\{'}\ \ \mathit{comm}\ \ \text{'\}'} \\
&\quad \mid\ \text{'if'}\ \ \mathit{boolexp}\ \ \text{'\{'}\ \ \mathit{comm}\ \ \text{'\}'}\ \ \text{'else'}\ \ \text{'\{'}\ \ \mathit{comm}\ \ \text{'\}'} \\
&\quad \mid\ \text{'while'}\ \ \mathit{boolexp}\ \ \text{'\{'}\ \ \mathit{comm}\ \ \text{'\}'}
\end{aligned}
$$

Ejercicio 4:

A continuación modificamos la semántica big-step para incluir la asignación de variables como expresiones y para las secuencias de expresiones.

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

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

Ejercicio 5:

Asumimos que la relación $\Downarrow_{exp}$ es determinista y procedemos por inducción en la última regla de derivación. Queremos probar : $c \rightsquigarrow c'$, $c \rightsquigarrow c'' \Rightarrow c' = c''$

• Si $c \rightsquigarrow c'$ usando como última regla ASS:
c tiene la forma $\langle v = e, \sigma \rangle$ y tenemos la premisa $\langle e, \sigma \rangle \Downarrow_{exp} \langle n, \sigma' \rangle$, inmediatamente debido a la regla ASS obtenemos que $c' = \langle \textbf{skip}, [\sigma' \mid v : n] \rangle$.
Supongamos entonces, que esta relación no es determinista, es decir que $c' \neq c''$. Por la forma que tiene c observemos que la única regla que podemos usar en la derivación $c \rightsquigarrow c''$ es la regla ASS, entonces tenemos que: $\langle v = e, \sigma \rangle \rightsquigarrow \langle \textbf{skip}, [\sigma'' \mid v : n'] \rangle$ con la premisa $\langle e, \sigma \rangle \Downarrow_{exp} \langle n', \sigma'' \rangle$ Como $c' \neq c''$ vemos que $\sigma' \neq \sigma'' \vee n \neq n''$. Esto es una contradicción ya que por determinismo de la relación $\Downarrow_{exp}$ necesariamente debe ocurrir que $\sigma' = \sigma'' \wedge n = n''$.

$\therefore\ c' = c''$

• Si $c \rightsquigarrow c'$ usando como última regla SEQ$_1$:
c tiene la forma $\langle \textbf{skip}; c_1, \sigma \rangle$ y tenemos que $c' = \langle c_1, \sigma \rangle$. Por la forma que tiene c, observemos que no es posible aplicar ninguna otra regla de inferencia, por lo tanto en la derivación $c \rightsquigarrow c''$ se tiene que $c'' = \langle c_1, \sigma \rangle$.

$\therefore\ c' = c''$

• Si $c \rightsquigarrow c'$ usando como última regla SEQ$_2$:
Tenemos entonces que c tiene la forma: $\langle c_0; c_1, \sigma \rangle$ y además: $\langle c_0,\ \sigma \rangle \rightsquigarrow \langle c_0', \sigma' \rangle$. Ahora, analicemos que pasa en $c \rightsquigarrow c''$: ¿qué reglas podemos aplicar?

- No podemos aplicar SEQ$_1$ ya que esto nos diría que $c_0 = \textbf{skip}$ y esto nos contradice $\langle c_0, \sigma \rangle \rightsquigarrow \langle c_0',\ \sigma \rangle$ ya que si $c_0 = \textbf{skip}$ el estado no debería modificarse.
- En cuanto a las demás reglas, es claro que por la forma que tiene c no es posible aplicarlas.

Por lo tanto, la única regla que podemos aplicar es SEQ$_2$. Si suponemos que $c' \neq c''$ tenemos entonces que c tiene la forma: $\langle c_0; c_1, \sigma \rangle$ y además: $\langle c_0, \sigma \rangle \rightsquigarrow \langle c_0'', \sigma'' \rangle$. Luego sigue que c'' tiene la forma: $\langle c_0''; c1, \sigma'' \rangle$. Sin embargo, observemos que $\langle c_0, \sigma \rangle \rightsquigarrow \langle c_0',\ \sigma' \rangle$ es una subderivación y por lo tanto vale nuestra Hipótesis Inductiva, es decir necesariamente:

$$\langle c_0,\ \sigma \rangle \rightsquigarrow \langle c_0', \sigma' \rangle \wedge \langle c_0,\ \sigma \rangle \rightsquigarrow \langle c_0'',\ \sigma'' \rangle \Rightarrow \langle c_0', \sigma' \rangle = \langle c_0'',\ \sigma'' \rangle$$

Por lo tanto, $c_0' = c_0''$ y $\sigma' = \sigma''$.

$\therefore\ c' = c''$

• Si $c \rightsquigarrow c'$ usando como última regla IF$_1$:
c tiene la forma $\langle \textbf{if}\ \mathrm{b}\ \textbf{then}\ c_0\ \textbf{else}\ c_1,\ \sigma \rangle$ y además $\langle b, \sigma \rangle \Downarrow_{exp} \langle \textbf{true}, \sigma' \rangle$. Entonces c' tiene la forma $\langle c_0, \sigma' \rangle$. ¿Qué pasa en el caso de $c \rightsquigarrow c''$?. Sabemos que por la forma que tiene c, por el deterministo de $\Downarrow_{exp}$ y por el hecho de que $\langle b, \sigma \rangle \Downarrow_{exp} \langle \textbf{true}, \sigma' \rangle$ sólo podemos aplicar la regla IF$_1$. Inmediatamente de esta conclusión surge que $c' = c''$.

• Si $c \rightsquigarrow c'$ usando como última regla IF$_2$:

(este caso es análogo a la regla IF$_1$)

• Si $c \rightsquigarrow c'$ usando como última regla WHILE$_1$: tenemos entonces que

- c tiene la forma $\langle \textbf{while} \text{ b } \textbf{do } c_0, \sigma\rangle$.
- $\langle b, \sigma\rangle \Downarrow_{exp} \langle \textbf{true}, \sigma'\rangle$
- $c' = \langle c0; \textbf{while} \text{ b } \textbf{do } c_0, \sigma'\rangle$

Luego en $c \rightsquigarrow c''$ ocurre que por la forma de c y por el determinismo de $\Downarrow_{exp}$, la única regla que podemos usar es WHILE_1. Por estas dos cosas surge inmediatamente que $c' = c''$.

- Si $c \rightsquigarrow c'$ usando como última regla WHILE_2: tenemos entonces que
  - c tiene la forma $\langle \textbf{while} \text{ b } \textbf{do } c_0,\ \sigma\rangle$.
  - $\langle b, \sigma\rangle \Downarrow_{exp} \langle \textbf{false}, \sigma'\rangle$
  - $c' = \langle \textbf{skip}, \sigma'\rangle$

Luego en $c \rightsquigarrow c''$ ocurre que por la forma de c y por el determinismo de $\Downarrow_{exp}$, la única regla que podemos usar es WHILE_2. Por estas dos cosas surge inmediatamente que $c' = c''$.

Por lo tanto, concluimos que la relación $\rightsquigarrow$ es determinista.

Ejercicio 6:

Utilizamos seis árboles distintos para realizar la derivación en varios pasos. Al final de la misma uniremos todos los resultados en un único árbol utilizando las reglas de la clausura reflexivo-transitiva.
En el primer árbol probamos $t_1 \rightsquigarrow t_2$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle 1,\ [[\sigma|\ x:\ 2]\,|\ y:\ 2]\rangle \Downarrow_{\text{exp}} \langle 1,\ [[\sigma|\ x:\ 2]\,|\ y:\ 2]\rangle}\ \textsc{NVal}}{\langle y = 1,\ [[\sigma|\ x:\ 2]\,|\ y:\ 2]\rangle \Downarrow_{\text{exp}} \langle 1,\ [[\sigma|\ x:\ 2]\,|\ y:\ 1]\rangle}\ \textsc{EAssgn}}{\langle x = y = 1,\ [[\sigma|\ x:\ 2]\,|\ y:\ 2]\rangle \rightsquigarrow \langle \textbf{skip},\ [[\sigma|\ x:\ 1]\,|\ y:\ 1]\rangle}\ \textsc{Ass}}{\underbrace{\left\langle x = y = 1;\ \underbrace{\textbf{while } x > 0 \textbf{ do } x = x -_b y}_{c_1},\ [[\sigma|\ x:\ 2]\,|\ y:\ 2]\right\rangle}_{t_1} \rightsquigarrow \underbrace{\langle \textbf{skip};\ c_1,\ [[\sigma|\ x:\ 1]\,|\ y:\ 1]\rangle}_{t_2}}\ \textsc{Seq}_2$$

En este segundo árbol probamos $t_2 \rightsquigarrow t_3$

$$\dfrac{}{\underbrace{\left\langle \textbf{skip};\ \underbrace{\textbf{while } x > 0 \textbf{ do } x = x -_b y}_{c_1},\ [[\sigma|\ x:\ 1]\,|\ y:\ 1]\right\rangle}_{t_2} \rightsquigarrow \underbrace{\langle c_1,\ [[\sigma|\ x:\ 1]\,|\ y:\ 1]\rangle}_{t_3}}\ \textsc{Seq}_1$$

En este tercer árbol probamos $t_3 \rightsquigarrow t_4$.

$$\dfrac{\dfrac{\dfrac{}{\langle x,\ \sigma'\rangle \Downarrow_{\text{exp}} \langle 1,\ \sigma'\rangle}\ \textsc{Var} \qquad \dfrac{}{\langle 0,\ \sigma'\rangle \Downarrow_{\text{exp}} \langle 0,\ \sigma'\rangle}\ \textsc{NVal}}{\langle x > 0,\ \sigma'\rangle \Downarrow_{\text{exp}} \langle \textbf{true},\ \sigma'\rangle}\ \textsc{Gt}}{\underbrace{\left\langle \underbrace{\textbf{while } x > 0 \textbf{ do } x = x -_b y}_{c_1},\ \underbrace{[[\sigma|\ x:\ 1]\,|\ y:\ 1]}_{\sigma'}\right\rangle}_{t_3} \rightsquigarrow \underbrace{\langle x = x -_b y;\ c_1,\ \sigma'\rangle}_{t_4}}\ \textsc{While}_1$$

En este cuarto árbol probamos $t_4 \rightsquigarrow t_5$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle x,\ \sigma' \rangle \Downarrow_{\text{exp}} \langle 1,\ \sigma' \rangle}\ \textsc{Var} \qquad \dfrac{}{\langle y,\ \sigma' \rangle \Downarrow_{\text{exp}} \langle 1,\ \sigma' \rangle}\ \textsc{Var}}{\langle x -_b y,\ \sigma' \rangle \Downarrow_{\text{exp}} \langle 0,\ \sigma' \rangle}\ \textsc{Minus}}{\langle x = x -_b y,\ \sigma' \rangle \rightsquigarrow \langle \textbf{skip},\ \sigma'' \rangle}\ \textsc{Ass}}{\underbrace{\left\langle x = x -_b y;\ \underbrace{\textbf{while}\ x > 0\ \textbf{do}\ x = x -_b y}_{c_1},\ \underbrace{[[\sigma|\ x:\ 1]\ |\ y:\ 1]}_{\sigma'} \right\rangle}_{t_4} \rightsquigarrow \underbrace{\left\langle \textbf{skip};\ c_1,\ \underbrace{[[\sigma|\ x:\ 0]\ |\ y:\ 1]}_{\sigma''} \right\rangle}_{t_5}}\ \textsc{Seq}_2$$

En este quinto árbol probamos $t_5 \rightsquigarrow t_6$.

$$\dfrac{}{\underbrace{\left\langle \textbf{skip};\ \underbrace{\textbf{while}\ x > 0\ \textbf{do}\ x = x -_b y}_{c_1},\ [[\sigma|\ x:\ 0]\ |\ y:\ 1] \right\rangle}_{t_5} \rightsquigarrow \underbrace{\langle c_1,\ [[\sigma|\ x:\ 0]\ |\ y:\ 1] \rangle}_{t_6}}\ \textsc{Seq}_1$$

Por último probamos $t_6 \rightsquigarrow t_7$.

$$\dfrac{\dfrac{\dfrac{}{\langle x,\ \sigma' \rangle \Downarrow_{\text{exp}} \langle 0,\ \sigma' \rangle}\ \textsc{Var} \qquad \dfrac{}{\langle 0,\ \sigma' \rangle \Downarrow_{\text{exp}} \langle 0,\ \sigma' \rangle}\ \textsc{NVal}}{\langle x > 0,\ \sigma' \rangle \Downarrow_{\text{exp}} \langle \textbf{false},\ \sigma' \rangle}\ \textsc{Gt}}{\underbrace{\left\langle \textbf{while}\ x > 0\ \textbf{do}\ x = x -_b y,\ \underbrace{[[\sigma|\ x:\ 0]\ |\ y:\ 1]}_{\sigma'} \right\rangle}_{t_6} \rightsquigarrow \underbrace{\langle \textbf{skip},\ \sigma' \rangle}_{t_7}}\ \textsc{While}_2$$

Ahora, utilizando las siguientes reglas de la clausura reflexivo transitiva:

$$\dfrac{t \rightsquigarrow t'}{t \rightsquigarrow^* t'}\ \text{E}_1 \qquad \dfrac{t \rightsquigarrow^* t' \quad t' \rightsquigarrow^* t''}{t \rightsquigarrow^* t''}\ \text{E}_2$$

probamos que $t_1 \rightsquigarrow^* t_7$:

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

$\therefore t_1 \rightsquigarrow^* t_7$

Ejercicio 10:

A continuación agregamos la regla de producción para el comando **for** a la gramática y extendemos la semántica operacional.

Regla de producción en la gramática abstracta de LIS

$$\begin{aligned} \mathit{comm} ::=\ & \textbf{skip} \\ |\ & \mathit{var} = \mathit{intexp} \\ |\ & \mathit{comm};\ \mathit{comm} \\ |\ & \textbf{if}\ \mathit{boolexp}\ \textbf{then}\ \mathit{comm}\ \textbf{else}\ \mathit{comm} \\ |\ & \textbf{while}\ \mathit{boolexp}\ \textbf{do}\ \mathit{comm} \\ |\ & \textbf{for}\ (\mathit{intexp}\ ;\ \mathit{boolexp}\ ;\ \mathit{intexp})\ \mathit{comm} \end{aligned}$$

Semántica operacional para el comando for

$$\frac{\langle e_1,\ \sigma\rangle \Downarrow_{\text{exp}} \langle n,\ \sigma'\rangle}{\langle \textbf{for}(e_1;\ e_2\ ; e_3)\ c,\ \sigma\rangle \rightsquigarrow \langle \textbf{while}\ e_2\ \textbf{do}\ (c;\ \textbf{if}\ e_3 == 0\ \textbf{then skip else skip}),\ \sigma'\rangle}\ \textsc{For}$$