Trabajo Práctico 1

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

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

Sintaxis Concreta

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

Ejercicio 2:

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

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

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

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

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

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

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

Ejercicio 4:

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 $c' = \langle \textbf{skip}, [\sigma' | v : n]\rangle$.
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: $\langle v = e, \sigma\rangle \rightsquigarrow \langle \textbf{skip}, [\sigma'' | v : n']\rangle$ 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 *SEQ1*:
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 *SEQ2*:
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 *SEQ1* 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 $c_0 = skip$ el estado σ no debería pasar a ser σ'.
- 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 *SEQ2*. 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:

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

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

$\therefore\ c' = c''$

• Si $c \rightsquigarrow c'$ usando como última regla *IF1*:
c tiene la forma $\langle \textbf{if}\ \text{b}\ \textbf{then}\ c_0\ \textbf{else}\ c_1,\ \sigma\rangle$ 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 y por el hecho de que $\langle b, \sigma\rangle \Downarrow_{exp} \langle \textbf{true}, \sigma'\rangle$ sólo podemos aplicar la regla *IF1*. Recordemos que la relación $\Downarrow_{exp}$ es determinista y por eso podemos decir que sólo podamos aplicar *IF1*. inmediatamente de esta conclusión surge que $c' = c''$.

• Si $c \rightsquigarrow c'$ usando como última regla *IF2*:

(este caso es análogo a la regla *IF1*)

• Si $c \rightsquigarrow c'$ usando como última regla *WHILE1*: 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 *WHILE1*. Por estas dos cosas surge inmediatamente que $c' = c''$.

• Si $c \rightsquigarrow c'$ usando como última regla *WHILE2*: tenemos entonces que

- c tiene la forma $\langle \textbf{while } \mathrm{b} \textbf{ do } c_0,\ \sigma \rangle$.
- $\langle b, \sigma \rangle \Downarrow_{exp} \langle \textbf{false}, \sigma' \rangle$
- $c' = \langle \textbf{skip}, \sigma' \rangle$

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 *WHILE1*. Por estas dos cosas surge inmediatamente que $c' = c''$.

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

Ejercicio 6: