Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingenieria Y Agrimensura

Análisis de Lenguajes de Programación R322

Trabajo Práctico I - Informe

Alumnos:

Tomas Octavio Castagnino

Ernesto Jesús Savio

Julián Paz

Ejercicio 1.

La sintaxis abstracta de este lenguaje es la siguiente:

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

$$
\begin{aligned}
\mathit{boolexp} ::= {} & \mathbf{true} \mid \mathbf{false} \\
& \mid \mathit{intexp} == \mathit{intexp} \\
& \mid \mathit{intexp} \ != \mathit{intexp} \\
& \mid \mathit{intexp} > \mathit{intexp} \\
& \mid \mathit{intexp} < \mathit{intexp} \\
& \mid \mathit{boleanexp} \wedge \mathit{boleanexp} \\
& \mid \mathit{boleanexp} \vee \mathit{boleanexp} \\
& \mid \neg \ \mathit{boleanexp}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{comm} ::= {} & \mathbf{skip} \\
& \mid \mathit{var} = \mathit{intexp} \\
& \mid \mathit{comm}; \ \mathit{comm} \\
& \mid \mathbf{if} \ \mathit{boleanexp} \ \mathbf{then\ else} \ \mathit{comm} \\
& \mid \mathbf{repeat} \ \mathit{comm} \ \mathbf{until} \ \mathit{boolexp}
\end{aligned}
$$

La sintaxis concreta de este lenguaje es la siguiente:

$$
\begin{array}{lcl}
\mathit{digit} & ::= & \textbf{'0'} \mid \textbf{'1'} \mid \cdots \mid \textbf{'9'} \\
\mathit{letter} & ::= & \textbf{'a'} \mid \cdots \mid \textbf{'Z'} \\
\mathit{nat} & ::= & \mathit{digit} \mid \mathit{digit}\ \mathit{nat} \\
\mathit{var} & ::= & \mathit{letter} \mid \mathit{letter}\ \mathit{var} \\
\mathit{intexp} & ::= & \mathit{nat} \\
 & \mid & \mathit{var} \\
 & \mid & \text{'}-\text{'}\ \ \mathit{intexp} \\
 & \mid & \mathit{var}\ \text{'}{+}{+}\text{'} \\
 & \mid & \mathit{intexp}\ \text{'}+\text{'}\ \ \mathit{intexp} \\
 & \mid & \mathit{intexp}\ \text{'}-\text{'}\ \ \mathit{intexp} \\
 & \mid & \mathit{intexp}\ \text{'}*\text{'}\ \ \mathit{intexp} \\
 & \mid & \mathit{intexp}\ \text{'}/\text{'}\ \mathit{intexp} \\
 & \mid & \text{'}(\text{'}\ \mathit{intexp}\ \text{'})\text{'} \\
\mathit{boolexp} & ::= & \textbf{'true'} \mid \textbf{'false'} \\
 & \mid & \mathit{intexp}\ \text{'}{=}{=}\text{'}\ \ \mathit{intexp} \\
 & \mid & \mathit{intexp}\ \text{'}!{=}\text{'}\ \ \mathit{intexp} \\
 & \mid & \mathit{intexp}\ \text{'}<\text{'}\ \ \mathit{intexp} \\
 & \mid & \mathit{intexp}\ \text{'}>\text{'}\ \ \mathit{intexp} \\
 & \mid & \mathit{intexp}\ \text{'}\&\&\text{'}\ \mathit{intexp} \\
 & \mid & \mathit{intexp}\ \text{'}||\text{'}\ \mathit{intexp} \\
 & \mid & \text{'}!\text{'}\ \mathit{boolexp} \\
 & \mid & \text{'}(\text{'}\ \mathit{boolexp}\ \text{'})\text{'}
\end{array}
$$

$$
\begin{array}{lcl}
\mathit{comm} & ::= & \textbf{skip} \\
 & \mid & \text{'}\textbf{case}\text{'}\ \text{'}\{\text{'}\ \mathit{caseComm}\ \text{'}\}\text{'} \\
 & \mid & \mathit{var}\ \text{'}={}\text{'}\ \ \mathit{intexp} \\
 & \mid & \mathit{comm}\ \text{';'}\ \ \mathit{comm} \\
 & \mid & \textbf{'if'}\ \mathit{boolexp}\ \text{'}\{\text{'}\ \mathit{comm}\ \text{'}\}\text{'} \\
 & \mid & \textbf{'if'}\ \mathit{boolexp}\ \text{'}\{\text{'}\ \mathit{comm}\ \text{'}\}\text{'}\ \textbf{'else'}\ \text{'}\{\text{'}\ \mathit{comm}\ \text{'}\}\text{'} \\
 & \mid & \textbf{'repeat'}\ \text{'}\{\text{'}\ \mathit{comm}\ \text{'}\}\text{'}\ \textbf{'until'}\ \mathit{boolexp}
\end{array}
$$

$$
\begin{array}{lcl}
\mathit{caseComm} & = & \mathit{boolexp}\ \text{':'}\ \ \text{'}\{\text{'}\ \mathit{comm}\ \text{'}\}\text{'}\ \mathit{caseComm} \\
 & \mid & \epsilon
\end{array}
$$

Ejercicio 3.

Para poder hacer los parsers tuvimos que pensar en como representar el orden de precedencia. Para ello fue útil pensar que nuestra sintaxis concreta tiene las siguientes reglas:

$intExp := (intExp\ \text{'+'} \mid intExp\ \text{'-'} \mid \epsilon)\ intTerm$
$intTerm := (intTerm\ \text{'*'} \mid intTerm\ \text{'/'} \mid \epsilon)\ intUnary$
$intUnary := \text{'-'}\ intUnary \mid intAtom$
$intAtom := nat \mid var\ (\text{'++'} \mid \epsilon) \mid \text{'('}\ intExp\ \text{')'}$

$boolExp := (boolExp\ \text{'||'} \mid \epsilon)\ boolTerm$
$boolTerm := (\ boolTerm\ \text{'\&\&'} \mid \epsilon)\ boolUnary$
$boolUnary := \text{'!'}\ boolUnary \mid boolAtom$
$boolAtom := \text{'}\mathbf{true}\text{'} \mid \text{'}\mathbf{false}\text{'} \mid \text{'('}\ boolExp\ \text{')'} \mid boolComp$
$boolComp := intExp\ (\text{'=='}\ intExp \mid \text{'!='}\ intExp \mid \text{'<'}\ intExp \mid \text{'>'}\ intExp)$

$commExp := (commExp\ \text{';'} \mid \epsilon)\ commAtom$
$commAtom := \text{'}\mathbf{skip}\text{'} \mid commIf \mid commRepeat \mid commCase \mid commAss$

$commIf := \text{'}\mathbf{if}\text{'}\ boolExp\ \text{'}\mathbf{then}\text{'}\ commExp\ (\ \text{'}\mathbf{else}\text{'}\ commExp \mid \epsilon)$
$commRepeat := \text{'}\mathbf{repeat}\text{'}\ \text{'\{'}\ comm\ \text{'\}'}\ \mathbf{\text{'until'}}\ boolexp$
$commCase := \text{'}\mathbf{case}\text{'}\ \text{'\{'}\ caseComm\ \text{'\}'}$
$commAss := var\ \text{'='}\ intexp$
$caseComm = boolexp\ \text{':'}\ \text{'\{'}\ comm\ \text{'\}'}\ caseComm$

Ejercicio 4. La regla de derivación para este ejercicio es:

$$\frac{x \in Dom(\sigma)}{< x{+}{+}, \sigma > \Downarrow_{exp} < x, [\sigma \mid x : \sigma(x) + 1] >}\ \text{VarInc}$$

Ejercicio 5.

Para demostrar que la relacion de evaluación de un paso $\rightsquigarrow$ es determinista tenemos que demostrar que si $\langle c, \sigma\rangle \rightsquigarrow \langle c_1, \sigma_1\rangle$ y $\langle c, \sigma\rangle \rightsquigarrow \langle c_2, \sigma_2\rangle$ entonces $\langle c_1, \sigma_1\rangle = \langle c_2, \sigma_2\rangle$. Para eso vamos a hacer induccion sobre $\langle c, \sigma\rangle \rightsquigarrow \langle c_1, \sigma_1\rangle$. Hacemos analisis por casos sobre la ultima regla aplicada en la derivacion. Pudiendo usar la Hipotesis Inductiva en los casos que contiene subderivaciones.

- *Ass*. Entonces la forma de c es $v = e$ y si $\langle e, \sigma\rangle \Downarrow_{exp} \langle n, \sigma_1\rangle$ sabemos que $\langle c_1, \sigma_1\rangle = \langle \textbf{skip}, [\sigma_1 | v : n]\rangle$. En la derivacion de $\langle c, \sigma\rangle \rightsquigarrow \langle c_2, \sigma_2\rangle$ vemos que la ultima regal aplicada no puede ser otra que no sea *Ass* por la forma de c. Sabiendo que $\Downarrow_{exp}$ es determinsta sabemos que $\sigma_1 = \sigma_2$. Como la regla usa ese *Ass* y $\Downarrow_{exp}$ es determinista concluimos entonces que $\langle c_1, \sigma_1\rangle = \langle c_2, \sigma_2\rangle$.

- Seq_1. Entonces la forma de c es $\textbf{skip}; c'$. En la derivacion de $\langle c, \sigma\rangle \rightsquigarrow \langle c_2, \sigma_2\rangle$ vemos que la ultima regla aplicada no puede ser Seq_2 porque **skip** no deriva a nada entonces la ultima regla usada es Seq_1. Concluimos entonces que $\langle c_1, \sigma_1\rangle = \langle c_2, \sigma_2\rangle$.

- Seq_2. Entonces la forma de c es $co_0; co_1$ y $\langle co_0, \sigma\rangle \rightsquigarrow \langle co1'_0, \sigma 1'\rangle$. O sea, $\langle c_1, \sigma_1\rangle = \langle co1'_0; co_1, \sigma 1'\rangle$.

  En la derivacion de $\langle c, \sigma\rangle \rightsquigarrow \langle c_2, \sigma_2\rangle$ vemos que la ultima regla aplicada no puede ser Seq_1 porque para usar esta regla co_0 deberia ser **skip** y sabemos que co_0 deriva, por lo que entonces $co_0 \neq \textbf{skip}$.

  Ahora, sabemos que para la derivacion de $\langle c, \sigma\rangle \rightsquigarrow \langle c_2, \sigma_2\rangle$ se usa la regla Seq_2. Y teniendo que $\langle co_0, \sigma\rangle \rightsquigarrow \langle co2'_0, \sigma 2'\rangle$ queda que $\langle c_2, \sigma_2\rangle = \langle co2'_0; co_1, \sigma 2'\rangle$.

  Como co_0 es una subderivacion de c por Hipotesis Inductiva tenemos que su derivacion es determinista. Esto es que como $\langle co_0, \sigma\rangle \rightsquigarrow \langle co1'_0, \sigma 1'\rangle$ y $\langle co_0, \sigma\rangle \rightsquigarrow \langle co2'_0, \sigma 2'\rangle$ entonces $\langle co1'_0, \sigma 1'\rangle = \langle co2'_0, \sigma 2'\rangle$.

  Por lo tanto, $\langle c_1, \sigma_1\rangle = \langle co1'_0; co_1, \sigma 1'\rangle = \langle co2'_0; co_1, \sigma 2'\rangle = \langle c_2, \sigma_2\rangle$. Quedando entonces que $\langle c_1, \sigma_1\rangle = \langle c_2, \sigma_2\rangle$. En consecuencia la regla $Seq2$ es determinista.

- If_1. Supongamos que la última regla de derivación que se aplicó fue If_1. Por lo tanto la derivación tendrá la siguiente forma:

$$\frac{\langle b, \sigma\rangle \Downarrow_{exp} \langle true, \sigma'\rangle}{\langle if\ b\ then\ \hat{c}\ else\ \hat{c}', \sigma\rangle \rightsquigarrow \langle \hat{c}, \sigma'\rangle}\ If_1$$

  Esto nos lleva a decir que $c = (if\ \hat{b}\ then\ \hat{c}\ else\ \hat{c}')$. Ahora veamos que sucede en la derivación $\langle c, \sigma\rangle \rightsquigarrow \langle c_2, \sigma_2\rangle$

  Como c es un $if\ then\ else$ solo es posible aplicar las reglas If_1 o If_2.

  Por otro lado, como $\Downarrow_{exp}$ es determinista resulta que:

$$\langle b, \sigma\rangle \Downarrow_{exp} \langle true, \sigma'\rangle\ (1)$$

  Luego como el estado inicial en la segunda derivación es el mismo que la primer derivación y vale (1), resulta que solo es posible aplicar la regla If_1, por lo tanto tenemos que $\langle c_2, \sigma_2\rangle$ tiene la forma:

$$\langle c_2, \sigma_2\rangle = \langle \hat{c}, \sigma'\rangle = \langle c_1, \sigma_1\rangle$$

  Como ambas derivaciones derivan a lo mismo, resulta que la regla If_1 debe ser determinista.

- If_2. (Análogo a If_1)

- *Repeat*. Supongamos que la ultima regla de evaluación aplicada es *Repeat*. Por la forma de la misma, tenemos que: $\langle c, \sigma\rangle = \langle repeat\ c\ until\ b, \sigma_1\rangle$ y $\langle c_1, \sigma\rangle = \langle c; if\ b\ then\ skip\ else\ repeat\ c\ until\ b, \sigma\rangle$

  Como c tiene la forma *repeat until*, no queda otra opción que aplicar la regla *Repeat* en la segunda derivación.

Al aplicar esta regla obtenemos que:

$$\langle c_2, \sigma_2 \rangle = \langle c; if\ b\ then\ skip\ else\ repeat\ c\ until\ b, \sigma \rangle = \langle c_1, \sigma_1 \rangle$$

$\therefore \langle c_1, \sigma_1 \rangle = \langle c_2, \sigma_2 \rangle$. Por lo tanto la regla *Repeat* es determinista.

Ejercicio 6. Denotemos c_1 y c_2 al primer y segundo programa respectivamente. Para ver que c_1 y c_2 son semáticamente equivalentes si para todo estado $\sigma \in \Sigma$ vale que:

$$\langle c_1, \sigma\rangle \to^* \langle skip, \sigma'\rangle \text{ si y solo si } \langle c_2, \sigma\rangle \to^* \langle skip, \sigma'\rangle$$

Sea $\sigma \in \Sigma$ un estado cualquiera. Luego tenemos que:

Programa A:

$$\dfrac{\dfrac{\dfrac{x \in Dom(\sigma)}{\langle x, \sigma\rangle \ \Downarrow_{exp} \ \langle \sigma(x), \sigma\rangle}\,\text{Var} \quad \dfrac{}{\langle 1, \sigma\rangle \ \Downarrow_{exp} \ \langle 1, \sigma\rangle}\,\text{Const}}{\langle x+1, \sigma\rangle \ \Downarrow_{exp} \ \langle \sigma(x)+1, \sigma\rangle}}{\langle x = x+1, \sigma\rangle \ \to \ \langle skip, [\sigma \mid x : \sigma(x)+1\ ]\rangle}\,\text{Ass}$$

Luego,

$$\dfrac{\langle x = x+1, \sigma\rangle \to \langle skip, [\sigma \mid x : \sigma(x)+1]\rangle}{\langle x = x+1; y = x, \sigma\rangle \ \to \ \langle skip; y = x, [\sigma \mid x : \sigma(x)+1\ ]\rangle}\,\text{Seq2}$$

Luego,

$$\dfrac{}{\langle skip; y = x, [\sigma \mid x : \sigma(x)+1]\rangle \ \to \ \langle y = x, [\sigma \mid x : \sigma(x)+1\ ]\rangle}\,\text{Seq1}$$

Por último tenemos que,

$$\dfrac{\dfrac{x \in Dom([\sigma \mid x : \sigma(x)+1])}{\langle x, [\sigma \mid x : \sigma(x)+1\ ]\rangle \ \Downarrow_{exp} \ \langle \sigma(x)+1, [\sigma \mid x : \sigma(x)+1\ ]\rangle}\,\text{Var}}{\langle y = x, [\sigma \mid x : \sigma(x)+1\ ]\rangle \ \to \ \langle skip, [\ [\sigma \mid x : \sigma(x)+1\ ] \mid y : \sigma(x)+1]\rangle}\,\text{Ass}$$

Por lo tanto tenemos que:

$$\langle y = x++, \sigma\rangle \ \to^* \ \langle skip, [\ [\sigma \mid x : \sigma(x)+1\ ] \mid y : \sigma(x)+1]\rangle$$

Programa B:

$$\dfrac{x \in Dom(\sigma)}{\langle x++, \sigma\rangle \ \Downarrow_{exp} \ \langle x, [\sigma \mid x : \sigma(x)+1]\rangle}\,\text{VarInc}$$

Luego aplicamos sobre este resultado:

$$\dfrac{}{\langle x, [\sigma \mid x : \sigma(x)+1]\rangle \ \Downarrow_{exp} \ \langle \sigma(x)+1, [\sigma \mid x : \sigma(x)+1]\rangle}\,\text{Var}$$

Por último aplicamos la regla para la asignación:

$$\dfrac{\langle x++, \sigma\rangle \ \Downarrow_{exp} \ \langle \sigma(x)+1, [\sigma \mid x : \sigma(x)+1]\rangle}{\langle y = x++, \sigma\rangle \ \to \ \langle skip, [\ [\sigma \mid x : \sigma(x)+1\ ] \mid y : \sigma(x)+1]\rangle}\,\text{Ass}$$

Por lo tanto podemos decir que:

$$\langle y = x++, \sigma\rangle \ \to^* \ \langle skip, [\ [\sigma \mid x : \sigma(x)+1\ ] \mid y : \sigma(x)+1]\rangle$$

Habiendo analizado ambos programas podemos afirmar que los mismos derivan a:

$$\langle skip,\ [\ [\sigma \mid x : \sigma(x) + 1]\ y : \sigma(x) + 1]\rangle \ (1)$$

y σ es un estado aleatorio resulta que para todo estado $\sigma \in \Sigma$ vale (1), en consecuencia ambos programas son semánticamente equivalentes.