TP1 - Análisis de Lenguajes de Programación

Nicolas Becerra, Federico Buczek, Ignacio Rímini

1 de Octubre, 2025

Ejercicio 1. Extender sintaxis concreta y abstracta de LIS para incluir operador ++ y al comando case.

- **Sintaxis Abstracta**

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

- **Sintaxis Concreta**

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

$$
\mathit{casebody} ::= \epsilon \mid \mathit{boolexp}\ \text{':'}\ \text{'\{'}\ \mathit{comm}\ \text{'\}'}\ \mathit{casebody}
$$

Ejercicio 4. Modificar semántica big-step de expresiones enteras para incluir al operador ++.

A continuación, se detalla una regla de semántica big-step para el operador ++ (INC):

$$
\frac{x \in \mathrm{dom}(\sigma)}{\langle x\texttt{++}, \sigma\rangle \Downarrow_{\exp} \langle \sigma x + 1, [\sigma \mid x : \sigma x + 1]\rangle}\ \textsc{Inc}
$$

Ejercicio 5. Demostración de determinismo de evaluación en un paso $\rightsquigarrow$.

Teorema: La relación de evaluación en un paso $\rightsquigarrow$ es determinista. Es decir, para cualquier cualquier comando c y estado σ, 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 $c_1 = c_2$ y $\sigma_1 = \sigma_2$.

Demostración. Por inducción sobre la derivación $\langle c, \sigma\rangle \rightsquigarrow \langle c_1, \sigma_1\rangle$.

- Si la última regla aplicada es Ass, entonces c tiene la forma $v = e$ donde $\langle e, \sigma\rangle \Downarrow_{\text{exp}} \langle n, \sigma'\rangle$ para algún n y σ'. Además, $c_1 = \textbf{skip}$ y $\sigma_1 = [\sigma' \mid v : n]$.

  Claramente, la única regla que se puede aplicar sobre $\langle c, \sigma\rangle$ es Ass ya que el lado izquierdo del resto de las reglas tiene una estructura diferente. Ahora, la última regla en la derivación de $\langle c, \sigma\rangle \rightsquigarrow \langle c_2, \sigma_2\rangle$ debe ser Ass. Entonces $c_2 = \textbf{skip}$.

  Cómo $\Downarrow_{\text{exp}}$ es determinista, $\sigma_1 = \sigma_2$. Finalmente, $c_1 = c_2$ y $\sigma_1 = \sigma_2$ como queriamos.

- Si la última regla aplicada es Seq_1, entonces $c = \textbf{skip} \; ; \; c_d$ para algún c_d. Observemos que sólo Seq_1 se puede aplicar sobre $\langle c, \sigma\rangle$ (no se puede aplicar Seq_2 ya que requiere que $\langle \textbf{skip}, \sigma\rangle \rightsquigarrow \langle c_0', \sigma_0'\rangle$ pero esto no es posible ya que el lado izquierdo de ninguna regla tiene la forma $\langle \textbf{skip}, \sigma\rangle$).

  Finalmente, $c_1 = c_d = c_2$ y $\sigma_1 = \sigma = \sigma_2$.

- Si la última regla aplicada es Seq_2, entonces $c = c_a; c_b$ y $c_1 = c_a'; c_b$. Otra vez, la única regla aplicable es Seq_2 (no se puede aplicar Seq_1 ya que $\langle c_a, \sigma\rangle \rightsquigarrow \langle c_a', \sigma_1\rangle$ y, por lo tanto, c_a no puede ser **skip**).

  Entonces $c_2 = c_a''; c_b$ y $\langle c_a, \sigma\rangle \rightsquigarrow \langle c_a'', \sigma_2\rangle$. Ahora, esta subderivación debe ser determinista por la Hipótesis Inductiva y tenemos qué $c_a' = c_a''$ y $\sigma_1 = \sigma_2$. Finalmente, $c_1 = c_a'; c_b = c_a''; c_b = c_2$ y $\sigma_1 = \sigma_2$.

- Si la última regla aplicada es If_1, entonces $c = \textbf{if } b \textbf{ then } c_a \textbf{ else } c_b$ donde $\langle b, \sigma\rangle \Downarrow_{\text{exp}} \langle \textbf{true}, \sigma_1\rangle$. La única regla aplicable sobre $\langle c, \sigma\rangle$ es If_1 (no puede ser If_2 ya que no podemos tener $\langle b, \sigma\rangle \Downarrow_{\text{exp}} \langle \textbf{false}, \sigma'\rangle$ porque $\Downarrow_{\text{exp}}$ es determinista).

  Como $\Downarrow_{\text{exp}}$ es determinista tenemos qué $\sigma_1 = \sigma_2$. Finalmente, $c_1 = c_a = c_a = c_2$ ya que If_1 es la única regla aplicable.

- Si la última regla aplicada es If_2, entonces $c = \textbf{if } b \textbf{ then } c_a \textbf{ else } c_b$ donde $\langle b, \sigma\rangle \Downarrow_{\text{exp}} \langle \textbf{false}, \sigma_1\rangle$. La única regla aplicable sobre $\langle c, \sigma\rangle$ es If_2 (no puede ser If_1 ya que no podemos tener $\langle b, \sigma\rangle \Downarrow_{\text{exp}} \langle \textbf{true}, \sigma'\rangle$ porque $\Downarrow_{\text{exp}}$ es determinista).

  Como $\Downarrow_{\text{exp}}$ es determinista tenemos qué $\sigma_1 = \sigma_2$. Finalmente, $c_1 = c_a = c_a = c_2$ ya que If_2 es la única regla aplicable.

- Si la última regla aplicada es Repeat, tenemos que $c = \textbf{repeat } c' \textbf{ until } b$. Claramente Repeat es la única regla aplicable sobre $\langle c, \sigma\rangle$ ya que el lado izquierdo del resto tiene una forma diferente. Finalmente, $c_1 = \textbf{if } b \textbf{ then skip else repeat } c' \textbf{ until } b = c_2$ y $\sigma_1 = \sigma = \sigma_2$ como queriamos.

Ejercicio 6. Probar equivalencia semántica entre programas.

Usaremos la notación $[\sigma \mid x : n_1, y : n_2]$ para representar $[[\sigma \mid x : n_1] \mid y : n_2]$.

Sea $\sigma \in \Sigma$ un estado arbitrario. Si $x \notin \mathrm{dom}(\sigma)$ entonces ambos programas quedan atascados (solo podemos aplicar la regla Ass pero no cumplimos las premisas) y el resultado es trivialmente verdadero. Suponemos que $x \in \mathrm{dom}(\sigma)$ de forma tal que $\sigma(x) = n_0$.

- Probemos que $\langle (x = x + 1; y = x), \sigma \rangle \rightsquigarrow^* \langle \mathbf{skip}, [\sigma \mid x : n_0 + 1, y : n_0 + 1] \rangle$:

$$\dfrac{\dfrac{\dfrac{\dfrac{x \in \mathrm{dom}(\sigma)}{\langle x, \sigma \rangle \Downarrow_{\mathrm{exp}} \langle n_0, \sigma \rangle}\,\textsc{Var} \qquad \dfrac{}{\langle 1, \sigma \rangle \Downarrow_{\mathrm{exp}} \langle 1, \sigma \rangle}\,\textsc{NVal}}{\langle x + 1, \sigma \rangle \Downarrow_{\mathrm{exp}} \langle n_0 + 1, \sigma \rangle}\,\textsc{Plus}}{\langle x = x + 1, \sigma \rangle \rightsquigarrow \langle \mathbf{skip}, [\sigma \mid x : n_0 + 1] \rangle}\,\textsc{Ass}}{\langle x = x + 1; y = x, \sigma \rangle \rightsquigarrow \langle \mathbf{skip}; y = x, [\sigma \mid x : n_0 + 1] \rangle}\,\textsc{Sec}_2$$

$$\dfrac{}{\langle \mathbf{skip}; y = x, [\sigma \mid x : n_0 + 1] \rangle \rightsquigarrow \langle y = x, [\sigma \mid x : n_0 + 1] \rangle}\,\textsc{Sec}_1$$

$$\dfrac{\dfrac{x \in \mathrm{dom}([\sigma \mid x : n_0 + 1])}{\langle x, [\sigma \mid x : n_0 + 1] \rangle \Downarrow_{\mathrm{exp}} \langle n_0 + 1, [\sigma \mid x : n_0 + 1] \rangle}\,\textsc{Var}}{\langle y = x, [\sigma \mid x : n_0 + 1] \rangle \rightsquigarrow \langle \mathbf{skip}, [[\sigma \mid x : n_0 + 1, y : n_0 + 1] \rangle}\,\textsc{Ass}$$

- Probemos que $\langle (y = x{+}{+}), \sigma \rangle \rightsquigarrow^* \langle \mathbf{skip}, [[\sigma \mid x : n_0 + 1, y : n_0 + 1] \rangle$:

$$\dfrac{\dfrac{x \in \mathrm{dom}(\sigma)}{\langle x{+}{+}, \sigma \rangle \Downarrow_{\mathrm{exp}} \langle n_0 + 1, [\sigma \mid x : n_0 + 1] \rangle}\,\textsc{Inc}}{\langle y = x{+}{+}, \sigma \rangle \rightsquigarrow \langle \mathbf{skip}, [\sigma \mid x : n_0 + 1, y : n_0 + 1] \rangle}\,\textsc{Ass}$$