Lenguajes y Compiladores

Miguel Pagano

5 de mayo de 2023

Repaso

Comandos

$$
\begin{array}{rl}
\langle comm \rangle ::= & \textbf{skip} \\
\mid & \langle var \rangle := \langle intexp \rangle \\
\mid & \langle comm \rangle \ ; \langle comm \rangle \\
\mid & \textbf{if } \langle boolexp \rangle \textbf{ then } \langle comm \rangle \textbf{ else } \langle comm \rangle \\
\mid & \textbf{newvar } \langle var \rangle := \langle intexp \rangle \textbf{ in } \langle comm \rangle \\
\mid & \textbf{while } \langle boolexp \rangle \textbf{ do } \langle comm \rangle \\
\mid & \textbf{fail} \\
\mid & \textbf{catchin } \langle comm \rangle \textbf{ with } \langle comm \rangle
\end{array}
$$

Expresiones

$\langle natconst \rangle ::= \ \mathbf{0} \mid \mathbf{1} \mid \mathbf{2} \mid \ldots$

$\langle intexp \rangle ::= \ \langle natconst \rangle$

$\quad \mid \quad \langle var \rangle$

$\quad \mid \quad \text{-}\, \langle intexp \rangle$

$\quad \mid \quad \langle intexp \rangle \oplus \langle intexp \rangle$

$\oplus \in \{+, -, *, /, \%, \mathbf{rem}\}$

$\langle boolconst \rangle ::= \ \mathbf{true} \mid \mathbf{false}$

$\langle boolexp \rangle ::= \ \langle boolconst \rangle$

$\quad \mid \quad \neg\, \langle boolexp \rangle$

$\quad \mid \quad \langle boolexp \rangle \otimes \langle boolexp \rangle$

$\quad \mid \quad \langle boolexp \rangle \oslash \langle boolexp \rangle$

$\otimes \in \{<, \leqslant, =, \neq, \geqslant, >\}$

$\oslash \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$

Significado de los comandos de LIS

La semántica denotacional daba el significado de los comandos como una función de estados en "comportamientos":

Funciones semánticas

$$
\begin{aligned}
[\![\_]\!]^{intexp} &\in \langle intexp \rangle \to (\Sigma \to \mathbb{Z}) \\
[\![\_]\!]^{boolexp} &\in \langle boolexp \rangle \to (\Sigma \to \{V, F\}) \\
[\![\_]\!]^{comm} &\in \langle comm \rangle \to (\Sigma \to \Sigma')
\end{aligned}
$$

Recordemos que $\Sigma' = (\Sigma \cup (\{\mathbf{abort}\} \times \Sigma))_\perp$.

Semántica Operacional. Ideas.

- En la semántica denotacional no aparecen los estados intermedios.
- La semantica ***operacional*** explicita esos estados intermedios.
- La ejecución de un programa es una secuencia de ***configuraciones***:
$$\gamma_0 \to \gamma_1 \to \gamma_2 \to \dots$$
- Una configuración, γ, puede ser ***terminal***: un estado o un estado fallido;
- o puede ser ***no-terminal***: un comando (lo que falta de ejecutar) y un estado.

Semántica Operacional. Ideas.

- Una ejecución será una secuencia ***maximal*** de configuraciones donde cada par sucesivo está relacionado por la relación $\rightarrow$.
- La secuencia puede ser finita, entonces termina en un estado (correcto o fallido),
- o puede ser una secuencia infinita: el programa no termina.

Relación de un paso (o small-step)

$$\begin{aligned}
\Gamma_T &= \Sigma \cup (\{\textbf{abort}\} \times \Sigma) && \text{configuración terminal}\\
\Gamma_{NT} &= \langle comm \rangle \times \Sigma && \text{configuración no-terminal}\\
\Gamma &= \Gamma_T \cup \Gamma_{NT} && \text{configuración}\\
\rightarrow &\subseteq \Gamma_{NT} \times \Gamma && \text{relación}
\end{aligned}$$

A continuación definimos la relación $\rightarrow$ inductivamente.

$$(\text{skip})\ \frac{}{\langle \mathbf{skip}, \sigma\rangle \rightarrow \sigma}$$

$$(\text{assign})\ \frac{}{\langle v := e, \sigma\rangle \rightarrow [\sigma \,|\, v : [\![e]\!]\sigma]}$$

$$\text{fail}\ \frac{}{\langle \mathbf{fail}, \sigma\rangle \rightarrow \langle \mathbf{abort}, \sigma\rangle}$$

La relación $\rightarrow$. Composición secuencial.

$$\text{(seq-ok)}\ \frac{\langle c_0, \sigma\rangle \rightarrow \sigma'}{\langle c_0\,;\, c_1, \sigma\rangle \rightarrow \langle c_1, \sigma'\rangle}$$

$$\text{(seq-ab)}\ \frac{\langle c_0, \sigma\rangle \rightarrow \langle \mathbf{abort}, \sigma'\rangle}{\langle c_0\,;\, c_1, \sigma\rangle \rightarrow \langle \mathbf{abort}, \sigma'\rangle}$$

$$\text{(seq-nt)}\ \frac{\langle c_0, \sigma\rangle \rightarrow \langle c_0', \sigma'\rangle}{\langle c_0\,;\, c_1, \sigma\rangle \rightarrow \langle c_0'\,;\, c_1, \sigma'\rangle}$$

La relación $\to$. Condicional.

$$(\text{if-true})\ \frac{}{\langle \textbf{if}\ b\ \textbf{then}\ c_0\ \textbf{else}\ c_1, \sigma\rangle \to \langle c_0, \sigma\rangle}\ \text{si}\ [\![b]\!]\sigma = T$$

$$(\text{if-false})\ \frac{}{\langle \textbf{if}\ b\ \textbf{then}\ c_0\ \textbf{else}\ c_1, \sigma\rangle \to \langle c_1, \sigma\rangle}\ \text{si}\ [\![b]\!]\sigma = F$$

La relación $\to$. Ciclo.

$$\text{(while-false)}\ \frac{}{\langle \textbf{while}\ b\ \textbf{do}\ c, \sigma\rangle \to \sigma}\ \text{si}\ [\![b]\!]\sigma = F$$

$$\text{(while-true)}\ \frac{}{\langle \textbf{while}\ b\ \textbf{do}\ c, \sigma\rangle \to \langle c\,;\ \textbf{while}\ b\ \textbf{do}\ c, \sigma\rangle}\ \text{si}\ [\![b]\!]\sigma = T$$

La relación $\rightarrow$. Variables locales.

$$\text{(loc-ok)}\ \frac{\langle c, [\sigma \,|\, v : [\![e]\!]\sigma] \rangle \rightarrow \sigma'}{\langle \mathbf{newvar}\, v := e \,\mathbf{in}\, c, \sigma \rangle \rightarrow [\sigma' \,|\, v : \sigma\, v]}$$

$$\text{(loc-abort)}\ \frac{\langle c, [\sigma \,|\, v : [\![e]\!]\sigma] \rangle \rightarrow \langle \mathbf{abort}, \sigma' \rangle}{\langle \mathbf{newvar}\, v := e \,\mathbf{in}\, c, \sigma \rangle \rightarrow \langle \mathbf{abort}, [\sigma' \,|\, v : \sigma\, v] \rangle}$$

$$\text{(loc-nt)}\ \frac{\langle c, [\sigma \,|\, v : [\![e]\!]\sigma] \rangle \rightarrow \langle c', \sigma' \rangle}{\langle \mathbf{newvar}\, v := e \,\mathbf{in}\, c, \sigma \rangle \rightarrow \langle \mathbf{newvar}\, v := \sigma'\, v \,\mathbf{in}\, c', [\sigma' \,|\, v : \sigma\, v] \rangle}$$

La relación $\rightarrow$. Captura de Fallas.

$$\text{(catch-ok)}\ \frac{\langle c,\sigma\rangle \rightarrow \sigma'}{\langle \textbf{catchin}\, c\, \textbf{with}\, \hat{c},\sigma\rangle \rightarrow \sigma'}$$

$$\text{(catch-abort)}\ \frac{\langle c,\sigma\rangle \rightarrow \langle \textbf{abort},\sigma'\rangle}{\langle \textbf{catchin}\, c\, \textbf{with}\, \hat{c},\sigma\rangle \rightarrow \langle \hat{c},\sigma'\rangle}$$

$$\text{(catch-nt)}\ \frac{\langle c,\sigma\rangle \rightarrow \langle c',\sigma'\rangle}{\langle \textbf{catchin}\, c\, \textbf{with}\, \hat{c},\sigma\rangle \rightarrow \langle \textbf{catchin}\, c'\, \textbf{with}\, \hat{c},\sigma'\rangle}$$

Progreso
Para todo $\langle c, \sigma \rangle$ existe γ tal que $\langle c, \sigma \rangle \to \gamma$.

Determinismo
Si $\langle c, \sigma \rangle \to \gamma$ y $\langle c, \sigma \rangle \to \gamma'$, entonces $\gamma = \gamma'$.

Función
La relación $\to$ define una función de Γ_{NT} a Γ.

Ejecuciones. Formalmente.

Definimos $\to^*$ como la clausura reflexiva y transitiva de $\to$:

$$\text{(refl)}\ \frac{}{\gamma \to^* \gamma}$$

$$\text{(paso)}\ \frac{\gamma \to \gamma'}{\gamma \to^* \gamma'}$$

$$\text{(trans)}\ \frac{\gamma_1 \to^* \gamma_2 \qquad \gamma_2 \to^* \gamma_3}{\gamma_1 \to^* \gamma_3}$$

Ejecución. Formalmente.

Una ejecución es una secuencia maximal de transiciones.

Es decir, $\rho : \gamma_0 \to^* \gamma_n$ es maximal si γ_n es una configuración terminal o si ρ es infinita.

Si ρ es una secuencia infinita decimos que diverge y escribimos $\gamma \uparrow$.

Definimos una nueva semántica usando la ejecución:

$$\{\!\{\_\}\!\} : \langle comm \rangle \to \Sigma \to \Sigma'$$

$$\{\!\{c\}\!\}\sigma = \begin{cases} \perp & \text{si } \langle c, \sigma \rangle \uparrow \\ \gamma & \text{si } \langle c, \sigma \rangle \to^* \gamma \in \Gamma_T \end{cases}$$

Teorema de Corrección

Teorema

Para todo comando c, $\{\!\{c\}\!\} = [\![c]\!]$.

Prueba: práctico.