Lenguajes y Compiladores

Matías Steinberg y Miguel Pagano

5 de abril de 2023

Repaso

Un Lenguaje Imperativo Simple

Comandos

$$
\begin{array}{rcl}
\langle comm \rangle & ::= & \textbf{skip} \\
& | & \langle var \rangle := \langle intexp \rangle \\
& | & \langle comm \rangle \ ; \langle comm \rangle \\
& | & \textbf{if } \langle boolexp \rangle \textbf{ then } \langle comm \rangle \textbf{ else } \langle comm \rangle \\
& | & \textbf{newvar } \langle var \rangle := \langle intexp \rangle \textbf{ in } \langle comm \rangle \\
& | & \textbf{while } \langle boolexp \rangle \textbf{ do } \langle comm \rangle
\end{array}
$$

Un Lenguaje Imperativo Simple

Expresiones

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

$\langle intexp \rangle ::= \langle natconst \rangle$
$\quad \mid \quad \langle var \rangle$
$\quad \mid \quad - \langle intexp \rangle$
$\quad \mid \quad \langle intexp \rangle \oplus \langle intexp \rangle$

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

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

$\langle boolexp \rangle ::= \langle boolconst \rangle$
$\quad \mid \quad \neg \langle boolexp \rangle$
$\quad \mid \quad \langle boolexp \rangle \oslash \langle boolexp \rangle$
$\quad \mid \quad \langle boolexp \rangle \owedge \langle boolexp \rangle$

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

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

Significado de los comandos de LIS

Funciones semánticas

$$[\![\_]\!]^{intexp} \in \langle intexp \rangle \to (\Sigma \to \mathbb{Z})$$

Significado de los comandos de LIS

Funciones semánticas

$$[\![\_]\!]^{intexp} \in \langle intexp \rangle \to (\Sigma \to \mathbb{Z})$$
$$[\![\_]\!]^{boolexp} \in \langle boolexp \rangle \to (\Sigma \to \{V, F\})$$

Significado de los comandos de LIS

Funciones semánticas

$$[\![\_]\!]^{intexp} \in \langle intexp \rangle \to (\Sigma \to \mathbb{Z})$$
$$[\![\_]\!]^{boolexp} \in \langle boolexp \rangle \to (\Sigma \to \{V, F\})$$
$$[\![\_]\!]^{comm} \in \langle comm \rangle \to (\Sigma \to \Sigma_{\perp})$$

Significado de los comandos de LIS

Funciones semánticas

$$[\![\_]\!]^{intexp} \in \langle intexp \rangle \to (\Sigma \to \mathbb{Z})$$
$$[\![\_]\!]^{boolexp} \in \langle boolexp \rangle \to (\Sigma \to \{V, F\})$$
$$[\![\_]\!]^{comm} \in \langle comm \rangle \to (\Sigma \to \Sigma_{\perp})$$

Significado de los comandos de LIS

Funciones semánticas

$$[\![\_]\!]^{intexp} \in \langle intexp \rangle \to (\Sigma \to \mathbb{Z})$$
$$[\![\_]\!]^{boolexp} \in \langle boolexp \rangle \to (\Sigma \to \{V, F\})$$
$$[\![\_]\!]^{comm} \in \langle comm \rangle \to (\Sigma \to \Sigma_\perp)$$

Observación: en el codominio de $[\![\_]\!]^{comm}$ aparece $\Sigma_\perp$ para dar cuenta de la no terminación.

Ecuaciones Semánticas de Comandos

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

Ecuaciones Semánticas de Comandos

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

$$[\![v := e]\!]\,\sigma = [\sigma \mid v : [\![e]\!]\,\sigma]$$

Ecuaciones Semánticas de Comandos

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

$$[\![v := e]\!]\,\sigma = [\sigma \mid v : [\![e]\!]\,\sigma]$$

$$[\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!]\,\sigma = \begin{cases} [\![c]\!]\,\sigma & \text{si } [\![b]\!]\,\sigma \\ [\![c']\!]\,\sigma & \text{si } \neg[\![b]\!]\,\sigma \end{cases}$$

Ecuaciones Semánticas de Comandos

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

$$[\![v := e]\!]\,\sigma = [\sigma \mid v : [\![e]\!]\,\sigma]$$

$$[\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!]\,\sigma = \begin{cases} [\![c]\!]\,\sigma & \text{si } [\![b]\!]\,\sigma \\ [\![c']\!]\,\sigma & \text{si } \neg[\![b]\!]\,\sigma \end{cases}$$

$$[\![c_0; c_1]\!]\,\sigma = [\![c_1]\!]\,\perp\!\!\!\perp([\![c_0]\!]\,\sigma)$$

Ecuaciones Semánticas de Comandos

Bloque local

Después de ejecutar el cuerpo hay que restaurar el valor de la variable.

Ecuaciones Semánticas de Comandos

Bloque local

Después de ejecutar el cuerpo hay que restaurar el valor de la variable.

$$\begin{aligned} rest_{v,\sigma} &: \Sigma \to \Sigma \\ rest_{v,\sigma}(\sigma') &= [\sigma' \,|\, v : \sigma v] \end{aligned}$$

Acá σ y v son parámetros fijos; el argumento es σ'.

Ecuaciones Semánticas de Comandos

Bloque local

Después de ejecutar el cuerpo hay que restaurar el valor de la variable.

$$\begin{aligned} rest_{v,\sigma} &: \Sigma \to \Sigma \\ rest_{v,\sigma}(\sigma') &= [\sigma' \mid v : \sigma v] \end{aligned}$$

Acá σ y v son parámetros fijos; el argumento es σ'.

$$[\![\textbf{newvar}\ v := e\ \textbf{in}\ c]\!]\ \sigma = (rest_{v,\sigma})_{\perp\!\!\perp}([\![c]\!]\ [\sigma | v : [\![e]\!]\ \sigma])$$

Ecuaciones Semánticas de Comandos

La ecuación

Sea $b \in \langle boolexp \rangle$ y $c \in \langle comm \rangle$.

$$F_{b,c} \quad : \quad (\Sigma \to \Sigma_\perp) \to (\Sigma \to \Sigma_\perp)$$

$$F_{b,c}\, f\, \sigma \quad = \quad \begin{cases} \sigma & \text{si } \neg [\![b]\!]\sigma \\ f_{\perp\!\!\perp}([\![c]\!]\,\sigma) & \text{si } [\![b]\!]\sigma \end{cases}$$

Ecuaciones Semánticas de Comandos

La ecuación

Sea $b \in \langle boolexp \rangle$ y $c \in \langle comm \rangle$.

$$F_{b,c} \quad : \quad (\Sigma \to \Sigma_\perp) \to (\Sigma \to \Sigma_\perp)$$

$$F_{b,c}\, f\, \sigma \quad = \quad \begin{cases} \sigma & \text{si } \neg [\![b]\!]\sigma \\ f_{\perp\!\perp}([\![c]\!]\, \sigma) & \text{si } [\![b]\!]\sigma \end{cases}$$

$$[\![\textbf{while } b \textbf{ do } c]\!]\, \sigma = \left(\bigsqcup_{i \in \mathbb{N}} F^i\, \perp \right) \sigma$$

Fallas

Falla

Extender el lenguaje con fallas (de hecho una única falla).

```
if n == 0:
   raise Exception("acá no dividimos por 0")
else:
   m = e / n
```

- ¿Cómo extendemos el lenguaje?

Falla

Extender el lenguaje con fallas (de hecho una única falla).

```
if n == 0:
    raise Exception("acá no dividimos por 0")
else:
    m = e / n
```

- ¿Cómo extendemos el lenguaje?
- Lo comparemos con **for** $v := e_0$ **to** e_1 **do** c.

Falla

Extender el lenguaje con fallas (de hecho una única falla).

```
if n == 0:
    raise Exception("acá no dividimos por 0")
else:
    m = e / n
```

- ¿Cómo extendemos el lenguaje?
- Lo comparemos con **for** $v := e_0$ **to** e_1 **do** c.
- ¿Se necesitan los mismos cambios en la semántica?

Falla

Extender el lenguaje con fallas (de hecho una única falla).

```
if n == 0:
    raise Exception("acá no dividimos por 0")
else:
    m = e / n
```

- ¿Cómo extendemos el lenguaje?
- Lo comparemos con **for** $v := e_0$ **to** e_1 **do** c.
- ¿Se necesitan los mismos cambios en la semántica?
- ¿Qué otros cambios necesitaríamos si quisiéramos distintos errores?

Extensión de lenguaje y de semántica

Sintaxis

$\langle comm \rangle ::= \ldots \mid \textbf{fail}$

Extensión de lenguaje y de semántica

Sintaxis

$\langle comm \rangle ::= \ldots \mid \textbf{fail}$

Semántica

Estados correctos (sin ninguna etiqueta) o estados erróneos.

$$\hat{\Sigma} = \Sigma \cup (\{\textbf{abort}\} \times \Sigma) \qquad \Sigma' = (\hat{\Sigma})_\perp$$

Extensión de lenguaje y de semántica

Sintaxis

$\langle comm \rangle ::= \ldots \mid \textbf{fail}$

Semántica

Estados correctos (sin ninguna etiqueta) o estados erróneos.

$$\hat{\Sigma} = \Sigma \cup (\{\textbf{abort}\} \times \Sigma) \qquad \Sigma' = (\hat{\Sigma})_\perp$$

Nueva semántica de programas:

$$[\![\_]\!] : \langle comm \rangle \to (\Sigma \to \Sigma')$$

Extensión de lenguaje y de semántica

Sintaxis

$\langle comm \rangle ::= \ldots \mid \textbf{fail}$

Semántica

Estados correctos (sin ninguna etiqueta) o estados erróneos.

$$\hat{\Sigma} = \Sigma \cup (\{\textbf{abort}\} \times \Sigma) \qquad \Sigma' = (\hat{\Sigma})_\perp$$

Nueva semántica de programas:

$$[\![\_]\!] : \langle comm \rangle \to (\Sigma \to \Sigma')$$
$$[\![\textbf{fail}]\!]\, \sigma = \langle \textbf{abort}, \sigma \rangle$$

Revisando las Ecuaciones Semánticas de Comandos

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

Revisando las Ecuaciones Semánticas de Comandos

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

$$[\![v := e]\!]\,\sigma = [\sigma \mid v : [\![e]\!]\,\sigma]$$

Revisando las Ecuaciones Semánticas de Comandos

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

$$[\![v := e]\!]\,\sigma = [\sigma \mid v : [\![e]\!]\,\sigma]$$

$$[\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!]\,\sigma = \begin{cases} [\![c]\!]\,\sigma & \text{si } [\![b]\!]\,\sigma \\ [\![c']\!]\,\sigma & \text{si } \neg[\![b]\!]\,\sigma \end{cases}$$

Revisando las Ecuaciones Semánticas de Comandos

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

$$[\![v := e]\!]\,\sigma = [\sigma \mid v : [\![e]\!]\,\sigma]$$

$$[\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!]\,\sigma = \begin{cases} [\![c]\!]\,\sigma & \text{si } [\![b]\!]\,\sigma \\ [\![c']\!]\,\sigma & \text{si } \neg[\![b]\!]\,\sigma \end{cases}$$

$$[\![c_0; c_1]\!]\,\sigma = [\![c_1]\!] \amalg ([\![c_0]\!]\,\sigma)$$

Revisando las Ecuaciones Semánticas de Comandos

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

$$[\![v := e]\!]\,\sigma = [\sigma \mid v : [\![e]\!]\,\sigma]$$

$$[\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!]\,\sigma = \begin{cases} [\![c]\!]\,\sigma & \text{si } [\![b]\!]\,\sigma \\ [\![c']\!]\,\sigma & \text{si } \neg[\![b]\!]\,\sigma \end{cases}$$

$$[\![c_0; c_1]\!]\,\sigma = [\![c_1]\!] \mathbin{\perp\!\!\!\perp} ([\![c_0]\!]\,\sigma)$$

$$[\![\textbf{newvar } v := e \textbf{ in } c]\!]\,\sigma = (rest_{v,\sigma}) \mathbin{\perp\!\!\!\perp} ([\![c]\!]\,[\sigma | v : [\![e]\!]\,\sigma])$$

Propagación de errores

$$(\_)_* : (\Sigma \to \Sigma') \to (\Sigma' \to \Sigma')$$

$$f_* (\bot) = \bot$$

$$f_* (\langle \mathbf{abort}, \sigma \rangle) = \langle \mathbf{abort}, \sigma \rangle$$

$$f_* (\sigma) = f(\sigma)$$

Funciones de propagación de errores

Propagación de errores

$$(\_)_* : (\Sigma \to \Sigma') \to (\Sigma' \to \Sigma')$$
$$f_* (\bot) = \bot$$
$$f_* (\langle \mathbf{abort}, \sigma \rangle) = \langle \mathbf{abort}, \sigma \rangle$$
$$f_* (\sigma) = f(\sigma)$$

Como *map* para listas

$$(\_)_\dagger : (\Sigma \to \Sigma) \to (\Sigma' \to \Sigma')$$
$$f_\dagger (\bot) = \bot$$
$$f_\dagger (\langle \mathbf{abort}, \sigma \rangle) = \langle \mathbf{abort}, f(\sigma) \rangle$$
$$f_\dagger (\sigma) = f(\sigma)$$

Revisando las ecuaciones

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

Revisando las ecuaciones

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

$$[\![v := e]\!]\,\sigma = [\sigma \mid v : [\![e]\!]\,\sigma]$$

Revisando las ecuaciones

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

$$[\![v := e]\!]\,\sigma = [\sigma \mid v : [\![e]\!]\,\sigma]$$

$$[\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!]\,\sigma = \begin{cases} [\![c]\!]\,\sigma & \text{si } [\![b]\!]\,\sigma \\ [\![c']\!]\,\sigma & \text{si } \neg[\![b]\!]\,\sigma \end{cases}$$

Revisando las ecuaciones

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

$$[\![v := e]\!]\,\sigma = [\sigma \mid v : [\![e]\!]\,\sigma]$$

$$[\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!]\,\sigma = \begin{cases} [\![c]\!]\,\sigma & \text{si } [\![b]\!]\,\sigma \\ [\![c']\!]\,\sigma & \text{si } \neg[\![b]\!]\,\sigma \end{cases}$$

$$[\![c_0; c_1]\!]\,\sigma = [\![c_1]\!]_* ([\![c_0]\!]\,\sigma)$$

Revisando las ecuaciones

Comandos sencillos

$$[\![\textbf{skip}]\!]\,\sigma = \sigma$$

$$[\![v := e]\!]\,\sigma = [\sigma \mid v : [\![e]\!]\,\sigma]$$

$$[\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!]\,\sigma = \begin{cases} [\![c]\!]\,\sigma & \text{si } [\![b]\!]\,\sigma \\ [\![c']\!]\,\sigma & \text{si } \neg[\![b]\!]\,\sigma \end{cases}$$

$$[\![c_0; c_1]\!]\,\sigma = [\![c_1]\!]_* ([\![c_0]\!]\,\sigma)$$

$$[\![\textbf{newvar } v := e \textbf{ in } c]\!]\,\sigma = (rest_{v,\sigma})_\dagger ([\![c]\!]\,[\sigma | v : [\![e]\!]\,\sigma])$$

Ecuaciones Semánticas de Comandos

La ecuación

Sea $b \in \langle boolexp \rangle$ y $c \in \langle comm \rangle$.

$$F_{b,c} \quad : \quad (\Sigma \to \Sigma') \to (\Sigma \to \Sigma')$$

$$F_{b,c}\,(f)\,(\sigma) \quad = \quad \begin{cases} \sigma & \text{si } \neg [\![b]\!]\sigma \\ f_*([\![c]\!]\,\sigma) & \text{si } [\![b]\!]\sigma \end{cases}$$

Ecuaciones Semánticas de Comandos

La ecuación

Sea $b \in \langle boolexp \rangle$ y $c \in \langle comm \rangle$.

$$F_{b,c} \quad : \quad (\Sigma \to \Sigma') \to (\Sigma \to \Sigma')$$

$$F_{b,c}\,(f)\,(\sigma) \quad = \quad \begin{cases} \sigma & \text{si } \neg [\![b]\!]\sigma \\ f_*([\![c]\!]\,\sigma) & \text{si } [\![b]\!]\sigma \end{cases}$$

$$[\![\textbf{while } b \textbf{ do } c]\!]\,\sigma = \left(\bigsqcup_{i \in \mathbb{N}} F^i\,\bot \right) \sigma$$

Cómo “capturar” fallas

Ejemplo

```
def safediv(n,m):
  if m == 0:
    raise Exception("A")
  else:
    return (n / m)

try:
     safediv(10,0)
except Exception("A"):
     print("err")
```

Capturando fallas

Sintaxis

$\langle comm \rangle ::= \ldots \mid \textbf{fail} \mid \textbf{catchin}\ \langle comm \rangle\ \textbf{with}\ \langle comm \rangle$

Capturando fallas

Sintaxis

$$\langle comm \rangle ::= \ldots \mid \textbf{fail} \mid \textbf{catchin}\ \langle comm \rangle\ \textbf{with}\ \langle comm \rangle$$

Semántica

$$[\![\_]\!] : \langle comm \rangle \to (\Sigma \to \Sigma')$$

Capturando fallas

Sintaxis

$$\langle comm \rangle ::= \ldots \mid \textbf{fail} \mid \textbf{catchin}\ \langle comm \rangle\ \textbf{with}\ \langle comm \rangle$$

Semántica

$$[\![\_]\!] : \langle comm \rangle \to (\Sigma \to \Sigma')$$

$$[\![\textbf{catchin}\ c_0\ \textbf{with}\ c_1]\!]\ \sigma = [\![c_1]\!] \mathbin{+} ([\![c_0]\!]\ \sigma)$$

Capturando fallas

Sintaxis

$\langle comm \rangle ::= \ldots \mid \textbf{fail} \mid \textbf{catchin}\ \langle comm \rangle\ \textbf{with}\ \langle comm \rangle$

Semántica

$$[\![\_]\!] : \langle comm \rangle \to (\Sigma \to \Sigma')$$

$$[\![\textbf{catchin}\ c_0\ \textbf{with}\ c_1]\!]\ \sigma = [\![c_1]\!]\ {}_{+}\ ([\![c_0]\!]\ \sigma)$$

donde $_+$ está definido por

$$(\_)_+ : (\Sigma \to \Sigma') \to (\Sigma' \to \Sigma')$$

$$f_+\ (\bot) = \bot$$

$$f_+\ (\langle \textbf{abort}, \sigma \rangle) = f\ (\sigma)$$

$$f_+\ (\sigma) = \sigma$$