Estructura de la materia a grandes rasgos:

Primera Parte: Lenguaje imperativo

Segunda Parte: Lenguaje aplicativo puro, y lenguaje aplicativo con referencias y asignación

Ejes de contenidos de la primer parte

1. Introducción a la sintaxis y la semántica de lenguajes
2. El problema de dar significado a la recursión e iteración
3. Un Lenguaje Imperativo Simple

Fallas en el Lenguaje Imperativo

Ahora un programa tiene 3 comportamientos posibles:

1. da un estado final
2. aborta y da un estado final
3. no termina

Fallas en el Lenguaje Imperativo

Para la incorporación de la posibilidad de transferencia de control por fallas, se agregan excepciones al lenguaje:

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

Dominio de resultados posibles:

$\Sigma' = \Sigma \cup \{\textbf{abort}\} \times \Sigma$ (con el orden discreto)

Función semántica:

$$[\![\_]\!] \in \langle comm \rangle \to \Sigma \to \Sigma'_\perp$$

Ecuaciones semánticas

$$
\begin{aligned}
[\![\textbf{skip}]\!]_\sigma &= \sigma \\
[\![\textbf{fail}]\!]_\sigma &= \langle \textbf{abort}, \sigma \rangle \\
[\![v := e]\!]_\sigma &= [\sigma | v : [\![e]\!]_\sigma] \\
[\![\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1]\!]_\sigma &= \begin{cases} [\![c_0]\!]_\sigma & \text{si } [\![b]\!]_\sigma \\ [\![c_1]\!]_\sigma & \text{si no} \end{cases}
\end{aligned}
$$

Operadores de transferencia de control: $*$

Dada $f \in \Sigma \to \Sigma'_\perp$, denotamos por f_* la siguiente extensión de f a $\Sigma'_\perp$:

$$f_* \in \Sigma'_\perp \to \Sigma'_\perp$$

$$f_* x = \begin{cases} f\sigma & \text{si } x = \sigma \in \Sigma \\ x & \text{si no} \end{cases}$$

En este caso, la presencia de una situación abortiva determina que no se transfiere el control a f.
Servirá para describir el significado de $c_0; c_1$, ya que si ocurre una situación de excepción al ejecutar c_0, el control no es tranferido a c_1.

Operadores de transferencia de control: $+$

Dado $f \in \Sigma \to \Sigma'_\perp$, denotamos por f_+ la siguiente extensión de f a $\Sigma'_\perp$:

$$f_+ \in \Sigma'_\perp \to \Sigma'_\perp$$

$$f_+ x = \begin{cases} f\sigma & \text{si } x = \langle \textbf{abort}, \sigma \rangle \in \{\textbf{abort}\} \times \Sigma \\ x & \text{si no} \end{cases}$$

En una clara dualidad con la definición de f_*, la definición de f_+ determina lo contrario: se tranfiere el control a f sólo en caso de excepción. Esto corresponderá a **catchin** c **with** c'.

Operadores de transferencia de control: †

Dado $f \in \Sigma \to \Sigma$, denotamos por $f_\dagger$ la siguiente extensión de f a $\Sigma'_\perp$:

$$f_\dagger \in \Sigma'_\perp \to \Sigma'_\perp$$

$$f_\dagger x = \begin{cases} \langle \textbf{abort}, f\sigma \rangle & x = \langle \textbf{abort}, \sigma \rangle \\ fx & x \in \Sigma \\ \perp & x = \perp \end{cases}$$

Note que aquí hay una transferencia de control a f en cualquier situación (abortiva o no). Servirá para restaurar el valor de las variables locales.

Restantes ecuaciones semánticas

$$
\begin{aligned}
[\![c_0; c_1]\!]_\sigma &= [\![c_1]\!]_*([\![c_0]\!]_\sigma) \\
[\![\textbf{catchin } c_0 \textbf{ with } c_1]\!]_\sigma &= [\![c_1]\!]_+([\![c_0]\!]_\sigma) \\
[\![\textbf{newvar } \mathrm{v} := \mathrm{e} \textbf{ in } \mathrm{c}]\!]_\sigma &= (\lambda\sigma' \in \Sigma.\ [\sigma' | v : \sigma\ v])_\dagger([\![c]\!][\sigma | v : [\![e]\!]_\sigma]) \\
[\![\textbf{while } b \textbf{ do } c]\!] &= \bigsqcup_{i=0}^{\infty} F^i\ \bot_{\Sigma \to \Sigma_\bot}
\end{aligned}
$$

donde

$$
F\ w\ \sigma = \begin{cases} w_*([\![c]\!]_\sigma) & \text{si } [\![b]\!]_\sigma \\ \sigma & \text{si no} \end{cases}
$$

Semántica de transiciones

Describe cómo se realiza el cómputo.

$$\langle \mathsf{x} := 1; \mathsf{y} := 2 * \mathsf{y}, \sigma \rangle \rightarrow \langle \mathsf{y} := 2 * \mathsf{y}, [\sigma | \mathsf{x} : 1] \rangle \rightarrow [\sigma | \mathsf{x} : 1 | \mathsf{y} : 2 * \sigma \mathsf{y}]$$

La relación $\rightarrow$ describe un paso de ejecución (*small-step semantics*).

En cada paso se pasa de una *configuración* a otra.

Semántica de transiciones

Conjunto de configuraciones:

$$\Gamma = \Gamma_t \cup \Gamma_n$$

$$\begin{aligned} \Gamma_t &= \Sigma && \text{(configuraciones terminales)} \\ \Gamma_n &= \langle comm \rangle \times \Sigma && \text{(configuraciones no terminales)} \end{aligned}$$

Definición axiomática de $\rightarrow$

$$\frac{}{\langle \textbf{skip}, \sigma \rangle \rightarrow \sigma}$$

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

Definición axiomática de $\rightarrow$

$$\frac{\langle c_0, \sigma\rangle \rightarrow \sigma'}{\langle c_0; c_1, \sigma\rangle \rightarrow \langle c_1, \sigma'\rangle}$$

$$\frac{\langle c_0, \sigma\rangle \rightarrow \langle c_0', \sigma'\rangle}{\langle c_0; c_1, \sigma\rangle \rightarrow \langle c_0'; c_1, \sigma'\rangle}$$

Note como estas reglas permiten construir un paso intermedio de ejecución, como por ejemplo el primer paso de la ejecución de arriba:

$$\frac{\langle \mathsf{x} := 1, \sigma\rangle \rightarrow [\sigma|\mathsf{x} : 1]}{\langle \mathsf{x} := 1; \mathsf{y} := 2 * \mathsf{y}, \sigma\rangle \rightarrow \langle \mathsf{y} := 2 * \mathsf{y}, [\sigma|\mathsf{x} : 1]\rangle}$$

Definición axiomática de $\rightarrow$

$$\frac{(\llbracket e \rrbracket \sigma = V)}{\langle \textbf{if } e \textbf{ then } c \textbf{ else } c', \sigma \rangle \rightarrow \langle c, \sigma \rangle}$$

$$\frac{(\llbracket e \rrbracket \sigma = F)}{\langle \textbf{if } e \textbf{ then } c \textbf{ else } c', \sigma \rangle \rightarrow \langle c', \sigma \rangle}$$

$$\frac{(\llbracket e \rrbracket \sigma = F)}{\langle \textbf{while } e \textbf{ do } c, \sigma \rangle \rightarrow \sigma}$$

$$\frac{(\llbracket e \rrbracket \sigma = T)}{\langle \textbf{while } e \textbf{ do } c, \sigma \rangle \rightarrow \langle c;\ \textbf{while } e \textbf{ do } c, \sigma \rangle}$$

Definición axiomática de $\rightarrow$

$$\frac{\langle c, [\sigma | v : [\![e]\!]_\sigma] \rangle \rightarrow \sigma'}{\langle \textbf{newvar } v := e \textbf{ in } c, \sigma \rangle \rightarrow [\sigma' | v : \sigma v]}$$

$$\frac{\langle c, [\sigma | v : [\![e]\!]_\sigma] \rangle \rightarrow \langle c', \sigma' \rangle}{\langle \textbf{newvar } v := e \textbf{ in } c, \sigma \rangle \rightarrow \langle \textbf{newvar } v := \sigma' v \textbf{ in } c', [\sigma' | v : \sigma v] \rangle}$$

Determinismo y continuación

Determinismo: $\rightarrow$ define una función: ninguna configuración no terminal puede mover (en un sólo paso) hacia más de una configuración

Continuación: ninguna configuración no terminal puede mover hacia menos de una configuración (no se traba).

Ejecución

Por *ejecución* entendemos una secuencia $c_0 \to c_1 \to c_2 \to \ldots$ maximal, esto es, que no puede prolongarse más de lo que está.

Dicha ejecución es infinita o termina en una configuración terminal σ.

Si la ejecución es infinita decimos que c_0 *diverge* y escribimos $c_0 \uparrow$.

$$\{\!\{c\}\!\}\sigma = \begin{cases} \bot & \text{si } \langle c, \sigma \rangle \uparrow \\ \sigma' & \text{si existe } \sigma' \text{ tal que } \langle c, \sigma \rangle \to^* \sigma' \end{cases}$$

Corrección de la semántica operacional

Lema 1

1. Si $\langle c_0, \sigma\rangle \to^* \sigma'$, entonces $\langle c_0; c_1, \sigma\rangle \to^* \langle c_1, \sigma'\rangle$
2. Si $\langle c, [\sigma | v : [\![e]\!]\sigma]\rangle \to^* \sigma'$, entonces

$$\langle \textbf{newvar } v := e \textbf{ in } c, \sigma\rangle \to^* [\sigma' | v : \sigma v].$$

3. Si $\langle c, [\sigma | v : [\![e]\!]\sigma]\rangle \to^* \langle c', \sigma'\rangle$, entonces

$$\langle \textbf{newvar } v := e \textbf{ in } c, \sigma\rangle \to^* \langle \textbf{newvar } v := \sigma' x \textbf{ in } c', [\sigma' | v : \sigma v]\rangle$$

Corrección de la semántica operacional

Lema 2

1. $\langle c, \sigma \rangle \to \sigma' \Longrightarrow [\![c]\!]\sigma = \sigma'$.
2. $\langle c, \sigma \rangle \to \langle c', \sigma' \rangle \Longrightarrow [\![c]\!]\sigma = [\![c']\!]\sigma'$.

Lema 3 $[\![c]\!]\sigma = \sigma' \Longrightarrow \langle c, \sigma \rangle \to^* \sigma'$

Teorema Para todo comando c se tiene $\{\!\{c\}\!\} = [\![c]\!]$.

Semántica operacional de las fallas

Configuraciones terminales:

$$\Gamma_t \quad = \quad \Sigma \cup \{\textbf{abort}\} \times \Sigma$$

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

$$\frac{\langle c_0, \sigma \rangle \rightarrow \langle \textbf{abort}, \sigma' \rangle}{\langle c_0 ; c_1, \sigma \rangle \rightarrow \langle \textbf{abort}, \sigma' \rangle}$$

Semántica operacional de las fallas

$$\frac{\langle c_0, \sigma\rangle \to \sigma'}{\langle \textbf{catchin } c_0 \textbf{ with } c_1, \sigma\rangle \to \sigma'}$$

$$\frac{\langle c_0, \sigma\rangle \to \langle \textbf{abort}, \sigma'\rangle}{\langle \textbf{catchin } c_0 \textbf{ with } c_1, \sigma\rangle \to \langle c_1, \sigma'\rangle}$$

Semántica operacional de las fallas

$$\frac{\langle c_0, \sigma\rangle \to \sigma'}{\langle \textbf{catchin } c_0 \textbf{ with } c_1, \sigma\rangle \to \sigma'}$$

$$\frac{\langle c_0, \sigma\rangle \to \langle \textbf{abort}, \sigma'\rangle}{\langle \textbf{catchin } c_0 \textbf{ with } c_1, \sigma\rangle \to \langle c_1, \sigma'\rangle}$$

$$\frac{\langle c_0, \sigma\rangle \to \langle c_0', \sigma'\rangle}{\langle \textbf{catchin } c_0 \textbf{ with } c_1, \sigma\rangle \to \langle \textbf{catchin } c_0' \textbf{ with } c_1, \sigma'\rangle}$$

Semántica operacional de las fallas

$$\frac{\langle c, [\sigma | v : [\![e]\!]\sigma\rangle \rightarrow \langle \textbf{abort}, \sigma'\rangle}{\langle \textbf{newvar } v := e \textbf{ in } c, \sigma\rangle \rightarrow \langle \textbf{abort}, [\sigma' | v : \sigma v]\rangle}$$

El if y el while habían sido definidos con suficiente generalidad para que no requieran revisión.

Corrección de la semántica operacional de las fallas

La relación $\rightarrow$ sigue siendo una función en el lenguaje con fallas, toda configuración γ tiene una única ejecución que puede ser infinita (γ diverge) o terminar en una configuración terminal que puede ser de la forma σ o $\langle \textbf{abort}, \sigma \rangle$.

Se puede definir:

$$\{\!\{c\}\!\}\sigma = \begin{cases} \bot & \text{si } \langle c, \sigma \rangle \uparrow \\ \sigma' & \text{si existe } \sigma' \text{ tal que } \langle c, \sigma \rangle \rightarrow^* \sigma' \\ \langle \textbf{abort}, \sigma' \rangle & \text{si existe } \sigma' \text{ tal que } \langle c, \sigma \rangle \rightarrow^* \langle \textbf{abort}, \sigma' \rangle \end{cases}$$

y obtendremos de manera similar a LIS que para todo comando c se tiene $\{\!\{c\}\!\} = [\![c]\!]$.