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

Más sobre dominios: Producto de posets

Si $P_0, P_1, ..., P_{n-1}$ son posets, entonces $P_0 \times P_1, ... \times P_{n-1}$ también lo es, donde el orden entre tuplas se define componente a componente.

Más sobre dominios: Producto de posets

Si $P_0, P_1, ..., P_{n-1}$ son posets, entonces $P_0 \times P_1, ... \times P_{n-1}$ también lo es, donde el orden entre tuplas se define componente a componente.

Cadenas

$$\langle p_0^0, p_0^1, ..., p_0^{n-1} \rangle \leq \langle p_1^0, p_1^1, ..., p_1^{n-1} \rangle \leq \cdots \langle p_k^0, p_k^1, ..., p_k^{n-1} \rangle \leq \cdots$$

Componente a componente forman cadenas en los respectivos órdenes:

$$p_0^0 \leq p_1^0 \leq ... \leq p_k^0 \leq ...$$
$$p_0^1 \leq p_1^1 \leq ... \leq p_k^1 \leq ...$$
$$\vdots$$

Producto de dominios

Si $P_0, P_1, ..., P_{n-1}$ son dominios, entonces $P_0 \times P_1, ... \times P_{n-1}$ también lo es, donde el mínimo es la tupla que consiste del mínimo de cada uno de los dominios.

Todas las funciones sencillas usuales (proyecciones, constructor de tuplas, etc) son trivialmente continuas.

Uniones Disjuntas

Dados conjuntos $P_0, P_1, ..., P_{n-1}$ se define la unión disjunta

$$P_0 + P_1 + ... + P_{n-1} = \{\langle i, p\rangle : p \in P_i\}.$$

Se definen las inyecciones $\iota_i \in P_i \to P_0 + P1 + ... + P_{n-1}$ mediante $\iota_i p = \langle i, p\rangle$

Uniones Disjuntas de posets

Dados posets $P_0, P_1, ..., P_{n-1}$, entonces $P_0 + P_1 + ... + P_{n-1}$ es un poset, donde el orden "no mezcla" los órdenes de los diferentes conjuntos dados:

$$\langle i, p\rangle \leq \langle j, q\rangle \iff i = j \ \wedge \ p \leq_i q$$

Cadenas Una cadena de $P_0 + P_1 + ... + P_{n-1}$ es una que proviene enteramente de uno de los P_i:

$$\langle i, p_1 \rangle \leq \langle i, p_2 \rangle \leq ... \leq \langle i, p_n \rangle \leq ...$$

donde los p_j son todos elementos de P_i.

Unión de dominios

Dados dominios $P_0, P_1, ..., P_{n-1}$, entonces $P_0 + P_1 + ... + P_{n-1}$ en general **no** es un dominio (sólo lo es en el caso trivial $n = 1$).

Todas las funciones sencillas usuales (inyecciones, análisis por casos, etc) son trivialmente continuas.

Lenguaje Imperativo con Fallas y Output

Los comportamientos posibles de un programa en un estado dado son ahora los siguientes:

- se genera una cantidad finita de output y luego "se cuelga"
- se genera una cantidad finita de output y luego termina
- se genera una cantidad finita de output y luego falla
- se genera una cantidad infinita de output

Output: sintaxis

Agregamos el comando

$$\langle comm \rangle ::= !\ \langle intexp \rangle$$

Dominio Ω. Primera aproximación.

El dominio Ω, que representa el conjunto de estos comportamientos, será definido a través de una ecuación recursiva de dominios.

La misma tendrá como solución un dominio que "contenga" los siguientes comportamientos:

- se genera $n_1, .., n_k$ y luego se cuelga
- se genera $n_1, .., n_k$ y luego termina con estado σ
- se genera $n_1, .., n_k$ y luego aborta con estado Σ
- se genera $n_1, .., n_k, ...$ (y no termina)

Dominio Ω. Primera aproximación.

¿Qué objetos forman parte del dominio $(\Sigma' + \mathbf{Z} \times \{\perp\})_\perp$

Los de la forma:

- σ
- $\langle \mathbf{abort}, \sigma \rangle$
- $\langle k, \perp \rangle$
- $\perp$

En términos de los ι

$$\Sigma \xrightarrow{\iota_{norm}} \Sigma' \xrightarrow{\iota_0} \Sigma' + \mathbf{Z} \times \{\bot\}$$

$$\Sigma \xrightarrow{\iota_{abnorm}} \mathbf{Z} \times \{\bot\} \xrightarrow{\iota_1} \Sigma' + \mathbf{Z} \times \{\bot\}$$

$$\Sigma' + \mathbf{Z} \times \{\bot\} \xrightarrow{\iota_\bot} (\Sigma' + \mathbf{Z} \times \{\bot\})_\bot$$

Dominio Ω. Primera aproximación.

¿Qué objetos forman parte del dominio $(\Sigma' + \mathbf{Z} \times \{\perp\})_\perp$

Los de la forma:

- $\iota_\perp(\iota_0(\iota_{norm}\sigma))$
- $\iota_\perp(\iota_0(\iota_{abnorm}\sigma))$
- $\iota_\perp(\iota_1(k, \perp))$
- $\perp$

Dominio Ω. Primera aproximación.

¿Qué objetos forman parte del dominio

$$(\Sigma' + \mathbf{Z} \times (\Sigma' + \mathbf{Z} \times \{\perp\})_\perp)_\perp$$

Hay que combinar los patrones:

- $\iota_\perp(\iota_0(\ \dots\ )$
- $\iota_\perp(\iota_0(\ \dots\ ))$
- $\iota_\perp(\iota_1(k,\ \dots\ ))$

con los 4 "tipos" de objetos de arriba.

Dominio Ω. Primera aproximación.

El comportamiento

"genera los output 3 y 4 y luego termina en el estado σ"

es representado por:

$$\iota_{\perp}(\iota_1(3, \iota_{\perp}(\iota_1(4, \iota_{\perp}(\iota_0(\iota_{norm}(\sigma)))))))$$

Dominios recursivos

Domino semántico para LIS con fallas y output

$$\Omega \quad \approx \quad (\Sigma' + \mathbf{Z} \times \Omega)_\perp$$

El símbolo $\approx$ significa isomorfismo, el cual está dado por las funciones continuas (una inversa de la otra):

$$\phi \;\in\; \Omega \to (\Sigma' + \mathbf{Z} \times \Omega)_\perp \qquad\qquad \psi \;\in\; (\Sigma' + \mathbf{Z} \times \Omega)_\perp \to \Omega$$

Funciones auxiliares

$$
\begin{tikzcd}
\Sigma \arrow[r, "\iota_{norm}"] & \Sigma' \arrow[dr, "\iota_0"] & & & \\
\Sigma \arrow[dr, "\iota_{abnorm}"'] & & \Sigma' + \mathbf{Z} \times \Omega \arrow[r, "\iota_\perp"] & (\Sigma' + \mathbf{Z} \times \Omega)_\perp \arrow[r, "\psi"] & \Omega \\
& \mathbf{Z} \times \Omega \arrow[ur, "\iota_1"] & & &
\end{tikzcd}
$$

Notación

Para expresar las ecuaciones semánticas en adelante serán útiles las siguientes composiciones.

$$\begin{aligned}
\iota_{term} &= \psi \cdot \iota_{\perp} \cdot \iota_0 \cdot \iota_{norm} \in \Sigma \to \Omega \\
\iota_{abort} &= \psi \cdot \iota_{\perp} \cdot \iota_0 \cdot \iota_{abnorm} \in \Sigma \to \Omega \\
\iota_{out} &= \psi \cdot \iota_{\perp} \cdot \iota_1 \in \mathbf{Z} \times \Sigma \to \Omega \\
\perp_{\Omega} &= \psi(\perp) \in \Omega
\end{aligned}$$

Dominio Ω

El comportamiento

"genera los output 3 y 4 y luego termina en el estado σ"

es representado por:

$$\iota_{out}(3, \iota_{out}(4, \iota_{term}\sigma))$$

Output: sintaxis

Agregamos el comando

$$\langle comm \rangle ::= \,!\, \langle intexp \rangle$$

Ecuaciones semánticas

$$
\begin{aligned}
[\![\mathbf{skip}]\!]_\sigma &= \iota_{term}\sigma \\
[\![\mathbf{fail}]\!]_\sigma &= \iota_{abort}\sigma \\
[\![v := e]\!]_\sigma &= \iota_{term}[\sigma | v : [\![e]\!]_\sigma] \\
[\![!e]\!]_\sigma &= \iota_{out}([\![e]\!]_\sigma, \iota_{term}\sigma)
\end{aligned}
$$

Ecuaciones semánticas

$$[\![\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1]\!]_\sigma = \begin{cases} [\![c_0]\!]_\sigma & \text{si } [\![b]\!]_\sigma \\ [\![c_1]\!]_\sigma & \text{si no} \end{cases}$$

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

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

$$[\![\textbf{newvar } \text{v:= e } \textbf{in } \text{c}]\!]_\sigma = (\lambda\sigma' \in \Sigma.\ [\sigma'|v : \sigma\ v])_\dagger([\![c]\!][\sigma|v : [\![e]\!]_\sigma])$$

$$[\![\textbf{while } b \textbf{ do } c]\!] = \bigsqcup_{i=0}^{\infty} F^i \perp_{\Sigma\to\Omega}$$

donde

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

Operadores redefinidos

$$f_* x = \begin{cases} \perp_\Omega & x = \perp_\Omega \\ f\sigma & x = \iota_{term}\sigma \\ \iota_{abort}\sigma & x = \iota_{abort}\sigma \\ \iota_{out}(n, f_*\omega) & x = \iota_{out}(n, \omega) \end{cases}$$

$$f_+ x = \begin{cases} \perp_\Omega & x = \perp_\Omega \\ \iota_{term}\sigma & x = \iota_{term}\sigma \\ f\sigma & x = \iota_{abort}\sigma \\ \iota_{out}(n, f_+\omega) & x = \iota_{out}(n, \omega) \end{cases}$$

Operadores redefinidos

$$f_{\dagger} x = \begin{cases} \perp_{\Omega} & x = \perp_{\Omega} \\ \iota_{term}(f\sigma) & x = \iota_{term}\sigma \\ \iota_{abort}(f\sigma) & x = \iota_{abort}\sigma \\ \iota_{out}(n, f_{\dagger}\omega) & x = \iota_{out}(n, \omega) \end{cases}$$

Input

Sintaxis abstracta:

$$\langle comm \rangle \;::=\; ?\, \langle var \rangle$$

Domino semántico:

$$\Omega \quad \approx \quad (\Sigma' + \mathbf{Z} \times \Omega + \mathbf{Z} \to \Omega)_\perp$$

Isomorfismos:

$$\phi \;\in\; \Omega \to (\Sigma' {+} \mathbf{Z} {\times} \Omega {+} \mathbf{Z} \to \Omega)_\perp \qquad \psi \;\in\; (\Sigma' {+} \mathbf{Z} {\times} \Omega {+} \mathbf{Z} \to \Omega)_\perp \to \Omega$$

$$\Sigma \xrightarrow{\iota_{norm}} \Sigma' \xrightarrow{\iota_0}$$

$$\Sigma \xrightarrow[\iota_{abnorm}]{} $$

$$\mathbf{Z} \times \Omega \xrightarrow{\iota_1} \quad \Sigma' + \mathbf{Z} \times \Omega + \mathbf{Z} \to \Omega$$

$$\downarrow$$

$$\mathbf{Z} \to \Omega \xrightarrow{\iota_2}$$

$$\downarrow$$

$$\downarrow \iota_\perp$$

$$(\Sigma' + \mathbf{Z} \times \Omega + \mathbf{Z} \to \Omega)_\perp \xrightarrow{\psi} \Omega$$

Composiciones útiles

$$
\begin{aligned}
\iota_{term} &= \psi \cdot \iota_{\perp} \cdot \iota_0 \cdot \iota_{norm} \in \Sigma \to \Omega \\
\iota_{abort} &= \psi \cdot \iota_{\perp} \cdot \iota_0 \cdot \iota_{abnorm} \in \Sigma \to \Omega \\
\iota_{out} &= \psi \cdot \iota_{\perp} \cdot \iota_1 \in \mathbf{Z} \times \Sigma \to \Omega \\
\iota_{in} &= \psi \cdot \iota_{\perp} \cdot \iota_2 \in (\mathbf{Z} \to \Sigma) \to \Omega \\
\perp_{\Omega} &= \psi(\perp) \in \Omega
\end{aligned}
$$

$$
\begin{array}{lllll}
\Sigma & \xrightarrow{\iota_{norm}} & & & \\
 & & \Sigma' & \xrightarrow{\iota_0} & \\
\Sigma & \xrightarrow[\iota_{abnorm}]{} & & & \\
 & & & & \Sigma' + \mathbf{Z}\times\Omega + \mathbf{Z}\to\Omega \\
 & & \mathbf{Z}\times\Omega & \xrightarrow{\iota_1} & \downarrow \\
 & & \mathbf{Z}\to\Omega & \xrightarrow{\iota_2} & \downarrow \\
 & & & & \downarrow \iota_\perp \\
 & & & & (\Sigma' + \mathbf{Z}\times\Omega + \mathbf{Z}\to\Omega)_\perp \xrightarrow{\psi} \Omega
\end{array}
$$

Ecuación semántica para el input:

$$[\![?v]\!]_\sigma \quad = \quad \iota_{in}(\lambda n \in \mathbf{Z}.\ \iota_{term}[\sigma|v : n])$$

Observación: Las restantes ecuaciones semánticas no se alteran, sólo es necesario actualizar las funciones de transferencia de control.

$$f_* x = \begin{cases} \bot_\Omega & x = \bot_\Omega \\ f\sigma & x = \iota_{term}\sigma \\ \iota_{abort}\sigma & x = \iota_{abort}\sigma \\ \iota_{out}(n, f_*\omega) & x = \iota_{out}(n, \omega) \\ \iota_{in}(f_* \cdot g) & x = \iota_{in} g \end{cases}$$

$$f_+ x = \begin{cases} \bot_\Omega & x = \bot_\Omega \\ \iota_{term}\sigma & x = \iota_{term}\sigma \\ f\sigma & x = \iota_{abort}\sigma \\ \iota_{out}(n, f_+\omega) & x = \iota_{out}(n, \omega) \\ \iota_{in}(f_+ \cdot g) & x = \iota_{in} g \end{cases}$$

$$f_{\dagger}x = \begin{cases} \bot_{\Omega} & x = \bot_{\Omega} \\ \iota_{term}(f\sigma) & x = \iota_{term}\sigma \\ \iota_{abort}(f\sigma) & x = \iota_{abort}\sigma \\ \iota_{out}(n, f_{\dagger}\omega) & x = \iota_{out}(n, \omega) \\ \iota_{in}(f_{\dagger} \cdot g) & x = \iota_{in}g \end{cases}$$