Lenguaje Imperativo con Output e Input

Miguel Pagano

10 de mayo de 2023

Repaso

Comandos

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

Un Lenguaje Imperativo (no tan) Simple

Expresiones

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

$\langle intexp \rangle ::= \langle natconst \rangle$
$\quad \mid \; \langle var \rangle$
$\quad \mid \; \text{-} \, \langle intexp \rangle$
$\quad \mid \; \langle intexp \rangle \oplus \langle intexp \rangle$

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

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

$\langle boolexp \rangle ::= \langle boolconst \rangle$
$\quad \mid \; \neg \, \langle boolexp \rangle$
$\quad \mid \; \langle intexp \rangle \otimes \langle intexp \rangle$
$\quad \mid \; \langle boolexp \rangle \circledast \langle boolexp \rangle$

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

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

Output

Significado de los comandos de LIS

Posibles comportamientos

1. Cantidad finita de output y un estado final (exitoso o no): $\langle 4, 3, 2, 1, [\sigma \mid x : 0] \rangle$ ó $\langle 1, 2, \langle \mathbf{abort}, \sigma \rangle \rangle$.
2. Cantidad finita de output: $\langle 2 \rangle$.
3. Cantidad infinita de output: $\langle -1, -2, -3, -4, \ldots$.

de qué programas?

1. $!4; !3; !2; !1$ (ó $!4; !3; !2; !1; \mathbf{fail}$)
2. $!x; (\mathbf{while}\, x = 2\, \mathbf{do}\, \mathbf{skip}); !x$,
3. $!x; \mathbf{while}\, x \neq 0\, \mathbf{do}\, !x; x := x - 1$.

Dominio de salidas

$$\Omega = \{\langle n_0, \ldots, n_k, \sigma' \rangle | n_i \in \mathbb{Z}, \sigma' \in \hat{\Sigma}\}$$
$$\cup \{\langle n_0, \ldots, n_k \rangle | n_i \in \mathbb{Z}\}$$
$$\cup \{\langle n_0, n_1, n_2, n_3 \ldots, n_k, \ldots \rangle | n_i \in \mathbb{Z}\}$$
$$\text{donde } \hat{\Sigma} = \Sigma \cup (\{\mathbf{abort}\} \times \Sigma)$$

Concatenación de tuplas
Podemos “concatenar” ciertos elementos de Ω y obtener elementos de Ω:

$$\langle n_0, \ldots, n_k \rangle +\!\!+ \langle m_0, \ldots, s \rangle = \langle n_0, \ldots, n_k, m_0, \ldots, s \rangle$$

Ahí s puede ser un entero o un estado (fallido o no).

Dominio de salidas

Estructura de dominio

1. El orden está dado por ser prefijo: $\langle 4, 3, 2, 1\rangle \leqslant \langle 4, 3, 2, 1, \sigma\rangle$.

Dominio de salidas

Estructura de dominio

1. El orden está dado por ser prefijo: $\langle 4, 3, 2, 1 \rangle \leqslant \langle 4, 3, 2, 1, \sigma \rangle$.
2. El supremo de una cadena no interesante es el último elemento de la cadena.

Dominio de salidas

Estructura de dominio

1. El orden está dado por ser prefijo: $\langle 4, 3, 2, 1\rangle \leqslant \langle 4, 3, 2, 1, \sigma\rangle$.
2. El supremo de una cadena no interesante es el último elemento de la cadena.
3. El supremo de una cadena interesante es ***la*** tupla que coincide en todas las posiciones de cada tupla de la cadena.

Dominio de salidas

Estructura de dominio

1. El orden está dado por ser prefijo: $\langle 4, 3, 2, 1 \rangle \leqslant \langle 4, 3, 2, 1, \sigma \rangle$.
2. El supremo de una cadena no interesante es el último elemento de la cadena.
3. El supremo de una cadena interesante es ***la*** tupla que coincide en todas las posiciones de cada tupla de la cadena.

Dominio de salidas

Estructura de dominio

1. El orden está dado por ser prefijo: $\langle 4, 3, 2, 1\rangle \leqslant \langle 4, 3, 2, 1, \sigma\rangle$.
2. El supremo de una cadena no interesante es el último elemento de la cadena.
3. El supremo de una cadena interesante es ***la*** tupla que coincide en todas las posiciones de cada tupla de la cadena.

Observaciones

1. El menor elemento está dado por $\langle\,\rangle$.

Dominio de salidas

Estructura de dominio

1. El orden está dado por ser prefijo: $\langle 4, 3, 2, 1 \rangle \leqslant \langle 4, 3, 2, 1, \sigma \rangle$.
2. El supremo de una cadena no interesante es el último elemento de la cadena.
3. El supremo de una cadena interesante es ***la*** tupla que coincide en todas las posiciones de cada tupla de la cadena.

Observaciones

1. El menor elemento está dado por $\langle \, \rangle$.
2. En una cadena interesante de tuplas hay tuplas de cada longitud,

Dominio de salidas

Estructura de dominio

1. El orden está dado por ser prefijo: $\langle 4, 3, 2, 1\rangle \leqslant \langle 4, 3, 2, 1, \sigma\rangle$.
2. El supremo de una cadena no interesante es el último elemento de la cadena.
3. El supremo de una cadena interesante es ***la*** tupla que coincide en todas las posiciones de cada tupla de la cadena.

Observaciones

1. El menor elemento está dado por $\langle\,\rangle$.
2. En una cadena interesante de tuplas hay tuplas de cada longitud,
3. por lo tanto el supremo es efectivamente único.

Semántica

Nueva semántica de programas:

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

Extensión de lenguaje y de semántica

Semántica

Nueva semántica de programas:

$$\llbracket \_ \rrbracket : \langle comm \rangle \to (\Sigma \to \Omega)$$

$$\llbracket !e \rrbracket \sigma = \langle \llbracket e \rrbracket \sigma, \sigma \rangle$$

Ecuaciones Semánticas de Comandos (sin output)

Comandos sencillos

$$[\![\mathbf{skip}]\!]\sigma = \langle \sigma \rangle$$

Ecuaciones Semánticas de Comandos (sin output)

Comandos sencillos

$$
\begin{aligned}
[\![\textbf{skip}]\!]\sigma &= \langle \sigma \rangle \\
[\![v := e]\!]\sigma &= \langle [\sigma \mid v : [\![e]\!]\sigma] \rangle
\end{aligned}
$$

Ecuaciones Semánticas de Comandos (sin output)

Comandos sencillos

$$[\![\textbf{skip}]\!]\sigma = \langle\sigma\rangle$$

$$[\![v := e]\!]\sigma = \langle[\sigma \mid v : [\![e]\!]\sigma]\rangle$$

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

Ecuaciones Semánticas de Comandos (sin output)

Comandos sencillos

$$[\![\textbf{skip}]\!]\sigma = \langle \sigma \rangle$$

$$[\![v := e]\!]\sigma = \langle [\sigma \mid v : [\![e]\!]\sigma] \rangle$$

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

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

Ecuaciones Semánticas de Comandos (sin output)

Comandos sencillos

$$[\![\mathbf{skip}]\!]\sigma = \langle\sigma\rangle$$

$$[\![v := e]\!]\sigma = \langle[\sigma \mid v : [\![e]\!]\sigma]\rangle$$

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

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

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

Ecuaciones Semánticas de Comandos (sin output)

La ecuación

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

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

Ecuaciones Semánticas de Comandos (sin output)

La ecuación

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

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

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

No hay que cambiar nada, sólo redefinir los operadores.

Funciones de propagación de errores

Propagación de errores

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

Como map para listas

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

Propagación de errores

$$
\begin{aligned}
(\_)_*\colon (\Sigma \to \Omega) &\to (\Omega \to \Omega) \\
f_*\left(\langle n_0, \ldots, n_{k-1} \rangle\right) &= \langle n_0, \ldots, n_{k-1} \rangle \\
f_*\left(\langle n_0, \ldots, n_{k-1}, \sigma \rangle\right) &= \langle n_0, \ldots, n_{k-1} \rangle \mathbin{+\!\!+} f(\sigma) \\
f_*\left(\langle n_0, \ldots, n_{k-1}, \langle \mathbf{abort}, \sigma \rangle \rangle\right) &= \langle n_0, \ldots, n_{k-1}, \langle \mathbf{abort}, \sigma \rangle \rangle \\
f_*\left(\langle n_0, \ldots, n_{k-1}, \ldots \rangle\right) &= \langle n_0, \ldots, n_{k-1}, \ldots \rangle
\end{aligned}
$$

Como map **para listas**

$$
\begin{aligned}
(\_)_{\dagger} &: (\Sigma \to \Sigma) \to (\Omega \to \Omega) \\
f_{\dagger}\left(\langle n_0, \ldots, n_{k-1} \rangle\right) &= \langle n_0, \ldots, n_{k-1} \rangle \\
f_{\dagger}\left(\langle n_0, \ldots, n_{k-1}, \sigma \rangle\right) &= \langle n_0, \ldots, n_{k-1}, f(\sigma) \rangle \\
f_{\dagger}\left(\langle n_0, \ldots, n_{k-1}, \langle \mathbf{abort}, \sigma \rangle \rangle\right) &= \langle n_0, \ldots, n_{k-1}, \langle \mathbf{abort}, f\left(\sigma\right) \rangle \rangle \\
f_{\dagger}\left(\langle n_0, \ldots, n_{k-1}, \ldots \rangle\right) &= \langle n_0, \ldots, n_{k-1}, \ldots \rangle
\end{aligned}
$$

Una presentación más estructural para Ω

Consideremos las siguientes funciones:

$$\begin{aligned} \iota_{\perp} &: \{\langle\,\rangle\} \to \Omega & \iota_{term} &: \Sigma \to \Omega \\ \iota_{\perp}(\langle\,\rangle) &= \langle\,\rangle & \iota_{term}(\sigma) &= \langle \sigma \rangle \\ \iota_{abort} &: \Sigma \to \Omega & \iota_{out} &: \mathbb{Z} \times \Omega \to \Omega \\ \iota_{abort}(\sigma) &= \langle \sigma \rangle & \iota_{out}(\langle n, \omega \rangle) &= \langle n \rangle +\!\!+\, \omega \end{aligned}$$

Propagación de errores

$$
\begin{aligned}
(\_)_* \colon (\Sigma \to \Omega) &\to (\Omega \to \Omega) \\
f_*\,(\perp) &= \perp \\
f_*\,(\iota_{term}(\sigma)) &= f(\sigma) \\
f_*\,(\iota_{abort}(\sigma)) &= \iota_{abort}(\sigma) \\
f_*\,(\iota_{out}(\langle n, \omega\rangle)) &= \iota_{out}(\langle n, f_*(\omega)\rangle)
\end{aligned}
$$

Observación: Es fundamental que f_* sea continua.

Una nueva definición de Omega

$$\Omega \underset{\overleftarrow{\psi}}{\overset{\xrightarrow{\phi}}{\approx}} (\hat{\Sigma} + \mathbb{Z} \times \Omega)_{\perp}$$

Una nueva definición de Omega

$$\Omega \underset{\underset{\psi}{\leftarrow}}{\overset{\overset{\phi}{\rightarrow}}{\approx}} (\hat{\Sigma} + \mathbb{Z} \times \Omega)_\perp$$

$$\begin{tikzcd}
\Sigma \arrow[dr, "\iota_{norm}"'] \arrow[drrrr, bend left, "\iota_{term}"] & & & & \\
& \hat{\Sigma} \arrow[r, "\iota_0"] & \hat{\Sigma} + \mathbb{Z} \times \Omega \arrow[r, "\iota_\uparrow"] & (\hat{\Sigma} + \mathbb{Z} \times \Omega)_\perp \arrow[r, "\psi"] & \Omega \\
\Sigma \arrow[ur, "\iota_{abnorm}"] \arrow[urrrr, bend right, "\iota_{abort}"'] & \mathbb{Z} \times \Omega \arrow[ur, "\iota_1"] \arrow[urrr, "\iota_{out}"] & & &
\end{tikzcd}$$

Observación: El orden de Ω lo podemos pensar como el orden $(\hat{\Sigma} + \mathbb{Z} \times \Omega)_\perp$.

Una nueva definición de Omega

$$\begin{tikzcd}
\Sigma \arrow[dr, "\iota_{norm}"'] \arrow[drrrr, bend left, "\iota_{term}"] & & & & \\
& \hat{\Sigma} \arrow[r, "\iota_0"] & \hat{\Sigma} + \mathbb{Z} \times \Omega \arrow[r, "\iota_{\uparrow}"] & (\hat{\Sigma} + \mathbb{Z} \times \Omega)_{\perp} \arrow[r, "\psi"] & \Omega \\
\Sigma \arrow[ur, "\iota_{abnorm}"] \arrow[urrrr, bend right, "\iota_{abort}"'] & \mathbb{Z} \times \Omega \arrow[ur, "\iota_1"] \arrow[urrr, "\iota_{out}"] & & &
\end{tikzcd}$$

Una nueva definición de Omega

$$
\begin{tikzcd}
\Sigma \arrow[dr, "\iota_{norm}"'] \arrow[drrrr, bend left, "\iota_{term}"] & & & & \\
& \hat{\Sigma} \arrow[r, "\iota_0"] & \hat{\Sigma} + \mathbb{Z} \times \Omega \arrow[r, "\iota_{\uparrow}"] & (\hat{\Sigma} + \mathbb{Z} \times \Omega)_{\perp} \arrow[r, "\psi"] & \Omega \\
\Sigma \arrow[ur, "\iota_{abnorm}"] \arrow[urrrr, bend right, "\iota_{abort}"'] & \mathbb{Z} \times \Omega \arrow[ur, "\iota_1"] \arrow[urrr, "\iota_{out}"] & & &
\end{tikzcd}
$$

$$\iota_{term} = \psi \circ \iota_{\uparrow} \circ \iota_0 \circ \iota_{norm}$$

$$\iota_{abort} = \psi \circ \iota_{\uparrow} \circ \iota_0 \circ \iota_{abnorm}$$

$$\iota_{out} = \psi \circ \iota_{\uparrow} \circ \iota_1$$

Observación: Podemos extender Ω con nuevas construcciones.

Propagación de comportamientos y map

$$
\begin{aligned}
(\_)_* : (\Sigma \to \Omega) &\to (\Omega \to \Omega) \\
f_* \, (\bot) &= \bot \\
f_* \, (\iota_{term}(\sigma)) &= f(\sigma) \\
f_* \, (\iota_{abort}(\sigma)) &= \iota_{abort}(\sigma) \\
f_* \, (\iota_{out}(n, \omega)) &= \iota_{out}(n, f_*(\omega))
\end{aligned}
$$

$$
\begin{aligned}
(\_)_\dagger : (\Sigma \to \Sigma) &\to (\Omega \to \Omega) \\
f_\dagger \, (\bot) &= \bot \\
f_\dagger \, (\iota_{term}(\sigma)) &= \iota_{term}(f(\sigma)) \\
f_\dagger \, (\iota_{abort}(\sigma)) &= \iota_{abort}(f(\sigma)) \\
f_\dagger \, (\iota_{out}(\langle n, \omega \rangle)) &= \iota_{out}(\langle n, f_\dagger(\omega) \rangle)
\end{aligned}
$$

Una nueva definición de Omega

$$\Omega \underset{\underset{\psi}{\leftarrow}}{\overset{\overset{\phi}{\rightarrow}}{\approx}} (\hat{\Sigma} + \mathbb{Z} \times \Omega)_{\perp}$$

Una nueva definición de Omega

$$\Omega \overset{\overset{\phi}{\rightarrow}}{\underset{\underset{\psi}{\leftarrow}}{\approx}} (\hat{\Sigma} + \mathbb{Z} \times \Omega)_{\perp}$$

$$\begin{tikzcd}
\Sigma \arrow[dr, "\iota_{norm}"'] \arrow[rrrrd, bend left, "\iota_{term}"] & & & & \\
 & \hat{\Sigma} \arrow[r, "\iota_0"] & \hat{\Sigma} + \mathbb{Z} \times \Omega \arrow[r, "\iota_{\uparrow}"] & (\hat{\Sigma} + \mathbb{Z} \times \Omega)_{\perp} \arrow[r, "\psi"] & \Omega \\
\Sigma \arrow[ur, "\iota_{abnorm}"] \arrow[rrrru, bend right, "\iota_{abort}"'] & \mathbb{Z} \times \Omega \arrow[ur, "\iota_1"] \arrow[rrru, bend right=15, "\iota_{out}"] & & &
\end{tikzcd}$$

Una nueva definición de Omega

$$
\begin{tikzcd}
\Sigma \arrow[drrrr, bend left, "\iota_{term}"] \arrow[dr, "\iota_{norm}"'] & & & & \\
& \hat{\Sigma} \arrow[r, "\iota_0"] & \hat{\Sigma} + \mathbb{Z} \times \Omega \arrow[r, "\iota_\uparrow"] & (\hat{\Sigma} + \mathbb{Z} \times \Omega)_\perp \arrow[r, "\psi"] & \Omega \\
\Sigma \arrow[ur, "\iota_{abnorm}"] \arrow[urrrr, bend right, "\iota_{abort}"'] & \mathbb{Z} \times \Omega \arrow[ur, "\iota_1"'] \arrow[urrr, "\iota_{out}"] & & &
\end{tikzcd}
$$

Una nueva definición de Omega

$$
\begin{tikzcd}
\Sigma \arrow[dr, "\iota_{norm}"'] \arrow[drrrr, bend left, "\iota_{term}"] & & & & \\
 & \hat{\Sigma} \arrow[r, "\iota_0"] & \hat{\Sigma} + \mathbb{Z} \times \Omega \arrow[r, "\iota_\uparrow"] & (\hat{\Sigma} + \mathbb{Z} \times \Omega)_\perp \arrow[r, "\psi"] & \Omega \\
\Sigma \arrow[ur, "\iota_{abnorm}"] \arrow[urrrr, bend right, "\iota_{abort}"'] & \mathbb{Z} \times \Omega \arrow[ur, "\iota_1"] \arrow[urrr, "\iota_{out}"] & & &
\end{tikzcd}
$$

$$\iota_{term} = \psi \circ \iota_\uparrow \circ \iota_0 \circ \iota_{norm}$$

$$\iota_{abort} = \psi \circ \iota_\uparrow \circ \iota_0 \circ \iota_{abnorm}$$

$$\iota_{out} = \psi \circ \iota_\uparrow \circ \iota_1$$

Observación: Podemos extender Ω con nuevas construcciones.

Comandos

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

Un dominio con comportamientos para Input

$$\Omega \underset{\overleftarrow{\psi}}{\overset{\overrightarrow{\phi}}{\approx}} (\hat{\Sigma} + \mathbb{Z} \times \Omega + \mathbb{Z} \to \Omega)_\perp$$

Un dominio con comportamientos para Input

$$\Omega \underset{\psi}{\overset{\phi}{\underset{\leftarrow}{\overset{\rightarrow}{\approx}}}} (\hat{\Sigma} + \mathbb{Z} \times \Omega + \mathbb{Z} \to \Omega)_{\perp}$$

$$\begin{aligned} \iota_{term} &= \psi \circ \iota_{\uparrow} \circ \iota_0 \circ \iota_{norm} \\ \iota_{abort} &= \psi \circ \iota_{\uparrow} \circ \iota_0 \circ \iota_{abnorm} \\ \iota_{out} &= \psi \circ \iota_{\uparrow} \circ \iota_1 \\ \iota_{in} &= \psi \circ \iota_{\uparrow} \circ \iota_2 & : (\mathbb{Z} \to \Omega) \to \Omega \end{aligned}$$

Observación: Podemos extender Ω con nuevas construcciones.

Propagación de comportamientos y map

$$
\begin{aligned}
(\_)_* &: (\Sigma \to \Omega) \to (\Omega \to \Omega) \\
f_*\,(\bot) &= \bot \\
f_*\,(\iota_{term}(\sigma)) &= f(\sigma) \\
f_*\,(\iota_{abort}(\sigma)) &= \iota_{abort}(\sigma) \\
f_*\,(\iota_{out}(\langle n, \omega \rangle)) &= \iota_{out}(\langle n, f_*(\omega) \rangle) \\
f_*\,(\iota_{in}(g)) &= \iota_{in}(\lambda n \in \mathbb{Z} \,.\, f_*(g(n)))
\end{aligned}
$$

Propagación de “restauración” (map)

$$
\begin{aligned}
(\_)_\dagger : (\Sigma \to \Sigma) &\to (\Omega \to \Omega) \\
f_\dagger\,(\bot) &= \bot \\
f_\dagger\,(\iota_{term}(\sigma)) &= \iota_{term}(f(\sigma)) \\
f_\dagger\,(\iota_{abort}(\sigma)) &= \iota_{abort}(f(\sigma)) \\
f_\dagger\,(\iota_{out}(n, \omega)) &= \iota_{out}(n, f_\dagger(\omega)) \\
f_\dagger\,(\iota_{in}(g)) &= \iota_{in}(\lambda n \in \mathbb{Z}\,.\, f_\dagger(g(n)))
\end{aligned}
$$