

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:

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

Fallas en el Lenguaje Imperativo

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

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

Dominio de resultados posibles:

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

Fallas en el Lenguaje Imperativo

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

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

Dominio de resultados posibles:

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

Función semántica:

$$\llbracket _ \rrbracket \in \langle comm \rangle \rightarrow \Sigma \rightarrow \Sigma'_{\perp}$$

Ecuaciones semánticas

$$\llbracket \text{skip} \rrbracket_{\sigma} = \sigma$$

$$\llbracket \text{fail} \rrbracket_{\sigma} = \langle \text{abort}, \sigma \rangle$$

Ecuaciones semánticas

$$\llbracket \text{skip} \rrbracket_{\sigma} = \sigma$$

$$\llbracket \text{fail} \rrbracket_{\sigma} = \langle \text{abort}, \sigma \rangle$$

$$\llbracket v := e \rrbracket_{\sigma} = [\sigma | v : \llbracket e \rrbracket_{\sigma}]$$

Ecuaciones semánticas

$$\llbracket \text{skip} \rrbracket_{\sigma} = \sigma$$

$$\llbracket \text{fail} \rrbracket_{\sigma} = \langle \text{abort}, \sigma \rangle$$

$$\llbracket v := e \rrbracket_{\sigma} = [\sigma | v : \llbracket e \rrbracket_{\sigma}]$$

$$\llbracket \text{if } b \text{ then } c_0 \text{ else } c_1 \rrbracket_{\sigma} = \begin{cases} \llbracket c_0 \rrbracket_{\sigma} & \text{si } \llbracket b \rrbracket_{\sigma} \\ \llbracket c_1 \rrbracket_{\sigma} & \text{si no} \end{cases}$$

Operadores de transferencia de control: *

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

$$f_* \in \Sigma'_{\perp} \rightarrow \Sigma'_{\perp}$$

Operadores de transferencia de control: *

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

$$f_* \in \Sigma'_{\perp} \rightarrow \Sigma'_{\perp}$$

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

En este caso, la presencia de una situación abortiva determina que no se transfiera el control a f .

Operadores de transferencia de control: *

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

$$f_* \in \Sigma'_{\perp} \rightarrow \Sigma'_{\perp}$$

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

En este caso, la presencia de una situación abortiva determina que no se transfiera 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 transferido a c_1 .

Operadores de transferencia de control: +

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

$$f_+ \in \Sigma'_{\perp} \rightarrow \Sigma'_{\perp}$$

Operadores de transferencia de control: +

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

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

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

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

Operadores de transferencia de control: †

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

$$f_{\dagger} \in \Sigma'_{\perp} \rightarrow \Sigma'_{\perp}$$

Operadores de transferencia de control: †

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

$$f_{\dagger} \in \Sigma'_{\perp} \rightarrow \Sigma'_{\perp}$$

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

Operadores de transferencia de control: †

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

$$f_{\dagger} \in \Sigma'_{\perp} \rightarrow \Sigma'_{\perp}$$

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

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

$$\begin{aligned} \llbracket c_0; c_1 \rrbracket_\sigma &= \llbracket c_1 \rrbracket_* (\llbracket c_0 \rrbracket_\sigma) \\ \llbracket \text{catchin } c_0 \text{ with } c_1 \rrbracket_\sigma &= \llbracket c_1 \rrbracket_+ (\llbracket c_0 \rrbracket_\sigma) \end{aligned}$$

Restantes ecuaciones semánticas

$$\llbracket c_0; c_1 \rrbracket_\sigma = \llbracket c_1 \rrbracket_* (\llbracket c_0 \rrbracket_\sigma)$$

$$\llbracket \text{catchin } c_0 \text{ with } c_1 \rrbracket_\sigma = \llbracket c_1 \rrbracket_+ (\llbracket c_0 \rrbracket_\sigma)$$

$$\llbracket \text{newvar } v := e \text{ in } c \rrbracket_\sigma = (\lambda \sigma' \in \Sigma. [\sigma' | v : \sigma v])_+ (\llbracket c \rrbracket [\sigma | v : \llbracket e \rrbracket_\sigma])$$

Restantes ecuaciones semánticas

$$\llbracket c_0; c_1 \rrbracket_\sigma = \llbracket c_1 \rrbracket_* (\llbracket c_0 \rrbracket_\sigma)$$

$$\llbracket \text{catchin } c_0 \text{ with } c_1 \rrbracket_\sigma = \llbracket c_1 \rrbracket_+ (\llbracket c_0 \rrbracket_\sigma)$$

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

$$\llbracket \text{while } b \text{ do } c \rrbracket = \bigsqcup_{i=0}^{\infty} F^i \perp_{\Sigma \rightarrow \Sigma \perp}$$

donde

$$F \ w \ \sigma = \begin{cases} w_* (\llbracket c \rrbracket_\sigma) & \text{si } \llbracket b \rrbracket_\sigma \\ \sigma & \text{si no} \end{cases}$$

Semántica de transiciones

Describe cómo se realiza el cómputo.

Semántica de transiciones

Describe cómo se realiza el cómputo.

$\langle x := 1; y := 2 * y, \sigma \rangle$

Semántica de transiciones

Describe cómo se realiza el cómputo.

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

Semántica de transiciones

Describe cómo se realiza el cómputo.

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

Semántica de transiciones

Describe cómo se realiza el cómputo.

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

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

Semántica de transiciones

Describe cómo se realiza el cómputo.

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

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:

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

Semántica de transiciones

Conjunto de configuraciones:

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

$\Gamma_t = \Sigma$ (configuraciones terminales)

$\Gamma_n = \langle comm \rangle \times \Sigma$ (configuraciones no terminales)

Definición axiomática de \rightarrow

$$\overline{\langle \mathbf{skip}, \sigma \rangle} \rightarrow \sigma$$

Definición axiomática de \rightarrow

$$\overline{\langle \mathbf{skip}, \sigma \rangle \rightarrow \sigma}$$

$$\overline{\langle v := e, \sigma \rangle \rightarrow [\sigma | v : \llbracket e \rrbracket \sigma]}$$

Definición axiomática de \rightarrow

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

Definición axiomática de \rightarrow

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

$$\frac{\langle c_0, \sigma \rangle \rightarrow \langle c'_0, \sigma' \rangle}{\langle c_0; c_1, \sigma \rangle \rightarrow \langle c'_0; c_1, \sigma' \rangle}$$

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:

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

Definición axiomática de \rightarrow

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

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

Definición axiomática de \rightarrow

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

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

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

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

Definición axiomática de \rightarrow

$$\frac{\langle c, [\sigma | v : \llbracket e \rrbracket_\sigma] \rangle \rightarrow \sigma'}{\langle \mathbf{newvar} \ v := e \ \mathbf{in} \ c, \sigma \rangle \rightarrow [\sigma' | v : \sigma v]}$$

$$\frac{\langle c, [\sigma | v : \llbracket e \rrbracket_\sigma] \rangle \rightarrow \langle c', \sigma' \rangle}{\langle \mathbf{newvar} \ v := e \ \mathbf{in} \ c, \sigma \rangle \rightarrow \langle \mathbf{newvar} \ v := \sigma' v \ \mathbf{in} \ c', [\sigma' | v : \sigma v] \rangle}$$

Determinismo y continuación

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

Determinismo y continuación

Determinismo: → 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 $c_0 \rightarrow c_1 \rightarrow c_2 \rightarrow \dots$ maximal, esto es, que no puede prolongarse más de lo que está.

Ejecución

Por *ejecución* entendemos una secuencia $c_0 \rightarrow c_1 \rightarrow c_2 \rightarrow \dots$ maximal, esto es, que no puede prolongarse más de lo que está. Dicha ejecución es infinita o termina en una configuración terminal σ .

Ejecución

Por *ejecución* entendemos una secuencia $c_0 \rightarrow c_1 \rightarrow c_2 \rightarrow \dots$ 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 $c_0 \uparrow$.

Ejecución

Por *ejecución* entendemos una secuencia $c_0 \rightarrow c_1 \rightarrow c_2 \rightarrow \dots$ 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 $c_0 \uparrow$.

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

Corrección de la semántica operacional

Lema 1

1 Si $\langle c_0, \sigma \rangle \rightarrow^* \sigma'$, entonces $\langle c_0; c_1, \sigma \rangle \rightarrow^* \langle c_1, \sigma' \rangle$

Corrección de la semántica operacional

Lema 1

1 Si $\langle c_0, \sigma \rangle \rightarrow^* \sigma'$, entonces $\langle c_0; c_1, \sigma \rangle \rightarrow^* \langle c_1, \sigma' \rangle$

2 Si $\langle c, [\sigma | v : \llbracket e \rrbracket \sigma] \rangle \rightarrow^* \sigma'$, entonces

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

Corrección de la semántica operacional

Lema 1

- 1 Si $\langle c_0, \sigma \rangle \rightarrow^* \sigma'$, entonces $\langle c_0; c_1, \sigma \rangle \rightarrow^* \langle c_1, \sigma' \rangle$
- 2 Si $\langle c, [\sigma | v : \llbracket e \rrbracket \sigma] \rangle \rightarrow^* \sigma'$, entonces

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

- 3 Si $\langle c, [\sigma | v : \llbracket e \rrbracket \sigma] \rangle \rightarrow^* \langle c', \sigma' \rangle$, entonces

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

Corrección de la semántica operacional

Lema 2

$$\textcircled{1} \quad \langle c, \sigma \rangle \rightarrow \sigma' \implies \llbracket c \rrbracket \sigma = \sigma'.$$

$$\textcircled{2} \quad \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle \implies \llbracket c \rrbracket \sigma = \llbracket c' \rrbracket \sigma'.$$

Corrección de la semántica operacional

Lema 2

- 1 $\langle c, \sigma \rangle \rightarrow \sigma' \implies \llbracket c \rrbracket \sigma = \sigma'$.
- 2 $\langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle \implies \llbracket c \rrbracket \sigma = \llbracket c' \rrbracket \sigma'$.

Lema 3 $\llbracket c \rrbracket \sigma = \sigma' \implies \langle c, \sigma \rangle \rightarrow^* \sigma'$

Corrección de la semántica operacional

Lema 2

$$\textcircled{1} \quad \langle c, \sigma \rangle \rightarrow \sigma' \implies \llbracket c \rrbracket \sigma = \sigma'.$$

$$\textcircled{2} \quad \langle c, \sigma \rangle \rightarrow \langle c', \sigma' \rangle \implies \llbracket c \rrbracket \sigma = \llbracket c' \rrbracket \sigma'.$$

$$\textbf{Lema 3} \quad \llbracket c \rrbracket \sigma = \sigma' \implies \langle c, \sigma \rangle \rightarrow^* \sigma'$$

Teorema Para todo comando c se tiene $\{\{c\}\} = \llbracket c \rrbracket$.

Semántica operacional de las fallas

Configuraciones terminales:

$$\Gamma_t = \Sigma \cup \{\mathbf{abort}\} \times \Sigma$$

Semántica operacional de las fallas

Configuraciones terminales:

$$\Gamma_t = \Sigma \cup \{\mathbf{abort}\} \times \Sigma$$

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

Semántica operacional de las fallas

Configuraciones terminales:

$$\Gamma_t = \Sigma \cup \{\mathbf{abort}\} \times \Sigma$$

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

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

Semántica operacional de las fallas

$$\frac{\langle c_0, \sigma \rangle \rightarrow \sigma'}{\langle \text{catchin } c_0 \text{ with } c_1, \sigma \rangle \rightarrow \sigma'}$$

Semántica operacional de las fallas

$$\frac{\langle c_0, \sigma \rangle \rightarrow \sigma'}{\langle \mathbf{catchin} \ c_0 \ \mathbf{with} \ c_1, \sigma \rangle \rightarrow \sigma'}$$

$$\frac{\langle c_0, \sigma \rangle \rightarrow \langle \mathbf{abort}, \sigma' \rangle}{\langle \mathbf{catchin} \ c_0 \ \mathbf{with} \ c_1, \sigma \rangle \rightarrow \langle c_1, \sigma' \rangle}$$

Semántica operacional de las fallas

$$\frac{\langle c_0, \sigma \rangle \rightarrow \sigma'}{\langle \mathbf{catchin} \ c_0 \ \mathbf{with} \ c_1, \sigma \rangle \rightarrow \sigma'}$$

$$\frac{\langle c_0, \sigma \rangle \rightarrow \langle \mathbf{abort}, \sigma' \rangle}{\langle \mathbf{catchin} \ c_0 \ \mathbf{with} \ c_1, \sigma \rangle \rightarrow \langle c_1, \sigma' \rangle}$$

$$\frac{\langle c_0, \sigma \rangle \rightarrow \langle c'_0, \sigma' \rangle}{\langle \mathbf{catchin} \ c_0 \ \mathbf{with} \ c_1, \sigma \rangle \rightarrow \langle \mathbf{catchin} \ c'_0 \ \mathbf{with} \ c_1, \sigma' \rangle}$$

Semántica operacional de las fallas

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

Semántica operacional de las fallas

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

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 $\langle \mathbf{abort}, \sigma \rangle$.

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 $\langle \mathbf{abort}, \sigma \rangle$.

Se puede definir:

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

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