



**PSC 2022/23** (375AA, 9CFU)

Principles for Software Composition

Roberto Bruni

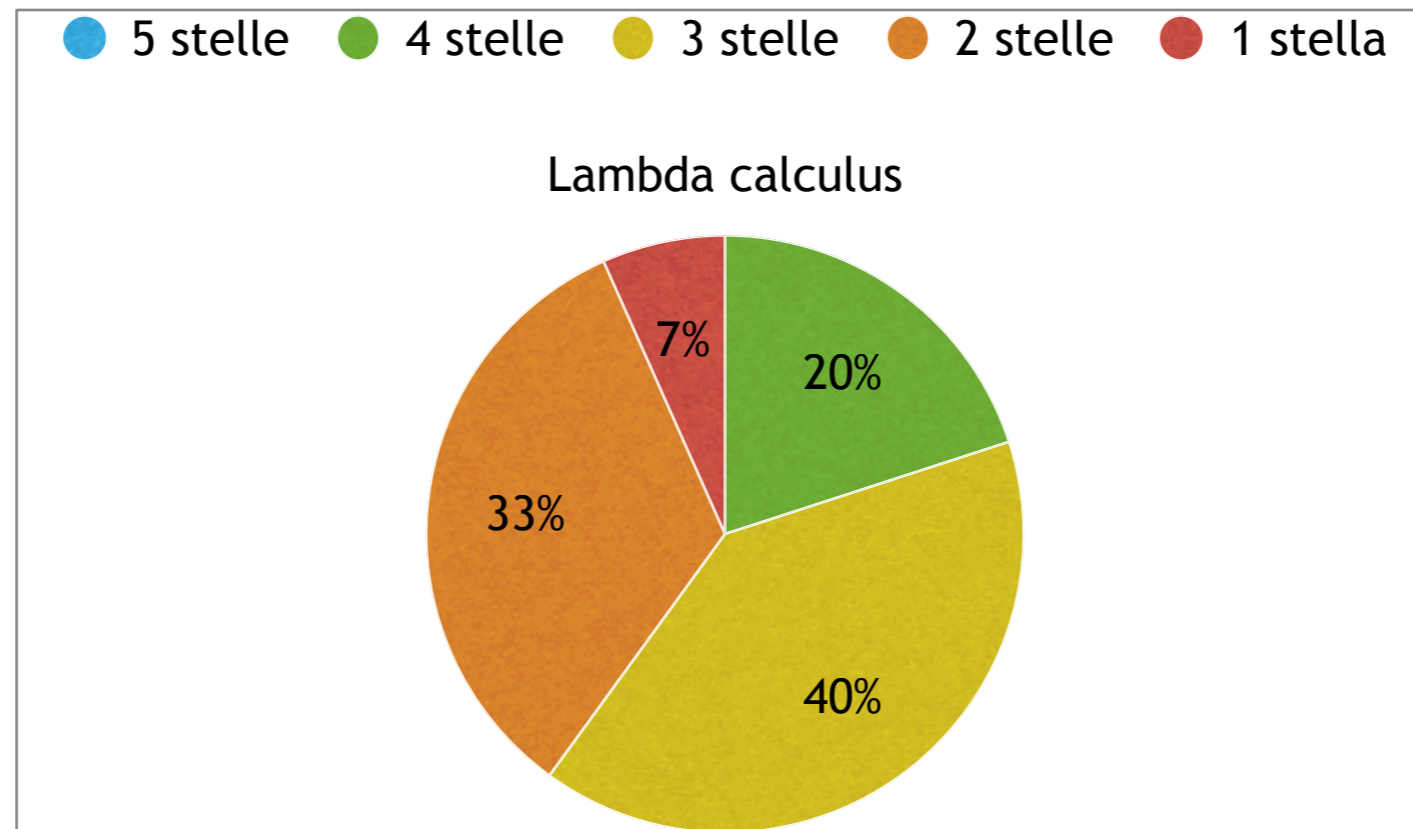
<http://www.di.unipi.it/~bruni/>

<http://didawiki.di.unipi.it/doku.php/magistraleinformatica/psc/start>

**09 - Denotational semantics of commands**

# Lambda notation

# From your forms



(over 15 answers)

# Lambda notation

## Key ingredients

anonymous functions

$\lambda x. e$   $x$  serves as a formal parameter in  $e$

denotes a function that waits for one value to be substituted for  $x$  and then evaluates  $e$

application

$e_1 e_2$   $e_2$  is the argument passed to the function  $e_1$

denotes the application of the function  $e_1$  to  $e_2$

reduces the need of parentheses  $e_1(e_2)$

# Function definition

$$f(x) \triangleq x^2 - 2 \cdot x + 5$$

$$f \triangleq \lambda x. (x^2 - 2 \cdot x + 5)$$

unnecessary parentheses  
added for clarity

# Associative rules

$e_1 e_2 e_3$  is read  $(e_1 e_2) e_3$  application is left-associative

$\lambda x. \lambda y. \lambda z. e$  is read  $\lambda x. (\lambda y. (\lambda z. e))$  abstraction is right-associative

# Scoping

$\lambda x. e$

the scope of  $x$  is  $e$

$x$  not visible outside  $e$

like a local variable

# Alpha-conversion

$\lambda x. (x^2 - 2 \cdot x + 5)$

names of formal parameters  
are inessential:

$\lambda y. (y^2 - 2 \cdot y + 5)$

the two expressions denote  
the same function

$\lambda x. e \equiv \lambda y. (e[y/x])$  (under suitable conditions on  $e, y$ )

capture-avoiding  
substitution

(to be formalised later)



# Application (beta rule)

$(\lambda x. e) e_0$

application of a function

$\equiv$

$e[e_0/x]$

evaluation via substitution

capture-avoiding  
substitution

# Example

$\lambda x. (x^2 - 2 \cdot x + 5)$  a function

$(\lambda x. (x^2 - 2 \cdot x + 5)) 2$  its application

$\equiv$

$2^2 - 2 \cdot 2 + 5 = 5$  its evaluation

# Example

$\lambda x. \lambda y. (x^2 - 2 \cdot y + 5)$  a function

$(\lambda x. \lambda y. (x^2 - 2 \cdot y + 5)) 2$  its application

$\equiv$

$\lambda y. (2^2 - 2 \cdot y + 5)$  its evaluation

it is still a function!

# Example

$\lambda f. \lambda x. (x^2 + f\ 1)$

a function

$(\lambda f. \lambda x. (x^2 + f\ 1)) (\lambda y. (2 \cdot y))$

its application

$\equiv$

(the argument is a function!)

$\lambda x. (x^2 + (\lambda y. (2 \cdot y))\ 1)$

its evaluation

higher-order: functions as arguments/results

# Example

$$\lambda f. \lambda x. (x^2 + f 1)$$

a function

$$(\lambda f. \lambda x. (x^2 + f 1)) (\lambda y. (2 \cdot y)) 3$$

its application

$\equiv$

$$\lambda x. (x^2 + (\lambda y. (2 \cdot y)) 1) 3$$

its evaluation  
its application

$\equiv$

$$3^2 + (\lambda y. (2 \cdot y)) 1$$

its evaluation  
its application

$\equiv$

$$3^2 + 2 \cdot 1 = 11$$

its evaluation

# Conditional

$e \rightarrow e_1, e_2$

if  $e$  then  $e_1$  else  $e_2$

example

$\min \triangleq \lambda x. \lambda y. x < y \rightarrow x, y$

# From recursion to fixpoint

$$\mathit{fact} \ n = (n < 2) \rightarrow 1, n \cdot \mathit{fact}(n - 1)$$

$$\mathit{fact} = \lambda n. (n < 2) \rightarrow 1, n \cdot \mathit{fact}(n - 1)$$

$$\mathit{fact} = (\lambda f. \lambda n. (n < 2) \rightarrow 1, n \cdot f(n - 1)) \ \mathit{fact}$$

$$\Gamma = \lambda f. \lambda n. (n < 2) \rightarrow 1, n \cdot f(n - 1)$$

$$\mathit{fact} = \Gamma(\mathit{fact})$$

$$\mathit{fact} = \mathit{fix} \ \Gamma$$

# From recursion to fixpoint

$$\Gamma = \lambda f . \lambda n . (n < 2) \rightarrow 1, n \cdot f(n - 1)$$

$$id = \lambda x . x$$

$$\Gamma id = (\lambda f . \lambda n . (n < 2) \rightarrow 1, n \cdot f(n - 1)) id$$

$$= \lambda n . (n < 2) \rightarrow 1, n \cdot id(n - 1)$$

$$= \lambda n . (n < 2) \rightarrow 1, n \cdot (n - 1)$$

$$\neq id$$



# From recursion to fixpoint

$$\Gamma = \lambda f . \lambda n . (n < 2) \rightarrow 1, n \cdot f(n - 1)$$

$$succ = \lambda x . x + 1$$

$$\begin{aligned} \Gamma succ &= (\lambda f . \lambda n . (n < 2) \rightarrow 1, n \cdot f(n - 1)) succ \\ &= \lambda n . (n < 2) \rightarrow 1, n \cdot succ(n - 1) \\ &= \lambda n . (n < 2) \rightarrow 1, n \cdot n \\ &\neq succ \end{aligned}$$

# From recursion to fixpoint

$$\Gamma = \lambda f . \lambda n . (n < 2) \rightarrow 1, n \cdot f(n - 1)$$

$$\textit{square} = \lambda x . x^2$$

$$\Gamma \textit{ square} = (\lambda f . \lambda n . (n < 2) \rightarrow 1, n \cdot f(n - 1)) \textit{ square}$$

$$= \lambda n . (n < 2) \rightarrow 1, n \cdot \textit{square}(n - 1)$$

$$= \lambda n . (n < 2) \rightarrow 1, n \cdot (n - 1)^2$$

$$\neq \textit{square}$$

# From recursion to fixpoint

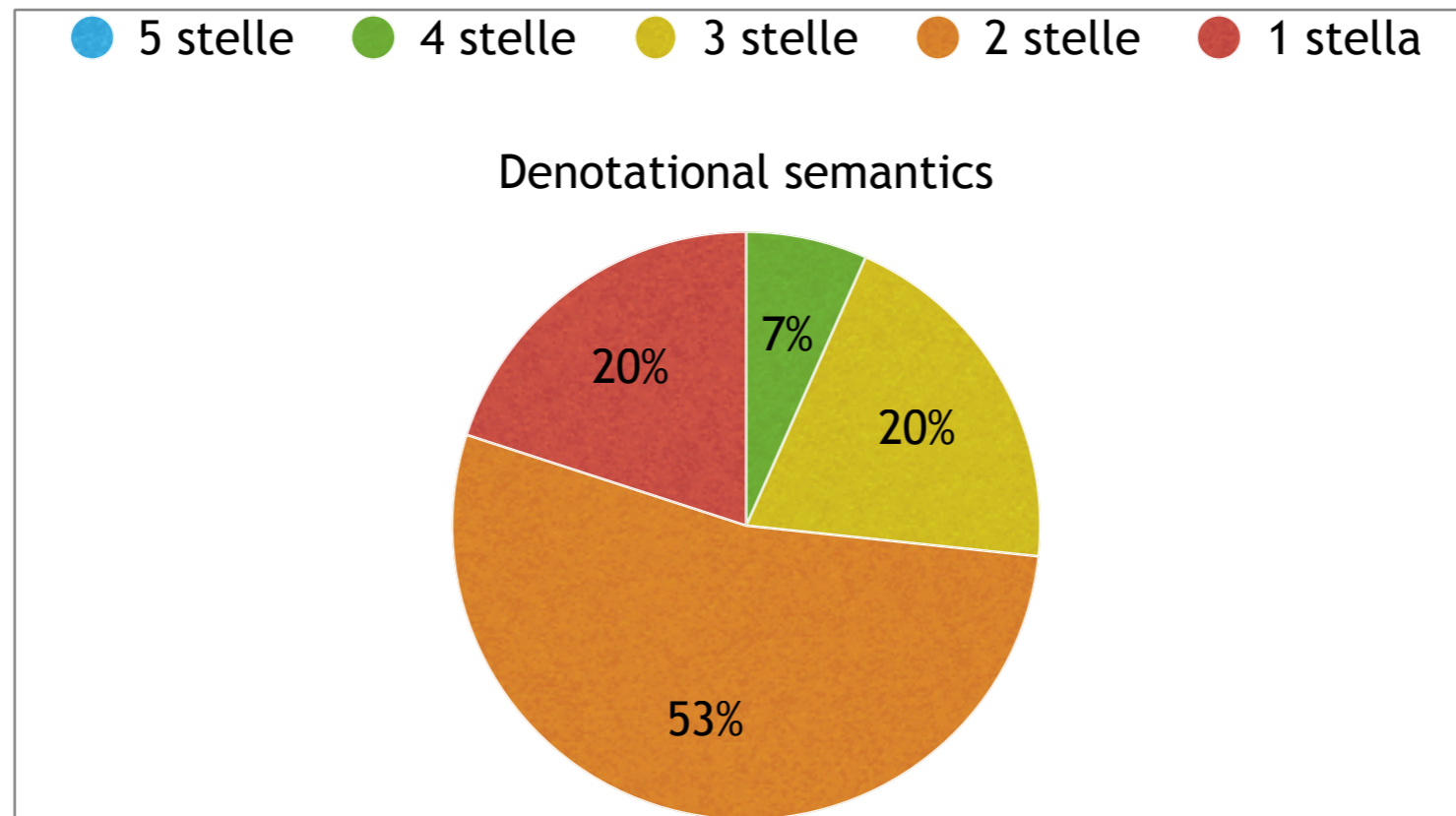
$$\Gamma = \lambda f . \lambda n . (n < 2) \rightarrow 1 , n \cdot f(n - 1)$$

$$fact = \lambda x . x!$$

$$\begin{aligned} \Gamma fact &= (\lambda f . \lambda n . (n < 2) \rightarrow 1 , n \cdot f(n - 1)) fact \\ &= \lambda n . (n < 2) \rightarrow 1 , n \cdot fact(n - 1) \\ &= \lambda n . (n < 2) \rightarrow 1 , n \cdot (n - 1)! \\ &= fact \end{aligned}$$

# Denotational semantics of commands

# From your forms



(over 15 answers)

# Denotational semantics

$$\mathcal{C} : Com \rightarrow (\Sigma \rightarrow \Sigma)$$

$$\mathcal{C} : Com \rightarrow (\Sigma \rightarrow \Sigma_{\perp})$$

$$\mathcal{C} [\mathbf{skip}] \sigma \stackrel{\text{def}}{=} \sigma$$

$$\mathcal{C} [x := a] \sigma \stackrel{\text{def}}{=} \sigma[\mathcal{A} [a] \sigma / x]$$

$$\mathcal{C} [c_0; c_1] \sigma \stackrel{\text{def}}{=} \mathcal{C} [c_1]^* (\mathcal{C} [c_0] \sigma)$$

$$\mathcal{C} [\mathbf{if } b \mathbf{ then } c_0 \mathbf{ else } c_1] \sigma \stackrel{\text{def}}{=} \mathcal{B} [b] \sigma \rightarrow \mathcal{C} [c_0] \sigma, \mathcal{C} [c_1] \sigma$$

$$\mathcal{C} [\mathbf{while } b \mathbf{ do } c] \sigma \stackrel{\text{def}}{=} ?$$

## Lifting

$$(\cdot)^* : (\Sigma \rightarrow \Sigma_{\perp}) \rightarrow (\Sigma_{\perp} \rightarrow \Sigma_{\perp})$$

$$f : \Sigma \rightarrow \Sigma_{\perp} \quad f^* : \Sigma_{\perp} \rightarrow \Sigma_{\perp}$$

$$f^*(x) = \begin{cases} \perp & \text{if } x = \perp \\ f(x) & \text{otherwise} \end{cases}$$

# Denotational sem. (ctd)

$$\mathcal{C} \llbracket \mathbf{while} \ b \ \mathbf{do} \ c \rrbracket \sigma \stackrel{\text{def}}{=} \mathcal{B} \llbracket b \rrbracket \sigma \rightarrow \mathcal{C} \llbracket \mathbf{while} \ b \ \mathbf{do} \ c \rrbracket^* (\mathcal{C} \llbracket c \rrbracket \sigma), \sigma$$

$$\mathcal{C} \llbracket \mathbf{while} \ b \ \mathbf{do} \ c \rrbracket \stackrel{\text{def}}{=} \lambda \sigma. \mathcal{B} \llbracket b \rrbracket \sigma \rightarrow \mathcal{C} \llbracket \mathbf{while} \ b \ \mathbf{do} \ c \rrbracket^* (\mathcal{C} \llbracket c \rrbracket \sigma), \sigma$$

$\equiv$

$$(\lambda \varphi. \lambda \sigma. \mathcal{B} \llbracket b \rrbracket \sigma \rightarrow \varphi^* (\mathcal{C} \llbracket c \rrbracket \sigma), \sigma) \ \mathcal{C} \llbracket \mathbf{while} \ b \ \mathbf{do} \ c \rrbracket$$

$$\Gamma_{b,c} \stackrel{\text{def}}{=} \lambda \varphi. \lambda \sigma. \mathcal{B} \llbracket b \rrbracket \sigma \rightarrow \varphi^* (\mathcal{C} \llbracket c \rrbracket \sigma), \sigma$$

$$\mathcal{C} \llbracket \mathbf{while} \ b \ \mathbf{do} \ c \rrbracket = \Gamma_{b,c} \ \mathcal{C} \llbracket \mathbf{while} \ b \ \mathbf{do} \ c \rrbracket$$

$p = f(p)$  a fixpoint equation!

# Denotational sem. (ctd)

$$\underbrace{\mathcal{C} [\mathbf{while} \ b \ \mathbf{do} \ c]}_{\Sigma \rightarrow \Sigma_{\perp}} = \Gamma_{b,c} \underbrace{\mathcal{C} [\mathbf{while} \ b \ \mathbf{do} \ c]}_{\Sigma \rightarrow \Sigma_{\perp}}$$

$$\mathcal{C} : Com \rightarrow (\Sigma \rightarrow \Sigma_{\perp})$$

$$\Gamma_{b,c} \stackrel{\text{def}}{=} \lambda \varphi. \lambda \sigma. \underbrace{\mathcal{B} [b] \sigma}_{\Sigma_{\perp}} \rightarrow \underbrace{\varphi^* (\mathcal{C} [c] \sigma)}_{\Sigma_{\perp}}, \sigma$$

$$\underbrace{\lambda \varphi. \lambda \sigma. \mathcal{B} [b] \sigma \rightarrow \varphi^* (\mathcal{C} [c] \sigma), \sigma}_{\Sigma \rightarrow \Sigma_{\perp}}$$

$$\underbrace{\lambda \varphi. \lambda \sigma. \mathcal{B} [b] \sigma \rightarrow \varphi^* (\mathcal{C} [c] \sigma), \sigma}_{(\Sigma \rightarrow \Sigma_{\perp}) \rightarrow \Sigma \rightarrow \Sigma_{\perp}}$$

$$\varphi : \Sigma \rightarrow \Sigma_{\perp}$$

$$\varphi^* : \Sigma_{\perp} \rightarrow \Sigma_{\perp}$$

$$\mathcal{C} [c] \sigma : \Sigma_{\perp}$$

$$\varphi^* (\mathcal{C} [c] \sigma) : \Sigma_{\perp}$$

$$\Gamma_{b,c} : (\Sigma \rightarrow \Sigma_{\perp}) \rightarrow \Sigma \rightarrow \Sigma_{\perp}$$

partial functions

$$\Sigma \rightarrow \Sigma$$

sets of pairs

$$(\sigma, \sigma')$$

CPO<sub>⊥</sub>



# Monotone and continuous

$$\Gamma_{b,c} \stackrel{\text{def}}{=} \lambda \varphi. \lambda \sigma. \mathcal{B} \llbracket b \rrbracket \sigma \rightarrow \varphi^*(\mathcal{C} \llbracket c \rrbracket \sigma), \sigma$$

Take  $R_{b,c} = \left\{ \frac{(\sigma'', \sigma')}{(\sigma, \sigma')} \mathcal{B} \llbracket b \rrbracket \sigma \wedge \mathcal{C} \llbracket c \rrbracket \sigma = \sigma'' \ , \ \frac{}{(\sigma, \sigma)} \mathcal{B} \llbracket \neg b \rrbracket \sigma \right\}$

clearly  $\hat{R}_{b,c} = \Gamma_{b,c}$  when we see  $\Gamma_{b,c}$  as operating over partial functions

$\hat{R}_{b,c}$  is (monotone and) continuous, and so is  $\Gamma_{b,c}$

$$\mathcal{C} \llbracket \mathbf{while} \ b \ \mathbf{do} \ c \rrbracket \stackrel{\text{def}}{=} \text{fix } \Gamma_{b,c} = \bigsqcup_{n \in \mathbb{N}} \Gamma_{b,c}^n (\perp_{\Sigma \rightarrow \Sigma_{\perp}})$$

$$\begin{array}{c} \perp_{\Sigma \rightarrow \Sigma_{\perp}} \\ | \\ \lambda \sigma. \perp \end{array}$$

# Bottom

$\Sigma_{\perp}$  has a bottom element:  $\perp$

$\Sigma \rightarrow \Sigma_{\perp}$  has a bottom element:  $\lambda\sigma. \perp$

to avoid ambiguities

we denote the bottom element of a domain  $D$  by  $\perp_D$

$\perp_{\Sigma_{\perp}}$

$\perp_{\Sigma \rightarrow \Sigma_{\perp}}$

# Example

$w = \text{while true do skip}$

$$\begin{aligned}\Gamma_{\text{true,skip}} \varphi \sigma &= \mathcal{B} [\text{true}] \sigma \rightarrow \varphi^* (\mathcal{C} [\text{skip}] \sigma), \sigma \\ &= \text{true} \rightarrow \varphi^* (\mathcal{C} [\text{skip}] \sigma), \sigma \\ &= \varphi^* (\mathcal{C} [\text{skip}] \sigma) \\ &= \varphi^* \sigma \\ &= \varphi \sigma\end{aligned}$$

$\Gamma_{\text{true,skip}} \varphi = \varphi$        $\Gamma_{\text{true,skip}}$  is the identity function  
every element is a  
fixpoint

$$\text{fix } \Gamma_{\text{true,skip}} = \lambda \sigma. \perp_{\Sigma_{\perp}}$$

# Example

$$w \triangleq \text{while } \underbrace{x > 1}_b \text{ do } \underbrace{x := x - 1}_c$$

$$\begin{aligned} \Gamma_{b,c} \varphi \sigma &= \mathcal{B}[x > 1]\sigma \rightarrow \varphi^*(\mathcal{C}[x := x - 1]\sigma), \sigma \\ &= (\sigma(x) > 1) \rightarrow \varphi^*(\sigma[\sigma(x)-1/x]), \sigma \end{aligned}$$

$$\widehat{R}_{b,c} \triangleq \left\{ \frac{(\sigma, \sigma)}{(\sigma, \sigma)} \sigma(x) \leq 1 \quad , \quad \frac{(\sigma'', \sigma')}{(\sigma, \sigma')} \sigma(x) > 1 \wedge \sigma'' = \sigma[\sigma(x)-1/x] \right\}$$

$$\widehat{R}_{b,c} \triangleq \left\{ \frac{(\sigma, \sigma)}{(\sigma, \sigma)} \sigma(x) \leq 1 \quad , \quad \frac{(\sigma[\sigma(x)-1/x], \sigma')}{(\sigma, \sigma')} \sigma(x) > 1 \right\}$$

# Example

$w \triangleq \mathbf{while } x > 1 \mathbf{ do } x := x - 1$

$$\hat{R}_{b,c} \triangleq \left\{ \frac{(\sigma, \sigma)}{(\sigma, \sigma)} \sigma(x) \leq 1, \frac{(\sigma[\sigma(x)-1/x], \sigma')}{(\sigma, \sigma')} \sigma(x) > 1 \right\}$$

$$\hat{R}_{b,c}^0(\emptyset) = \emptyset$$

$$\hat{R}_{b,c}^1(\emptyset) = \{(\sigma, \sigma) \mid \sigma(x) \leq 1\}$$

$$\hat{R}_{b,c}^2(\emptyset) = \hat{R}_{b,c}^1(\emptyset) \cup \{(\sigma, \sigma[1/x]) \mid \sigma(x) = 2\}$$

$$\hat{R}_{b,c}^3(\emptyset) = \hat{R}_{b,c}^2(\emptyset) \cup \{(\sigma, \sigma[1/x]) \mid \sigma(x) = 3\}$$

...

$$\hat{R}_{b,c}^n(\emptyset) = \{(\sigma, \sigma) \mid \sigma(x) \leq 1\} \cup \{(\sigma, \sigma[1/x]) \mid 1 < \sigma(x) \leq n\}$$

...

$$\mathcal{C}[[w]] = \text{fix}(\hat{R}_{b,c}) = \{(\sigma, \sigma) \mid \sigma(x) \leq 1\} \cup \{(\sigma, \sigma[1/x]) \mid 1 < \sigma(x)\}$$