

MPP 2026/27 (0077A, 9CFU)

Models for Programming Paradigms

Roberto Bruni

Filippo Bonchi

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

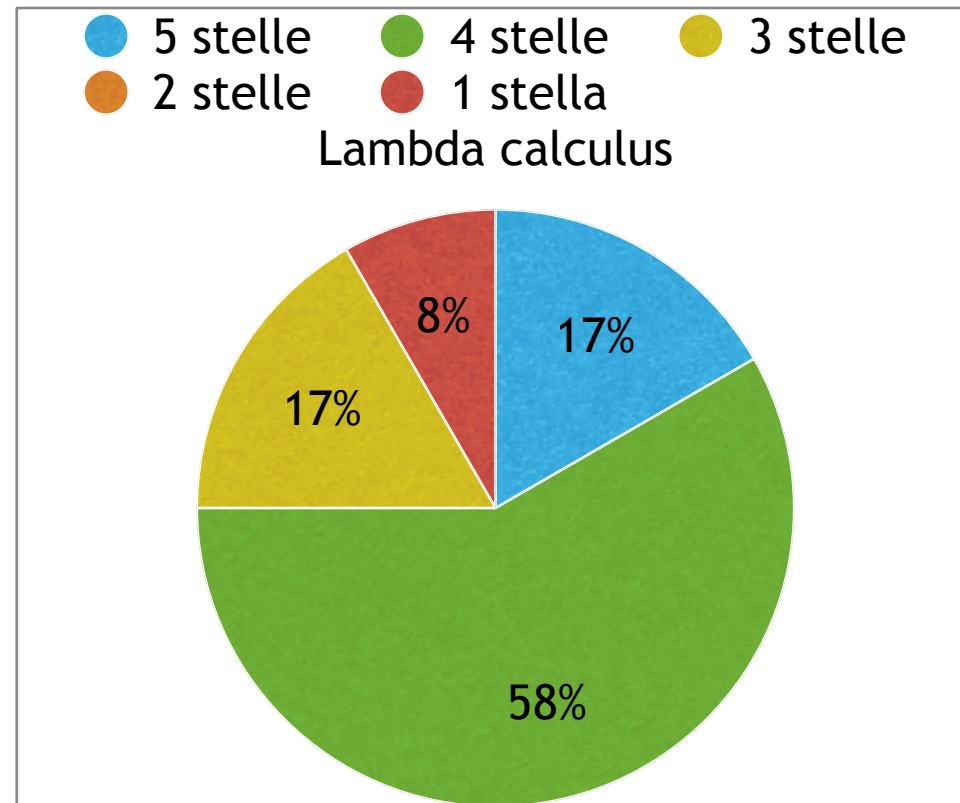


<https://didawiki.di.unipi.it/doku.php/magistraleinformatica/mpp/start>

09 - Denotational semantics of commands

Lambda notation

From your forms



(over 12 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

$\equiv$

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

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

$\text{min} \triangleq \lambda x. \lambda y. x < y \rightarrow x, y$

From recursion to fixpoint

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

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

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

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

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

$$\textit{fact} = \text{fix } \Gamma$$

From recursion to fixpoint

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

$$id = \lambda x . x$$

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

From recursion to fixpoint

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

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

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

From recursion to fixpoint

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

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

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

From recursion to fixpoint

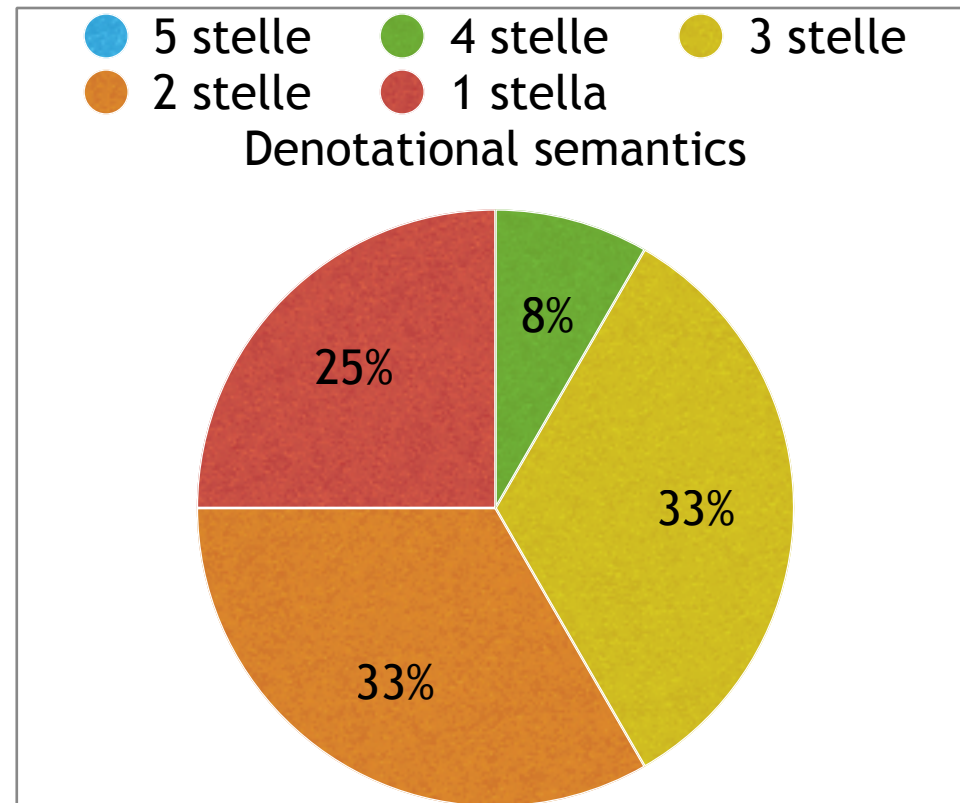
$$\Gamma = \lambda f . \lambda n . (n < 2) \rightarrow 1 , n \cdot f(n - 1)$$

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

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

Denotational semantics of commands

From your forms



(over 12 answers)

Denotational semantics

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

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

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

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

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

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

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

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)

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

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

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

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

Denotational sem. (ctd)

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

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

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

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

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

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

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

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

partial functions

$$\Sigma \multimap \Sigma$$

sets of pairs

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

$\text{CPO}_{\perp}$

Monotone and continuous

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

$$\text{Take } R_{b,c} = \left\{ \frac{(\sigma'', \sigma')}{(\sigma, \sigma')} \mathcal{B}[[b]]\sigma \wedge \mathcal{C}[[c]]\sigma = \sigma'' \quad , \quad \frac{}{(\sigma, \sigma)} \mathcal{B}[[\neg b]]\sigma \right\}$$

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

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

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

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

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

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

$$\Gamma_{\text{true,skip}} \varphi = \varphi$$

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

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

Example

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

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

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

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

Example

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

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

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

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

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

$\sigma(x) = 2$ if $(\sigma[\sigma(x)-1/x], \sigma') = (\sigma[1/x], \sigma') \in \hat{R}_{b,c}^1(\emptyset)$, then $(\sigma, \sigma') \in \hat{R}_{b,c}^2(\emptyset)$ since $(\sigma[1/x], \sigma[1/x]) \in \hat{R}_{b,c}^1(\emptyset)$, then $(\sigma, \sigma[1/x]) \in \hat{R}_{b,c}^2(\emptyset)$

$\sigma(x) = 3$ if $(\sigma[\sigma(x)-1/x], \sigma') = (\sigma[2/x], \sigma') \in \hat{R}_{b,c}^1(\emptyset)$, then $(\sigma, \sigma') \in \hat{R}_{b,c}^2(\emptyset)$ but $\forall \sigma'. (\sigma[2/x], \sigma') \notin \hat{R}_{b,c}^1(\emptyset)!$

Example

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

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

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

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

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

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

...

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

...

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

Exercises

[Exercise] Define by well-founded recursion the function $vars$ that, given an arithmetic expression a , returns the set of identifiers that appear in a . Then, prove by rule induction that $\forall a \in Aexp, \forall \sigma \in \Sigma, \forall n \in \mathbb{Z}$

$\langle a, \sigma \rangle \rightarrow n$ implies $\forall \sigma'. ((\forall y \in vars(a). \sigma(y) = \sigma'(y)) \Rightarrow \langle a, \sigma' \rangle \rightarrow n)$.

if two memories coincide on all variables
that appear in one expression, then
evaluating the expression in the two memories
give the same result

Vars #1

$$vars : Aexp \rightarrow \wp(Ids)$$

$$vars(n) \triangleq \emptyset$$

$$vars(x) \triangleq \{x\}$$

$$vars(a_1 \text{ op } a_2) \triangleq vars(a_1) \cup vars(a_2)$$

(well founded recursion by
immediate subterm relation)

Vars #1

$$P(\langle a, \sigma \rangle \rightarrow n) \triangleq \forall \sigma'. (\forall y \in \text{vars}(a). \sigma'(y) = \sigma(y)) \Rightarrow \langle a, \sigma' \rangle \rightarrow n$$

$$\frac{}{\langle n, \sigma \rangle \rightarrow n} \text{ (num)}$$

$$P(\langle n, \sigma \rangle \rightarrow n) \triangleq \forall \sigma'. \boxed{(\forall y \in \boxed{\text{vars}(n)}. \sigma'(y) = \sigma(y))} \Rightarrow \langle n, \sigma' \rangle \rightarrow n$$

tt

$$\text{by (num) } \langle n, \sigma' \rangle \rightarrow n$$

Vars #1

$$P(\langle a, \sigma \rangle \rightarrow n) \triangleq \forall \sigma'. (\forall y \in \text{vars}(a). \sigma'(y) = \sigma(y)) \Rightarrow \langle a, \sigma' \rangle \rightarrow n$$

$$\frac{}{\langle x, \sigma \rangle \rightarrow \sigma(x)} \text{(ide)}$$

$$P(\langle x, \sigma \rangle \rightarrow \sigma(x)) \triangleq \forall \sigma' \left[\begin{array}{c} (\forall y \in \boxed{\text{vars}(x)} \cdot \sigma'(y) = \sigma(y)) \\ \{x\} \end{array} \Rightarrow \langle x, \sigma' \rangle \rightarrow \sigma(x) \right]$$
$$\sigma'(x) = \sigma(x)$$

Assume $\sigma'(x) = \sigma(x)$

by (ide) $\langle x, \sigma' \rangle \rightarrow \sigma'(x) = \sigma(x)$

Vars #1

$$\frac{\langle a_1, \sigma \rangle \rightarrow n_1 \quad \langle a_2, \sigma \rangle \rightarrow n_2}{\langle a_1 \text{ op } a_2, \sigma \rangle \rightarrow n_1 \text{ op } n_2}$$

Assume

$$P(\langle a_i, \sigma \rangle \rightarrow n_i) \triangleq \forall \sigma'. (\forall y \in \text{vars}(a_i). \sigma'(y) = \sigma(y)) \Rightarrow \langle a_i, \sigma' \rangle \rightarrow n_i$$

We want to prove

$$P(\langle a_1 \text{ op } a_2, \sigma \rangle \rightarrow n_1 \text{ op } n_2) \triangleq \forall \sigma'.$$

$$(\forall y \in \text{vars}(a_1 \text{ op } a_2). \sigma'(y) = \sigma(y)) \Rightarrow \langle a_1 \text{ op } a_2, \sigma' \rangle \rightarrow n_1 \text{ op } n_2$$

Assume $\forall y \in \boxed{\text{vars}(a_1 \text{ op } a_2)} \sigma'(y) = \sigma(y)$

$$\text{vars}(a_1) \cup \text{vars}(a_2)$$

$$(\forall y \in \text{vars}(a_1). \sigma'(y) = \sigma(y)) \wedge (\forall y \in \text{vars}(a_2). \sigma'(y) = \sigma(y))$$

by inductive hypotheses

$$\langle a_1, \sigma' \rangle \rightarrow n_1 \quad \langle a_2, \sigma' \rangle \rightarrow n_2$$

by (op) $\langle a_1 \text{ op } a_2, \sigma' \rangle \rightarrow n_1 \text{ op } n_2$

[Exercise] Define by well-founded recursion the function $vars$ that, given a command, returns the set of identifiers that appear on the left-hand side of some assignment. Then, prove by rule induction that $\forall c \in Com, \forall \sigma, \sigma' \in \Sigma$

$$\langle c, \sigma \rangle \rightarrow \sigma' \quad \text{implies} \quad \forall x \notin vars(c). \sigma(x) = \sigma'(x).$$

if a variable does not appear in an assignment
then its initial value is preserved in the final store

Vars #2

$$vars : Com \rightarrow \wp(Ide)$$

$$\begin{aligned} vars(\mathbf{skip}) &\triangleq \emptyset \\ vars(x := a) &\triangleq \{x\} \\ vars(c_1; c_2) &\triangleq vars(c_1) \cup vars(c_2) \\ vars(\mathbf{if } b \mathbf{ then } c_1 \mathbf{ else } c_2) &\triangleq vars(c_1) \cup vars(c_2) \\ vars(\mathbf{while } b \mathbf{ do } c) &\triangleq vars(c) \end{aligned}$$

(well founded recursion by
immediate subterm relation)

Vars #2

$$P(\langle c, \sigma \rangle \rightarrow \sigma') \triangleq \forall y \notin \text{vars}(c). \sigma'(y) = \sigma(y)$$

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

We want to prove

$$P(\langle \mathbf{skip}, \sigma \rangle \rightarrow \sigma) \triangleq \forall y \notin \boxed{\text{vars}(\mathbf{skip})} \sigma(y) = \sigma(y)$$

$\emptyset$

$$\forall y. \sigma(y) = \sigma(y)$$

obvious

Vars #2

$$P(\langle c, \sigma \rangle \rightarrow \sigma') \triangleq \forall y \notin \text{vars}(c). \sigma'(y) = \sigma(y)$$

$$\frac{\langle a, \sigma \rangle \rightarrow n}{\langle x := a, \sigma \rangle \rightarrow \sigma[n/x]}$$

We want to prove

$$P(\langle x := a, \sigma \rangle \rightarrow \sigma[n/x]) \triangleq \forall y \notin \boxed{\text{vars}(x := a)} \cdot \sigma[n/x](y) = \sigma(y)$$

$\{x\}$

$$\forall y \neq x. \sigma[n/x](y) = \sigma(y)$$

by def of $\sigma[n/x]$

Vars #2

$$\frac{\langle c_1, \sigma \rangle \rightarrow \sigma'' \quad \langle c_2, \sigma'' \rangle \rightarrow \sigma'}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma'}$$

Assume $P(\langle c_1, \sigma \rangle \rightarrow \sigma'') \triangleq \forall y \notin \text{vars}(c_1). \sigma''(y) = \sigma(y)$

$P(\langle c_2, \sigma'' \rangle \rightarrow \sigma') \triangleq \forall y \notin \text{vars}(c_2). \sigma'(y) = \sigma''(y)$

We want to prove

$$P(\langle c_1; c_2, \sigma \rangle \rightarrow \sigma') \triangleq \forall y \notin \boxed{\text{vars}(c_1; c_2)}. \sigma'(y) = \sigma(y)$$

$$\text{vars}(c_1) \cup \text{vars}(c_2)$$

$$\forall y \notin \text{vars}(c_1) \cup \text{vars}(c_2). \sigma'(y) = \sigma(y)$$

Take $y \notin \text{vars}(c_1) \cup \text{vars}(c_2)$

Since $y \notin \text{vars}(c_2)$ then by ind. hyp. $\sigma'(y) = \sigma''(y)$

Since $y \notin \text{vars}(c_1)$ then by ind. hyp. $\sigma''(y) = \sigma(y)$

Thus $\sigma'(y) = \sigma(y)$

Vars #2

$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{tt} \quad \langle c_1, \sigma \rangle \rightarrow \sigma'}{\langle \mathbf{if } b \mathbf{ then } c_1 \mathbf{ else } c_2, \sigma \rangle \rightarrow \sigma'}$$

Assume $P(\langle c_1, \sigma \rangle \rightarrow \sigma') \triangleq \forall y \notin \text{vars}(c_1). \sigma'(y) = \sigma(y)$

We want to prove

$$P(\langle \mathbf{if } b \mathbf{ then } c_1 \mathbf{ else } c_2, \sigma \rangle \rightarrow \sigma') \triangleq \forall y \notin \boxed{\begin{array}{c} \text{vars}(\mathbf{if } b \mathbf{ then } c_1 \mathbf{ else } c_2) \\ \text{vars}(c_1) \cup \text{vars}(c_2) \end{array}} \sigma'(y) = \sigma(y)$$
$$\forall y \notin \text{vars}(c_1) \cup \text{vars}(c_2). \sigma'(y) = \sigma(y)$$

Take $y \notin \text{vars}(c_1) \cup \text{vars}(c_2)$

Since $y \notin \text{vars}(c_1)$ then by ind. hyp. $\sigma'(y) = \sigma(y)$

Vars #2

$$\frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma}$$

We want to prove

$$P(\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma) \triangleq \forall y \notin \boxed{\text{vars}(\text{while } b \text{ do } c)} \cdot \sigma(y) = \sigma(y)$$

$$\text{vars}(c)$$

$$\forall y \notin \text{vars}(c). \sigma(y) = \sigma(y)$$

obvious

Vars #2

$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{true} \quad \langle c, \sigma \rangle \rightarrow \sigma'' \quad \langle \mathbf{while } b \mathbf{ do } c, \sigma'' \rangle \rightarrow \sigma'}{\langle \mathbf{while } b \mathbf{ do } c, \sigma \rangle \rightarrow \sigma'}$$

Assume $P(\langle c, \sigma \rangle \rightarrow \sigma'') \triangleq \forall y \notin \mathit{vars}(c). \sigma''(y) = \sigma(y)$

$$P(\langle \mathbf{while } b \mathbf{ do } c, \sigma'' \rangle \rightarrow \sigma') \triangleq \forall y \notin \boxed{\mathit{vars}(\mathbf{while } b \mathbf{ do } c)} \sigma'(y) = \sigma''(y)$$

$\mathit{vars}(c)$

We want to prove

$$P(\langle \mathbf{while } b \mathbf{ do } c, \sigma \rangle \rightarrow \sigma') \triangleq \forall y \notin \boxed{\mathit{vars}(\mathbf{while } b \mathbf{ do } c)} \sigma'(y) = \sigma(y)$$

$\mathit{vars}(c)$

$\forall y \notin \mathit{vars}(c). \sigma'(y) = \sigma(y)$

Take $y \notin \mathit{vars}(c)$

Since $y \notin \mathit{vars}(c)$ then by ind. hyp. $\sigma'(y) = \sigma''(y)$

Since $y \notin \mathit{vars}(c)$ then by ind. hyp. $\sigma''(y) = \sigma(y)$

Thus $\sigma'(y) = \sigma(y)$

11/09/2018

[Ex. 1] Suppose we replace the **while-do** command of IMP with the command **for a do c** whose operational semantics is defined by the rules:

$$\frac{\langle a, \sigma \rangle \rightarrow 0}{\langle \mathbf{for\ } a \mathbf{\ do\ } c, \sigma \rangle \rightarrow \sigma} \quad \frac{\langle a, \sigma \rangle \rightarrow n \neq 0 \quad \langle c, \sigma \rangle \rightarrow \sigma'' \quad \langle \mathbf{for\ } a - 1 \mathbf{\ do\ } c, \sigma'' \rangle \rightarrow \sigma'}{\langle \mathbf{for\ } a \mathbf{\ do\ } c, \sigma \rangle \rightarrow \sigma'}$$

1. Show a **for-do** program that diverges.
2. Extend the proof of determinacy taking into account the new construct.