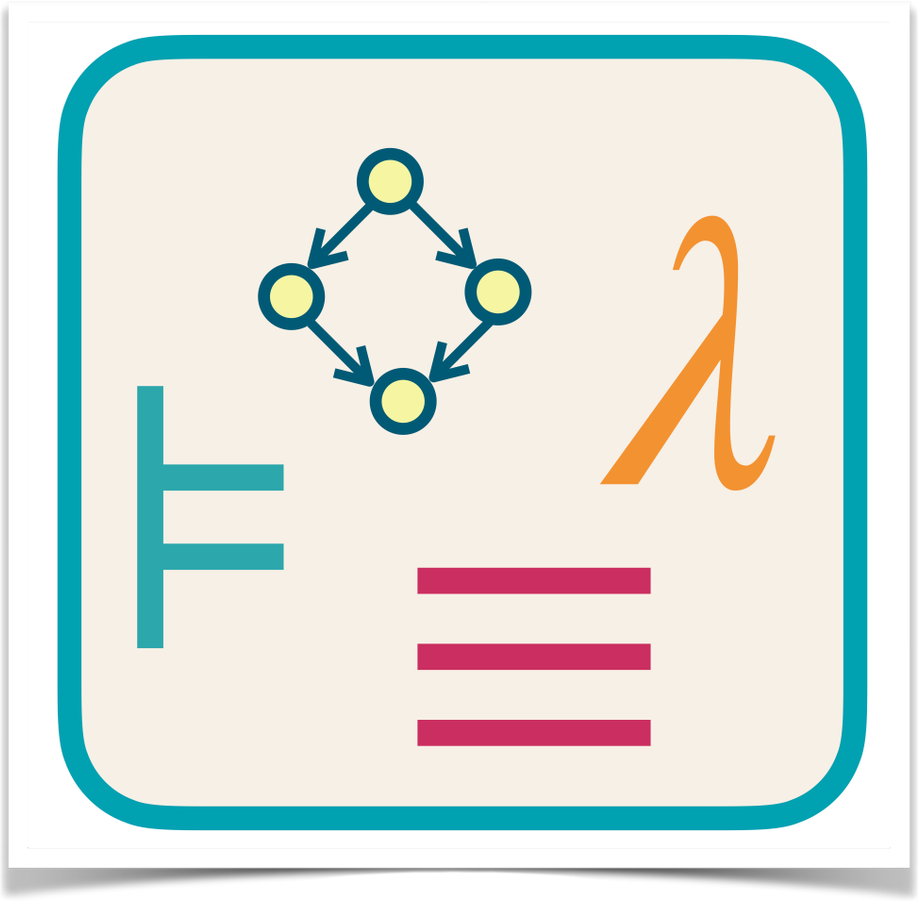


LPL 2026/27 (063AA, 9CFU)

Programming Languages

Fabio Gadducci

and Laboratory

Vincenzo Ciancia

09 - Denotational semantics of commands

Lambda notation

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

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

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

names of formal parameters are inessential: 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

$\equiv$

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

$e \to e_1, e_2$ if e then e_1 else e_2

example $\min \triangleq \lambda x.\ \lambda y.\ x < y \to x, y$

From recursion to fixpoint

$$fact\ n = (n < 2) \to 1\ , n \cdot fact(n-1)$$

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

$$fact = (\lambda f\ .\ \lambda n\ .\ (n < 2) \to 1\ , n \cdot f(n-1))\ fact$$

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

$$fact = \Gamma(fact)$$

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

From recursion to fixpoint

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

$$id = \lambda x\,.\,x$$

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

From recursion to fixpoint

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

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

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

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

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

$$\neq succ$$

From recursion to fixpoint

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

$$square = \lambda x\,.\,x^2$$

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

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

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

$$\neq square$$

From recursion to fixpoint

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

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

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

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

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

$$= fact$$

Denotational semantics of commands

Denotational semantics

$\mathscr{C} : Com \to (\Sigma \rightharpoonup \Sigma)$

$\mathscr{C} : Com \to (\Sigma \to \Sigma_\perp)$

$$\mathscr{C}[\![\textbf{skip}]\!]\,\sigma \stackrel{\text{def}}{=} \sigma$$

$$\mathscr{C}[\![x := a]\!]\,\sigma \stackrel{\text{def}}{=} \sigma[{}^{\mathscr{A}[\![a]\!]\sigma}/_{x}]$$

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

Lifting

$$(\cdot)^* : (\Sigma \to \Sigma_\perp) \to (\Sigma_\perp \to \Sigma_\perp)$$

$$f : \Sigma \to \Sigma_\perp \qquad f^* : \Sigma_\perp \to \Sigma_\perp$$

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

$$\mathscr{C}[\![\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1]\!]\,\sigma \stackrel{\text{def}}{=} \mathscr{B}[\![b]\!]\,\sigma \to \mathscr{C}[\![c_0]\!]\,\sigma, \mathscr{C}[\![c_1]\!]\,\sigma$$

$$\mathscr{C}[\![\textbf{while } b \textbf{ do } c]\!]\,\sigma \stackrel{\text{def}}{=} \ ?$$

Denotational sem. (ctd)

$$\mathscr{C}[\![\textbf{while } b \textbf{ do } c]\!]\,\sigma \stackrel{\text{def}}{=} \mathscr{B}[\![b]\!]\,\sigma \to \mathscr{C}[\![\textbf{while } b \textbf{ do } c]\!]^*(\mathscr{C}[\![c]\!]\,\sigma), \sigma$$

$$\mathscr{C}[\![\textbf{while } b \textbf{ do } c]\!] \stackrel{\text{def}}{=} \lambda\sigma.\, \mathscr{B}[\![b]\!]\,\sigma \to \mathscr{C}[\![\textbf{while } b \textbf{ do } c]\!]^*(\mathscr{C}[\![c]\!]\,\sigma), \sigma$$

$$\equiv$$

$$(\lambda\varphi.\ \lambda\sigma.\ \mathscr{B}[\![b]\!]\,\sigma \to \varphi^*(\mathscr{C}[\![c]\!]\,\sigma), \sigma)\ \ \mathscr{C}[\![\textbf{while } b \textbf{ do } c]\!]$$

$$\Gamma_{b,c} \stackrel{\text{def}}{=} \lambda\varphi.\ \lambda\sigma.\ \mathscr{B}[\![b]\!]\,\sigma \to \varphi^*(\mathscr{C}[\![c]\!]\,\sigma), \sigma$$

$$\mathscr{C}[\![\textbf{while } b \textbf{ do } c]\!] = \Gamma_{b,c}\,\mathscr{C}[\![\textbf{while } b \textbf{ do } c]\!]$$

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

Denotational sem. (ctd)

$$\underbrace{\mathscr{C}[\![\textbf{while } b \textbf{ do } c]\!]}_{\Sigma\to\Sigma_\perp} = \Gamma_{b,c}\,\underbrace{\mathscr{C}[\![\textbf{while } b \textbf{ do } c]\!]}_{\Sigma\to\Sigma_\perp}$$

$$\mathscr{C} : Com \to (\Sigma \to \Sigma_\perp)$$

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

$$\underbrace{\qquad\qquad\qquad\qquad}_{(\Sigma\to\Sigma_\perp)\to\Sigma\to\Sigma_\perp}$$

$$\varphi : \Sigma \to \Sigma_\perp$$

$$\varphi^* : \Sigma_\perp \to \Sigma_\perp$$

$$\mathscr{C}[\![c]\!]\,\sigma : \Sigma_\perp$$

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

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

partial functions

$$\Sigma \rightharpoonup \Sigma$$

sets of pairs

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

$$\mathrm{CPO}_\perp$$

Monotone and continuous

$$\Gamma_{b,c} \stackrel{\text{def}}{=} \lambda\varphi.\ \lambda\sigma.\ \mathscr{B}[\![b]\!]\sigma \to \varphi^*(\mathscr{C}[\![c]\!]\sigma), \sigma$$

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

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

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

$$\mathscr{C}[\![\textbf{while } b \textbf{ do } c]\!] \stackrel{\text{def}}{=} \text{fix } \Gamma_{b,c} = \bigsqcup_{n \in \mathbb{N}} \Gamma_{b,c}^n(\bot_{\Sigma \to \Sigma_\bot})$$

$$\bot_{\Sigma \to \Sigma_\bot} = \lambda\sigma.\ \bot$$

Bottom

$\Sigma_\bot$ has a bottom element: $\bot$

$\Sigma \to \Sigma_\bot$ has a bottom element: $\lambda\sigma.\ \bot$

to avoid ambiguities

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

$\bot_{\Sigma_\bot}$ $\bot_{\Sigma\to\Sigma_\bot}$

Example

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

$$\begin{aligned}
\Gamma_{\textbf{true},\textbf{skip}}\,\varphi\,\sigma &= \mathscr{B}\,[\![\textbf{true}]\!]\,\sigma \to \varphi^*\,(\mathscr{C}\,[\![\textbf{skip}]\!]\,\sigma)\,,\sigma \\
&= \textbf{true} \to \varphi^*\,(\mathscr{C}\,[\![\textbf{skip}]\!]\,\sigma)\,,\sigma \\
&= \varphi^*\,(\mathscr{C}\,[\![\textbf{skip}]\!]\,\sigma) \\
&= \varphi^*\,\sigma \\
&= \varphi\,\sigma
\end{aligned}$$

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

$$\text{fix}\ \Gamma_{\textbf{true},\textbf{skip}} = \lambda\sigma.\ \bot_{\Sigma_\bot}$$

Example

$$w \triangleq \textbf{while } \underbrace{x > 1}_{b} \textbf{ do } \underbrace{x := x - 1}_{c}$$

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

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

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

Example

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

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

$$\widehat{R}^0_{b,c}(\varnothing) = \varnothing$$

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

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

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

$$\cdots$$

$$\widehat{R}^n_{b,c}(\varnothing) = \{(\sigma,\sigma) \mid \sigma(x) \leq 1\} \cup \{(\sigma, \sigma[1/x]) \mid 1 < \sigma(x) \leq n\}$$

$$\cdots$$

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

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 $\forall a \in Aexp$, $\forall \sigma \in \Sigma$, $\forall n \in \mathbb{Z}$

$$\langle a, \sigma \rangle \to n \qquad \text{implies} \qquad \forall \sigma'.\, (\, (\forall y \in vars(a).\, \sigma(y) = \sigma'(y)) \Rightarrow \langle a, \sigma' \rangle \to n\,).$$

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

$$vars : Aexp \to \wp(Ide)$$

$$\begin{array}{rcl} vars(n) & \triangleq & \emptyset \\ vars(x) & \triangleq & \{x\} \\ vars(a_1 \text{ op } a_2) & \triangleq & vars(a_1) \cup vars(a_2) \end{array}$$

(well founded recursion by
immediate subterm relation)

Vars #1

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

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

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

$\emptyset$

$\mathbf{tt}$

by (num) $\langle n, \sigma'\rangle \to n$

Vars #1

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

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

$$P(\langle x, \sigma\rangle \to \sigma(x)) \triangleq \forall \sigma'.\ (\forall y \in \underbrace{vars(x)}_{\{x\}}.\ \sigma'(y) = \sigma(y)) \Rightarrow \langle x, \sigma'\rangle \to \sigma(x)$$

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

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

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

Vars #1

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

Assume

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

We want to prove

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

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

Assume $\forall y \in \underbrace{vars(a_1 \text{ op } a_2)}_{vars(a_1) \cup vars(a_2)}.\ \sigma'(y) = \sigma(y)$

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

by inductive hypotheses

$\langle a_1, \sigma'\rangle \to n_1 \qquad \langle a_2, \sigma'\rangle \to n_2$

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

[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 $\forall c \in Com$, $\forall \sigma, \sigma' \in \Sigma$

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

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

Vars #2

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

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

(well founded recursion by immediate subterm relation)

Vars #2

$$P(\langle c, \sigma\rangle \to \sigma') \triangleq \forall y \notin vars(c).\ \sigma'(y) = \sigma(y)$$

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

We want to prove

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

$$\emptyset$$

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

obvious

Vars #2

$$P(\langle c, \sigma\rangle \to \sigma') \triangleq \forall y \notin vars(c).\ \sigma'(y) = \sigma(y)$$

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

We want to prove

$$P(\langle x := a, \sigma\rangle \to \sigma[{}^{n}/_{x}]) \triangleq \boxed{\forall y \notin \underbrace{\boxed{vars(x := a)}}_{\{x\}}.\ \sigma[{}^{n}/_{x}](y) = \sigma(y)}$$

$$\forall y \neq x.\ \sigma[{}^{n}/_{x}](y) = \sigma(y)$$

by def of $\sigma[{}^{n}/_{x}]$

Vars #2

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

Assume $P(\langle c_1, \sigma\rangle \to \sigma'') \triangleq \forall y \notin vars(c_1).\ \sigma''(y) = \sigma(y)$

$P(\langle c_2, \sigma''\rangle \to \sigma') \triangleq \forall y \notin vars(c_2).\ \sigma'(y) = \sigma''(y)$

We want to prove

$$P(\langle c_1; c_2, \sigma\rangle \to \sigma') \triangleq \forall y \notin \underbrace{vars(c_1; c_2)}_{vars(c_1) \cup vars(c_2)}.\ \sigma'(y) = \sigma(y)$$

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

Take $y \notin vars(c_1) \cup vars(c_2)$

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

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

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

Vars #2

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

Assume $P(\langle c_1, \sigma\rangle \to \sigma') \triangleq \forall y \notin vars(c_1).\ \sigma'(y) = \sigma(y)$

We want to prove

$$P(\langle \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, \sigma\rangle \to \sigma') \triangleq \boxed{\forall y \notin \boxed{vars(\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2)}.\ \sigma'(y) = \sigma(y)}$$

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

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

Take $y \notin vars(c_1) \cup vars(c_2)$

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

Vars #2

$$\frac{\langle b, \sigma\rangle \to \textbf{false}}{\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \to \sigma}$$

We want to prove

$$P(\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \to \sigma) \triangleq \forall y \notin \boxed{vars(\textbf{while } b \textbf{ do } c)}. \ \sigma(y) = \sigma(y)$$

$$vars(c)$$

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

obvious

Vars #2

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

Assume $P(\langle c, \sigma\rangle \to \sigma'') \triangleq \forall y \notin vars(c).\ \sigma''(y) = \sigma(y)$

$$P(\langle \textbf{while } b \textbf{ do } c, \sigma''\rangle \to \sigma') \triangleq \forall y \notin \boxed{vars(\textbf{while } b \textbf{ do } c)}^{vars(c)}\ \sigma'(y) = \sigma''(y)$$

We want to prove

$$P(\langle \textbf{while } b \textbf{ do } c, \sigma\rangle \to \sigma') \triangleq \boxed{\forall y \notin \boxed{vars(\textbf{while } b \textbf{ do } c)}^{vars(c)}\ \sigma'(y) = \sigma(y)}^{\forall y \notin vars(c).\ \sigma'(y) = \sigma(y)}$$

Take $y \notin vars(c)$

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

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

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