Syntactic categories

- *numerals*

$$N \in \mathbf{Num}$$

with syntax given by:

$N ::= 0 \mid 1 \mid 2 \mid \cdots$

- *variables*

$$x \in \mathbf{Var}$$

with syntax given by:

$x ::= \cdots$ *sequences of letters and digits beginning with a letter* $\cdots$

- *(arithmetic) expressions*

$$e \in \mathbf{Exp}$$

with syntax given by:

$e ::= N \mid x \mid e_1 + e_2 \mid e_1 * e_2 \mid e_1 - e_2$

Inductive definitions

Free variables in expressions $FV(e) \subset \mathbf{Var}$:

$$\begin{aligned} FV(N) &= \emptyset \\ FV(x) &= \{x\} \\ FV(e_1 + e_2) &= FV(e_1) \cup FV(e_2) \\ FV(e_1 * e_2) &= FV(e_1) \cup FV(e_2) \\ FV(e_1 - e_2) &= FV(e_1) \cup FV(e_2) \end{aligned}$$

Fact: *For each expression $e \in \mathbf{Exp}$, the set $FV(e)$ of its free variables is finite.*

For any sets X and Y, define the *information ordering* on partial functions from X to Y:

$$f \sqsubseteq g \text{ iff } [f(x) \text{ is defined implies } g(x) \text{ is defined and } f(x) = g(x)]$$

This is a *complete partial order* (c.p.o.) on $X \rightharpoonup Y$, i.e., it has the least element $\emptyset_{X \rightharpoonup Y}$ and limits (least upper bounds) of increasing chains:

$$f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq f_3 \sqsubseteq \cdots \sqsubseteq \bigcup_{n \in \mathbb{N}} f_n$$

Putting $X = Y = \mathbf{State}$, we get a c.p.o. on the set $\mathbf{STMT}$.

Fact: *For any $S \in \mathbf{Stmt}$ and $b \in \mathbf{BExp}$, the function*

$$\Phi(F) = cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S]\!]; F, id_{\mathbf{State}})$$

is monotone and even continuous:

- $\Phi(f) \sqsubseteq \Phi(g)$ *if* $f \sqsubseteq g$,
- $\Phi\left(\bigcup_{n \in \mathbb{N}} f_n\right) = \bigcup_{n \in \mathbb{N}} \Phi(f_n)$ *if* $f_1 \sqsubseteq f_2 \sqsubseteq f_3 \sqsubseteq \cdots$.

Fact: (Kleene fixpoint theorem) *Every continuous function Φ on a c.p.o. X has a least fixpoint, defined by:*

$$\bigcup_{n \in \mathbb{N}} \Phi^n(\emptyset).$$

In practice, all "reasonable" functions that we are likely to write are continuous.

Structural induction

Given a property $P(\_)$ of expressions:

IF

- $P(N)$, for all $N \in \mathbf{Num}$
- $P(x)$, for all $x \in \mathbf{Var}$
- $P(e_1 + e_2)$ follows from $P(e_1)$ and $P(e_2)$, for all $e_1, e_2 \in \mathbf{Exp}$
- $P(e_1 * e_2)$ follows from $P(e_1)$ and $P(e_2)$, for all $e_1, e_2 \in \mathbf{Exp}$
- $P(e_1 - e_2)$ follows from $P(e_1)$ and $P(e_2)$, for all $e_1, e_2 \in \mathbf{Exp}$

THEN

- $P(e)$ for all $e \in \mathbf{Exp}$.

$$\mathcal{E} : \mathbf{Exp} \to (\mathbf{State} \to \mathbf{Int})$$

defined in the obvious way:

$$\begin{aligned} \mathcal{E}[\![N]\!]\, s &= \mathcal{N}[\![N]\!] \\ \mathcal{E}[\![x]\!]\, s &= s\, x \\ \mathcal{E}[\![e_1 + e_2]\!]\, s &= \mathcal{E}[\![e_1]\!]\, s + \mathcal{E}[\![e_2]\!]\, s \\ \mathcal{E}[\![e_1 * e_2]\!]\, s &= \mathcal{E}[\![e_1]\!]\, s * \mathcal{E}[\![e_2]\!]\, s \\ \mathcal{E}[\![e_1 - e_2]\!]\, s &= \mathcal{E}[\![e_1]\!]\, s - \mathcal{E}[\![e_2]\!]\, s \end{aligned}$$

Semantyka małych kroków

- define *configurations*: $\gamma \in \Gamma$
- indicate which of them are *terminal*: $\mathrm{T} \subseteq \Gamma$
- define a *(one-step) transition relation*: $\Rightarrow \subseteq \Gamma \times \Gamma$
  - for $\gamma \in \mathrm{T}$, typically $\gamma \not\Rightarrow$
- study *computations*: (finite or infinite) sequences of configurations

$$\gamma_0, \gamma_1, \ldots, \gamma_i, \gamma_{i+1}, \ldots,$$

such that $\gamma_i \Rightarrow \gamma_{i+1}$, written as:

$$\gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_i \Rightarrow \gamma_{i+1} \Rightarrow \cdots$$

  - terminating: $\gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_n$, $\gamma_n \in \mathrm{T}$
  - blocking: $\gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_n$, $\gamma_n \notin \mathrm{T}$ and $\gamma_n \not\Rightarrow$
  - infinite (looping): $\gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots$

- $\gamma \Rightarrow^k \gamma'$ for $k \geq 0$, if there is a computation $\gamma = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_k = \gamma'$
- $\gamma \Rightarrow^* \gamma'$ if $\gamma \Rightarrow^k \gamma'$ for some $k \geq 0$

TINY: operational semantics

Configurations: $\Gamma = (\mathbf{Stmt} \times \mathbf{State}) \cup \mathbf{State}$

Terminal configurations: $\mathrm{T} = \mathbf{State}$

Transition relation contains only:

$$\langle x := e, s\rangle \Rightarrow s[x \mapsto (\mathcal{E}[\![e]\!]\, s)]$$

$$\langle \mathbf{skip}, s\rangle \Rightarrow s$$

$$\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s'\rangle \quad \text{if } \langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle$$

$$\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle \quad \text{if } \langle S_1, s\rangle \Rightarrow s'$$

$$\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \Rightarrow \langle S_1, s\rangle \quad \text{if } \mathcal{B}[\![b]\!]\, s = \mathbf{tt}$$

$$\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \Rightarrow \langle S_2, s\rangle \quad \text{if } \mathcal{B}[\![b]\!]\, s = \mathbf{ff}$$

$$\langle \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \Rightarrow \langle S; \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \quad \text{if } \mathcal{B}[\![b]\!]\, s = \mathbf{tt}$$

$$\langle \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \Rightarrow s \quad \text{if } \mathcal{B}[\![b]\!]\, s = \mathbf{ff}$$

Fact: TINY *is deterministic, i.e.: for each configuration* $\langle S, s\rangle$ *if* $\langle S, s\rangle \Rightarrow \gamma_1$ *and* $\langle S, s\rangle \Rightarrow \gamma_2$ *then* $\gamma_1 = \gamma_2$.

Some variants

- instead of the current rule for $:=$:

$$\langle x := e, s\rangle \Rightarrow \langle \mathbf{skip}, s[x \mapsto (\mathcal{E}[\![e]\!]\, s)]\rangle$$

- instead of the current rules for **if**:

$$\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \Rightarrow \langle S_1', s'\rangle \quad \text{if } \mathcal{B}[\![b]\!]\, s = \mathbf{tt} \text{ and } \langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle$$
$$\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \Rightarrow s' \quad \text{if } \mathcal{B}[\![b]\!]\, s = \mathbf{tt} \text{ and } \langle S_1, s\rangle \Rightarrow s'$$
$$\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \Rightarrow \langle S_2', s'\rangle \quad \text{if } \mathcal{B}[\![b]\!]\, s = \mathbf{ff} \text{ and } \langle S_2, s\rangle \Rightarrow \langle S_2', s'\rangle$$
$$\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \Rightarrow s' \quad \text{if } \mathcal{B}[\![b]\!]\, s = \mathbf{ff} \text{ and } \langle S_2, s\rangle \Rightarrow s'$$

Non-deterministic computation

Sometimes non-determinism arises naturally. For example, in an operational semantics for arithmetic expressions, we would likely have:

$$\langle e_1 + e_2, s\rangle \Rightarrow \langle e_1' + e_2, s\rangle \quad \text{if } \langle e_1, s\rangle \Rightarrow \langle e_1', s\rangle$$
$$\langle e_1 + e_2, s\rangle \Rightarrow \langle e_1 + e_2', s\rangle \quad \text{if } \langle e_2, s\rangle \Rightarrow \langle e_2', s\rangle$$
$$\langle N_1 + N_2, s\rangle \Rightarrow \langle M, s\rangle \quad \text{if } N_1 + N_2 = M$$

Fact: *Weak confluence does not imply confluence:*

Fact: **(Newman's Lemma)** *If* $\Rightarrow$ *is strongly normalizing (i.e., it has no infinite paths) and weakly confluent then it is confluent.*

Confluence

Consider any set Γ and a relation $\Rightarrow \subseteq \Gamma \times \Gamma$.

Definition: $\Rightarrow$ *is weakly confluent if for every* $\gamma, \gamma_1, \gamma_2 \in \Gamma$ *such that*

$$\gamma \Rightarrow \gamma_1 \qquad \gamma \Rightarrow \gamma_2$$

there exists $\gamma_3 \in \Gamma$ *such that*

$$\gamma_1 \Rightarrow^* \gamma_3 \qquad \gamma_2 \Rightarrow^* \gamma_3.$$

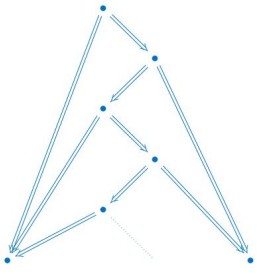
$\Rightarrow$ *is confluent if for every* $\gamma, \gamma_1, \gamma_2 \in \Gamma$ *such that*

$$\gamma \Rightarrow^* \gamma_1 \qquad \gamma \Rightarrow^* \gamma_2$$

there exists $\gamma_3 \in \Gamma$ *such that*

$$\gamma_1 \Rightarrow^* \gamma_3 \qquad \gamma_2 \Rightarrow^* \gamma_3.$$

Confluence is sometimes called *Church-Rosser property*.



Semantyka operacyjna dużych kroków (naturalna)

TINY: natural semantics

$$\langle x := e, s\rangle \rightsquigarrow s[x \mapsto (\mathcal{E}[\![e]\!]\ s)]$$

$$\langle \mathbf{skip}, s\rangle \rightsquigarrow s$$

$$\frac{\langle S_1, s\rangle \rightsquigarrow s' \qquad \langle S_2, s'\rangle \rightsquigarrow s''}{\langle S_1; S_2, s\rangle \rightsquigarrow s''}$$

$$\frac{\langle S_1, s\rangle \rightsquigarrow s'}{\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \rightsquigarrow s'} \text{ if } \mathcal{B}[\![b]\!]\ s = \mathbf{tt}$$

$$\frac{\langle S_2, s\rangle \rightsquigarrow s'}{\langle \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s\rangle \rightsquigarrow s'} \text{ if } \mathcal{B}[\![b]\!]\ s = \mathbf{ff}$$

$$\frac{\langle S, s\rangle \rightsquigarrow s' \qquad \langle \mathbf{while}\ b\ \mathbf{do}\ S, s'\rangle \rightsquigarrow s''}{\langle \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \rightsquigarrow s''} \text{ if } \mathcal{B}[\![b]\!]\ s = \mathbf{tt}$$

$$\langle \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \rightsquigarrow s \quad \text{if } \mathcal{B}[\![b]\!]\ s = \mathbf{ff}$$

Configurations:
$\Gamma = (\mathbf{Stmt} \times \mathbf{State}) \cup \mathbf{State}$
Terminal configurations: as before
$\mathrm{T} = \mathbf{State}$
Transitions: as given here

Overall idea:

- define *configurations*: $\gamma \in \Gamma$
- indicate which of them are *terminal*: $\mathrm{T} \subseteq \Gamma$
- instead of computations, consider (define) *transitions* directly *to final configurations* that are reached by computations: $\rightsquigarrow \subseteq \Gamma \times \mathrm{T}$

We give

– axioms, like $\langle x := e, s\rangle \rightsquigarrow s[x \mapsto (\mathcal{E}[\![e]\!]\ s)]$, and

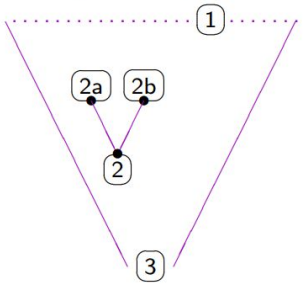
– rules, like $\dfrac{\langle S_1, s\rangle \rightsquigarrow s' \qquad \langle S_2, s'\rangle \rightsquigarrow s''}{\langle S_1; S_2, s\rangle \rightsquigarrow s''}$

to *derive* (or better: *prove*) *judgements* of the form

$$\langle S, s\rangle \rightsquigarrow s'$$

Actually: we give axiom and rule *schemata*, which are *generic* in the choice of elements to be substituted for meta-variables used ($x \in \mathbf{Var}$, $e \in \mathbf{Exp}$, $s, s', s'' \in \mathbf{State}$, $S_1, S_2 \in \mathbf{Stmt}$, etc).

Finite *proof tree* (or *derivation tree*):



- leaves: labelled by axioms, e.g.
  (1) : $\langle x := e, s\rangle \rightsquigarrow s[x \mapsto (\mathcal{E}[\![e]\!]\ s)]$
- other nodes: labelled according to the rules, e.g.
  $\dfrac{(2a) : \langle S_1, s\rangle \rightsquigarrow s' \qquad (2b) : \langle S_2, s'\rangle \rightsquigarrow s''}{(2) : \langle S_1; S_2, s\rangle \rightsquigarrow s''}$
- root: judgement proved, e.g. (3) : $\langle S, s\rangle \rightsquigarrow s'$

Induction on the structure of derivation trees

To prove if $\vdash \langle S, s\rangle \rightsquigarrow s'$ then $P(S, s, s')$ show:

– $P(x := e, s, s[x \mapsto (\mathcal{E}[\![e]\!]\ s)])$

– $P(\mathbf{skip}, s, s)$

– $P(S_1; S_2, s, s'')$ follows from $P(S_1, s, s')$ and $P(S_2, s', s'')$

– $P(\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s, s')$ follows from $P(S_1, s, s')$ whenever $\mathcal{B}[\![b]\!]\ s = \mathbf{tt}$

– $P(\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2, s, s')$ follows from $P(S_2, s, s')$ whenever $\mathcal{B}[\![b]\!]\ s = \mathbf{ff}$

– $P(\mathbf{while}\ b\ \mathbf{do}\ S, s, s'')$ follows from $P(S, s, s')$ and $P(\mathbf{while}\ b\ \mathbf{do}\ S, s', s'')$ whenever $\mathcal{B}[\![b]\!]\ s = \mathbf{tt}$

– $P(\mathbf{while}\ b\ \mathbf{do}\ S, s, s)$ whenever $\mathcal{B}[\![b]\!]\ s = \mathbf{ff}$

Fact: TINY *is deterministic, i.e.:*

for each $\vdash \langle S, s\rangle \rightsquigarrow s'$, *if* $\vdash \langle S, s\rangle \rightsquigarrow s''$ *then* $s' = s''$.

Statements $S_1, S_2 \in \mathbf{Stmt}$ are *naturally equivalent* (equivalent w.r.t. the natural semantics)

$$S_1 \equiv_{\mathcal{NS}} S_2$$

if for all states $s, s' \in \mathbf{State}$,

$$\vdash \langle S_1, s\rangle \rightsquigarrow s' \text{ iff } \vdash \langle S_2, s\rangle \rightsquigarrow s'$$

Fact: *The semantic equivalence is preserved by the linguistic constructs:*

- *if* $S_1 \equiv_{\mathcal{NS}} S_1'$ *and* $S_2 \equiv_{\mathcal{NS}} S_2'$ *then*
  $S_1; S_2 \equiv_{\mathcal{NS}} S_1'; S_2'$

Semantyka denotacyjna

- define syntax (*syntactic domains*)
- define *semantic domains*
- define *semantic functions*
- use *compositional* definitions

Semantic functions are defined *compositionally*, so that the denotation of a phrase depends only on the denotations of its immediate components:

$$\mathcal{C}[\![\varphi(ph_1,\ldots,ph_n)]\!] = \Phi(\mathcal{C}[\![ph_1]\!],\ldots,\mathcal{C}[\![ph_n]\!])$$

Such a *semantic clause* is given for each syntactic construct.

Syntax | Semantics

$N \in \mathbf{Num} ::= 0 \mid 1 \mid 2 \mid \cdots$

$(x \in \mathbf{Var} ::= \cdots)$

$e \in \mathbf{Exp} ::= N \mid x \mid e_1 + e_2 \mid e_1 * e_2 \mid e_1 - e_2$

$b \in \mathbf{BExp} ::= \mathbf{true} \mid \mathbf{false} \mid e_1 \leq e_2 \mid \neg b' \mid b_1 \wedge b_2$

$S \in \mathbf{Stmt} ::= x := e \mid \mathbf{skip} \mid S_1; S_2 \mid \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2 \mid \mathbf{while}\ b\ \mathbf{do}\ S'$

$$\mathbf{Int} = \{0, 1, -1, 2, -2, \ldots\}$$
$$\mathbf{Bool} = \{\mathbf{tt}, \mathbf{ff}\}$$
$$\mathbf{State} = \mathbf{Var} \to \mathbf{Int}$$
$$\mathbf{EXP} = \mathbf{State} \to \mathbf{Int}$$
$$\mathbf{BEXP} = \mathbf{State} \to \mathbf{Bool}$$
$$\mathbf{STMT} = \mathbf{State} \rightharpoonup \mathbf{State}$$

Some auxiliary notation:

- *λ-notation*: $\lambda x{:}D.E$ stands for the function that maps any $d \in D$ to $E[d/x]$
- *identity*: $id_D = \lambda x{:}D.x$
- *function composition*: the composition of $f\colon D_1 \to D_2$ and $g\colon D_2 \to D_3$ is written as $f;g\colon D_1 \to D_3$
- *conditional*: $ifte_D\colon \mathbf{Bool} \times D \times D \to D$ is defined by

$$ifte_D(c, d_1, d_2) = \begin{cases} d_1 & \text{if } c = \mathbf{tt} \\ d_2 & \text{if } c = \mathbf{ff} \end{cases}$$

(the index D will often be omitted)

Semantic functions:

$\mathcal{N}\colon \mathbf{Num} \to \mathbf{Int}$
$\mathcal{E}\colon \mathbf{Exp} \to \mathbf{EXP}$
$\mathcal{B}\colon \mathbf{BExp} \to \mathbf{BEXP}$
$\mathcal{S}\colon \mathbf{Stmt} \to \mathbf{STMT}$

- *indexing*: given any function $f\colon D_1 \times \cdots \times D_n \to D$, for any domain I,

$$lift^I(f)\colon (I \to D_1) \times \cdots \times (I \to D_n) \to (I \to D)$$

is defined as follows:

$$lift^I(f)(fd_1,\ldots,fd_n) = \lambda i{:}I.f(fd_1(i),\ldots,fd_n(i))$$

$\mathcal{N}\colon \mathbf{Num} \to \mathbf{Int}$

$\mathcal{N}[\![0]\!] = 0$
$\mathcal{N}[\![1]\!] = 1$
$\mathcal{N}[\![2]\!] = 2$
$\ldots$

$\mathcal{E}\colon \mathbf{Exp} \to \mathbf{EXP}$

$\mathcal{E}[\![N]\!]s = \mathcal{N}[\![N]\!] \qquad \mathcal{E}[\![x]\!]s = s\,x$
$\mathcal{E}[\![e_1 + e_2]\!]s = \mathcal{E}[\![e_1]\!]s + \mathcal{E}[\![e_2]\!]s$
$\mathcal{E}[\![e_1 * e_2]\!]s = \mathcal{E}[\![e_1]\!]s * \mathcal{E}[\![e_2]\!]s$
$\mathcal{E}[\![e_1 - e_2]\!]s = \mathcal{E}[\![e_1]\!]s - \mathcal{E}[\![e_2]\!]s$

$\mathcal{B}\colon \mathbf{BExp} \to \mathbf{BEXP}$

$\mathcal{B}[\![\mathbf{true}]\!]s = \mathbf{tt} \qquad \mathcal{B}[\![\mathbf{false}]\!]s = \mathbf{ff} \qquad \mathcal{B}[\![\neg b]\!]s = \neg(\mathcal{B}[\![b]\!]s)$
$\mathcal{B}[\![e_1 \leq e_2]\!]s = (\mathcal{E}[\![e_1]\!]s \leq \mathcal{E}[\![e_2]\!]s) \qquad \mathcal{B}[\![b_1 \wedge b_2]\!]s = (\mathcal{B}[\![b_1]\!]s \wedge \mathcal{B}[\![b_2]\!]s)$

$$\begin{aligned}
\mathcal{S}[\![x := e]\!]s &= s[x \mapsto \mathcal{E}[\![e]\!]\ s] \\
\mathcal{S}[\![\mathbf{skip}]\!]s &= s \\
\mathcal{S}[\![S_1; S_2]\!]s &= \mathcal{S}[\![S_2]\!](\mathcal{S}[\![S_1]\!]s) \\
\mathcal{S}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2]\!]s &= ifte_{\mathbf{State}}(\mathcal{B}[\![b]\!]s, \mathcal{S}[\![S_1]\!]s, \mathcal{S}[\![S_2]\!]s) \\
\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]s &= ifte_{\mathbf{State}}(\mathcal{B}[\![b]\!]s, \mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]s', s) \\
&\quad \text{where } s' = \mathcal{S}[\![S]\!]s
\end{aligned}$$

$$\begin{aligned}
\mathcal{S}[\![x := e]\!] &= \lambda s{:}\mathbf{State}.s[x \mapsto \mathcal{E}[\![e]\!]\ s] \\
\mathcal{S}[\![\mathbf{skip}]\!] &= id_{\mathbf{State}} \\
\mathcal{S}[\![S_1; S_2]\!] &= \mathcal{S}[\![S_1]\!];\mathcal{S}[\![S_2]\!] \\
\mathcal{S}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2]\!] &= cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S_1]\!], \mathcal{S}[\![S_2]\!]) \\
\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!] &= cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S]\!];\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!], id_{\mathbf{State}})
\end{aligned}$$

$\Phi\colon \mathbf{STMT} \to \mathbf{STMT}$ is defined as follows:

$$\Phi(F) = cond(\mathcal{B}[\![b]\!], \mathcal{S}[\![S]\!];F, id_{\mathbf{State}})$$

$\langle \mathbf{while}\ b\ \mathbf{do}\ S, s\rangle \Rightarrow^* s'$ iff $fix(\Phi)\ s = s'$

$$\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!] = fix(\Phi) = \bigcup_{n\geq 0} \Phi^n(\emptyset_{\mathbf{State}\rightharpoonup\mathbf{State}})$$

State i Env

$$\mathcal{N}: \mathbf{Num} \to \mathbf{Int}$$

$$\mathcal{E}: \mathbf{Exp} \to \underbrace{\mathbf{VEnv} \to \mathbf{Store} \to (\mathbf{Int} + \{??\})}_{\mathbf{EXP}}$$

$$\mathbf{VEnv} = \mathbf{Var} \to (\mathbf{Loc} + \{??\})$$

$$\mathbf{Store} = \mathbf{Loc} \to (\mathbf{Int} + \{??\})$$

$$\mathcal{B}: \mathbf{BExp} \to \underbrace{\mathbf{VEnv} \to \mathbf{Store} \to (\mathbf{Bool} + \{??\})}_{\mathbf{BEXP}}$$

$$\mathcal{E}[\![x]\!]\ \rho_V\ s = s\ l \text{ where } l = \rho_V\ x$$

$$\mathcal{E}[\![e_1 + e_2]\!]\ \rho_V\ s = n_1 + n_2 \text{ where } n_1 = \mathcal{E}[\![e_1]\!]\ \rho_V\ s, n_2 = \mathcal{E}[\![e_2]\!]\ \rho_V\ s$$

Statements do not modify the environment!

$$\mathcal{S}: \mathbf{Stmt} \to \underbrace{\mathbf{VEnv} \to \mathbf{Store} \to (\mathbf{Store} + \{??\})}_{\mathbf{STMT}}$$

$$\mathcal{S}[\![x := e]\!]\ \rho_V\ s = s[l \mapsto n] \text{ where } l = \rho_V\ x, n = \mathcal{E}[\![e]\!]\ \rho_V\ s$$

$$\mathcal{S}[\![\mathbf{skip}]\!]\ \rho_V = id_{\mathbf{Store}}$$

$$\mathcal{S}[\![S_1; S_2]\!]\ \rho_V = \mathcal{S}[\![S_1]\!]\ \rho_V ; \mathcal{S}[\![S_2]\!]\ \rho_V$$

$$\mathcal{S}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2]\!]\ \rho_V = cond(\mathcal{B}[\![b]\!]\ \rho_V, \mathcal{S}[\![S_1]\!]\ \rho_V, \mathcal{S}[\![S_2]\!]\ \rho_V)$$

$$\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]\ \rho_V = cond(\mathcal{B}[\![b]\!]\ \rho_V, \mathcal{S}[\![S]\!]\ \rho_V ; \mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]\ \rho_V, id_{\mathbf{Store}})$$

$$\mathcal{D}_V: \mathbf{VDecl} \to \underbrace{\mathbf{VEnv} \to \mathbf{Store} \to (\mathbf{VEnv} \times \mathbf{Store} + \{??\})}_{\mathbf{VDECL}}$$

$$\mathcal{D}_V[\![\varepsilon]\!]\ \rho_V\ s = \langle \rho_V, s \rangle$$

$$\mathcal{D}_V[\![\mathbf{var}\ x; D_V]\!]\ \rho_V\ s = \mathcal{D}_V[\![D_V]\!]\ \rho'_V\ s' \text{ where } l = newloc(s), \rho'_V = \rho_V[x \mapsto l], s' = s[l \mapsto ??]$$

$$\mathcal{S}[\![\mathbf{begin}\ D_V\ S\ \mathbf{end}]\!]\ \rho_V\ s = \mathcal{S}[\![S]\!]\ \rho'_V\ s' \text{ where } \langle \rho'_V, s' \rangle = \mathcal{D}_V[\![D_V]\!]\ \rho_V\ s$$

Wyjątki (na kontynuacjach)

$$S \in \mathbf{Stmt} ::= \ldots \mid \mathbf{try}\ S_1\ \mathbf{catch}(\chi)\ S_2 \mid \mathbf{throw}\ \chi$$

$$\chi \in \mathbf{EXN} ::= \ldots$$

$$\mathbf{XEnv} = \mathbf{EXN} \to (\mathbf{Cont} + \{??\})$$

$$\mathcal{S}: \mathbf{Stmt} \to \underbrace{\mathbf{VEnv} \to \mathbf{PEnv} \to \mathbf{XEnv} \to \mathbf{Cont} \to \mathbf{Cont}}_{\mathbf{STMT}}$$

$$\mathcal{D}_P: \mathbf{PDecl} \to \underbrace{\mathbf{VEnv} \to \mathbf{PEnv} \to \mathbf{XEnv} \to \mathbf{Cont}_{\mathbf{Dp}} \to \mathbf{Cont}}_{\mathbf{PDECL}}$$

$$\mathcal{S}[\![\mathbf{try}\ S_1\ \mathbf{catch}(\chi)\ S_2]\!]\ \rho_V\ \rho_P\ \rho_X\ \kappa = \mathcal{S}[\![S_1]\!]\ \rho_V\ \rho_P\ \rho_X[\chi \mapsto \kappa']\ \kappa \text{ where } \kappa' = \mathcal{S}[\![S_2]\!]\ \rho_V\ \rho_P\ \rho_X\ \kappa$$

$$\mathcal{S}[\![\mathbf{throw}\ \chi]\!]\ \rho_V\ \rho_P\ \rho_X\ \kappa = \rho_X\ \chi$$

Kontynuacje c.d. (z innej strony)

$$\mathcal{P}[\![\mathbf{prog}\ S]\!]\ i = \mathcal{S}[\![S]\!]\ \rho_V^{\emptyset}\ \rho_P^{\emptyset}\ \kappa^{\emptyset}\ \langle s^{\emptyset}, i \rangle \text{ where } \rho_V^{\emptyset}\ x = ??, \rho_P^{\emptyset}\ p = ??, \kappa^{\emptyset}\ s = \mathbf{eof}, s^{\emptyset}\ next = 0, s^{\emptyset}\ l = ??$$

$$\mathcal{D}_P[\![\varepsilon]\!]\ \rho_V\ \rho_P\ \kappa_P = \kappa_P\ \rho_P$$

$$\mathcal{D}_P[\![\mathbf{proc}\ p\ \mathbf{is}\ (S); D_P]\!]\ \rho_V\ \rho_P = \mathcal{D}_P[\![D_P]\!]\ \rho_V\ \rho_P[p \mapsto P] \text{ where } P = \mathcal{S}[\![S]\!]\ \rho_V\ \rho_P[p \mapsto P]$$

$$\mathcal{D}_V[\![\mathbf{var}\ x; D_V]\!]\ \rho_V\ \kappa_V\ \langle s, i \rangle = \mathcal{D}_V[\![D_V]\!]\ \rho'_V\ \kappa_V\ \langle s', i \rangle \text{ where } l = s\ next, \rho'_V = \rho_V[x \mapsto l], s' = s[l \mapsto ??, next \mapsto l + 1]$$

$$\mathcal{D}_V[\![\mathbf{var}\ x = e; D_V]\!]\ \rho_V\ \kappa_V\ \langle s, i \rangle = \mathcal{E}[\![e]\!]\ \rho_V\ (\lambda n{:}\mathbf{Int}.\mathcal{D}_V[\![D_V]\!]\ \rho'_V\ \kappa_V\ \langle s', i \rangle) \text{ where } l = s\ next, \rho'_V = \rho_V[x \mapsto l], s' = s[l \mapsto n, next \mapsto l + 1]$$

$$\mathcal{S}[\![\mathbf{begin}\ D_V\ D_P\ S\ \mathbf{end}]\!]\ \rho_V\ \rho_P\ \kappa = \mathcal{D}_V[\![D_V]\!]\ \rho_V\ \lambda\rho'_V{:}\mathbf{VEnv}.\mathcal{D}_P[\![D_P]\!]\ \rho'_V\ \rho_P\ \lambda\rho'_P{:}\mathbf{PEnv}.\mathcal{S}[\![S]\!]\ \rho'_V\ \rho'_P\ \kappa$$

Procedury rekurencyjne

$$\mathcal{D}^{\mathrm{P}}_{\mathrm{ds}}[\![\texttt{proc}\ p\ \texttt{is}\ S;\ D_P]\!] env_V\ env_P = \mathcal{D}^{\mathrm{P}}_{\mathrm{ds}}[\![D_P]\!] env_V\ (env_P[p \mapsto \mathrm{FIX}\ F])$$

where $F\ g = \mathcal{S}_{\mathrm{ds}}[\![S]\!] env_V\ (env_P[p \mapsto g])$

$$\mathcal{D}^{\mathrm{P}}_{\mathrm{ds}}[\![\varepsilon]\!] env_V = \mathrm{id}$$

Procedury, static binding

$$\mathcal{S}: \mathbf{Stmt} \to \underbrace{\mathbf{VEnv} \to \mathbf{PEnv} \to \mathbf{Store} \to (\mathbf{Store} + \{??\})}_{\mathbf{STMT}}$$

$$\mathbf{PEnv} = \mathbf{IDE} \to (\mathbf{PROC}_0 + \{??\})$$

$$\mathbf{PROC}_0 = \mathbf{Store} \to (\mathbf{Store} + \{??\})$$

$$\mathcal{D}_P: \mathbf{PDecl} \to \underbrace{\mathbf{VEnv} \to \mathbf{PEnv} \to (\mathbf{PEnv} + \{??\})}_{\mathbf{PDECL}}$$

$$\mathcal{S}[\![\mathbf{call}\ p]\!]\ \rho_V\ \rho_P = P \text{ where } P = \rho_P\ p$$

$$\mathcal{D}_P[\![\mathbf{proc}\ p\ \mathbf{is}\ (S); D_P]\!]\ \rho_V\ \rho_P = \mathcal{D}_P[\![D_P]\!]\ \rho_V\ \rho_P[p \mapsto P] \text{ where } P = \mathcal{S}[\![S]\!]\ \rho_V\ \rho_P[p \mapsto P]$$

All other stay the same, like:

$$\mathcal{S}[\![x := e]\!]\ \rho_V\ \rho_P\ s = s[l \mapsto n] \text{ where } l = \rho_V\ x, n = \mathcal{E}[\![e]\!]\ \rho_V\ s$$

$$\mathcal{S}[\![\mathbf{skip}]\!]\ \rho_V\ \rho_P = id_{\mathbf{Store}}$$

$$\mathcal{S}[\![S_1; S_2]\!]\ \rho_V\ \rho_P = \mathcal{S}[\![S_1]\!]\ \rho_V\ \rho_P; \mathcal{S}[\![S_2]\!]\ \rho_V\ \rho_P$$

$$\mathcal{S}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2]\!]\ \rho_V\ \rho_P = cond(\mathcal{B}[\![b]\!]\ \rho_V, \mathcal{S}[\![S_1]\!]\ \rho_V\ \rho_P, \mathcal{S}[\![S_2]\!]\ \rho_V\ \rho_P)$$

Parameter passing:

- call by value
- call by variable
- call by name

$$\begin{aligned} S \in \mathbf{Stmt} &::= \ldots \mid \mathbf{begin}\ D_V\ D_P\ S\ \mathbf{end} \\ &\mid \mathbf{call}\ p \mid \mathbf{call}\ p(\mathbf{vr}\ x) \mid \mathbf{call}\ p(\mathbf{vl}\ e) \mid \mathbf{call}\ p(\mathbf{nm}\ e) \\ D_V \in \mathbf{VDecl} &::= \mathbf{var}\ x; D_V \mid \varepsilon \\ D_P \in \mathbf{PDecl} &::= \mathbf{proc}\ p\ \mathbf{is}\ (S); D_P \mid \mathbf{proc}\ p(\mathbf{vr}\ x)\ \mathbf{is}\ (S); D_P \\ &\mid \mathbf{proc}\ p(\mathbf{vl}\ x)\ \mathbf{is}\ (S); D_P \mid \mathbf{proc}\ p(\mathbf{nm}\ x)\ \mathbf{is}\ (S); D_P \mid \varepsilon \end{aligned}$$

$$\mathbf{PEnv} = \mathbf{IDE} \to (\mathbf{PROC}_0 + \mathbf{PROC}_1^{\mathsf{vr}} + \mathbf{PROC}_1^{\mathsf{vl}} + \mathbf{PROC}_1^{\mathsf{nm}} + \{??\})$$

$$\mathbf{PROC}_0 = \mathbf{Store} \to (\mathbf{Store} + \{??\})$$

$$\mathbf{PROC}_1^{\mathsf{vr}} = \mathbf{Loc} \to \mathbf{PROC}_0$$

$$\mathbf{PROC}_1^{\mathsf{vl}} = \mathbf{Int} \to \mathbf{PROC}_0$$

$$\mathbf{PROC}_1^{\mathsf{nm}} = (\mathbf{Store} \to (\mathbf{Int} + \{??\})) \to \mathbf{PROC}_0$$

$$\mathcal{S}[\![\mathbf{call}\ p]\!]\ \rho_V\ \rho_P = P \text{ where } P = \rho_P\ p \in \mathbf{PROC}_0$$

$$\mathcal{D}_P[\![\mathbf{proc}\ p\ \mathbf{is}\ (S); D_P]\!]\ \rho_V\ \rho_P = \mathcal{D}_P[\![D_P]\!]\ \rho_V\ \rho_P[p \mapsto P] \text{ where } P = \mathcal{S}[\![S]\!]\ \rho_V\ \rho_P[p \mapsto P]$$

$$\mathcal{S}[\![\mathbf{call}\ p(\mathbf{vr}\ y)]\!]\ \rho_V\ \rho_P = P\ l \text{ where } P = \rho_P\ p \in \mathbf{PROC}_1^{\mathsf{vr}},\ l = \rho_V\ y$$

$$\mathcal{D}_P[\![\mathbf{proc}\ p(\mathbf{vr}\ x)\ \mathbf{is}\ (S); D_P]\!]\ \rho_V\ \rho_P = \mathcal{D}_P[\![D_P]\!]\ \rho_V\ \rho_P[p \mapsto P] \text{ where } P\ l = \mathcal{S}[\![S]\!]\ \rho_V[x \mapsto l]\ \rho_P[p \mapsto P]$$

$$\mathcal{S}[\![\mathbf{call}\ p(\mathbf{vl}\ e)]\!]\ \rho_V\ \rho_P\ s = P\ n\ s \text{ where } P = \rho_P\ p \in \mathbf{PROC}_1^{\mathsf{vl}},\ n = \mathcal{E}[\![e]\!]\ \rho_V\ s$$

$$\mathcal{D}_P[\![\mathbf{proc}\ p(\mathbf{vl}\ x)\ \mathbf{is}\ (S); D_P]\!]\ \rho_V\ \rho_P = \mathcal{D}_P[\![D_P]\!]\ \rho_V\ \rho_P[p \mapsto P] \text{ where } P\ n\ s = \mathcal{S}[\![S]\!]\ \rho'_V\ \rho_P[p \mapsto P]\ s' \text{ where } l = s\ next,\ \rho'_V = \rho_V[x \mapsto l],\ s' = s[l \mapsto n, next \mapsto l + 1]$$

Call by name requires some corrections $\rho_V[x \mapsto E] \notin \mathbf{VEnv}$

$$\mathcal{S}[\![\mathbf{call}\ p(\mathbf{nm}\ e)]\!]\ \rho_V\ \rho_P = P\ (\mathcal{E}[\![e]\!]\ \rho_V) \text{ where } P = \rho_P\ p \in \mathbf{PROC}_1^{\mathsf{nm}}$$

$$\mathcal{D}_P[\![\mathbf{proc}\ p(\mathbf{nm}\ x)\ \mathbf{is}\ (S); D_P]\!]\ \rho_V\ \rho_P = \mathcal{D}_P[\![D_P]\!]\ \rho_V\ \rho_P[p \mapsto P] \text{ where } P\ E = \mathcal{S}[\![S]\!]\ \rho_V[x \mapsto E]\ \rho_P[p \mapsto P]$$

$$\mathbf{VEnv} = \mathbf{Var} \to (\mathbf{Loc} + (\mathbf{Store} \to (\mathbf{Int} + \{??\})) + \{??\})$$

$$\mathcal{E}[\![x]\!]\ \rho_V\ s = \text{let } v = \rho_V\ x \text{ in if } v \in \mathbf{Loc} \text{ then } s\ v \text{ if } v \in (\mathbf{Store} \to (\mathbf{Int} + \{??\})) \text{ then } v\ s$$

Input / output

$$\mathbf{Stream} = \mathbf{Int} \times \mathbf{Stream} + \{\mathbf{eof}\}$$
$$\mathbf{Input} = \mathbf{Stream}$$
$$\mathbf{Output} = \mathbf{Stream}$$
$$\mathbf{State} = \mathbf{Store} \times \mathbf{Input} \times \mathbf{Output}$$

$$\mathcal{E}[\![x]\!]\ \rho_V\ \langle s, i, o\rangle = s\ l \text{ where } l = \rho_V\ x$$

$$\mathcal{S}\colon \mathbf{Stmt} \to \underbrace{\mathbf{VEnv} \to \mathbf{PEnv} \to \mathbf{State} \to (\mathbf{State} + \{??\})}_{\text{STMT}}$$

$$\mathcal{S}[\![x := e]\!]\ \rho_V\ \rho_P\ \langle s, i, o\rangle = \langle s[l \mapsto n], i, o\rangle \text{ where } l = \rho_V\ x, n = \mathcal{E}[\![e]\!]\ \rho_V\ \langle s, i, o\rangle$$

$$\mathcal{S}[\![\mathbf{read}\ x]\!]\ \rho_V\ \rho_P\ \langle s, i, o\rangle = \langle s[l \mapsto n], i', o\rangle \text{ where } l = \rho_V\ x, \langle n, i'\rangle = i$$

$$\mathcal{S}[\![\mathbf{write}\ e]\!]\ \rho_V\ \rho_P\ \langle s, i, o\rangle = \langle s, i, \langle n, o\rangle\rangle \text{ where } n = \mathcal{E}[\![e]\!]\ \rho_V\ \langle s, i, o\rangle$$

$$\mathbf{Prog} ::= \mathbf{prog}\ S \qquad \mathcal{P}\colon \mathbf{Prog} \to \underbrace{\mathbf{Input} \to (\mathbf{Output} + \{??\})}_{\text{PROG}}$$

$$\mathcal{P}[\![\mathbf{prog}\ S]\!]\ i = o' \text{ where } \mathcal{S}[\![S]\!]\ \rho_V^{\emptyset}\ \rho_P^{\emptyset}\ \langle s^{\emptyset}, i, \mathbf{eof}\rangle = \langle s', i', o'\rangle,$$
$$\rho_V^{\emptyset}\ x = ??, \rho_P^{\emptyset}\ p = ??, s^{\emptyset}\ next = 0, s^{\emptyset}\ l = ??$$

Kontynuacje

$$\mathbf{Cont} = \mathbf{State} \to \mathbf{Ans}$$

$$\mathbf{Cont} = \mathbf{State} \to \mathbf{Output}$$
$$\mathbf{Cont_E} = \mathbf{Int} \to \mathbf{Cont}$$
$$\mathbf{Cont_B} = \mathbf{Bool} \to \mathbf{Cont}$$
$$\mathbf{Cont_{D_V}} = \mathbf{VEnv} \to \mathbf{Cont}$$
$$\mathbf{Cont_{D_P}} = \mathbf{PEnv} \to \mathbf{Cont}$$

$$\mathbf{Input} = \mathbf{Int} \times \mathbf{Input} + \{\mathbf{eof}\}$$
$$\mathbf{State} = \mathbf{Store} \times \mathbf{Input}$$
$$\mathbf{Output} = \mathbf{Int} \times \mathbf{Output} + \{\mathbf{eof}, ??\}$$

$$\mathbf{PROC_0} = \mathbf{Cont} \to \mathbf{Cont}$$
$$\mathbf{PROC_1^{vr}} = \mathbf{Loc} \to \mathbf{PROC_0}$$
$$\mathbf{PEnv} = \mathbf{IDE} \to (\mathbf{PROC_0} + \mathbf{PROC_1^{vr}} + \{??\})$$

$$\mathcal{E}\colon \mathbf{Exp} \to \underbrace{\mathbf{VEnv} \to \mathbf{Cont_E} \to \mathbf{Cont}}_{\text{EXP}}$$
$$\mathcal{B}\colon \mathbf{BExp} \to \underbrace{\mathbf{VEnv} \to \mathbf{Cont_B} \to \mathbf{Cont}}_{\text{BEXP}}$$
$$\mathcal{S}\colon \mathbf{Stmt} \to \underbrace{\mathbf{VEnv} \to \mathbf{PEnv} \to \mathbf{Cont} \to \mathbf{Cont}}_{\text{STMT}}$$
$$\mathcal{D_V}\colon \mathbf{VDecl} \to \underbrace{\mathbf{VEnv} \to \mathbf{Cont_{D_V}} \to \mathbf{Cont}}_{\text{VDECL}}$$
$$\mathcal{D_P}\colon \mathbf{PDecl} \to \underbrace{\mathbf{VEnv} \to \mathbf{PEnv} \to \mathbf{Cont_{D_P}} \to \mathbf{Cont}}_{\text{PDECL}}$$
$$\mathcal{P}\colon \mathbf{Prog} \to \underbrace{\mathbf{Input} \to \mathbf{Output}}_{\text{PROG}}$$

$$\mathcal{E}[\![x]\!]\ \rho_V\ \kappa_E = \lambda\langle s, i\rangle{:}\mathbf{State}.\kappa_E\ n\ \langle s, i\rangle \text{ where } l = \rho_V\ x, n = s\ l$$

this means: $??$ if $\rho_V\ x = ??$ or $s\ l = ??$

$$\mathcal{E}[\![e_1 + e_2]\!]\ \rho_V\ \kappa_E =$$
$$\mathcal{E}[\![e_1]\!]\ \rho_V\ \lambda n_1{:}\mathbf{Int}.\mathcal{E}[\![e_2]\!]\ \rho_V\ \lambda n_2{:}\mathbf{Int}.\kappa_E\ (n_1 + n_2)$$

$$\mathcal{S}[\![x := e]\!]\ \rho_V\ \rho_P\ \kappa = \mathcal{E}[\![e]\!]\ \rho_V\ (\lambda n{:}\mathbf{Int}.\lambda\langle s, i\rangle{:}\mathbf{State}.\kappa\ \langle s[l \mapsto n], i\rangle) \text{ where } l = \rho_V\ x$$

$$\mathcal{S}[\![\mathbf{skip}]\!]\ \rho_V\ \rho_P = id_{\mathbf{Cont}}$$

$$\mathcal{S}[\![S_1; S_2]\!]\ \rho_V\ \rho_P\ \kappa = \mathcal{S}[\![S_1]\!]\ \rho_V\ \rho_P\ (\mathcal{S}[\![S_2]\!]\ \rho_V\ \rho_P\ \kappa)$$

$$\mathcal{S}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2]\!]\ \rho_V\ \rho_P\ \kappa =$$
$$\mathcal{B}[\![b]\!]\ \rho_V\ \lambda v{:}\mathbf{Bool}.ifte(v, \mathcal{S}[\![S_1]\!]\ \rho_V\ \rho_P\ \kappa, \mathcal{S}[\![S_2]\!]\ \rho_V\ \rho_P\ \kappa)$$

$$\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]\ \rho_V\ \rho_P\ \kappa =$$
$$\mathcal{B}[\![b]\!]\ \rho_V\ \lambda v{:}\mathbf{Bool}.ifte(v, \mathcal{S}[\![S]\!]\ \rho_V\ \rho_P\ (\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]\ \rho_V\ \rho_P\ \kappa), \kappa)$$

$$\mathcal{S}[\![\mathbf{call}\ p]\!]\ \rho_V\ \rho_P = P \text{ where } P = \rho_P\ p$$

$$\mathcal{S}[\![\mathbf{call}\ p(\mathbf{vr}\ x)]\!]\ \rho_V\ \rho_P = P\ l \text{ where } P = \rho_P\ p \in \mathbf{PROC_1^{vr}},\ l = \rho_V\ x$$

$$\mathcal{S}[\![\mathbf{read}\ x]\!]\ \rho_V\ \rho_P\ \kappa\ \langle s, i\rangle = \kappa\ \langle s[l \mapsto n], i'\rangle \text{ where } l = \rho_V\ x, \langle n, i'\rangle = i$$

$$\mathcal{S}[\![\mathbf{write}\ e]\!]\ \rho_V\ \rho_P\ \kappa = \mathcal{E}[\![e]\!]\ \rho_V\ \lambda n{:}\mathbf{Int}.\lambda\langle s, i\rangle{:}\mathbf{State}.\langle n, \kappa\ \langle s, i\rangle\rangle$$

$$\mathcal{S}[\![\mathbf{begin}\ D_V\ D_P\ S\ \mathbf{end}]\!]\ \rho_V\ \rho_P\ \kappa =$$
$$\mathcal{D_V}[\![D_V]\!]\ \rho_V\ \lambda\rho_V'{:}\mathbf{VEnv}.\mathcal{D_P}[\![D_P]\!]\ \rho_V'\ \rho_P\ \lambda\rho_P'{:}\mathbf{PEnv}.\mathcal{S}[\![S]\!]\ \rho_V'\ \rho_P'\ \kappa$$

$$\mathcal{S}[\![\mathbf{abort}]\!]\ \rho_V\ \rho_P\ \kappa = id_{\mathbf{Store}}$$

$$\mathcal{B}[\![\mathbf{true}]\!]\ \rho_V\ \kappa_B = \kappa_B\ \mathbf{tt}$$

$$\mathcal{B}[\![e_1 \le e_2]\!]\ \rho_V\ \kappa_B =$$
$$\mathcal{E}[\![e_1]\!]\ \rho_V\ \lambda n_1{:}\mathbf{Int}.\mathcal{E}[\![e_2]\!]\ \rho_V\ \lambda n_2{:}\mathbf{Int}.\kappa_B\ ifte(n_1 \le n_2, \mathbf{tt}, \mathbf{ff})$$

Weryfikacja $\{\varphi\}\, S\, \{\psi\}$

Whenever the program S starts in a state satisfying the precondtion φ
and terminates successfully, then the final state satisfies the postcondition ψ

$$\varphi \in \mathbf{Form} ::= b \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \neg\varphi' \mid \exists x.\varphi' \mid \forall x.\varphi'$$

$\mathcal{F}\colon \mathbf{Form} \to \mathbf{State} \to \mathbf{Bool}$

$$\{\varphi\} = \{s \in \mathbf{State} \mid \mathcal{F}[\![\varphi]\!]\, s = \mathbf{tt}\}$$

$$A\,[\![S]\!] = \{s \in \mathbf{State} \mid \mathcal{S}[\![S]\!]\, a = s, \text{for some } a \in A\}$$

$\models \{\varphi\}\, S\, \{\psi\}$ iff $\{\varphi\}\,[\![S]\!] \subseteq \{\psi\}$

The partial correctness judgement $\{\varphi\}\, S\, \{\psi\}$ holds, written $\models \{\varphi\}\, S\, \{\psi\}$,
if for all states $s \in \mathbf{State}$

if $\mathcal{F}[\![\varphi]\!]\, s = \mathbf{tt}$ and $\mathcal{S}[\![S]\!]\, s \in \mathbf{State}$
then $\mathcal{F}[\![\psi]\!]\, (\mathcal{S}[\![S]\!]\, s) = \mathbf{tt}$

$$\frac{}{\{\varphi[x \mapsto e]\}\, x := e\, \{\varphi\}} \qquad \frac{}{\{\varphi\}\, \mathbf{skip}\, \{\varphi\}}$$

$$\frac{\{\varphi\}\, S_1\, \{\theta\} \quad \{\theta\}\, S_2\, \{\psi\}}{\{\varphi\}\, S_1; S_2\, \{\psi\}} \qquad \frac{\{\varphi \wedge b\}\, S_1\, \{\psi\} \quad \{\varphi \wedge \neg b\}\, S_2\, \{\psi\}}{\{\varphi\}\, \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\, \{\psi\}}$$

$$\frac{\{\varphi \wedge b\}\, S\, \{\varphi\}}{\{\varphi\}\, \mathbf{while}\ b\ \mathbf{do}\ S\, \{\varphi \wedge \neg b\}} \qquad \frac{\varphi' \Rightarrow \varphi \quad \{\varphi\}\, S\, \{\psi\} \quad \psi \Rightarrow \psi'}{\{\varphi'\}\, S\, \{\psi'\}}$$

$\{n \geq 0\}$
$rt := 0;$
$\{n \geq 0 \wedge rt = 0\}$
$sqr := 1;$
$\{n \geq 0 \wedge rt = 0 \wedge sqr = 1\}$
while $\{sqr = (rt+1)^2 \wedge rt^2 \leq n\}$ $sqr \leq n$ **do**
  $rt := rt + 1;$
  $\{sqr = rt^2 \wedge sqr \leq n\}$
  $sqr := sqr + 2 * rt + 1$
$\{rt^2 \leq n < (rt+1)^2\}$

Total correctness = partial correctness + successful termination $[\varphi]\, S\, [\psi]$

Whenever the program S starts in a state satisfying the precondition φ
then it terminates successfully in a final state that satisfies the postcondition ψ

The total correctness judgement $[\varphi]\, S\, [\psi]$ holds, written $\models [\varphi]\, S\, [\psi]$,
if for all states $s \in \mathbf{State}$

if $\mathcal{F}[\![\varphi]\!]\, s = \mathbf{tt}$ then $\mathcal{S}[\![S]\!]\, s \in \mathbf{State}$ and $\mathcal{F}[\![\psi]\!]\, (\mathcal{S}[\![S]\!]\, s) = \mathbf{tt}$

Reguły dowodzenia jw. poza while:

$$\frac{(nat(l) \wedge \varphi(l+1)) \Rightarrow b \qquad [nat(l) \wedge \varphi(l+1)]\, S\, [\varphi(l)] \qquad \varphi(0) \Rightarrow \neg b}{[\exists l.nat(l) \wedge \varphi(l)]\ \mathbf{while}\ b\ \mathbf{do}\ S\, [\varphi(0)]}$$

$\varphi(l)$ is a formula with a free variable l that does not occur in **while** b **do** S,
$nat(l)$ stands for $0 \leq l$, and
$\varphi(l+1)$ and $\varphi(0)$ result by substituting, respectively, $l+1$ and 0 for l in $\varphi(l)$.

To prove

$[\varphi]\ \mathbf{while}\ b\ \mathbf{do}\ S\, [\varphi \wedge \neg b]$

- show "partial correctness": $[\varphi \wedge b]\, S\, [\varphi]$
- show "termination": find a set W with a well-founded relation $\succ \subseteq W \times W$ and a function $w\colon \mathbf{State} \to W$ such that for all states $s \in \{\varphi \wedge b\}$,

$$w(s) \succ w(\mathcal{S}[\![S]\!]\, s)$$

BTW: $w\colon \mathbf{State} \rightharpoonup W$ may be partial as long as it is defined on $\{\varphi\}$.

A relation $\succ \subseteq W \times W$ is *well-founded* if there is no infinite chain

$$a_0 \succ a_1 \succ \ldots \succ a_i \succ a_{i+1} \succ \ldots$$

$\langle \mathbf{Nat}, > \rangle$