

Semantics and Verification of Software

Lecture 15: Extension by Blocks and Procedures II (Denotational Semantics)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)



noll@cs.rwth-aachen.de

<http://www-i2.informatik.rwth-aachen.de/i2/svsw13/>

Summer Semester 2013

- 1 Recapitulation: Operational Semantics of Blocks and Procedures
- 2 Denotational Semantics: Handling Variable Declarations
- 3 Denotational Semantics: Handling Procedures
- 4 Two Examples
- 5 Justification of Fixpoint Semantics
- 6 Summary: Blocks and Procedures

Syntactic categories:

Category	Domain	Meta variable
Procedure identifiers	$PVar = \{P, Q, \dots\}$	P
Procedure declarations	$PDec$	p
Variable declarations	$VDec$	v
Commands (statements)	Cmd	c

Context-free grammar:

$p ::= \text{proc } P \text{ is } c; p \mid \varepsilon \in PDec$

$v ::= \text{var } x; v \mid \varepsilon \in VDec$

$c ::= \text{skip} \mid x := a \mid c_1; c_2 \mid \text{if } b \text{ then } c_1 \text{ else } c_2 \mid \text{while } b \text{ do } c \mid$
 $\text{call } P \mid \text{begin } v \text{ } p \text{ } c \text{ end} \in Cmd$

- So far: **states** $\Sigma = \{\sigma \mid \sigma : Var \rightarrow \mathbb{Z}\}$
- Now: explicit control over all (nested) **instances** of a variable:
 - **variable environments** $VEnv := \{\rho \mid \rho : Var \dashrightarrow Loc\}$
 - **locations** $Loc := \mathbb{N}$
 - **stores** $Sto := \{\sigma \mid \sigma : Loc \dashrightarrow \mathbb{Z}\}$
(**partial** function to maintain allocation information)

$\Rightarrow$ **Two-level access** to a variable $x \in Var$:

- 1 determine current memory location of x :

$$l := \rho(x)$$

- 2 reading/writing access to σ at position l

- Thus: previous **state** information represented as $\sigma \circ \rho$

- **Effect of procedure call** determined by its body and variable and procedure environment of its declaration:

$$PEnv := \{\pi \mid \pi : PVar \dashrightarrow Cmd \times VEnv \times PEnv\}$$

denotes the set of **procedure environments**

- **Effect of declaration**: update of environment (and store)

$$\text{upd}_V[\![\cdot]\!] : VDec \times VEnv \times Sto \rightarrow VEnv \times Sto$$

$$\text{upd}_V[\![\text{var } x; v]\!](\rho, \sigma) := \text{upd}_V[\![v]\!](\rho[x \mapsto l_x], \sigma[l_x \mapsto 0])$$

$$\text{upd}_V[\![\varepsilon]\!](\rho, \sigma) := (\rho, \sigma)$$

$$\text{upd}_P[\![\cdot]\!] : PDec \times VEnv \times PEnv \rightarrow PEnv$$

$$\text{upd}_P[\![\text{proc } P \text{ is } c; p]\!](\rho, \pi) := \text{upd}_P[\![p]\!](\rho, \pi[P \mapsto (c, \rho, \pi)])$$

$$\text{upd}_P[\![\varepsilon]\!](\rho, \pi) := \pi$$

where $l_x := \min\{l \in Loc \mid \sigma(l) = \perp\}$

Definition (Execution relation)

For $c \in \text{Cmd}$, $\sigma, \sigma' \in \text{Sto}$, $\rho \in \text{VEnv}$, and $\pi \in \text{PEnv}$, the **execution relation** $(\rho, \pi) \vdash \langle c, \sigma \rangle \rightarrow \sigma'$ ("in environment (ρ, π) , statement c transforms store σ into σ' ") is defined by the following rules:

$$\text{(skip)} \frac{}{(\rho, \pi) \vdash \langle \text{skip}, \sigma \rangle \rightarrow \sigma}$$

$$\text{(asgn)} \frac{\langle a, \sigma \circ \rho \rangle \rightarrow z}{(\rho, \pi) \vdash \langle x := a, \sigma \rangle \rightarrow \sigma[\rho(x) \mapsto z]}$$

$$\text{(seq)} \frac{(\rho, \pi) \vdash \langle c_1, \sigma \rangle \rightarrow \sigma' \quad (\rho, \pi) \vdash \langle c_2, \sigma' \rangle \rightarrow \sigma''}{(\rho, \pi) \vdash \langle c_1; c_2, \sigma \rangle \rightarrow \sigma''}$$

$$\text{(if-t)} \frac{\langle b, \sigma \circ \rho \rangle \rightarrow \text{true} \quad (\rho, \pi) \vdash \langle c_1, \sigma \rangle \rightarrow \sigma'}{(\rho, \pi) \vdash \langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'}$$

$$\text{(if-f)} \frac{\langle b, \sigma \circ \rho \rangle \rightarrow \text{false} \quad (\rho, \pi) \vdash \langle c_2, \sigma \rangle \rightarrow \sigma'}{(\rho, \pi) \vdash \langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'}$$

Definition (Execution relation; continued)

$$\text{(wh-f)} \frac{\langle b, \sigma \circ \rho \rangle \rightarrow \text{false}}{(\rho, \pi) \vdash \langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma}$$

$$\text{(wh-t)} \frac{\langle b, \sigma \circ \rho \rangle \rightarrow \text{true} \quad (\rho, \pi) \vdash \langle c, \sigma \rangle \rightarrow \sigma' \quad (\rho, \pi) \vdash \langle \text{while } b \text{ do } c, \sigma' \rangle \rightarrow \sigma''}{(\rho, \pi) \vdash \langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma''}$$

$$\text{(call)} \frac{(\rho', \pi'[P \mapsto (c, \rho', \pi')]) \vdash \langle c, \sigma \rangle \rightarrow \sigma'}{(\rho, \pi) \vdash \langle \text{call } P, \sigma \rangle \rightarrow \sigma'} \quad \text{if } \pi(P) = (c, \rho', \pi')$$

$$\text{(block)} \frac{\text{upd}_v \llbracket v \rrbracket (\rho, \sigma) = (\rho', \sigma') \quad (\rho', \text{upd}_p \llbracket p \rrbracket (\rho', \pi)) \vdash \langle c, \sigma' \rangle \rightarrow \sigma''}{(\rho, \pi) \vdash \langle \text{begin } v \text{ } p \text{ } c \text{ end}, \sigma \rangle \rightarrow \sigma''}$$

- 1 Recapitulation: Operational Semantics of Blocks and Procedures
- 2 Denotational Semantics: Handling Variable Declarations**
- 3 Denotational Semantics: Handling Procedures
- 4 Two Examples
- 5 Justification of Fixpoint Semantics
- 6 Summary: Blocks and Procedures

Handling Variable Declarations

Exactly as in operational semantics:

- **Variable environments** keep location information:

$$VEnv := \{\rho \mid \rho : Var \dashrightarrow Loc\}$$

with $Loc := \mathbb{N}$

- **Effect of variable declaration**: update of environment and store

$$\begin{aligned} \text{upd}_v[\![\cdot]\!] &: VDec \times VEnv \times Sto \rightarrow VEnv \times Sto \\ \text{upd}_v[\![\text{var } x; v]\!](\rho, \sigma) &:= \text{upd}_v[\![v]\!](\rho[x \mapsto l_x], \sigma[l_x \mapsto 0]) \\ \text{upd}_v[\![\varepsilon]\!](\rho, \sigma) &:= (\rho, \sigma) \end{aligned}$$

where $l_x := \min\{l \in Loc \mid \sigma(l) = \perp\}$

Statement Semantics Using Locations

- **First step:** reformulation of Definition 5.1 using **locations**
- **So far:** $\mathcal{C}[\cdot] : Cmd \rightarrow (\Sigma \dashrightarrow \Sigma)$

Definition 15.1 (Denotational semantics using locations)

The **(denotational) semantic functional for statements**,

$$\mathcal{C}'[\cdot] : Cmd \rightarrow VEnv \rightarrow (Sto \dashrightarrow Sto),$$

is given by:

$$\begin{aligned}\mathcal{C}'[\text{skip}]\rho &:= \text{id}_{Sto} \\ \mathcal{C}'[x := a]\rho \sigma &:= \sigma[\rho(x) \mapsto \mathcal{A}[a](\text{lookup } \rho \sigma)] \\ \mathcal{C}'[c_1; c_2]\rho &:= (\mathcal{C}'[c_2]\rho) \circ (\mathcal{C}'[c_1]\rho) \\ \mathcal{C}'[\text{if } b \text{ then } c_1 \text{ else } c_2]\rho &:= \text{cond}(\mathcal{B}[b] \circ (\text{lookup } \rho), \mathcal{C}'[c_1]\rho, \mathcal{C}'[c_2]\rho) \\ \mathcal{C}'[\text{while } b \text{ do } c]\rho &:= \text{fix}(\Phi)\end{aligned}$$

where $\text{lookup} : VEnv \rightarrow Sto \rightarrow \Sigma$ with $\text{lookup } \rho \sigma := \sigma \circ \rho$ and

$$\begin{aligned}\Phi : (Sto \dashrightarrow Sto) &\rightarrow (Sto \dashrightarrow Sto) : \\ f &\mapsto \text{cond}(\mathcal{B}[b] \circ (\text{lookup } \rho), f \circ \mathcal{C}'[c]\rho, \text{id}_{Sto})\end{aligned}$$

- 1 Recapitulation: Operational Semantics of Blocks and Procedures
- 2 Denotational Semantics: Handling Variable Declarations
- 3 Denotational Semantics: Handling Procedures**
- 4 Two Examples
- 5 Justification of Fixpoint Semantics
- 6 Summary: Blocks and Procedures

Procedure Environments

- **Procedure environments** now store **semantic** information:
 - So far: $PEnv := \{\pi \mid \pi : PVar \dashrightarrow Cmd \times VEnv \times PEnv\}$
 - Now: $PEnv := \{\pi \mid \pi : PVar \dashrightarrow (Sto \dashrightarrow Sto)\}$, to be used in $\mathfrak{C}''[\cdot] : Cmd \rightarrow VEnv \rightarrow PEnv \rightarrow (Sto \dashrightarrow Sto)$
- **Procedure declarations** (“**proc** P **is** c ”) update procedure environment:

$$upd_p[\cdot] : PDec \times VEnv \times PEnv \rightarrow PEnv$$

- **non-recursive** case: P not (indirectly) called within c
 $\Rightarrow \pi(P)$ immediately given by $\mathfrak{C}''[c]\rho \pi$

$$upd_p[\text{proc } P \text{ is } c; p](\rho, \pi) := upd_p[p](\rho, \pi[P \mapsto \mathfrak{C}''[c]\rho \pi])$$

- **recursive** case: $\pi(P)$ must be a solution of equation
 $f = \mathfrak{C}''[c]\rho \pi[P \mapsto f]$
(cf. fixpoint semantics of **while** loop – Slide 5.15)

$$upd_p[\text{proc } P \text{ is } c; p](\rho, \pi) := upd_p[p](\rho, \pi[P \mapsto \text{fix}(\Psi)])$$

where $\Psi : (Sto \dashrightarrow Sto) \rightarrow (Sto \dashrightarrow Sto) : f \mapsto \mathfrak{C}''[c]\rho \pi[P \mapsto f]$

- $upd_p[\varepsilon](\rho, \pi) := \pi$
- **Remark:** non-recursive is special case of recursive situation

Statement Semantics Including Procedures

So far: $\mathcal{E}'[\cdot] : \text{Cmd} \rightarrow \text{VEnv} \rightarrow (\text{Sto} \dashrightarrow \text{Sto})$

Definition 15.2 (Denotational semantics with procedures)

$$\mathcal{E}''[\cdot] : \text{Cmd} \rightarrow \text{VEnv} \rightarrow \text{PEnv} \rightarrow (\text{Sto} \dashrightarrow \text{Sto})$$

is given by:

$$\begin{aligned}\mathcal{E}''[\text{skip}] \rho \pi &:= \text{id}_{\text{Sto}} \\ \mathcal{E}''[x := a] \rho \pi \sigma &:= \sigma[\rho(x) \mapsto \mathcal{A}[a](\text{lookup } \rho \sigma)] \\ \mathcal{E}''[c_1; c_2] \rho \pi &:= (\mathcal{E}''[c_2] \rho \pi) \circ (\mathcal{E}''[c_1] \rho \pi) \\ \mathcal{E}''[\text{if } b \text{ then } c_1 \text{ else } c_2] \rho \pi &:= \text{cond}(\mathcal{B}[b] \circ (\text{lookup } \rho), \\ &\quad \mathcal{E}''[c_1] \rho \pi, \mathcal{E}''[c_2] \rho \pi) \\ \mathcal{E}''[\text{while } b \text{ do } c] \rho \pi &:= \text{fix}(\Phi) \\ \mathcal{E}''[\text{call } P] \rho \pi &:= \pi(P) \\ \mathcal{E}''[\text{begin } v \text{ } p \text{ } c \text{ end}] \rho \pi \sigma &:= \mathcal{E}''[c] \rho' \pi' \sigma'\end{aligned}$$

where $\text{upd}_v[v](\rho, \sigma) = (\rho', \sigma')$

$\text{upd}_p[p](\rho', \pi) = \pi'$

$\text{lookup } \rho \sigma := \sigma \circ \rho$

$\Phi(f) := \text{cond}(\mathcal{B}[b] \circ (\text{lookup } \rho), f \circ \mathcal{E}''[c] \rho \pi, \text{id}_{\text{Sto}})$

- 1 Recapitulation: Operational Semantics of Blocks and Procedures
- 2 Denotational Semantics: Handling Variable Declarations
- 3 Denotational Semantics: Handling Procedures
- 4 Two Examples
- 5 Justification of Fixpoint Semantics
- 6 Summary: Blocks and Procedures

Example 15.3 (Non-recursive procedure call)

(also demonstrates static scoping principle)

```
c = begin
  var x;
  proc P is x := x - 1;
  x := 2; } c1
  begin
    var x;
    x := 3;
    call P;
  end; } c2
end
```

- Initial environments/store: $\rho_\emptyset \in VEnv$, $\pi_\emptyset \in PEnv$, $\sigma_\emptyset \in Sto$
- Computation of $\mathcal{E}''[c]\rho_\emptyset \pi_\emptyset \sigma_\emptyset$: on the board

Example: Recursive Case

Example 15.4 (Recursive procedure call)

```
c = begin
  proc F is
    if x = 1 then
      skip;
    else
      y := x * y;
      x := x - 1;
      call F;
    } c1
  y := 1;
  call F; } c2
end
```

} p

- Initial environments/store: $\rho_1 := \rho_0[x \mapsto 0, y \mapsto 1] \in VEnv$, $\pi_0 \in PEnv$, $\sigma \in Sto$ (with $\sigma(0) \neq \perp$)
- Computation of $\mathcal{E}''[c]\rho_1 \pi_0 \sigma$: on the board

- 1 Recapitulation: Operational Semantics of Blocks and Procedures
- 2 Denotational Semantics: Handling Variable Declarations
- 3 Denotational Semantics: Handling Procedures
- 4 Two Examples
- 5 Justification of Fixpoint Semantics**
- 6 Summary: Blocks and Procedures

Lemma 15.5

- ① (cf. Lemma 6.9)
 $(\text{Sto} \dashrightarrow \text{Sto}, \sqsubseteq)$ is a **CCPO** where $f \sqsubseteq g$ iff for all $\sigma, \sigma' \in \Sigma$:
 $f(\sigma) = \sigma' \Rightarrow g(\sigma) = \sigma'$
- ② (cf. Lemmata 6.13 and 6.16)
Let $b \in \text{BExp}$, $c \in \text{Cmd}$, $\rho \in \text{VEnv}$, $\pi \in \text{PEnv}$, and
 $\Phi : (\text{Sto} \dashrightarrow \text{Sto}) \rightarrow (\text{Sto} \dashrightarrow \text{Sto})$ with
 $\Phi(f) := \text{cond}(\mathfrak{B}[\![b]\!] \circ (\text{lookup } \rho), f \circ \mathfrak{C}''[\![c]\!]\rho \pi, \text{id}_{\text{Sto}})$. Then Φ is
monotonic and continuous w.r.t. $(\text{Sto} \dashrightarrow \text{Sto}, \sqsubseteq)$.
- ③ Let $\text{proc } P \text{ is } c \in \text{PDec}$, $\rho \in \text{VEnv}$, $\pi \in \text{PEnv}$, and
 $\Psi : (\text{Sto} \dashrightarrow \text{Sto}) \rightarrow (\text{Sto} \dashrightarrow \text{Sto})$ with $\Psi(f) := \mathfrak{C}''[\![c]\!]\rho \pi[P \mapsto f]$.
Then Ψ is **monotonic and continuous** w.r.t. $(\text{Sto} \dashrightarrow \text{Sto}, \sqsubseteq)$.

Proof.

omitted



- 1 Recapitulation: Operational Semantics of Blocks and Procedures
- 2 Denotational Semantics: Handling Variable Declarations
- 3 Denotational Semantics: Handling Procedures
- 4 Two Examples
- 5 Justification of Fixpoint Semantics
- 6 Summary: Blocks and Procedures

Summary: Blocks and Procedures

- **Blocks** allow to declare local variables and recursive procedures
- Requires concept of **locations** to support instantiation of variables
- **Static scoping**: meaning of identifier determined by declaration context (rather than calling context)
- Meaning of **variable declaration**: storage allocation
- Meaning of **procedure call**:
 - operationally: **execution** of procedure body
 - ⇒ procedure environment records statement ("symbol table")
 - denotationally: **application** of procedure meaning
 - ⇒ procedure environment records (partial) store transformation
 - recursive behavior again handled by **fixpoint approach**
- Further extensions:
 - **axiomatic semantics** (for $\text{proc } P \text{ is } c \in PDec$)
 - non-recursive:
$$(\text{call}) \frac{\{A\} c \{B\}}{\{A\} \text{call } P \{B\}}$$
 - recursive:
$$(\text{call}) \frac{\{A\} \text{call } P \{B\} \vdash \{A\} c \{B\}}{\{A\} \text{call } P \{B\}}$$
 - **procedure parameters**
 - **higher-order procedures**