Semantics and Verification of Software

Summer Semester 2019

Lecture 16: Extension by Blocks and Procedures III (Axiomatic Semantics)

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ss-19/sv-sw/

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Outline of Lecture 16

Recap: Operational Semantics of Blocks and Procedures

Recap: Denotational Semantics of Blocks and Procedures

Axiomatic Semantics of Blocks and Procedures

Non-Recursive Procedures

Partial Correctness for Recursive Procedures

Total Correctness for Recursive Procedures

Recap: Operational Semantics of Blocks and Procedures

Procedure Environments and Declarations

- 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)

$$\mathrm{upd}_v[\![.]\!] : VDec \times VEnv \times Sto \to VEnv \times Sto$$

$$\begin{aligned} \mathrm{upd}_v[\![\texttt{var}\ x\,;v]\!](\rho,\sigma) &:= \mathrm{upd}_v[\![v]\!](\rho[x \mapsto l_x], \sigma[l_x \mapsto 0]) \\ \mathrm{upd}_v[\![\varepsilon]\!](\rho,\sigma) &:= (\rho,\sigma) \end{aligned}$$

$$\mathrm{upd}_p[\![.]\!] : PDec \times VEnv \times PEnv \to PEnv$$

$$\begin{aligned} \mathrm{upd}_p[\![\texttt{proc}\ P\ \texttt{is}\ c\ \texttt{end}\,;p]\!](\rho,\pi) &:= \mathrm{upd}_p[\![p]\!](\rho, \pi[P \mapsto (c,\rho,\pi)]) \\ \mathrm{upd}_p[\![\varepsilon]\!](\rho,\pi) &:= \pi \end{aligned}$$

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

Recap: Operational Semantics of Blocks and Procedures

Execution Relation I

Definition (Execution relation)

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

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

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

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

$$\text{(if-t)}\frac{\langle b, \sigma \circ \rho \rangle \to \mathsf{true} \qquad (\rho, \pi) \vdash \langle c_1, \sigma \rangle \to \sigma'}{(\rho, \pi) \vdash \langle \mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2\ \mathtt{end}, \sigma \rangle \to \sigma'}$$

Execution Relation II

Definition (Execution relation; continued)

$$\text{(if-f)}\ \frac{\langle b, \sigma \circ \rho\rangle \to \mathsf{false} \qquad (\rho, \pi) \vdash \langle c_2, \sigma\rangle \to \sigma'}{(\rho, \pi) \vdash \langle \mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2\ \mathtt{end}, \sigma\rangle \to \sigma'}$$

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

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

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

$$\text{(block)}\ \frac{\mathsf{upd}_v[\![v]\!](\rho, \sigma) = (\rho', \sigma') \qquad \mathsf{upd}_p[\![p]\!](\rho', \pi) = \pi' \qquad (\rho', \pi') \vdash \langle c, \sigma'\rangle \to \sigma''}{(\rho, \pi) \vdash \langle \mathtt{begin}\ v\ p\ c\ \mathtt{end}, \sigma\rangle \to \sigma''}$$

Outline of Lecture 16

Recap: Operational Semantics of Blocks and Procedures

Recap: Denotational Semantics of Blocks and Procedures

Axiomatic Semantics of Blocks and Procedures

Non-Recursive Procedures

Partial Correctness for Recursive Procedures

Total Correctness for Recursive 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 $\mathfrak{C}''[\![.]\!] : Cmd \to VEnv \to PEnv' \to (Sto \dashrightarrow Sto)$
- Procedure declarations (“$\texttt{proc}\ P\ \texttt{is}\ c\ \texttt{end}$”) update procedure environment:

$$\mathrm{upd}_p[\![.]\!] : PDec \times VEnv \times PEnv' \to PEnv'$$

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

$$\mathrm{upd}_p[\![\texttt{proc}\ P\ \texttt{is}\ c\ \texttt{end}\,;p]\!](\rho,\pi) := \mathrm{upd}_p[\![p]\!](\rho,\pi[P \mapsto \mathfrak{C}''[\![c]\!]\rho\ \pi])$$

  - recursive case: $\pi(P)$ must be a solution of equation $f = \mathfrak{C}''[\![c]\!]\rho\ \pi[P \mapsto f]$
    (cf. fixpoint semantics of $\texttt{while}$ loop – Slide 6.12):

$$\mathrm{upd}_p[\![\texttt{proc}\ P\ \texttt{is}\ c\ \texttt{end}\,;p]\!](\rho,\pi) := \mathrm{upd}_p[\![p]\!](\rho,\pi[P \mapsto \mathrm{fix}(\Psi)])$$

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

Recap: Denotational Semantics of Blocks and Procedures

Statement Semantics Including Procedures

Definition (Denotational semantics with procedures)

$$\mathfrak{C}''[\![.]\!] : \mathit{Cmd} \to \mathit{VEnv} \to \mathit{PEnv}' \to (\mathit{Sto} \dashrightarrow \mathit{Sto})$$

is given by

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

where

$$\begin{aligned}
\mathrm{upd}_v[\![v]\!](\rho, \sigma) &= (\rho', \sigma')\\
\mathrm{upd}_p[\![p]\!](\rho', \pi) &= \pi'\\
\mathrm{lookup}\ \rho\ \sigma &:= \sigma \circ \rho\\
\Phi(f) &:= \mathrm{cond}(\mathfrak{B}[\![b]\!] \circ (\mathrm{lookup}\ \rho), f \circ \mathfrak{C}''[\![c]\!]\rho\ \pi, \mathrm{id}_{Sto})
\end{aligned}$$

Outline of Lecture 16

Recap: Operational Semantics of Blocks and Procedures

Recap: Denotational Semantics of Blocks and Procedures

Axiomatic Semantics of Blocks and Procedures

Non-Recursive Procedures

Partial Correctness for Recursive Procedures

Total Correctness for Recursive Procedures

Axiomatic Semantics of Blocks and Procedures

The Approach

- For simplicity:
  - ignore nested blocks (i.e., all variables and procedures are global)
  - consider only statements with at most one procedure declaration

Axiomatic Semantics of Blocks and Procedures

The Approach

- For simplicity:
  - ignore nested blocks (i.e., all variables and procedures are global)
  - consider only statements with at most one procedure declaration
- Start with non-recursive procedures
  - approach: prove partial/total correctness of procedure call by showing partial/total correctness of procedure body (similarly to “inlining” in operational semantics)
  - non-termination due to recursive calls excluded
  - partial correctness coincides with total correctness (up to loops)

Axiomatic Semantics of Blocks and Procedures

The Approach

- For simplicity:
  - ignore nested blocks (i.e., all variables and procedures are global)
  - consider only statements with at most one procedure declaration
- Start with non-recursive procedures
  - approach: prove partial/total correctness of procedure call by showing partial/total correctness of procedure body (similarly to “inlining” in operational semantics)
  - non-termination due to recursive calls excluded
  - partial correctness coincides with total correctness (up to loops)
- Next step: recursive procedures
  - approach: prove partial/total correctness of procedure call by showing partial/total correctness of procedure body under the assumption that recursive calls behave correctly
  - non-terminating recursive calls possible
  - requires additional means to ensure total correctness

Outline of Lecture 16

Recap: Operational Semantics of Blocks and Procedures

Recap: Denotational Semantics of Blocks and Procedures

Axiomatic Semantics of Blocks and Procedures

Non-Recursive Procedures

Partial Correctness for Recursive Procedures

Total Correctness for Recursive Procedures

Proof Rules for Non-Recursive Procedures

Idea: a property that holds for the body of a procedure also applies to its calls

Definition 16.1 (Proof rule for partial correctness; extends Definition 9.12)

For $\texttt{proc}\ P\ \texttt{is}\ c\ \texttt{end} \in PDec$:

$$\text{(call)}\frac{\{A\}\,c\,\{B\}}{\{A\}\,\texttt{call}\ P\,\{B\}}$$

Proof Rules for Non-Recursive Procedures

Idea: a property that holds for the body of a procedure also applies to its calls

Definition 16.1 (Proof rule for partial correctness; extends Definition 9.12)

For $\texttt{proc}\ P\ \texttt{is}\ c\ \texttt{end} \in PDec$:

$$\text{(call)}\ \frac{\{A\}\ c\ \{B\}}{\{A\}\ \texttt{call}\ P\ \{B\}}$$

Definition 16.2 (Proof rule for total correctness; extends Definition 11.13)

For $\texttt{proc}\ P\ \texttt{is}\ c\ \texttt{end} \in PDec$:

$$\text{(call)}\ \frac{\{A\}\ c\ \{\Downarrow B\}}{\{A\}\ \texttt{call}\ P\ \{\Downarrow B\}}$$

An Example Proof

Example 16.3

Let $c \in Cmd$ be given by

```
begin
  var x; var y; var t;
  proc P is
    t := x; x := y; y := t
  end;
  call P
end
```

and $i, j \in LVar$. Then:

$$\vdash \{\mathtt{x} = i \wedge \mathtt{y} = j\}\, \mathtt{call\ P}\, \{\Downarrow \mathtt{x} = j \wedge \mathtt{y} = i\}$$

(on the board)

Outline of Lecture 16

Recap: Operational Semantics of Blocks and Procedures

Recap: Denotational Semantics of Blocks and Procedures

Axiomatic Semantics of Blocks and Procedures

Non-Recursive Procedures

Partial Correctness for Recursive Procedures

Total Correctness for Recursive Procedures

Proof Rules for Recursive Procedures

Observation: previous proof rules insufficient to handle recursive case

- correctness proof of `call` P requires correctness proof of body c
- correctness proof of c requires correctness proof of each `call` P within c
- ...

Partial Correctness for Recursive Procedures

Proof Rules for Recursive Procedures

Observation: previous proof rules insufficient to handle recursive case

- correctness proof of $\texttt{call}\ P$ requires correctness proof of body c
- correctness proof of c requires correctness proof of each $\texttt{call}\ P$ within c
- ...

Idea: employ inductive reasoning

- prove correctness of call by showing correctness of body under the assumption that each recursive call satisfies the correctness property
- requires extension of proof system by conditional provability relations of the form

$$\{C\}\,\texttt{call}\ P\,\{D\} \vdash \{A\}\,c\,\{B\}$$

meaning: “assuming that $\{C\}\,\texttt{call}\ P\,\{D\}$ is provable, $\{A\}\,c\,\{B\}$ can also be shown”

Partial Correctness for Recursive Procedures

Proof Rules for Recursive Procedures

Observation: previous proof rules insufficient to handle recursive case

- correctness proof of $\texttt{call}\ P$ requires correctness proof of body c
- correctness proof of c requires correctness proof of each $\texttt{call}\ P$ within c
- ...

Idea: employ inductive reasoning

- prove correctness of call by showing correctness of body under the assumption that each recursive call satisfies the correctness property
- requires extension of proof system by conditional provability relations of the form

$$\{C\}\,\texttt{call}\ P\,\{D\} \vdash \{A\}\,c\,\{B\}$$

  meaning: "assuming that $\{C\}\,\texttt{call}\ P\,\{D\}$ is provable, $\{A\}\,c\,\{B\}$ can also be shown"

Definition 16.4 (Proof rule for partial correctness; extends Definition 9.12)

For $\texttt{proc}\ P\ \texttt{is}\ c\ \texttt{end} \in \mathit{PDec}$:

$$\text{(call)}\ \frac{\{A\}\,\texttt{call}\ P\,\{B\} \vdash \{A\}\,c\,\{B\}}{\{A\}\,\texttt{call}\ P\,\{B\}}$$

Another Example Proof

Example 16.5 (cf. Example 15.4)

$$c = \texttt{begin}\ \texttt{proc F is}\ \underbrace{\texttt{if x = 1 then skip; else y := x * y; x := x - 1; call F end}}_{c_{\mathrm{F}}}\ \texttt{end}\ \texttt{y := 1; call F; end}$$

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

To prove:

$$\vdash \{\mathtt{x} > 0 \land i = \mathtt{y} \cdot \mathtt{x}!\}\ \mathtt{call\ F}\ \{\mathtt{y} = i\}$$

(on the board)

Outline of Lecture 16

Recap: Operational Semantics of Blocks and Procedures

Recap: Denotational Semantics of Blocks and Procedures

Axiomatic Semantics of Blocks and Procedures

Non-Recursive Procedures

Partial Correctness for Recursive Procedures

Total Correctness for Recursive Procedures

Total Correctness for Recursive Procedures

Proof Rules for Total Correctness

- Approach: to ensure termination, we have to bound the depth of recursive calls
- Guaranteed by equipping preconditions with depth bound parameter (similarly to iteration counter for `while` loop in Definition 11.13)

Proof Rules for Total Correctness

- Approach: to ensure termination, we have to bound the depth of recursive calls
- Guaranteed by equipping preconditions with depth bound parameter (similarly to iteration counter for `while` loop in Definition 11.13)

Definition 16.6 (Proof rule for total correctness; extends Definition 11.13)

For $\texttt{proc}\ P\ \texttt{is}\ c\ \texttt{end} \in PDec$:

$$\text{(call)}\ \frac{\{i \geq 0 \wedge A(i)\}\ \texttt{call}\ P\ \{\Downarrow B\} \vdash \{i \geq 0 \wedge A(i+1)\}\ c\ \{\Downarrow B\} \qquad \models \neg A(0)}{\{\exists i. i \geq 0 \wedge A(i)\}\ \texttt{call}\ P\ \{\Downarrow B\}}$$

- Premises:
  - if $\{A(i)\}\ \texttt{call}\ P\ \{\Downarrow B\}$ is provable for all recursive calls of depth at most $i \geq 0$, then we can prove that a call at level $i+1$ will be correct
  - $\neg A(0)$ disables calls at level 0
- Conclusion: for any depth $i \geq 0$ of recursive calls, we have a proof of $\{A(i)\}\ \texttt{call}\ P\ \{\Downarrow B\}$

Yet Another Example Proof

Example 16.7 (cf. Example 15.4)

$$
c = \texttt{begin} \quad \texttt{proc F is} \quad \left.\begin{array}{l}\texttt{if x = 1 then} \\ \quad\texttt{skip;} \\ \texttt{else} \\ \quad\texttt{y := x * y;} \\ \quad\texttt{x := x - 1;} \\ \quad\texttt{call F} \\ \texttt{end}\end{array}\right\} c_{\mathrm{F}} \quad \texttt{end} \quad \texttt{y := 1;} \quad \texttt{call F;} \quad \texttt{end}
$$

To prove:

$$\vdash \{\texttt{x} > \texttt{0}\}\, \texttt{call F}\, \{\Downarrow \mathsf{true}\}$$

(on the board)