Semantics and Verification of Software

Lecture 13: Denotational Semantics of Procedures

Thomas Noll

Lehrstuhl für Informatik 2
RWTH Aachen University
noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/svsw/

Summer semester 2007

Outline

Denotational Semantics I

- As before: statements denote storage transformations
- New: dependence on environments

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

- Variable environment obtained as before:

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

- New declaration functional takes role of update functions:

$$\mathfrak{D}_v[\![.]\!] : VDec \to (VEnv \to VEnv)$$
$$\mathfrak{D}_v[\![\texttt{var}\ x\,;v]\!]\rho := \mathfrak{D}_v[\![v]\!]\rho[x \mapsto \mathsf{min}\{l \in \mathbb{N} \mid \rho(l) = \bot\}]$$
$$\mathfrak{D}_v[\![\varepsilon]\!]\rho := \rho$$

- Procedures now interpreted as storage transformations:

$$PDec := \{\pi \mid \pi : PVar \dashrightarrow (Sto \dashrightarrow Sto)\}$$

- Recursive procedure calls involve fixpoints:

$$\mathfrak{D}_p[\![.]\!] : PDec \to (VEnv \times PEnv \to PEnv)$$
$$\mathfrak{D}_p[\![\texttt{proc}\ P\ \texttt{is}\ c\,;p]\!](\rho,\pi) := \mathfrak{D}_p[\![p]\!](\rho,\pi[P \mapsto \mathsf{fix}(\Phi)])$$
$$\mathfrak{D}_p[\![\varepsilon]\!](\rho,\pi) := \pi$$

where

$$\Phi : (Sto \dashrightarrow Sto) \to (Sto \dashrightarrow Sto)$$
$$\Phi(f) := \mathfrak{C}[\![c]\!](\rho,\pi[P \mapsto f])$$

Denotational Semantics I

- As before: statements denote storage transformations
- New: dependence on environments

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

- Variable environment obtained as before:

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

- New declaration functional takes role of update functions:

$$\mathfrak{D}_v[\![.]\!] : VDec \to (VEnv \to VEnv)$$
$$\mathfrak{D}_v[\![\texttt{var}\ x\,;v]\!]\rho := \mathfrak{D}_v[\![v]\!]\rho[x \mapsto \mathsf{min}\{l \in \mathbb{N} \mid \rho(l) = \bot\}]$$
$$\mathfrak{D}_v[\![\varepsilon]\!]\rho := \rho$$

- Procedures now interpreted as storage transformations:

$$PDec := \{\pi \mid \pi : PVar \dashrightarrow (Sto \dashrightarrow Sto)\}$$

- Recursive procedure calls involve fixpoints:

$$\mathfrak{D}_p[\![.]\!] : PDec \to (VEnv \times PEnv \to PEnv)$$
$$\mathfrak{D}_p[\![\texttt{proc}\ P\ \texttt{is}\ c\,;p]\!](\rho,\pi) := \mathfrak{D}_p[\![p]\!](\rho,\pi[P \mapsto \mathsf{fix}(\Phi)])$$
$$\mathfrak{D}_p[\![\varepsilon]\!](\rho,\pi) := \pi$$

where

$$\Phi : (Sto \dashrightarrow Sto) \to (Sto \dashrightarrow Sto)$$
$$\Phi(f) := \mathfrak{C}[\![c]\!](\rho,\pi[P \mapsto f])$$

Denotational Semantics I

- As before: statements denote storage transformations
- New: dependence on environments

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

- Variable environment obtained as before:

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

- New declaration functional takes role of update functions:

$$\mathfrak{D}_v[\![.]\!] : VDec \to (VEnv \to VEnv)$$
$$\mathfrak{D}_v[\![\texttt{var}\ x\,;v]\!]\rho := \mathfrak{D}_v[\![v]\!]\rho[x \mapsto \mathsf{min}\{l \in \mathbb{N} \mid \rho(l) = \bot\}]$$
$$\mathfrak{D}_v[\![\varepsilon]\!]\rho := \rho$$

- Procedures now interpreted as storage transformations:

$$PDec := \{\pi \mid \pi : PVar \dashrightarrow (Sto \dashrightarrow Sto)\}$$

- Recursive procedure calls involve fixpoints:

$$\mathfrak{D}_p[\![.]\!] : PDec \to (VEnv \times PEnv \to PEnv)$$
$$\mathfrak{D}_p[\![\texttt{proc}\ P\ \texttt{is}\ c\,;p]\!](\rho,\pi) := \mathfrak{D}_p[\![p]\!](\rho, \pi[P \mapsto \mathsf{fix}(\Phi)])$$
$$\mathfrak{D}_p[\![\varepsilon]\!](\rho,\pi) := \pi$$

where

$$\Phi : (Sto \dashrightarrow Sto) \to (Sto \dashrightarrow Sto)$$
$$\Phi(f) := \mathfrak{C}[\![c]\!](\rho, \pi[P \mapsto f])$$

Denotational Semantics I

- As before: statements denote storage transformations
- New: dependence on environments

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

- Variable environment obtained as before:

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

- New declaration functional takes role of update functions:

$$\mathfrak{D}_v[\![.]\!] : VDec \to (VEnv \to VEnv)$$
$$\mathfrak{D}_v[\![\texttt{var}\ x\,;v]\!]\rho := \mathfrak{D}_v[\![v]\!]\rho[x \mapsto \mathsf{min}\{l \in \mathbb{N} \mid \rho(l) = \bot\}]$$
$$\mathfrak{D}_v[\![\varepsilon]\!]\rho := \rho$$

- Procedures now interpreted as storage transformations:

$$PDec := \{\pi \mid \pi : PVar \dashrightarrow (Sto \dashrightarrow Sto)\}$$

- Recursive procedure calls involve fixpoints:

$$\mathfrak{D}_p[\![.]\!] : PDec \to (VEnv \times PEnv \to PEnv)$$
$$\mathfrak{D}_p[\![\texttt{proc}\ P\ \texttt{is}\ c\,;p]\!](\rho,\pi) := \mathfrak{D}_p[\![p]\!](\rho,\pi[P \mapsto \mathsf{fix}(\Phi)])$$
$$\mathfrak{D}_p[\![\varepsilon]\!](\rho,\pi) := \pi$$

where

$$\Phi : (Sto \dashrightarrow Sto) \to (Sto \dashrightarrow Sto)$$
$$\Phi(f) := \mathfrak{C}[\![c]\!](\rho,\pi[P \mapsto f])$$

Denotational Semantics I

- As before: statements denote storage transformations
- New: dependence on environments

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

- Variable environment obtained as before:

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

- New declaration functional takes role of update functions:

$$\mathfrak{D}_v[\![.]\!] : VDec \to (VEnv \to VEnv)$$
$$\mathfrak{D}_v[\![\texttt{var}\ x\,;v]\!]\rho := \mathfrak{D}_v[\![v]\!]\rho[x \mapsto \mathsf{min}\{l \in \mathbb{N} \mid \rho(l) = \bot\}]$$
$$\mathfrak{D}_v[\![\varepsilon]\!]\rho := \rho$$

- Procedures now interpreted as storage transformations:

$$PDec := \{\pi \mid \pi : PVar \dashrightarrow (Sto \dashrightarrow Sto)\}$$

- Recursive procedure calls involve fixpoints:

$$\mathfrak{D}_p[\![.]\!] : PDec \to (VEnv \times PEnv \to PEnv)$$
$$\mathfrak{D}_p[\![\texttt{proc}\ P\ \texttt{is}\ c\,;p]\!](\rho,\pi) := \mathfrak{D}_p[\![p]\!](\rho,\pi[P \mapsto \mathsf{fix}(\Phi)])$$
$$\mathfrak{D}_p[\![\varepsilon]\!](\rho,\pi) := \pi$$

where

$$\Phi : (Sto \dashrightarrow Sto) \to (Sto \dashrightarrow Sto)$$
$$\Phi(f) := \mathfrak{C}[\![c]\!](\rho,\pi[P \mapsto f])$$

Denotational Semantics I

- As before: statements denote storage transformations
- New: dependence on environments

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

- Variable environment obtained as before:

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

- New declaration functional takes role of update functions:

$$\mathfrak{D}_v[\![.]\!] : VDec \to (VEnv \to VEnv)$$
$$\mathfrak{D}_v[\![\texttt{var}\ x\,;v]\!]\rho := \mathfrak{D}_v[\![v]\!]\rho[x \mapsto \mathsf{min}\{l \in \mathbb{N} \mid \rho(l) = \bot\}]$$
$$\mathfrak{D}_v[\![\varepsilon]\!]\rho := \rho$$

- Procedures now interpreted as storage transformations:

$$PDec := \{\pi \mid \pi : PVar \dashrightarrow (Sto \dashrightarrow Sto)\}$$

- Recursive procedure calls involve fixpoints:

$$\mathfrak{D}_p[\![.]\!] : PDec \to (VEnv \times PEnv \to PEnv)$$
$$\mathfrak{D}_p[\![\texttt{proc}\ P\ \texttt{is}\ c\,;p]\!](\rho,\pi) := \mathfrak{D}_p[\![p]\!](\rho,\pi[P \mapsto \mathsf{fix}(\Phi)])$$
$$\mathfrak{D}_p[\![\varepsilon]\!](\rho,\pi) := \pi$$

where

$$\Phi : (Sto \dashrightarrow Sto) \to (Sto \dashrightarrow Sto)$$
$$\Phi(f) := \mathfrak{C}[\![c]\!](\rho,\pi[P \mapsto f])$$

Denotational Semantics II

- Fixpoint approach motivated by equation

$$f = \mathfrak{C}[\![c]\!](\rho, \pi[P \mapsto f])$$

- $(Sto \dashrightarrow Sto, \sqsubseteq)$ is a chain–complete partial order
  (cf. Lemma 5.5 for $(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$)
- Φ is continuous
  (cf. Lemma 6.6 for $\Phi(f) := \mathsf{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma)$)

$\Longrightarrow$ Well–definedness of $\mathfrak{D}_p[\![.]\!]$

- In particular for a non–recursive procedure P (i.e., no P–call in c):

$$\mathfrak{D}_p[\![\texttt{proc } P \texttt{ is } c]\!](\rho, \pi) = \pi[P \mapsto \mathfrak{C}[\![c]\!](\rho, \pi)]$$

Denotational Semantics II

- Fixpoint approach motivated by equation

$$f = \mathfrak{C}[\![c]\!](\rho, \pi[P \mapsto f])$$

- $(Sto \dashrightarrow Sto, \sqsubseteq)$ is a chain–complete partial order
  (cf. Lemma 5.5 for $(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$)
- Φ is continuous
  (cf. Lemma 6.6 for $\Phi(f) := \mathsf{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma)$)

$\Longrightarrow$ Well–definedness of $\mathfrak{D}_p[\![.]\!]$

- In particular for a non–recursive procedure P (i.e., no P–call in c):

$$\mathfrak{D}_p[\![\texttt{proc } P \texttt{ is } c]\!](\rho, \pi) = \pi[P \mapsto \mathfrak{C}[\![c]\!](\rho, \pi)]$$

Denotational Semantics II

- Fixpoint approach motivated by equation

$$f = \mathfrak{C}[\![c]\!](\rho, \pi[P \mapsto f])$$

- $(Sto \dashrightarrow Sto, \sqsubseteq)$ is a chain–complete partial order
  (cf. Lemma 5.5 for $(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$)
- Φ is continuous
  (cf. Lemma 6.6 for $\Phi(f) := \mathsf{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma)$)

$\Longrightarrow$ Well–definedness of $\mathfrak{D}_p[\![.]\!]$

- In particular for a non–recursive procedure P (i.e., no P–call in c):

$$\mathfrak{D}_p[\![\texttt{proc } P \texttt{ is } c]\!](\rho, \pi) = \pi[P \mapsto \mathfrak{C}[\![c]\!](\rho, \pi)]$$

Denotational Semantics II

- Fixpoint approach motivated by equation

$$f = \mathfrak{C}[\![c]\!](\rho, \pi[P \mapsto f])$$

- $(\mathit{Sto} \dashrightarrow \mathit{Sto}, \sqsubseteq)$ is a chain–complete partial order
  (cf. Lemma 5.5 for $(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$)
- Φ is continuous
  (cf. Lemma 6.6 for $\Phi(f) := \mathsf{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma)$)

$\Longrightarrow$ Well–definedness of $\mathfrak{D}_p[\![.]\!]$

- In particular for a non–recursive procedure P (i.e., no P–call in c):

$$\mathfrak{D}_p[\![\texttt{proc } P \texttt{ is } c]\!](\rho, \pi) = \pi[P \mapsto \mathfrak{C}[\![c]\!](\rho, \pi)]$$

Definition 13.1 (Denotational semantics of statements)

The (denotational) semantic functional for statements,

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

is given by:

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

where

$$\Phi : (Sto \dashrightarrow Sto) \to (Sto \dashrightarrow Sto) : f \mapsto \mathsf{cond}(\mathfrak{B}[\![b]\!] \circ \rho, f \circ \mathfrak{C}[\![c]\!](\rho,\pi), \mathsf{id}_{Sto})$$

Example: Non–Recursive Case

Example 13.2

Let c be given by

```
begin
  var x;                    } v
  proc P is y := x;         } p
  x := 1;                   } c1
  begin                     }
    var x;                  }
    x := 2;                 } c2
    call P                  }
  end                       }
end.
```

Example: Non–Recursive Case

Example 13.2

Let c be given by

```
begin
  var x;                    } v
  proc P is y := x;         } p
  x := 1;                   } c1
  begin                     }
    var x;                  }
    x := 2;                 } c2
    call P                  }
  end                       }
end.
```

Let $\rho_0 := \rho_\emptyset[\mathtt{y} \mapsto 0] \in VEnv$,
$\rho_1 := \rho_0[\mathtt{x} \mapsto 1] \in VEnv$,
$\pi_1 := \pi_\emptyset[P \mapsto \mathfrak{C}[\![\mathtt{y := x}]\!](\rho_1, \pi_\emptyset)]$
$\in PEnv$.

Then $\mathfrak{D}_v[\![v]\!]\rho_0 = \rho_1$

$\mathfrak{D}_p[\![p]\!](\rho_1, \pi_\emptyset) = (\rho_1, \pi_1)$

Thus, for every $\sigma \in Sto$,

$$\pi_1(P)(\sigma) = \sigma[\rho_1(\mathtt{y}) \mapsto \sigma(\rho_1(\mathtt{x}))] = \sigma[0 \mapsto \sigma(1)].$$

Altogether, for any $\sigma_0 \in Sto$,

$$\begin{aligned}
&\mathfrak{C}[\![c]\!](\rho_0, \pi_\emptyset)(\sigma_0) \\
&= \mathfrak{C}[\![c_1; c_2]\!](\rho_1, \pi_1)(\sigma_0) \\
&= \mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\mathfrak{C}[\![c_1]\!](\rho_1, \pi_1)(\sigma_0)) \\
&= \mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\underbrace{\sigma_0[1 \mapsto 1]}_{=:\ \sigma_1})
\end{aligned}$$

Decomposing c_2 yields

$$\mathfrak{D}_v[\![\mathtt{var\ x;}]\!]\rho_1 = \underbrace{\rho_1[\mathtt{x} \mapsto 2]}_{=:\ \rho_2}$$

which gives the overall result

$$\begin{aligned}
&\mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\sigma_1) \\
&= \mathfrak{C}[\![\mathtt{call\ P}]\!](\rho_2, \pi_1)(\underbrace{\sigma_1[2 \mapsto 2]}_{=:\ \sigma_2}) \\
&= \sigma_2[0 \mapsto \sigma_2(1)] \\
&= \sigma_0[0 \mapsto 1, 1 \mapsto 1, 2 \mapsto 2]
\end{aligned}$$

Example: Non–Recursive Case

Example 13.2

Let c be given by

```
begin
  var x;                    } v
  proc P is y := x;         } p
  x := 1;                   } c1
  begin                     }
    var x;                  }
    x := 2;                 } c2
    call P                  }
  end                       }
end.
```

Let $\rho_0 := \rho_\emptyset[\mathtt{y} \mapsto 0] \in VEnv$,
$\rho_1 := \rho_0[\mathtt{x} \mapsto 1] \in VEnv$,
$\pi_1 := \pi_\emptyset[P \mapsto \mathfrak{C}[\![\mathtt{y := x}]\!](\rho_1, \pi_\emptyset)]$
$\in PEnv$.

Then $\mathfrak{D}_v[\![v]\!]\rho_0 = \rho_1$
$\mathfrak{D}_p[\![p]\!](\rho_1, \pi_\emptyset) = (\rho_1, \pi_1)$

Thus, for every $\sigma \in Sto$,

$$\begin{aligned}\pi_1(P)(\sigma) &= \sigma[\rho_1(\mathtt{y}) \mapsto \sigma(\rho_1(\mathtt{x}))] \\ &= \sigma[0 \mapsto \sigma(1)].\end{aligned}$$

Altogether, for any $\sigma_0 \in Sto$,

$$\begin{aligned}&\mathfrak{C}[\![c]\!](\rho_0, \pi_\emptyset)(\sigma_0) \\ &= \mathfrak{C}[\![c_1; c_2]\!](\rho_1, \pi_1)(\sigma_0) \\ &= \mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\mathfrak{C}[\![c_1]\!](\rho_1, \pi_1)(\sigma_0)) \\ &= \mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\underbrace{\sigma_0[1 \mapsto 1]}_{=:\,\sigma_1})\end{aligned}$$

Decomposing c_2 yields

$$\mathfrak{D}_v[\![\mathtt{var\ x;}]\!]\rho_1 = \underbrace{\rho_1[\mathtt{x} \mapsto 2]}_{=:\,\rho_2}$$

which gives the overall result

$$\begin{aligned}&\mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\sigma_1) \\ &= \mathfrak{C}[\![\mathtt{call\ P}]\!](\rho_2, \pi_1)(\underbrace{\sigma_1[2 \mapsto 2]}_{=:\,\sigma_2}) \\ &= \sigma_2[0 \mapsto \sigma_2(1)] \\ &= \sigma_0[0 \mapsto 1, 1 \mapsto 1, 2 \mapsto 2]\end{aligned}$$

Example: Non–Recursive Case

Example 13.2

Let c be given by

```
begin
  var x;                    } v
  proc P is y := x;         } p
  x := 1;                   } c1
  begin                     }
    var x;                  }
    x := 2;                 } c2
    call P                  }
  end                       }
end.
```

Let $\rho_0 := \rho_\emptyset[\mathtt{y} \mapsto 0] \in VEnv$,
$\rho_1 := \rho_0[\mathtt{x} \mapsto 1] \in VEnv$,
$\pi_1 := \pi_\emptyset[P \mapsto \mathfrak{C}[\![\mathtt{y := x}]\!](\rho_1, \pi_\emptyset)]$
$\in PEnv$.

Then $\mathfrak{D}_v[\![v]\!]\rho_0 = \rho_1$
$\mathfrak{D}_p[\![p]\!](\rho_1, \pi_\emptyset) = (\rho_1, \pi_1)$

Thus, for every $\sigma \in Sto$,
$\pi_1(P)(\sigma) = \sigma[\rho_1(\mathtt{y}) \mapsto \sigma(\rho_1(\mathtt{x}))]$
$= \sigma[0 \mapsto \sigma(1)]$.

Altogether, for any $\sigma_0 \in Sto$,
$\mathfrak{C}[\![c]\!](\rho_0, \pi_\emptyset)(\sigma_0)$
$= \mathfrak{C}[\![c_1; c_2]\!](\rho_1, \pi_1)(\sigma_0)$
$= \mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\mathfrak{C}[\![c_1]\!](\rho_1, \pi_1)(\sigma_0))$
$= \mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\underbrace{\sigma_0[1 \mapsto 1]}_{=:\ \sigma_1})$

Decomposing c_2 yields
$\mathfrak{D}_v[\![\mathtt{var\ x;}]\!]\rho_1 = \underbrace{\rho_1[\mathtt{x} \mapsto 2]}_{=:\ \rho_2}$

which gives the overall result
$\mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\sigma_1)$
$= \mathfrak{C}[\![\mathtt{call\ P}]\!](\rho_2, \pi_1)(\underbrace{\sigma_1[2 \mapsto 2]}_{=:\ \sigma_2})$
$= \sigma_2[0 \mapsto \sigma_2(1)]$
$= \sigma_0[0 \mapsto 1, 1 \mapsto 1, 2 \mapsto 2]$

Example: Non–Recursive Case

Example 13.2

Let c be given by

```
begin
  var x;                    } v
  proc P is y := x;         } p
  x := 1;                   } c1
  begin                     }
    var x;                  }
    x := 2;                 } c2
    call P                  }
  end                       }
end.
```

Let $\rho_0 := \rho_\emptyset[\mathtt{y} \mapsto 0] \in VEnv$,
$\rho_1 := \rho_0[\mathtt{x} \mapsto 1] \in VEnv$,
$\pi_1 := \pi_\emptyset[P \mapsto \mathfrak{C}[\![\mathtt{y := x}]\!](\rho_1, \pi_\emptyset)] \in PEnv$.

Then $\mathfrak{D}_v[\![v]\!]\rho_0 = \rho_1$

$\mathfrak{D}_p[\![p]\!](\rho_1, \pi_\emptyset) = (\rho_1, \pi_1)$

Thus, for every $\sigma \in Sto$,

$$\pi_1(P)(\sigma) = \sigma[\rho_1(\mathtt{y}) \mapsto \sigma(\rho_1(\mathtt{x}))] = \sigma[0 \mapsto \sigma(1)].$$

Altogether, for any $\sigma_0 \in Sto$,

$$\begin{aligned}
&\mathfrak{C}[\![c]\!](\rho_0, \pi_\emptyset)(\sigma_0) \\
&= \mathfrak{C}[\![c_1; c_2]\!](\rho_1, \pi_1)(\sigma_0) \\
&= \mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\mathfrak{C}[\![c_1]\!](\rho_1, \pi_1)(\sigma_0)) \\
&= \mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\underbrace{\sigma_0[1 \mapsto 1]}_{=:\ \sigma_1})
\end{aligned}$$

Decomposing c_2 yields

$$\mathfrak{D}_v[\![\mathtt{var\ x;}]\!]\rho_1 = \underbrace{\rho_1[\mathtt{x} \mapsto 2]}_{=:\ \rho_2}$$

which gives the overall result

$$\begin{aligned}
&\mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\sigma_1) \\
&= \mathfrak{C}[\![\mathtt{call\ P}]\!](\rho_2, \pi_1)(\underbrace{\sigma_1[2 \mapsto 2]}_{=:\ \sigma_2}) \\
&= \sigma_2[0 \mapsto \sigma_2(1)] \\
&= \sigma_0[0 \mapsto 1, 1 \mapsto 1, 2 \mapsto 2]
\end{aligned}$$

Example: Non–Recursive Case

Example 13.2

Let c be given by

```
begin
  var x;                    } v
  proc P is y := x;         } p
  x := 1;                   } c1
  begin                     }
    var x;                  }
    x := 2;                 } c2
    call P                  }
  end                       }
end.
```

Let $\rho_0 := \rho_\emptyset[\mathtt{y} \mapsto 0] \in VEnv$,
$\rho_1 := \rho_0[\mathtt{x} \mapsto 1] \in VEnv$,
$\pi_1 := \pi_\emptyset[P \mapsto \mathfrak{C}[\![\mathtt{y := x}]\!](\rho_1, \pi_\emptyset)]$
$\in PEnv$.

Then $\mathfrak{D}_v[\![v]\!]\rho_0 = \rho_1$
$\mathfrak{D}_p[\![p]\!](\rho_1, \pi_\emptyset) = (\rho_1, \pi_1)$

Thus, for every $\sigma \in Sto$,
$\pi_1(P)(\sigma) = \sigma[\rho_1(\mathtt{y}) \mapsto \sigma(\rho_1(\mathtt{x}))]$
$= \sigma[0 \mapsto \sigma(1)]$.

Altogether, for any $\sigma_0 \in Sto$,
$\mathfrak{C}[\![c]\!](\rho_0, \pi_\emptyset)(\sigma_0)$
$= \mathfrak{C}[\![c_1; c_2]\!](\rho_1, \pi_1)(\sigma_0)$
$= \mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\mathfrak{C}[\![c_1]\!](\rho_1, \pi_1)(\sigma_0))$
$= \mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\underbrace{\sigma_0[1 \mapsto 1]}_{=:\ \sigma_1})$

Decomposing c_2 yields
$\mathfrak{D}_v[\![\mathtt{var\ x;}]\!]\rho_1 = \underbrace{\rho_1[\mathtt{x} \mapsto 2]}_{=:\ \rho_2}$

which gives the overall result
$\mathfrak{C}[\![c_2]\!](\rho_1, \pi_1)(\sigma_1)$
$= \mathfrak{C}[\![\mathtt{call\ P}]\!](\rho_2, \pi_1)(\underbrace{\sigma_1[2 \mapsto 2]}_{=:\ \sigma_2})$
$= \sigma_2[0 \mapsto \sigma_2(1)]$
$= \sigma_0[0 \mapsto 1, 1 \mapsto 1, 2 \mapsto 2]$

Example: Recursive Case I

Example 13.3

The semantics of procedure declaration p

```
proc Mult is
  begin
    if x > 0 then z := z+y; x := x-1; call Mult
             else skip
  end
```

$\Big\}\, c$

with respect to variable environment $\rho_0 := \rho_\emptyset[\mathtt{x} \mapsto 0, \mathtt{y} \mapsto 1, \mathtt{z} \mapsto 2]$ is given by

$$\mathfrak{D}_p[\![p]\!](\rho_0, \pi_\emptyset) = \pi_\emptyset[\mathtt{Mult} \mapsto \mathrm{fix}(\Phi)]$$

where, for every $f : Sto \dashrightarrow Sto$ and $\sigma \in Sto$,

$$\Phi(f)(\sigma) = \mathfrak{C}[\![c]\!](\rho_0, \pi_\emptyset[\mathtt{Mult} \mapsto f])(\sigma)$$

$$= \begin{cases} f(\sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto \sigma(0) - 1]) & \text{if } \sigma(0) > 0 \\ \sigma & \text{otherwise} \end{cases}$$

Example: Recursive Case I

Example 13.3

The semantics of procedure declaration p

```
proc Mult is
  begin
    if x > 0 then z := z+y; x := x-1; call Mult
             else skip
  end
```

$\Big\}\, c$

with respect to variable environment $\rho_0 := \rho_\emptyset[\mathtt{x} \mapsto 0, \mathtt{y} \mapsto 1, \mathtt{z} \mapsto 2]$ is given by

$$\mathfrak{D}_p[\![p]\!](\rho_0, \pi_\emptyset) = \pi_\emptyset[\mathtt{Mult} \mapsto \mathsf{fix}(\Phi)]$$

where, for every $f : \mathit{Sto} \dashrightarrow \mathit{Sto}$ and $\sigma \in \mathit{Sto}$,

$$\begin{aligned}\Phi(f)(\sigma) &= \mathfrak{C}[\![c]\!](\rho_0, \pi_\emptyset[\mathtt{Mult} \mapsto f])(\sigma) \\ &= \begin{cases} f(\sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto \sigma(0) - 1]) & \text{if } \sigma(0) > 0 \\ \sigma & \text{otherwise} \end{cases}\end{aligned}$$

Example: Recursive Case II

Example 13.3 (continued)

Computation of $\mathsf{fix}(\Phi) = \bigsqcup_{n \in \mathbb{N}} \Phi^n(f_\emptyset)$ by fixpoint iteration (Theorem 7.1):

$$f_0(\sigma) := f_\emptyset(\sigma) = \bot$$

$$f_1(\sigma) := \Phi(f_0)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 0 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$f_2(\sigma) := \Phi(f_1)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 1 \\ \sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 1 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$f_3(\sigma) := \Phi(f_2)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 2 \\ \sigma[2 \mapsto \sigma(2) + 2 * \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 2 \\ \sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 1 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$\vdots$$

$$f_n(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) \geq n \\ \sigma[2 \mapsto \sigma(2) + \sigma(0) * \sigma(1), 0 \mapsto 0] & \text{if } 0 < \sigma(0) < n \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

In the limit we obtain

$$\mathsf{fix}(\Phi)(\sigma) = \begin{cases} \sigma[2 \mapsto \sigma(2) + \sigma(0) * \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) > 0 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

Example: Recursive Case II

Example 13.3 (continued)

Computation of $\mathsf{fix}(\Phi) = \bigsqcup_{n \in \mathbb{N}} \Phi^n(f_\emptyset)$ by fixpoint iteration (Theorem 7.1):

$$
\begin{aligned}
f_0(\sigma) &:= f_\emptyset(\sigma) \\
&= \bot \\
f_1(\sigma) &:= \Phi(f_0)(\sigma) \\
&= \begin{cases} \bot & \text{if } \sigma(0) > 0 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}
\end{aligned}
$$

Example: Recursive Case II

Example 13.3 (continued)

Computation of $\mathsf{fix}(\Phi) = \bigsqcup_{n\in\mathbb{N}} \Phi^n(f_\emptyset)$ by fixpoint iteration (Theorem 7.1):

$$f_0(\sigma) := f_\emptyset(\sigma) = \bot$$

$$f_1(\sigma) := \Phi(f_0)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 0 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$f_2(\sigma) := \Phi(f_1)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 1 \\ \sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 1 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$f_3(\sigma) := \Phi(f_2)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 2 \\ \sigma[2 \mapsto \sigma(2) + 2 * \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 2 \\ \sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 1 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$\vdots$$

$$f_n(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) \geq n \\ \sigma[2 \mapsto \sigma(2) + \sigma(0) * \sigma(1), 0 \mapsto 0] & \text{if } 0 < \sigma(0) < n \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

In the limit we obtain

$$\mathsf{fix}(\Phi)(\sigma) = \begin{cases} \sigma[2 \mapsto \sigma(2) + \sigma(0) * \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) > 0 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

Example: Recursive Case II

Example 13.3 (continued)

Computation of $\mathsf{fix}(\Phi) = \bigsqcup_{n\in\mathbb{N}} \Phi^n(f_\emptyset)$ by fixpoint iteration (Theorem 7.1):

$$f_0(\sigma) := f_\emptyset(\sigma) = \bot$$

$$f_1(\sigma) := \Phi(f_0)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 0 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$f_2(\sigma) := \Phi(f_1)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 1 \\ \sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 1 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$f_3(\sigma) := \Phi(f_2)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 2 \\ \sigma[2 \mapsto \sigma(2) + 2 * \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 2 \\ \sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 1 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$\vdots$$

$$f_n(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) \geq n \\ \sigma[2 \mapsto \sigma(2) + \sigma(0) * \sigma(1), 0 \mapsto 0] & \text{if } 0 < \sigma(0) < n \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

In the limit we obtain

$$\mathsf{fix}(\Phi)(\sigma) = \begin{cases} \sigma[2 \mapsto \sigma(2) + \sigma(0) * \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) > 0 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

Example: Recursive Case II

Example 13.3 (continued)

Computation of $\mathsf{fix}(\Phi) = \bigsqcup_{n\in\mathbb{N}} \Phi^n(f_\emptyset)$ by fixpoint iteration (Theorem 7.1):

$$f_0(\sigma) := f_\emptyset(\sigma) = \bot$$

$$f_1(\sigma) := \Phi(f_0)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 0 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$f_2(\sigma) := \Phi(f_1)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 1 \\ \sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 1 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$f_3(\sigma) := \Phi(f_2)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 2 \\ \sigma[2 \mapsto \sigma(2) + 2 * \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 2 \\ \sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 1 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$\vdots$$

$$f_n(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) \geq n \\ \sigma[2 \mapsto \sigma(2) + \sigma(0) * \sigma(1), 0 \mapsto 0] & \text{if } 0 < \sigma(0) < n \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

In the limit we obtain

$$\mathsf{fix}(\Phi)(\sigma) = \begin{cases} \sigma[2 \mapsto \sigma(2) + \sigma(0) * \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) > 0 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

Example: Recursive Case II

Example 13.3 (continued)

Computation of $\mathsf{fix}(\Phi) = \bigsqcup_{n\in\mathbb{N}} \Phi^n(f_\emptyset)$ by fixpoint iteration (Theorem 7.1):

$$f_0(\sigma) := f_\emptyset(\sigma) = \bot$$

$$f_1(\sigma) := \Phi(f_0)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 0 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$f_2(\sigma) := \Phi(f_1)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 1 \\ \sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 1 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$f_3(\sigma) := \Phi(f_2)(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) > 2 \\ \sigma[2 \mapsto \sigma(2) + 2 * \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 2 \\ \sigma[2 \mapsto \sigma(2) + \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) = 1 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

$$\vdots$$

$$f_n(\sigma) = \begin{cases} \bot & \text{if } \sigma(0) \geq n \\ \sigma[2 \mapsto \sigma(2) + \sigma(0) * \sigma(1), 0 \mapsto 0] & \text{if } 0 < \sigma(0) < n \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$

In the limit we obtain

$$\mathsf{fix}(\Phi)(\sigma) = \begin{cases} \sigma[2 \mapsto \sigma(2) + \sigma(0) * \sigma(1), 0 \mapsto 0] & \text{if } \sigma(0) > 0 \\ \sigma & \text{if } \sigma(0) \leq 0 \end{cases}$$