Semantics and Verification of Software

Lecture 7: Denotational Semantics of WHILE III (The Fixpoint Theorem and Its Application)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH AACHEN UNIVERSITY

noll@cs.rwth-aachen.de

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

Summer Semester 2013

Outline

1. Recapitulation: CCPOs and Continuous Functions
2. The Fixpoint Theorem
3. Application to fix(Φ)
4. Summary: Denotational Semantics
5. Equivalence of Operational and Denotational Semantics

Characterization of fix(Φ)

Goals:

- Prove existence of fix(Φ) for $\Phi(f) = \text{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma)$
- Show how it can be "computed" (more exactly: approximated)

Sufficient conditions:

on domain $\Sigma \dashrightarrow \Sigma$: chain-complete partial order

on function Φ: continuity

Chain Completeness

Definition (Chain completeness)

A partial order is called chain complete (CCPO) if every of its chains has a least upper bound.

Lemma

- $(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$ *is a CCPO with least element* $f_\emptyset$ *where* $\mathsf{graph}(f_\emptyset) = \emptyset$.
- *In particular, for every chain* $S \subseteq \Sigma \dashrightarrow \Sigma$,

$$\mathsf{graph}\left(\bigsqcup S\right) = \bigcup_{f \in S} \mathsf{graph}(f).$$

Monotonicity

Definition (Monotonicity)

Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be partial orders, and let $F : D \to D'$. F is called monotonic (w.r.t. $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$) if, for every $d_1, d_2 \in D$,

$$d_1 \sqsubseteq d_2 \Rightarrow F(d_1) \sqsubseteq' F(d_2).$$

Interpretation: monotonic functions "preserve information"

Lemma

Let $b \in BExp$, $c \in Cmd$, *and* $\Phi : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma)$ *with* $\Phi(f) := \mathrm{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathrm{id}_\Sigma)$. *Then* Φ *is monotonic w.r.t.* $(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$.

Continuity

A function F is continuous if the order of applying F and taking LUBs can be reversed:

Definition (Continuity)

Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be CCPOs and $F : D \to D'$ monotonic. Then F is called continuous (w.r.t. $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$) if, for every non-empty chain $S \subseteq D$,

$$F\left(\bigsqcup S\right) = \bigsqcup F(S).$$

Lemma

Let $b \in BExp$, $c \in Cmd$, *and* $\Phi(f) := \mathrm{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathrm{id}_\Sigma)$. *Then* Φ *is continuous w.r.t.* $(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$.

Outline

The Fixpoint Theorem

Alfred Tarski (1901–1983)

Bronislaw Knaster (1893–1990)

Theorem 7.1 (Fixpoint Theorem by Tarski and Knaster)

Let $(D, \sqsubseteq)$ *be a CCPO and* $F : D \to D$ *continuous. Then*

$$\mathrm{fix}(F) := \bigsqcup \left\{ F^n \left(\bigsqcup \emptyset \right) \mid n \in \mathbb{N} \right\}$$

is the least fixpoint of F *where*

$$F^0(d) := d \text{ and } F^{n+1}(d) := F(F^n(d)).$$

Alfred Tarski (1901–1983)

Bronislaw Knaster (1893–1990)

Theorem 7.1 (Fixpoint Theorem by Tarski and Knaster)

Let $(D, \sqsubseteq)$ *be a CCPO and* $F : D \to D$ *continuous. Then*

$$\mathrm{fix}(F) := \bigsqcup \left\{ F^n \left(\bigsqcup \emptyset \right) \mid n \in \mathbb{N} \right\}$$

is the least fixpoint of F *where*

$$F^0(d) := d \text{ and } F^{n+1}(d) := F(F^n(d)).$$

Proof.

on the board □

An Example

Example 7.2

- **Domain:** $(2^{\mathbb{N}}, \subseteq)$ (CCPO with $\bigsqcup S = \bigcup_{N \in S} N$ – see Ex. 6.7)

An Example

Example 7.2

- **Domain:** $(2^{\mathbb{N}}, \subseteq)$ (CCPO with $\bigsqcup S = \bigcup_{N \in S} N$ – see Ex. 6.7)
- **Function:** $F : 2^{\mathbb{N}} \to 2^{\mathbb{N}} : N \mapsto N \cup A$ for some fixed $A \subseteq \mathbb{N}$
  - F monotonic: $M \subseteq N \Rightarrow F(M) = M \cup A \subseteq N \cup A = F(N)$
  - F continuous: $F(\bigsqcup S) = F\left(\bigcup_{N \in S} N\right) = \left(\bigcup_{N \in S} N\right) \cup A = \bigcup_{N \in S}(N \cup A) = \bigcup_{N \in S} F(N) = \bigsqcup F(S)$

An Example

Example 7.2

- **Domain:** $(2^{\mathbb{N}}, \subseteq)$ (CCPO with $\bigsqcup S = \bigcup_{N \in S} N$ – see Ex. 6.7)
- **Function:** $F : 2^{\mathbb{N}} \to 2^{\mathbb{N}} : N \mapsto N \cup A$ for some fixed $A \subseteq \mathbb{N}$
  - F monotonic: $M \subseteq N \Rightarrow F(M) = M \cup A \subseteq N \cup A = F(N)$
  - F continuous: $F(\bigsqcup S) = F\left(\bigcup_{N \in S} N\right) = \left(\bigcup_{N \in S} N\right) \cup A = \bigcup_{N \in S}(N \cup A) = \bigcup_{N \in S} F(N) = \bigsqcup F(S)$
- **Fixpoint iteration:** $N_n := F^n(\bigsqcup \emptyset)$ where $\bigsqcup \emptyset = \emptyset$
  - $N_0 = \bigsqcup \emptyset = \emptyset$
  - $N_1 = F(N_0) = \emptyset \cup A = A$
  - $N_2 = F(N_1) = A \cup A = A = N_n$ for every $n \geq 1$

  $\Rightarrow$ $\text{fix}(F) = A$

An Example

Example 7.2

- **Domain:** $(2^{\mathbb{N}}, \subseteq)$ (CCPO with $\bigsqcup S = \bigcup_{N \in S} N$ – see Ex. 6.7)
- **Function:** $F : 2^{\mathbb{N}} \to 2^{\mathbb{N}} : N \mapsto N \cup A$ for some fixed $A \subseteq \mathbb{N}$
  - F monotonic: $M \subseteq N \Rightarrow F(M) = M \cup A \subseteq N \cup A = F(N)$
  - F continuous: $F(\bigsqcup S) = F\left(\bigcup_{N \in S} N\right) = \left(\bigcup_{N \in S} N\right) \cup A = \bigcup_{N \in S}(N \cup A) = \bigcup_{N \in S} F(N) = \bigsqcup F(S)$
- **Fixpoint iteration:** $N_n := F^n(\bigsqcup \emptyset)$ where $\bigsqcup \emptyset = \emptyset$
  - $N_0 = \bigsqcup \emptyset = \emptyset$
  - $N_1 = F(N_0) = \emptyset \cup A = A$
  - $N_2 = F(N_1) = A \cup A = A = N_n$ for every $n \geq 1$

  $\Rightarrow \text{fix}(F) = A$
- **Alternatively:** $F(N) := N \cap A$

  $\Rightarrow \text{fix}(F) = \emptyset$

Outline

Application to $\mathsf{fix}(\Phi)$

Altogether this completes the definition of $\mathfrak{C}[\![.]\!]$. In particular, for the `while` statement we obtain:

Corollary 7.3

Let $b \in BExp$, $c \in Cmd$, *and* $\Phi(f) := \mathsf{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma)$. *Then*

$$\mathsf{graph}(\mathsf{fix}(\Phi)) = \bigcup_{n \in \mathbb{N}} \mathsf{graph}(\Phi^n(f_\emptyset))$$

Application to $\mathrm{fix}(\Phi)$

Altogether this completes the definition of $\mathfrak{C}[\![.]\!]$. In particular, for the `while` statement we obtain:

Corollary 7.3

Let $b \in BExp$, $c \in Cmd$, *and* $\Phi(f) := \mathrm{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathrm{id}_\Sigma)$. *Then*

$$\mathrm{graph}(\mathrm{fix}(\Phi)) = \bigcup_{n \in \mathbb{N}} \mathrm{graph}(\Phi^n(f_\emptyset))$$

Proof.

Using

- Lemma 6.9
  - $(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$ CCPO with least element $f_\emptyset$
  - LUB = union of graphs
- Lemma 6.16 (Φ continuous)
- Theorem 7.1 (Fixpoint Theorem)

□

Denotational Semantics of Factorial Program I

Example 7.4 (Factorial program)

- Let $c \in Cmd$ be given by

```
y:=1; while ¬(x=1) do (y:=y*x; x:=x-1)
```

Denotational Semantics of Factorial Program I

Example 7.4 (Factorial program)

- Let $c \in Cmd$ be given by

  $$\texttt{y:=1; while } \neg\texttt{(x=1) do (y:=y*x; x:=x-1)}$$

- For every initial state $\sigma_0 \in \Sigma$, Def. 5.1 yields:

  $$\mathfrak{C}[\![c]\!](\sigma_0) = \mathsf{fix}(\Phi)(\sigma_1)$$

  where $\sigma_1 := \sigma_0[\mathrm{y} \mapsto 1]$ and, for every $f : \Sigma \dashrightarrow \Sigma$ and $\sigma \in \Sigma$,

  $$\begin{aligned}\Phi(f)(\sigma) &= \mathsf{cond}(\mathfrak{B}[\![\neg\texttt{(x=1)}]\!], f \circ \mathfrak{C}[\![\texttt{y:=y*x; x:=x-1}]\!], \mathsf{id}_\Sigma)(\sigma) \\ &= \begin{cases}\sigma & \text{if } \sigma(\mathrm{x}) = 1 \\ f(\sigma') & \text{otherwise}\end{cases}\end{aligned}$$

  with $\sigma' := \sigma[\mathrm{y} \mapsto \sigma(\mathrm{y}) * \sigma(\mathrm{x}), \mathrm{x} \mapsto \sigma(\mathrm{x}) - 1]$.

Denotational Semantics of Factorial Program I

Example 7.4 (Factorial program)

- Let $c \in Cmd$ be given by

  $$\texttt{y:=1; while } \neg\texttt{(x=1) do (y:=y*x; x:=x-1)}$$

- For every initial state $\sigma_0 \in \Sigma$, Def. 5.1 yields:

  $$\mathfrak{C}[\![c]\!](\sigma_0) = \mathsf{fix}(\Phi)(\sigma_1)$$

  where $\sigma_1 := \sigma_0[\mathtt{y} \mapsto 1]$ and, for every $f : \Sigma \dashrightarrow \Sigma$ and $\sigma \in \Sigma$,

  $$\begin{aligned}\Phi(f)(\sigma) &= \mathsf{cond}(\mathfrak{B}[\![\neg\texttt{(x=1)}]\!], f \circ \mathfrak{C}[\![\texttt{y:=y*x; x:=x-1}]\!], \mathsf{id}_\Sigma)(\sigma)\\ &= \begin{cases}\sigma & \text{if } \sigma(\mathtt{x}) = 1\\ f(\sigma') & \text{otherwise}\end{cases}\end{aligned}$$

  with $\sigma' := \sigma[\mathtt{y} \mapsto \sigma(\mathtt{y}) * \sigma(\mathtt{x}), \mathtt{x} \mapsto \sigma(\mathtt{x}) - 1]$.

- Approximations of least fixpoint of Φ according to Theorem 7.1:

  $$\mathsf{fix}(\Phi) = \bigsqcup\{\Phi^n(f_\emptyset) \mid n \in \mathbb{N}\}$$

  (where $\mathsf{graph}(f_\emptyset) = \emptyset$)

Denotational Semantics of Factorial Program II

$$\Phi(f)(\sigma) = \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ f(\sigma') & \text{otherwise} \end{cases} \qquad \sigma' = \sigma[\mathtt{y} \mapsto \sigma(\mathtt{y}) * \sigma(\mathtt{x}), \mathtt{x} \mapsto \sigma(\mathtt{x}) - 1]$$

Example 7.4 (Factorial program; continued)

$$\begin{aligned} f_0(\sigma) &:= \Phi^0(f_\emptyset)(\sigma) \\ &= f_\emptyset(\sigma) \\ &= \text{undefined} \end{aligned}$$

Denotational Semantics of Factorial Program II

$$\Phi(f)(\sigma) = \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ f(\sigma') & \text{otherwise} \end{cases} \qquad \sigma' = \sigma[\mathtt{y} \mapsto \sigma(\mathtt{y}) * \sigma(\mathtt{x}), \mathtt{x} \mapsto \sigma(\mathtt{x}) - 1]$$

Example 7.4 (Factorial program; continued)

$$\begin{aligned} f_0(\sigma) &:= \Phi^0(f_\emptyset)(\sigma) \\ &= f_\emptyset(\sigma) \\ &= \text{undefined} \end{aligned}$$

$$\begin{aligned} f_1(\sigma) &:= \Phi^1(f_\emptyset)(\sigma) \\ &= \Phi(f_0)(\sigma) \\ &= \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ f_0(\sigma') & \text{otherwise} \end{cases} \\ &= \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ \text{undefined} & \text{otherwise} \end{cases} \end{aligned}$$

Denotational Semantics of Factorial Program II

$$\Phi(f)(\sigma) = \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ f(\sigma') & \text{otherwise} \end{cases} \qquad \sigma' = \sigma[\mathtt{y} \mapsto \sigma(\mathtt{y}) * \sigma(\mathtt{x}), \mathtt{x} \mapsto \sigma(\mathtt{x}) - 1]$$

Example 7.4 (Factorial program; continued)

$$\begin{aligned} f_0(\sigma) &:= \Phi^0(f_\emptyset)(\sigma) \\ &= f_\emptyset(\sigma) \\ &= \text{undefined} \end{aligned}$$

$$\begin{aligned} f_1(\sigma) &:= \Phi^1(f_\emptyset)(\sigma) \\ &= \Phi(f_0)(\sigma) \\ &= \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ f_0(\sigma') & \text{otherwise} \end{cases} \\ &= \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ \text{undefined} & \text{otherwise} \end{cases} \end{aligned}$$

$$\begin{aligned} f_2(\sigma) &:= \Phi^2(f_\emptyset)(\sigma) \\ &= \Phi(f_1)(\sigma) \\ &= \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ f_1(\sigma') & \text{otherwise} \end{cases} \\ &= \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ \sigma' & \text{if } \sigma(\mathtt{x}) \neq 1 \text{ and } \sigma'(\mathtt{x}) = 1 \\ \text{undefined} & \text{if } \sigma(\mathtt{x}) \neq 1 \text{ and } \sigma'(\mathtt{x}) \neq 1 \end{cases} \\ &= \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ \sigma' & \text{if } \sigma(\mathtt{x}) = 2 \\ \text{undefined} & \text{if } \sigma(\mathtt{x}) \neq 1 \text{ and } \sigma(\mathtt{x}) \neq 2 \end{cases} \\ &= \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ \sigma[y \mapsto 2 * \sigma(\mathtt{y}), \\ \quad \mathtt{x} \mapsto 1] & \text{if } \sigma(\mathtt{x}) = 2 \\ \text{undefined} & \text{if } \sigma(\mathtt{x}) \neq 1 \\ & \text{and } \sigma(\mathtt{x}) \neq 2 \end{cases} \end{aligned}$$

Denotational Semantics of Factorial Program III

$$\Phi(f)(\sigma) = \begin{cases} \sigma & \text{if } \sigma(x) = 1 \\ f(\sigma') & \text{otherwise} \end{cases} \qquad \sigma' = \sigma[y \mapsto \sigma(y) * \sigma(x), x \mapsto \sigma(x) - 1]$$

Example 7.4 (Factorial program; continued)

$$\begin{aligned}
f_3(\sigma) &:= \Phi^3(f_\emptyset)(\sigma) \\
&= \Phi(f_2)(\sigma) \\
&= \begin{cases} \sigma & \text{if } \sigma(x) = 1 \\ f_2(\sigma') & \text{otherwise} \end{cases} \\
&= \begin{cases} \sigma & \text{if } \sigma(x) = 1 \\ \sigma' & \text{if } \sigma(x) \neq 1 \text{ and } \sigma'(x) = 1 \\ \sigma'[y \mapsto 2 * \sigma'(y), x \mapsto 1] & \text{if } \sigma(x) \neq 1 \text{ and } \sigma'(x) = 2 \\ \text{undefined} & \text{if } \sigma(x) \neq 1 \text{ and } \sigma'(x) \neq 1 \text{ and } \sigma'(x) \neq 2 \end{cases} \\
&= \begin{cases} \sigma & \text{if } \sigma(x) = 1 \\ \sigma' & \text{if } \sigma(x) = 2 \\ \sigma'[y \mapsto 2 * \sigma'(y), x \mapsto 1] & \text{if } \sigma(x) = 3 \\ \text{undefined} & \text{if } \sigma(x) \notin \{1, 2, 3\} \end{cases} \\
&= \begin{cases} \sigma & \text{if } \sigma(x) = 1 \\ \sigma[y \mapsto 2 * \sigma(y), x \mapsto 1] & \text{if } \sigma(x) = 2 \\ \sigma[y \mapsto 3 * 2 * \sigma(y), x \mapsto 1] & \text{if } \sigma(x) = 3 \\ \text{undefined} & \text{if } \sigma(x) \notin \{1, 2, 3\} \end{cases}
\end{aligned}$$

Denotational Semantics of Factorial Program IV

$$\Phi(f)(\sigma) = \begin{cases} \sigma & \text{if } \sigma(\mathrm{x}) = 1 \\ f(\sigma') & \text{otherwise} \end{cases} \qquad \sigma' = \sigma[\mathrm{y} \mapsto \sigma(\mathrm{y}) * \sigma(\mathrm{x}), \mathrm{x} \mapsto \sigma(\mathrm{x}) - 1]$$

Example 7.4 (Factorial program; continued)

- n-th approximation:

$$\begin{aligned} & f_n(\sigma) \\ & := \Phi^n(f_\emptyset)(\sigma) \\ & = \begin{cases} \sigma[\mathrm{y} \mapsto \sigma(\mathrm{x}) * (\sigma(\mathrm{x}) - 1) * \ldots * 2 * \sigma(\mathrm{y}), & \text{if } 1 \leq \sigma(\mathrm{x}) \leq n \\ \quad \mathrm{x} \mapsto 1] & \\ \text{undefined} & \text{if } \sigma(\mathrm{x}) \notin \{1, \ldots, n\} \end{cases} \\ & = \begin{cases} \sigma[\mathrm{y} \mapsto (\sigma(\mathrm{x}))! * \sigma(\mathrm{y}), \mathrm{x} \mapsto 1] & \text{if } 1 \leq \sigma(\mathrm{x}) \leq n \\ \text{undefined} & \text{if } \sigma(\mathrm{x}) \notin \{1, \ldots, n\} \end{cases} \end{aligned}$$

Denotational Semantics of Factorial Program IV

$$\Phi(f)(\sigma) = \begin{cases} \sigma & \text{if } \sigma(\mathtt{x}) = 1 \\ f(\sigma') & \text{otherwise} \end{cases} \qquad \sigma' = \sigma[\mathtt{y} \mapsto \sigma(\mathtt{y}) * \sigma(\mathtt{x}), \mathtt{x} \mapsto \sigma(\mathtt{x}) - 1]$$

Example 7.4 (Factorial program; continued)

- n-th approximation:

$$\begin{aligned} & f_n(\sigma) \\ & := \Phi^n(f_\emptyset)(\sigma) \\ & = \begin{cases} \sigma[\mathtt{y} \mapsto \sigma(\mathtt{x}) * (\sigma(\mathtt{x}) - 1) * \ldots * 2 * \sigma(\mathtt{y}), & \text{if } 1 \leq \sigma(\mathtt{x}) \leq n \\ \quad \mathtt{x} \mapsto 1] & \\ \text{undefined} & \text{if } \sigma(\mathtt{x}) \notin \{1, \ldots, n\} \end{cases} \\ & = \begin{cases} \sigma[\mathtt{y} \mapsto (\sigma(\mathtt{x}))! * \sigma(\mathtt{y}), \mathtt{x} \mapsto 1] & \text{if } 1 \leq \sigma(\mathtt{x}) \leq n \\ \text{undefined} & \text{if } \sigma(\mathtt{x}) \notin \{1, \ldots, n\} \end{cases} \end{aligned}$$

- Fixpoint:

$$\mathfrak{C}[\![c]\!](\sigma_0) = \text{fix}(\Phi)(\sigma_1) = \begin{cases} \sigma[\mathtt{y} \mapsto (\sigma(\mathtt{x}))!, \mathtt{x} \mapsto 1] & \text{if } \sigma(\mathtt{x}) \geq 1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

Outline

Summary: Denotational Semantics

- Semantic model: partial state transformations ($\Sigma \dashrightarrow \Sigma$)

Summary: Denotational Semantics

- Semantic model: partial state transformations $(\Sigma \dashrightarrow \Sigma)$
- Compositional definition of functional $\mathfrak{C}[\![.]\!] : Cmd \to (\Sigma \dashrightarrow \Sigma)$

Summary: Denotational Semantics

- Semantic model: partial state transformations $(\Sigma \dashrightarrow \Sigma)$
- Compositional definition of functional $\mathfrak{C}[\![.]\!] : Cmd \to (\Sigma \dashrightarrow \Sigma)$
- Capturing the recursive nature of loops by a fixpoint definition (for a continuous function on a CCPO)

Summary: Denotational Semantics

- Semantic model: partial state transformations $(\Sigma \dashrightarrow \Sigma)$
- Compositional definition of functional $\mathfrak{C}[\![.]\!] : Cmd \to (\Sigma \dashrightarrow \Sigma)$
- Capturing the recursive nature of loops by a fixpoint definition (for a continuous function on a CCPO)
- Approximation by fixpoint iteration

Outline

Equivalence of Semantics I

Remember: in Def. 4.1, $\mathfrak{O}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma)$ was given by

$$\mathfrak{O}[\![c]\!](\sigma) = \sigma' \iff \langle c, \sigma \rangle \to \sigma'$$

Remember: in Def. 4.1, $\mathfrak{O}[\![.]\!] : Cmd \to (\Sigma \dashrightarrow \Sigma)$ was given by

$$\mathfrak{O}[\![c]\!](\sigma) = \sigma' \iff \langle c, \sigma \rangle \to \sigma'$$

Theorem 7.5 (Coincidence Theorem)

For every $c \in Cmd$,

$$\mathfrak{O}[\![c]\!] = \mathfrak{C}[\![c]\!],$$

i.e., $\langle c, \sigma \rangle \to \sigma'$ iff $\mathfrak{C}[\![c]\!](\sigma) = \sigma'$, and thus $\mathfrak{O}[\![.]\!] = \mathfrak{C}[\![.]\!]$.

Equivalence of Semantics II

The proof of Theorem 7.5 employs the following auxiliary propositions:

Lemma 7.6

1. *For every* $a \in AExp$, $\sigma \in \Sigma$, *and* $z \in \mathbb{Z}$:

$$\langle a, \sigma \rangle \to z \iff \mathfrak{A}[\![a]\!](\sigma) = z.$$

Equivalence of Semantics II

The proof of Theorem 7.5 employs the following auxiliary propositions:

Lemma 7.6

1. *For every* $a \in AExp$, $\sigma \in \Sigma$, *and* $z \in \mathbb{Z}$*:*
$$\langle a, \sigma \rangle \to z \iff \mathfrak{A}[\![a]\!](\sigma) = z.$$
2. *For every* $b \in BExp$, $\sigma \in \Sigma$, *and* $t \in \mathbb{B}$*:*
$$\langle b, \sigma \rangle \to t \iff \mathfrak{B}[\![b]\!](\sigma) = t.$$

Equivalence of Semantics II

The proof of Theorem 7.5 employs the following auxiliary propositions:

Lemma 7.6

1. *For every* $a \in AExp$, $\sigma \in \Sigma$, *and* $z \in \mathbb{Z}$:

$$\langle a, \sigma \rangle \rightarrow z \iff \mathfrak{A}[\![a]\!](\sigma) = z.$$

2. *For every* $b \in BExp$, $\sigma \in \Sigma$, *and* $t \in \mathbb{B}$:

$$\langle b, \sigma \rangle \rightarrow t \iff \mathfrak{B}[\![b]\!](\sigma) = t.$$

Proof.

1. structural induction on a
2. structural induction on b

□

Equivalence of Semantics III

Proof (Theorem 7.5).

We have to show that

$$\langle c, \sigma \rangle \to \sigma' \iff \mathfrak{C}[\![c]\!](\sigma) = \sigma'$$

- $\Rightarrow$ by structural induction over the derivation tree of $\langle c, \sigma \rangle \to \sigma'$
- $\Leftarrow$ by structural induction over c (with a nested complete induction over fixpoint index n)

(on the board) □

Overview: Operational/Denotational Semantics

Definition (3.2; Execution relation for statements)

$$\text{(skip)}\ \frac{}{\langle \texttt{skip}, \sigma\rangle \to \sigma} \qquad \text{(asgn)}\ \frac{\langle a, \sigma\rangle \to z}{\langle x \texttt{ := } a, \sigma\rangle \to \sigma[x \mapsto z]}$$

$$\text{(seq)}\ \frac{\langle c_1, \sigma\rangle \to \sigma' \quad \langle c_2, \sigma'\rangle \to \sigma''}{\langle c_1 \texttt{;} c_2, \sigma\rangle \to \sigma''} \qquad \text{(if-t)}\ \frac{\langle b, \sigma\rangle \to \mathsf{true} \quad \langle c_1, \sigma\rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \to \sigma'}$$

$$\text{(if-f)}\ \frac{\langle b, \sigma\rangle \to \mathsf{false} \quad \langle c_2, \sigma\rangle \to \sigma'}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \to \sigma'} \qquad \text{(wh-f)}\ \frac{\langle b, \sigma\rangle \to \mathsf{false}}{\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \to \sigma}$$

$$\text{(wh-t)}\ \frac{\langle b, \sigma\rangle \to \mathsf{true} \quad \langle c, \sigma\rangle \to \sigma' \quad \langle \texttt{while } b \texttt{ do } c, \sigma'\rangle \to \sigma''}{\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \to \sigma''}$$

Definition (5.1; Denotational semantics of statements)

$$\begin{aligned}
\mathfrak{C}[\![\texttt{skip}]\!] &:= \mathrm{id}_\Sigma \\
\mathfrak{C}[\![x \texttt{ := } a]\!]\sigma &:= \sigma[x \mapsto \mathfrak{A}[\![a]\!]\sigma] \\
\mathfrak{C}[\![c_1 \texttt{;} c_2]\!] &:= \mathfrak{C}[\![c_2]\!] \circ \mathfrak{C}[\![c_1]\!] \\
\mathfrak{C}[\![\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2]\!] &:= \mathrm{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c_1]\!], \mathfrak{C}[\![c_2]\!]) \\
\mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!] &:= \mathrm{fix}(\Phi) \text{ where } \Phi(f) := \mathrm{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathrm{id}_\Sigma)
\end{aligned}$$