

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)



noll@cs.rwth-aachen.de

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

Summer Semester 2013

- 1 Recapitulation: CCPOs and Continuous Functions
- 2 The Fixpoint Theorem
- 3 Application to $\text{fix}(\Phi)$
- 4 Summary: Denotational Semantics
- 5 Equivalence of Operational and Denotational Semantics

Goals:

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

Sufficient conditions:

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

on function Φ : **continuity**

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 $\text{graph}(f_\emptyset) = \emptyset$.
- In particular, for every chain $S \subseteq \Sigma \dashrightarrow \Sigma$,

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

Definition (Monotonicity)

Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be partial orders, and let $F : D \rightarrow D'$. 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 $\Phi : (\Sigma \dashrightarrow \Sigma) \rightarrow (\Sigma \dashrightarrow \Sigma)$ with $\Phi(f) := \text{cond}(\mathcal{B}[[b]], f \circ \mathcal{C}[[c]], \text{id}_\Sigma)$. Then Φ is monotonic w.r.t. $(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$.

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 $F : D \rightarrow D'$ 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 $\Phi(f) := \text{cond}(\mathfrak{B} \llbracket b \rrbracket, f \circ \mathfrak{C} \llbracket c \rrbracket, \text{id}_\Sigma)$. Then Φ is continuous w.r.t. $(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$.

- 1 Recapitulation: CCPOs and Continuous Functions
- 2 The Fixpoint Theorem
- 3 Application to $\text{fix}(\Phi)$
- 4 Summary: Denotational Semantics
- 5 Equivalence of Operational and Denotational Semantics

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 $F : D \rightarrow D$ continuous. Then

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

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



Example 7.2

- **Domain:** $(2^{\mathbb{N}}, \subseteq)$ (CCPO with $\bigsqcup S = \bigcup_{N \in S} N$ – see Ex. 6.7)
- **Function:** $F : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}} : N \mapsto N \cup A$ 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: $F(\bigsqcup S) = F(\bigcup_{N \in S} N) = (\bigcup_{N \in S} N) \cup A = \bigcup_{N \in S} (N \cup A) = \bigcup_{N \in S} F(N) = \bigsqcup F(S)$
- **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:** $F(N) := N \cap A$ $\Rightarrow \text{fix}(F) = \emptyset$

- 1 Recapitulation: CCPOs and Continuous Functions
- 2 The Fixpoint Theorem
- 3 Application to $\text{fix}(\Phi)$
- 4 Summary: Denotational Semantics
- 5 Equivalence of Operational and Denotational Semantics

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

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

Corollary 7.3

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

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

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)



Example 7.4 (Factorial program)

- Let $c \in \text{Cmd}$ be given by

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

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

$$\mathcal{C}[[c]](\sigma_0) = \text{fix}(\Phi)(\sigma_1)$$

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

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

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

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

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

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

Denotational Semantics of Factorial Program II

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

Example 7.4 (Factorial program; continued)

$$\begin{aligned} f_0(\sigma) &:= \Phi^0(f_\emptyset)(\sigma) \\ &= f_\emptyset(\sigma) \\ &= \text{undefined} \\ f_1(\sigma) &:= \Phi^1(f_\emptyset)(\sigma) \\ &= \Phi(f_0)(\sigma) \\ &= \begin{cases} \sigma & \text{if } \sigma(x) = 1 \\ f_0(\sigma') & \text{otherwise} \end{cases} \\ &= \begin{cases} \sigma & \text{if } \sigma(x) = 1 \\ \text{undefined} & \text{otherwise} \end{cases} \\ f_2(\sigma) &:= \Phi^2(f_\emptyset)(\sigma) \\ &= \Phi(f_1)(\sigma) \\ &= \begin{cases} \sigma & \text{if } \sigma(x) = 1 \\ f_1(\sigma') & \text{otherwise} \end{cases} \\ &= \begin{cases} \sigma & \text{if } \sigma(x) = 1 \\ \sigma' & \text{if } \sigma(x) \neq 1 \text{ and } \sigma'(x) = 1 \\ \text{undefined} & \text{if } \sigma(x) \neq 1 \text{ and } \sigma'(x) \neq 1 \end{cases} \\ &= \begin{cases} \sigma & \text{if } \sigma(x) = 1 \\ \sigma' & \text{if } \sigma(x) = 2 \\ \text{undefined} & \text{if } \sigma(x) \neq 1 \text{ and } \sigma(x) \neq 2 \end{cases} \\ &= \begin{cases} \sigma & \text{if } \sigma(x) = 1 \\ \sigma[y \mapsto 2 * \sigma(y), \\ \quad x \mapsto 1] & \text{if } \sigma(x) = 2 \\ \text{undefined} & \text{if } \sigma(x) \neq 1 \\ & \text{and } \sigma(x) \neq 2 \end{cases} \end{aligned}$$

Denotational Semantics of Factorial Program III

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

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

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

Example 7.4 (Factorial program; continued)

- n -th approximation:

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

- Fixpoint:

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

- 1 Recapitulation: CCPOs and Continuous Functions
- 2 The Fixpoint Theorem
- 3 Application to $\text{fix}(\Phi)$
- 4 Summary: Denotational Semantics
- 5 Equivalence of Operational and Denotational Semantics

Summary: Denotational Semantics

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

- 1 Recapitulation: CCPOs and Continuous Functions
- 2 The Fixpoint Theorem
- 3 Application to $\text{fix}(\Phi)$
- 4 Summary: Denotational Semantics
- 5 Equivalence of Operational and Denotational Semantics

Remember: in Def. 4.1, $\mathcal{D}[\cdot] : Cmd \rightarrow (\Sigma \dashrightarrow \Sigma)$ was given by

$$\mathcal{D}[c](\sigma) = \sigma' \iff \langle c, \sigma \rangle \rightarrow \sigma'$$

Theorem 7.5 (Coincidence Theorem)

For every $c \in Cmd$,

$$\mathcal{D}[c] = \mathcal{C}[c],$$

i.e., $\langle c, \sigma \rangle \rightarrow \sigma'$ iff $\mathcal{C}[c](\sigma) = \sigma'$, and thus $\mathcal{D}[\cdot] = \mathcal{C}[\cdot]$.

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



Proof (Theorem 7.5).

We have to show that

$$\langle c, \sigma \rangle \rightarrow \sigma' \iff \mathcal{E}[\![c]\!](\sigma) = \sigma'$$

$\Rightarrow$ by structural induction over the derivation tree of $\langle c, \sigma \rangle \rightarrow \sigma'$

$\Leftarrow$ 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)

$$\begin{array}{lcl}(\text{skip}) \frac{}{\langle \text{skip}, \sigma \rangle \rightarrow \sigma} & (\text{asgn}) \frac{\langle a, \sigma \rangle \rightarrow z}{\langle x := a, \sigma \rangle \rightarrow \sigma[x \mapsto z]} \\(\text{seq}) \frac{\langle c_1, \sigma \rangle \rightarrow \sigma' \quad \langle c_2, \sigma' \rangle \rightarrow \sigma''}{\langle c_1; c_2, \sigma \rangle \rightarrow \sigma''} & (\text{if-t}) \frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c_1, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'} \\(\text{if-f}) \frac{\langle b, \sigma \rangle \rightarrow \text{false} \quad \langle c_2, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_1 \text{ else } c_2, \sigma \rangle \rightarrow \sigma'} & (\text{wh-f}) \frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma} \\(\text{wh-t}) \frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c, \sigma \rangle \rightarrow \sigma' \quad \langle \text{while } b \text{ do } c, \sigma' \rangle \rightarrow \sigma''}{\langle \text{while } b \text{ do } c, \sigma \rangle \rightarrow \sigma''}\end{array}$$

Definition (5.1; Denotational semantics of statements)

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