Semantics and Verification of Software

Lecture 7: Continuous Functions and Fixpoint Theorem

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/svsw08/

Winter semester 2008/09

Outline

Characterization of $\mathsf{fix}(\Phi)$ II

Goals:

- Prove existence of $\mathsf{fix}(\Phi)$ for $\Phi(f) = \mathsf{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma)$
- 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-Complete Partial Orders

Definition (Chain, (least) upper bound)

Let $(D, \sqsubseteq)$ be a partial order and $S \subseteq D$.

1. S is called a chain in D if, for every $s_1, s_2 \in S$,
$$s_1 \sqsubseteq s_2 \text{ or } s_2 \sqsubseteq s_1$$
(that is, S is a totally ordered subset of D).
2. An element $d \in D$ is called an upper bound of S if $s \sqsubseteq d$ for every $s \in S$ (notation: $S \sqsubseteq d$).
3. An upper bound d of S is called least upper bound (LUB) or supremum of S if $d \sqsubseteq d'$ for every upper bound d' of S (notation: $d = \bigsqcup S$).

Definition (Chain completeness)

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

Outline

Monotonicity I

Definition 7.1 (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 $d_1, d_2 \in D$,

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

Monotonicity I

Definition 7.1 (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 $d_1, d_2 \in D$,

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

Interpretation: monotonic functions "preserve information"

Monotonicity I

Definition 7.1 (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 $d_1, d_2 \in D$,

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

Interpretation: monotonic functions "preserve information"

Example 7.2

1. Let $T := \{S \subseteq \mathbb{N} \mid S \text{ finite}\}$. Then $F_1 : T \to \mathbb{N} : S \mapsto \sum_{n \in S} n$ is monotonic w.r.t. $(2^{\mathbb{N}}, \subseteq)$ and $(\mathbb{N}, \leq)$.

Monotonicity I

Definition 7.1 (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 $d_1, d_2 \in D$,

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

Interpretation: monotonic functions "preserve information"

Example 7.2

1. Let $T := \{S \subseteq \mathbb{N} \mid S \text{ finite}\}$. Then $F_1 : T \to \mathbb{N} : S \mapsto \sum_{n \in S} n$ is monotonic w.r.t. $(2^{\mathbb{N}}, \subseteq)$ and $(\mathbb{N}, \leq)$.
2. $F_2 : 2^{\mathbb{N}} \to 2^{\mathbb{N}} : S \mapsto \mathbb{N} \setminus S$ is not monotonic w.r.t. $(2^{\mathbb{N}}, \subseteq)$ (since, e.g., $\emptyset \subseteq \mathbb{N}$ but $F_2(\emptyset) = \mathbb{N} \not\subseteq F_2(\mathbb{N}) = \emptyset$).

Application to fix(Φ) I

Lemma 7.3

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

Application to fix(Φ) I

Lemma 7.3

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

Proof.

on the board □

Monotonicity II

The following lemma states how chains behave under montonic functions.

Lemma 7.4

Let $(D, \sqsubseteq)$ *and* $(D', \sqsubseteq')$ *be CCPOs,* $F : D \to D'$ *monotonic, and* $S \subseteq D$ *a chain in* D*. Then:*

1. $F(S) := \{F(d) \mid d \in S\}$ *is a chain in* D'.
2. $\bigsqcup F(S) \sqsubseteq' F(\bigsqcup S)$.

Monotonicity II

The following lemma states how chains behave under montonic functions.

Lemma 7.4

Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be CCPOs, $F : D \to D'$ monotonic, and $S \subseteq D$ a chain in D. Then:

1. $F(S) := \{F(d) \mid d \in S\}$ *is a chain in* D'.
2. $\bigsqcup F(S) \sqsubseteq' F(\bigsqcup S)$.

Proof.

on the board □

Continuity

A function F is continuous if applying F and taking LUBs can be exchanged

Definition 7.5 (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 $S \subseteq D$,

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

Continuity

A function F is continuous if applying F and taking LUBs can be exchanged

Definition 7.5 (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 7.6

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

Continuity

A function F is continuous if applying F and taking LUBs can be exchanged

Definition 7.5 (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 7.6

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

Proof.

on the board □

Outline

The Fixpoint Theorem

Theorem 7.7 (Fixpoint Theorem by Tarski and Knaster)

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

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

is the least fixpoint of F where

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

The Fixpoint Theorem

Theorem 7.7 (Fixpoint Theorem by Tarski and Knaster)

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

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

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 □

Application to fix(Φ) II

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

Corollary 7.8

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 fix(Φ) II

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

Corollary 7.8

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

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

Proof.

Using

- Lemma 6.9 ($(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$ CCPO with least element $f_\emptyset$; LUB = union of graphs)
- Lemma 7.6 (Φ continuous)
- Theorem 7.7 (Fixpoint Theorem)

□

Outline

Denotational Semantics of Factorial Program I

Example 7.9 (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 7.9 (Factorial program)

- Let $c \in \mathit{Cmd}$ 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. 4.6 yields:

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

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

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

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

Denotational Semantics of Factorial Program I

Example 7.9 (Factorial program)

- Let $c \in \mathit{Cmd}$ 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. 4.6 yields:
  $$\mathfrak{C}[\![c]\!](\sigma_0) = \mathsf{fix}(\Phi)(\sigma_1)$$
  where $\sigma_1 := \sigma_0[\texttt{y} \mapsto 1]$ and, for every $f : \Sigma \dashrightarrow \Sigma$ and $\sigma \in \Sigma$,
  $$\begin{aligned}\Phi(f)(\sigma) &= \mathsf{cond}(\mathfrak{B}[\![\neg\texttt{(x=1)}]\!], f \circ \mathfrak{C}[\![\texttt{y:=y*x; x:=x-1}]\!], \mathsf{id}_\Sigma)(\sigma)\\ &= \begin{cases}\sigma & \text{if } \sigma(\texttt{x}) = 1\\ f(\sigma') & \text{otherwise}\end{cases}\end{aligned}$$
  with $\sigma' := \sigma[\texttt{y} \mapsto \sigma(\texttt{y}) * \sigma(\texttt{x}), \texttt{x} \mapsto \sigma(\texttt{x}) - 1]$.
- Approximations of least fixpoint of Φ according to Theorem 7.7:
  $$\mathsf{fix}(\Phi) = \bigsqcup\{\Phi^n(f_\emptyset) \mid n \in \mathbb{N}\}$$
  (where $\mathsf{graph}(f_\emptyset) = \emptyset$)

Denotational Semantics of Factorial Program II

Example 7.9 (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

Example 7.9 (Factorial program; continued)

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

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

Denotational Semantics of Factorial Program II

Example 7.9 (Factorial program; continued)

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

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

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

Example 7.9 (Factorial program; continued)

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

Example 7.9 (Factorial program; continued)

- n-th approximation:

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

Example 7.9 (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:

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

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 $\mathfrak{C}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma)$
- 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