Semantics and Verification of Software

Lecture 7: Denotational Semantics

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

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

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.

Continuous Functions

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 $d_1, d_2 \in D$,

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

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 $S \subseteq D$,

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

Outline

Theorem 7.1 (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 7.2

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 5.12
  ($(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$ CCPO with least element $f_\emptyset$; LUB = union of graphs)
- Lemma 6.7 (Φ continuous)
- Theorem 7.1 (Fixpoint Theorem)

□

Outline

Denotational Semantics of Factorial Program I

Example 7.3 (Factorial program)

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

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

- For every initial state $\sigma_0 \in \Sigma$, Def. 4.8 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 $\sigma \in \Sigma$,

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

  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

Example 7.3 (Factorial program; continued)

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

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

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

Denotational Semantics of Factorial Program III

Example 7.3 (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 7.3 (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

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

Outline

Remember: in Def. 4.3, $\mathfrak{O}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma)$ was given by

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

Theorem 7.4 (Coincidence Theorem)

For every $c \in \mathit{Cmd}$,

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

i.e., $\mathfrak{O}[\![.]\!] = \mathfrak{C}[\![.]\!]$.

Equivalence of Semantics II

The proof of Theorem 7.4 employs the following auxiliary propositions:

Lemma 7.5

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* $t \in \mathbb{B}$*:*

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

Proof.

1. see Exercise 3.2
2. analogously

□

Equivalence of Semantics III

Proof (Theorem 7.4).

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) □

Reminder: Operational/Denotational Semantics

Definition (Operational semantics of statements)

Execution relation $\langle c, \sigma \rangle \to \sigma'$:

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

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

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

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

Definition (Denotational semantics of statements)

Denotational semantic functional for statements $\mathfrak{C}[\![.]\!] : Cmd \to (\Sigma \dashrightarrow \Sigma)$:

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

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