Semantics and Verification of Software

Lecture 6: Chain-Complete Partial Orders

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

Additional Literature

- G. Winskel: *The Formal Semantics of Programming Languages*, The MIT Press, 1996
  (Chapter 5; notations somewhat different)
- H.R. Nielson, F. Nielson: *Semantics with Applications: A Formal Introduction*, Wiley, 1992
  (Chapter 4;
  `http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.html`)
- J.E. Stoy: *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*, The MIT Press, 1977
  (very comprehensive but a bit outdated)

Semantics of Statements

Definition (Denotational semantics of statements)

The (denotational) semantic functional for statements,

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

is given by:

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

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

Why Fixpoints?

- Goal: preserve validity of equivalence
$$\mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!] \overset{(*)}{=} \mathfrak{C}[\![\texttt{if } b \texttt{ then } (c\texttt{;while } b \texttt{ do } c) \texttt{ else skip}]\!]$$
(cf. Lemma 4.3)
- Using the known parts of Def. 4.6, we obtain:
$$\begin{aligned} &\mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!] \\ \overset{(*)}{=}\ & \mathfrak{C}[\![\texttt{if } b \texttt{ then } (c\texttt{;while } b \texttt{ do } c) \texttt{ else skip}]\!] \\ \overset{\text{Def. 4.6}}{=}\ & \mathsf{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c\texttt{;while } b \texttt{ do } c]\!], \mathfrak{C}[\![\texttt{skip}]\!]) \\ \overset{\text{Def. 4.6}}{=}\ & \mathsf{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!] \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma) \end{aligned}$$
- Abbreviating $f := \mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!]$ this yields:
$$f = \mathsf{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma)$$
- Hence f must be a solution of this recursive equation
- In other words: f must be a fixpoint of the mapping
$$\Phi : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto \mathsf{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma)$$
(since the equation can be stated as $f = \Phi(f)$)

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

For $\Phi(f_0) = f_0$ and initial state $\sigma_0 \in \Sigma$, case distinction yields:

1. Loop $\texttt{while}\ b\ \texttt{do}\ c$ terminates after n iterations ($n \in \mathbb{N}$)
   $\implies f_0(\sigma_0) = \sigma_n$
2. Body c diverges in the nth iteration
   $\implies f_0(\sigma_0) = \text{undefined}$
3. Loop $\texttt{while}\ b\ \texttt{do}\ c$ diverges
   $\implies$ no condition on f_0 (only $f_0(\sigma_0) = f_0(\sigma_i)$ for every $i \in \mathbb{N}$)

- Not surprising since, e.g., the loop `while true do skip` yields for every $f : \Sigma \dashrightarrow \Sigma$:
$$\Phi(f) = \mathsf{cond}(\mathfrak{B}[\![\mathsf{true}]\!], f \circ \mathfrak{C}[\![\texttt{skip}]\!], \mathsf{id}_\Sigma) = f$$
- On the other hand, our operational understanding requires, for every $\sigma_0 \in \Sigma$,
$$\mathfrak{C}[\![\texttt{while true do skip}]\!]\sigma_0 = \text{undefined}$$

Conclusion

$\mathsf{fix}(\Phi)$ is the least defined fixpoint of Φ.

Making it Precise

To use fixpoint theory, the notion of "least defined" has to be made precise.

- Given $f, g : \Sigma \dashrightarrow \Sigma$, let

$$f \sqsubseteq g \iff \text{for every } \sigma, \sigma' \in \Sigma : f(\sigma) = \sigma' \implies g(\sigma) = \sigma'$$

  (g is "at least as defined" as f)
- Equivalent to requiring

$$\mathsf{graph}(f) \subseteq \mathsf{graph}(g)$$

  where

$$\mathsf{graph}(h) := \{(\sigma, \sigma') \mid \sigma \in \Sigma, \sigma' = h(\sigma) \text{ defined}\} \subseteq \Sigma \times \Sigma$$

  for every $h : \Sigma \dashrightarrow \Sigma$

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

Outline

Partial Orders

Definition 6.1 (Partial order)

A partial order (PO) $(D, \sqsubseteq)$ consists of a set D, called domain, and of a relation $\sqsubseteq \subseteq D \times D$ such that, for every $d_1, d_2, d_3 \in D$,

reflexivity: $d_1 \sqsubseteq d_1$

transitivity: $d_1 \sqsubseteq d_2$ and $d_2 \sqsubseteq d_3 \implies d_1 \sqsubseteq d_3$

antisymmetry: $d_1 \sqsubseteq d_2$ and $d_2 \sqsubseteq d_1 \implies d_1 = d_2$

It is called total if, in addition, always $d_1 \sqsubseteq d_2$ or $d_2 \sqsubseteq d_1$.

Example 6.2

1. $(\mathbb{N}, \leq)$ is a total partial order
2. $(2^{\mathbb{N}}, \subseteq)$ is a (non-total) partial order
3. $(\mathbb{N}, <)$ is not a partial order (since not reflexive)

Application to fix(Φ) I

Lemma 6.3

$(\Sigma \dashrightarrow \Sigma, \sqsubseteq)$ *is a partial order.*

Proof.

see Exercise 3 $\square$

Chains and Least Upper Bounds

Definition 6.4 (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: $d = \bigsqcup S$).

Example 6.5

1. Every subset $S \subseteq \mathbb{N}$ is a chain in $(\mathbb{N}, \leq)$.
It has a LUB (its greatest element) iff it is finite.
2. $\{\emptyset, \{0\}, \{0, 1\}, \dots\}$ is a chain in $(2^{\mathbb{N}}, \subseteq)$ with LUB $\mathbb{N}$.

Chain Completeness

Definition 6.6 (Chain completeness)

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

Example 6.7

1. $(2^{\mathbb{N}}, \subseteq)$ is a CCPO with $\bigsqcup S = \bigcup_{M \in S} M$ for every chain $S \subseteq 2^{\mathbb{N}}$.
2. $(\mathbb{N}, \leq)$ is not chain complete
   (since, e.g., the chain $\mathbb{N}$ has no upper bound).

Least Elements in CCPOs

Corollary 6.8

Every CCPO has a least element $\bigsqcup \emptyset$.

Proof.

Let $(D, \sqsubseteq)$ be a CCPO.

- By definition, $\emptyset$ is a chain in D.
- By definition, every $d \in D$ is an upper bound of $\emptyset$.
- Thus $\bigsqcup \emptyset$ exists and is the least element of D.

□

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

Lemma 6.9

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

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

Proof.

on the board □