Semantics and Verification of Software

Summer Semester 2015

Lecture 6: Denotational Semantics of WHILE I (The Approach)

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

http://moves.rwth-aachen.de/teaching/ss-15/sv-sw/

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Outline of Lecture 6

Denotational Semantics of WHILE

- Primary aspect of a program: its "effect", i.e., input/output behavior

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Denotational Semantics of WHILE

- Primary aspect of a program: its “effect”, i.e., input/output behavior
- In operational semantics: indirect definition of semantic functional

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

by execution relation

Denotational Semantics of WHILE

- Primary aspect of a program: its “effect”, i.e., input/output behavior
- In operational semantics: indirect definition of semantic functional

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

  by execution relation
- Now: abstract from operational details
- Denotational semantics: direct definition of program effect by induction on its syntactic structure

Outline of Lecture 6

Semantics of Arithmetic Expressions

Again: value of an expression determined by current state

Definition 6.1 (Denotational semantics of arithmetic expressions)

The (denotational) semantic functional for arithmetic expressions,

$$\mathfrak{A}[\![.]\!] : AExp \to (\Sigma \to \mathbb{Z}),$$

is given by:

$$\begin{aligned} \mathfrak{A}[\![z]\!]\sigma &:= z & \qquad \mathfrak{A}[\![a_1\texttt{+}a_2]\!]\sigma &:= \mathfrak{A}[\![a_1]\!]\sigma + \mathfrak{A}[\![a_2]\!]\sigma \\ \mathfrak{A}[\![x]\!]\sigma &:= \sigma(x) & \qquad \mathfrak{A}[\![a_1\texttt{-}a_2]\!]\sigma &:= \mathfrak{A}[\![a_1]\!]\sigma - \mathfrak{A}[\![a_2]\!]\sigma \\ & & \qquad \mathfrak{A}[\![a_1\texttt{*}a_2]\!]\sigma &:= \mathfrak{A}[\![a_1]\!]\sigma \cdot \mathfrak{A}[\![a_2]\!]\sigma \end{aligned}$$

Semantics of Boolean Expressions

Definition 6.2 (Denotational semantics of Boolean expressions)

The (denotational) semantic functional for Boolean expressions is given by $\mathfrak{B}[\![.]\!] : BExp \to (\Sigma \to \mathbb{B})$ where

$$
\begin{aligned}
\mathfrak{B}[\![t]\!]\sigma &:= t \\
\mathfrak{B}[\![a_1 = a_2]\!]\sigma &:= \begin{cases} \text{true} & \text{if } \mathfrak{A}[\![a_1]\!]\sigma = \mathfrak{A}[\![a_2]\!]\sigma \\ \text{false} & \text{otherwise} \end{cases} \\
\mathfrak{B}[\![a_1 > a_2]\!]\sigma &:= \begin{cases} \text{true} & \text{if } \mathfrak{A}[\![a_1]\!]\sigma > \mathfrak{A}[\![a_2]\!]\sigma \\ \text{false} & \text{otherwise} \end{cases} \\
\mathfrak{B}[\![\neg b]\!]\sigma &:= \begin{cases} \text{true} & \text{if } \mathfrak{B}[\![b]\!]\sigma = \text{false} \\ \text{false} & \text{otherwise} \end{cases} \\
\mathfrak{B}[\![b_1 \wedge b_2]\!]\sigma &:= \begin{cases} \text{true} & \text{if } \mathfrak{B}[\![b_1]\!]\sigma = \mathfrak{B}[\![b_2]\!]\sigma = \text{true} \\ \text{false} & \text{otherwise} \end{cases} \\
\mathfrak{B}[\![b_1 \vee b_2]\!]\sigma &:= \begin{cases} \text{false} & \text{if } \mathfrak{B}[\![b_1]\!]\sigma = \mathfrak{B}[\![b_2]\!]\sigma = \text{false} \\ \text{true} & \text{otherwise} \end{cases}
\end{aligned}
$$

Outline of Lecture 6

Denotational Semantics of Statements

The Goal

- Now: semantic functional

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

Denotational Semantics of Statements

The Goal

- Now: semantic functional

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

- Same type as operational functional

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

(in fact, both will turn out to be the same
$\Rightarrow$ equivalence of operational and denotational semantics)

Denotational Semantics of Statements

Auxiliary Functions

Inductive definition of $\mathfrak{C}[\![.]\!]$ employs following auxiliary functions:

- identity on states: $\mathrm{id}_\Sigma : \Sigma \dashrightarrow \Sigma : \sigma \mapsto \sigma$

Denotational Semantics of Statements

Auxiliary Functions

Inductive definition of $\mathfrak{C}[\![.]\!]$ employs following auxiliary functions:

- identity on states: $\mathrm{id}_\Sigma : \Sigma \dashrightarrow \Sigma : \sigma \mapsto \sigma$
- (strict) composition of partial state transformations:

$$\circ : (\Sigma \dashrightarrow \Sigma) \times (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma)$$

where, for every $f, g : \Sigma \dashrightarrow \Sigma$ and $\sigma \in \Sigma$,

$$(g \circ f)(\sigma) := \begin{cases} g(f(\sigma)) & \text{if } f(\sigma) \text{ defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Auxiliary Functions

Inductive definition of $\mathfrak{C}[\![.]\!]$ employs following auxiliary functions:

- identity on states: $\mathrm{id}_\Sigma : \Sigma \dashrightarrow \Sigma : \sigma \mapsto \sigma$
- (strict) composition of partial state transformations:

$$\circ : (\Sigma \dashrightarrow \Sigma) \times (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma)$$

  where, for every $f, g : \Sigma \dashrightarrow \Sigma$ and $\sigma \in \Sigma$,

$$(g \circ f)(\sigma) := \begin{cases} g(f(\sigma)) & \text{if } f(\sigma) \text{ defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

- semantic conditional:

$$\mathrm{cond} : (\Sigma \to \mathbb{B}) \times (\Sigma \dashrightarrow \Sigma) \times (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma)$$

  where, for every $p : \Sigma \to \mathbb{B}$, $f, g : \Sigma \dashrightarrow \Sigma$, and $\sigma \in \Sigma$,

$$\mathrm{cond}(p, f, g)(\sigma) := \begin{cases} f(\sigma) & \text{if } p(\sigma) = \mathrm{true} \\ g(\sigma) & \text{otherwise} \end{cases}$$

Semantics of Statements I

Definition 6.3 (Denotational semantics of statements)

The (denotational) semantic functional for statements,

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

is given by:

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

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

Semantics of Statements II

Remarks:

- Definition of $\mathfrak{C}[\![c]\!]$ given by induction on syntactic structure of $c \in Cmd$
  - in particular, $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]$ only refers to $\mathfrak{B}[\![b]\!]$ and $\mathfrak{C}[\![c]\!]$ (and not to $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]$ again)
  - note difference to $\mathfrak{O}[\![c]\!]$:

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

Semantics of Statements II

Remarks:

- Definition of $\mathfrak{C}[\![c]\!]$ given by induction on syntactic structure of $c \in Cmd$
  - in particular, $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]$ only refers to $\mathfrak{B}[\![b]\!]$ and $\mathfrak{C}[\![c]\!]$ (and not to $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]$ again)
  - note difference to $\mathfrak{O}[\![c]\!]$:

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

- In $\mathfrak{C}[\![c_1\,;c_2]\!] := \mathfrak{C}[\![c_2]\!] \circ \mathfrak{C}[\![c_1]\!]$, function composition $\circ$ has to be strict since non-termination of c_1 implies non-termination of $c_1\,;c_2$

Semantics of Statements II

Remarks:

- Definition of $\mathfrak{C}[\![c]\!]$ given by induction on syntactic structure of $c \in Cmd$
  - in particular, $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]$ only refers to $\mathfrak{B}[\![b]\!]$ and $\mathfrak{C}[\![c]\!]$ (and not to $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]$ again)
  - note difference to $\mathfrak{O}[\![c]\!]$:

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

- In $\mathfrak{C}[\![c_1\,;c_2]\!] := \mathfrak{C}[\![c_2]\!] \circ \mathfrak{C}[\![c_1]\!]$, function composition $\circ$ has to be strict since non-termination of c_1 implies non-termination of $c_1\,;c_2$
- In $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] := \mathsf{fix}(\Phi)$, fix denotes a fixpoint operator (which remains to be defined)
  $\Rightarrow$ "fixpoint semantics"

Semantics of Statements II

Remarks:

- Definition of $\mathfrak{C}[\![c]\!]$ given by induction on syntactic structure of $c \in Cmd$
  - in particular, $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]$ only refers to $\mathfrak{B}[\![b]\!]$ and $\mathfrak{C}[\![c]\!]$ (and not to $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]$ again)
  - note difference to $\mathfrak{O}[\![c]\!]$:

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

- In $\mathfrak{C}[\![c_1 ; c_2]\!] := \mathfrak{C}[\![c_2]\!] \circ \mathfrak{C}[\![c_1]\!]$, function composition $\circ$ has to be strict since non-termination of c_1 implies non-termination of $c_1 ; c_2$
- In $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] := \mathsf{fix}(\Phi)$, fix denotes a fixpoint operator (which remains to be defined)
  $\Rightarrow$ “fixpoint semantics”

But: why fixpoints?

Denotational Semantics of Statements

Why Fixpoints?

- Goal: preserve validity of equivalence

$$\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] \stackrel{(*)}{=} \mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ c;\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \texttt{else}\ \texttt{skip}\ \texttt{end}]\!]$$

  (cf. Lemma 4.3)

Why Fixpoints?

- Goal: preserve validity of equivalence

$$\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] \overset{(*)}{=} \mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ c;\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \texttt{else}\ \texttt{skip}\ \texttt{end}]\!]$$

  (cf. Lemma 4.3)
- Using the known parts of Definition 6.3, we obtain:

$$\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]$$

Denotational Semantics of Statements

Why Fixpoints?

- Goal: preserve validity of equivalence
$$\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] \stackrel{(*)}{=} \mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ c\texttt{;while}\ b\ \texttt{do}\ c\ \texttt{end}\ \texttt{else}\ \texttt{skip}\ \texttt{end}]\!]$$
(cf. Lemma 4.3)
- Using the known parts of Definition 6.3, we obtain:
$$\begin{aligned}&\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]\\ \stackrel{(*)}{=}\quad &\mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ c\texttt{;while}\ b\ \texttt{do}\ c\ \texttt{end}\ \texttt{else}\ \texttt{skip}\ \texttt{end}]\!]\end{aligned}$$

Why Fixpoints?

- Goal: preserve validity of equivalence

$$\mathfrak{C}[\![\texttt{while } b \texttt{ do } c \texttt{ end}]\!] \overset{(*)}{=} \mathfrak{C}[\![\texttt{if } b \texttt{ then } c\texttt{;while } b \texttt{ do } c \texttt{ end else skip end}]\!]$$

  (cf. Lemma 4.3)
- Using the known parts of Definition 6.3, we obtain:

$$\begin{aligned}
&\mathfrak{C}[\![\texttt{while } b \texttt{ do } c \texttt{ end}]\!] \\
\overset{(*)}{=}\ & \mathfrak{C}[\![\texttt{if } b \texttt{ then } c\texttt{;while } b \texttt{ do } c \texttt{ end else skip end}]\!] \\
\overset{\text{Def. 6.3}}{=}\ & \text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c\texttt{;while } b \texttt{ do } c \texttt{ end}]\!], \mathfrak{C}[\![\texttt{skip}]\!])
\end{aligned}$$

Why Fixpoints?

- Goal: preserve validity of equivalence

$$\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] \stackrel{(*)}{=} \mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ c;\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \texttt{else}\ \texttt{skip}\ \texttt{end}]\!]$$

  (cf. Lemma 4.3)
- Using the known parts of Definition 6.3, we obtain:

$$\begin{aligned} &\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] \\ \stackrel{(*)}{=}\ &\mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ c;\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}\ \texttt{else}\ \texttt{skip}\ \texttt{end}]\!] \\ \stackrel{\text{Def. 6.3}}{=}\ &\mathrm{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c;\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!], \mathfrak{C}[\![\texttt{skip}]\!]) \\ \stackrel{\text{Def. 6.3}}{=}\ &\mathrm{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] \circ \mathfrak{C}[\![c]\!], \mathrm{id}_\Sigma) \end{aligned}$$

Why Fixpoints?

- Goal: preserve validity of equivalence
$$\mathfrak{C}[\![\texttt{while } b \texttt{ do } c \texttt{ end}]\!] \overset{(*)}{=} \mathfrak{C}[\![\texttt{if } b \texttt{ then } c\texttt{;while } b \texttt{ do } c \texttt{ end else skip end}]\!]$$
(cf. Lemma 4.3)
- Using the known parts of Definition 6.3, we obtain:
$$\begin{aligned} &\mathfrak{C}[\![\texttt{while } b \texttt{ do } c \texttt{ end}]\!] \\ \overset{(*)}{=} \quad & \mathfrak{C}[\![\texttt{if } b \texttt{ then } c\texttt{;while } b \texttt{ do } c \texttt{ end else skip end}]\!] \\ \overset{\text{Def. 6.3}}{=} \quad & \text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c\texttt{;while } b \texttt{ do } c \texttt{ end}]\!], \mathfrak{C}[\![\texttt{skip}]\!]) \\ \overset{\text{Def. 6.3}}{=} \quad & \text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![\texttt{while } b \texttt{ do } c \texttt{ end}]\!] \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma) \end{aligned}$$
- Abbreviating $f := \mathfrak{C}[\![\texttt{while } b \texttt{ do } c \texttt{ end}]\!]$ this yields:
$$f = \text{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma)$$

Denotational Semantics of Statements

Why Fixpoints?

- Goal: preserve validity of equivalence
$$\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] \stackrel{(*)}{=} \mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ c\texttt{;while}\ b\ \texttt{do}\ c\ \texttt{end else skip end}]\!]$$
(cf. Lemma 4.3)
- Using the known parts of Definition 6.3, we obtain:
$$\begin{aligned} &\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] \\ \stackrel{(*)}{=}\ & \mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ c\texttt{;while}\ b\ \texttt{do}\ c\ \texttt{end else skip end}]\!] \\ \stackrel{\text{Def. 6.3}}{=}\ & \mathrm{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c\texttt{;while}\ b\ \texttt{do}\ c\ \texttt{end}]\!], \mathfrak{C}[\![\texttt{skip}]\!]) \\ \stackrel{\text{Def. 6.3}}{=}\ & \mathrm{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] \circ \mathfrak{C}[\![c]\!], \mathrm{id}_\Sigma) \end{aligned}$$
- Abbreviating $f := \mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]$ this yields:
$$f = \mathrm{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathrm{id}_\Sigma)$$
- Hence f must be a solution of this recursive equation

Why Fixpoints?

- Goal: preserve validity of equivalence

$$\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] \stackrel{(*)}{=} \mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ c\texttt{;while}\ b\ \texttt{do}\ c\ \texttt{end else skip end}]\!]$$

  (cf. Lemma 4.3)
- Using the known parts of Definition 6.3, we obtain:

$$\begin{aligned}
&\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]\\
\stackrel{(*)}{=}\ &\mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ c\texttt{;while}\ b\ \texttt{do}\ c\ \texttt{end else skip end}]\!]\\
\stackrel{\text{Def. 6.3}}{=}\ &\text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c\texttt{;while}\ b\ \texttt{do}\ c\ \texttt{end}]\!], \mathfrak{C}[\![\texttt{skip}]\!])\\
\stackrel{\text{Def. 6.3}}{=}\ &\text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!] \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma)
\end{aligned}$$

- Abbreviating $f := \mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]$ this yields:

$$f = \text{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma)$$

- Hence f must be a solution of this recursive equation
- In other words: f must be a fixpoint of the mapping

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

  (since the equation can be stated as $f = \Phi(f)$)

Well-Definedness of Fixpoint Semantics

But: fixpoint property not sufficient to obtain a well-defined semantics

Well-Definedness of Fixpoint Semantics

But: fixpoint property not sufficient to obtain a well-defined semantics

Potential problems:

Existence: there does not need to exist any fixpoint. Examples:

1. $\phi_1 : \mathbb{N} \to \mathbb{N} : n \mapsto n + 1$ has no fixpoint
2. $\Phi_1 : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto \begin{cases} g_1 & \text{if } f = g_2 \\ g_2 & \text{otherwise} \end{cases}$

   has no fixpoint if $g_1 \neq g_2$

Denotational Semantics of Statements

Well-Definedness of Fixpoint Semantics

But: fixpoint property not sufficient to obtain a well-defined semantics

Potential problems:

Existence: there does not need to exist any fixpoint. Examples:

1. $\phi_1 : \mathbb{N} \to \mathbb{N} : n \mapsto n + 1$ has no fixpoint
2. $\Phi_1 : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto \begin{cases} g_1 & \text{if } f = g_2 \\ g_2 & \text{otherwise} \end{cases}$

   has no fixpoint if $g_1 \neq g_2$

Solution: in our setting, fixpoints always exist

Well-Definedness of Fixpoint Semantics

But: fixpoint property not sufficient to obtain a well-defined semantics

Potential problems:

Existence: there does not need to exist any fixpoint. Examples:

1. $\phi_1 : \mathbb{N} \to \mathbb{N} : n \mapsto n + 1$ has no fixpoint
2. $\Phi_1 : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto \begin{cases} g_1 & \text{if } f = g_2 \\ g_2 & \text{otherwise} \end{cases}$

   has no fixpoint if $g_1 \neq g_2$

Solution: in our setting, fixpoints always exist

Uniqueness: there might exist several fixpoints. Examples:

1. $\phi_2 : \mathbb{N} \to \mathbb{N} : n \mapsto n^3$ has fixpoints $\{0, 1\}$
2. every state transformation f is a fixpoint of $\Phi_2 : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto f$

Well-Definedness of Fixpoint Semantics

But: fixpoint property not sufficient to obtain a well-defined semantics

Potential problems:

Existence: there does not need to exist any fixpoint. Examples:

1. $\phi_1 : \mathbb{N} \to \mathbb{N} : n \mapsto n + 1$ has no fixpoint
2. $\Phi_1 : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto \begin{cases} g_1 & \text{if } f = g_2 \\ g_2 & \text{otherwise} \end{cases}$

   has no fixpoint if $g_1 \neq g_2$

Solution: in our setting, fixpoints always exist

Uniqueness: there might exist several fixpoints. Examples:

1. $\phi_2 : \mathbb{N} \to \mathbb{N} : n \mapsto n^3$ has fixpoints $\{0, 1\}$
2. every state transformation f is a fixpoint of $\Phi_2 : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto f$

Solution: uniqueness guaranteed by choosing a special fixpoint

Outline of Lecture 6

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

- Let $b \in \mathit{BExp}$ and $c \in \mathit{Cmd}$

Characterisation of fix(Φ)

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

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

- Let $b \in BExp$ and $c \in Cmd$
- Let $\Phi(f) := \mathsf{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathsf{id}_\Sigma)$
- Let $f_0 : \Sigma \dashrightarrow \Sigma$ be a fixpoint of Φ, i.e., $\Phi(f_0) = f_0$

Characterisation of $\mathrm{fix}(\Phi)$

- Let $b \in BExp$ and $c \in Cmd$
- Let $\Phi(f) := \mathrm{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \mathrm{id}_\Sigma)$
- Let $f_0 : \Sigma \dashrightarrow \Sigma$ be a fixpoint of Φ, i.e., $\Phi(f_0) = f_0$
- Given some initial state $\sigma_0 \in \Sigma$, we will distinguish the following cases:
  1. loop `while` b `do` c `end` terminates after n iterations ($n \in \mathbb{N}$)
  2. body c diverges in the n-th iteration (as it contains a non-terminating `while` statement)
  3. loop `while` b `do` c `end` itself diverges

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

Case 1: Termination of Loop

- Loop `while` b `do` c `end` terminates after n iterations ($n \in \mathbb{N}$)

Characterisation of $\mathrm{fix}(\Phi)$

Case 1: Termination of Loop

- Loop `while` b `do` c `end` terminates after n iterations ($n \in \mathbb{N}$)
- Formally: there exist $\sigma_1, \ldots, \sigma_n \in \Sigma$ such that

$$\mathfrak{B}[\![b]\!]\sigma_i = \begin{cases} \text{true} & \text{if } 0 \leq i < n \\ \text{false} & \text{if } i = n \end{cases} \quad \text{and}$$

$$\mathfrak{C}[\![c]\!]\sigma_i = \sigma_{i+1} \quad \text{for every } 0 \leq i < n$$

Case 1: Termination of Loop

- Loop `while` b `do` c `end` terminates after n iterations ($n \in \mathbb{N}$)
- Formally: there exist $\sigma_1, \ldots, \sigma_n \in \Sigma$ such that

$$\begin{aligned} \mathfrak{B}[\![b]\!]\sigma_i &= \begin{cases} \text{true} & \text{if } 0 \leq i < n \\ \text{false} & \text{if } i = n \end{cases} && \text{and} \\ \mathfrak{C}[\![c]\!]\sigma_i &= \sigma_{i+1} && \text{for every } 0 \leq i < n \end{aligned}$$

- Now the definition $\Phi(f) := \text{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma)$ implies, for every $0 \leq i < n$,

$$\begin{aligned} \Phi(f_0)(\sigma_i) &= (f_0 \circ \mathfrak{C}[\![c]\!])(\sigma_i) && \text{since } \mathfrak{B}[\![b]\!]\sigma_i = \text{true} \\ &= f_0(\sigma_{i+1}) && \text{and} \\ \Phi(f_0)(\sigma_n) &= \sigma_n && \text{since } \mathfrak{B}[\![b]\!]\sigma_n = \text{false} \end{aligned}$$

Case 1: Termination of Loop

- Loop `while` b `do` c `end` terminates after n iterations ($n \in \mathbb{N}$)
- Formally: there exist $\sigma_1, \ldots, \sigma_n \in \Sigma$ such that

$$\mathfrak{B}[\![b]\!]\sigma_i = \begin{cases} \text{true} & \text{if } 0 \leq i < n \\ \text{false} & \text{if } i = n \end{cases} \quad \text{and}$$
$$\mathfrak{C}[\![c]\!]\sigma_i = \sigma_{i+1} \qquad \text{for every } 0 \leq i < n$$

- Now the definition $\Phi(f) := \text{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma)$ implies, for every $0 \leq i < n$,

$$\begin{aligned} \Phi(f_0)(\sigma_i) &= (f_0 \circ \mathfrak{C}[\![c]\!])(\sigma_i) && \text{since } \mathfrak{B}[\![b]\!]\sigma_i = \text{true} \\ &= f_0(\sigma_{i+1}) && \text{and} \\ \Phi(f_0)(\sigma_n) &= \sigma_n && \text{since } \mathfrak{B}[\![b]\!]\sigma_n = \text{false} \end{aligned}$$

- Since $\Phi(f_0) = f_0$ it follows that

$$f_0(\sigma_i) = \begin{cases} f_0(\sigma_{i+1}) & \text{if } 0 \leq i < n \\ \sigma_n & \text{if } i = n \end{cases}$$

  and hence

$$f_0(\sigma_0) = f_0(\sigma_1) = \ldots f_0(\sigma_n) = \sigma_n$$

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Characterisation of $\text{fix}(\Phi)$

Case 1: Termination of Loop

- Loop `while` b `do` c `end` terminates after n iterations ($n \in \mathbb{N}$)
- Formally: there exist $\sigma_1, \ldots, \sigma_n \in \Sigma$ such that

$$\begin{aligned} \mathfrak{B}[\![b]\!]\sigma_i &= \begin{cases} \text{true} & \text{if } 0 \leq i < n \\ \text{false} & \text{if } i = n \end{cases} && \text{and} \\ \mathfrak{C}[\![c]\!]\sigma_i &= \sigma_{i+1} && \text{for every } 0 \leq i < n \end{aligned}$$

- Now the definition $\Phi(f) := \text{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma)$ implies, for every $0 \leq i < n$,

$$\begin{aligned} \Phi(f_0)(\sigma_i) &= (f_0 \circ \mathfrak{C}[\![c]\!])(\sigma_i) && \text{since } \mathfrak{B}[\![b]\!]\sigma_i = \text{true} \\ &= f_0(\sigma_{i+1}) && \text{and} \\ \Phi(f_0)(\sigma_n) &= \sigma_n && \text{since } \mathfrak{B}[\![b]\!]\sigma_n = \text{false} \end{aligned}$$

- Since $\Phi(f_0) = f_0$ it follows that

$$f_0(\sigma_i) = \begin{cases} f_0(\sigma_{i+1}) & \text{if } 0 \leq i < n \\ \sigma_n & \text{if } i = n \end{cases}$$

  and hence

$$f_0(\sigma_0) = f_0(\sigma_1) = \ldots f_0(\sigma_n) = \sigma_n$$

$\Rightarrow$ All fixpoints f_0 coincide on σ_0 (with result σ_n)!

Characterisation of $\mathrm{fix}(\Phi)$

Case 2: Divergence of Body

- Body c diverges in the n-th iteration
  (since it contains a non-terminating `while` statement)

Case 2: Divergence of Body

- Body c diverges in the n-th iteration
  (since it contains a non-terminating `while` statement)
- Formally: there exist $\sigma_1, \ldots, \sigma_{n-1} \in \Sigma$ such that

$$\begin{aligned} \mathfrak{B}[\![b]\!]\sigma_i &= \text{true} & \text{for every } 0 \leq i < n \text{ and} \\ \mathfrak{C}[\![c]\!]\sigma_i &= \begin{cases} \sigma_{i+1} & \text{if } 0 \leq i \leq n-2 \\ \text{undefined} & \text{if } i = n-1 \end{cases} \end{aligned}$$

Characterisation of $\mathrm{fix}(\Phi)$

Case 2: Divergence of Body

- Body c diverges in the n-th iteration (since it contains a non-terminating `while` statement)
- Formally: there exist $\sigma_1, \ldots, \sigma_{n-1} \in \Sigma$ such that

$$\mathfrak{B}[\![b]\!]\sigma_i = \mathrm{true} \qquad \text{for every } 0 \leq i < n \text{ and}$$

$$\mathfrak{C}[\![c]\!]\sigma_i = \begin{cases} \sigma_{i+1} & \text{if } 0 \leq i \leq n-2 \\ \text{undefined} & \text{if } i = n-1 \end{cases}$$

- Just as in the previous case (setting $\sigma_n :=$ undefined) it follows that

$$f_0(\sigma_0) = \text{undefined}$$

Case 2: Divergence of Body

- Body c diverges in the n-th iteration
  (since it contains a non-terminating `while` statement)
- Formally: there exist $\sigma_1, \ldots, \sigma_{n-1} \in \Sigma$ such that

$$\mathfrak{B}[\![b]\!]\sigma_i = \text{true} \qquad \text{for every } 0 \leq i < n \text{ and}$$
$$\mathfrak{C}[\![c]\!]\sigma_i = \begin{cases} \sigma_{i+1} & \text{if } 0 \leq i \leq n-2 \\ \text{undefined} & \text{if } i = n-1 \end{cases}$$

- Just as in the previous case (setting $\sigma_n :=$ undefined) it follows that

$$f_0(\sigma_0) = \text{undefined}$$

$\Rightarrow$ Again all fixpoints f_0 coincide on σ_0 (with undefined result)!

Characterisation of $\text{fix}(\Phi)$

Case 3: Divergence of Loop

- Loop `while` b `do` c `end` diverges

Software Modeling
and Verification Chair
RWTH AACHEN UNIVERSITY

Characterisation of $\mathrm{fix}(\Phi)$

Case 3: Divergence of Loop

- Loop `while` b `do` c `end` diverges
- Formally: there exist $\sigma_1, \sigma_2, \ldots \in \Sigma$ such that

$$\begin{aligned} \mathfrak{B}[\![b]\!]\sigma_i &= \mathrm{true} && \text{and} \\ \mathfrak{C}[\![c]\!]\sigma_i &= \sigma_{i+1} && \text{for every } i \in \mathbb{N} \end{aligned}$$

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

Case 3: Divergence of Loop

- Loop `while` b `do` c `end` diverges
- Formally: there exist $\sigma_1, \sigma_2, \ldots \in \Sigma$ such that

$$\begin{aligned} \mathfrak{B}[\![b]\!]\sigma_i &= \mathrm{true} && \text{and} \\ \mathfrak{C}[\![c]\!]\sigma_i &= \sigma_{i+1} && \text{for every } i \in \mathbb{N} \end{aligned}$$

- Here only derivable:

$$f_0(\sigma_0) = f_0(\sigma_i) \quad \text{for every } i \in \mathbb{N}$$

Case 3: Divergence of Loop

- Loop `while` b `do` c `end` diverges
- Formally: there exist $\sigma_1, \sigma_2, \ldots \in \Sigma$ such that

$$\begin{array}{rl} \mathfrak{B}[\![b]\!]\sigma_i = \text{true} & \text{and} \\ \mathfrak{C}[\![c]\!]\sigma_i = \sigma_{i+1} & \text{for every } i \in \mathbb{N} \end{array}$$

- Here only derivable:

$$f_0(\sigma_0) = f_0(\sigma_i) \quad \text{for every } i \in \mathbb{N}$$

$\Rightarrow$ Value of $f_0(\sigma_0)$ not determined!

Characterisation of $\mathrm{fix}(\Phi)$

Summary

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

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

Characterisation of fix(Φ)

Summary

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

1. Loop `while` b `do` c `end` terminates after n iterations ($n \in \mathbb{N}$)
   $\Rightarrow f_0(\sigma_0) = \sigma_n$
2. Body c diverges in the n-th iteration
   $\Rightarrow f_0(\sigma_0) = \text{undefined}$
3. Loop `while` b `do` c `end` diverges
   $\Rightarrow$ 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., for the loop `while` true `do` `skip` `end` every $f : \Sigma \dashrightarrow \Sigma$ is a fixpoint:

$$\Phi(f) = \text{cond}(\mathfrak{B}[\![\text{true}]\!], f \circ \mathfrak{C}[\![\texttt{skip}]\!], \text{id}_\Sigma) = f$$

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

Summary

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

1. Loop `while` b `do` c `end` terminates after n iterations ($n \in \mathbb{N}$)
   $\Rightarrow f_0(\sigma_0) = \sigma_n$
2. Body c diverges in the n-th iteration
   $\Rightarrow f_0(\sigma_0) = \text{undefined}$
3. Loop `while` b `do` c `end` diverges
   $\Rightarrow$ 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., for the loop `while` true `do skip end` every $f : \Sigma \dashrightarrow \Sigma$ is a fixpoint:
  $$\Phi(f) = \mathsf{cond}(\mathfrak{B}[\![\mathsf{true}]\!], f \circ \mathfrak{C}[\![\mathtt{skip}]\!], \mathsf{id}_\Sigma) = f$$
- On the other hand, our operational understanding requires, for every $\sigma_0 \in \Sigma$,
  $$\mathfrak{C}[\![\mathtt{while}\ \mathsf{true}\ \mathtt{do}\ \mathtt{skip}\ \mathtt{end}]\!]\sigma_0 = \text{undefined}$$

Characterisation of $\mathrm{fix}(\Phi)$

Summary

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

1. Loop `while` b `do` c `end` terminates after n iterations ($n \in \mathbb{N}$)
   $\Rightarrow f_0(\sigma_0) = \sigma_n$
2. Body c diverges in the n-th iteration
   $\Rightarrow f_0(\sigma_0) = \text{undefined}$
3. Loop `while` b `do` c `end` diverges
   $\Rightarrow$ 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., for the loop `while` true `do skip end` every $f : \Sigma \dashrightarrow \Sigma$ is a fixpoint:

$$\Phi(f) = \mathrm{cond}(\mathfrak{B}[\![\mathrm{true}]\!], f \circ \mathfrak{C}[\![\mathtt{skip}]\!], \mathrm{id}_\Sigma) = f$$

- On the other hand, our operational understanding requires, for every $\sigma_0 \in \Sigma$,

$$\mathfrak{C}[\![\mathtt{while}\ \mathrm{true}\ \mathtt{do}\ \mathtt{skip}\ \mathtt{end}]\!]\sigma_0 = \text{undefined}$$

Conclusion

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

Outline of Lecture 6

Making It Precise I

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

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

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

  (g is "at least as defined" as f)

Making It Precise I

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

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

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

  (g is “at least as defined” as f)

- Equivalent to requiring

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

  where

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

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

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Making It Precise II

Example 6.4

Let $x \in Var$ be fixed, and let $f_0, f_1, f_2, f_3 : \Sigma \dashrightarrow \Sigma$ be given by

$$
\begin{aligned}
f_0(\sigma) &:= \text{undefined} \\
f_1(\sigma) &:= \begin{cases} \sigma & \text{if } \sigma(x) \text{ even} \\ \text{undefined} & \text{otherwise} \end{cases} \\
f_2(\sigma) &:= \begin{cases} \sigma & \text{if } \sigma(x) \text{ odd} \\ \text{undefined} & \text{otherwise} \end{cases} \\
f_3(\sigma) &:= \sigma
\end{aligned}
$$

Making It Precise II

Example 6.4

Let $x \in Var$ be fixed, and let $f_0, f_1, f_2, f_3 : \Sigma \dashrightarrow \Sigma$ be given by

$$f_0(\sigma) := \text{undefined}$$
$$f_1(\sigma) := \begin{cases} \sigma & \text{if } \sigma(x) \text{ even} \\ \text{undefined} & \text{otherwise} \end{cases}$$
$$f_2(\sigma) := \begin{cases} \sigma & \text{if } \sigma(x) \text{ odd} \\ \text{undefined} & \text{otherwise} \end{cases}$$
$$f_3(\sigma) := \sigma$$

This implies $f_0 \sqsubseteq f_1 \sqsubseteq f_3$, $f_0 \sqsubseteq f_2 \sqsubseteq f_3$, $f_1 \not\sqsubseteq f_2$, and $f_2 \not\sqsubseteq f_1$

Making It Precise

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

Now $\mathsf{fix}(\Phi)$ can be characterised by:

- $\mathsf{fix}(\Phi)$ is a fixpoint of Φ, i.e.,

$$\Phi(\mathsf{fix}(\Phi)) = \mathsf{fix}(\Phi)$$

- $\mathsf{fix}(\Phi)$ is minimal with respect to $\sqsubseteq$, i.e., for every $f_0 : \Sigma \dashrightarrow \Sigma$ such that $\Phi(f_0) = f_0$,

$$\mathsf{fix}(\Phi) \sqsubseteq f_0$$

Making It Precise

Characterisation of $\text{fix}(\Phi)$ I

Now $\text{fix}(\Phi)$ can be characterised by:

- $\text{fix}(\Phi)$ is a fixpoint of Φ, i.e.,

$$\Phi(\text{fix}(\Phi)) = \text{fix}(\Phi)$$

- $\text{fix}(\Phi)$ is minimal with respect to $\sqsubseteq$, i.e., for every $f_0 : \Sigma \dashrightarrow \Sigma$ such that $\Phi(f_0) = f_0$,

$$\text{fix}(\Phi) \sqsubseteq f_0$$

Example 6.5

For `while true do skip end` we obtain for every $f : \Sigma \dashrightarrow \Sigma$:

$$\Phi(f) = \text{cond}(\mathfrak{B}[\![\text{true}]\!], f \circ \mathfrak{C}[\![\texttt{skip}]\!], \text{id}_\Sigma) = f$$

Making It Precise

Characterisation of $\text{fix}(\Phi)$ I

Now $\text{fix}(\Phi)$ can be characterised by:

- $\text{fix}(\Phi)$ is a fixpoint of Φ, i.e.,

$$\Phi(\text{fix}(\Phi)) = \text{fix}(\Phi)$$

- $\text{fix}(\Phi)$ is minimal with respect to $\sqsubseteq$, i.e., for every $f_0 : \Sigma \dashrightarrow \Sigma$ such that $\Phi(f_0) = f_0$,

$$\text{fix}(\Phi) \sqsubseteq f_0$$

Example 6.5

For `while true do skip end` we obtain for every $f : \Sigma \dashrightarrow \Sigma$:

$$\Phi(f) = \text{cond}(\mathfrak{B}[\![\text{true}]\!], f \circ \mathfrak{C}[\![\texttt{skip}]\!], \text{id}_\Sigma) = f$$

$\Rightarrow \text{fix}(\Phi) = f_\emptyset$ where $f_\emptyset(\sigma) := \text{undefined}$ for every $\sigma \in \Sigma$ (that is, $\text{graph}(f_\emptyset) = \emptyset$)

Making It Precise

Characterisation 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)

Making It Precise

Characterisation of $\mathrm{fix}(\Phi)$ II

Goals:

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

Sufficient conditions:

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

on function Φ: monotonicity and continuity