Semantics and Verification of Software

Lecture 5: Denotational Semantics of WHILE I (Fixpoint Semantics)

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/svsw11/

Winter Semester 2011/12

Technologie-Beratung @ SAP:

Über ein spannendes Unternehmen, innovative Technologien und Ihren möglichen Berufseinstieg als Trainee im Consulting

SAP Firmenvortrag
Dienstag, 29. November 2011
17:30 bis 18:30
Hörsaal AH1, Ahornstraße

Vortragsinhalte

- Vorstellung der SAP als Unternehmen und Arbeitgeber
- Schwerpunkte der Technologie-Beratung bei SAP
- Trainee-Programm für Berufseinsteiger
  in der Technologie-Beratung
- Erfahrungsbericht eines Trainees über den Berufseinstieg
- Fragen- und Antwortrunde

Trainee Program
IT Transformation
Consulting
Start Januar 2012

Sie wollen die Welt des Business bewegen? Finden Sie weitere Informationen unter www.sap.com/careers

Outline

Summary: Operational Semantics

- Formalized by evaluation/execution relations

Summary: Operational Semantics

- Formalized by evaluation/execution relations
- Inductively defined by derivation trees using structural operational rules

Summary: Operational Semantics

- Formalized by evaluation/execution relations
- Inductively defined by derivation trees using structural operational rules
- Enables proofs about operational behavior of programs using structural induction on derivation trees

Summary: Operational Semantics

- Formalized by evaluation/execution relations
- Inductively defined by derivation trees using structural operational rules
- Enables proofs about operational behavior of programs using structural induction on derivation trees
- Semantic functional characterizes complete input/output behavior of programs

Outline

Denotational Semantics of WHILE

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

Denotational Semantics of WHILE

- Primary aspect of a program: its “effect”, i.e., input/output behavior
- In operational semantics: indirect definition of semantic functional $\mathfrak{O}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma)$ 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 $\mathfrak{O}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma)$ by execution relation
- Now: abstract from operational details

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 semanics: direct definition of program effect by induction on its syntactic structure

Outline

Semantics of Arithmetic Expressions

Again: value of an expression determined by current state

Definition 5.1 (Denotational semantics of arithmetic expressions)

The (denotational) semantic functional for arithmetic expressions,

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

is given by:

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

Semantics of Boolean Expressions

Definition 5.2 (Denotational semantics of Boolean expressions)

The (denotational) semantic functional for Boolean expressions,

$$\mathfrak{B}[\![.]\!] : \mathit{BExp} \to (\Sigma \to \mathbb{B}),$$

is given by:

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

Outline

Semantics of Statements I

- Now: semantic functional

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

Semantics of Statements I

- Now: semantic functional
$$\mathfrak{C}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma)$$
- Same type as operational functional
$$\mathfrak{O}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma)$$
(in fact, both will turn out to be the same
$\Longrightarrow$ equivalence of operational and denotational semantics)

Semantics of Statements I

- Now: semantic functional
$$\mathfrak{C}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma)$$
- Same type as operational functional
$$\mathfrak{O}[\![.]\!] : \mathit{Cmd} \to (\Sigma \dashrightarrow \Sigma)$$
(in fact, both will turn out to be the same
$\Longrightarrow$ equivalence of operational and denotational semantics)
- Inductive definition employs 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 II

Definition 5.3 (Denotational semantics of statements)

The (denotational) semantic functional for statements,

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

is given by:

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

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 III

Remarks:

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

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

Semantics of Statements III

Remarks:

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

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

- 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 III

Remarks:

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

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

- 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 $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!] := \mathsf{fix}(\Phi)$, fix denotes a fixpoint operator (which remains to be defined)
  $\Longrightarrow$ "fixpoint semantics"

Semantics of Statements III

Remarks:

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

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

- 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 $\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!] := \mathsf{fix}(\Phi)$, fix denotes a fixpoint operator (which remains to be defined)
  $\Longrightarrow$ “fixpoint semantics”

But: why fixpoints?

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)

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. 5.3, we obtain:

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

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. 5.3, we obtain:

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

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. 5.3, we obtain:
  $$\begin{aligned} &\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!] \\ \overset{(*)}{=}\quad &\mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ (c\texttt{;while}\ b\ \texttt{do}\ c)\ \texttt{else skip}]\!] \\ \overset{\text{Def. 5.3}}{=}\quad &\text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c\texttt{;while}\ b\ \texttt{do}\ c]\!], \mathfrak{C}[\![\texttt{skip}]\!]) \end{aligned}$$

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. 5.3, we obtain:

$$\begin{aligned}
&\mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!]\\
\overset{(*)}{=}\quad & \mathfrak{C}[\![\texttt{if } b \texttt{ then } (c\texttt{;while } b \texttt{ do } c) \texttt{ else skip}]\!]\\
\overset{\text{Def. 5.3}}{=}\quad & \mathrm{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c\texttt{;while } b \texttt{ do } c]\!], \mathfrak{C}[\![\texttt{skip}]\!])\\
\overset{\text{Def. 5.3}}{=}\quad & \mathrm{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!] \circ \mathfrak{C}[\![c]\!], \mathrm{id}_\Sigma)
\end{aligned}$$

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. 5.3, we obtain:
  $$\begin{aligned} &\mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!] \\ \overset{(*)}{=}\quad &\mathfrak{C}[\![\texttt{if}\ b\ \texttt{then}\ (c\texttt{;while}\ b\ \texttt{do}\ c)\ \texttt{else skip}]\!] \\ \overset{\text{Def. 5.3}}{=}\quad &\text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c\texttt{;while}\ b\ \texttt{do}\ c]\!], \mathfrak{C}[\![\texttt{skip}]\!]) \\ \overset{\text{Def. 5.3}}{=}\quad &\text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!] \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma) \end{aligned}$$
- Abbreviating $f := \mathfrak{C}[\![\texttt{while}\ b\ \texttt{do}\ c]\!]$ this yields:
  $$f = \text{cond}(\mathfrak{B}[\![b]\!], f \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma)$$

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. 5.3, we obtain:
  $$\begin{aligned}&\mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!]\\ \overset{(*)}{=}\ &\mathfrak{C}[\![\texttt{if } b \texttt{ then } (c\texttt{;while } b \texttt{ do } c) \texttt{ else skip}]\!]\\ \overset{\text{Def. 5.3}}{=}\ &\text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c\texttt{;while } b \texttt{ do } c]\!], \mathfrak{C}[\![\texttt{skip}]\!])\\ \overset{\text{Def. 5.3}}{=}\ &\text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!] \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma)\end{aligned}$$
- Abbreviating $f := \mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!]$ 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

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. 5.3, we obtain:
  $$\begin{aligned} &\mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!] \\ \overset{(*)}{=} \quad & \mathfrak{C}[\![\texttt{if } b \texttt{ then } (c\texttt{;while } b \texttt{ do } c) \texttt{ else skip}]\!] \\ \overset{\text{Def. 5.3}}{=} \quad & \text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![c\texttt{;while } b \texttt{ do } c]\!], \mathfrak{C}[\![\texttt{skip}]\!]) \\ \overset{\text{Def. 5.3}}{=} \quad & \text{cond}(\mathfrak{B}[\![b]\!], \mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!] \circ \mathfrak{C}[\![c]\!], \text{id}_\Sigma) \end{aligned}$$
- Abbreviating $f := \mathfrak{C}[\![\texttt{while } b \texttt{ do } c]\!]$ 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. $\Phi_1 : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto \begin{cases} g_1 & \text{if } f = g_2 \\ g_2 & \text{otherwise} \end{cases}$

   (where $g_1 \neq g_2$) has no fixpoint

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. $\Phi_1 : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto \begin{cases} g_1 & \text{if } f = g_2 \\ g_2 & \text{otherwise} \end{cases}$

   (where $g_1 \neq g_2$) has no fixpoint

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. $\Phi_1 : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto \begin{cases} g_1 & \text{if } f = g_2 \\ g_2 & \text{otherwise} \end{cases}$ (where $g_1 \neq g_2$) has no fixpoint

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. $\Phi_1 : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma) : f \mapsto \begin{cases} g_1 & \text{if } f = g_2 \\ g_2 & \text{otherwise} \end{cases}$

   (where $g_1 \neq g_2$) has no fixpoint

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

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

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

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

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

Characterization 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., $\Phi(f_0) = f_0$

Characterization 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 terminates after n iterations ($n \in \mathbb{N}$)
  2. body c diverges in the nth iteration
     (since it contains a non-terminating `while` statement)
  3. loop `while` b `do` c itself diverges

Case 1: Termination of Loop

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

Case 1: Termination of Loop

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

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

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

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

- 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

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

Case 1: Termination of Loop

- Loop `while` b `do` c 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

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

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

Case 2: Divergence of Body

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

Case 2: Divergence of Body

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

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

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

Case 2: Divergence of Body

- Body c diverges in the nth 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

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

Case 2: Divergence of Body

- Body c diverges in the nth 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

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

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

Case 3: Divergence of Loop

- Loop `while` b `do` c diverges

Case 3: Divergence of Loop

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

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

Case 3: Divergence of Loop

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

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

- Here only derivable:

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

Case 3: Divergence of Loop

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

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

- Here only derivable:

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

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

Summary

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

1. Loop `while` b `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 `while` b `do` c diverges
   $\implies$ no condition on f_0 (only $f_0(\sigma_0) = f_0(\sigma_i)$ for every $i \in \mathbb{N}$)

Summary

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

1. Loop `while` b `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 `while` b `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., for the loop `while true do skip` every $f : \Sigma \dashrightarrow \Sigma$ is a fixpoint:

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

Summary

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., for the loop $\texttt{while true do skip}$ every $f : \Sigma \dashrightarrow \Sigma$ is a fixpoint:
$$\Phi(f) = \text{cond}(\mathfrak{B}[\![\text{true}]\!], f \circ \mathfrak{C}[\![\texttt{skip}]\!], \text{id}_\Sigma) = f$$
- On the other hand, our operational understanding requires, for every $\sigma_0 \in \Sigma$,
$$\mathfrak{C}[\![\texttt{while true do skip}]\!]\sigma_0 = \text{undefined}$$

Summary

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., for the loop $\texttt{while true do skip}$ every $f : \Sigma \dashrightarrow \Sigma$ is a fixpoint:
  $$\Phi(f) = \text{cond}(\mathfrak{B}[\![\text{true}]\!], f \circ \mathfrak{C}[\![\texttt{skip}]\!], \text{id}_\Sigma) = f$$
- 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

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