



# Semantics and Verification of Software

Winter Semester 2017/18

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/ws-1718/sv-sw/>

# The Denotational Approach

---

## Outline of Lecture 6

The Denotational Approach

Denotational Semantics of Expressions

Denotational Semantics of Statements

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

Making It Precise

# The Denotational Approach

---

## Denotational Semantics of WHILE

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

# The Denotational Approach

---

## Denotational Semantics of WHILE

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

$$\mathcal{D}[\cdot] : \text{Cmd} \rightarrow (\Sigma \dashrightarrow \Sigma)$$

by execution relation

# The Denotational Approach

---

## Denotational Semantics of WHILE

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

$$\mathcal{D}[\cdot] : \text{Cmd} \rightarrow (\Sigma \dashrightarrow \Sigma)$$

by execution relation

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

# Denotational Semantics of Expressions

---

## Outline of Lecture 6

The Denotational Approach

Denotational Semantics of Expressions

Denotational Semantics of Statements

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

Making It Precise

# Denotational Semantics of Expressions

---

## 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,

$$\mathcal{A}[\cdot] : AExp \rightarrow (\Sigma \rightarrow \mathbb{Z}),$$

is given by:

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

# Denotational Semantics of Expressions

## Semantics of Boolean Expressions

### Definition 6.2 (Denotational semantics of Boolean expressions)

The (denotational) semantic functional for Boolean expressions is given by

$\mathcal{B}[\cdot] : BExp \rightarrow (\Sigma \rightarrow \mathbb{B})$  where

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



# Denotational Semantics of Statements

---

## Outline of Lecture 6

The Denotational Approach

Denotational Semantics of Expressions

Denotational Semantics of Statements

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

Making It Precise

# Denotational Semantics of Statements

---

## The Goal

- Now: semantic functional

$$\mathcal{C}[\cdot] : \text{Cmd} \rightarrow (\Sigma \dashrightarrow \Sigma)$$

# Denotational Semantics of Statements

---

## The Goal

- Now: semantic functional

$$\mathcal{E}[\cdot] : Cmd \rightarrow (\Sigma \dashrightarrow \Sigma)$$

- Same type as operational functional

$$\mathcal{O}[\cdot] : Cmd \rightarrow (\Sigma \dashrightarrow \Sigma)$$

(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  $\mathcal{C}[\cdot]$  employs following auxiliary functions:

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

# Denotational Semantics of Statements

---

## Auxiliary Functions

Inductive definition of  $\mathcal{C}[\cdot]$  employs following auxiliary functions:

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

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

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

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

# Denotational Semantics of Statements

---

## Auxiliary Functions

Inductive definition of  $\mathcal{C}[\cdot]$  employs following auxiliary functions:

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

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

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**:

$$\text{cond} : (\Sigma \rightarrow \mathbb{B}) \times (\Sigma \dashrightarrow \Sigma) \times (\Sigma \dashrightarrow \Sigma) \rightarrow (\Sigma \dashrightarrow \Sigma)$$

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

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

# Denotational Semantics of Statements

## Semantics of Statements I

### Definition 6.3 (Denotational semantics of statements)

The (denotational) semantic functional for statements,

$$\mathcal{E}[\cdot] : \text{Cmd} \rightarrow (\Sigma \dashrightarrow \Sigma),$$

is given by:

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

where  $\Phi : (\Sigma \dashrightarrow \Sigma) \rightarrow (\Sigma \dashrightarrow \Sigma) : f \mapsto \text{cond}(\mathcal{B}[b], f \circ \mathcal{E}[c], \text{id}_\Sigma)$

# Denotational Semantics of Statements

---

## Semantics of Statements II

### Remarks:

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



# Denotational Semantics of Statements

---

## Semantics of Statements II

### Remarks:

- In  $\mathcal{E}[[c_1; c_2]] := \mathcal{E}[[c_2]] \circ \mathcal{E}[[c_1]]$ , function composition  $\circ$  has to be **strict** since non-termination of  $c_1$  implies non-termination of  $c_1; c_2$
- Definition of  $\mathcal{E}[[c]]$  given by **induction on syntactic structure** of  $c \in \text{Cmd}$ 
  - in particular,  $\mathcal{E}[[\text{while } b \text{ do } c \text{ end}]]$  only refers to  $\mathcal{B}[[b]]$  and  $\mathcal{E}[[c]]$   
(and not to  $\mathcal{E}[[\text{while } b \text{ do } c \text{ end}]]$  again)
  - note difference to  $\mathcal{D}[[c]]$ :

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

# Denotational Semantics of Statements

---

## Semantics of Statements II

### Remarks:

- In  $\mathcal{C}[[c_1; c_2]] := \mathcal{C}[[c_2]] \circ \mathcal{C}[[c_1]]$ , function composition  $\circ$  has to be **strict** since non-termination of  $c_1$  implies non-termination of  $c_1; c_2$
- Definition of  $\mathcal{C}[[c]]$  given by **induction on syntactic structure** of  $c \in \text{Cmd}$ 
  - in particular,  $\mathcal{C}[[\text{while } b \text{ do } c \text{ end}]]$  only refers to  $\mathcal{B}[[b]]$  and  $\mathcal{C}[[c]]$  (and not to  $\mathcal{C}[[\text{while } b \text{ do } c \text{ end}]]$  again)
  - note difference to  $\mathcal{D}[[c]]$ :

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

- In  $\mathcal{C}[[\text{while } b \text{ do } c \text{ end}]] := \text{fix}(\Phi)$ , **fix** denotes a fixpoint operator (which remains to be defined)  
 $\Rightarrow$  “**fixpoint semantics**”

# Denotational Semantics of Statements

---

## Semantics of Statements II

### Remarks:

- In  $\mathcal{C}[[c_1; c_2]] := \mathcal{C}[[c_2]] \circ \mathcal{C}[[c_1]]$ , function composition  $\circ$  has to be **strict** since non-termination of  $c_1$  implies non-termination of  $c_1; c_2$
- Definition of  $\mathcal{C}[[c]]$  given by **induction on syntactic structure** of  $c \in \text{Cmd}$ 
  - in particular,  $\mathcal{C}[[\text{while } b \text{ do } c \text{ end}]]$  only refers to  $\mathcal{B}[[b]]$  and  $\mathcal{C}[[c]]$  (and not to  $\mathcal{C}[[\text{while } b \text{ do } c \text{ end}]]$  again)
  - note difference to  $\mathcal{D}[[c]]$ :

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

- In  $\mathcal{C}[[\text{while } b \text{ do } c \text{ end}]] := \text{fix}(\Phi)$ , **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**

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

(cf. Lemma 4.3)

# Denotational Semantics of Statements

---

## Why Fixpoints?

- Goal: preserve **validity of equivalence**

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

(cf. Lemma 4.3)

- Using the known parts of Definition 6.3, we obtain:

$$\mathcal{C}[\text{while } b \text{ do } c \text{ end}]$$

# Denotational Semantics of Statements

---

## Why Fixpoints?

- Goal: preserve **validity of equivalence**

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

(cf. Lemma 4.3)

- Using the known parts of Definition 6.3, we obtain:

$$\begin{aligned} &\mathcal{C}[\text{while } b \text{ do } c \text{ end}] \\ &\stackrel{(*)}{=} \mathcal{C}[\text{if } b \text{ then } c; \text{while } b \text{ do } c \text{ end else skip end}] \end{aligned}$$

# Denotational Semantics of Statements

---

## Why Fixpoints?

- Goal: preserve **validity of equivalence**

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

(cf. Lemma 4.3)

- Using the known parts of Definition 6.3, we obtain:

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

# Denotational Semantics of Statements

---

## Why Fixpoints?

- Goal: preserve **validity of equivalence**

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

(cf. Lemma 4.3)

- Using the known parts of Definition 6.3, we obtain:

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



# Denotational Semantics of Statements

---

## Why Fixpoints?

- Goal: preserve **validity of equivalence**

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

(cf. Lemma 4.3)

- Using the known parts of Definition 6.3, we obtain:

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

- Abbreviating  $f := \mathcal{C}[\text{while } b \text{ do } c \text{ end}]$  this yields:

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

# Denotational Semantics of Statements

---

## Why Fixpoints?

- Goal: preserve **validity of equivalence**

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

(cf. Lemma 4.3)

- Using the known parts of Definition 6.3, we obtain:

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

- Abbreviating  $f := \mathcal{C}[\text{while } b \text{ do } c \text{ end}]$  this yields:

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

- Hence  $f$  must be a **solution** of this recursive equation

# Denotational Semantics of Statements

---

## Why Fixpoints?

- Goal: preserve **validity of equivalence**

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

(cf. Lemma 4.3)

- Using the known parts of Definition 6.3, we obtain:

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

- Abbreviating  $f := \mathcal{C}[\text{while } b \text{ do } c \text{ end}]$  this yields:

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

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

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

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

# Denotational Semantics of Statements

---

## Well-Definedness of Fixpoint Semantics

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

# 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.  $\varphi_1 : \mathbb{N} \rightarrow \mathbb{N} : n \mapsto n + 1$  has no fixpoint

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

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.  $\varphi_1 : \mathbb{N} \rightarrow \mathbb{N} : n \mapsto n + 1$  has no fixpoint

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

has no fixpoint if  $g_1 \neq g_2$

**Solution:** in our setting, fixpoints always exist

# 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.  $\varphi_1 : \mathbb{N} \rightarrow \mathbb{N} : n \mapsto n + 1$  has no fixpoint
2.  $\Phi_1 : (\Sigma \dashrightarrow \Sigma) \rightarrow (\Sigma \dashrightarrow \Sigma) : f \mapsto \begin{cases} g_1 & \text{if } f = g_2 \\ g_2 & \text{otherwise} \end{cases}$   
has no fixpoint if  $g_1 \neq g_2$

**Solution:** in our setting, **fixpoints always exist**

**Uniqueness:** there might exist several fixpoints. Examples:

1.  $\varphi_2 : \mathbb{N} \rightarrow \mathbb{N} : n \mapsto n^3$  has fixpoints  $\{0, 1\}$
2. every state transformation  $f$  is a fixpoint of  $\Phi_2 : (\Sigma \dashrightarrow \Sigma) \rightarrow (\Sigma \dashrightarrow \Sigma) : f \mapsto f$

# 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.  $\varphi_1 : \mathbb{N} \rightarrow \mathbb{N} : n \mapsto n + 1$  has no fixpoint
2.  $\Phi_1 : (\Sigma \dashrightarrow \Sigma) \rightarrow (\Sigma \dashrightarrow \Sigma) : f \mapsto \begin{cases} g_1 & \text{if } f = g_2 \\ g_2 & \text{otherwise} \end{cases}$   
has no fixpoint if  $g_1 \neq g_2$

**Solution:** in our setting, **fixpoints always exist**

**Uniqueness:** there might exist several fixpoints. Examples:

1.  $\varphi_2 : \mathbb{N} \rightarrow \mathbb{N} : n \mapsto n^3$  has fixpoints  $\{0, 1\}$
2. every state transformation  $f$  is a fixpoint of  $\Phi_2 : (\Sigma \dashrightarrow \Sigma) \rightarrow (\Sigma \dashrightarrow \Sigma) : f \mapsto f$

**Solution:** uniqueness guaranteed by **choosing a special fixpoint**



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

---

## Outline of Lecture 6

The Denotational Approach

Denotational Semantics of Expressions

Denotational Semantics of Statements

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

Making It Precise

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

---

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

- Let  $b \in BExp$  and  $c \in Cmd$

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

---

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

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

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

---

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

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

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

---

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

- Let  $b \in BExp$  and  $c \in Cmd$
- Let  $\Phi(f) := \text{cond}(\mathcal{B}[[b]], f \circ \mathcal{C}[[c]], \text{id}_\Sigma)$
- Let  $f_0 : \Sigma \dashrightarrow \Sigma$  be a fixpoint of  $\Phi$ , 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 $\text{fix}(\Phi)$

---

## Case 1: Termination of Loop

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

# 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  $\sigma_1, \dots, \sigma_n \in \Sigma$  such that

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

# 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  $\sigma_1, \dots, \sigma_n \in \Sigma$  such that

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

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

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



# 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  $\sigma_1, \dots, \sigma_n \in \Sigma$  such that

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

- 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

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

# 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  $\sigma_1, \dots, \sigma_n \in \Sigma$  such that

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

- 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

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

$\Rightarrow$  All fixpoints  $f_0$  coincide on  $\sigma_0$  (with result  $\sigma_n$ )!

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

---

## Case 2: Divergence of Body

- Body  $c$  diverges in the  $n$ -th iteration ( $n \geq 1$ )  
(since it contains a non-terminating `while` statement)

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

---

## Case 2: Divergence of Body

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

---

## Case 2: Divergence of Body

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

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

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

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

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

---

## Case 2: Divergence of Body

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

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

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

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

$\Rightarrow$  Again all fixpoints  $f_0$  coincide on  $\sigma_0$  (with undefined result)!

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

---

## Case 3: Divergence of Loop

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

## Case 3: Divergence of Loop

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

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



## Case 3: Divergence of Loop

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

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

- Here only derivable:

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

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

---

## Case 3: Divergence of Loop

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

$$\begin{array}{l} \mathcal{B}[[b]]\sigma_i = \text{true} \quad \text{and} \\ \mathcal{C}[[c]]\sigma_i = \sigma_{i+1} \quad \text{for every } i \in \mathbb{N} \end{array}$$

- Here only derivable:

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

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

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

---

## Summary

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

---

## Summary

### 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:

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

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

## Summary

### 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:

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

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

$$\mathcal{C}[\text{while true do skip end}]\sigma_0 = \text{undefined}$$

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

## Summary

### 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:

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

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

$$\mathcal{C}[\text{while true do skip end}]\sigma_0 = \text{undefined}$$

## Conclusion

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

# Making It Precise

---

## Outline of Lecture 6

The Denotational Approach

Denotational Semantics of Expressions

Denotational Semantics of Statements

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

Making It Precise

# Making It Precise

---

## 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

---

## 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  $h : \Sigma \dashrightarrow \Sigma$

# Making It Precise

---

## Making It Precise II

### Example 6.4

Let  $x \in \text{Var}$  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

---

## Making It Precise II

### Example 6.4

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

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

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

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

- $\text{fix}(\Phi)$  is a **fixpoint** of  $\Phi$ , 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  $\Phi(f_0) = f_0$ ,

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

## Making It Precise

---

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

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

- $\text{fix}(\Phi)$  is a **fixpoint** of  $\Phi$ , 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  $f : \Sigma \dashrightarrow \Sigma$ :

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

# Making It Precise

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

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

- $\text{fix}(\Phi)$  is a **fixpoint** of  $\Phi$ , 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  $f : \Sigma \dashrightarrow \Sigma$ :

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

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

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

### Goals:

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

# Making It Precise

---

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

### Goals:

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

### Sufficient conditions:

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

on function  $\Phi$ : **monotonicity** and **continuity**