

Korrekte Software: Grundlagen und Methoden  
 Vorlesung 3 vom 27.04.21  
 Denotationale Semantik

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2021

## Organisatorisches

- ▶ Übungsblätter: 17 Repositories.
- ▶ Organisation der Repos:
  - ▶ Flach (keine Unterverzeichnisse Abgaben, ueb01 o.ä.)
  - ▶ Lösung in uebung-XX.pdf oder uebung-XX.md.
  - ▶ Abgabe auch in Markdown möglich (git-freundlicher).
  - ▶ Bewertungen in 00-BEWERTUNG.md (fortgeschrieben).
- ▶ Feedback.

## Fahrplan

- ▶ Einführung
- ▶ Operationale Semantik
- ▶ **Denotationale Semantik**
- ▶ Äquivalenz der Operationalen und Denotationalen Semantik
- ▶ Der Floyd-Hoare-Kalkül I
- ▶ Der Floyd-Hoare-Kalkül II: Invarianten
- ▶ Korrektheit des Floyd-Hoare-Kalküls
- ▶ Strukturierte Datentypen
- ▶ Verifikationsbedingungen
- ▶ Vorwärts mit Floyd und Hoare
- ▶ Modellierung
- ▶ Spezifikation von Funktionen
- ▶ Referenzen und Speichermodelle
- ▶ Ausblick und Rückblick

## Überblick

- ▶ Denotationale Semantik für C0
- ▶ Fixpunkte

## Denotationale Semantik — Motivation

### ▶ Operationale Semantik:

Eine Menge von Regeln, die einen Zustand und ein Programm in einen neuen Zustand oder Fehler überführen

$$\langle c, \sigma \rangle \rightarrow_{Stmt} \sigma' \perp$$

### ▶ Denotationale Semantik:

Eine Menge von Regeln, die ein Programm in eine **partielle Funktion** von Zustand nach Zustand überführen

$$\llbracket c \rrbracket_c : \Sigma \rightarrow \Sigma$$

## Denotationale Semantik — Motivation

Zwei Programme sind äquivalent gdw. sie immer zum selben Zustand (oder Fehler) auswerten

$$c_0 \sim c_1 \text{ iff } (\forall \sigma, \sigma'. \langle c_0, \sigma \rangle \rightarrow_{Stmt} \sigma' \Leftrightarrow \langle c_1, \sigma \rangle \rightarrow_{Stmt} \sigma')$$

oder

Zwei Programme sind äquivalent gdw. sie dieselbe partielle Funktion **denotieren**

$$c_0 \sim c_1 \text{ iff } \{(\sigma, \sigma') \mid \langle c_0, \sigma \rangle \rightarrow_{Stmt} \sigma'\} = \{(\sigma, \sigma') \mid \langle c_1, \sigma \rangle \rightarrow_{Stmt} \sigma'\}$$

## Kompositionalität

- ▶ Semantik von zusammengesetzten Ausdrücken durch Kombination der Semantiken der Teilausdrücke
- ▶ Bsp: Semantik einer Sequenz von Anweisungen durch Verknüpfung der Semantik der einzelnen Anweisungen

- ▶ Operationale Semantik ist **nicht** kompositional:

```
x = 3;
y = x + 7; // (*)
z = x + y;
```

- ▶ Semantik von Zeile (\*) ergibt sich aus der Ableitung davor
- ▶ Kann nicht unabhängig abgeleitet werden

- ▶ Denotationale Semantik ist kompositional.

- ▶ Wesentlicher Baustein: **partielle Funktionen**

## Partielle Funktion

### Definition (Partielle Funktion)

Eine **partielle Funktion**  $f : X \rightarrow Y$  ist eine Relation  $f \subseteq X \times Y$  so dass wenn  $(x, y_1) \in f$  und  $(x, y_2) \in f$  dann  $y_1 = y_2$  (**Rechtseindeutigkeit**)

- ▶ Notation: für  $f : X \rightarrow Y$ ,  $(x, y) \in f \Leftrightarrow f(x) = y$ .
- ▶ Wir benutzen beide Notationen, aber für die denotationale Semantik die Paar-Notation.
- ▶ Zustände sind partielle Abbildungen ( $\rightarrow$  letzte Vorlesung)
- ▶ Insbesondere **Systemzustände**  $\Sigma \stackrel{\text{def}}{=} \text{Loc} \rightarrow \mathbf{V}$

## Beispiel

Als Beispiel betrachten wir die partielle Funktion  $div3 : \{0 \dots 10\} \rightarrow \mathbb{N}$

$$div3(x) = y \quad \text{g.d.w.} \quad 3 \cdot y = x$$

► Zuordnung:

0  $\mapsto$  0  
1  
2  
3  $\mapsto$  1  
4  
5  
6  $\mapsto$  2  
7  
8  
9  $\mapsto$  3  
10

► Notation als Relation (**Graph**):

$$div3 \stackrel{def}{=} \{(0, 0), (3, 1), (6, 2), (9, 3)\}$$

► Wir schreiben

$$\begin{aligned} div3(3) &= 1 && \text{für } (3, 1) \in div3 \\ div3(5) &= \perp && \text{für es gibt kein } y \text{ mit } (5, y) \in div3 \\ div3(5) &= \perp && \text{für } \forall y. (5, y) \notin div3 \end{aligned}$$

► Achtung, Partialität!

## Arbeitsblatt 3.1: Relationen als Funktionen

Definiert wie im Beispiel eben die Funktion  $sqrt : \{0, \dots, 100\} \rightarrow \mathbb{N}$  mit

$$sqrt(x) = y \quad \text{g.d.w.} \quad y^2 = x$$

Was ist der Wert folgender Ausdrücke:

$$t_1 = 5 - sqrt(32) \quad t_2 = sqrt(49) + sqrt(0) \quad t_3 = \sqrt{3} \cdot sqrt(3) \quad t_4 = \frac{sqrt(64)}{0}$$

## Denotierende Funktionen (Denotate)

► Arithmetische Ausdrücke:  $a \in \mathbf{Aexp}$  denotiert eine partielle Funktion  $\Sigma \rightarrow \mathbb{Z}$

► Boolesche Ausdrücke:  $b \in \mathbf{Bexp}$  denotiert eine partielle Funktion  $\Sigma \rightarrow \mathbb{B}$

► Anweisungen:  $c \in \mathbf{Stmt}$  denotiert eine partielle Funktion  $\Sigma \rightarrow \Sigma$

## Denotat von Aexp

$$\llbracket a \rrbracket_{\mathcal{A}} : \mathbf{Aexp} \rightarrow (\Sigma \rightarrow \mathbb{Z})$$

$$\llbracket n \rrbracket_{\mathcal{A}} = \{(\sigma, \llbracket n \rrbracket) \mid \sigma \in \Sigma\}$$

$$\llbracket x \rrbracket_{\mathcal{A}} = \{(\sigma, \sigma(x)) \mid \sigma \in \Sigma, x \in \text{Dom}(\sigma)\}$$

$$\llbracket a_0 + a_1 \rrbracket_{\mathcal{A}} = \{(\sigma, n_0 + n_1) \mid (\sigma, n_0) \in \llbracket a_0 \rrbracket_{\mathcal{A}} \wedge (\sigma, n_1) \in \llbracket a_1 \rrbracket_{\mathcal{A}}\}$$

$$\llbracket a_0 - a_1 \rrbracket_{\mathcal{A}} = \{(\sigma, n_0 - n_1) \mid (\sigma, n_0) \in \llbracket a_0 \rrbracket_{\mathcal{A}} \wedge (\sigma, n_1) \in \llbracket a_1 \rrbracket_{\mathcal{A}}\}$$

$$\llbracket a_0 * a_1 \rrbracket_{\mathcal{A}} = \{(\sigma, n_0 * n_1) \mid (\sigma, n_0) \in \llbracket a_0 \rrbracket_{\mathcal{A}} \wedge (\sigma, n_1) \in \llbracket a_1 \rrbracket_{\mathcal{A}}\}$$

$$\llbracket a_0 / a_1 \rrbracket_{\mathcal{A}} = \{(\sigma, n_0 \div n_1) \mid (\sigma, n_0) \in \llbracket a_0 \rrbracket_{\mathcal{A}} \wedge (\sigma, n_1) \in \llbracket a_1 \rrbracket_{\mathcal{A}} \wedge n_1 \neq 0\}$$

## Rechtseindeutigkeit

Lemma (Partielle Funktion)

$\llbracket - \rrbracket_{\mathcal{A}}$  ist rechtseindeutig und damit eine **partielle Funktion**.

Beweis.

z.z.: wenn  $(\sigma, v_1) \in \llbracket a \rrbracket_{\mathcal{A}}$ ,  $(\sigma, v_2) \in \llbracket a \rrbracket_{\mathcal{A}}$  dann  $v_1 = v_2$ .

Strukturelle Induktion über **Aexp**:

► Induktionsbasis sind  $n \in \mathbf{Z}$  und  $x \in \mathbf{Idt}$ .

Sei  $a \equiv x$ , dann  $v_1 = \sigma(x) = v_2$ .

► Induktionsschritt sind die anderen Klauseln.

Sei  $a \equiv a_1 + a_2$ .

Induktionsannahme ist: wenn  $(\sigma, n_i) \in \llbracket a_i \rrbracket_{\mathcal{A}}$ ,  $(\sigma, m_i) \in \llbracket a_i \rrbracket_{\mathcal{A}}$  dann  $n_i = m_i$ .

Sei  $v_1 = (\sigma, n_1 + n_2)$  mit  $(\sigma, n_1) \in \llbracket a_1 \rrbracket_{\mathcal{A}}$ ,  $(\sigma, n_2) \in \llbracket a_2 \rrbracket_{\mathcal{A}}$ , und  $v_2 = m_1 + m_2$  mit

$(\sigma, m_1) \in \llbracket a_1 \rrbracket_{\mathcal{A}}$ ,  $(\sigma, m_2) \in \llbracket a_2 \rrbracket_{\mathcal{A}}$ .

Aus der Annahme folgt  $n_1 = m_1$  und  $n_2 = m_2$ , deshalb  $v_1 = v_2$ .

## Kompositionalität und Striktheit

► Die Rechtseindeutigkeit erlaubt die Notation als partielle Funktion:

$$\begin{aligned} \llbracket 3 * (x + y) \rrbracket_{\mathcal{A}}(\sigma) &= \llbracket 3 \rrbracket_{\mathcal{A}}(\sigma) \cdot (\llbracket x \rrbracket_{\mathcal{A}}(\sigma) + \llbracket y \rrbracket_{\mathcal{A}}(\sigma)) \\ &= 3 \cdot (\llbracket x \rrbracket_{\mathcal{A}}(\sigma) + \llbracket y \rrbracket_{\mathcal{A}}(\sigma)) \\ &= 3 \cdot (\sigma(x) + \sigma(y)) \end{aligned}$$

► Diese Notation versteckt die **Partialität**:

$$\llbracket 1 + x/0 \rrbracket_{\mathcal{A}}(\sigma) = 1 + \sigma(x)/0 = 1 + \perp = \perp$$

► Wenn ein Teilausdruck undefiniert ist, wird der gesamte Ausdruck undefiniert:  $\llbracket - \rrbracket_{\mathcal{A}}$  ist **strikt** für alle arithmetischen Operatoren.

## Arbeitsblatt 3.2: Semantik I

Hier üben wir noch einmal den Zusammenhang zwischen den beiden Notationen. Gegeben sei der Zustand  $s = \langle x \mapsto 3, y \mapsto 4 \rangle$  und der Ausdruck  $a = 7 * x + y$ . Berechnen Sie die Semantik zum einen als Relation (füllen Sie die Fragezeichen aus):

$(s, ?) : \llbracket [7] \rrbracket$   
 $(s, ?) : \llbracket [x] \rrbracket$   
 $(s, ?) : \llbracket [7*x] \rrbracket$   
 $(s, ?) : \llbracket [y] \rrbracket$   
 $(s, ?) : \llbracket [7*x + y] \rrbracket$

Berechnen Sie zum anderen die Semantik in der Funktionsnotation:

$$\llbracket [7*x+y] \rrbracket(s) = \llbracket [7*x] \rrbracket(s) + \llbracket [y] \rrbracket(s) = \dots = ?$$

Ist das Ergebnis am Ende gleich?

## Lösung

## Denotat von Bexp

$$\llbracket a \rrbracket_B : \text{Bexp} \rightarrow (\Sigma \rightarrow \mathbb{B})$$

$$\llbracket 1 \rrbracket_B = \{(\sigma, \text{true}) \mid \sigma \in \Sigma\}$$

$$\llbracket 0 \rrbracket_B = \{(\sigma, \text{false}) \mid \sigma \in \Sigma\}$$

$$\llbracket a_0 == a_1 \rrbracket_B = \{(\sigma, \text{true}) \mid \sigma \in \Sigma, (\sigma, n_0) \in \llbracket a_0 \rrbracket_A, (\sigma, n_1) \in \llbracket a_1 \rrbracket_A, n_0 = n_1\} \\ \cup \{(\sigma, \text{false}) \mid \sigma \in \Sigma, (\sigma, n_0) \in \llbracket a_0 \rrbracket_A, (\sigma, n_1) \in \llbracket a_1 \rrbracket_A, n_0 \neq n_1\}$$

$$\llbracket a_0 < a_1 \rrbracket_B = \{(\sigma, \text{true}) \mid \sigma \in \Sigma, (\sigma, n_0) \in \llbracket a_0 \rrbracket_A, (\sigma, n_1) \in \llbracket a_1 \rrbracket_A, n_0 < n_1\} \\ \cup \{(\sigma, \text{false}) \mid \sigma \in \Sigma, (\sigma, n_0) \in \llbracket a_0 \rrbracket_A, (\sigma, n_1) \in \llbracket a_1 \rrbracket_A, n_0 \geq n_1\}$$

## Denotat von Bexp

$$\llbracket a \rrbracket_B : \text{Bexp} \rightarrow (\Sigma \rightarrow \mathbb{B})$$

$$\llbracket !b \rrbracket_B = \{(\sigma, \text{true}) \mid \sigma \in \Sigma, (\sigma, \text{false}) \in \llbracket b \rrbracket_B\}$$

$$\cup \{(\sigma, \text{false}) \mid \sigma \in \Sigma, (\sigma, \text{true}) \in \llbracket b \rrbracket_B\}$$

$$\llbracket b_1 \ \&\& \ b_2 \rrbracket_B = \{(\sigma, \text{false}) \mid \sigma \in \Sigma, (\sigma, \text{false}) \in \llbracket b_1 \rrbracket_B\}$$

$$\cup \{(\sigma, \text{true}) \mid \sigma \in \Sigma, (\sigma, \text{true}) \in \llbracket b_1 \rrbracket_B, (\sigma, \text{true}) \in \llbracket b_2 \rrbracket_B\}$$

$$\llbracket b_1 \ \|\ b_2 \rrbracket_B = \{(\sigma, \text{true}) \mid \sigma \in \Sigma, (\sigma, \text{true}) \in \llbracket b_1 \rrbracket_B\}$$

$$\cup \{(\sigma, \text{false}) \mid \sigma \in \Sigma, (\sigma, \text{false}) \in \llbracket b_1 \rrbracket_B, (\sigma, \text{false}) \in \llbracket b_2 \rrbracket_B\}$$

## Kompositionalität und Striktheit

### Lemma (Partielle Funktion)

$\llbracket - \rrbracket_B$  ist *rechtseindeutig* und damit eine *partielle Funktion*.

- ▶ Beweis analog zu  $\llbracket - \rrbracket_A$ .
- ▶ Ist  $\llbracket - \rrbracket_B$  strikt? Natürlich nicht:
- ▶ Sei  $\llbracket b_1 \rrbracket_B(\sigma) = \text{false}$ , dann  $\llbracket b_1 \ \&\& \ b_2 \rrbracket_B(\sigma) = \llbracket b_1 \rrbracket_B(\sigma) = \text{false}$
- ▶ Wir können deshalb nicht so einfach schreiben  $\llbracket b_1 \ \&\& \ b_2 \rrbracket_B(\sigma) = \llbracket b_1 \rrbracket_B(\sigma) \wedge \llbracket b_2 \rrbracket_B(\sigma)$
- ▶ Die normale zwewertige Logik behandelt Definiertheit gar nicht. Bei uns müssen die logischen Operatoren links-strikt sein:

$$\perp \wedge a = \perp \qquad \text{false} \wedge a = \text{false} \qquad \text{true} \wedge a = a \\ \perp \vee a = \perp \qquad \text{true} \vee a = \text{true} \qquad \text{false} \vee a = a$$

## Arbeitsblatt 3.3: Semantik II

Wir üben noch einmal die Nichtstriktkeit. Gegeben  $s = \langle x \mapsto 7 \rangle$  und  $b \equiv (7 == x) \ \|\ (x/0 == 1)$

Berechnen Sie die Semantik in den Notationen von oben:

$$\langle s, ? \rangle : \llbracket (7 == x) \ \|\ (x/0 == 1) \rrbracket$$

...

$$\llbracket (7 == x) \ \|\ (x/0 == 1) \rrbracket(\langle s \rangle) = \dots ?$$

Hilfreiche Notation:  $a \wedge b = a \ \&\& \ b$ ,  $a \vee b = a \ \|\ b$

## Lösung

$$\langle s, 7 \rangle : \llbracket 7 \rrbracket \\ \langle s, 3 \rangle : \llbracket x \rrbracket \\ \langle s, \text{True} \rangle : \llbracket 7 == x \rrbracket \\ \langle s, \text{True} \rangle : \llbracket (7 == x) \ \|\ (x/0 == 1) \rrbracket$$

Alternativ:

$$\llbracket (7 == x) \ \|\ (x/0 == 1) \rrbracket(\langle s \rangle) = \\ \llbracket (7 == x) \rrbracket(\langle s \rangle) \vee \llbracket (x/0 == 1) \rrbracket(\langle s \rangle) = \\ (\llbracket 7 \rrbracket(\langle s \rangle) = \llbracket x \rrbracket(\langle s \rangle) \vee \llbracket [x/0] \rrbracket(\langle s \rangle) = \llbracket 1 \rrbracket(\langle s \rangle)) = \\ (7 = 7 \ \vee \ \text{bot} = 1) = \\ (\text{True} \vee \ \text{bot}) = \\ \text{True}$$

## Denotationale Semantik von Anweisungen

- ▶ Zuweisung: punktweise Änderung des Zustands  $\sigma$  zu  $\sigma[x \mapsto n]$
- ▶ Sequenz: Komposition von Relationen

### Definition (Komposition von Relationen)

Für zwei Relationen  $R \subseteq X \times Y, S \subseteq Y \times Z$  ist ihre **Komposition**

$$R \circ S \stackrel{\text{def}}{=} \{(x, z) \mid \exists y \in Y. (x, y) \in R \wedge (y, z) \in S\}$$

Wenn  $R, S$  zwei partielle Funktionen sind, ist  $R \circ S$  ihre Funktionskomposition.

- ▶ Leere Sequenz: Leere Funktion? Nein, Identität. Für Menge  $X$ ,

$$\text{Id}_X \stackrel{\text{def}}{=} X \times X = \{(x, x) \mid x \in X\}$$

ist die **Identitätsfunktion** ( $\text{Id}_X(x) = x$ ).

## Arbeitsblatt 3.4: Komposition von Relationen

Zur Übung: betrachten Sie folgende Relationen:

$$R = \{(1, 7), (2, 3), (3, 9), (4, 3)\}$$

$$cS = \{(1, 0), (2, 0), (3, 1), (3, 5), (4, 7), (5, 9), (7, 3), (8, 15)\}$$

Berechnen Sie  $R \circ S = \{(1, ?), \dots\}$

## Denotat von Stmt

$$\llbracket \cdot \rrbracket_C : \text{Stmt} \rightarrow (\Sigma \rightarrow \Sigma)$$

$$\llbracket x = a \rrbracket_C = \{(\sigma, \sigma[x \mapsto n]) \mid \sigma \in \Sigma \wedge (\sigma, n) \in \llbracket a \rrbracket_A\}$$

$$\llbracket c_1; c_2 \rrbracket_C = \llbracket c_1 \rrbracket_C \circ \llbracket c_2 \rrbracket_C$$

$$\llbracket \{\} \rrbracket_C = \text{Id}_\Sigma$$

$$\llbracket \text{if } (b) \ c_0 \ \text{else} \ c_1 \rrbracket_C = \{(\sigma, \sigma') \mid (\sigma, \text{true}) \in \llbracket b \rrbracket_B \wedge (\sigma, \sigma') \in \llbracket c_0 \rrbracket_C\} \\ \cup \{(\sigma, \sigma') \mid (\sigma, \text{false}) \in \llbracket b \rrbracket_B \wedge (\sigma, \sigma') \in \llbracket c_1 \rrbracket_C\}$$

Aber was ist

$$\llbracket \text{while } (b) \ c \rrbracket_C = ??$$

## Denotationale Semantik von while

- Sei  $w \equiv \text{while } (b) \ c$  (und  $\sigma \in \Sigma$ ). Operational gilt:

$$w \sim \text{if } (b) \ \{c; w\} \ \text{else } \{\}$$

- Dann sollte auch gelten

$$\llbracket w \rrbracket_c \stackrel{?}{=} \llbracket \text{if } (b) \ \{c; w\} \ \text{else } \{\} \rrbracket_c$$

- Das ist eine **rekursive** Definition von  $\llbracket w \rrbracket_c$ :

$$x = F(x)$$

- Das ist ein **Fixpunkt**:

$$x = \text{fix}(F)$$

- Was ist das?

## Fixpunkte

### Definition (Fixpunkt)

Für  $f : X \rightarrow X$  ist ein **Fixpunkt** ein  $x \in X$  so dass  $f(x) = x$ .

- Hat jede Funktion  $f : X \rightarrow X$  einen Fixpunkt? Nein
- Kann eine Funktion mehrere Fixpunkte haben? Ja — aber nur einen kleinsten.
- Beispiele
  - Fixpunkte von  $f(x) = \sqrt{x}$  sind 0 und 1; ebenfalls für  $f(x) = x^2$ .
  - Für die Sortierfunktion sind alle sortierten Listen Fixpunkte
  - Die Funktion  $f(x) = x + 1$  hat keinen Fixpunkt in  $\mathbb{Z}$
  - Die Funktion  $f(X) = \mathbb{P}(X)$  hat überhaupt keinen Fixpunkt
- $\text{fix}(f)$  ist also der **kleinste Fixpunkt** von  $f$ .

## Konstruktion des kleinsten Fixpunktes (Kurzversion)

- Gegeben Funktion  $\Gamma$  auf Denotaten  $\Gamma : (\Sigma \rightarrow \Sigma) \rightarrow (\Sigma \rightarrow \Sigma)$

- Wir konstruieren eine Sequenz  $\Gamma^i : \Sigma \rightarrow \Sigma$  (mit  $i \in \mathbb{N}$ ) von Funktionen:

$$\begin{aligned} \Gamma^0(s) &\stackrel{\text{def}}{=} \perp \\ \Gamma^{i+1}(s) &\stackrel{\text{def}}{=} \Gamma(\Gamma^i)(s) \end{aligned}$$

- Dann ist

$$\text{fix}(\Gamma) \stackrel{\text{def}}{=} \bigcup_{i \in \mathbb{N}} \Gamma^i$$

- Verkürzte Version — der Fixpunkt muss so nicht existieren (er tut es aber für alle Programme)

## Denotationale Semantik für die Iteration

- Sei  $w \equiv \text{while } (b) \ c$

- Konstruktion: "Auffalten" der Schleife ( $f$  ist ein Denotat):

$$\begin{aligned} \Gamma(f) = & \{(\sigma, \sigma') \mid (\sigma, \text{true}) \in \llbracket b \rrbracket_B \wedge (\sigma, \sigma') \in \llbracket c \rrbracket_c \circ f\} \\ & \cup \{(\sigma, \sigma) \mid (\sigma, \text{false}) \in \llbracket b \rrbracket_B\} \end{aligned}$$

- $b$  und  $c$  sind Parameter von  $\Gamma$

- Dann ist

$$\llbracket w \rrbracket_c = \text{fix}(\Gamma)$$

## Denotation für Stmt

$$\llbracket \cdot \rrbracket_c : \{\text{Stmt} \rightarrow (\Sigma \rightarrow \Sigma)\}$$

$$\llbracket x = a \rrbracket_c = \{(\sigma, \sigma[x \mapsto a]) \mid \sigma \in \Sigma \wedge (\sigma, a) \in \llbracket a \rrbracket_A\}$$

$$\llbracket c_1; c_2 \rrbracket_c = \llbracket c_1 \rrbracket_c \circ \llbracket c_2 \rrbracket_c$$

$$\llbracket \{\} \rrbracket_c = \text{Id}_\Sigma$$

$$\begin{aligned} \llbracket \text{if } (b) \ c_0 \ \text{else} \ c_1 \rrbracket_c = & \{(\sigma, \sigma') \mid (\sigma, \text{true}) \in \llbracket b \rrbracket_B \wedge (\sigma, \sigma') \in \llbracket c_0 \rrbracket_c\} \\ & \cup \{(\sigma, \sigma') \mid (\sigma, \text{false}) \in \llbracket b \rrbracket_B \wedge (\sigma, \sigma') \in \llbracket c_1 \rrbracket_c\} \end{aligned}$$

$$\llbracket \text{while } (b) \ c \rrbracket_c = \text{fix}(\Gamma)$$

$$\begin{aligned} \Gamma(s) = & \{(\sigma, \sigma') \mid (\sigma, \text{true}) \in \llbracket b \rrbracket_B \wedge (\sigma, \sigma') \in \llbracket c \rrbracket_c \circ s\} \\ & \cup \{(\sigma, \sigma) \mid (\sigma, \text{false}) \in \llbracket b \rrbracket_B\} \end{aligned}$$

## Der Fixpunkt bei der Arbeit (I)

```
while (x < 0) {
  x = x + 1;
}
```

$$\Gamma(f)(\sigma) \stackrel{\text{def}}{=} \begin{cases} \sigma & \sigma(x) \geq 0 \\ f(\sigma[x \mapsto \sigma(x) + 1]) & \sigma(x) < 0 \end{cases}$$

- Wir betrachten den Zustand  $s = \langle x \mapsto ? \rangle$  (nur eine Variable):

s	$\Gamma^0(s)$	$\Gamma^1(s)$	$\Gamma^2(s)$	$\Gamma^3(s)$
-2	$\perp$	$\Gamma^0(s[x \mapsto -1]) = \perp$	$\Gamma^1(s[x \mapsto -1]) = \perp$	$\Gamma^2(s[x \mapsto -1]) = 0$
-1	$\perp$	$\Gamma^0(s[x \mapsto 0]) = \perp$	$\Gamma^1(s[x \mapsto 0]) = 0$	$\Gamma^2(s[x \mapsto 0]) = 0$
0	$\perp$	0	0	0
1	$\perp$	1	1	1

## Der Fixpunkt bei der Arbeit (II)

```
x = 0;
while (n > 0) {
  x = x + n;
  n = n - 1;
}
```

$$\Gamma(f)(\sigma) = \begin{cases} \sigma & \sigma(n) \leq 0 \\ f(\sigma[x \mapsto \sigma(x) + \sigma(n)][n \mapsto \sigma(n) - 1]) & \sigma(n) > 0 \end{cases}$$

- Wir betrachten Zustände  $s = \langle x \mapsto ?, n \mapsto ? \rangle$  (zwei Variablen).

- Der Wert von  $x$  im Initialzustand ist dabei unerheblich:

s	$\Gamma^0(s)$	$\Gamma^1(s)$	$\Gamma^2(s)$	$\Gamma^3(s)$	$\Gamma^4(s)$	$\Gamma^5(s)$
n	x	x	x	x	x	x
-1	$\perp$	$\perp$	0	-1	0	-1
0	$\perp$	$\perp$	0	0	0	0
1	$\perp$	$\perp$	$\perp$	1	0	1
2	$\perp$	$\perp$	$\perp$	$\perp$	3	0
3	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	6
4	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	10

## Der Fixpunkt bei der Arbeit (III)

- Kleine Änderung im Beispielprogramm:

```
x = 0;
while (n != 0) {
  x = x + n;
  n = n - 1;
}
```

$$\Gamma(f)(\sigma) = \begin{cases} \sigma & \sigma(n) = 0 \\ f(\sigma[x \mapsto \sigma(x) + \sigma(n)][n \mapsto \sigma(n) - 1]) & \text{sonst} \end{cases}$$

- Jetzt ergibt sich:

s	$\Gamma^0(s)$	$\Gamma^1(s)$	$\Gamma^2(s)$	$\Gamma^3(s)$	$\Gamma^4(s)$
n	x	x	x	x	x
-2	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
-1	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
0	$\perp$	$\perp$	0	0	0
1	$\perp$	$\perp$	$\perp$	1	0
2	$\perp$	$\perp$	$\perp$	$\perp$	3
3	$\perp$	$\perp$	$\perp$	$\perp$	6

## Der Fixpunkt bei der Arbeit (IV)

```
while (1) {
  x = x+1;
}
```

$$\Gamma(f)(\sigma) \stackrel{\text{def}}{=} f(\sigma[x \mapsto \sigma(x) + 1])$$

Jetzt ergibt sich:

s	$\Gamma^0(s)$	$\Gamma^1(s)$	$\Gamma^2(s)$	$\Gamma^3(s)$
-2	⊥	⊥	⊥	⊥
-1	⊥	⊥	⊥	⊥
0	⊥	⊥	⊥	⊥
1	⊥	⊥	⊥	⊥
2	⊥	⊥	⊥	⊥
3	⊥	⊥	⊥	⊥



## Arbeitsblatt 3.5: Semantik III

Wir betrachten das Beispielprogramm:

```
x = 1;
while (n > 0) {
  x = x*n;
  n = n-1;
}
```

Berechnen Sie wie oben den Fixpunkt:

s	$G^0$	$G^1$	$G^2$	$G^3$	$G^4$
n	x n	x n	x n	x n	x n
0					
1					
2					
3					



## Arbeitsblatt 3.5: Semantik III

Wir betrachten das Beispielprogramm:

```
x = 1;
while (n > 0) {
  x = x*n;
  n = n-1;
}
```



## Der Fixpunkt bei der Arbeit (V)

```
x = 0;
i = 0;
while (i <= n) {
  x = x+i;
  i = i+1;
}
```

$$\Gamma(f)(\sigma) \stackrel{\text{def}}{=} \begin{cases} \sigma & \sigma(i) > \sigma(n) \\ f(\sigma[x \mapsto \sigma(x) + \sigma(i)][i \mapsto \sigma(i) + 1]) & \text{sonst} \end{cases}$$

Wir betrachten nur die while-Schleife mit  $s = \langle n \mapsto ?, i \mapsto ?, x \mapsto ? \rangle$ .

s	$\Gamma^0(s)$	$\Gamma^1(s)$	$\Gamma^2(s)$	$\Gamma^3(s)$	$\Gamma^4(s)$
n i	n i x	n i x	n i x	n i x	n i x
0 0	⊥ ⊥ ⊥	⊥ ⊥ ⊥	0 1 x	0 1 x	0 1 x
0 1	⊥ ⊥ ⊥	0 1 x	0 1 x	0 1 x	0 1 x
1 0	⊥ ⊥ ⊥	⊥ ⊥ ⊥	⊥ ⊥ ⊥	1 2 x+1	1 2 x+1
1 1	⊥ ⊥ ⊥	⊥ ⊥ ⊥	1 2 x+1	1 2 x+1	1 2 x+1
1 2	⊥ ⊥ ⊥	1 2 x	1 2 x	1 2 x	1 2 x
2 0	⊥ ⊥ ⊥	⊥ ⊥ ⊥	⊥ ⊥ ⊥	⊥ ⊥ ⊥	2 3 x+3
2 1	⊥ ⊥ ⊥	⊥ ⊥ ⊥	⊥ ⊥ ⊥	2 3 x+3	2 3 x+3
2 2	⊥ ⊥ ⊥	⊥ ⊥ ⊥	2 3 x+2	2 3 x+2	2 3 x+2
2 3	⊥ ⊥ ⊥	2 3 x	2 3 x	2 3 x	2 3 x



## Weitere Eigenschaften der denotationalen Semantik

Lemma (Partielle Funktion)

$\llbracket - \rrbracket_c$  ist *rechtseindeutig* und damit eine **partielle Funktion**.

- Beweis über strukturelle Induktion über  $c \in \text{Stmt}$  und über **Fixpunktinduktion**:
  - Zu zeigen: wenn  $s$  rechtseindeutig, dann ist  $\Gamma(s)$  rechtseindeutig
  - Dann ist  $\text{fix}(\Gamma)$  rechtseindeutig.
- Eigenschaften der Iteration:
  - Sei  $w \equiv \text{while}(b) c$
  - Dann

$$\llbracket w \rrbracket_c = \llbracket \text{if}(b) \{c; w\} \text{ else } \{\} \rrbracket_c \quad (1)$$

$$(\sigma, \sigma') \in \llbracket w \rrbracket_c \implies (\sigma', \text{false}) \in \llbracket b \rrbracket_B \quad (2)$$



## Beweis (1)

$$\begin{aligned} \llbracket w \rrbracket_c &= \text{fix}(\Gamma) \\ &= \Gamma(\text{fix}(\Gamma)) \\ &= \Gamma(\llbracket w \rrbracket_c) \\ &= \{(\sigma, \sigma') \mid (\sigma, \text{true}) \in \llbracket b \rrbracket_B \wedge (\sigma, \sigma') \in \llbracket c \rrbracket_c \circ \llbracket w \rrbracket_c\} \\ &\quad \cup \{(\sigma, \sigma) \mid (\sigma, \text{false}) \in \llbracket b \rrbracket_B\} \\ &= \{(\sigma, \sigma') \mid (\sigma, \text{true}) \in \llbracket b \rrbracket_B \wedge (\sigma, \sigma') \in \llbracket c \rrbracket_c \circ \llbracket w \rrbracket_c\} \\ &\quad \cup \{(\sigma, \sigma) \mid (\sigma, \text{false}) \in \llbracket b \rrbracket_B \wedge (\sigma, \sigma) \in \llbracket \{\} \rrbracket_c\} \\ &= \llbracket \text{if}(b) \{c; w\} \text{ else } \{\} \rrbracket_c \end{aligned}$$

Note

$$\text{fix}(\Gamma) = \Gamma(\text{fix}(\Gamma)) \Gamma(s) = \{(\sigma, \sigma') \mid (\sigma, \text{true}) \in \llbracket b \rrbracket_B \wedge (\sigma, \sigma') \in \llbracket c \rrbracket_c \circ s\} \llbracket \text{if}(b) c_0 \text{ else } c_1 \rrbracket_c = \{(\sigma, \sigma') \mid (\sigma, \text{true}) \in \llbracket b \rrbracket_B \wedge (\sigma, \sigma') \in \llbracket c \rrbracket_c \circ s\} \cup \{(\sigma, \sigma) \mid (\sigma, \text{false}) \in \llbracket b \rrbracket_B\}$$



## Zusammenfassung

- Die denotationale Semantik bildet Programme (Ausdrücke) auf **partielle Funktionen**  $\Sigma \rightarrow \Sigma$  ab.
- Zentral ist der Begriff des **kleinsten Fixpunktes**, der die Semantik der while-Schleife bildet.
- undefiniertheit wird **implizit** behandelt (durch die Partialität von  $\Sigma \rightarrow \Sigma$ ).
  - Nicht-Termination und undefiniertheit sind semantisch äquivalent.
- Genaueres Verhältnis zur **operationalen Semantik?** Nächste Vorlesung

