

Korrekte Software: Grundlagen und Methoden
Vorlesung 2 vom 10.04.24
Operationale Semantik

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2024

11:48:46 2024-07-04

1 [41]

erko U

Fahrplan

- ▶ Einführung
- ▶ **Operationale Semantik**
- ▶ Denotationale Semantik
- ▶ Äquivalenz der Operationalen und Denotationalen Semantik
- ▶ Der Floyd-Hoare-Kalkül
- ▶ Invarianten im Floyd-Hoare-Kalkül
- ▶ Korrektheit des Floyd-Hoare-Kalküls
- ▶ Strukturierte Datentypen
- ▶ Verifikationsbedingungen
- ▶ Funktionen und Prozeduren I
- ▶ Funktionen und Prozeduren II
- ▶ Referenzen
- ▶ Ausblick und Rückblick

Korrekte Software

2 [41]

erko U

Zutaten

```
// GGT(A,B)
if (a == 0) r = b;
else {
  while (b != 0) {
    if (a <= b)
      b = b - a;
    else a = a - b;
  }
  r = a;
}
```

- ▶ Programme berechnen **Werte**
- ▶ Basierend auf
 - ▶ Werte sind **Variablen** zugewiesen
 - ▶ Evaluation von **Ausdrücken**
- ▶ Folgt dem Programmablauf

Korrekte Software

3 [41]

erko U

Unsere Programmiersprache

Wir betrachten einen Ausschnitt der Programmiersprache **C** (**C0**).

Ausbaustufe 1 kennt folgende Konstrukte:

- ▶ Typen: **int**;
- ▶ Ausdrücke: Variablen, Literale (für ganze Zahlen), arithmetische Operatoren (für ganze Zahlen), Relationen (**=**, **<**, **>**, **...**), boolsche Operatoren (**&&**, **||**);
- ▶ Anweisungen:
 - ▶ Fallunterscheidung (**if...else...**), Iteration (**while**), Zuweisung, Blöcke;
 - ▶ Sequenzierung und leere Anweisung sind implizit

Korrekte Software

4 [41]

erko U

C0: Ausdrücke und Anweisungen

Aexp $a ::= \mathbf{Z} \mid \mathbf{Idt} \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \mid a_1 / a_2$
Bexp $b ::= \mathbf{1} \mid \mathbf{0} \mid a_1 == a_2 \mid a_1 < a_2 \mid ! b \mid b_1 \&\& b_2 \mid b_1 || b_2$
Exp $e ::= a \mid b$
Stmt $c ::= \mathbf{Idt} = \mathbf{Exp} \mid$
 $\quad \mid \mathbf{if} (b) c_1 \mathbf{else} c_2$
 $\quad \mid \mathbf{while} (b) c$
 $\quad \mid c_1; c_2$
 $\quad \mid \{\}$

NB: Nicht die **konkrete** Syntax.

Korrekte Software

5 [41]

erko U

Was braucht die Semantik?

```
p = 1;
c = 1;
while (c <= n) {
  p = p * c;
  c = c + 1;
}
```

- ▶ Ein Programm besteht aus **Anweisungen** und **Ausdrücken**.
- ▶ Ausdrücke werden **zustandsabhängig** ausgewertet.
- ▶ Anweisungen **überführen** Zustände.
Woraus besteht die Semantik?
 - 1 Mathematische Modellierung des **Zustands**
 - 2 Auswertung von (arithmetischen und boolschen) Ausdrücken
 - 3 Auswertung von Anweisungen: Zustandsübergänge

Korrekte Software

6 [41]

erko U

Semantik von C0

- ▶ Die (operationale) Semantik einer imperativen Sprache wie C0 ist ein **Zustandsübergang**: das System hat einen impliziten Zustand, der durch Zuweisung von **Werten** an **Adressen** geändert werden kann.

Systemzustände

- ▶ Ausdrücke werten zu **Werten V** (hier ganze Zahlen) aus.
- ▶ Adressen **Loc** sind hier Programmvariablen (Namen): **Loc = Idt**
- ▶ Ein **Systemzustand** bildet Adressen auf Werte ab: $\Sigma = \mathbf{Loc} \rightarrow \mathbf{V}$
- ▶ Ein Programm bildet einen Anfangszustand **möglicherweise** auf einen Endzustand ab (wenn es **terminiert**).

Korrekte Software

7 [41]

erko U

Partielle, endliche Abbildungen I

Zustände sind **partielle, endliche Abbildungen** (finite partial maps)

$$f : X \rightarrow A$$

Notation:

- ▶ $f(x)$ für den Wert von x in f (*lookup*)
- ▶ $f(x) = \perp$ wenn x nicht in f (*undefined*)
- ▶ $f[x \mapsto n]$ für den Update an der Stelle x mit dem Wert n :

$$f[x \mapsto n](y) \stackrel{\text{def}}{=} \begin{cases} n & \text{if } x = y \\ f(y) & \text{otherwise} \end{cases}$$

Korrekte Software

8 [41]

erko U

Partielle, endliche Abbildungen II

Zustände sind **partielle, endliche Abbildungen** (finite partial maps)

$$f : X \rightarrow A$$

Notation:

- $\langle x \mapsto n, y \mapsto m \rangle$ u.ä. für konkrete Abbildungen.
- $\langle \rangle$ ist die leere (überall undefinierte Abbildung):

$$\text{für alle } x \in X \text{ gilt: } \langle \rangle(x) = \perp$$

- Die Domäne eines Zustands sind alle Stellen, an denen er definiert ist:

$$\text{Dom}(f) \stackrel{\text{def}}{=} \{x \in X \mid f(x) \neq \perp\}$$

- Updates sind "linksassoziativ":

$$f[x \mapsto n][y \mapsto m] = (f[x \mapsto n])[y \mapsto m]$$

Arbeitsblatt 2.1: Zustände!

- Wie sieht ein Zustand aus, der a den Wert 6 und c den Wert 2 zuweist.

- Welches sind Zustände, und welche nicht:

- Ⓐ $\langle x \mapsto 1, a \mapsto 3 \rangle$
- Ⓑ $\langle x \mapsto y, b \mapsto 6 \rangle$
- Ⓒ $\langle x \mapsto 2, b \mapsto 6, x \mapsto 5 \rangle$
- Ⓓ $\langle x \mapsto 3, b \mapsto 6, y \mapsto 5 \rangle$

- Update von Zuständen:

- Ⓐ $\langle x \mapsto 1, a \mapsto 3 \rangle[y \mapsto 1] = ??$
- Ⓑ $\langle x \mapsto 1, a \mapsto 3 \rangle[x \mapsto 3] = ??$
- Ⓒ $\langle x \mapsto 1, a \mapsto 3 \rangle[x \mapsto 3][y \mapsto 1][x \mapsto 4] = ??$

Operationale Semantik: Arithmetische Ausdrücke

Ein arithmetischer Ausdruck a wertet unter Zustand σ zu einer ganzen Zahl n (Wert) aus.

$$\mathbf{Aexp} \ a ::= \mathbf{Z} \mid \mathbf{ldt} \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \mid a_1 / a_2 \quad \langle a, \sigma \rangle \rightarrow_{Aexp} n$$

Regeln:

$$\frac{n \in \mathbf{Z}}{\langle n, \sigma \rangle \rightarrow_{Aexp} [n]} \quad \frac{x \in \mathbf{ldt}, x \in \text{Dom}(\sigma), \sigma(x) = v}{\langle x, \sigma \rangle \rightarrow_{Aexp} v}$$

Operationale Semantik: Arithmetische Ausdrücke

$$\mathbf{Aexp} \ a ::= \mathbf{Z} \mid \mathbf{ldt} \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \mid a_1 / a_2 \quad \langle a, \sigma \rangle \rightarrow_{Aexp} n$$

$$\frac{\langle a_1, \sigma \rangle \rightarrow_{Aexp} n_1 \quad \langle a_2, \sigma \rangle \rightarrow_{Aexp} n_2 \quad n_i \in \mathbb{Z}, n \text{ Summe } n_1 \text{ und } n_2}{\langle a_1 + a_2, \sigma \rangle \rightarrow_{Aexp} n}$$

$$\frac{\langle a_1, \sigma \rangle \rightarrow_{Aexp} n_1 \quad \langle a_2, \sigma \rangle \rightarrow_{Aexp} n_2 \quad n_i \in \mathbb{Z}, n \text{ Differenz } n_1 \text{ und } n_2}{\langle a_1 - a_2, \sigma \rangle \rightarrow_{Aexp} n}$$

$$\frac{\langle a_1, \sigma \rangle \rightarrow_{Aexp} n_1 \quad \langle a_2, \sigma \rangle \rightarrow_{Aexp} n_2 \quad n_i \in \mathbb{Z}, n \text{ Produkt } n_1 \text{ und } n_2}{\langle a_1 * a_2, \sigma \rangle \rightarrow_{Aexp} n}$$

$$\frac{\langle a_1, \sigma \rangle \rightarrow_{Aexp} n_1 \quad \langle a_2, \sigma \rangle \rightarrow_{Aexp} n_2 \quad n_i \in \mathbb{Z}, n_2 \neq 0, n \text{ Quotient } n_1, n_2}{\langle a_1 / a_2, \sigma \rangle \rightarrow_{Aexp} n}$$

Ableitungen

- Regeln werden von **unten** nach **oben** gelesen

- Regeln werden **komponiert** — es entsteht ein **Ableitungsbaum**

Beispiel: Auswertung von $x+3$ mit $\sigma \stackrel{\text{def}}{=} \langle x \mapsto 6, y \mapsto 5 \rangle$.

$$\frac{x \in \text{dom}(\sigma), \sigma(x) = ? \quad x \in \text{dom}(\sigma), \sigma(x) = 6}{\langle x, \sigma \rangle \rightarrow_{Aexp} ?} \quad \frac{\langle x, \sigma \rangle \rightarrow_{Aexp} 6 \quad \langle 3, \sigma \rangle \rightarrow_{Aexp} 3}{\langle x + 3, \sigma \rangle \rightarrow_{Aexp} 6 + 3} \quad \frac{\langle x, \sigma \rangle \rightarrow_{Aexp} ? \quad \langle 3, \sigma \rangle \rightarrow_{Aexp} 3}{\langle x + 3, \sigma \rangle \rightarrow_{Aexp} ? + 3} \quad \frac{\langle x + 3, \sigma \rangle \rightarrow_{Aexp} ? + 3}{\langle x + 3, \sigma \rangle \rightarrow_{Aexp} 6 + 39}$$

Längere Beispiel-Ableitungen

Sei $\sigma \stackrel{\text{def}}{=} \langle x \mapsto 6, y \mapsto 5 \rangle$.

$$\frac{x \in \text{dom}(\sigma), \sigma(x) = 6 \quad y \in \text{dom}(\sigma), \sigma(y) = 5}{\langle x, \sigma \rangle \rightarrow_{Aexp} 6 \quad \langle y, \sigma \rangle \rightarrow_{Aexp} 5} \quad \frac{x \in \text{dom}(\sigma), \sigma(x) = 6 \quad y \in \text{dom}(\sigma), \sigma(y) = 5}{\langle x, \sigma \rangle \rightarrow_{Aexp} 6 \quad \langle y, \sigma \rangle \rightarrow_{Aexp} 5} \quad \frac{\langle x + y, \sigma \rangle \rightarrow_{Aexp} 11 \quad \langle (x + y) * (x - y), \sigma \rangle \rightarrow_{Aexp} 11}{\langle (x + y) * (x - y), \sigma \rangle \rightarrow_{Aexp} 11}$$

$$\frac{\langle x, \sigma \rangle \rightarrow_{Aexp} 6 \quad \langle x, \sigma \rangle \rightarrow_{Aexp} 6 \quad \langle y, \sigma \rangle \rightarrow_{Aexp} 5 \quad \langle y, \sigma \rangle \rightarrow_{Aexp} 5}{\langle x * x, \sigma \rangle \rightarrow_{Aexp} 36 \quad \langle y * y, \sigma \rangle \rightarrow_{Aexp} 25} \quad \frac{\langle x * x, \sigma \rangle \rightarrow_{Aexp} 36 \quad \langle y * y, \sigma \rangle \rightarrow_{Aexp} 25}{\langle (x * x) - (y * y), \sigma \rangle \rightarrow_{Aexp} 11}$$

Arbeitsblatt 2.2: Auswertung

Konstruiert wie oben die Ableitung für den Ausdruck $(3*a)/b$ mit $\sigma \stackrel{\text{def}}{=} \langle a \mapsto 8, b \mapsto 7 \rangle$.

Hinweis: wahrscheinlich einfacher auf Papier...

Eigenschaften der Semantik

- **Frage:** Gegeben einen Ausdruck a , leitet **jeder** Zustand σ zu einem Wert n ab?

- **Antwort:** Nein.

- Betrachte folgende Beispiele für $a \stackrel{\text{def}}{=} y+3/x$

$$\langle a, \langle y \mapsto 5 \rangle \rangle \rightarrow_{Aexp} ??? \quad (1)$$

$$\langle a, \langle y \mapsto 5, x \mapsto 0 \rangle \rangle \rightarrow_{Aexp} ??? \quad (2)$$

- In diesen Beispielen läßt sich kein **vollständiger** Ableitungsbaum konstruieren.

- Die Auswertung ist **undefiniert** — die Semantik ist **partiell**.

Operationale Semantik: Boolesche Ausdrücke

► **Bexp** $b ::= 1 \mid 0 \mid a_1 == a_2 \mid a_1 < a_2 \mid !b \mid b_1 \&\& b_2 \mid b_1 \parallel b_2$
 $\langle b, \sigma \rangle \rightarrow_{Bexp} true \mid false$

Regeln:

$$\frac{}{\langle 1, \sigma \rangle \rightarrow_{Bexp} true} \quad \frac{}{\langle 0, \sigma \rangle \rightarrow_{Bexp} false}$$

$$\frac{\langle a_1, \sigma \rangle \rightarrow_{Aexp} n_1 \quad \langle a_2, \sigma \rangle \rightarrow_{Aexp} n_2 \quad n_1 \text{ und } n_2 \text{ gleich}}{\langle a_1 == a_2, \sigma \rangle \rightarrow_{Bexp} true}$$

$$\frac{\langle a_1, \sigma \rangle \rightarrow_{Aexp} n_1 \quad \langle a_2, \sigma \rangle \rightarrow_{Aexp} n_2 \quad n_1 \text{ und } n_2 \text{ ungleich}}{\langle a_1 == a_2, \sigma \rangle \rightarrow_{Bexp} false}$$

Operationale Semantik: Boolesche Ausdrücke

► **Bexp** $b ::= 1 \mid 0 \mid a_1 == a_2 \mid a_1 < a_2 \mid !b \mid b_1 \&\& b_2 \mid b_1 \parallel b_2$
 $\langle b, \sigma \rangle \rightarrow_{Bexp} true \mid false$

Regeln:

$$\frac{\langle b, \sigma \rangle \rightarrow_{Bexp} true}{\langle !b, \sigma \rangle \rightarrow_{Bexp} false} \quad \frac{\langle b, \sigma \rangle \rightarrow_{Bexp} false}{\langle !b, \sigma \rangle \rightarrow_{Bexp} true}$$

$$\frac{\langle b_1, \sigma \rangle \rightarrow_{Bexp} false}{\langle b_1 \&\& b_2, \sigma \rangle \rightarrow_{Bexp} false} \quad \frac{\langle b_1, \sigma \rangle \rightarrow_{Bexp} true \quad \langle b_2, \sigma \rangle \rightarrow_{Bexp} t}{\langle b_1 \&\& b_2, \sigma \rangle \rightarrow_{Bexp} t}$$

$$\frac{\langle b_1, \sigma \rangle \rightarrow_{Bexp} true}{\langle b_1 \parallel b_2, \sigma \rangle \rightarrow_{Bexp} true} \quad \frac{\langle b_1, \sigma \rangle \rightarrow_{Bexp} false \quad \langle b_2, \sigma \rangle \rightarrow_{Bexp} t}{\langle b_1 \parallel b_2, \sigma \rangle \rightarrow_{Bexp} t}$$

Arbeitsblatt 2.3: Boolesche Ausdrücke

Konstruiert die Auswertung des Ausdrucks $b = x == 7 \&\& y == 3$ unter folgenden Zuständen:

- ① $\sigma_1 \stackrel{def}{=} \langle x \mapsto 7, y \mapsto 3 \rangle$
- ② $\sigma_2 \stackrel{def}{=} \langle x \mapsto 6, y \mapsto 3 \rangle$
- ③ $\sigma_3 \stackrel{def}{=} \langle y \mapsto 6 \rangle$
- ④ $\sigma_4 \stackrel{def}{=} \langle x \mapsto 7 \rangle$
- ⑤ $\sigma_5 \stackrel{def}{=} \langle x \mapsto 2 \rangle$

Striktheit

- Eine partielle Funktion f ist **strikt** wenn $f(x)$ undefiniert ist, sobald x undefiniert ist.
- In unserer Semantik sind alle Operatoren (arithmetisch und boolesch) strikt, **bis auf** $\&\&$ und $\parallel$ im **ersten** Argument.
 - Operational nennt man das auch abgekürzte Auswertung (*short-circuit evaluation*)
 - Das erlaubt Idiome wie **if** $(x != 0 \&\& 3/x > 1) \{ \dots \}$
- Wie erkennt man Striktheit an den **Regeln**?
 Alle Variablen der Konklusion kommen in den Bedingungen vor.

Operationale Semantik: Anweisungen

► **Stmt** $c ::= \text{Idt} = \text{Exp} \mid \text{if } (b) \ c_1 \ \text{else } c_2 \mid \text{while } (b) \ c \mid c_1; c_2 \mid \{ \}$

Beispiel:

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

$$\langle x = 5, \sigma \rangle \rightarrow_{Stmt} \sigma' \sigma[x \mapsto 5]$$

wobei $\sigma'(x) = 5$ und $\sigma'(y) = \sigma(y)$ für alle $y \neq x$
 bzw. $\sigma' \stackrel{def}{=} \sigma[x \mapsto 5]$

Operationale Semantik: Anweisungen

► **Stmt** $c ::= \text{Idt} = \text{Exp} \mid \text{if } (b) \ c_1 \ \text{else } c_2 \mid \text{while } (b) \ c \mid c_1; c_2 \mid \{ \}$

Regeln:

$$\frac{}{\langle \{ \}, \sigma \rangle \rightarrow_{Stmt} \sigma} \quad \frac{\langle a, \sigma \rangle \rightarrow_{Aexp} n \in \mathbb{Z}}{\langle x = a, \sigma \rangle \rightarrow_{Stmt} \sigma[x \mapsto n]}$$

$$\frac{\langle c_1, \sigma \rangle \rightarrow_{Stmt} \sigma' \quad \langle c_2, \sigma' \rangle \rightarrow_{Stmt} \sigma''}{\langle c_1; c_2, \sigma \rangle \rightarrow_{Stmt} \sigma''}$$

$$\frac{\langle b, \sigma \rangle \rightarrow_{Bexp} true \quad \langle c_1, \sigma \rangle \rightarrow_{Stmt} \sigma'}{\langle \text{if } (b) \ c_1 \ \text{else } c_2, \sigma \rangle \rightarrow_{Stmt} \sigma'} \quad \frac{\langle b, \sigma \rangle \rightarrow_{Bexp} false \quad \langle c_2, \sigma \rangle \rightarrow_{Stmt} \sigma'}{\langle \text{if } (b) \ c_1 \ \text{else } c_2, \sigma \rangle \rightarrow_{Stmt} \sigma'}$$

Operationale Semantik: Anweisungen

► **Stmt** $c ::= \text{Idt} = \text{Exp} \mid \text{if } (b) \ c_1 \ \text{else } c_2 \mid \text{while } (b) \ c \mid c_1; c_2 \mid \{ \}$

Regeln:

$$\frac{\langle b, \sigma \rangle \rightarrow_{Bexp} false}{\langle \text{while } (b) \ c, \sigma \rangle \rightarrow_{Stmt} \sigma}$$

$$\frac{\langle b, \sigma \rangle \rightarrow_{Bexp} true \quad \langle c, \sigma \rangle \rightarrow_{Stmt} \sigma' \quad \langle \text{while } (b) \ c, \sigma' \rangle \rightarrow_{Stmt} \sigma''}{\langle \text{while } (b) \ c, \sigma \rangle \rightarrow_{Stmt} \sigma''}$$

Beispiel

```
x = 1;
while (y != 0) {
    y = y - 1;
    x = 2 * x;
}
// x = 2^y
σ  $\stackrel{def}{=} \langle y \mapsto 2 \rangle$ 
```

dfki

26 [41]

27 [41]

Korrekte Software

28 [41]

29 [41]

Korrekte Software

31 [41]

Korrekte Software

Was haben wir gezeigt?

```
// (y ↦ 2)                σ1
x = 1;
// (y ↦ 2, x ↦ 1)
while (y != 0) {
  y = y - 1;
  x = 2 * x;
}
// (y ↦ 0, x ↦ 4)        σE
```

- Für **einen festen Anfangszustand** $\sigma_1 = \langle y \mapsto 2 \rangle$ gilt am Ende $\sigma_E(x) = 4 = 2^2 = 2^{\sigma_1(y)}$.
- Gilt das für alle?
- Für welche nicht?
- Wie kann man das für alle Anfangs-Zustände, für die es gilt, zeigen?

Was passiert hier?

```
// (y ↦ -1)
x = 1;
while (y != 0) {
  y = y - 1;
  x = 2 * x;
}
```

- Ableitung terminiert nicht (Ableitungsbaum der Auswertung der while-Schleife wächst unendlich)
- In linearer Schreibweise geht es immer wieder unten weiter.

Arbeitsblatt 2.4: Programme!

- Werten Sie das nebenstehende Programm aus für den Anfangszustand $\langle x \mapsto 5, y \mapsto 2 \rangle$

```
while (y != 0) {
  x = x * x;
  y = y - 1;
}
```

- Geben Sie die Auswertung in abgekürzter Schreibweise an.
- Welche Beziehung gilt am Ende des Programms zwischen den Werten von x und y im Endzustand und im Anfangszustand?

Äquivalenz arithmetischer Ausdrücke

Gegeben zwei Aexp a_1 and a_2

- Sind sie gleich?

$$a_1 \sim_{Aexp} a_2 \text{ gdw } \forall \sigma, n. \langle a_1, \sigma \rangle \rightarrow_{Aexp} n \Leftrightarrow \langle a_2, \sigma \rangle \rightarrow_{Aexp} n$$

$(x*x) + 2*x*y + (y*y)$ und $(x+y) * (x+y)$

- Wann sind sie gleich?

$$\forall \sigma, n. \langle a_1, \sigma \rangle \rightarrow_{Aexp} n \Leftrightarrow \langle a_2, \sigma \rangle \rightarrow_{Aexp} n$$

$x*x$ und $8*x+9$
 $x*x$ und $x*x+1$

Äquivalenz Boolescher Ausdrücke

Gegeben zwei Bexp-Ausdrücke b_1 and b_2

- Sind sie gleich?

$$b_1 \sim_{Bexp} b_2 \text{ iff } \forall \sigma, b. \langle b_1, \sigma \rangle \rightarrow_{Bexp} b \Leftrightarrow \langle b_2, \sigma \rangle \rightarrow_{Bexp} b$$

$A \mid\mid (A \&\& B)$ und A

Beweisen

Zwei Programme c_0, c_1 sind äquivalent gdw. sie die gleichen Zustandsveränderungen bewirken. Formal definieren wir

Definition

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

Ein einfaches Beispiel:

Lemma

Sei $w \equiv \text{while } (b) \ c$ mit $b \in \mathbf{Bexp}$, $c \in \mathbf{Stmt}$.
 Dann gilt: $w \sim \text{if } (b) \ \{c; w\} \ \text{else } \{\}$

Beweis

- Gegeben beliebiger Programmmzustand σ .
- **Zu zeigen:** sowohl w also auch $\text{if } (b) \ \{c; w\} \ \text{else } \{\}$ werten zum gleichen Programmmzustand aus (wenn sie auswerten).
- Der Beweis geht per Fallunterscheidung über die Auswertung von Teilausdrücken bzw. Teilprogrammen.

Beweis

- ① $\langle b, \sigma \rangle \rightarrow_{Bexp} \text{false}$:

$$\langle \text{while } (b) \ c, \sigma \rangle \rightarrow_{Stmt} \sigma$$

$$\langle \text{if } (b) \ \{c; w\} \ \text{else } \{\}, \sigma \rangle \rightarrow_{Stmt} \langle \{\}, \sigma \rangle \rightarrow_{Stmt} \sigma$$

- ② $\langle b, \sigma \rangle \rightarrow_{Bexp} \text{true}$: Sei $\langle c, \sigma \rangle \rightarrow_{Stmt} \sigma'$, dann:

$$\langle \text{while } (b) \ c, \sigma \rangle \rightarrow_{Stmt} \langle c, \sigma \rangle \rightarrow_{Stmt} \sigma'$$

$$\langle w, \sigma' \rangle \rightarrow_{Stmt} \sigma''$$

$$\langle \text{if } (b) \ \{c; w\} \ \text{else } \{\}, \sigma \rangle \rightarrow_{Stmt} \langle \{c; w\}, \sigma \rangle \rightarrow_{Stmt} \langle c, \sigma \rangle \rightarrow_{Stmt} \sigma'$$

$$\langle w, \sigma' \rangle \rightarrow_{Stmt} \sigma''$$

- ③ $\langle b, \sigma \rangle$ wertet gar nicht aus — dann werten weder w noch $\text{if } (b) \ \{c; w\} \ \text{else } \{\}$ aus.

Zusammenfassung

- ▶ Operationale Semantik als ein Mittel zur Beschreibung der Semantik
- ▶ Auswertungsregeln:
 - ▶ arbeiten entlang der syntaktischen Struktur
 - ▶ werten (zu gegebenen Zustand) Ausdrücke zu Werten aus (Zahlen, Booleschen Werten)
 - ▶ und (zu gegebenen Zustand) Programme zu Zuständen
- ▶ Fragen zu Programmen: Gleichheit