Korrekte Software: Grundlagen und Methoden

Vorlesung 2 vom 10.04.24

Operationale Semantik

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2024

Fahrplan

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

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

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

NB: Nicht die **konkrete** Syntax.

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

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

Partielle, endliche Abbildungen I

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

$$f : X \rightharpoonup A$$

Notation:

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

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

Partielle, endliche Abbildungen II

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

$$f : X \rightharpoonup A$$

Notation:

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

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

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

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

- 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.

$$\textbf{Aexp}\ a ::= \mathbf{Z} \mid \mathbf{Idt} \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \mid a_1\ /\ a_2 \qquad \langle a, \sigma \rangle \to_{Aexp} n$$

Regeln:

$$\frac{n \in \mathbf{Z}}{\langle n, \sigma \rangle \to_{Aexp} [\![n]\!]} \qquad \frac{x \in \mathbf{Idt}, x \in Dom(\sigma), \sigma(x) = v}{\langle x, \sigma \rangle \to_{Aexp} v}$$

Operationale Semantik: Arithmetische Ausdrücke

$$\textbf{Aexp}\ a ::= \mathbf{Z} \mid \mathbf{Idt} \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \mid a_1\ /\ a_2 \qquad \langle a, \sigma \rangle \to_{Aexp} n$$

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

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

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

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

Ableitungen

- Regeln werden von **unten** nach **oben** gelesen
- Regeln werden **komponiert** — es entsteht ein **Ableitungsbaum**

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

$$\frac{\dfrac{x \in \mathrm{dom}(\sigma), \sigma(x) =? 6}{\langle x, \sigma \rangle \to_{Aexp} ?} \quad \dfrac{x \in \mathrm{dom}(\sigma), \sigma(x) = 6}{\langle x, \sigma \rangle \to_{Aexp} 6} \quad \dfrac{}{\langle x, \sigma \rangle \to_{Aexp} 6} \quad \dfrac{}{\langle 3, \sigma \rangle \to_{Aexp} [\![3]\!]} \quad \dfrac{}{\langle 3, \sigma \rangle \to_{Aexp} 3}}{\langle x + 3, \sigma \rangle \to_{Aexp} ? + ?} \quad \dfrac{\langle x, \sigma \rangle \to_{Aexp} 6 \qquad \langle 3, \sigma \rangle \to_{Aexp} 3}{\langle x + 3, \sigma \rangle \to_{Aexp} 6 + 39}$$

Längere Beispiel-Ableitungen

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

$$\frac{\dfrac{\dfrac{x \in \mathrm{dom}(\sigma), \sigma(x) = 6}{\langle x, \sigma \rangle \to_{Aexp} 6} \quad \dfrac{y \in \mathrm{dom}(\sigma), \sigma(y) = 5}{\langle y, \sigma \rangle \to_{Aexp} 5}}{\langle x + y, \sigma \rangle \to_{Aexp} 11} \quad \dfrac{\dfrac{x \in \mathrm{dom}(\sigma), \sigma(x) = 6}{\langle x, \sigma \rangle \to_{Aexp} 6} \quad \dfrac{y \in \mathrm{dom}(\sigma), \sigma(y) = 5}{\langle y, \sigma \rangle \to_{Aexp} 5}}{\langle x - y, \sigma \rangle \to_{Aexp} 1}}{\langle (x + y) * (x - y), \sigma \rangle \to_{Aexp} 11}$$

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

Arbeitsblatt 2.2: Auswertung

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

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 $a \stackrel{def}{=}$y+3/x

$$\langle a, \langle y \mapsto 5 \rangle \rangle \to_{Aexp} ??? \quad (1)$$
$$\langle a, \langle y \mapsto 5, x \mapsto 0 \rangle \rangle \to_{Aexp} ??? \quad (2)$$

- In diesen Beispielen läßt sich kein **vollständiger** Ableitungsbaum konstruieren.
- Die Auswertung ist **undefiniert** — die Semantik ist **partiell**.

Operationale Semantik: Boolesche Ausdrücke

- **Bexp** $b ::= \mathbf{1} \mid \mathbf{0} \mid a_1 == a_2 \mid a_1 < a_2 \mid !b \mid b_1 \,\&\&\, b2 \mid b_1 \mid\mid b_2$

$$\langle b, \sigma\rangle \to_{Bexp} true \mid false$$

Regeln:

$$\frac{}{\langle \mathbf{1}, \sigma\rangle \to_{Bexp} true} \qquad \frac{}{\langle \mathbf{0}, \sigma\rangle \to_{Bexp} false}$$

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

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

Operationale Semantik: Boolesche Ausdrücke

- **Bexp** $b ::= \mathbf{1} \mid \mathbf{0} \mid a_1 == a_2 \mid a_1 < a_2 \mid !b \mid b_1 \,\&\&\, b2 \mid b_1 \mid\mid b_2$

$$\langle b, \sigma\rangle \to_{Bexp} true \mid false$$

Regeln:

$$\frac{\langle b, \sigma\rangle \to_{Bexp} true}{\langle !b, \sigma\rangle \to_{Bexp} false} \qquad \frac{\langle b, \sigma\rangle \to_{Bexp} false}{\langle !b, \sigma\rangle \to_{Bexp} true}$$

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

$$\frac{\langle b_1, \sigma\rangle \to_{Bexp} true}{\langle b_1 \mid\mid b_2, \sigma\rangle \to_{Bexp} true} \qquad \frac{\langle b_1, \sigma\rangle \to_{Bexp} false \qquad \langle b_2, \sigma\rangle \to_{Bexp} t}{\langle b_1 \mid\mid b_2, \sigma\rangle \to_{Bexp} t}$$

Arbeitsblatt 2.3: Boolsche 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 || im **ersten** Argument.
  - Operational nennt man das auch abgekürzte Auswertung (*short-circuit evaluation*)
  - Das erlaubt Idiome wie **if** (x != 0 && 3/x > 1) { ... }
- Wie erkennt man Striktheit an den **Regeln**?
  Alle Variablen der Konklusion kommen in den Bedingungen vor.

Operationale Semantik: Anweisungen

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

Beispiel:

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

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

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

Regeln:

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

$$\frac{\langle c_1, \sigma\rangle \to_{Stmt} \sigma' \qquad \langle c_2, \sigma'\rangle \to_{Stmt} \sigma''}{\langle c_1; c_2, \sigma\rangle \to_{Stmt} \sigma''}$$

$$\frac{\langle b, \sigma\rangle \to_{Bexp} true \qquad \langle c_1, \sigma\rangle \to_{Stmt} \sigma'}{\langle \mathbf{if}\ (b)\ c_1\ \mathbf{else}\ c_2, \sigma\rangle \to_{Stmt} \sigma'} \qquad \frac{\langle b, \sigma\rangle \to_{Bexp} false \qquad \langle c_2, \sigma\rangle \to_{Stmt} \sigma'}{\langle \mathbf{if}\ (b)\ c_1\ \mathbf{else}\ c_2, \sigma\rangle \to_{Stmt} \sigma'}$$

Operationale Semantik: Anweisungen

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

Regeln:

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

$$\frac{\langle b, \sigma\rangle \to_{Bexp} true \qquad \langle c, \sigma\rangle \to_{Stmt} \sigma' \qquad \langle \mathbf{while}\ (b)\ c, \sigma'\rangle \to_{Stmt} \sigma''}{\langle \mathbf{while}\ (b)\ c, \sigma\rangle \to_{Stmt} \sigma''}$$

Beispiel

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

$\sigma \stackrel{def}{=} \langle y \mapsto 2\rangle$

$$\dfrac{\dfrac{}{\langle 1,\sigma\rangle \to_{Aexp} 1}\quad \dfrac{\dfrac{}{\langle y,\sigma_1\rangle \to_{Aexp} 2}\quad \dfrac{}{\langle y!=0,\sigma_1\rangle \to_{Bexp} true}\quad \dfrac{}{\langle y=y-1;x=2*x,\sigma_1\rangle \to_{Stmt}?}\ (A)}{\langle \textbf{while}\ (y!=0)\ \{y=y-1;x=2*x\},\sigma_1\rangle \to_{Stmt}?}\quad \dfrac{}{\langle w,?\rangle \to_{Stmt}?}\ (B)}{\langle x=1;\underbrace{\textbf{while}\ (y!=0)\ \{y=y-1;x=2*x\}}_{w},\sigma\rangle \to_{Stmt}?}$$

(A)

$$\dfrac{\dfrac{}{\langle y-1,\sigma_1\rangle \to_{Aexp} 1}\quad \dfrac{}{\langle 2*x,\sigma_2\rangle \to_{Aexp} 2}}{\dfrac{}{\langle y=y-1;x=2*x,\sigma_1\rangle \to_{Stmt} \sigma_1[y\mapsto 1]:=\sigma_2}\quad \dfrac{}{\langle x=2*x,\sigma_2\rangle \to_{Stmt} \sigma_2[x\mapsto 2]:=\sigma_3}}{\langle y=y-1;x=2*x,\sigma_1\rangle \to_{Stmt} \sigma_3}$$

$$\dfrac{\dfrac{}{\langle 1,\sigma\rangle \to_{Aexp} 1}\quad \dfrac{\dfrac{}{\langle y,\sigma_1\rangle \to_{Aexp} 2}\quad \dfrac{}{\langle y!=0,\sigma_1\rangle \to_{Bexp} true}\quad \dfrac{}{\langle y=y-1;x=2*x,\sigma_1\rangle \to_{Stmt} \sigma_3}\ (A)}{\langle \textbf{while}\ (y!=0)\ \{y=y-1;x=2*x\},\sigma_1\rangle \to_{Stmt}?}\quad \dfrac{}{\langle w,\sigma_3\rangle \to_{Stmt}?}\ (B)}{\langle x=1;\underbrace{\textbf{while}\ (y!=0)\ \{y=y-1;x=2*x\}}_{w},\sigma\rangle \to_{Stmt}?}$$

(B)

$$\dfrac{\dfrac{}{\langle y,\sigma_3\rangle \to_{Aexp} 1}\quad \dfrac{}{\langle y!=0,\sigma_3\rangle \to_{Bexp} true}}{\dfrac{\dfrac{}{\langle y-1,\sigma_3\rangle \to_{Aexp} 0}\quad \dfrac{}{\langle 2*x,\sigma_4\rangle \to_{Aexp} 4}}{\dfrac{}{\langle y=y-1;x=2*x,\sigma_3\rangle \to_{Stmt} \sigma_3[y\mapsto 0]:=\sigma_4}\quad \dfrac{}{\langle x=2*x,\sigma_4\rangle \to_{Stmt} \sigma_4[x\mapsto 4]:=\sigma_5}}{\langle y=y-1;x=2*x,\sigma_3\rangle \to_{Stmt} \sigma_5}\quad \dfrac{}{\langle w,\sigma_5\rangle \to_{Stmt} \sigma_5}\ (C)}{\langle w,\sigma_3\rangle \to_{Stmt} \sigma_5}$$

$$\left.\dfrac{\dfrac{}{\langle y,\sigma_5\rangle \to_{Aexp} 0}\quad \dfrac{}{\langle y!=0,\sigma_3\rangle \to_{Bexp} false}}{\langle w,\sigma_5\rangle \to_{Stmt} \sigma_5}\right\}(C)$$

$$\underbrace{\textbf{while}\ (y!=0)\ \{y=y-1;x=2*x\}}_{w}$$

$$\dfrac{\dots\quad \dfrac{\dfrac{}{\langle y,\sigma_1\rangle \to_{Aexp} 2}\quad \dfrac{}{\langle y!=0,\sigma_1\rangle \to_{Bexp} true}\quad \dfrac{}{\langle y=y-1;x=2*x,\sigma_1\rangle \to_{Stmt} \sigma_3}\ (A)}{\langle \textbf{while}\ (y!=0)\ \{y=y-1;x=2*x\},\sigma_1\rangle \to_{Stmt} \sigma_5}\quad \dfrac{}{\langle w,\sigma_3\rangle \to_{Stmt} \sigma_5}\ (B)}{\langle x=1;\underbrace{\textbf{while}\ (y!=0)\ \{y=y-1;x=2*x\}}_{w},\sigma\rangle \to_{Stmt} \sigma_5}$$

$$\begin{aligned}\sigma_5 &= \sigma_4[x\mapsto 4] = \sigma_3[y\mapsto 0][x\mapsto 4] = \sigma_2[x\mapsto 2][y\mapsto 0][x\mapsto 4]\\ &= \sigma_1[y\mapsto 1][x\mapsto 2][y\mapsto 0][x\mapsto 4] = \langle y\mapsto 2\rangle[y\mapsto 1][x\mapsto 2][y\mapsto 0][x\mapsto 4]\\ &= \langle y\mapsto 0, x\mapsto 4\rangle\end{aligned}$$

und es gilt $\sigma_5(x) = 4 = 2^2 = 2^{\sigma_1(y)}$

Lineare, abgekürzte Schreibweise

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


Lineare, abgekürzte Schreibweise

```
// ⟨y ↦ 2⟩
x = 1;                                  // Ableitung für ⟨x = 1, ⟨y ↦ 2⟩⟩ →Stmt ⟨y ↦ 2, x ↦ 1⟩
// ⟨y ↦ 2, x ↦ 1⟩
while(y!=0) // ⟨y! = 0, ⟨y ↦ 2, x ↦ 1⟩⟩ →Bexp true
|           y = y - 1;                 // Ableitung für ⟨y = y − 1, ⟨y ↦ 2, x ↦ 1⟩⟩ →Stmt ⟨y ↦
1, x ↦ 1⟩
|           // ⟨y ↦ 1, x ↦ 1⟩
|           x = 2 * x;                 // Ableitung für ⟨x = 2 ∗ x, ⟨y ↦ 1, x ↦ 1⟩⟩ →Stmt ...
|           // ⟨y ↦ 1, x ↦ 2⟩
while (y != 0) {
 y = y - 1;
 x = 2 * x;
}
```


Lineare, abgekürzte Schreibweise

```
// ⟨y ↦ 2⟩
x = 1;
// ⟨y ↦ 2, x ↦ 1⟩
while(y!=0) // ⟨y! = 0, ⟨y ↦ 2, x ↦ 1⟩⟩ →Bexp true
|           y = y - 1;                 // Ableitung für y = y − 1
|           // ⟨y ↦ 1, x ↦ 1⟩
|           x = 2 * x;                 // Ableitung für x = 2 ∗ x
|           // ⟨y ↦ 1, x ↦ 2⟩
while(y!=0) // ⟨y! = 0, ⟨y ↦ 1, x ↦ 2⟩⟩ →Bexp true
|           y = y - 1;
|           // ⟨y ↦ 0, x ↦ 2⟩
|           x = 2 * x;
|           // ⟨y ↦ 0, x ↦ 4⟩
while(y!=0) // ⟨y! = 0, ⟨y ↦ 0, x ↦ 4⟩⟩ →Bexp false
// ⟨y ↦ 0, x ↦ 4⟩
```

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 $\langle x \mapsto 5, y \mapsto 2 \rangle$
- 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?

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


Äquivalenz arithmetischer Ausdrücke

Gegeben zwei Aexp a_1 and a_2

- Sind sie gleich?

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

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

- Wann sind sie gleich?

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

```
x*x     und     8*x+9
x*x     und     x*x+1
```


Äquivalenz Boolscher Ausdrücke

Gegeben zwei Bexp-Ausdrücke b_1 and b_2

- Sind sie gleich?

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

```
A || (A && B)     und     A
```


Beweisen

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

Definition

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

Ein einfaches Beispiel:

Lemma

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

Beweis

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

Beweis

1. $\langle b, \sigma \rangle \to_{Bexp} \mathit{false}$:

$$\langle \textbf{while } (b)\ c, \sigma \rangle \to_{Stmt} \sigma$$
$$\langle \textbf{if } (b)\ \{c; w\} \textbf{ else } \{\}, \sigma \rangle \to_{Stmt} \langle \{\}, \sigma \rangle \to_{Stmt} \sigma$$

2. $\langle b, \sigma \rangle \to_{Bexp} \mathit{true}$: Sei $\langle c, \sigma \rangle \to_{Stmt} \sigma'$, dann:

$$\langle \overbrace{\textbf{while } (b)\ c}^{w}, \sigma \rangle \to_{Stmt} \langle c, \sigma \rangle \to_{Stmt} \sigma'$$
$$\langle w, \sigma' \rangle \to_{Stmt} \sigma''$$
$$\langle \textbf{if } (b)\ \{c; w\} \textbf{ else } \{\}, \sigma \rangle \to_{Stmt} \langle \{c; w\}, \sigma \rangle \to_{Stmt} \langle c, \sigma \rangle \to_{Stmt} \sigma'$$
$$\langle w, \sigma' \rangle \to_{Stmt} \sigma''$$

3. $\langle b, \sigma \rangle$ wertet gar nicht aus — dann werten weder w noch **if** $(b)\ \{c; w\}$ **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, Boolschen Werten)
  - und (zu gegebenen Zustand) Programme zu Zuständen
- Fragen zu Programmen: Gleichheit