Universität Bremen

Deutsches Forschungszentrum für Künstliche Intelligenz
German Research Center for Artificial Intelligence

Systems of High Safety and Security

Lecture 8 from 02.12.2025: Axiomatic Semantics: Specifying Correctness

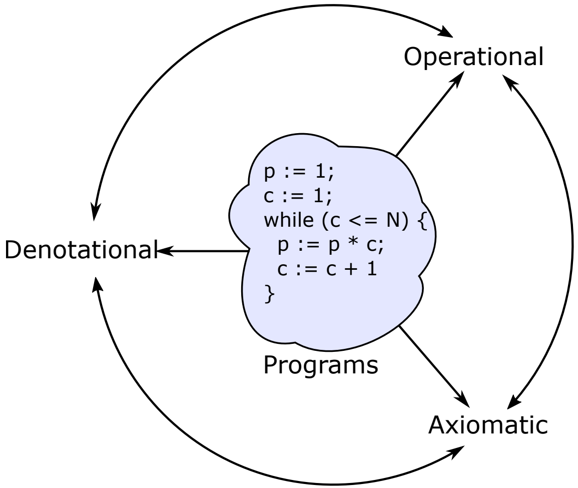
Winter term 2025/26

Christoph Lüth

Roadmap

- Introduction
- Legal Requirements - Norms and Standards
- The Development Process
- Hazard Analysis
- The Big Picture: Hybrid Systems
- Temporal Logic with LTL and CTL
- Operational Semantics
- Axiomatic Semantics - Specifying Correctness
- Floyd-Hoare Logic
- A Simple Compiler and its Correctness
- Hardware Verification
- A Simple TinyRV32 Core
- Conclusions

Three Semantics — One View



Denotational Semantics

Denotationale Semantik — Motivation

- **Operationale Semantik:**

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

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

- **Denotationale Semantik:**

  Eine Menge von Regeln, die ein Programm in eine $\underbrace{\textbf{partielle Funktion}}_{\text{Denotat}}$ von Zustand nach Zustand überführen

$$[\![c]\!]_{\mathcal{C}} : \Sigma \rightharpoonup \Sigma$$

Denotationale Semantik — 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 Funktionen und ihre Graphen

- Der **Graph** einer partiellen Funktion $f : X \rightharpoonup Y$ ist eine Relation

$$grph(f) \subseteq X \times Y \stackrel{def}{=} \{(x, f(x)) \mid x \in \mathrm{dom}(f)\}$$

- Wir können eine partielle Funktion durch ihren Graph definieren:

Definition (Partielle Funktion)

Eine **partielle Funktion** $f : X \rightharpoonup Y$ 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**)

- Wir benutzen beide Notationen, aber für die denotationale Semantik die Graph-Notation.
- **Systemzustände** sind partielle Abbildungen $\Sigma \stackrel{def}{=} \mathbf{Loc} \rightharpoonup \mathbb{Z}$ ($\longrightarrow$ letzte VL)

Denotierende Funktionen (Denotate)

- Arithmetische Ausdrücke: $a \in \mathbf{Exp}$ denotieren eine partielle Funktion $\Sigma \rightharpoonup \mathbb{Z}$
- Boolsche Ausdrücke: $b \in \mathbf{Exp}$ denotieren eine partielle Funktion $\Sigma \rightharpoonup \mathbb{B}$
- Anweisungen: $c \in \mathbf{Stmt}$ denotieren eine partielle Funktion $\Sigma \rightharpoonup \Sigma$

Denotat von arithmetischen Ausdrücken

$$[\![a]\!]_{\mathcal{A}} : \textbf{Exp} \to (\Sigma \rightharpoonup \mathbb{Z})$$

$$\begin{aligned}
[\![n]\!]_{\mathcal{A}} =& \{(\sigma, [\![n]\!]) \mid \sigma \in \Sigma\} \\
[\![x]\!]_{\mathcal{A}} =& \{(\sigma, \sigma(x)) \mid \sigma \in \Sigma, x \in \mathrm{dom}(\sigma)\} \\
[\![a_0 + a_1]\!]_{\mathcal{A}} =& \{(\sigma, n_0 + n_1) \mid (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}}\} \\
[\![a_0 - a_1]\!]_{\mathcal{A}} =& \{(\sigma, n_0 - n_1) \mid (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}}\} \\
[\![a_0 * a_1]\!]_{\mathcal{A}} =& \{(\sigma, n_0 * n_1) \mid (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}}\} \\
[\![a_0 / a_1]\!]_{\mathcal{A}} =& \{(\sigma, n_0 \div n_1) \mid (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}} \wedge n_1 \neq 0\}
\end{aligned}$$

Denotat von Boolschen Ausdrücken: Relationen

$$[\![a]\!]_{\mathcal{B}} : \textbf{Exp} \to (\Sigma \rightharpoonup \mathbb{B})$$

$$\begin{aligned}
[\![\mathit{true}]\!]_{\mathcal{B}} =& \{(\sigma, \mathit{true}) \mid \sigma \in \Sigma\} \\
[\![\mathit{false}]\!]_{\mathcal{B}} =& \{(\sigma, \mathit{false}) \mid \sigma \in \Sigma\} \\
[\![a_0 == a_1]\!]_{\mathcal{B}} =& \{(\sigma, \mathit{true}) \mid \sigma \in \Sigma, (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}}, (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}}, n_0 = n_1\} \\
& \cup \{(\sigma, \mathit{false}) \mid \sigma \in \Sigma, (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}}, (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}}, n_0 \neq n_1\} \\
[\![a_0 < a_1]\!]_{\mathcal{B}} =& \{(\sigma, \mathit{true}) \mid \sigma \in \Sigma, (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}}, (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}}, n_0 < n_1\} \\
& \cup \{(\sigma, \mathit{false}) \mid \sigma \in \Sigma, (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}}, (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}}, n_0 \geq n_1\}
\end{aligned}$$

Denotat von Boolschen Ausdrücken: Konnektive

$$[\![a]\!]_{\mathcal{B}} : \mathbf{Exp} \to (\Sigma \rightharpoonup \mathbb{B})$$

$$\begin{aligned}
[\![!b]\!]_{\mathcal{B}} &= \{(\sigma, \mathit{true}) \mid \sigma \in \Sigma, (\sigma, \mathit{false}) \in [\![b]\!]_{\mathcal{B}}\} \\
&\quad \cup \{(\sigma, \mathit{false}) \mid \sigma \in \Sigma, (\sigma, \mathit{true}) \in [\![b]\!]_{\mathcal{B}}\} \\
[\![b_1 \,\&\&\, b2]\!]_{\mathcal{B}} &= \{(\sigma, \mathit{false}) \mid \sigma \in \Sigma, (\sigma, \mathit{false}) \in [\![b_1]\!]_{\mathcal{B}}\} \\
&\quad \cup \{(\sigma, t_2) \mid \sigma \in \Sigma, (\sigma, \mathit{true}) \in [\![b_1]\!]_{\mathcal{B}}, (\sigma, t_2) \in [\![b_2]\!]_{\mathcal{B}}\} \\
[\![b_1 \mid\mid b_2]\!]_{\mathcal{B}} &= \{(\sigma, \mathit{true}) \mid \sigma \in \Sigma, (\sigma, \mathit{true}) \in [\![b_1]\!]_{\mathcal{B}}\} \\
&\quad \cup \{(\sigma, t_2) \mid \sigma \in \Sigma, (\sigma, \mathit{false}) \in [\![b_1]\!]_{\mathcal{B}}, (\sigma, t_2) \in [\![b_2]\!]_{\mathcal{B}}\}
\end{aligned}$$

Kompositionalität und Striktheit

Lemma (Partielle Funktion)

(i) $[\![-]\!]_{\mathcal{A}}$ *ist rechtseindeutig und damit eine **partielle Funktion**.*

(ii) $[\![-]\!]_{\mathcal{B}}$ *ist rechtseindeutig und damit eine **partielle Funktion**.*

- Beweis durch **strukturelle Induktion** über e.
- Ist $[\![-]\!]_{\mathcal{A}}$ strikt?

Kompositionalität und Striktheit

Lemma (Partielle Funktion)

(i) $[\![-]\!]_{\mathcal{A}}$ *ist rechtseindeutig und damit eine* ***partielle Funktion****.*

(ii) $[\![-]\!]_{\mathcal{B}}$ *ist rechtseindeutig und damit eine* ***partielle Funktion****.*

- Beweis durch **strukturelle Induktion** über e.
- Ist $[\![-]\!]_{\mathcal{A}}$ strikt? Ja — es werden immer alle Argumente ausgewertet.
- Ist $[\![-]\!]_{\mathcal{B}}$ strikt?

Kompositionalität und Striktheit

Lemma (Partielle Funktion)

1. $[\![-]\!]_{\mathcal{A}}$ *ist rechtseindeutig und damit eine* ***partielle Funktion***.
2. $[\![-]\!]_{\mathcal{B}}$ *ist rechtseindeutig und damit eine* ***partielle Funktion***.

- Beweis durch **strukturelle Induktion** über e.
- Ist $[\![-]\!]_{\mathcal{A}}$ strikt? Ja — es werden immer alle Argumente ausgewertet.
- Ist $[\![-]\!]_{\mathcal{B}}$ strikt? Natürlich nicht:
- Sei $[\![b_1]\!]_{\mathcal{B}}(\sigma) = \mathit{false}$, dann $[\![b_1 \;\&\&\; b_2]\!]_{\mathcal{B}}(\sigma) = [\![b_1]\!]_{\mathcal{B}}(\sigma) = \mathit{false}$

Kompositionalität und Striktheit

Lemma (Partielle Funktion)

(i) $[\![-]\!]_{\mathcal{A}}$ *ist rechtseindeutig und damit eine* ***partielle Funktion****.*

(ii) $[\![-]\!]_{\mathcal{B}}$ *ist rechtseindeutig und damit eine* ***partielle Funktion****.*

- Beweis durch **strukturelle Induktion** über e.
- Ist $[\![-]\!]_{\mathcal{A}}$ strikt? Ja — es werden immer alle Argumente ausgewertet.
- Ist $[\![-]\!]_{\mathcal{B}}$ strikt? Natürlich nicht:
- Sei $[\![b_1]\!]_{\mathcal{B}}(\sigma) = \mathit{false}$, dann $[\![b_1 \;\&\&\; b_2]\!]_{\mathcal{B}}(\sigma) = [\![b_1]\!]_{\mathcal{B}}(\sigma) = \mathit{false}$
- Deshalb **nicht** $[\![b_1 \;\&\&\; b_2]\!]_{\mathcal{B}}(\sigma) = [\![b_1]\!]_{\mathcal{B}}(\sigma) \wedge [\![b_2]\!]_{\mathcal{B}}(\sigma)$
- Logische Operatoren müssen links nicht-strikt sein.

Denotat von Stmt

$$[\![.]\!]_{\mathcal{C}} : \textbf{Stmt} \to (\Sigma \rightharpoonup \Sigma)$$

$$
\begin{aligned}
[\![x = a]\!]_{\mathcal{C}} &= \{(\sigma, \sigma[x \mapsto n]) \mid \sigma \in \Sigma \wedge (\sigma, n) \in [\![a]\!]_{\mathcal{A}}\} \\
[\![c_1 ; c_2]\!]_{\mathcal{C}} &= [\![c_1]\!]_{\mathcal{C}} \circ [\![c_2]\!]_{\mathcal{C}} && \text{Komposition von Relationen} \\
[\![\textbf{nil}]\!]_{\mathcal{C}} &= \textbf{Id}_{\Sigma} && \textbf{Id} := \{(\sigma, \sigma) | \sigma \in \Sigma\} \\
[\![\textbf{if } (b) \textbf{ then } c_0 \textbf{ else } c_1]\!]_{\mathcal{C}} &= \{(\sigma, \sigma') \mid (\sigma, \mathit{true}) \in [\![b]\!]_{\mathcal{B}} \wedge (\sigma, \sigma') \in [\![c_0]\!]_{\mathcal{C}}\} \\
&\quad \cup \{(\sigma, \sigma') \mid (\sigma, \mathit{false}) \in [\![b]\!]_{\mathcal{B}} \wedge (\sigma, \sigma') \in [\![c_1]\!]_{\mathcal{C}}\}
\end{aligned}
$$

Denotat von Stmt

$$[\![.]\!]_{\mathcal{C}} : \textbf{Stmt} \to (\Sigma \rightharpoonup \Sigma)$$

$$\begin{aligned}
[\![x = a]\!]_{\mathcal{C}} &= \{(\sigma, \sigma[x \mapsto n]) \mid \sigma \in \Sigma \land (\sigma, n) \in [\![a]\!]_{\mathcal{A}}\} \\
[\![c_1; c_2]\!]_{\mathcal{C}} &= [\![c_1]\!]_{\mathcal{C}} \circ [\![c_2]\!]_{\mathcal{C}} && \text{Komposition von Relationen} \\
[\![\textbf{nil}]\!]_{\mathcal{C}} &= \textbf{Id}_{\Sigma} && \textbf{Id} := \{(\sigma, \sigma) | \sigma \in \Sigma\} \\
[\![\textbf{if } (b) \textbf{ then } c_0 \textbf{ else } c_1]\!]_{\mathcal{C}} &= \{(\sigma, \sigma') \mid (\sigma, \mathit{true}) \in [\![b]\!]_{\mathcal{B}} \land (\sigma, \sigma') \in [\![c_0]\!]_{\mathcal{C}}\} \\
&\quad \cup \{(\sigma, \sigma') \mid (\sigma, \mathit{false}) \in [\![b]\!]_{\mathcal{B}} \land (\sigma, \sigma') \in [\![c_1]\!]_{\mathcal{C}}\}
\end{aligned}$$

Aber was ist

$$[\![\textbf{while } (b) \; c]\!]_{\mathcal{C}} =$$

Denotat von Stmt

$$[\![.]\!]_{\mathcal{C}} : \textbf{Stmt} \to (\Sigma \rightharpoonup \Sigma)$$

$$\begin{aligned}
[\![x = a]\!]_{\mathcal{C}} =& \{(\sigma, \sigma[x \mapsto n]) \mid \sigma \in \Sigma \land (\sigma, n) \in [\![a]\!]_{\mathcal{A}}\} \\
[\![c_1; c_2]\!]_{\mathcal{C}} =& [\![c_1]\!]_{\mathcal{C}} \circ [\![c_2]\!]_{\mathcal{C}} && \text{Komposition von Relationen} \\
[\![\textbf{nil}]\!]_{\mathcal{C}} =& \textbf{Id}_{\Sigma} && \textbf{Id} := \{(\sigma, \sigma) | \sigma \in \Sigma\} \\
[\![\textbf{if}\ (b)\ \textbf{then}\ c_0\ \textbf{else}\ c_1]\!]_{\mathcal{C}} =& \{(\sigma, \sigma') \mid (\sigma, \mathit{true}) \in [\![b]\!]_{\mathcal{B}} \land (\sigma, \sigma') \in [\![c_0]\!]_{\mathcal{C}}\} \\
& \cup \{(\sigma, \sigma') \mid (\sigma, \mathit{false}) \in [\![b]\!]_{\mathcal{B}} \land (\sigma, \sigma') \in [\![c_1]\!]_{\mathcal{C}}\}
\end{aligned}$$

Aber was ist

$$\begin{aligned}
[\![\textbf{while}\ (b)\ c]\!]_{\mathcal{C}} = & \{(\sigma, \sigma') \mid (\sigma, \mathit{true}) \in [\![b]\!]_{\mathcal{B}} \land (\sigma, \sigma') \in [\![c]\!]_{\mathcal{C}} \circ [\![\textbf{while}\ (b)\ c]\!]_{\mathcal{C}}\} \\
& \cup \{(\sigma, \sigma) \mid (\sigma, \mathit{false}) \in [\![b]\!]_{\mathcal{B}}\}
\end{aligned}$$

Denotat von Stmt

$$[\![.]\!]_{\mathcal{C}} : \textbf{Stmt} \to (\Sigma \rightharpoonup \Sigma)$$

$$\begin{aligned}
[\![x = a]\!]_{\mathcal{C}} &= \{(\sigma, \sigma[x \mapsto n]) \mid \sigma \in \Sigma \wedge (\sigma, n) \in [\![a]\!]_{\mathcal{A}}\} \\
[\![c_1; c_2]\!]_{\mathcal{C}} &= [\![c_1]\!]_{\mathcal{C}} \circ [\![c_2]\!]_{\mathcal{C}} && \text{Komposition von Relationen} \\
[\![\textbf{nil}]\!]_{\mathcal{C}} &= \textbf{Id}_{\Sigma} && \textbf{Id} := \{(\sigma, \sigma) | \sigma \in \Sigma\} \\
[\![\textbf{if } (b) \textbf{ then } c_0 \textbf{ else } c_1]\!]_{\mathcal{C}} &= \{(\sigma, \sigma') \mid (\sigma, \mathit{true}) \in [\![b]\!]_{\mathcal{B}} \wedge (\sigma, \sigma') \in [\![c_0]\!]_{\mathcal{C}}\} \\
&\quad \cup \{(\sigma, \sigma') \mid (\sigma, \mathit{false}) \in [\![b]\!]_{\mathcal{B}} \wedge (\sigma, \sigma') \in [\![c_1]\!]_{\mathcal{C}}\}
\end{aligned}$$

Aber was ist

$$\begin{aligned}
[\![\textbf{while } (b) \ c]\!]_{\mathcal{C}} = \ & \{(\sigma, \sigma') \mid (\sigma, \mathit{true}) \in [\![b]\!]_{\mathcal{B}} \wedge (\sigma, \sigma') \in [\![c]\!]_{\mathcal{C}} \circ [\![\textbf{while } (b) \ c]\!]_{\mathcal{C}}\} \\
& \cup \{(\sigma, \sigma) \mid (\sigma, \mathit{false}) \in [\![b]\!]_{\mathcal{B}}\}
\end{aligned}$$

Problem: rekursive Definition, Konstruktion über **Fixpunkt**.

Equivalence of Semantics

- Für alle $c \in \mathbf{Stmt}$, für alle Zustände σ, σ':

$$\langle c, \sigma \rangle \to_{Stmt} \sigma' \Longleftrightarrow (\sigma, \sigma') \in [\![c]\!]_{\mathcal{C}}$$

- Für alle $e \in \mathbf{Exp}$, für alle Zustände σ:

$$\langle e, \sigma \rangle \to_{Exp} v \Longleftrightarrow (\sigma, v) \in [\![e]\!]_{\mathcal{A}} (v \in \mathbb{Z})$$
$$\langle e, \sigma \rangle \to_{Exp} v \Longleftrightarrow (\sigma, v) \in [\![e]\!]_{\mathcal{B}} (v \in \mathbb{B})$$

- Beweis durch Induktion über Struktur von e, c und Ableitung von $\langle c, \sigma \rangle \to_{Stmt} \sigma'$

Axiomatic Semantics

Axiomatic Semantics: Idea

- What is calculated?

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

Axiomatic Semantics: Idea

- What is calculated? $p = n!$
- Why? How can we **prove** it?

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

Axiomatic Semantics: Idea

- What is calculated? $p = n!$
- Why? How can we **prove** it?
- We need to calculate the state **symbolically**.

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

Axiomatic Semantics: Idea

- What is calculated? $p = n!$
- Why? How can we **prove** it?
- We need to calculate the state **symbolically**.

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

- Operational/denotional semantics not suitable for **correctness proofs**.

Axiomatic Semantics: Idea

- What is calculated? $p = n!$
- Why? How can we **prove** it?
- We need to calculate the state **symbolically**.

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

- Operational/denotional semantics not suitable for **correctness proofs**.
- Basic principle:
  1. Abstraction
  2. State-dependent **assertions**
  3. Calculating validity of assertions by **rules**.

Foundations of Axiomatic Semantics

```
// (A)
p= 1;
c= 1;
// (B)
while (c <= n) {
  // (C)
  p= p * c;
  c= c + 1;
  // (D)
}
// (E)
```

- **Zusicherungen** über den Zustand
- Beispiele:
  - (B): Hier gilt $p = c = 1$
  - (D): Hier ist c ist um eines größer als der Wert von c an Punkt (C)
- Gesamtaussage: Wenn bei (A) der Wert von $n \geq 0$ ist, dann ist bei (E) $p = n!$

Foundations of Axiomatic Semantics

```
// (A)
p= 1;
c= 1;
// (B)
while (c <= n) {
  // (C)
  p= p * c;
  c= c + 1;
  // (D)
}
// (E)
```

- **Zusicherungen** über den Zustand
- Beispiele:
  - (B): Hier gilt $p = c = 1$
  - (D): Hier ist c ist um eines größer als der Wert von c an Punkt (C)
- Gesamtaussage: Wenn bei (A) der Wert von $n \geq 0$ ist, dann ist bei (E) $p = n!$
- Beobachtung:
  - n ist eine "'Eingabevariable"', der Wert am Anfang des Programmes (A) ist relevant;

Foundations of Axiomatic Semantics

```
// (A)
p= 1;
c= 1;
// (B)
while (c <= n) {
  // (C)
  p= p * c;
  c= c + 1;
  // (D)
}
// (E)
```

- **Zusicherungen** über den Zustand
- Beispiele:
  - (B): Hier gilt $p = c = 1$
  - (D): Hier ist c ist um eines größer als der Wert von c an Punkt (C)
- Gesamtaussage: Wenn bei (A) der Wert von $n \geq 0$ ist, dann ist bei (E) $p = n!$
- Beobachtung:
  - n ist eine "'Eingabevariable"', der Wert am Anfang des Programmes (A) ist relevant;
  - p ist eine "'Ausgabevariable"', der Wert am Ende des Programmes (E) ist relevant;

Foundations of Axiomatic Semantics

```
// (A)
p= 1;
c= 1;
// (B)
while (c <= n) {
  // (C)
  p= p * c;
  c= c + 1;
  // (D)
}
// (E)
```

- **Zusicherungen** über den Zustand
- Beispiele:
  - (B): Hier gilt $p = c = 1$
  - (D): Hier ist c ist um eines größer als der Wert von c an Punkt (C)
- Gesamtaussage: Wenn bei (A) der Wert von $n \geq 0$ ist, dann ist bei (E) $p = n!$
- Beobachtung:
  - n ist eine "'Eingabevariable"', der Wert am Anfang des Programmes (A) ist relevant;
  - p ist eine "'Ausgabevariable"', der Wert am Ende des Programmes (E) ist relevant;
  - c ist eine "'Arbeitsvariable"', der Wert am Anfang und Ende ist irrelevant

Was berechnet dieses Programm?

```
// (A)
x= 1;
c= 1;
// (B)
while (c <= y) {
  // (C)
  x= 2*x;
  c= c+1;
  // (D)
 }
// (E)
```

Betrachtet nebenstehendes Programm.
Analog zu dem Beispiel auf der vorherigen Folie:

1. Was berechnet das Programm?
2. Welches sind "‘Eingabevariablen"’, welches "‘Ausgabevariablen"’, welches sind "‘Arbeitsvariablen"’?
3. Welche Zusicherungen und Zusammenhänge gelten zwischen den Variablen an den Punkten (A) bis (E)?

Towards Axiomatic Semantics

- Kern der Floyd-Hore-Logik sind **zustandsabhängige Aussagen**
- Aber: wie können wir Aussagen **jenseits** des Zustandes treffen?
- Einfaches Beispiel:

```
x = x+ 1;
```

- Der Wert von x wird um 1 erhöht
- Der Wert von x ist hinterher größer als vorher

Towards Axiomatic Semantics

- Kern der Floyd-Hore-Logik sind **zustandsabhängige Aussagen**
- Aber: wie können wir Aussagen **jenseits** des Zustandes treffen?
- Einfaches Beispiel:

```
x = x+ 1;
```

  - Der Wert von x wird um 1 erhöht
  - Der Wert von x ist hinterher größer als vorher
- Wir benötigen **zustandsfreie** Aussagen, um von Zuständen unabhängig **vergleichen** zu können.
- Die Logik **abstrahiert** den Effekt von Programmen.

Basic Building Blocks

- **Logische Variablen** (zustandsfrei) und **Programmvariablen** (zustandsabhängig)
- **Zusicherungen** mit logischen und Programmvariablen
- **Floyd-Hoare-Tripel** $\{P\}\, c\, \{Q\}$
  - Vorbedingung P (Zusicherung)
  - Programm c
  - Nachbedingung Q (Zusicherung)
- Floyd-Hoare-Logik abstrahiert von Programmen zu logischen Formeln.

Assertions

- Erweiterung von **Exp** durch
  - **Logische** Variablen **Var** $v := N, M, L, U, V, X, Y, Z$
  - Definierte Funktionen und Prädikate $n!, x^y, \ldots$
  - Implikation und Quantoren $b_1 \longrightarrow b_2, \forall v.\, b, \exists v.\, b$
- Formal:

$$
\begin{aligned}
\textbf{Aexpv}\quad a &::= \mathbb{Z} \mid \textbf{Idt} \mid \textbf{Var} \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2 \mid a_1 / a_2 \\
&\quad \mid f(e_1, \ldots, e_n) \\
\textbf{Assn}\quad b &::= \mathit{true} \mid \mathit{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \\
&\quad \mid \neg b \mid b_1 \wedge b2 \mid b_1 \vee b_2 \\
&\quad \mid b_1 \longrightarrow b_2 \mid p(e_1, \ldots, e_n) \mid \forall v.\, b \mid \exists v.\, b
\end{aligned}
$$

Denotationale Semantik von Zusicherungen

- Erste Näherung: Funktion

$$\llbracket a \rrbracket_{\mathcal{A}} : \textbf{Aexpv} \to (\Sigma \rightharpoonup \mathbb{Z})$$
$$\llbracket b \rrbracket_{\mathcal{B}} : \textbf{Assn} \to (\Sigma \rightharpoonup \mathbb{B})$$

  - **Konservative** Erweiterung von $\llbracket a \rrbracket_{\mathcal{A}} : \textbf{Exp} \to (\Sigma \rightharpoonup \mathbb{Z})$
- Aber: was ist mit den logischen Variablen?

Denotationale Semantik von Zusicherungen

- Erste Näherung: Funktion

$$\llbracket a \rrbracket_{\mathcal{A}} : \mathbf{Aexpv} \to (\Sigma \rightharpoonup \mathbb{Z})$$
$$\llbracket b \rrbracket_{\mathcal{B}} : \mathbf{Assn} \to (\Sigma \rightharpoonup \mathbb{B})$$

  - **Konservative** Erweiterung von $\llbracket a \rrbracket_{\mathcal{A}} : \mathbf{Exp} \to (\Sigma \rightharpoonup \mathbb{Z})$
- Aber: was ist mit den logischen Variablen?
- Zusätzlicher Parameter **Belegung** der logischen Variablen $I : \mathbf{Var} \to \mathbb{Z}$

$$\llbracket a \rrbracket_{\mathcal{A}} : \mathbf{Aexpv} \to (\mathbf{Var} \to \mathbb{Z}) \to (\Sigma \rightharpoonup \mathbb{Z})$$
$$\llbracket b \rrbracket_{\mathcal{B}} : \mathbf{Assn} \to (\mathbf{Var} \to \mathbb{Z}) \to (\Sigma \rightharpoonup \mathcal{B})$$

- Bemerkung: $I : \mathbf{Var} \to \mathbb{Z}$ ist immer eine **totale Funktion** im Gegensatz zu einem Zustand.

Denotat von Exp

$$[\![a]\!]_{\mathcal{A}} : \textbf{Exp} \to (\Sigma \rightharpoonup \mathbb{Z})$$

$$\begin{aligned}
[\![n]\!]_{\mathcal{A}} =& \{(\sigma, [\![n]\!]) \mid \sigma \in \Sigma\} \\
[\![x]\!]_{\mathcal{A}} =& \{(\sigma, \sigma(x)) \mid \sigma \in \Sigma, x \in \mathrm{dom}(\sigma)\} \\
[\![a_0 + a_1]\!]_{\mathcal{A}} =& \{(\sigma, n_0 + n_1) \mid (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}}\} \\
[\![a_0 - a_1]\!]_{\mathcal{A}} =& \{(\sigma, n_0 - n_1) \mid (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}}\} \\
[\![a_0 * a_1]\!]_{\mathcal{A}} =& \{(\sigma, n_0 \times n_1) \mid (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}}\} \\
[\![a_0 / a_1]\!]_{\mathcal{A}} =& \{(\sigma, n_0 \div n_1) \mid (\sigma, n_0) \in [\![a_0]\!]_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]_{\mathcal{A}} \wedge n_1 \neq 0\}
\end{aligned}$$

Denotat von Aexpv

$$[\![a]\!]_{\mathcal{A}} : \textbf{Aexpv} \to (\textbf{Var} \to \mathbb{Z}) \to\!\to (\Sigma \rightharpoonup \mathbb{Z})$$

Sei I : **Var** $\to \mathbb{Z}$ eine beliebige Belegung

$$
\begin{aligned}
[\![n]\!]^{I}_{\mathcal{A}} &= \{(\sigma, [\![n]\!]) \mid \sigma \in \Sigma\} \\
[\![x]\!]^{I}_{\mathcal{A}} &= \{(\sigma, \sigma(x)) \mid \sigma \in \Sigma, x \in \mathrm{dom}(\sigma)\} \\
[\![a_0 + a_1]\!]^{I}_{\mathcal{A}} &= \{(\sigma, n_0 + n_1) \mid (\sigma, n_0) \in [\![a_0]\!]^{I}_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]^{I}_{\mathcal{A}}\} \\
[\![a_0 - a_1]\!]^{I}_{\mathcal{A}} &= \{(\sigma, n_0 - n_1) \mid (\sigma, n_0) \in [\![a_0]\!]^{I}_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]^{I}_{\mathcal{A}}\} \\
[\![a_0 * a_1]\!]^{I}_{\mathcal{A}} &= \{(\sigma, n_0 \times n_1) \mid (\sigma, n_0) \in [\![a_0]\!]^{I}_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]^{I}_{\mathcal{A}}\} \\
[\![a_0 / a_1]\!]^{I}_{\mathcal{A}} I &= \{(\sigma, n_0 \div n_1) \mid (\sigma, n_0) \in [\![a_0]\!]^{I}_{\mathcal{A}} \wedge (\sigma, n_1) \in [\![a_1]\!]^{I}_{\mathcal{A}} \wedge n_1 \neq 0\} \\
[\![X]\!]^{I}_{\mathcal{A}} &= \{(\sigma, I(X)) \mid \sigma \in \Sigma, X \in V\}
\end{aligned}
$$

Erfüllung von Zusicherungen

- Wann gilt eine Zusicherung $b \in \mathbf{Assn}$ in einem Zustand σ?
  - Auswertung (denotationale Semantik) ergibt *true*
  - Belegung ist zusätzlicher Parameter

Erfülltheit von Zusicherungen

$b \in \mathbf{Assn}$ ist in Zustand σ mit Belegung I erfüllt ($\sigma \models^I b$), gdw

$$[\![b]\!]^I_{\mathcal{B}}(\sigma) = \mathit{true}$$

Floyd-Hoare-Tripel

Partielle Korrektheit ($\models \{P\}\, c\, \{Q\}$)

$\{P\}\, c\, \{Q\}$ ist **partiell korrekt**, wenn für all Belegungen I und alle Zustände σ, die P erfüllen, gilt: **wenn** die Ausführung von c mit σ in einem Zustand τ terminiert, **dann** erfüllt τ mit Belegung I Q.

$$\models \{P\}\, c\, \{Q\} \Longleftrightarrow \forall I.\, \forall \sigma.\, \sigma \models^I P \wedge \exists \tau.\, (\sigma, \tau) \in [\![c]\!]_{\mathcal{C}} \longrightarrow \tau \models^I Q$$

- Gleiche Belegung der logischen Variablen in P und Q erlaubt **Vergleich** zwischen Zuständen

Totale Korrektheit ($\models [P]\, c\, [Q]$)

$[P]\, c\, [Q]$ ist **total korrekt**, wenn für all Belegungen I und alle Zustande σ, die P erfüllen, die Ausführung von c mit σ in einem Zustand τ terminiert, und τ mit der Belegung I erfüllt Q.

$$\models [P]\, c\, [Q] \Longleftrightarrow \forall I.\, \forall \sigma.\, \sigma \models^I P \longrightarrow \exists \tau.\, (\sigma, \tau) \in [\![c]\!]_{\mathcal{C}} \wedge \tau \models^I Q$$

Beispiele

- Folgendes **gilt**:

$$\models \{true\}\ \mathbf{while}(1)\{\ \}\ \{true\}$$

Beispiele

- Folgendes **gilt**:

$$\models \{true\}\ \textbf{while}(1)\{\ \}\ \{true\}$$

- Folgendes gilt **nicht**:

$$\models [true]\ \textbf{while}(1)\{\ \}\ [true]$$

Beispiele

- Folgendes **gilt**:

$$\models \{\mathit{true}\}\ \textbf{while}(1)\{\ \}\ \{\mathit{true}\}$$

- Folgendes gilt **nicht**:

$$\models [\mathit{true}]\ \textbf{while}(1)\{\ \}\ [\mathit{true}]$$

- Folgende **gelten**:

$$\models \{\mathit{false}\}\ \textbf{while}\ (\mathit{true})\ \textbf{nil}\ \{\mathit{true}\}$$
$$\models [\mathit{false}]\ \textbf{while}\ (\mathit{true})\ \textbf{nil}\ [\mathit{true}]$$

Wegen *ex falso quodlibet*: $\mathit{false} \longrightarrow \phi$

Gültigkeit und Herleitbarkeit

- **Semantische Gültigkeit:** $\models \{P\}\, c\, \{Q\}$
  - Definiert durch denotationale Semantik:

$$\models \{P\}\, c\, \{Q\} \Longleftrightarrow \forall I.\, \forall \sigma.\, \sigma \models^{I} P \land \exists \tau.\, (\sigma, \tau) \in [\![c]\!]_{\mathcal{C}} \longrightarrow \tau \models^{I} Q$$

  - Problem: müssten Semantik von c ausrechnen

Gültigkeit und Herleitbarkeit

- **Semantische Gültigkeit:** $\models \{P\}\, c\, \{Q\}$
  - Definiert durch denotationale Semantik:
    $$\models \{P\}\, c\, \{Q\} \Longleftrightarrow \forall I.\, \forall \sigma.\, \sigma \models^I P \wedge \exists \tau.\, (\sigma, \tau) \in [\![c]\!]_{\mathcal{C}} \longrightarrow \tau \models^I Q$$
  - Problem: müssten Semantik von c ausrechnen
- **Syntaktische Herleitbarkeit:** $\vdash \{P\}\, c\, \{Q\}$
  - Durch **Regeln** definiert
  - Kann **hergeleitet** werden
  - Muss **korrekt** bezüglich semantischer Gültigkeit gezeigt werden
- Generelles Vorgehen in der Logik — **nächste Vorlesung**