Korrekte Software: Grundlagen und Methoden
Vorlesung 8 vom 29.05.18: Modellierung und Spezifikation

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2018

Fahrplan

- Einführung
- Operationale Semantik
- Denotationale Semantik
- Äquivalenz der Operationalen und Denotationalen Semantik
- Die Floyd-Hoare-Logik
- Invarianten und die Korrektheit des Floyd-Hoare-Kalküls
- Strukturierte Datentypen
- Modellierung und Spezifikation
- Verifikationsbedingungen
- Vorwärts mit Floyd und Hoare
- Funktionen und Prozeduren
- Referenzen
- Ausblick und Rückblick

Erstes Beispiel: Ein Feld initialisieren

```
1 // {n ≤ 0}
2 i= 0;
3 while (i< n) {
4   a[i]= i;
5   i= i+1;
6   // {(∀j.0 ≤ j < i ⟶ a[j] = j) ∧ i ≤ n}
7   }
8 // {∀j.0 ≤ j < n ⟶ a[j] = j}
```

- Wichtiges Theorem:

$$(\forall j.\, 0 \le j < n \longrightarrow P[j]) \land P[n] \longrightarrow \forall j.\, 0 \le j < n+1 \longrightarrow P[j]$$

Längeres Beispiel: Suche nach dem maximalen Element

```
1  // {0 < n}
2  i= 0;
3  r= 0;
4  while (i< n) {
5    if (a[r] < a[i]) {
6      r= i;
7      }
8    else {
9      }
10   i= i+1;
11   // {(∀j.0 ≤ j < i ⟶ a[j] ≤ a[r]) ∧ 0 ≤ i ≤ n ∧ 0 ≤ r < n}
12   }
13 // {(∀j.0 ≤ j < n ⟶ a[j] ≤ a[r]) ∧ 0 ≤ r < n}
```


Sortierte Arrays?

- Wie formulieren wir, dass ein Array sortiert ist? Ggf. bis zu einem bestimmten Punkt n sortiert ist?

```
int a[8];
// {∀0 ≤ j ≤ n < 6.a[j] ≤ a[j+1]}
```

- Alternativ würden man auch gerne ein Prädikat definieren können

$//\{\forall a.\mathit{sorteduntil}(a,0) \longleftrightarrow \mathit{true}\}$
$//\{\forall a.\forall i.i \ge 0 \longrightarrow (\mathit{sorteduntil}(a,i+1) \longleftrightarrow (a[i] \le a[i+1] \land \mathit{sorteduntil}(a,i)))\}$

Formelsprache mit Quantoren

- Wir brauchen Programmausdrücken wie **Aexp**
- Wir müssen neue Funktionen verwendenn können
  - Etwa eine Fakultätsfunkion
- Wir müssen neue Prädikate definieren können
  - Etwa einen `sorteduntil`
- Wir müssen Formeln bilden können
  - Analog zu **Bexp**
  - Zusätzlich mit Implikation $\longrightarrow$, Äquivalenz $\longleftrightarrow$
  - Zusätzlich Quantoren über logische Variablen wie in

$$(\forall j.\, 0 \le j < n \longrightarrow P[j]) \land P[n] \longrightarrow \forall j.\, 0 \le j < n+1 \longrightarrow P[j]$$
$$\forall i.i \ge 0 \longrightarrow (\mathit{sorteduntil}(a,i+1) \longleftrightarrow (a[i] \le a[i+1] \land \mathit{sorteduntil}(a,i)))$$

Was brauchen wir?

- Definiere Terme als Variablen und Funktionen besimmter Stelligkeit
- Definiere Literale und Formeln
- Interpretation von Formeln
  - mit und ohne Programmvariablen

Zusicherungen (Assertions)

- Erweiterung von **Aexp** and **Bexp** durch
  - **Logische** Variablen **Var** $v := N, M, L, U, V, X, Y, Z$
  - ~~Definierte Funktionen und Prädikate über **Aexp**~~ ~~$n!, \sum_{i=1}^{n} i, \ldots$~~
  - Funktionen und Prädikate selbst definieren
  - Implikation, Äquivalenzen und Quantoren
    $b_1 \longrightarrow b_2, b_1 \longleftrightarrow b_2, \forall v_{(\mathbf{Z}|\mathbf{C}|\mathrm{Array})}.\, b, \exists v_{(\mathbf{Z}|\mathbf{C}|\mathrm{Array})}.\, b$
- Formal:

$$\begin{aligned}
\textbf{Lexp}\ & l ::= \mathbf{Idt} \mid l[a] \mid l.\mathbf{Idt} \\
\textbf{Aexpv}\ & a ::= \mathbf{Z} \mid \mathbf{Idt} \mid \mathbf{Var} \mid \mathbf{C} \mid \mathbf{Lexp} \\
& \quad \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2 \\
& \quad \mid f(e_1, \ldots, e_n) \\
\textbf{Assn}\ & b ::= \mathbf{1} \mid \mathbf{0} \mid a_1 == a_2 \mid a_1 != a_2 \mid a_1 <= a_2 \\
& \quad \mid !\, b \mid b_1 \,\&\&\, b2 \mid b_1 \,||\, b_2 \\
& \quad \mid b_1 \longrightarrow b_2 \mid b_1 \longleftrightarrow b_2 \mid p(e_1, \ldots, e_n) \\
& \quad \mid \forall v_{(\mathbf{Z}|\mathbf{C}|\mathrm{Array})}.\, b \mid \exists v_{(\mathbf{Z}|\mathbf{C}|\mathrm{Array})}.\, b
\end{aligned}$$

Erfüllung von Zusicherungen

- Wann gilt eine Zusicherung $b \in \mathbf{Assn}$ in einem Zustand σ?
  - Auswertung (denotationale Semantik) ergibt *true*
- **Belegung** der logischen Variablen: $I : \mathbf{Var} \to (\mathbf{Z} \cup \mathbf{C} \cup \text{Array})$
- Semantik von b unter der Belegung I: $\mathcal{B}_v[\![b]\!]^I, \mathcal{A}_v[\![a]\!]^I$

$$\mathcal{A}_v[\![I]\!]^I = \{(\sigma, \sigma(i)) \mid (\sigma, i) \in \mathcal{L}_v[\![I]\!]^I, i \in Dom(\sigma)\}$$

Denotationale Semantik

- Denotation für **Lexp**unter der Belegung I:

$$\begin{aligned}\mathcal{L}_v[\![x]\!]^I &= \{(\sigma, x) \mid \sigma \in \Sigma\}\\ \mathcal{L}_v[\![m[a]]\!]^I &= \{(\sigma, I[i]) \mid (\sigma, I) \in \mathcal{L}_v[\![m]\!]^I, (\sigma, i) \in \mathcal{A}[\![a]\!]^I\}\\ \mathcal{L}_v[\![m.i]\!]^I &= \{(\sigma, m.i) \mid (\sigma, I) \in \mathcal{L}_v[\![m]\!]^I\}\end{aligned}$$

- Denotation für **Aexp**unter der Belegung I:

$$\begin{aligned}\mathcal{A}_v[\![I]\!]^I &= \{(\sigma, \sigma(i) \mid (\sigma, i) \in \mathcal{L}_v[\![I]\!]^I, i \in Dom(\sigma)\} && I \in \mathbf{Lexp}\\ \mathcal{A}_v[\![v]\!]^I &= \{(\sigma, I(v))\} && v \in \mathbf{Var}\end{aligned}$$

Grenzen der Denotationalen Semantik

- Problem mit selbst-definierten Funktionen: $\mathcal{A}_v[\![f(a_1, \ldots, a_n)]\!]^I = ?$
- Problem mit selbst-definierten Prädikaten: $\mathcal{B}_v[\![p(a_1, \ldots, a_n)]\!]^I = ?$
- Betrachte Gleichung:

$$\forall v_{\mathbf{Z}}.f(v) = t$$

so dass all in t vorkommenden Operationen eine denotationale Semantik haben (und f nicht in t vorkommt).

  - Dies gibt der Funktion f eine Semantik, nämlich f.a. Programmzustände σ:

$$\mathcal{A}_v[\![f(a)]\!]^I(\sigma) = \mathcal{A}_v[\![t]\!]^{I'}(\sigma) \text{ mit } I' := I[\mathcal{A}_v[\![a]\!]^I(\sigma)/v]$$

  - Was ist wenn es mehrere solche Gleichungen gibt? Was ist wenn das nicht terminiert?
    $\Longrightarrow$ Nur eindeutig definierte, terminierende Definition (konservative Erweiterungen)
    $\Longrightarrow$ Wird nicht weiter vertieft in dieser Vorlesung; alle im Folgenden verwendete Definition sind derart.
  - Analog für Prädikate

Erfüllung von Zusicherungen

- Wann gilt eine Zusicherung $b \in \mathbf{Assn}$ in einem Zustand σ?
  - Auswertung (denotationale Semantik) ergibt *true*
- **Belegung** der logischen Variablen: $I : \mathbf{Var} \to (\mathbf{Z} \cup \mathbf{C} \cup \text{Array})$
- Semantik von b unter der Belegung I:

$$\begin{aligned}\mathcal{B}_v[\![\forall v_{\mathbf{Z}}.b]\!]^I = \{&(\sigma, 1) \mid \text{für alle } i \in \mathbf{Z} \text{ gilt } (\sigma, 1) \in \mathcal{B}_v[\![b]\!]^{I[i/v]}\}\\ \cup \{&(\sigma, 0) \mid \text{für ein } i \in \mathbf{Z} \text{ gilt } (\sigma, 0) \in \mathcal{B}_v[\![b]\!]^{I[i/v]}\}\\ \mathcal{B}_v[\![\exists v_{\mathbf{Z}}.b]\!]^I = \{&(\sigma, 1) \mid \text{für ein } i \in \mathbf{Z} \text{ gilt } (\sigma, 1) \in \mathcal{B}_v[\![b]\!]^{I[i/v]}\}\\ \cup \{&(\sigma, 0) \mid \text{für alle } i \in \mathbf{Z} \text{ gilt } (\sigma, 0) \in \mathcal{B}_v[\![b]\!]^{I[i/v]}\}\end{aligned}$$

Analog für $\forall v_{\mathbf{C}}.b$, $\exists v_{\mathbf{C}}.b$, $\forall v_{\text{Array}}.b$ und $\exists v_{\text{Array}}.b$

Erfülltheit von Zusicherungen

Erfülltheit von Zusicherungen

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

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

Denotation für Zuweisungen

$$\mathcal{C}_v[\![m = e]\!]^I = \{(\sigma, \sigma[v/I]) \mid (\sigma, I) \in \mathcal{L}_v[\![m]\!]^I, (\sigma, v) \in \mathcal{A}_v[\![e]\!]^I\}$$

Formeln ohne Programmvariablen, ohne Arrays, ohne Strukturen

- Eine Formel $b \in \mathbf{Assn}$ ist pur, wenn sie weder Programmvariablen, noch Strukturen, noch Felder enthält (also keine Teilterme aus **Lexp** und **Idt**.
- Eine Formel ist geschlossen, wenn sie pur ist und keine freien logischen Variablen enthält.
- Sei $\mathbf{Assn}^c \subseteq \mathbf{Assn}$ die Menge der geschlossenen Formeln

Lemma

Für eine geschlossene Formel b ist der Wahrheitswert $\mathcal{B}_v[\![b]\!]^I(\sigma)$ von b unabhängig von I und σ.

- Sei Γ eine endliche Menge von Formeln, dann definieren wir

$$\bigwedge \Gamma := \begin{cases} b_1 \,\&\&\, \ldots \,\&\&\, b_n & \text{für alle } b_i \in \Gamma, \Gamma \neq \emptyset \\ 1 & \text{falls } \Gamma = \emptyset \end{cases}$$

Erfülltheit von Zusicherungen unter Kontext

Erfülltheit von Zusicherungen unter Kontext

Sei $\Gamma \subseteq \mathbf{Assn}^c$ eine endliche Menge und $b \in \mathbf{Assn}$. Im Kontext Γ ist b in Zustand σ mit Belegung I erfüllt ($\Gamma, \sigma \models^I b$), gdw

$$\mathcal{B}_v[\![\Gamma \longrightarrow b]\!]^I(\sigma) = true$$

Floyd-Hoare-Tripel mit Kontext

- Sei $\Gamma \in \mathbf{Assn}^c$ und $P, Q \subseteq \mathbf{Assn}$

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

c ist **partiell korrekt**, wenn für alle Zustände σ und alle Belegungen I die unter Kontext Γ P erfüllen, gilt:
wenn die Ausführung von c mit σ in σ' terminiert, **dann** erfüllen σ' und I im Kontext Γ auch Q.

$$\Gamma \models \{P\}\, c\, \{Q\} \iff \forall I.\, \forall \sigma.\, \Gamma, \sigma \models^I P \wedge \exists \sigma'.\, (\sigma, \sigma') \in \mathcal{C}[\![c]\!] \Longrightarrow \Gamma, \sigma' \models^I Q$$

Floyd-Hoare-Kalkül mit Kontext

$$\frac{}{\Gamma \vdash \{P[e/x]\}\, x = e\, \{P\}}$$

$$\frac{\Gamma \vdash \{A \,\&\&\, b\}\, c_0\, \{B\} \qquad \Gamma \vdash \{A \,\&\&\, \neg b\}\, c_1\, \{B\}}{\Gamma \vdash \{A\}\ \textbf{if}\ (b)\ c_0\ \textbf{else}\ c_1\, \{B\}}$$

$$\frac{\Gamma \vdash \{A \wedge b\}\, c\, \{A\}}{\Gamma \vdash \{A\}\ \textbf{while}(b)\ c\, \{A \wedge \neg b\}}$$

$$\frac{\Gamma \vdash \{A\}\, c_1\, \{B\} \qquad \Gamma \vdash \{B\}\, c_2\, \{C\}}{\Gamma \vdash \{A\}\, c_1; c_2\, \{C\}}$$

Floyd-Hoare-Kalkül mit Kontext

$$\frac{\Gamma \longrightarrow (A' \longrightarrow A) \qquad \Gamma \vdash \{A\}\, c\, \{B\} \qquad \Gamma \longrightarrow (B \longrightarrow B')}{\Gamma \vdash \{A'\}\, c\, \{B'\}}$$

und es muss gezeigt werden für alle Zustände σ und Belegungen I dass $\Gamma \longrightarrow (A' \longrightarrow A)$ wahr bzw. dass

$$\mathcal{B}_v[\![\Gamma \longrightarrow (A' \longrightarrow A)]\!]^I(\sigma) = 1$$

(Analog für $\Gamma \longrightarrow (B \longrightarrow B')$).

Problem

$\mathcal{B}_v[\![.]\!]^I(\sigma)$ *im Allgemeinen nicht berechenbar wegen*

$$\mathcal{B}_v[\![\forall_{\mathbf{Z}} v.b]\!]^I = \{(\sigma,1) \mid \textit{für alle } i \in \mathbf{Z} \textit{ gilt } (\sigma,1) \in \mathcal{B}_v[\![b]\!]^{I[i/v]}\}$$
$$\cup \{(\sigma,0) \mid \textit{für ein } i \in \mathbf{Z} \textit{ gilt } (\sigma,0) \in \mathcal{B}_v[\![b]\!]^{I[i/v]}\}$$

Spezifikation von Funktionen

- Verwendung von geschlossenen Formeln zur Spezifikation von Funktionen und Prädikaten als Kontext
- Fakultät: $n!$ aka factorial(n):

$$\text{factorial}(0) == 1$$
$$\forall n_{\mathbf{Z}}.(n > 0) \longrightarrow \text{factorial}(n+1) == (n+1) \times \text{factorial}(n)$$

- $\sum_{i=0}^{n}$ aka sumup(n):

$$\text{sumup}(0) == 0$$
$$\forall x_{\mathbf{Z}}.(x \geq 0) \longrightarrow \text{sumup}(x+1) == (x+1) + \text{sumup}(x)$$

Spezifikation von Prädikaten

- Gerade und Ungerade:

$$\forall n_{\mathbf{Z}}.\, \text{Gerade}(n) \longleftrightarrow \exists q_{\mathbf{Z}}.n == 2 \times q$$
$$\forall n_{\mathbf{Z}}.\, \text{UnGerade}(n) \longleftrightarrow \exists q_{\mathbf{Z}}.n == 2 \times q + 1$$

- Sortierte Arrays:

$$\forall a_{\text{Array}[\mathbf{Z}]}.\, \text{sorteduntil}(a, 0) \longleftrightarrow 1$$
$$\forall a_{\text{Array}[\mathbf{Z}]}.\forall i_{\mathbf{Z}}.i \geq 0 \longrightarrow (\text{sorteduntil}(a, i+1) \longleftrightarrow (a[i] \leq a[i+1] \wedge \text{sorteduntil}(a, i)))$$

Das Fakultätsbeispiel

$\Gamma = \{\text{Def. von factorial}\}$:

```
/** { 1 == factorial(0) && 0 <= n } */
/** { 1 == factorial(1- 1) && 1 <= 1 && 1-1 <= n } */
p= 1;
/** { p == factorial(1- 1) && 1 <= 1 && 1-1 <= n } */
c= 1;
/** { p == factorial(c- 1) && 1 <= c && c-1 <= n } */
while (c<= n) {
  /** { p == factorial(c-1) && 1<= c && c-1 <= n && c <= n } */
  /** { p*c == factorial(c-1)* c && 1 <= c && c <= n } */
  /** { p*c == factorial(c) && 1 <= c && c <= n } */
  /** { p*c == factorial((c+1)- 1) && 1 <= c+1 && (c+1)-1 <= n } */
  p= p*c;
  /** { p == factorial((c+1)-1) && 1<= c+1 && (c+1)-1 <= n } */
  c= c+1;
  /** { p == factorial(c-1) && 1 <= c && c-1 <= n } */
  }
/** { p == factorial(c-1) && 1<= c && c-1 <= n &&
      factorial(c <= n) } */
/** { p == factorial(c-1) && 1<= c && c-1 <= n && c > n } */
/** { p == factorial(c-1) && 1<= c && c-1 == n } */
/** { p == factorial(n) } */
```


Zurück zum Problem

- Man braucht Rechenverfahren, um $\Gamma \longrightarrow (A' \longrightarrow A)$ etc. zu beweisen
- Man braucht Regeln um Wahre Aussagen herleiten zu können
  - Analog zu operationaler Semantik, aber für logisches Schließen
  - Regelmenge K (Kalkülregeln) die Formeln aus anderen Formeln ableiten.

Natürliches Schließen — Die Regeln

$$\frac{\phi \quad \psi}{\phi \,\&\&\, \psi}\ \&\&I \qquad \frac{\phi \,\&\&\, \psi}{\phi}\ \&\&E_L \qquad \frac{\phi \,\&\&\, \psi}{\psi}\ \&\&E_R$$

$$\frac{\begin{array}{c}[\phi]\\ \vdots\\ \psi\end{array}}{\phi \longrightarrow \psi}\longrightarrow I \qquad \frac{\phi \quad \phi \longrightarrow \psi}{\psi}\longrightarrow E$$

$$\frac{\mathbf{0}}{\phi}\,\mathbf{0} \qquad \frac{\begin{array}{c}[\phi \longrightarrow \mathbf{0}]\\ \vdots\\ \mathbf{0}\end{array}}{\phi}\ \text{raa}$$

Die fehlenden Schlußregeln

$$\frac{\begin{array}{c}[\phi]\\ \vdots\\ \mathbf{0}\end{array}}{\neg\phi}\,\neg I \qquad \frac{\phi \quad \neg\phi}{\mathbf{0}}\,\neg E$$

$$\frac{\phi}{\phi \,||\, \psi}\,||I_L \qquad \frac{\psi}{\phi \,||\, \psi}\,||I_R \qquad \frac{\phi \,||\, \psi \quad \begin{array}{c}[\phi]\\ \vdots\\ \sigma\end{array} \quad \begin{array}{c}[\psi]\\ \vdots\\ \sigma\end{array}}{\sigma}\,||E$$

$$\frac{\phi \longrightarrow \psi \quad \psi \longrightarrow \phi}{\phi \longleftrightarrow \psi}\longleftrightarrow I \qquad \frac{\phi \quad \phi \longleftrightarrow \psi}{\psi}\longleftrightarrow E_L \qquad \frac{\psi \quad \phi \longleftrightarrow \psi}{\phi}\longleftrightarrow E_R$$

Regeln für die Gleichheit

- **Reflexivität**, **Symmetrie**, **Transitivität**:

$$\frac{}{x == x}\ \text{refl} \qquad \frac{x == y}{y == x}\ \text{sym} \qquad \frac{x == y \quad y == z}{x == z}\ \text{trans}$$

- **Kongruenz**:

$$\frac{x_1 == y_1, \ldots, x_n == y_n}{f(x_1, \ldots, x_n) == f(y_1, \ldots, y_n)}\ \text{cong}$$

- **Substitutivität**:

$$\frac{x_1 = y_1, \ldots, x_m = y_m \quad P(x_1, \ldots, x_m)}{P(y_1, \ldots, y_m)}\ \text{subst}$$

Prädikatenlogik mit Induktion

- Sei $\text{fix}(\hat{K})$ die Menge aller mittels Kalkülregeln ableitbaren Formeln.
- K ist korrekt:

$$\text{f.a. } F \in \text{fix}(K) \text{ gilt: f.a. } \sigma, I.\ \mathcal{B}_\nu[\![F]\!]^I(\sigma) = 1.$$

- K ist vollständig:

$$\text{f.a. } F \text{ mit f.a. } \sigma, I.\ \mathcal{B}_\nu[\![F]\!]^I(\sigma) = 1 \text{ gilt: } F \in \text{fix}(K).$$

- Eine Logik L ist entscheidbar, wenn für alle Formeln F entschieden werden kann ob $\mathcal{B}_\nu[\![F]\!]^I(\sigma) = 1$ oder $\mathcal{B}_\nu[\![F]\!]^I(\sigma) = 0$
- Für unsere Belange brauchen wir als Logik Prädikatenlogik mit Gleichheit und Induktion (wegen Rekursion):
  - Es gibt für diese Logik korrekte Kalküle, aber keine vollständigen (und sie ist auch nicht entscheidbar).

Fahrplan

- Einführung
- Operationale Semantik
- Denotationale Semantik
- Äquivalenz der Operationalen und Denotationalen Semantik
- Die Floyd-Hoare-Logik
- Invarianten und die Korrektheit des Floyd-Hoare-Kalküls
- Strukturierte Datentypen
- Modellierung und Spezifikation
- Verifikationsbedingungen
- Vorwärts mit Floyd und Hoare
- Funktionen und Prozeduren
- Referenzen
- Ausblick und Rückblick