

Korrekte Software: Grundlagen und Methoden

Vorlesung 11 vom 06.06.16: Funktionen und Prozeduren

Serge Autexier, Christoph Lüth

Universität Bremen

Sommersemester 2016

Fahrplan

- ▶ Einführung
- ▶ Die Floyd-Hoare-Logik
- ▶ Operationale Semantik
- ▶ Denotationale Semantik
- ▶ Äquivalenz der Semantiken
- ▶ Verifikation: Vorwärts oder Rückwärts?
- ▶ Korrektheit des Hoare-Kalküls
- ▶ Einführung in Isabelle/HOL
- ▶ Weitere Datentypen: Strukturen und Felder
- ▶ Funktionen und Prozeduren
- ▶ Referenzen und Zeiger
- ▶ Frame Conditions & Modification Clauses
- ▶ Ausblick und Rückblick

Funktionen & Prozeduren

- ▶ **Funktionen** sind das zentrale Modularisierungskonzept von C
 - ▶ Kleinste Einheit
 - ▶ NB. Prozeduren sind nur Funktionen vom Typ **void**
 - ▶ Auch in den meisten anderen Sprachen, meist mit Zustandsverkapselung (Methoden)
- ▶ Wir brauchen:
 1. Von Anweisungen zu Funktionen: Deklarationen und Parameter
 2. Semantik von Funktionsdefinition und Funktionsaufruf
 3. Spezifikation von Funktionen
 4. Beweisregeln für Funktionsdefinition und Funktionsaufruf

Motivation

- ▶ Funktionen sind **zentrales Modularisierungskonzept**
- ▶ Wir müssen Funktionen **modular** verifizieren können
- ▶ Semantik von Deklarationen und Parameter — straightforward
- ▶ Semantik von **Rückgabewerten** — Erweiterung der Semantik
- ▶ **Funktionsaufrufe** — Environment, um Funktionsbezeichnern eine Semantik zu geben
 - ▶ C kennt nur call by value
- ▶ Spezifikation von Funktionen: **Vor-/Nachzustand** statt logischer Variablen

Hoare-Kalkül für Funktionspezifikationen

FunDef ::= **Type** *Id*(**Param**^{*}) **FunSpec**⁺ **Blk**

Param ::= **Type** *Id*

FunSpec ::= /** **pre** **Bexpr** **post** **Bexpr** */

Blk ::= {**Decl**^{*} **Stmt**}

Decl ::= **Type** *Id* = **Init** | **Type** *Id*

- ▶ Hoare-Tripel:

$$\{P\} c \{Q_L \mid Q_G\}$$

Beispiel

```
int factorial(int x)
/** pre x >= 0
    post \result = \old(x)! */ {
int r = 0;
if (x = 0) {
    return 1;
}
else {
    r = factorial(x - 1);
}
return r * x;
}
```

$$\tau_0 f(\tau_1 v_1, \dots, \tau_n v_n) \text{ /** pre } P \text{ post } Q \text{ */ } c$$

Beispiel

```
int factorial(int x)
/** pre x >= 0
    post \result = \old(x)! */ {
int r = 0;
if (x = 0) {
    return 1;
}
else {
    r = factorial(x - 1);
}
return r * x;
}
```

$$\frac{\{P'\}c\{Q|Q\}}{\tau_0 f(\tau_1 v_1, \dots, \tau_n v_n) /** pre P post Q */ c}$$

Beispiel

```
int factorial(int x)
/** pre x >= 0
    post \result = \old(x)! */ {
int r = 0;
if (x = 0) {
    return 1;
}
else {
    r = factorial(x - 1);
}
return r * x;
}
```

$$\frac{P \longrightarrow P'_{\backslash \text{old}(\$) \rightarrow \$} \quad \{P'\}c\{Q|Q\}}{\tau_0 f(\tau_1 v_1, \dots, \tau_n v_n) \text{ /** pre } P \text{ post } Q \text{ */ } c}$$

Beispiel

```
int factorial(int x)
/** pre x >= 0
    post \result = \old(x)! */ {
int r = 0;
if (x = 0) {
    return 1;
}
else {
    r = factorial(x - 1);
}
return r * x;
}
```

$$\frac{x \geq 0 \longrightarrow P'_{\text{old}(\$) \rightarrow \$} \{P'\}c\{\backslash\text{result} = \backslash\text{old}(x)! \mid \backslash\text{result} = \backslash\text{old}(x)!\}}{\text{int factorial(int x) /** pre x >= 0 post \result = \old(x)! */ c}}$$

Hoare-Kalkül mit return

$$\overline{\{P\}\{\}\{P|Q\}}$$

$$\frac{\{P\}c\{Q'_1|Q_2\} \quad \{Q'_1\}cs\{Q_1|Q_2\}}{\{P\}c \ cs\{Q_1|Q_2\}}$$

$$\overline{\{Q_1[e/l]\}l = e\{Q_1|Q_2\}}$$

$$\frac{\{P \wedge b\}c_1\{Q_1|Q_2\} \quad \{P \wedge \neg b\}c_2\{Q_1|Q_2\}}{\{P\}\text{if } b \text{ } c_1 \text{ else } c_2\{Q_1|Q_2\}}$$

While, Weakening

$$\frac{\{P \wedge b\}c\{P|Q\}}{\{P\}\text{while } (b) \ c\{P \wedge \neg(b)|Q\}}$$

$$\frac{P \longrightarrow P' \quad \{P'\}c\{Q'_1|Q'_2\} \quad Q'_1 \longrightarrow Q_1 \quad Q'_2 \longrightarrow Q_2}{\{P\}c\{Q_1|Q_2\}}$$

Hoare-Kalkül mit return

$$\overline{\{Q[e/\backslash result]\} \text{return } e \{P|Q\}}$$

$$\frac{Q \text{ enthält kein } \backslash result}{\{Q\} \text{return} \{P|Q\}}$$

Funktionsaufruf

$$\frac{(V_1 = e_1 \wedge \dots \wedge V_n = e_n \wedge Q_1) \longrightarrow f.pre(e_1, \dots, e_n) \quad P := f.post(e_1, \dots, e_n) \setminus_{result \rightarrow l, \setminus_{old(v_i) \rightarrow V_i}}}{\{(V_1 = e_1 \wedge \dots \wedge V_n = e_n \wedge Q_1 \wedge P)[f(e_1, \dots, e_n)/l]\} \\ l = f(e_1, \dots, e_n) \\ \{V_1 = e_1 \wedge \dots \wedge V_n = e_n \wedge Q_1 \wedge P | Q_2\}}$$

f mit formalen Parametern v_1, \dots, v_n ; V_1, \dots, V_n logische Variablen

```

int factorial(int x)
/** pre x >= 0
    post \result = \old(x)! */ {
/** { COND: x >= 0  $\longrightarrow$  x >= 0  $\wedge$  x! = x!
    x >= 0  $\wedge$  x! = \old(x)! } */
    int r = 0;
/** {x >= 0  $\wedge$  x! = \old(x)!} */
    if (x == 0) {
/** {x >= 0  $\wedge$  x == 0  $\wedge$  x! = \old(x)!} */
/** {x >= 0  $\wedge$  x == 0  $\wedge$  1 = \old(x)!} */
        return 1;
/** { r * x = \old(i)! | \result = \old(x)!} */
    }
    else { ... }
/** { r * x = \old(i)! | \result = \old(i)! } */
    return r * x;
/** {\result = \old(i)! | \result = \old(i)!} */

```

```

else {
/** {x >= 0 ∧ ¬(x = 0) ∧ x! = \old(x)!} */
/** {COND: x >= 0 ∧ ¬(x = 0) ∧
      factorial(x-1) * x = \old(x)! ∧ V = x → x - 1 >= 0
      x >= 0 ∧ ¬(x = 0) ∧ factorial(x-1) * x = \old(x)! ∧
      V = x ∧ factorial(x-1) = (V-1)!} */
    r = factorial(x - 1);
/** {x >= 0 ∧ ¬(x = 0) ∧ r * x = \old(x)! ∧
      V = x ∧ r = (V-1)!} */
/** { x >= 0 ∧ ¬(x = 0) ∧ r * x = \old(x)! } */
/** { r * x = \old(i)! } */
}
/** { r * x = \old(i)! | \result =\old(i)! } */
return r * x;
/** {\result = \old(i)! | \result =\old(i)!} */
}

```

Approximative schwächste Vorbedingung (Revisited)

$$\text{awp}(\Gamma, \{ \}, P) \stackrel{\text{def}}{=} P$$

$$\text{awp}(\Gamma, l = f(e_1, \dots, e_n), P) \stackrel{\text{def}}{=} P[F(\llbracket e_1 \rrbracket, \dots, \llbracket e_n \rrbracket) / \llbracket l \rrbracket]$$

mit $\text{post}(\Gamma!f) = (\forall v_1, \dots, v_n. \text{result} = F(v_1, \dots, v_n))$

$$\text{awp}(\Gamma, l = e, P) \stackrel{\text{def}}{=} P[\llbracket e \rrbracket / \llbracket l \rrbracket]$$

$$\text{awp}(\Gamma, \{c \ c_s\}, P) \stackrel{\text{def}}{=} \text{awp}(\Gamma, c, \text{awp}(\{c_s\}, P))$$

$$\text{awp}(\Gamma, \text{if } (b) \{c_0\} \text{ else } \{c_1\}, P) \stackrel{\text{def}}{=} (b \wedge \text{awp}(\Gamma, c_0, P)) \vee (\neg b \wedge \text{awp}(\Gamma, c_1, P))$$

$$\text{awp}(\Gamma, /** \{q\} */, P) \stackrel{\text{def}}{=} \llbracket q \rrbracket$$

$$\text{awp}(\Gamma, \text{while } (b) /** \text{inv } i */ c, P) \stackrel{\text{def}}{=} \llbracket i \rrbracket$$

$$\text{awp}(\Gamma, \text{return } e, P) \stackrel{\text{def}}{=} \text{post}(\Gamma)[\llbracket e \rrbracket / \text{result}]$$

$$\text{awp}(\Gamma, \text{return}, P) \stackrel{\text{def}}{=} \text{post}(\Gamma)$$

Approximative schwächste Vorbedingung (Revisited)

$$\begin{aligned} \text{wvc}(\Gamma, \{ \}, P) &\stackrel{\text{def}}{=} \emptyset \\ \text{wvc}(\Gamma, x = e, P) &\stackrel{\text{def}}{=} \emptyset \\ \text{wvc}(\Gamma, x = f(e_1, \dots, e_n), P) &\stackrel{\text{def}}{=} P \longrightarrow \text{pre}(\Gamma!f)(\llbracket e_1 \rrbracket, \dots, \llbracket e_n \rrbracket) \\ \text{wvc}(\Gamma, \{c \ c_s\}, P) &\stackrel{\text{def}}{=} \text{wvc}(\Gamma, c, \text{awp}(\{c_s\}, P)) \\ &\quad \cup \text{wvc}(\Gamma, \{c_s\}, P) \\ \text{wvc}(\Gamma, \mathbf{if} (b) \ c_0 \ \mathbf{else} \ c_1, P) &\stackrel{\text{def}}{=} \text{wvc}(\Gamma, c_0, P) \cup \text{wvc}(\Gamma, c_1, P) \\ \text{wvc}(\Gamma, /** \{q\} */ , P) &\stackrel{\text{def}}{=} \{ \llbracket q \rrbracket \longrightarrow P \} \\ \text{wvc}(\Gamma, \mathbf{while} (b) /** \mathbf{inv} \ i */ \ c, P) &\stackrel{\text{def}}{=} \text{wvc}(\Gamma, c, \llbracket i \rrbracket) \\ &\quad \cup \{ \llbracket i \rrbracket \wedge b \longrightarrow \text{awp}(\Gamma, c, \llbracket i \rrbracket) \} \\ &\quad \cup \{ \llbracket i \rrbracket \wedge \neg b \longrightarrow P \} \\ \text{wvc}(\Gamma, \mathbf{return} \ e, P) &\stackrel{\text{def}}{=} \emptyset \end{aligned}$$