Totale Korrektheit: HK_{TK} (1)

Zur Erinnerung sei hier der Hoare-Kalkül HK_{TK} für totale Korrektheit noch einmal wiederholt...

$$[\text{skip}]\quad \frac{—}{\{p\}\ skip\ \{p\}}$$

$$[\text{abort}]\quad \frac{—}{\{p\}\ abort\ \{q\}}$$

$$[\text{ass}]\quad \frac{—}{\{p[t\backslash x]\}\ x{:=}t\ \{p\}}$$

$$[\text{comp}]\quad \frac{\{p\}\ \pi_1\ \{r\},\ \{r\}\ \pi_2\ \{q\}}{\{p\}\ \pi_1;\pi_2\ \{q\}}$$

$$[\text{ite}]\quad \frac{\{p \wedge b\}\ \pi_1\ \{q\},\ \{p \wedge \neg b\}\ \pi_2\ \{q\}}{\{p\}\ \text{if } b \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi } \{q\}}$$

$$[\text{cons}]\quad \frac{p \Rightarrow p_1,\ \{p_1\}\ \pi\ \{q_1\},\ q_1 \Rightarrow q}{\{p\}\ \pi\ \{q\}}$$

Totale Korrektheit: HK_{TK} (2)

$$[\text{while}_{TK}]\quad \frac{I \wedge b \Rightarrow u[t/v],\ \{I \wedge b \wedge t{=}w\}\ \pi\ \{I \wedge t{<}w\}}{\{I\}\ \text{while } b \text{ do } \pi \text{ od } \{I \wedge \neg b\}}$$

wobei

- u Boolescher Ausdruck über der Variablen v,
- t arithmetischer Term,
- w Variable, die in I, b, π und t nicht frei vorkommt,
- $M =_{df} \{\sigma(v) \mid \sigma \in \Sigma\ \wedge\ [\![\, u\,]\!]_B(\sigma) = \mathtt{tt}\}$ noethersch geordnete Menge (sog. noethersche Halbordnung).
  $\leadsto$ *Terminationsordnung*!

Bemerkung: In den obigen Regeln verwenden wir geschweifte statt eckiger Klammern für zugesicherte Eigenschaften, um einen Bezeichnungskonflikt mit der ebenfalls durch eckige Klammern bezeichneten *syntaktischen Substitution* zu vermeiden.

Nachträge

- Substitutionen
  - *Syntaktische* Substitution
  - *Semantische* Substitution
  - *Substitutionslemma*
- Noethersche Ordnungen

Syntaktische Substitution

Definition

Die *syntaktische Substitution* für arithmetische Terme ist eine dreistellige Abbildung

$$\cdot[\cdot/\cdot] : \textbf{Aexpr} \times \textbf{Aexpr} \times \textbf{Var} \to \textbf{Aexpr}$$

die induktiv definiert ist durch

$$\begin{array}{rcll} n[t/x] & =_{df} & n & \text{für } n \in \textbf{Num} \\ y[t/x] & =_{df} & \begin{cases} t & \text{falls } y = x \\ y & \text{sonst} \end{cases} & \\ (t_1\ op\ t_2)[t/x] & =_{df} & (t_1[t/x]\ op\ t_2[t/x]) & \text{für } op \in \{+, *, -, ...\} \end{array}$$

Semantische Substitution

Definition

Die *semantische Substitution* ist eine dreistellige Abbildung

$$\cdot[\cdot/\cdot] : \Sigma \times \mathbb{Z} \times \textbf{Var} \to \Sigma$$

die definiert ist durch

$$\sigma[z/x](y) =_{df} \begin{cases} z & \text{falls } y = x \\ \sigma(y) & \text{sonst} \end{cases}$$

Substitutionslemma

Wichtig:

Substitutionslemma

$$[\![\, e[t/x] \,]\!]_A(\sigma) = [\![\, e \,]\!]_A(\sigma[[\![\, t \,]\!]_A(\sigma)/x])$$

wobei

- $[t/x]$ die *syntaktische Substitution* und
- $[[\![\, t \,]\!]_A(\sigma)/x]$ die *semantische Substitution*

bezeichnen.

Wohlfundierte oder Noethersche Ordnungen (1)

Definition

Sei P eine Menge und sei $<$ eine irreflexive und transitive Relation auf P.

Dann ist das Paar $(P, <)$ eine *irreflexive partielle Ordnung*.

Beispiele:

- $(\mathbb{Z}, <)$, $(\mathbb{Z}, >)$, $(\mathbb{N}, <)$, $(\mathbb{N}, >)$

Wohlfundierte oder Noethersche Ordnungen (2)

Definition

Sei $(P, <)$ eine irreflexive partielle Ordnung und sei W eine Teilmenge von P.

Dann heißt die Relation $<$ auf W *wohlfundiert*, wenn es keine unendlich absteigende Kette

$$\ldots < w_2 < w_1 < w_0$$

von Elementen $w_i \in W$ gibt.

Das Paar $(W, <)$ heißt dann eine *wohlfundierte Struktur* oder auch eine *wohlfundierte* oder *Noethersche Ordnung*.

Sprechweise: Gilt $w < w'$ für $w, w' \in W$, sagen wir, w *ist kleiner als* w' oder w' *ist größer als* w.

Beispiele:

- $(\mathbb{N}, <)$, aber nicht $(\mathbb{Z}, <)$, $(\mathbb{Z}, >)$ oder $(\mathbb{N}, >)$

Wohlfundierte oder Noethersche Ordnungen (3)

Konstruktionsprinzipien für wohlfundierte Ordnungen aus gegebenen wohlfundierten Ordnungen...

Lemma

Seien $(W_1, <_1)$ und $(W_2, <_2)$ zwei wohlfundierte Ordnungen.

Dann sind auch

- $(W_1 \times W_2, <_{com})$ mit *komponentenweiser* Ordnung definiert durch

$$(m_1, m_2) <_{com} (n_1, n_2) \;\text{ gdw. }\; m_1 <_1 n_1 \wedge m_2 <_2 n_2$$

- $(W_1 \times W_2, <_{lex})$ mit *lexikographischer* Ordnung def. durch

$$(m_1, m_2) <_{lex} (n_1, n_2) \;\text{ gdw.}$$

$$(m_1 <_1 n_1) \;\vee\; (m_1 = n_1 \wedge m_2 <_2 n_2)$$

wohlfundierte Ordnungen.

Anmerkungen zu...

...den der

- Konsequenzregel [cons] und der
- Schleifenregeln [while$_{PK}$] und [while$_{TK}$]

von HK_{PK} bzw. HK_{TK} zugrundeliegenden Intuitionen.

Zur Konsequenzregel (1)

[cons] $\dfrac{p \Rightarrow p_1,\ \{p_1\}\ \pi\ \{q_1\},\ q_1 \Rightarrow q}{\{p\}\ \pi\ \{q\}}$

Intuitiv:

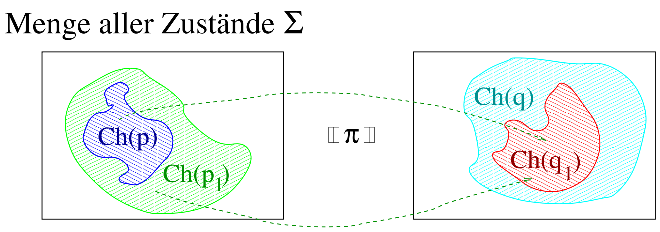
Die Konsequenzregel

- ...stellt die Schnittstelle zwischen Programmverifikation und den logischen Formeln der Zusicherungssprache dar
- ...erlaubt es,
  - Vorbedingungen zu *verstärken* (Übergang von p_1 zu p möglich, falls $p \Rightarrow p_1$ ($\Leftrightarrow$ $Ch(p) \subseteq Ch(p_1)$))
  - Nachbedingungen *abzuschwächen* (Übergang von q_1 zu q möglich, falls $q_1 \Rightarrow q$ ($\Leftrightarrow$ $Ch(q_1) \subseteq Ch(q)$))

  ...um so die Anwendung anderer Beweisregeln zu ermöglichen.

Zur Konsequenzregel (2)

Veranschaulichung von Verstärkung und Abschwächung:



$p \Longrightarrow p_1 \qquad \{p_1\}\ \pi\ \{q_1\} \qquad q_1 \Longrightarrow q$

z.B.: $x>5 \Longrightarrow x>0 \qquad \{x>0\}\ \pi\ \{y>5\} \qquad y>5 \Longrightarrow y>0$

Zur while-Regel in HK_{PK}

[while] $\dfrac{\{I \wedge b\}\ \pi\ \{I\}}{\{I\}\ \text{while } b \text{ do } \pi \text{ od } \{I \wedge \neg b\}}$

Intuitiv:

- Das durch I beschriebene Prädikat gilt
  - ...*vor* und *nach* jeder Ausführung des Rumpfes der while-Schleife
  - ...und wird deswegen als *Invariante* der while-Schleife bezeichnet.
- Die while-Regel besagt weiter, dass
  - wenn zusätzlich (zur Invarianten) auch b vor jeder Ausführung des Schleifenrumpfs gilt, dass nach Beendigung der while-Schleife $\neg b$ wahr ist.

Zur while-Regel in HK_{TK} (1)

Erinnerung:

[while$_{TK}$] $\dfrac{I \wedge b \Rightarrow u[t/v],\ \{I \wedge b \wedge t{=}w\}\ \pi\ \{I \wedge t{<}w\}}{\{I\}\ \text{while } b \text{ do } \pi \text{ od } \{I \wedge \neg b\}}$

wobei

- u Boolescher Ausdruck über der Variablen v,
- t arithmetischer Term,
- w Variable, die in I, b, π und t nicht frei vorkommt,
- $M =_{df} \{\sigma(v) \mid \sigma \in \Sigma \ \wedge\ [\![\, u \,]\!]_B(\sigma) = \mathtt{tt}\}$ noethersch geordnete Menge (sog. noethersche Halbordnung).
  $\rightsquigarrow$ *Terminationsordnung*!

Zur while-Regel in HK_{TK} (2)

- Prämisse 1: $I \wedge b \Rightarrow u[t/v]$
  Wann immer der Schleifenrumpf noch einmal ausgeführt wird (d.h. $I \wedge b$ ist wahr), gilt, dass $u[t/v]$ wahr ist, woraus aufgrund der Definition von M folgt, dass der Wert von t Element einer noethersch geordneten Menge ist.
- Prämisse 2: $\{I \wedge b \wedge t = w\}\ \pi\ \{I \wedge t < w\}$
  - w speichert den initialen Wert von t (w ist sog. *logische Variable*), d.h. den Wert, den t vor Eintritt in die Schleife hat (gilt, da w als logische Variable insbesondere nicht in π vorkommt)
  - Zusammen damit, dass der Wert von w (als logische Variable) invariant unter der Ausführung des Schleifenrumpfs ist, garantiert $t < w$ in der Nachbedingung von Prämisse 2, dass der Wert von t nach jeder Ausführung des Schleifenrumpfs bzgl. der noetherschen Ordnung abgenommen hat.
- Zusammen implizieren die obigen beiden Punkte die Terminierung der while-Schleife, da es in einer noethersch geordneten Menge keine unendlich absteigenden Ketten gibt. Folglich kann die Bedingung $I \wedge b$ in Prämisse 1 nicht unendlich oft wahr sein, da dies zusammen mit Prämisse 2 ein unendliches Absteigen erforderte.)

Programm- vs. logische Variablen

Wir unterscheiden in Zusicherungen $\{p\}\ \pi\ \{q\}$ zwischen...

- *Programmvariablen*
  ...Variablen, die in π vorkommen
- *logischen Variablen*
  ...Variablen, die in π nicht vorkommen

Logische Variablen erlauben...

- sich *initiale* Werte von Programmvariablen zu "merken", um in Nachbedingungen geeignet darauf Bezug zu nehmen.

Beispiel:

- $\{x = n\}\ y := 1;$ while $x \neq 1$ do $y := y * x; x := x - 1$ od $\{y = n! \wedge n > 0\}$
  ...die Nachbedingung macht eine Aussage über den Zusammenhang des Anfangswertes von x (gespeichert in n) und des schließlichen Wertes von y.
- $\{x = n\}\ y := 1;$ while $x \neq 1$ do $y := y * x; x := x - 1$ od $\{y = x! \wedge x > 0\}$
  ...die Nachbedingung macht eine Aussage über den Zusammenhang der schließlichen Werte von x und y.

HK_{TK} versus HK_{PK}

Beachte:

HK_{TK} und HK_{PK} sind bis auf die Schleifenregel identisch...

- *Totale Korrektheit*: [while$_{TK}$]

  [while$_{TK}$] $\dfrac{I \wedge b \Rightarrow u[t/v],\ \{I \wedge b \wedge t{=}w\}\ \pi\ \{I \wedge t{<}w\}}{\{I\}\ \text{while } b \text{ do } \pi \text{ od } \{I \wedge \neg b\}}$

- *Partielle Korrektheit*: [while$_{PK}$]

  [while$_{PK}$] $\dfrac{\{I \wedge b\}\ \pi\ \{I\}}{\{I\}\ \text{while } b \text{ do } \pi \text{ od } \{I \wedge \neg b\}}$

Bew. totale Korrektheit: Fakultät (1)

Beweise, dass das Hoare-Tripel

$$[a > 0]$$
$$x := a;\ y := 1;\ \text{while } x > 1 \text{ do } y := y * x; x := x - 1 \text{ od}$$
$$[y = a!]$$

gültig ist im Sinne totaler Korrektheit.

Wir entwickeln den Beweis in der Folge Schritt für Schritt!

Wahl von Invariante und Terminierungsterm

Schritt 1

"Träumen"...

- der Invariante: $y * x! = a! \wedge x > 0$
- des Terminierungsterms: $t \equiv x$
- von u: $u \equiv v \geq 0$

...um die [while]-Regel anwenden zu können.

Beachte:

- Aus der Wahl von $u \equiv v \geq 0$ und von $b \equiv x > 1$ folgt:
  - $M = \{0, 1, 2, 3, 4, \ldots\}$
  - $(v \geq 0)[x/v] \equiv x \geq 0$

  ...und somit insgesamt: $I \wedge b \Rightarrow x \in M$ mit $(M, <)$ Noethersch geordnet.

Hinweis zur Notation: $\equiv$ steht für *syntaktisch gleich*

Wahl von Invariante und Terminierungsterm

Mit der vorherigen Wahlt von I, t und u gilt:

$$\begin{array}{rcl}
M & =_{df} & \{\sigma(v) \mid \sigma \in \Sigma\ \wedge\ [\![\, u \,]\!]_B(\sigma) = \mathtt{tt}\} \\
 & = & \{\sigma(v) \mid \sigma \in \Sigma\ \wedge\ [\![\, v \geq 0 \,]\!]_B(\sigma) = \mathtt{tt}\} \\
 & = & \{\sigma(v) \mid \sigma \in \Sigma\ \wedge\ \mathit{groessergleich}([\![\, v \,]\!]_A(\sigma), [\![\, 0 \,]\!]_A(\sigma))\} \\
 & = & \{\sigma(v) \mid \sigma \in \Sigma\ \wedge\ \mathit{groessergleich}(\sigma(v), \mathbf{0}) = \mathtt{tt}\} \\
 & = & \{\sigma(v) \mid \sigma \in \Sigma\ \wedge\ \sigma(v) \geq \mathbf{0}\} \\
 & = & \mathbb{N} \cup \{\mathbf{0}\}
\end{array}$$

Damit haben wir insbesondere:

- $(M, <) = (\mathbb{N} \cup \{\mathbf{0}\}, <)$ ist noethersch geordnet.
- $u[t/x] = (v \geq 0)[x/v] = x \geq 0$

Bew. totaler Korrektheit: Fakultät (4)

Schritt 2

Behandlung des Rumpfs der while-Schleife...

Der Nachweis der Gültigkeit von

$$y * x! = a! \wedge x > 0 \wedge x > 1 \Rightarrow x \geq 0$$
$$[y * x! = a! \wedge x > 0 \wedge x > 1 \wedge x = w]$$
$$y := y * x;$$
$$x := x - 1;$$
$$[y * x! = a! \wedge x > 0 \wedge x < w]$$

erlaubte mithilfe der [while]-Regel den Übergang zu:

$$[y * x! = a! \wedge x > 0]$$
$$\text{while } x > 1 \text{ do}$$
$$y * x! = a! \wedge x > 0 \wedge x > 1 \Rightarrow x \geq 0$$
$$[y * x! = a! \wedge x > 0 \wedge x > 1 \wedge x = w]$$
$$y := y * x;$$
$$x := x - 1;$$
$$[y * x! = a! \wedge x > 0 \wedge x < w]$$
$$\text{od [while]}$$
$$[y * x! = a! \wedge x > 0 \wedge \neg(x > 1)]$$

Bew. totaler Korrektheit: Fakultät (5)

Behandlung des Rumpfs der while-Schleife im Detail:

$$y * x! = a! \wedge x > 0 \wedge x > 1 \Rightarrow x \geq 0$$
$$[y * x! = a! \wedge x > 0 \wedge x > 1 \wedge x = w]$$

$$y := y * x;$$
$$x := x - 1;$$
$$[y * x! = a! \wedge x > 0 \wedge x < w]$$

Bew. totaler Korrektheit: Fakultät (6)

Wegen *Rückwärts*zuweisungsregel wird der Rumpf der while-Schleife von hinten nach vorne bearbeitet:

$$y * x! = a! \wedge x > 0 \wedge x > 1 \Rightarrow x \geq 0$$
$$[y * x! = a! \wedge x > 0 \wedge x > 1]$$

$$y := y * x;$$
$$[y * (x-1)! = a! \wedge x - 1 > 0 \wedge x - 1 < w]$$
$$x := x - 1; \text{ [ass]}$$
$$[y * x! = a! \wedge x > 0 \wedge x < w]$$

Bew. totaler Korrektheit: Fakultät (7)

Nach abermaliger Anwendung der [ass]-Regel erhalten wir...

$$y * x! = a! \wedge x > 0 \wedge x > 1 \Rightarrow x \geq 0$$
$$[y * x! = a! \wedge x > 0 \wedge x > 1 \wedge x = w]$$

$$[(y * x) * (x-1)! = a! \wedge x - 1 > 0 \wedge x - 1 < w]$$
$$y := y * x; \text{ [ass]}$$
$$[y * (x-1)! = a! \wedge x - 1 > 0 \wedge x - 1 < w]$$
$$x := x - 1; \text{ [ass]}$$
$$[y * x! = a! \wedge x > 0]$$

...wobei noch eine “Beweislücke” verbleibt!

Bew. totaler Korrektheit: Fakultät (8)

Schluss der "Beweislücke" in der zugrundeliegenden Theorie:

$$y * x! = a! \wedge x > 0 \wedge x > 1 \Rightarrow x \geq 0$$
$$[y * x! = a! \wedge x > 0 \wedge x > 1]$$
$$\Downarrow \text{[cons]}$$
$$[(y * x) * (x-1)! = a! \wedge x - 1 > 0]$$
$$y := y * x; \text{ [ass]}$$
$$[y * (x-1)! = a! \wedge x - 1 > 0]$$
$$x := x - 1; \text{ [ass]}$$
$$[y * x! = a! \wedge x > 0]$$

Bew. totaler Korrektheit: Fakultät (9)

Anwendung der [while]-Regel liefert nun wie gewünscht:

$$[y * x! = a! \wedge x > 0]$$
$$\text{while } x > 1 \text{ do}$$
$$y * x! = a! \wedge x > 0 \wedge x > 1 \Rightarrow x \geq 0$$
$$[y * x! = a! \wedge x > 0 \wedge x > 1]$$
$$\Downarrow \text{[cons]}$$
$$[(y * x) * (x-1)! = a! \wedge x - 1 > 0]$$
$$y := y * x; \text{ [ass]}$$
$$[y * (x-1)! = a! \wedge x - 1 > 0]$$
$$x := x - 1; \text{ [ass]}$$
$$[y * x! = a! \wedge x > 0]$$
$$\text{od [while]}$$
$$[y * x! = a! \wedge x > 0 \wedge \neg(x > 1)]$$

Bew. totaler Korrektheit: Fakultät (10)

Schritt 3

Zur gewünschten Nachbedingung verbleibt offenbar ebenfalls eine Beweislücke:

$$[y * x! = a! \wedge x > 0]$$
$$\text{while } x > 1 \text{ do}$$
$$y * x! = a! \wedge x > 0 \wedge x > 1 \Rightarrow x \geq 0$$
$$[y * x! = a! \wedge x > 0 \wedge x > 1]$$
$$\Downarrow \text{[cons]}$$
$$[(y * x) * (x-1)! = a! \wedge x - 1 > 0]$$
$$y := y * x; \text{ [ass]}$$
$$[y * (x-1)! = a! \wedge x - 1 > 0]$$
$$x := x - 1; \text{ [ass]}$$
$$[y * x! = a! \wedge x > 0]$$
$$\text{od [while]}$$
$$[y * x! = a! \wedge x > 0 \wedge \neg(x > 1)]$$

$$\{y = a!\}$$

Bew. totaler Korrektheit: Fakultät (11)

Schluss der Beweislücke in der zugrundeliegenden Theorie:

$$[y * x! = a! \wedge x > 0]$$
$$\text{while } x > 1 \text{ do}$$
$$y * x! = a! \wedge x > 0 \wedge x > 1 \Rightarrow x \geq 0$$
$$[y * x! = a! \wedge x > 0 \wedge x > 1]$$
$$\Downarrow \text{[cons]}$$
$$[(y * x) * (x-1)! = a! \wedge x - 1 > 0]$$
$$y := y * x; \text{ [ass]}$$
$$[y * (x-1)! = a! \wedge x - 1 > 0]$$
$$x := x - 1; \text{ [ass]}$$
$$[y * x! = a! \wedge x > 0]$$
$$\text{od [while]}$$
$$[y * x! = a! \wedge x > 0 \wedge \neg(x > 1)]$$
$$\Downarrow \text{[cons]}$$
$$[y * x! = a! \wedge x > 0 \wedge x \leq 1]$$
$$\Downarrow \text{[cons]}$$
$$[y * x! = a! \wedge x = 1]$$
$$\Downarrow \text{[cons]}$$
$$[y = a!]$$

Bew. totaler Korrektheit: Fakultät (12)

Aus Platzgründen etwas verkürzt dargestellt:

$$\begin{array}{c}
[y * x! = a! \land x > 0] \\
\text{while } x > 1 \text{ do} \\
y * x! = a! \land x > 0 \land x > 1 \Rightarrow x \geq 0 \\
[y * x! = a! \land x > 0 \land x > 1] \\
\Downarrow \text{[cons]} \\
[(y * x) * (x-1)! = a! \land x - 1 > 0] \\
y := y * x; \text{ [ass]} \\
[y * (x-1)! = a! \land x - 1 > 0] \\
x := x - 1; \text{ [ass]} \\
[y * x! = a! \land x > 0] \\
\text{od [while]} \\
[y * x! = a! \land x > 0 \land \neg(x > 1)] \\
\Downarrow \text{[cons]} \\
[y = a!]
\end{array}$$

Bew. totaler Korrektheit: Fakultät (13)

Schritt 4

Es verbleibt, die Beweislücke zur gewünschten Vorbedingung zu schließen:

$$\begin{array}{c}
[a > 0] \\
\\
x := a; \\
y := 1; \\
[y * x! = a! \land x > 0] \\
\text{while } x > 1 \text{ do} \\
y * x! = a! \land x > 0 \land x > 1 \Rightarrow x \geq 0 \\
[y * x! = a! \land x > 0 \land x > 1] \\
\Downarrow \text{[cons]} \\
[(y * x) * (x-1)! = a! \land x - 1 > 0] \\
y := y * x; \text{ [ass]} \\
[y * (x-1)! = a! \land x - 1 > 0] \\
x := x - 1; \text{ [ass]} \\
[y * x! = a! \land x > 0] \\
\text{od [while]} \\
[y * x! = a! \land x > 0 \land \neg(x > 1)] \\
\Downarrow \text{[cons]} \\
[y = a!]
\end{array}$$

Bew. totaler Korrektheit: Fakultät (14)

Einmalige Anwendung der [ass]-Regel liefert:

$$\begin{array}{c}
[a > 0] \\
\\
x := a; \\
[1 * x! = a! \land x > 0] \\
y := 1; \text{ [ass]} \\
[y * x! = a! \land x > 0] \\
\text{while } x > 1 \text{ do} \\
y * x! = a! \land x > 0 \land x > 1 \Rightarrow x \geq 0 \\
[y * x! = a! \land x > 0 \land x > 1] \\
\Downarrow \text{[cons]} \\
[(y * x) * (x-1)! = a! \land x - 1 > 0] \\
y := y * x; \text{ [ass]} \\
[y * (x-1)! = a! \land x - 1 > 0] \\
x := x - 1; \text{ [ass]} \\
[y * x! = a! \land x > 0] \\
\text{od [while]} \\
[y * x! = a! \land x > 0 \land \neg(x > 1)] \\
\Downarrow \text{[cons]} \\
[y = a!]
\end{array}$$

Bew. totaler Korrektheit: Fakultät (15)

Abermalige Anwendung der [ass]-Regel liefert:

$$\begin{array}{c}
[a > 0] \\
\\
[1 * a! = a! \land a > 0] \\
x := a; \text{ [ass]} \\
[1 * x! = a! \land x > 0] \\
y := 1; \text{ [ass]} \\
[y * x! = a! \land x > 0] \\
\text{while } x > 1 \text{ do} \\
y * x! = a! \land x > 0 \land x > 1 \Rightarrow x \geq 0 \\
[y * x! = a! \land x > 0 \land x > 1] \\
\Downarrow \text{[cons]} \\
[(y * x) * (x-1)! = a! \land x - 1 > 0] \\
y := y * x; \text{ [ass]} \\
[y * (x-1)! = a! \land x - 1 > 0] \\
x := x - 1; \text{ [ass]} \\
[y * x! = a! \land x > 0] \\
\text{od [while]} \\
[y * x! = a! \land x > 0 \land \neg(x > 1)] \\
\Downarrow \text{[cons]} \\
[y = a!]
\end{array}$$

Bew. totaler Korrektheit: Fakultät (16)

Schluss der letzten Beweislücke in der zugrundeliegenden Theorie:

$$
\begin{array}{c}
[a > 0] \\
\Downarrow \text{[cons]} \\
[1 * a! = a! \wedge a > 0] \\
x := a; \text{ [ass]} \\
[1 * x! = a! \wedge x > 0] \\
y := 1; \text{ [ass]} \\
[y * x! = a! \wedge x > 0] \\
\text{while } x > 1 \text{ do} \\
y * x! = a! \wedge x > 0 \wedge x > 1 \Rightarrow x \geq 0 \\
[y * x! = a! \wedge x > 0 \wedge x > 1] \\
\Downarrow \text{[cons]} \\
[(y * x) * (x-1)! = a! \wedge x - 1 > 0] \\
y := y * x; \text{ [ass]} \\
[y * (x-1)! = a! \wedge x - 1 > 0] \\
x := x - 1; \text{ [ass]} \\
[y * x! = a! \wedge x > 0] \\
\text{od [while]} \\
[y * x! = a! \wedge x > 0 \wedge \neg(x > 1)] \\
\Downarrow \text{[cons]} \\
[y = a!]
\end{array}
$$

Überblick (17)

$$
\begin{array}{c}
[a > 0] \\
\Downarrow \text{[cons]} \\
[1 * a! = a! \wedge a > 0] \\
x := a; \text{ [ass]} \\
[1 * x! = a! \wedge x > 0] \\
y := 1; \text{ [ass]} \\
[y * x! = a! \wedge x > 0] \\
\text{while } x > 1 \text{ do} \\
y * x! = a! \wedge x > 0 \wedge x > 1 \Rightarrow x \geq 0 \\
[y * x! = a! \wedge x > 0 \wedge x > 1] \\
\Downarrow \text{[cons]} \\
[(y * x) * (x-1)! = a! \wedge x - 1 > 0] \\
y := y * x; \text{ [ass]} \\
[y * (x-1)! = a! \wedge x - 1 > 0] \\
x := x - 1; \text{ [ass]} \\
[y * x! = a! \wedge x > 0] \\
\text{od [while]} \\
[y * x! = a! \wedge x > 0 \wedge \neg(x > 1)] \\
\Downarrow \text{[cons]} \\
[y * x! = a! \wedge x > 0 \wedge x \leq 1] \\
\Downarrow \text{[cons]} \\
[y * x! = a! \wedge x = 1] \\
\Downarrow \text{[cons]} \\
[y = a!]
\end{array}
$$

Bew. totaler Korrektheit: Fakultät (18)

Damit haben wir wie gewünscht insgesamt gezeigt:

Die Hoaresche Zusicherung

$$[a > 0]$$

$$x := a;\ y := 1;\ \text{while } x > 1 \text{ do } y := y * x; x := x - 1 \text{ od}$$

$$[y = a!]$$

ist gültig im Sinne totaler Korrektheit.

Nachtrag zur totalen Korrektheit (1)

Oft, insbesondere für die von uns betrachteten Beispiele, reicht folgende, weniger allgemeine Regel für while-Schleifen, um Terminierung und insgesamt totale Korrektheit zu zeigen.

$$[\text{while}'_{TK}] \quad \frac{I \Rightarrow t \geq 0,\ \{I \wedge b \wedge t = w\}\ \pi\ \{I \wedge t < w\}}{\{I\}\ \text{while } b \text{ do } \pi \text{ od } \{I \wedge \neg b\}}$$

wobei

- t arithmetischer Term über ganzen Zahlen,
- w ganzzahlige Variable, die in I, b, π und t nicht frei vorkommt,

Beachte: Statt beliebiger Terminationsordnungen hier Festlegung auf eine spezielle Noethersche Ordnung als Terminationsordnung, nämlich $(\mathbb{N}, <)$.

Nachtrag zur totalen Korrektheit (2)

Beweistechnische Anmerkung:

"Zerlegt" man [while$'_{TK}$] wie folgt:

$$[\text{while}''_{TK}] \quad \frac{I \Rightarrow t \geq 0,\ \{I \wedge b\}\ \pi\ \{I\},\ \{I \wedge b \wedge t{=}w\}\ \pi\ \{t{<}w\}}{\{I\}\ \text{while}\ b\ \text{do}\ \pi\ \text{od}\ \{I \wedge \neg b\}}$$

wird deutlich, dass der Nachweis totaler Korrektheit einer Hoareschen Zusicherung besteht aus

- dem Nachweis ihrer partiellen Korrektheit
- dem Nachweis der Termination

Diese Trennung kann im Beweis explizit vollzogen werden. Der Gesamtbeweis wird dadurch modular. Oft gilt, dass der Terminationsnachweis einfach ist.

Randbemerkung: Die obige Trennung kann für [while$_{TK}$] analog vorgenommen werden.

Zur Korrektheit und Vollständigkeit Hoarescher Beweiskalküle

Sei K ein Hoarescher Beweiskalkül (z.B. HK_{PK} und HK_{TK}).

Dann heißt K...

- *korrekt* (engl. *sound*), falls gilt: Ist eine Korrektheitsformel mit K herleitbar/beweisbar, dann ist sie auch semantisch gültig. In Zeichen:

$$\vdash\ \{p\}\ \pi\ \{q\} \Rightarrow\ \models\ \{p\}\ \pi\ \{q\}$$

- *vollständig* (engl. *complete*), falls gilt: Ist eine Korrektheitsformel semantisch gültig, dann ist sie auch mit K herleitbar/beweisbar.

$$\models\ \{p\}\ \pi\ \{q\} \Rightarrow\ \vdash\ \{p\}\ \pi\ \{q\}$$

Zur Korrektheit von HK_{PK} und HK_{TK}

Theorem [Korrektheit von HK_{PK} und HK_{TK}]

1. HK_{PK} ist korrekt, d.h. jede mit HK_{PK} ableitbare Korrektheitsformel ist gültig im Sinne partieller Korrektheit:

$$\vdash_{pk}\ \{p\}\ \pi\ \{q\} \Rightarrow\ \models_{pk}\ \{p\}\ \pi\ \{q\}$$

2. HK_{TK} ist korrekt, d.h. jede mit HK_{TK} ableitbare Korrektheitsformel ist gültig im Sinne totaler Korrektheit:

$$\vdash_{tk}\ [p]\ \pi\ [q] \Rightarrow\ \models_{tk}\ [p]\ \pi\ [q]$$

Beweis ...durch Induktion über die Anzahl der Regelanwendungen im Beweisbaum zur Ableitung der Korrektheitsformel.

Zur Vollständigkeit Hoarescher Beweiskalküle

Generell müssen wir unterscheiden zwischen Vollständigkeit

- *extensionaler* und
- *intensionaler*

Ansätze.

Extensionale vs. intensionale Ansätze

- *Extensional*

  $\leadsto$ Vor- und Nachbedingungen sind durch *Prädikate* beschrieben.

- *Intensional*

  $\leadsto$ Vor- und Nachbedingungen sind durch *Formeln einer Zusicherungssprache* beschrieben.

Zur Vollständigkeit von HK_{PK} & HK_{TK}

Für den extensionalen Ansatz gilt:

Theorem [Vollständigkeit von HK_{PK} und HK_{TK}]

1. HK_{PK} ist vollständig, d.h. jede im Sinne partieller Korrektheit gültige Korrektheitsformel ist mit HK_{PK} ableitbar:

$$\models_{pk} \ \{p\} \ \pi \ \{q\} \Rightarrow \ \vdash_{pk} \ \{p\} \ \pi \ \{q\}$$

2. HK_{TK} ist vollständig, d.h. jede im Sinne totaler Korrektheit gültige Korrektheitsformel ist mit HK_{TK} ableitbar:

$$\models_{tk} \ [p] \ \pi \ [q] \Rightarrow \ \vdash_{tk} \ [p] \ \pi \ [q]$$

Beweis ...durch strukturelle Induktion über den Aufbau von π.

Zur Vollständigkeit von HK_{PK} & HK_{TK}

Für intensionale Ansätze (durch unterschiedliche Wahlen der Zusicherungssprache) gilt Vollständigkeit i.a. nur relativ zur *Entscheidbarkeit* und *Ausdruckskraft* der Zusicherungssprache.

Intuition

- *Entscheidbarkeit*

  ...ist die Gültigkeit von Formeln der Zusicherungssprache algorithmisch verifizierbar bzw. falsifizierbar?

- *Ausdruckskraft*

  ...lassen sich alle Prädikate, insbesondere schwächste und und schwächste liberale Vorbedingungen und Terminationsfunktionen, durch Formeln der Zusicherungssprache beschreiben?

  $\leadsto$ *tieferliegende Frage*: ...lassen sich schwächste Vorbedingungen etc. syntaktisch ausdrücken?

Stichwort: *Relative Vollständigkeit im Sinne von Cook.*

Nachträge zu(r) Vorwärtszuweisungsregel(n)

- Eine *Vorwärtsregel* für die Zuweisung wie

  [ass$_{fwd}$] $\dfrac{\quad — \quad}{\{p\} \ x{:=}t \ \{\exists \, z. \ p[z/x] \wedge \ x{=}t[z/x]\}}$

  mag natürlich erscheinen, ist aber beweistechnisch unangenehm durch das Mitschleppen quantifizierter Formeln.

- *Beachte*: Folgende scheinbar naheliegende quantorfreie Realisierung der Vorwärtszuweisungsregel ist nicht korrekt:

  [ass$_{naive}$] $\dfrac{\quad — \quad}{\{p\} \ x{:=}t \ \{p[t/x]\}}$

  Beweis: Übungsaufgabe

Automatische Ansätze zur Programmverifikation (1)

…*Theorema*-Projekt am RISC, Linz: `http://www.theorema.org`

The Theorema project aims at extending current computer algebra systems by facilities for supporting mathematical proving. The present early-prototype version of the Theorema software system is implemented in Mathematica . The system consists of a general higher-order predicate logic prover and a collection of special provers that call each other depending on the particular proof situations. The individual provers imitate the proof style of human mathematicians and produce human-readable proofs in natural language presented in nested cells. The special provers are intimately connected with the functors that build up the various mathematical domains.

The long-term goal of the project is to produce a complete system which supports the mathematician in creating interactive textbooks, i.e. books containing, besides the ordinary passive text, active text representing algorithms in executable format, as well as proofs which can be studied at various levels of detail, and whose routine parts can be automatically generated. This system will provide a uniform (logic and software) framework in which a working mathematician, without leaving the system, can get computer-support while looping through all phases of the mathematical problem solving cycle

[…]

Automatische Ansätze zur Programmverifikation (2)

Einige Artikel zu Programmverifikation mit *Theorema*:

- Laura Ildico Kovacs and Tudor Jebelean. *Practical Aspects of Imperative Program Verification using Theorema*. In Proceedings of the 5th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2003), Timisoara, Romania, October 1-4, 2003.

  `apache.risc.uni-linz.ac.at/internals/ActivityDB/publications/download/risc_464/synasc03.pdf`

- Laura Ildico Kovacs and Tudor Jebelean. *Generation of Invariants in Theorema*. In Proceedings of the 10th International Symphosium of Mathematics and its Applications, Timisoara, Romania, November 6-9, 2003.

  `www.theorema.org/publication/2003/Laura/Poli_Timisoara_nov.pdf`