Semantik von Programmiersprachen

Markus Lohrey

Universität Siegen

Sommersemester 2015

Literatur

- J. Loeckx, K. Sieber: The Foundations of Program Verification. Wiley/Teubner, 1987.
- Glynn Winskel: The Formal Semantics of Programming Langugages. MIT Press, 1993 (2nd edition 1994).

1.1 Was ist Semantik

Programmiersprachen werden von drei Blickwinkeln betrachtet:

a) Syntax (elementare lexikalische und grammatikalische Struktur; keine Beziehung zur Bedeutung oder Interpretation)

b) Semantik (Bedeutung, Interpretation; z.B. eine Zahl, eine Funktion,...)

c) Pragmatik (Brauchbarkeit, Lesbarkeit, Übersetzbarkeit,...)

Semantik von Programmiersprachen dient nicht nur dazu, den Sinn eines Programms festzulegen, sondern sie soll auch z.B. ein Hilfsmittel für Korrektheitsbeweise darstellen.

1.2 Einfache Beispiele

Beispiel 1.2.1 Binärzahlen

Syntaktische Beschreibung:

$$\text{BIN} = \{w \in \{0,1\}^* \mid w = 0 \text{ oder } w \text{ beginnt mit } 1\}$$

Semantik: Sei $w \in \text{BIN}$.

- $w = 0$: Wert von w = Zahl 0.
- $w = 1$: Wert von w = Zahl 1.
- $w = w'0$ und die Länge von w' ist größer als 0: Wert von w = Doppelte des Wertes von w'.
- $w = w'1$ und die Länge von w' ist größer als 0: Wert von w = Doppelte des Wertes von w' plus 1.

Beachte: In den ersten beiden Fällen ist $w = 0$ ($w = 1$) der Buchstabe 0 (1) während die Zahl 0 (1) das Element von $\mathbb{N}$ bezeichnet.

Beispiel 1.2.1 Unäre Zahlen

Syntaktische Beschreibung:

$$\text{UNAT} = 0^*1$$

Semantik:

Sei $w = 0^n1$. Dann ist der Wert von w die Zahl n.

Beispiel 1.2.1 Unäre Zahlen

Syntaktische Beschreibung:

$$\text{UNAT} = 0^*1$$

Semantik:

Sei $w = 0^n1$. Dann ist der Wert von w die Zahl n.

Probleme, wie *"Ist n eine Primzahl?"*, können für BIN schwer und für UNAT ganz einfach sein.

Beispiel 1.2.2 Rationale (oder reguläre) Ausdrücke

Syntax: Sei Γ ein endliches Alphabet.

Die Menge der regulären Ausdrücke $\mathcal{E}_\Gamma$ über dem Alphabet Γ ist die kleinste Menge mit folgenden Eigenschaften:

- $\emptyset$ und alle $a \in \Gamma$ sind Elemente von $\mathcal{E}_\Gamma$.

Beispiel 1.2.2 Rationale (oder reguläre) Ausdrücke

Syntax: Sei Γ ein endliches Alphabet.

Die Menge der regulären Ausdrücke $\mathcal{E}_\Gamma$ über dem Alphabet Γ ist die kleinste Menge mit folgenden Eigenschaften:

- $\emptyset$ und alle $a \in \Gamma$ sind Elemente von $\mathcal{E}_\Gamma$.
- Wenn α und β Elemente von $\mathcal{E}_\Gamma$ sind, dann sind auch $(\alpha\beta)$, $(\alpha \cup \beta)$ und $(\alpha)^*$ in $\mathcal{E}_\Gamma$.

ε ist eine Abkürzung von $\emptyset^*$

Beispiel 1.2.2 Rationale (oder reguläre) Ausdrücke

Syntax: Sei Γ ein endliches Alphabet.

Die Menge der regulären Ausdrücke $\mathcal{E}_\Gamma$ über dem Alphabet Γ ist die kleinste Menge mit folgenden Eigenschaften:

- $\emptyset$ und alle $a \in \Gamma$ sind Elemente von $\mathcal{E}_\Gamma$.
- Wenn α und β Elemente von $\mathcal{E}_\Gamma$ sind, dann sind auch $(\alpha\beta)$, $(\alpha \cup \beta)$ und $(\alpha)^*$ in $\mathcal{E}_\Gamma$.

ε ist eine Abkürzung von $\emptyset^*$

Semantik:

Wie üblich (GTI) kann ein Ausdruck mit einer Teilmenge von Γ^* identifiziert werden, also mit einer formalen Sprache.

Beispiel 1.2.3 Nicht-deterministische endliche Automaten (NFA)

Syntax: $(Q, \Gamma, \delta, I, F) \in \mathrm{NFA}_\Gamma$, falls

- Q endliche Menge
- Γ endl. Alphabet $(Q \cap \Gamma = \emptyset)$
- $\delta \subseteq Q \times \Gamma \times Q$
- $I \subseteq Q$
- $F \subseteq Q$

Semantik:

Der Wert von $A = (Q, \Gamma, \delta, I, F) \in \mathrm{NFA}_\Gamma$ ist die Menge der von dem Automat A akzeptierten Wörter aus Γ^*.

1.3 Semantik-Funktionen

Als Semantik-Funktion bezeichnen wir die Abbildung, die zu jeder syntaktisch korrekten Konstruktion ihre Bedeutung als Resultat liefert:

$$\mathcal{B} : \mathsf{BIN} \to \mathbb{N}$$

Beispiel: $\mathcal{B}(101010) = 42$.

1.3 Semantik-Funktionen

Als Semantik-Funktion bezeichnen wir die Abbildung, die zu jeder syntaktisch korrekten Konstruktion ihre Bedeutung als Resultat liefert:

$$\mathcal{B} : \mathsf{BIN} \to \mathbb{N}$$

Beispiel: $\mathcal{B}(101010) = 42$.

Was ist 42?

1.3 Semantik-Funktionen

Als Semantik-Funktion bezeichnen wir die Abbildung, die zu jeder syntaktisch korrekten Konstruktion ihre Bedeutung als Resultat liefert:

$$\mathcal{B} : \mathsf{BIN} \to \mathbb{N}$$

Beispiel: $\mathcal{B}(101010) = 42$.

Was ist 42?

In der Mengenleere kann man 42 wie folgt definieren:

1.3 Semantik-Funktionen

Als Semantik-Funktion bezeichnen wir die Abbildung, die zu jeder syntaktisch korrekten Konstruktion ihre Bedeutung als Resultat liefert:

$$\mathcal{B} : \mathsf{BIN} \to \mathbb{N}$$

Beispiel: $\mathcal{B}(101010) = 42$.

Was ist 42?

In der Mengenleere kann man 42 wie folgt definieren:

$$42 = \{0, \ldots 41\}$$
$$41 = \{0, \ldots 40\}$$

1.3 Semantik-Funktionen

Als Semantik-Funktion bezeichnen wir die Abbildung, die zu jeder syntaktisch korrekten Konstruktion ihre Bedeutung als Resultat liefert:

$$\mathcal{B} : \mathsf{BIN} \to \mathbb{N}$$

Beispiel: $\mathcal{B}(101010) = 42$.

Was ist 42?

In der Mengenleere kann man 42 wie folgt definieren:

$$\begin{aligned} 42 &= \{0, \ldots 41\} \\ 41 &= \{0, \ldots 40\} \\ &\vdots \\ 0 &= \emptyset \end{aligned}$$

1.3 Semantik-Funktionen

Als Semantik-Funktion bezeichnen wir die Abbildung, die zu jeder syntaktisch korrekten Konstruktion ihre Bedeutung als Resultat liefert:

$$\mathcal{B} : \mathsf{BIN} \to \mathbb{N}$$

Beispiel: $\mathcal{B}(101010) = 42$.

Was ist 42?

In der Mengenleere kann man 42 wie folgt definieren:

$$\begin{aligned} 42 &= \{0, \ldots 41\} \\ 41 &= \{0, \ldots 40\} \\ &\vdots \\ 0 &= \emptyset \end{aligned}$$

Alle mathematischen Objekte, die hier untersucht werden, sind letztendlich Mengen.

1.4 Die Sprache IMP

IMP steht für Imperative Sprache

1.4 Die Sprache IMP

IMP steht für Imperative Sprache

Grundbereiche:

- Natürliche Zahlen $\mathbb{N}$,
- Wahrheitswerte $\mathbb{B} = \{\textbf{true}, \textbf{false}\} = \{1, 0\}$.

Variablen: $\mathbb{V} = \{X_1, X_2, X_3, \ldots\}$ (abzählbare Menge von Bezeichnern)

Mit $\textbf{Loc} \subseteq \mathbb{V}$ (Locations) wird die Menge der in einem Programm benutzten Variablen bezeichnen.

Menge der Speicherzustände: $\Sigma = \mathbb{N}^{\textbf{Loc}} = \{\sigma \colon \textbf{Loc} \to \mathbb{N}\}$

Konvention:

Falls z.B. $\mathbf{Loc} = \{X_1, X_2, X_3\}$ mit der impliziten Ordnung $X_1 < X_2 < X_2$, dann (übersichtlicher):

$$\Sigma = \mathbb{N}^{\mathbf{Loc}} = \mathbb{N} \times \mathbb{N} \times \mathbb{N} = \mathbb{N}^3$$

Beispiel: $(3, 1, 2)$ ist dann der Speicherzustand $\sigma \in \Sigma$ mit

$$\sigma(X_1) = 3 \quad \sigma(X_2) = 1 \quad \sigma(X_3) = 2.$$

IMP verfügt über drei Arten syntaktischer Konstrukte:

- Arithmetische Ausdrücke
  (**Aexp**)
- Boolesche Ausdrücke
  (**Bexp**)
- Anweisungen bzw. Programme
  (**Cmd**)

Aexp

Syntaktische Definition:

$$a ::= n \mid X \mid (a_1 + a_2) \mid (a_1 - a_2) \mid (a_1 \cdot a_2)$$

Dabei ist $n \in \mathbb{N}$, $X \in \mathbb{V}$ und $a_1, a_2 \in$ **Aexp**.

Die Klammern können manchmal weggelassen werden, wenn dadurch keine Mehrdeutigkeit entsteht.

Das heißt also:

1. Natürliche Zahlen und Variablen bilden arithmetische Ausdrücke.
2. Wenn a_1 und a_2 zu **Aexp** gehören, dann auch $(a_1 + a_2)$, $(a_1 - a_2)$, $(a_1 \cdot a_2)$.
3. Sonst gehört nichts zu **Aexp**.

Beispiel: $((((X_1 \cdot 3) - 5) + X_2) - (X_3 \cdot 88)) \in$ **Aexp**

Beispiel: $((((X_1 \cdot 3) - 5) + X_2) - (X_3 \cdot 88)) \in$ **Aexp**

Streng genommmen gilt z.B. $5X_2 \notin$ **Aexp**

Auswertung: Arithmetische Ausdrücke werten sich je nach Speicherzustand zu einer natürlichen Zahl aus.

Formal haben wir also eine Funktion vom Typ

$$\textbf{Aexp} \times \Sigma \to \mathbb{N}$$

Wir schreiben $(a, \sigma) \to n$, falls der Ausdruck a im Speicherzustand σ zur Zahl n ausgewertet wird.

Die Auswertung ist wie folgt definiert:

$$
\begin{array}{rll}
(n, \sigma) & \to n & (n \in \mathbb{N}, \sigma \in \Sigma) \\
(X, \sigma) & \to \sigma(X) & (X \in \mathbf{Loc}, \sigma \in \Sigma) \\
((a_1 + a_2), \sigma) & \to n_1 + n_2 & \text{wobei } (a_i, \sigma) \to n_i \text{ für } i \in \{1, 2\} \\
((a_1 - a_2), \sigma) & \to \max\{0, n_1 - n_2\} & \text{wobei } (a_i, \sigma) \to n_i \text{ für } i \in \{1, 2\} \\
((a_1 \cdot a_2), \sigma) & \to n_1 \cdot n_2 & \text{wobei } (a_i, \sigma) \to n_i \text{ für } i \in \{1, 2\}
\end{array}
$$

Die Zeichen $+, -, \cdot$ auf der rechten Seite bezeichnen die bekannten Operationen auf natürlichen Zahlen!

Wir ordnen also einem arithmetischen Ausdruck und einer Speicherzustand eine Zahl zu.

Alternative Sichtweise: Wir können auch einem arithmetischen Ausdruck eine Funktion von Speicherzuständen in Zahlen zuordnen:

Hierzu definieren wir die Semantik-Funktion

$$\mathcal{A}\colon \textbf{Aexp} \to (\Sigma \to \mathbb{N}) \text{ oder } \mathcal{A}\colon \textbf{Aexp} \to \mathbb{N}^{\Sigma}$$

wie folgt (anstatt $\mathcal{A}(a)$ schreiben wir $\mathcal{A}[\![a]\!]$ und anstatt $\mathcal{A}[\![a]\!](\sigma)$ schreiben wir $\mathcal{A}[\![a]\!]\sigma$):

$$\mathcal{A}[\![a]\!]\sigma = n \quad \Longleftrightarrow \quad (a, \sigma) \to n$$

Bexp

Syntaktische Definition:

$$b ::= \textbf{true} \mid \textbf{false} \mid a_1 = a_2 \mid a_1 < a_2 \mid a_1 > a_2 \mid a_1 \neq a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \mid b_1 \Rightarrow b_2 \mid b_1 \Leftrightarrow b_2$$

Dabei seien $a_1, a_2 \in \textbf{Aexp}$, $b, b_1, b_2 \in \textbf{Bexp}$.

Auswertung

Wir definieren $(b, \sigma) \to t$ für geeignetes $t \in \mathbb{B}$ und schreiben auch hier

$$\mathcal{B}[\![b]\!]\sigma = t$$

für eine Funktion

$$\mathcal{B}\colon \textbf{Bexp} \to (\Sigma \to \mathbb{B})$$

Die Auswertung boolescher Ausdrücke geschieht nach folgenden Regeln:

$$
\begin{array}{rll}
(t, \sigma) \to t & & \text{für } t \in \mathbb{B}, \sigma \in \Sigma \\
(a_1 = a_2, \sigma) \to \textbf{true} & & \text{falls } (a_i, \sigma) \to n_i \text{ für } i \in \{1, 2\} \text{ und } n_1 = n_2 \text{ gilt} \\
(a_1 = a_2, \sigma) \to \textbf{false} & & \text{falls } (a_i, \sigma) \to n_i \text{ für } i \in \{1, 2\} \text{ und } n_1 \neq n_2 \text{ gilt}
\end{array}
$$

Für die Ausdrücke $(a_1 < a_2, \sigma)$, $(a_1 > a_2, \sigma)$ und $(a_1 \neq a_2, \sigma)$ gelten analoge Regeln.

$$(\neg b, \sigma) \to \textbf{true} \text{ wenn } (b, \sigma) \to \textbf{false}$$
$$(\neg b, \sigma) \to \textbf{false} \text{ wenn } (b, \sigma) \to \textbf{true}$$
$$(b_1 \vee b_2, \sigma) \to \textbf{true} \text{ wenn } (b_1, \sigma) \to \textbf{true} \text{ oder } (b_2, \sigma) \to \textbf{true} \text{ gilt}$$
$$(b_1 \vee b_2, \sigma) \to \textbf{false} \text{ wenn } (b_1, \sigma) \to \textbf{false} \text{ und } (b_2, \sigma) \to \textbf{false} \text{ gilt}$$
$$(b_1 \wedge b_2, \sigma) \to \textbf{true} \text{ wenn } (b_1, \sigma) \to \textbf{true} \text{ und } (b_2, \sigma) \to \textbf{true} \text{ gilt}$$
$$(b_1 \wedge b_2, \sigma) \to \textbf{false} \text{ wenn } (b_1, \sigma) \to \textbf{false} \text{ oder } (b_2, \sigma) \to \textbf{false} \text{ gilt}$$
$$(b_1 \Rightarrow b_2, \sigma) \to \textbf{true} \text{ wenn } (b_1, \sigma) \to \textbf{false} \text{ oder } (b_2, \sigma) \to \textbf{true} \text{ gilt}$$
$$(b_1 \Rightarrow b_2, \sigma) \to \textbf{false} \text{ wenn } (b_1, \sigma) \to \textbf{true} \text{ und } (b_2, \sigma) \to \textbf{false} \text{ gilt}$$

Analog für $(b_1 \Leftrightarrow b_2, \sigma)$.

Cmd

Syntaktische Definition ($X \in \mathbb{V}$, $a \in$ **Aexp**, $b \in$ **Bexp**, $c, c_1, c_2 \in$ **Cmd**):

$$c ::= \textbf{skip} \mid X := a \mid c_1; c_2 \mid \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ fi} \mid \textbf{while } b \textbf{ do } c \textbf{ od}$$

Auswertung

Einem gegebenen Programm $c \in$ **Cmd** und einem Speicherzustand $\sigma \in \Sigma$ wird ein neuer Speicherzustand $\sigma' \in \Sigma$ zugeordnet:

$$\textbf{Cmd} \times \Sigma \rightarrow_p \Sigma$$

oder auch

$$\mathcal{C} : \textbf{Cmd} \rightarrow (\Sigma \rightarrow_p \Sigma)$$

Das p am Pfeil steht für „partiell“.

Für eine partielle Funktion $f : A \rightarrow B$ kann $f(a)$ für ein $a \in A$ undefiniert sein.

Beispiel:

$$
\begin{array}{rcl}
c & \equiv & X_2 := 1; \\
 & & \textbf{while } X_1 > 1 \textbf{ do} \\
 & & \qquad X_2 := X_2 \cdot X_1; \\
 & & \qquad X_1 := X_1 - 1 \\
 & & \textbf{od}
\end{array}
$$

Σ besteht aus Zahlenpaaren, da nur X_1 und X_2 vorkommen:

$$\Sigma = \{(n_1, n_2) \mid n_1, n_2 \in \mathbb{N}\}$$

Dabei sei für $\sigma = (n_1, n_2) \in \Sigma$ erfüllt:

$$\sigma(X_1) = n_1, \qquad \sigma(X_2) = n_2$$

Was ist $\mathcal{C}[\![c]\!]\sigma$?

Offensichtlich gilt:

- Wenn $\sigma = (n_1, n_2)$ und $\sigma' = (n_1, n_2')$, dann $\mathcal{C}[\![c]\!]\sigma = \mathcal{C}[\![c]\!]\sigma'$.
- Für alle $\sigma = (n_1, n_2)$ mit $n_1 > 0$ existiert ein n mit $\mathcal{C}[\![c]\!]\sigma = (1, n)$.

(Warum?)

Was ist $\mathcal{C}[\![c]\!]\sigma$?

Offensichtlich gilt:

- Wenn $\sigma = (n_1, n_2)$ und $\sigma' = (n_1, n_2')$, dann $\mathcal{C}[\![c]\!]\sigma = \mathcal{C}[\![c]\!]\sigma'$.
- Für alle $\sigma = (n_1, n_2)$ mit $n_1 > 0$ existiert ein n mit $\mathcal{C}[\![c]\!]\sigma = (1, n)$.

(Warum?)

In $\mathcal{C}[\![c]\!](n_1, n_2) = (1, n)$ hängt also eigentlich n nur von n_1 ab.

Daher betrachten wir

$$\mathsf{IN}(X_1)\ c\ \mathsf{OUT}(X_2)$$

Was ist der Wert von X_2 nach Ausführung von c, wenn zuvor X_1 einen gegebenen Wert N hatte?

Diese Betrachtungen erfolgten auf einem intuitiven Begriff der Anweisungen unserer Programmiersprache IMP.

Diese Betrachtungen erfolgten auf einem intuitiven Begriff der Anweisungen unserer Programmiersprache IMP.

Um die Wirkung der Anweisung c formal fassen zu können und eine Behauptung wie, c berechne die Fakultätsfunktion, auch beweisbar zu machen, verwenden wir die formale Semantik!

1.6 Operationale Semantik der Sprache IMP (am Beispiel)

Wir denken uns eine

IMP-Maschine

die das Programm unserer naiven Vorstellung gemäß abarbeitet.

1.6 Operationale Semantik der Sprache IMP (am Beispiel)

Wir denken uns eine

IMP-Maschine

die das Programm unserer naiven Vorstellung gemäß abarbeitet.

Sei z.B. der Startzustand $\sigma = (3, 0)$ gegeben, d.h.

$$\sigma(X_1) = 3$$

$$\sigma(X_2) = 0$$

Dann erhalten wir:

$(c, (3, 0)) \quad \rightarrow (w, (3, 1))$ $\qquad\qquad$ $w =$ gesamte while-Schleife

$$
\begin{aligned}
(c, (3, 0)) &\longrightarrow (w, (3, 1)) && w = \text{gesamte while-Schleife} \\
&\longrightarrow (c_1; c_2; w, (3, 1)) && c_1\colon X_2 := X_2 \cdot X_1 \\
& && c_2\colon X_1 := X_1 - 1
\end{aligned}
$$

$$
\begin{aligned}
(c,(3,0)) &\longrightarrow (w,(3,1)) && w = \text{gesamte while-Schleife} \\
&\longrightarrow (c_1;c_2;w,(3,1)) && c_1\colon X_2 := X_2 \cdot X_1 \\
& && c_2\colon X_1 := X_1 - 1 \\
&\longrightarrow (c_2;w,(3,3))
\end{aligned}
$$

$$
\begin{aligned}
(c,(3,0)) &\longrightarrow (w,(3,1)) && w = \text{gesamte while-Schleife} \\
&\longrightarrow (c_1;c_2;w,(3,1)) && c_1\colon X_2 := X_2 \cdot X_1 \\
& && c_2\colon X_1 := X_1 - 1 \\
&\longrightarrow (c_2;w,(3,3)) \\
&\longrightarrow (w,(2,3))
\end{aligned}
$$

$$
\begin{aligned}
(c,(3,0)) &\longrightarrow (w,(3,1)) && w = \text{gesamte while-Schleife}\\
&\longrightarrow (c_1;c_2;w,(3,1)) && c_1\colon X_2 := X_2 \cdot X_1\\
& && c_2\colon X_1 := X_1 - 1\\
&\longrightarrow (c_2;w,(3,3))\\
&\longrightarrow (w,(2,3))\\
&\longrightarrow (c_1;c_2;w,(2,3))
\end{aligned}
$$

$$
\begin{aligned}
(c,(3,0)) &\longrightarrow (w,(3,1)) && w = \text{gesamte while-Schleife}\\
&\longrightarrow (c_1;c_2;w,(3,1)) && c_1\colon X_2 := X_2 \cdot X_1\\
& && c_2\colon X_1 := X_1 - 1\\
&\longrightarrow (c_2;w,(3,3))\\
&\longrightarrow (w,(2,3))\\
&\longrightarrow (c_1;c_2;w,(2,3))\\
&\longrightarrow (c_2;w,(2,6))
\end{aligned}
$$

$$
\begin{aligned}
(c,(3,0)) &\longrightarrow (w,(3,1)) && w = \text{gesamte while-Schleife}\\
&\longrightarrow (c_1;c_2;w,(3,1)) && c_1\colon X_2 := X_2 \cdot X_1\\
& && c_2\colon X_1 := X_1 - 1\\
&\longrightarrow (c_2;w,(3,3))\\
&\longrightarrow (w,(2,3))\\
&\longrightarrow (c_1;c_2;w,(2,3))\\
&\longrightarrow (c_2;w,(2,6))\\
&\longrightarrow (w,(1,6))
\end{aligned}
$$

$$
\begin{aligned}
(c,(3,0)) &\longrightarrow (w,(3,1)) && w = \text{gesamte while-Schleife}\\
&\longrightarrow (c_1;c_2;w,(3,1)) && c_1\colon X_2 := X_2 \cdot X_1\\
& && c_2\colon X_1 := X_1 - 1\\
&\longrightarrow (c_2;w,(3,3))\\
&\longrightarrow (w,(2,3))\\
&\longrightarrow (c_1;c_2;w,(2,3))\\
&\longrightarrow (c_2;w,(2,6))\\
&\longrightarrow (w,(1,6))\\
&\longrightarrow (\textbf{skip},(1,6))
\end{aligned}
$$

$$
\begin{aligned}
(c,(3,0)) &\longrightarrow (w,(3,1)) && w = \text{gesamte while-Schleife}\\
&\longrightarrow (c_1;c_2;w,(3,1)) && c_1\colon X_2 := X_2 \cdot X_1\\
& && c_2\colon X_1 := X_1 - 1\\
&\longrightarrow (c_2;w,(3,3))\\
&\longrightarrow (w,(2,3))\\
&\longrightarrow (c_1;c_2;w,(2,3))\\
&\longrightarrow (c_2;w,(2,6))\\
&\longrightarrow (w,(1,6))\\
&\longrightarrow (\textbf{skip},(1,6))\\
&\longrightarrow (1,6)
\end{aligned}
$$

$$
\begin{aligned}
(c,(3,0)) &\longrightarrow (w,(3,1)) && w = \text{gesamte while-Schleife}\\
&\longrightarrow (c_1;c_2;w,(3,1)) && c_1\colon X_2 := X_2 \cdot X_1\\
& && c_2\colon X_1 := X_1 - 1\\
&\longrightarrow (c_2;w,(3,3))\\
&\longrightarrow (w,(2,3))\\
&\longrightarrow (c_1;c_2;w,(2,3))\\
&\longrightarrow (c_2;w,(2,6))\\
&\longrightarrow (w,(1,6))\\
&\longrightarrow (\textbf{skip},(1,6))\\
&\longrightarrow (1,6)
\end{aligned}
$$

Der Endzustand $(1,6)$ sagt, dass $3! = 6$ richtig berechnet wird.

1.7 Denotationale Semantik der Sprache IMP (am Beispiel)

Zu einem Programm der Form

$$p \equiv \mathsf{IN}(X_1, \ldots, X_m)\ c\ \mathsf{OUT}(Y_1, \ldots, Y_n)$$

betrachten wir die partielle Funktion

$$\mathcal{C}[\![p]\!] : \mathbb{N}^m \ \rightarrow_p \ \mathbb{N}^n,$$

die wie folgt definiert ist:

Wird c auf den Speicherzustand, der durch Eingabe $k_1, \ldots, k_m$ für $X_1, \ldots, X_m$ gegeben ist (alle anderen Variablen 0), gestartet, so ist

$$\mathcal{C}[\![p]\!](k_1, \ldots, k_m)$$

genau dann definiert, wenn c terminiert. Der Funktionswert ist das n-Tupel der Variablenwerte von $Y_1, \ldots, Y_n$ nach der Ausführung.

Ohne Angabe von IN und OUT betrachten wir

$$\mathcal{C}[\![c]\!] : \Sigma \ \rightarrow_p \ \Sigma,$$

d.h. alle verwendeten Variablen werden sowohl als Eingabe-, wie auch als Ausgabewerte angesehen.

Beispiel:

$c \equiv$ **while** $X_1 \neq X_2$ **do**
if $X_1 > X_2$ **then**
$X_1 := X_1 - X_2$
else
$X_2 := X_2 - X_1$
fi
od

Betrachte $p \equiv \mathsf{IN}(X_1, X_2)\ c\ \mathsf{OUT}(X_2)$.

Ohne Angabe von IN und OUT betrachten wir

$$\mathcal{C}[\![c]\!] : \Sigma \ \rightarrow_p \ \Sigma,$$

d.h. alle verwendeten Variablen werden sowohl als Eingabe-, wie auch als Ausgabewerte angesehen.

Beispiel:

$$\begin{array}{lll} c \ \equiv & \textbf{while } X_1 \neq X_2 \textbf{ do} & \\ & \quad \textbf{if } X_1 > X_2 \textbf{ then} & \\ & \quad\quad X_1 := X_1 - X_2 & \\ & \quad \textbf{else} & \\ & \quad\quad X_2 := X_2 - X_1 & \\ & \quad \textbf{fi} & \\ & \textbf{od} & \end{array}$$

Betrachte $p \equiv \mathsf{IN}(X_1, X_2)\ c\ \mathsf{OUT}(X_2)$.

Welche Funktion wird berechnet?

Behauptung: Für $f = \mathcal{C}[\![p]\!]$ gilt:

$$
\begin{aligned}
f(0,0) &= 0 \\
f(m,0) &= \text{undef} \quad (m > 0) \\
f(0,n) &= \text{undef} \quad (n > 0) \\
f(m,n) &= \text{ggT}(m,n) \quad (m > 0, n > 0)
\end{aligned}
$$

Beweis:

Behauptung: Für $f = \mathcal{C}[\![p]\!]$ gilt:

$$
\begin{aligned}
f(0,0) &= 0 \\
f(m,0) &= \text{undef} \quad (m > 0) \\
f(0,n) &= \text{undef} \quad (n > 0) \\
f(m,n) &= \text{ggT}(m,n) \quad (m > 0, n > 0)
\end{aligned}
$$

Beweis:

Mühsam „zu Fuß“.

Behauptung: Für $f = \mathcal{C}[\![p]\!]$ gilt:

$$
\begin{aligned}
&f(0,0) = 0 \\
&f(m,0) = \text{undef} \quad (m > 0) \\
&f(0,n) = \text{undef} \quad (n > 0) \\
&f(m,n) = \text{ggT}(m,n) \quad (m > 0, n > 0)
\end{aligned}
$$

Beweis:

Mühsam „zu Fuß“.

Das Programm berechnet also **nicht** den ggT.

1.8 Axiomatische Semantik der Sprache IMP (am Beispiel)

Verwende Zusicherungen und Invarianten.

Betrachte wir noch einmal das Fakultätsprogramm:

$$
\begin{array}{rcl}
c & \equiv & X_2 := 1; \\
 & & \textbf{while } X_1 > 1 \textbf{ do} \\
 & & \qquad X_2 := X_2 \cdot X_1; \\
 & & \qquad X_1 := X_1 - 1 \\
 & & \textbf{od}
\end{array}
$$

Ziel: Beweise $\{X_1 = N\}\ c\ \{X_2 = N!\}$.

Das Ziel bedeutet:

Wenn c auf einen Speicherzustand mit $X_1 = N$ angewendet wird, dann hat nach der Ausführung X_2 den Wert $N!$. Insbesondere terminiert c.

Dies soll nun bewiesen werden.

Annahme: $X_1 > 0$ (sonst ist das Ziel trivialerweise erreicht, da $0! = 1$)

Das Ziel bedeutet:

Wenn c auf einen Speicherzustand mit $X_1 = N$ angewendet wird, dann hat nach der Ausführung X_2 den Wert $N!$. Insbesondere terminiert c.

Dies soll nun bewiesen werden.

Annahme: $X_1 > 0$ (sonst ist das Ziel trivialerweise erreicht, da $0! = 1$)

Wegen $c \equiv X_2 := 1; w$ mit

$$w \equiv \textbf{while } X_1 > 1 \textbf{ do } \ldots \textbf{ od}$$

erhalten wir die Äquivalenz von

$$\{X_1 = N\}\ c\ \{X_2 = N!\}$$

mit

$$\{X_1 = N \wedge X_2 = 1\}\ w\ \{X_2 = N!\}.$$

Aus $X_1 = N \wedge X_2 = 1$ folgt $X_2 \cdot (X_1!) = N!$

Aus $X_1 = N \wedge X_2 = 1$ folgt $X_2 \cdot (X_1!) = N!$

Nach Ausführung von w gilt $X_1 = 1$.

Aus $X_1 = N \wedge X_2 = 1$ folgt $X_2 \cdot (X_1!) = N!$

Nach Ausführung von w gilt $X_1 = 1$.

Wenn immer noch $X_2 \cdot X_1! = N!$ gilt, dann gilt $X_2 = N!$, wie gewünscht.

Aus $X_1 = N \wedge X_2 = 1$ folgt $X_2 \cdot (X_1!) = N!$

Nach Ausführung von w gilt $X_1 = 1$.

Wenn immer noch $X_2 \cdot X_1! = N!$ gilt, dann gilt $X_2 = N!$, wie gewünscht.

Aus $X_1 = N \wedge X_2 = 1$ folgt $X_2 \cdot (X_1!) = N!$

Nach Ausführung von w gilt $X_1 = 1$.

Wenn immer noch $X_2 \cdot X_1! = N!$ gilt, dann gilt $X_2 = N!$, wie gewünscht.

Wir wollen zeigen, dass $X_2 \cdot X_1! = N!$ eine Schleifen-Invariante von w ist.

Aus $X_1 = N \wedge X_2 = 1$ folgt $X_2 \cdot (X_1!) = N!$

Nach Ausführung von w gilt $X_1 = 1$.

Wenn immer noch $X_2 \cdot X_1! = N!$ gilt, dann gilt $X_2 = N!$, wie gewünscht.

Wir wollen zeigen, dass $X_2 \cdot X_1! = N!$ eine Schleifen-Invariante von w ist.

Aus $X_1 = N \wedge X_2 = 1$ folgt $X_2 \cdot (X_1!) = N!$

Nach Ausführung von w gilt $X_1 = 1$.

Wenn immer noch $X_2 \cdot X_1! = N!$ gilt, dann gilt $X_2 = N!$, wie gewünscht.

Wir wollen zeigen, dass $X_2 \cdot X_1! = N!$ eine Schleifen-Invariante von w ist.

Es genügt folgende Aussage zu zeigen:

Aus $X_1 = N \wedge X_2 = 1$ folgt $X_2 \cdot (X_1!) = N!$

Nach Ausführung von w gilt $X_1 = 1$.

Wenn immer noch $X_2 \cdot X_1! = N!$ gilt, dann gilt $X_2 = N!$, wie gewünscht.

Wir wollen zeigen, dass $X_2 \cdot X_1! = N!$ eine Schleifen-Invariante von w ist.

Es genügt folgende Aussage zu zeigen:

$\{X_1 > 1 \wedge X_2 \cdot X_1! = N!\}$

$$X_2 := X_2 \cdot X_1; X_1 := X_1 - 1$$

$$\{X_2 \cdot (X_1!) = N! \wedge X_1 \geq 1\}.$$

Aus $X_1 = N \wedge X_2 = 1$ folgt $X_2 \cdot (X_1!) = N!$

Nach Ausführung von w gilt $X_1 = 1$.

Wenn immer noch $X_2 \cdot X_1! = N!$ gilt, dann gilt $X_2 = N!$, wie gewünscht.

Wir wollen zeigen, dass $X_2 \cdot X_1! = N!$ eine Schleifen-Invariante von w ist.

Es genügt folgende Aussage zu zeigen:

$\{X_1 > 1 \wedge X_2 \cdot X_1! = N!\}$

$$X_2 := X_2 \cdot X_1; X_1 := X_1 - 1$$

$$\{X_2 \cdot (X_1!) = N! \wedge X_1 \geq 1\}.$$

Diese kann man in zwei Schritten herleiten.

Schritt 1.

$$\{X_1 > 1 \wedge X_2 \cdot X_1! = N!\}\ X_2 := X_2 \cdot X_1$$

Schritt 1.

$$\{X_1 > 1 \wedge X_2 \cdot X_1! = N!\}\ X_2 := X_2 \cdot X_1\ \{X_1 > 1 \wedge X_2 \cdot X_1! = N! \cdot X_1\}$$

Schritt 1.

$$\{X_1 > 1 \wedge X_2 \cdot X_1! = N!\}\ X_2 := X_2 \cdot X_1\ \{X_1 > 1 \wedge X_2 \cdot X_1! = N! \cdot X_1\}$$

Schritt 2. Wegen

$$\{X_1 > 1 \wedge X_2 \cdot X_1! = N! \cdot X_1\} = \{X_1 > 1 \wedge X_2 \cdot (X_1 - 1)! = N!\}$$

gilt:

$$\{X_1 > 1 \wedge X_2 \cdot (X_1 - 1)! = N!\}\ X_1 := X_1 - 1$$

Schritt 1.

$$\{X_1 > 1 \wedge X_2 \cdot X_1! = N!\}\ X_2 := X_2 \cdot X_1\ \{X_1 > 1 \wedge X_2 \cdot X_1! = N! \cdot X_1\}$$

Schritt 2. Wegen

$$\{X_1 > 1 \wedge X_2 \cdot X_1! = N! \cdot X_1\} = \{X_1 > 1 \wedge X_2 \cdot (X_1 - 1)! = N!\}$$

gilt:

$$\{X_1 > 1 \wedge X_2 \cdot (X_1 - 1)! = N!\}\ X_1 := X_1 - 1\ \{X_1 \geq 1 \wedge X_2 \cdot X_1! = N!\}$$

2. Mathematische Grundlagen

2.1 Relationen und Abbildungen

Definition (Relationen, Verkettung von Relationen

- Seien X, Y Mengen. Eine Teilmenge $R \subseteq X \times Y$ heißt Relation.
- Seien X, Y, Z Mengen, $R \subseteq X \times Y$, $S \subseteq Y \times Z$ Relationen.
  Mit $R \circ S$ bezeichnen wir die Relation

$$R \circ S = \{(x, z) \in X \times Z \mid \exists y \in Y : (x, y) \in R \wedge (y, z) \in S\}$$

- Für eine Menge X sei id_X die identische Relation:

$$\mathrm{id}_X = \{(x, x) \in X \times X \mid x \in X\}$$

Definition (partielle und totale Funktionen)

Eine Relation $R \subseteq X \times Y$ heißt partielle Abbildung (partielle Funktion, partiell definierte Abbildung), falls für alle $x \in X$ höchstens ein $y \in Y$ existiert mit $(x, y) \in R$.

Wir schreiben dafür auch $f : X \rightarrow_p Y$ mit

$$f(x) = \begin{cases} y & \text{falls } (x, y) \in R \\ \text{undefiniert} & \text{falls } \forall y \in Y : (x, y) \notin R \end{cases}$$

und sagen, R sei der Graph der partiellen Funktion f: $R = \text{graph}(f)$.

Die partielle Abbildung $f : X \rightarrow_p Y$ heißt total, falls gilt:

$$\forall x \, \exists y : (x, y) \in \text{graph}(f).$$

Die Menge $\mathrm{Rel}(X) = 2^{X \times X}$ der Relationen auf X bildet mit der Operation $\circ$ ein Monoid.

Das neutrale Element ist id_X.

Die Relation $R \in \mathrm{Rel}(X)$ heißt

- reflexiv, wenn $\mathrm{id}_X \subseteq R$
- irreflexiv, wenn $\mathrm{id}_X \cap R = \emptyset$
- symmetrisch, wenn $\forall x, y \in X : (x, y) \in R \Rightarrow (y, x) \in R$
- antisymmetrisch, wenn $\forall x, y \in X : (x, y) \in R \wedge (y, x) \in R \Rightarrow x = y$
- transitiv, wenn $\forall x, y, z \in X : (x, y) \in R \wedge (y, z) \in R \Rightarrow (x, z) \in R$

Wir definieren die Iterationen der Relation $R \subseteq X \times X$ wie folgt:

- $R^0 := \mathrm{id}_X$
- $R^1 := R$
- $R^{i+1} := R \circ R^i$ für $i \geq 1$
- $R^+ := \bigcup_{i \geq 1} R^i$ (transitive Hülle)
- $R^* := \bigcup_{i \geq 0} R^i$ (reflexive transitive Hülle)

Eine Relation R auf X heißt Äquivalenzrelation, wenn sie reflexiv, symmetrisch und transitiv ist.

Für $a \in X$ ist $[a]_R$ (oder kurz $[a]$) die Äquivalenzklasse von a:

$$[a] = \{b \in X \mid (a, b) \in R\}.$$

Es gilt:

- $a \in [a]$ und
- $[a] \cap [b] \neq \emptyset \Rightarrow [a] = [b]$.

Beweis: Übung

Also teilt R die Menge X in Äquivalenzklassen ein:

$$X = \bigcup_{a \in X} [a]$$

Beispiele:

- Auf $\mathbb{N}$ wird für jedes $k \geq 1$ eine Äquivalenzrelation definiert durch

$$m \equiv_k n \;\Leftrightarrow\; k \text{ teilt } m - n.$$

- Auf $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$ wird eine Äquivalenzrelation definiert durch

$$(a, b) \sim (a', b') \;\Leftrightarrow\; a \cdot b' = a' \cdot b.$$

  So definiert man die rationalen Zahlen $\mathbb{Q}$.

- Auf der Menge der booleschen Formeln mit den freien Variablen $\{x_1, \ldots, x_k\}$ wird für jede Belegung $\sigma \colon \{x_1, \ldots, x_k\} \to \mathbb{B}$ eine Äquivalenzrelation definiert durch

$$F \sim^\sigma G \;\Leftrightarrow\; \sigma(F) = \sigma(G).$$

- Auch die Relation $F \sim G$ definiert durch $\forall \sigma : F \sim^\sigma G$ ist eine Äquivalenzrelation.

Definition

Wir definieren eine Äquivalenzrelation $\sim$ auf der Menge **Cmd**:

Seien $P, Q \in$ **Cmd**, **Loc** sei die Menge der in P oder Q vorkommenden Variablen. Sei $P \sim Q$ genau dann, wenn für alle $\sigma, \sigma' \in \mathbb{N}^{\mathbf{Loc}} = \Sigma$ gilt:

- P angesetzt auf Speicherzustand σ terminiert genau dann, wenn Q angesetzt auf σ terminiert.
- P angesetzt auf Speicherzustand σ erzeugt die Speicherzustand σ' genau dann, wenn Q angesetzt auf σ die Speicherzustand σ' erzeugt.

P und Q sind also äquivalent, wenn sie dieselben Funktionen $\Sigma \rightarrow_p \Sigma$ definieren.

Beispiele $c, c_1, c_2 \in$ **Cmd**, $b \in$ **Bexp**):

- $X := 3 \sim X := 2; X := X + 1$
- **skip**; **skip**; **skip** $\sim$ **skip**
- c; **skip** $\sim$ c
- **if** b **then** c_1 **else** c_2 **fi** $\sim$ **if** $\neg b$ **then** c_2 **else** c_1 **fi**
- **while** b **do** c **od** $\sim$ **if** b **then** c; **while** b **do** c **od else skip fi**

2.2 Partielle Ordnungen

Definition (partielle Ordnung, lineare Ordnung)

- Eine Relation $R \subseteq X \times X$ heißt partielle Ordnung auf X, falls R reflexiv, transitiv und antisymmetrisch ist.
- Eine partielle Ordnung $\leq$ ist total oder linear, falls für alle $x, y \in X$ gilt: $x \leq y$ oder $y \leq x$.

Um die Grundmenge X explizit anzugeben, sagen wir auch häufig, dass das Paar $(X, \leq)$ eine partielle (bzw. lineare) Ordunung ist.

Beispiele:

- $(\mathbb{N}, \leq)$ ist eine lineare Ordnung, denn es gilt für alle $n \in \mathbb{N} : n \leq n$. Außerdem ist $\leq$ transitiv und antisymmetrisch und je zwei natürliche Zahlen sind der Größe nach vergleichbar.
- Für $k \geq 2$ definiere auf $\mathbb{N}^k$ die Relation $\leq$ durch

$$(x_1, \ldots, x_k) \leq (y_1, \ldots, y_k) \;\Leftrightarrow\; \forall i \in \{1, \ldots, k\} : x_i \leq y_i.$$

  Dann ist $(\mathbb{N}^k, \leq)$ für $k \geq 2$ eine partielle Ordnung, aber keine lineare Ordnung! (Betrachte z.B. $(0, 1)$ und $(1, 0)$.)
- $(\mathbb{N}, |)$ mit $m \mid n$, falls m ein Teiler von n ist, ist eine partielle Ordnung, aber keine lineare Ordnung.

Definition (wohlfundiert partielle Ordnung, Wohlordnung)

Eine partielle Ordnung $(X, \leq)$ heißt wohlfundiert, wenn es in jeder nichtleeren Teilmenge von X ein minimales Element gibt, d.h. für jede nichtleere Teilmenge $Y \subseteq X$ gilt:

$$\exists m \in Y \;\; \forall y \in Y : \;\; y \leq m \Rightarrow y = m.$$

Ist $(X, \leq)$ eine wohlfundierte totale Ordnung, so nennen wir X wohlgeordnet, $(X, \leq)$ heißt dann eine Wohlordnung.

Beispiel: $(\mathbb{N}^k, \leq)$ ist wohlfundiert für alle $k \geq 1$.

Beweis:

1. Fall: $Y \neq \emptyset$ sei endlich.

Starte mit beliebigem Element $m \in Y$.

Suche $y \in Y$ mit $y < m$ (d.h. $y \leq m$ und $y \neq m$).

Existiert es nicht, sind wir fertig: m ist minimales Element.

Sonst setzen wir $m := y$ und beginnen wieder von vorn.

Da Y endlich ist, bricht die Kette nach endlich vielen solchen Schritten ab.

2. Fall: Y unendlich.

Starte mit beliebigem Element $m \in Y$.

Setze $Y' := \{y \in Y \mid y \leq m\}$. Dann ist Y' endlich.

Jedes minimale Element von Y' ist auch minimales Element von Y.

Nach dem 1. Fall existiert also ein minimales Element von Y.

Eine Verallgemeinerung dieses Beispiels ist der folgende Satz:

Satz 1

Eine partielle Ordnung $(X, \leq)$ ist wohlfundiert genau dann, wenn jede Kette

$$x_1 \geq x_2 \geq x_3 \geq \cdots$$

stationär wird, d.h. $\exists n_0 \, \forall n > n_0 : x_{n+1} = x_n$.

Beweis:

$\Rightarrow$: Sei $x_1 \geq x_2 \geq \cdots$ eine nicht stationär werdende Kette.

Dann hat die Menge $Y = \{x_i \mid i \geq 0\}$ kein minimales Element.

$\Leftarrow$: Sei $Y \neq \emptyset$ eine Menge ohne minimales Element.

Wähle $y_1 \in Y$ beliebig.

Wähle $y_2 \in Y$ mit $y_2 < y_1$.

Wähle $y_3 \in Y$ mit $y_3 < y_2$, etc.

Dann ist $y_1 > y_2 > y_3 > \cdots$ eine nicht stationär werdende Kette.

Definition (obere Schranke, Supremum, vollständiger Verband, Antikette)

Sei $(X, \leq)$ eine partielle Ordnung.

- $x \in X$ heißt obere Schranke für $Y \subseteq X$, falls gilt:

$$\forall y \in Y : y \leq x$$

- $x \in X$ heißt kleinste obere Schranke oder Supremum für $Y \subseteq X$, falls x obere Schranke für Y ist und für alle oberen Schranken x' für Y gilt: $x \leq x'$.

  Wir schreiben dann $x = \sup(Y) = \sqcup Y$.
- Ein vollständiger Verband ist eine partielle Ordnung $(X, \leq)$, in der für jede Teilmenge ein Supremum existiert.
- $Y \subseteq X$ heißt Antikette, falls für alle $y_1 \neq y_2 \in Y$ gilt: $y_1 \not\leq y_2$ und $y_2 \not\leq y_1$.

Beachte:

- Eine obere Schranke von Y muss nicht zu Y gehören, ebenso für das Supremum von Y. (Beispiel: $\sup(\{x \in \mathbb{R} \mid x < 1\}) = 1$).
- Wenn Y ein Supremum hat, dann ist dieses eindeutig bestimmt.

Beispiele:

- $(\mathbb{N}, \leq)$ ist eine Wohlordnung, aber kein vollständiger Verband.
- $(\mathbb{N} \cup \{\infty\}, \leq)$ ist eine Wohlordnung und ein vollständiger Verband.
- $(\mathbb{R} \cup \{-\infty, \infty\}, \leq)$ ist lineare Ordnung, vollständiger Verband, aber keine Wohlordnung.
- $(2^M, \subseteq)$ für beliebige Menge M ist ein vollständiger Verband.

  $(2^M, \subseteq)$ ist nicht linear, falls $|M| \geq 2$.

  $(2^M, \subseteq)$ ist nicht wohlfundiert, falls M unendlich.

Definition (Wohl-quasi-Ordnung)

Eine partielle Ordnung $(X, \leq)$ heißt Wohl-quasi-Ordnung (w.q.o.), wenn sie wohlfundiert ist und in X keine unendlichen Antiketten existieren.

Satz 2

$(\mathbb{N}^k, \leq)$ *ist w.q.o. für* $k \geq 1$.

Genauer: Sei $(x_i)_{i \in \mathbb{N}}$ *mit* $x_i \in \mathbb{N}^k$ *eine unendliche Folge* ($x_i = x_j$ *für* $i \neq j$ *ist möglich). Dann enthält* $(x_i)_{i \in \mathbb{N}}$ *eine unendliche monotone Teilfolge, d.h., eine Folge* $(x_{i_\ell})_{\ell \in \mathbb{N}}$ *mit* $i_1 < i_2 < i_3 < \cdots$ *und* $x_{i_1} \leq x_{i_2} \leq x_{i_3} \leq \cdots$.

Beweis:

Induktionsanfang: $k = 1$:

1. Fall: Es existieren nur endlich viele verschiedene Werte in $(x_i)_{i \in \mathbb{N}}$.

Dann kommt ein Wert unendlich oft vor, und wir erhalten so eine Teilfolge $(x_{i_\ell})_{\ell \in \mathbb{N}}$ mit $i_1 < i_2 < \cdots$ und $x_{i_1} = x_{i_2} = x_{i_3} \cdots$.

2. Fall: Es gibt unendlich viele verschiedene Werte in $(x_i)_{i \in \mathbb{N}}$.

Wähle x_{i_1} beliebig (z. B. $i_1 = 1$).

Es gibt ein $i_2 > i_1$ mit $x_{i_2} > x_{i_1}$.

Es gibt ein $i_3 > i_2$ mit $x_{i_3} > x_{i_2}$, u.s.w.

Induktionsschritt: Sei $k > 1$.

Sei $x_i = (a_i, y_i)$ mit $a_i \in \mathbb{N}$ und $y_i \in \mathbb{N}^{k-1}$.

Nach Induktionsvoraussetzung existiert eine monotone Teilfolge $(a_{i_\ell})_{\ell \in \mathbb{N}}$ von $(a_i)_{i \in \mathbb{N}}$.

Betrachte nun die Folge $(y_{i_\ell})_{\ell \in \mathbb{N}}$ in $\mathbb{N}^{k-1}$.

Nach Induktionsvoraussetzung besitzt $(y_{i_\ell})_{\ell \in \mathbb{N}}$ eine monotone Teilfolge.

Kombiniere deren Folgenglieder mit den entsprechenden ersten Komponenten aus $(a_{i_\ell})_{\ell \in \mathbb{N}}$.

Wir erhalten so eine monotone Teilfolge von $(x_i)_{i \in \mathbb{N}}$.

2.3 Strukturelle Induktion

Wir formulieren zuerst die allgemeine noethersche Induktion:

Satz 3

Sei $(X, \leq)$ eine wohlfundierte partielle Ordnung, $P\colon X \to \mathbb{B}$ eine Eigenschaft. Dann gilt $\forall x\, P(x)$ genau dann, wenn

$$\forall y\, [(\forall x < y : P(x)) \Rightarrow P(y)]$$

Beweis:

$\Rightarrow$: klar

$\Leftarrow$: Sei $Y = \{y \in X \mid \neg P(y)\} \neq \emptyset$.

Dann existiert ein minimales Element $y \in Y$.

Wegen $y \in Y$ haben wir $\neg P(y)$, aber $\forall x < y : P(x)$.

Also ist $\forall y\, [(\forall x < y : P(x)) \Rightarrow P(y)]$ falsch.

Die wichtigste partielle Ordnung wird für uns die Teiltermrelation auf der Menge der Ausdrücke und Anweisungen von **IMP** sein.

Wir formalisieren das wie folgt:

Definition (Signatur)

Eine Signatur ist gegeben durch ein Tripel (S, Ω, τ). Dabei ist

- S eine Menge von Sorten,
- Ω eine Menge von Funktionssymbolen, und
- $\tau : \Omega \to S^+$.

Für $F \in \Omega$ und $\tau(F) = s_1 s_2 \dots s_k s$ heißt F eine k-stelliges Funktionssymbol mit Eingangssorten $s_1, \dots, s_k$ und Zielsorte s.

Falls $\tau(F) = s \in S$, nennen wir F eine Konstante.

Beispiele: (aus **IMP**):

- $\tau(\textbf{skip}) = \textbf{Cmd}$
- $\tau(\ ;\ ) = \textbf{Cmd Cmd Cmd}$
- $\tau(\textbf{if-then-else-fi}) = \textbf{Bexp Cmd Cmd Cmd}$
- $\tau(\textbf{while-do-od}) = \textbf{Bexp Cmd Cmd}$

Definition (Terme)

Sei (S, Ω, τ) eine Signatur, und sei für alle $s \in S$ eine Menge von Variablen V_s definiert. Die Menge der Terme über (S, Ω, τ) ist definiert wie folgt:

a) $\forall X \in V_s : X$ ist Term der Sorte s.

b) Sei $F \in \Omega, \tau(F) = s_1 s_2 \ldots s_k s$ und t_i Term der Sorte s_i $(i \in \{1, \ldots, k\})$. Dann ist $F(t_1, \ldots, t_k)$ ein Term der Sorte s.

c) Sonst gibt es keine Terme.

Definition (Teilterme)

Für $t = F(t_1, \ldots, t_n)$ nennen wir $t_1, \ldots, t_n$ direkte Teilterme von t, d.h. $(t_i, t) \in R$ $(i \in \{1, \ldots, n\})$ für die sogenannte direkte Teiltermrelation R.

Sei $\leq$ die reflexive transitive Hülle von R, d.h. $t \leq t'$ genau dann, wenn $t = t'$ oder wenn $t_0, t_1, \ldots, t_r$ existieren mit $r \geq 1$, $t_0 = t$, $t_r = t'$ und $(t_0, t_1), (t_1, t_2), \ldots, (t_{r-1}, t_r) \in R$.

Dann ist $\leq$ eine wohlfundierte partielle Ordnung.

Beweis: Übung

Noethersche Induktion kann also auf die partielle Ordnung $(\mathbf{IMP}, \leq)$ angewendet werden, um Aussagen über alle Anweisungen und Ausdrücke aus **IMP** zu erhalten.

Diesen Spezialfall der noetherschen Induktion nennen wir strukturelle Induktion.

Wir betrachten noch eine weitere Art der Induktion, die sogenannte Regelinduktion.

Hierbei ist ein festes Universum $\mathbb{U}$ gegeben, weiterhin eine Regelmenge R, deren Elemente die Form $(\{x_1, \ldots, x_n\}/y)$ haben.

Diese Regel sagt aus: Falls $x_1, \ldots, x_n$ gelten, gilt auch y.

Ist $n = 0$, die Regel also von der Form $(\emptyset/y)$, so nennen wir y ein Axiom.

Wir schreiben auch

$$\frac{}{y} \quad \text{bzw.} \quad \frac{x_1, \ldots, x_n}{y}$$

für $(\emptyset/y) \in R$ bzw. $(\{x_1, \ldots, x_n\}/y) \in R$.

Ein Beweis (oder eine Ableitung) für ein $y \in \mathbb{U}$ ist entweder eine Regel $(\emptyset / y) \in R$ oder ein Objekt der Form

$$(\{d_1, \ldots, d_n\}/y)$$

wobei $(\{x_1, \ldots, x_n\}/y) \in R$ gilt und für $i \in \{1, \ldots, n\}$ d_i ein Beweis für $x_i \in \mathbb{U}$ ist.

Wir schreiben $\vdash_R y$, wenn es für y eine Ableitung im Regelsystem R gibt.

Ein Beweis besteht also aus einer in Baumform zusammengesetzten Ansammlung von Regeln, etwa der Form:

$$\dfrac{\dfrac{\overline{x_1}\quad \overline{x_2}\cdots\overline{x_k}}{y_1}\quad \dfrac{\overline{z_1}\quad \overline{z_2}\cdots\overline{z_l}}{y_2}\quad \dfrac{\overline{w_1}\quad \overline{w_2}\cdots\overline{w_m}}{y_3}}{\text{ZIEL}}$$

Diese Bäume können eine beliebige endliche Tiefe haben.

Wie bei der Teiltermrelation können wir auch auf der Menge der Beweise eine wohlfundierte partielle Ordnung $\preceq$ einführen durch die Festlegung $d' \preceq d$, falls d' im Beweis d als Teilbeweis vorkommt.

Diese partielle Ordnung läßt sich zu einer Wohlordnung verfeinern, etwa indem man die gesamte Menge der Beweise längenlexikographisch (nach einer geeigneten Kodierung) anordnet.

Dann existiert zu jedem $y \in \mathbb{U}$ mit $\Vdash_R y$ ein minimaler Beweis $d(y)$.

Sei $I_R = \{y \in \mathbb{U} \mid \Vdash_R y\}$ die Menge aller ableitbaren Elemente des Universums.

Satz 4

Sei $P : I_R \to \{\textbf{true}, \textbf{false}\}$. *Dann gilt* $\forall y \in I_R :$ $P(y)$ *genau dann, wenn*

$$\forall(\{x_1, \ldots, x_n\}/y) \in R : \quad [(\forall i \in \{1, \ldots, n\}\ P(x_i)) \to P(y)] \qquad (\dagger)$$

Das aus Satz 4 folgende Induktionsprinzip nennen wir Regelinduktion.

Beweis: Definiere die Ordnung auf I_R entsprechend der Wohlordnung auf den minimalen Beweisen: $x \leq y$ falls $x = y$ oder $d(x) \prec d(y)$.

Dann ist $\leq$ eine Wohlordnung auf I_R.

$\Rightarrow$: Gelte $P(y)$ für alle $y \in I_R$, sei $(\{x_1, \ldots, x_n\}/y) \in R$ und gelte $\forall i \in \{1, \ldots, n\} : P(x_i)$.

Dann gilt insbesondere $x_i \in I_R$ (da P nur auf I_R definiert ist) und damit auch $y \in I_R$, also $P(y)$.

$\Leftarrow$: Annahme: $Y = \{y \in I_R \mid \neg P(y)\} \neq \emptyset$.

Sei $y \in Y$ minimal und sei $d(y) = (\{d_1, \ldots, d_n\}/y)$ der minimale Beweis für $y \in Y$.

Also gibt es eine Regel $(\{x_1, \ldots, x_n\}/y) \in R$, so dass d_i ein Beweis für x_i ist und $d_i \prec d(y)$.

Also gilt $d(x_i) \preceq d_i \prec d(y)$ und damit $x_i < y$ für alle $i \in \{1, \ldots, n\}$.

Da y minimal in Y war, gilt $x_i \notin Y$, d. h. $P(x_i)$.

Also ist (†) falsch.

2.4 Vollständige Halbordnungen

Definition (gerichtete Menge)

Sei $(X, \leq)$ eine partielle Ordnung. Eine Teilmenge $D \subseteq X$ ist gerichtet, wenn für alle $x, y \in D$ ein $z \in D$ existiert mit

$$x \leq z \quad \text{und} \quad y \leq z.$$

Definition (vollständige Halbordnung)

Eine partielle Ordnung heißt vollständig, wenn jede gerichtete Teilmenge eine kleinste obere Schranke (nicht notwendig in der Teilmenge) besitzt.

Eine vollständige partielle Ordnung nennen wir vollständige Halbordnung oder CPO (complete partial order).

Da die leere Menge offenbar gerichtet ist, muss sie in einer CPO ein Supremum haben, dieses muss das kleinste Element $\bot$ in der CPO sein.

Beispiele:

- Für endliche Menge X ist $(X, \leq)$ eine CPO genau dann, wenn X ein kleinstes Element hat.
- $(\mathbb{N}, \leq)$ ist keine CPO.
- $(\mathbb{N} \cup \{\infty\}, \leq)$ ist eine CPO ($\bot = 0$).
- $\mathbb{B}$ mit **true** $<$ **false** ist eine CPO.
- Für jede Menge Z bildet $(2^Z, \subseteq)$ eine CPO. Das kleinste Element ist $\bot = \emptyset$, allgemein ist $\sqcup D = \bigcup \{y \mid y \in D\}$.

- $((X \rightarrow_p Y), \sqsubseteq)$ ist eine CPO, wenn die partielle Ordnung $\sqsubseteq$ wie folgt definiert wird: $f \sqsubseteq g \Leftrightarrow \text{graph}(f) \subseteq \text{graph}(g)$.

  Es gilt nämlich für $\sqcup D = f$ für $D = \{f_i \mid i \in I\}$, wobei mit $f(x) = y$, falls ein i existiert, so dass $f_i(x) = y$, sonst ist $f(x)$ undefiniert.

  (Zeige, dass f wohldefiniert ist!)
- $(\Gamma^*, \leq)$, wobei $\Gamma = \{a, b\}$ und $\leq$ die Präfixordnung ist, ist keine CPO.
- $(\Gamma^\infty, \leq)$, wobei $\Gamma^\infty = \Gamma^* \cup \Gamma^\omega$ ist, ist eine CPO.
- $(\Sigma, \leq)$ mit $\Sigma = \mathbb{N}^{\mathbf{Loc}}$ ist keine CPO, wenn man $\leq$ komponentenweise definiert.
- $((\mathbb{N} \cup \{\infty\})^{\mathbf{Loc}}, \leq)$ ist eine CPO, wenn man $\leq$ komponentenweise definiert.
- $((\Sigma \cup \{\infty\}), \leq)$ mit $\Sigma = \mathbb{N}^{\mathbf{Loc}}$ ist eine CPO, wenn man $\leq$ komponentenweise definiert.

Definition (Fixpunkt)

Für eine Abbildung $f : X \to X$ nennen wir x einen Fixpunkt, wenn $f(x) = x$ gilt.

Definition (monotone und stetige Abbildung)

Seien $(X, \leq)$ und $(Y, \leq)$ zwei CPOs. (Die Ordnungsrelationen tragen hier zwar die gleiche Bezeichnung, sind aber im allgemeinen verschieden!)

Eine Abbildung $f : X \to Y$ heißt monoton, wenn aus $x \leq x'$ folgt: $f(x) \leq f(x')$.

Die Abbildung f heißt stetig, wenn sie monoton ist und für jede gerichtete nicht leere Teilmenge $D \subseteq X$ gilt: $\sqcup f(D) = f(\sqcup D)$.

Lemma 5

Ist $f : X \to Y$ monoton und D eine gerichtete Teilmenge von X, dann ist auch $f(D)$ gerichtet. Ferner sind sowohl $\sqcup f(D)$ als auch $f(\sqcup D)$ definiert, wenn X und Y CPOs bilden und D gerichtet ist.

Beweis: Seien $x, y \in f(D)$.

Also existieren $a, b \in D$ mit $x = f(a)$ und $y = f(b)$.

Da D gerichtet ist, gibt es ein $c \in D$ mit $a \leq c$ und $b \leq c$.

Da f monoton ist, gilt $x = f(a) \leq f(c) \in f(D)$ und
$y = f(b) \leq f(c) \in f(D)$.

Also ist $f(D) \subseteq Y$ gerichtet.

Wir beweisen nun den zentralen Fixpunktsatz von Kleene:

Satz 6

Sei $(X, \leq)$ eine CPO, und sei $f : X \to X$ eine stetige Funktion. Dann gibt es immer einen eindeutig bestimmten kleinsten Fixpunkt $fix(f)$ für f.

Es gilt also $f(fix(f)) = fix(f)$ und $fix(f) \leq y$ für jeden Fixpunkt y von f.

Beweis: Sei $\bot$ das kleinste Element in X.

Dann gilt $\bot \leq f(\bot)$, daher (wegen der Monotonie von f) auch

$$f(\bot) \leq f(f(\bot))$$

und weiter (mit Induktion) für alle $i \geq 0$:

$$f^i(\bot) \leq f^{i+1}(\bot)$$

Wir erhalten also eine Kette $\bot \leq f(\bot) \leq f^2(\bot) \leq f^3(\bot) \leq \cdots$

Setzen wir $x_f := \sqcup\{f^i(\bot) \mid i \geq 0\}$, so erhalten wir

$$\begin{aligned} f(x_f) &= f(\sqcup\{f^i(\bot) \mid i \geq 0\}) \\ &= \sqcup\{f(f^i(\bot)) \mid i \geq 0\} \\ &= \sqcup\{f^i(\bot) \mid i \geq 1\} \\ &= x_f \end{aligned}$$

Noch zu zeigen: $x_f \leq y$, falls y Fixpunkt von f ist.

Sei also $f(y) = y$.

Dann folgt $\forall i \geq 0 : f^i(\bot) \leq f^i(y) = y$.

Also ist y eine obere Schranke von $\{f^i(\bot) \mid i \geq 0\}$.

Da x_f das Supremum von $\{f^i(\bot) \mid i \geq 0\}$ ist, erhalten wir $x_f \leq y$.

2.5 Flache Bereiche

Als flachen Bereich wollen wir eine Menge bezeichnen, der wir ein minimales Element künstlich beifügen:

Definition (flache Bereiche)

Eine partielle Ordnung $(X, \leq)$ ist ein flacher Bereich, wenn ein Element $\bot$ existiert mit $\forall x \in X : \ \bot \ \leq x$ und $X \setminus \{\bot\}$ eine Antikette ist.

Offensichtlich ist jeder flache Bereich eine CPO.

Eine beliebige Menge machen wir wie folgt zum flachen Bereich:

Sei X eine Menge, $\bot \notin X$.

Wir definieren $X_\bot = X \cup \{\bot\}$ und machen $X_\bot$ zur Halbordnung durch die Definition $x \leq y$ genau dann, wenn $x = y$ oder $x = \bot$.

Betrachte die folgenden drei Mengen:

- $X \rightarrow_p Y$: Alle partiellen Funktionen von X nach Y.
- $X \rightarrow Y_\perp$: Alle Funktionen von X nach $Y_\perp$.
- $X_\perp \rightarrow Y_\perp$, wobei wir nur stetige Abbildungen, die das Minimum $\perp \in X_\perp$ auf $\perp \in Y_\perp$ abbilden, nehmen.

Offenbar gibt es bijektive Entsprechungen zwischen diesen Mengen:

$f : X \rightarrow_p Y$ wird ergänzt durch $f(x) = \perp$, falls nicht definiert.

$f : X \rightarrow Y_\perp$ wird ergänzt durch $f(\perp) = \perp$. Man sieht leicht, dass diese Abbildung stetig ist.

Eine stetige Abbildung $f : X_\perp \rightarrow Y_\perp$ mit $f(\perp) = \perp$ wird zur partiellen Funktion von X nach Y durch Einschränkung des Definitionsbereichs auf das Urbild $f^{-1}(Y)$ von Y.

Für zwei CPOs $(X, \leq)$ und $(Y, \leq)$ definieren wir

$$[X \to Y] = \{f : X \to Y \mid f \text{ ist stetig}\} \subseteq Y^X.$$

Auf $[X \to Y]$ kann man eine Ordnungsrelation $\sqsubseteq$ definieren durch

$$f \sqsubseteq g \;\Leftrightarrow\; \forall x \in X : \; f(x) \leq g(x).$$

Diese Relation ist nämlich offenbar transitiv, reflexiv und antisymmetrisch: Alle drei Eigenschaften übertragen sich direkt von den entsprechenden Eigenschaften der Relation $\leq$ auf $\sqsubseteq$.

Es gilt dann:

Satz 7

Für CPOs $(X, \leq)$ und $(Y, \leq)$ bildet auch $([X \to Y], \sqsubseteq)$ eine CPO.

Beweis: Wir müssen zeigen, dass für jede gerichtete Teilmenge von $[X \to Y]$ ein Supremum existiert.

Sei $D = \{f_i \mid i \in I\} \subseteq [X \to Y]$ gerichtet.

Für jedes $x \in X$ definieren wir $D_x = \{f_i(x) \mid i \in I\} \subseteq Y$.

D_x ist eine gerichtete Menge von Y, hat also ein Supremum $d_x = \sqcup D_x$.

Wir definieren eine Funktion $f : X \to Y$ durch $f(x) = d_x$.

Behauptung: $f \in [X \to Y]$.

Um diese Behauptung zu beweisen, müssen wir zeigen:

- f ist monoton.
- Für jede nicht leere gerichtete Menge $Z \subseteq X$ gilt $f(\sqcup Z) = \sqcup f(Z)$.

Nehmen wir für den Augenblick an, wir hätten diese beiden Eigenschaften von f gezeigt.

Dann gilt:

- f ist obere Schranke von D, da $f_i \sqsubseteq f$ für alle $i \in I$.
- Für jede obere Schranke g von D gilt $f \sqsubseteq g$.

Also gilt $f = \sqcup D$.

Wir müssen also noch die folgenden beiden Behauptungen zeigen.

Behauptung 1. f ist monoton.

Seien $x \leq y$ zwei Elemente von X und $i \in I$.

Dann gilt $f_i(x) \leq f_i(y) \leq f(y)$.

Also ist $f(y)$ obere Schranke für D_x und damit $f(x) = d_x = \sqcup D_x \leq f(y)$.

Behauptung 2. $f(\sqcup Z) = \sqcup f(Z)$ für jedes nicht leere gerichtete $Z \subseteq X$.

Wir setzen $z = \sqcup Z$.

Es gilt $f(z) = \sqcup D_z = \sqcup\{f_i(z) \mid i \in I\}$.

Aus der Stetigkeit von f_i folgt aber auch $f_i(z) = f_i(\sqcup Z) = \sqcup f_i(Z)$.

Es folgt

$$\begin{aligned} f(\sqcup Z) &= f(z) \\ &= \sqcup\{\sqcup f_i(Z) \mid i \in I\} \\ &= \sqcup\{f_i(x) \mid x \in Z,\ i \in I\} \\ &= \sqcup\{\sqcup\{f_i(x) \mid i \in I\} \mid x \in Z\} \\ &= \sqcup\{f(x) \mid x \in Z\} \\ &= \sqcup f(Z). \end{aligned}$$

Wir hatten schon einmal eine Relation $\sqsubseteq$ auf Funktionenmengen definiert, nämlich:

Für partielle Funktionen $f, g : X \rightarrow_p Y$ sei:

$$f \sqsubseteq g \;\Leftrightarrow\; \text{graph}(f) \subseteq \text{graph}(g)$$

Bilden wir zu den Mengen X und Y die flachen Bereiche $X_\bot$ und $Y_\bot$, dann sind f und g, ergänzt durch $f(\bot) = g(\bot) = \bot$, stetige Funktionen, d.h. $f, g \in [X_\bot \rightarrow Y_\bot]$.

Es gilt daher $\text{graph}(f) \subseteq \text{graph}(g)$ genau dann, wenn

$$\forall x : (f(x) \neq \bot \;\Rightarrow\; f(x) = g(x))$$

Das ist aber wiederum äquivalent zu

$$\forall x : f(x) \leq g(x),$$

denn wenn $f(x) = \bot$ ist, dann ist es kleiner oder gleich jedem Element.

Nun bemerken wir noch, dass das Supremum f_D einer gerichteten Menge $D \subseteq (X \ \to_p \ Y)$ nicht abweicht vom Supremum der Menge D als Teilmenge von $[X_\bot \to Y_\bot]$, nur der Wert $f_D(\bot) = \bot$ muss ergänzt werden.

Also sehen wir, dass nach Satz 7 auch $((X \ \to_p \ Y), \sqsubseteq)$ eine CPO ist.

3 Strukturelle operationale Semantik

3.1 Transitionssysteme

- Ein Transitionssystem ist gegeben durch einen gerichteten (nicht notwendig endlichen) Graph. Die Knoten eines Transitionssystems nennen wir Zustände, die Kanten Transitionen.
- Gibt es in einem Transitionssystem mit Knotenmenge Q und Kanten gegeben durch die zweistellige Relation $\rightarrow$ einen ausgezeichneten Startzustand q_0, so sprechen wir von dem initialen Transitionssystem $(Q, \rightarrow, q_0)$.

- Ist im initialen Transitionssystem $(Q, \rightarrow, q_0)$ jeder Zustand q auf einem gerichteten Weg von q_0 aus erreichbar, so heißt das System initial zusammenhängend.
- Endzustände des Systems sind Knoten mit Ausgangsgrad 0.
- Ein System heißt deterministisch, wenn für alle $u, v_1, v_2 \in Q$ aus $u \rightarrow v_1$ und $u \rightarrow v_2$ folgt $v_1 = v_2$.
- Es sei ein deterministisches Transitionssystem mit Knotenmenge Q und Kantenmenge $\rightarrow$ gegeben.

  Seien $X, Y \subseteq Q$ gegeben, so dass alle $y \in Y$ Ausgangsgrad 0 haben.

  Dann erhalten wir eine partielle Abbildung $f : X \rightarrow_p Y$ durch die Festlegung $f(x) = y$, falls es einen gerichteten Weg von x nach y gibt, $f(x)$ undefiniert, sonst.

  (Man überprüfe, dass diese Abbildung wohldefiniert ist, d.h. dass es maximal einen Knoten $y \in Y$ geben kann, der von x aus erreichbar ist.)

3.2 Einzelschrittsemantik

Bei der Definition der Mengen **Aexp** und **Bexp** hatten wir schon eine Auswertungsstrategie angegeben, d.h. eine Abbildung

$$\mathcal{A}\colon \mathbf{Aexp} \to (\Sigma \to \mathbb{N}).$$

Zur Erinnerung: Wir schreiben $\mathcal{A}[\![a]\!]\sigma = n$, falls der Ausdruck a unter der Belegung σ zu n ausgewertet wird.

Entsprechend definieren wir $\mathcal{B}\colon \mathbf{Bexp} \to (\Sigma \to \mathbb{B})$.

Bei der Einzelschritt-Semantik gehen wir davon aus, dass für alle arithmetischen oder booleschen Ausdrücke diese Abbildung in einem Schritt ausgeführt werden kann.

Dagegen müssen wir die Ausführung der Anweisungen (d.h. der Elemente von **Cmd**) durch Regeln definieren.

Wir benutzen ein Transitionssystem mit der Knotenmenge

$$(\mathbf{Cmd} \times \Sigma) \cup \Sigma.$$

Die Kanten werden wir so bilden, dass ein deterministisches Transitionssystem entsteht, in dem alle Knoten aus $(\mathbf{Cmd} \times \Sigma)$ Ausgangsgrad 1 haben, alle Knoten aus Σ dagegen Ausgangsgrad 0.

Es wird also durch die oben definierte partielle Abbildung f, die jedem Knoten das eindeutig bestimmte Ende eines bei diesem Knoten startenden Weges zuordnet (bzw. undefiniert, falls der Weg nicht endet), eine partielle Funktion von $(\mathbf{Cmd} \times \Sigma)$ nach Σ definiert.

Ordnet diese Funktion dem Paar (c, σ) das Element σ' zu, so schreiben wir

$$\mathcal{C}[\![c]\!]\sigma = \sigma' \text{ oder } \mathcal{C}(c, \sigma) = \sigma'.$$

Die so definierte Funktion $\mathcal{C}$ nennen wir auch $\mathcal{C}_{\text{sos}}$.

Wir definieren nun induktiv die Kantenmenge, die wir mit $\rightarrow_1$ bezeichnen, da es sich um die Einzelschrittsemantik handelt:

skip-Regel:

$$\frac{}{(\textbf{skip},\sigma) \rightarrow_1 \sigma}$$

Zuweisungs-Regel:

$$\frac{\mathcal{A}[\![a]\!]\sigma = m}{(X := a,\sigma) \rightarrow_1 \sigma[m/X]}$$

(Dabei ist $\sigma[m/X](X) = m$ und für alle $Y \neq X$ gilt $\sigma[m/X](Y) = \sigma(Y)$.)

Hintereinanderausführungsregeln:

$$\frac{(c_1,\sigma) \rightarrow_1 \sigma'}{(c_1;c_2,\sigma) \rightarrow_1 (c_2,\sigma')}$$

$$\frac{(c_1,\sigma) \rightarrow_1 (c_1',\sigma')}{(c_1;c_2,\sigma) \rightarrow_1 (c_1';c_2,\sigma')}$$

if-then-else-Regeln:

$$\frac{\mathcal{B}[\![b]\!]\sigma = \textbf{true}}{(\textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2\ \textbf{fi}, \sigma) \rightarrow_1 (c_1, \sigma)}$$

$$\frac{\mathcal{B}[\![b]\!]\sigma = \textbf{false}}{(\textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2\ \textbf{fi}, \sigma) \rightarrow_1 (c_2, \sigma)}$$

while-Regeln:

$$\frac{\mathcal{B}[\![b]\!]\sigma = \textbf{true}}{(\textbf{while}\ b\ \textbf{do}\ c_1\ \textbf{od}, \sigma) \rightarrow_1 (c_1; \textbf{while}\ b\ \textbf{do}\ c_1\ \textbf{od}, \sigma)}$$

$$\frac{\mathcal{B}[\![b]\!]\sigma = \textbf{false}}{(\textbf{while}\ b\ \textbf{do}\ c_1\ \textbf{od}, \sigma) \rightarrow_1 (\textbf{skip}, \sigma)}$$

Wir müssen nun zeigen:

1. Jedes $\sigma \in \Sigma$ hat als Knoten des Transitionssystems $(\Sigma \cup \mathbf{Cmd} \times \Sigma, \rightarrow_1)$ Ausgangsgrad 0.
2. Jedes $(c, \sigma) \in \mathbf{Cmd} \times \Sigma$ hat als Knoten des Transitionssystems $(\Sigma \cup \mathbf{Cmd} \times \Sigma, \rightarrow_1)$ Ausgangsgrad 1.

Daraus folgt dann, dass das Transitionssystem $(\Sigma \cup \mathbf{Cmd} \times \Sigma, \rightarrow_1)$ deterministisch ist und die Abbildung $\mathcal{C}_{\text{sos}}$ eine wohldefinierte partielle Funktion vom Typ $\mathbf{Cmd} \times \Sigma \ \rightarrow_p \ \Sigma$ darstellt:

$$\mathcal{C}_{\text{sos}}(c, \sigma) = \sigma' \ :\Longleftrightarrow \ (c, \sigma) \rightarrow_1^* \sigma'$$

Beweis, dass im Transitionssystem für die Einzelschrittsemantik alle $\sigma \in \Sigma$ Ausgangsgrad 0 haben, alle anderen Knoten Ausgangsgrad 1:

Zu $\sigma \in \Sigma$: Die Regeln definieren keine Transitionen der Form $\sigma \rightarrow_1 \cdots$.

Also hat jeder Knoten $\sigma \in \Sigma$ Ausgangsgrad 0.

Zu $(c, \sigma) \in \textbf{Cmd} \times \Sigma$:

1. Fall: $c = \textbf{skip}$: Die einzige Regel, die eine Transition

$$(\textbf{skip}, \sigma) \rightarrow_1 \cdots$$

erzeugt, ist die **skip**-Regel

$$\frac{}{(\textbf{skip}, \sigma) \rightarrow_1 \sigma}$$

Also ist $(\textbf{skip}, \sigma) \rightarrow_1 \sigma$ die einzige von $(\textbf{skip}, \sigma)$ ausgehende Transition, d.h. für alle $\sigma \in \Sigma$ hat $(\textbf{skip}, \sigma)$ Ausgangsgrad 1.

2. Fall: $c = (X := a)$: Die einzige Regel, die eine Transition

$$(X := a, \sigma) \rightarrow_1 \cdots$$

erzeugt, ist die Zuweisungsregel

$$\frac{\mathcal{A}[\![a]\!]\sigma = m}{(X := a, \sigma) \ \rightarrow_1 \ \sigma[m/X]}$$

Also ist $(X := a, \sigma) \rightarrow_1 \sigma[m/X]$ die einzige von $(X := a, \sigma)$ ausgehende Transition, d.h. für alle $\sigma \in \Sigma$ hat $(X := a, \sigma)$ Ausgangsgrad 1.

3. Fall: $c = c_1; c_2$

Wir nehmen induktiv an, dass der Ausgangsgrad von (c_i, τ) gleich 1 ist für alle $\tau \in \Sigma$ und $i \in \{1, 2\}$.

Es existieren zwei Regeltypen, die Regeln der Form $(c_1; c_2, \sigma) \rightarrow_1 \cdots$ erzeugen können.

Der Ausgangsgrad von (c_1, σ) ist 1, d.h. es gibt genau ein σ' oder ein (c_1', σ') mit $(c_1, \sigma) \rightarrow_1 \sigma'$ bzw. $(c_1, \sigma) \rightarrow_1 (c_1', \sigma')$.

Ist $(c_1, \sigma) \rightarrow_1 \sigma'$ die einzige von (c_1, σ) ausgehende Transition, so ist $(c_1; c_2, \sigma) \rightarrow_1 (c_2, \sigma')$ die einzige von $(c_1; c_2, \sigma)$ ausgehende Transition.

Ist $(c_1, \sigma) \rightarrow_1 (c_1', \sigma')$ die einzige von (c_1, σ) ausgehende Transition, so ist $(c_1; c_2, \sigma) \rightarrow_1 (c_1'; c_2, \sigma')$ die einzige von $(c_1; c_2, \sigma)$ ausgehende Transition.

Also ist für alle $c_1, c_2 \in \mathbf{Cmd}$ und alle $\sigma \in \Sigma$ der Ausgangsgrad von $(c_1; c_2, \sigma)$ gleich 1.

4. Fall: $c =$ **if** b **then** c_1 **else** c_2 **fi**:

Es gibt genau zwei Regeltypen, die Transitionen der Form $(c, \sigma) \to_1 \cdots$ definieren.

Da $\mathcal{B}[\![b]\!]\sigma =$ **true** oder $\mathcal{B}[\![b]\!]\sigma =$ **false** nur von b und σ abhängt, kann genau eine der Regelformen Anwendung finden.

Also existiert entweder die Transition $(c, \sigma) \to_1 (c_1, \sigma)$ oder $(c, \sigma) \to_1 (c_2, \sigma)$, d.h. für alle $\sigma \in \Sigma$ ist der Ausgangsgrad von (c, σ) gleich 1.

5. Fall: $c =$ **while** b **do** c **od**:

Analog zum Fall der **if**-Anweisungen können wir schließen, dass genau eine Transition der Form $(c, \sigma) \to_1 \cdots$ existiert.

Also ist der Ausgangsgrad von (c, σ) gleich 1 für alle **while**-Anweisungen c und alle $\sigma \in \Sigma$.

Damit ist gezeigt, dass die SOS-Semantik die behauptete Eigenschaft hat.

Als Anwendung der SOS-Einzelschrittsemantik wollen wir zeigen, dass in der Sprache IMP die Anweisungen $c_1; (c_2; c_3)$ und $(c_1; c_2); c_3$ äquivalent sind. („Assoziativität der Hintereinanderausführung“)

Wir müssen also zeigen:

$$\mathcal{C}_{\text{sos}}[\![c_1; (c_2; c_3)]\!] = \mathcal{C}_{\text{sos}}[\![(c_1; c_2); c_3]\!]$$

Das ist äquivalent mit $\mathcal{C}_{\text{sos}}[\![c]\!]\sigma = \mathcal{C}_{\text{sos}}[\![c']\!]\sigma$ für alle $\sigma \in \Sigma$, wobei wir $c = c_1; (c_2; c_3)$ und $c' = (c_1; c_2); c_3$ setzen.

Ebenso könnten wir sagen, dass $\mathcal{C}_{\text{sos}}(c, \sigma) = \mathcal{C}_{\text{sos}}(c', \sigma)$ für alle $\sigma \in \Sigma$ zu zeigen ist.

Wir definieren

$$\begin{aligned}
\sigma' &= \mathcal{C}_{\text{sos}}(c_1, \sigma) && \text{(falls das existiert)} \\
\sigma'' &= \mathcal{C}_{\text{sos}}(c_2, \sigma') && \text{(falls } \sigma' \text{ def. und ex.)} \\
\sigma''' &= \mathcal{C}_{\text{sos}}(c_3, \sigma'') && \text{(falls } \sigma'' \text{ def. und ex.)}
\end{aligned}$$

Es gibt 4 Fälle, je nachdem, welche der Speicherzustände $\sigma', \sigma'', \sigma'''$ definiert sind.

Wir bearbeiten nur den Fall, dass $\sigma', \sigma'', \sigma'''$ alle definiert sind. Die übrigen führen in ähnlicher Weise jeweils zu $\mathcal{C}_{\text{sos}}(c, \sigma) = \mathcal{C}_{\text{sos}}(c', \sigma) =$ undefiniert.

Wir zeigen $\mathcal{C}_{\text{sos}}(c, \sigma) = \sigma''' = \mathcal{C}_{\text{sos}}(c', \sigma)$.

Allgemein gilt nach der Definition der Funktion $\mathcal{C}_{\text{sos}}$ für alle $x \in \textbf{Cmd}$ und alle $\tau, \tau' \in \Sigma$:

$$\mathcal{C}_{\text{sos}}(x, \tau) = \tau' \iff (x, \tau) \to_1^* \tau'.$$

Auf Grund der Hintereinanderausführungsregeln können wir (durch mehrmaliges Anwenden) für alle $c_1, d \in\in \textbf{Cmd}$ und alle $\sigma, \sigma' \in \Sigma$ schließen:

$$(c_1, \sigma) \to_1^* \sigma' \implies (c_1; d, \sigma) \to_1^* (d, \sigma').$$

Insbesondere (durch Einsetzen der entsprechenden Werte für d etc.):

$$(c_1; (c_2; c_3), \sigma) \to_1^* (c_2; c_3, \sigma') \to_1^* (c_3, \sigma'') \to_1^* \sigma'''.$$

Andererseits aber auch $(c_1; c_2, \sigma) \to_1^* (c_2, \sigma') \to_1^* \sigma''$, d.h.

$$((c_1; c_2); c_3, \sigma) \to_1^* (c_3, \sigma'') \to_1^* \sigma'''.$$

Damit ist die Behauptung gezeigt.

3.3 Ergebnissemantik

Die Ergebnissemantik (= Bigstep-Semantik) führt die Auswertung einer Anweisung nicht schrittweise durch, sondern ordnet direkt jedem Paar (c, σ) eine neue Belegung σ' zu (falls möglich).

Wir benutzen wieder ein Transitionssystem, die Knotenmenge ist dieselbe wie bei der Einzelschrittsemantik, d.h. $(\mathbf{Cmd} \times \Sigma) \cup \Sigma$.

Die Kantenmenge wird neu definiert, diesmal wird sie mit $\rightarrow$ bezeichnet:

skip-Regel:

$$\frac{}{(\mathbf{skip},\sigma) \rightarrow \sigma}$$

Zuweisungs-Regel:

$$\frac{\mathcal{A}[\![a]\!]\sigma = m}{(X := a, \sigma) \rightarrow \sigma[m/X]}$$

Hintereinanderausführungs-Regel:

$$\frac{(c_1,\sigma) \rightarrow \sigma' \qquad (c_2,\sigma') \rightarrow \sigma''}{(c_1;c_2,\sigma) \rightarrow \sigma''}$$

if-then-else-Regeln:

$$\frac{\mathcal{B}[\![b]\!]\sigma = \textbf{true} \qquad (c_1,\sigma) \rightarrow \sigma'}{(\textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ fi}, \sigma) \rightarrow \sigma'}$$

$$\frac{\mathcal{B}[\![b]\!]\sigma = \textbf{false} \qquad (c_2,\sigma) \rightarrow \sigma'}{(\textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ fi}, \sigma) \rightarrow \sigma'}$$

while-Regeln:

$$\frac{\mathcal{B}[\![b]\!]\sigma = \textbf{true} \quad (c_1,\sigma) \rightarrow \sigma' \quad (\textbf{while } b \textbf{ do } c_1 \textbf{ od}, \sigma') \rightarrow \sigma''}{(\textbf{while } b \textbf{ do } c_1 \textbf{ od}, \sigma) \rightarrow \sigma''}$$

$$\frac{\mathcal{B}[\![b]\!]\sigma = \textbf{false}}{(\textbf{while } b \textbf{ do } c_1 \textbf{ od}, \sigma) \rightarrow \sigma}$$

Analog zur Einzelschrittsemantik zeigt man (Übung):

1. Jedes $\sigma \in \Sigma$ hat als Knoten des Transitionssystems $(\Sigma \cup \mathbf{Cmd} \times \Sigma, \rightarrow)$ Ausgangsgrad 0.
2. Jedes $(c, \sigma) \in \mathbf{Cmd} \times \Sigma$ hat als Knoten des Transitionssystems $(\Sigma \cup \mathbf{Cmd} \times \Sigma, \rightarrow)$ Ausgangsgrad 0 oder 1. Darüberhinaus haben diese Knoten alle den Eingangsgrad 0.

Hieraus folgt wieder, dass das Transitionssystem deterministisch ist, und die Vorschrift

$$\mathcal{C}_{\text{BIG}}(c, \sigma) = \sigma' \;:\Longleftrightarrow\; (c, \sigma) \rightarrow \sigma'$$

eine wohldefinierte partielle Funktion $\mathcal{C}_{\text{BIG}} : \mathbf{Cmd} \times \Sigma \;\rightarrow_p\; \Sigma$ definiert.

Die beiden vorgestellten Semantiken sind äquivalent:

Satz 8

*Für alle Anweisungen $c \in$ **Cmd** und alle Belegungen $\sigma \in \Sigma$ gilt*

$$\mathcal{C}_{BIG}(c, \sigma) = \mathcal{C}_{SOS}(c, \sigma),$$

d.h. $\mathcal{C}_{BIG}(c, \sigma)$ ist definiert genau dann, wenn auch $\mathcal{C}_{SOS}(c, \sigma)$ definiert ist und in dem Fall haben beide den gleichen Wert.

Beweis: Wir zeigen:

$$\forall c \in \textbf{Cmd}\ \forall \sigma \in \Sigma : \mathcal{C}_{\text{BIG}}(c, \sigma) = \mathcal{C}_{\text{SOS}}(c, \sigma).$$

Insbesondere: Die linke Seite ist genau dann definiert, wenn die rechte Seite definiert ist.

Beweis durch strukturelle Induktion über den Ausdruck c.

1.Fall: $c = \textbf{skip}$ oder $c = (X := a)$: trivial

2.Fall: $c = (c_1; c_2)$ für $c_1, c_2 \in \textbf{Cmd}$.

Induktionsvoraussetzung: $\mathcal{C}_{\text{BIG}}(c_i, \tau) = \mathcal{C}_{\text{SOS}}(c_i, \tau)$ für alle $\tau \in \Sigma$, $i \in \{1, 2\}$.

Falls $\mathcal{C}_{\text{BIG}}(c_1, \sigma) = \mathcal{C}_{\text{SOS}}(c_1, \sigma) =$ undefiniert, ergibt sich auch $\mathcal{C}_{\text{BIG}}(c_1; c_2, \sigma) = \mathcal{C}_{\text{SOS}}(c_1; c_2, \sigma) =$ undefiniert.

Sei also $\mathcal{C}_{\text{BIG}}(c_1, \sigma) = \mathcal{C}_{\text{SOS}}(c_1, \sigma) = \sigma'$. Daraus folgt $\mathcal{C}_{\text{BIG}}(c_1; c_2, \sigma) = \mathcal{C}_{\text{BIG}}(c_2, \sigma') = \mathcal{C}_{\text{SOS}}(c_2, \sigma')$.

Weiterhin folgt aus $\mathcal{C}_{\text{SOS}}(c_1, \sigma) = \sigma'$, dass im Transitionssystem der Einzelschrittsemantik ein Pfad $(c_1, \sigma) \to_1^* \sigma'$ existiert.

Damit ergibt sich auch der Pfad $(c_1; c_2, \sigma) \to_1^* (c_2, \sigma')$.

Also: $\mathcal{C}_{\text{SOS}}(c_1; c_2, \sigma) = \mathcal{C}_{\text{SOS}}(c_2, \sigma') = \mathcal{C}_{\text{BIG}}(c_1; c_2, \sigma)$.

3.Fall: $c =$ **if** b **then** c_1 **else** c_2 **fi**.

Induktionsvoraussetzung: $\mathcal{C}_{\text{BIG}}(c_i, \tau) = \mathcal{C}_{\text{SOS}}(c_i, \tau)$ für alle $\tau \in \Sigma$, $i \in \{1, 2\}$.

Falls $\mathcal{B}[\![b]\!]\sigma =$ **true** gilt, erhalten wir

$$\mathcal{C}_{\text{SOS}}(c, \sigma) = \mathcal{C}_{\text{SOS}}(c_1, \sigma) = \mathcal{C}_{\text{BIG}}(c_1, \sigma) = \mathcal{C}_{\text{BIG}}(c, \sigma).$$

Entsprechend ergibt sich im Fall $\mathcal{B}[\![b]\!]\sigma =$ **false**

$$\mathcal{C}_{\text{SOS}}(c, \sigma) = \mathcal{C}_{\text{SOS}}(c_2, \sigma) = \mathcal{C}_{\text{BIG}}(c_2, \sigma) = \mathcal{C}_{\text{BIG}}(c, \sigma).$$

4.Fall: $c =$ **while** b **do** c_1 **od**

Fall 4.1: $\mathcal{B}[\![b]\!]\sigma =$ **false**

Dann gilt $\mathcal{C}_{\text{SOS}}(c, \sigma) = \sigma = \mathcal{C}_{\text{BIG}}(c, \sigma)$.

Fall 4.2: $\mathcal{B}[\![b]\!]\sigma =$ **true**.

Wir gehen erst davon aus, dass $\mathcal{C}_{\text{SOS}}(c, \sigma)$ definiert ist: $\mathcal{C}_{\text{SOS}}(c, \sigma) = \sigma' \in \Sigma$.

Wir müssen zeigen, dass dann auch $\mathcal{C}_{\text{BIG}}(c, \sigma) = \sigma'$ gilt.

Im Transitionssystem der Einzelschrittsemantik existiert ein Weg der Form

$$(c, \sigma) \to_1 (c_1; c, \sigma) \to_1^* (c, \sigma'') \to_1^* \sigma', \text{ wobei } (c_1, \sigma) \to_1^* \sigma''.$$

Der Teilweg $(c, \sigma'') \to_1^* \sigma'$ ist kürzer als der Gesamtweg.

Induktiv (über die Weglänge) schließen wir daraus

$$\mathcal{C}_{\text{BIG}}(c, \sigma'') = \sigma'.$$

Aus $(c_1, \sigma) \to_1^* \sigma''$, d.h. $\mathcal{C}_{\text{SOS}}(c_1, \sigma) = \sigma''$, folgt induktiv

$$\mathcal{C}_{\text{BIG}}(c_1, \sigma) = \sigma''.$$

Insgesamt erhalten wir somit

$$\mathcal{C}_{\text{BIG}}(c, \sigma) = \sigma'' = \mathcal{C}_{\text{SOS}}(c, \sigma).$$

Für die Umkehrung gehen wir davon aus, dass $\mathcal{C}_{\text{BIG}}(c, \sigma)$ definiert ist:

$$\mathcal{C}_{\text{BIG}}(c, \sigma) = \sigma' \in \Sigma.$$

Wir müssen zeigen, dass dann auch $\mathcal{C}_{\text{SOS}}(c, \sigma) = \sigma'$ gilt.

Wegen $\mathcal{B}[\![b]\!]\sigma = \textbf{true}$ existiert ein Beweisbaum für $(c, \sigma) \to \sigma'$, in dessen zweiter Zeile (von unten) die folgenden Einträge stehen:

$$\mathcal{B}[\![b]\!]\sigma = \textbf{true}, \quad (c_1, \sigma) \to \sigma'' \quad \text{und} \quad (c, \sigma'') \to \sigma'$$

Induktiv über die Höhe des Beweisbaums können wir schließen, dass

$$\mathcal{C}_{\text{SOS}}(c, \sigma'') = \mathcal{C}_{\text{BIG}}(c, \sigma'') = \sigma'$$

gilt, und

$$\mathcal{C}_{\text{SOS}}(c_1, \sigma) = \mathcal{C}_{\text{BIG}}(c_1, \sigma) = \sigma''.$$

Daraus folgt $(c, \sigma) \to_1 (c_1; c, \sigma) \to_1^* (c, \sigma'') \to_1^* \sigma'$, d.h.

$$\mathcal{C}_{\text{SOS}}(c, \sigma) = \sigma' = \mathcal{C}_{\text{BIG}}(c, \sigma).$$

Als ein Beispiel für die Anwendung der Ergebnissemantik beweisen wir die Gleichwertigkeit der Schleife $w =$ **while** b **do** c **od** mit der Anweisung **if** b **then** $c; w$ **else skip fi**, indem wir zeigen, dass für alle $\sigma \in \Sigma$ gilt

$$\mathcal{C}_{\text{BIG}}(w, \sigma) = \mathcal{C}_{\text{BIG}}(\textbf{if } b \textbf{ then } c; w \textbf{ else skip fi}, \sigma).$$

1. Fall: $\mathcal{B}[\![b]\!]\sigma =$ **false**.

Dann gilt $\mathcal{C}_{\text{BIG}}(w, \sigma) = \sigma$ und

$$\mathcal{C}_{\text{BIG}}(\textbf{if } b \textbf{ then } c; w \textbf{ else skip fi}, \sigma) = \mathcal{C}_{\text{BIG}}(\textbf{skip}, \sigma) = \sigma.$$

2.Fall: $\mathcal{B}[\![b]\!]\sigma =$ **true**.

Gelte zunächst $\mathcal{C}_{\text{BIG}}(c, \sigma) =$ undefiniert. Dann gilt

$$\begin{aligned} \mathcal{C}_{\text{BIG}}(\textbf{if } b \textbf{ then } c; w \textbf{ else skip fi}, \sigma) &= \mathcal{C}_{\text{BIG}}(c; w, \sigma) \\ &= \text{undefiniert} \\ &= \mathcal{C}_{\text{BIG}}(w, \sigma) \end{aligned}$$

Gelte nun $\mathcal{C}_{\text{BIG}}(c, \sigma) = \sigma'$. Dann gilt

$$\begin{aligned}
\mathcal{C}_{\text{BIG}}(\textbf{if } b \textbf{ then } c; w \textbf{ else skip fi}, \sigma) &= \mathcal{C}_{\text{BIG}}(c; w, \sigma) \\
&= \mathcal{C}_{\text{BIG}}(w, \sigma') \\
&= \mathcal{C}_{\text{BIG}}(w, \sigma)
\end{aligned}$$

4. Denotatoniale Semantik

Wir lösen uns jetzt von dem Prinzip der „Ausführung“ eines Programms. Stattdessen identifizieren wir ein Programm mit der von ihm berechneten Funktion.

Als Auswertungsschema für Ausdrücke (arithmetische und boolesche) werden wir die Funktionen $\mathcal{A}$ und $\mathcal{B}$ wie bisher verwenden.

Die Anweisungen erhalten direkt eine Bedeutung in Form einer partiellen Funktion von Σ nach Σ:

$$\mathcal{C} : \mathbf{Cmd} \to (\Sigma \ \to_p \ \Sigma).$$

Sinnvollerweise soll $\mathcal{C}[\![c]\!]\sigma = \mathcal{C}_{\text{sos}}[\![c]\!]\sigma$ für alle $c \in \mathbf{Cmd}$ und $\sigma \in \Sigma$ gelten.

skip: $\mathcal{C}[\![\textbf{skip}]\!] \;=\; \mathrm{id}_\Sigma$

Zuweisung: $\mathcal{C}[\![X \;:=\; a]\!] : \Sigma \to \Sigma, \quad \sigma \mapsto \sigma[\mathcal{A}[\![a]\!]\sigma / X]$

Man schreibt hierfür auch $\mathcal{C}[\![X \;:=\; a]\!] \;=\; \lambda\sigma.\sigma[\mathcal{A}[\![a]\!]\sigma / X]$
(die Funktion, die σ auf $\sigma[\mathcal{A}[\![a]\!]\sigma / X]$ abbildet.)

Man prüft für $c = \textbf{skip}$ und $c = (X := a)$ leicht die Gültigkeit von

$$\mathcal{C}[\![c]\!]\sigma \;=\; \mathcal{C}_{\mathrm{sos}}[\![c]\!]\sigma.$$

Hintereinanderausführung: $\mathcal{C}[\![c_1; c_2]\!] = \mathcal{C}[\![c_2]\!] \circ \mathcal{C}[\![c_1]\!]$.

Unter der Induktionsvoraussetzung

$$\mathcal{C}[\![c_i]\!]\tau = \mathcal{C}_{\text{sos}}[\![c_i]\!]\tau$$

für alle $i \in \{1, 2\}$ und $\tau \in \Sigma$ folgt leicht

$$\mathcal{C}[\![c_1; c_2]\!]\sigma = \mathcal{C}_{\text{sos}}[\![c_1; c_2]\!]\sigma.$$

if-then-else:

Wir definieren zunächst die folgende Funktion:

$$\text{cond}\colon (\Sigma \to^p \mathbb{B}) \times (\Sigma \to^p \Sigma) \times (\Sigma \to^p \Sigma) \;\to\; (\Sigma \to^p \Sigma)$$

$$(\beta, f, g) \mapsto \text{cond}(\beta, f, g),$$

wobei $\text{cond}(\beta, f, g)(\sigma)$ nur definiert ist, falls $\beta(\sigma)$ definiert ist.
(Bei uns ist dies immer der Fall!)

Falls $\beta(\sigma)$ definiert, setzen wir:

$$\text{cond}(\beta, f, g)(\sigma) = \begin{cases} f(\sigma) & \text{falls } \beta(\sigma) = \textbf{true}, \\ g(\sigma) & \text{falls } \beta(\sigma) = \textbf{false}. \end{cases}$$

Nun definieren wir für $c = \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ fi}$

$$\mathcal{C}[\![c]\!] = \text{cond}(\mathcal{B}[\![b]\!], \mathcal{C}[\![c_1]\!], \mathcal{C}[\![c_2]\!]).$$

Man erhält für alle $\sigma \in \Sigma$:

$$\mathcal{C}[\![c]\!]\sigma = \mathcal{C}_{\text{sos}}[\![c]\!]\sigma.$$

while: Sei $w =$ **while** b **do** c **od**.

Induktionsvoraussetzung: $\mathcal{C}[\![c]\!]$ ist schon definiert und für alle $\sigma \in \Sigma$ gilt:

$$\mathcal{C}[\![c]\!]\sigma = \mathcal{C}_{\text{sos}}[\![c]\!]\sigma.$$

Wir wissen auch

$$\mathcal{C}_{\text{sos}}[\![w]\!]\sigma = \mathcal{C}_{\text{sos}}[\![\textbf{if } b \textbf{ then } c; w \textbf{ else skip fi}]\!]\sigma,$$

also

$$\mathcal{C}_{\text{sos}}[\![w]\!] = \text{cond}(\mathcal{B}[\![b]\!], \mathcal{C}_{\text{sos}}[\![w]\!] \circ \mathcal{C}[\![c]\!], \mathcal{C}[\![\textbf{skip}]\!]).$$

$\mathcal{C}_{\text{sos}}[\![w]\!]$ taucht hier links und rechts auf, löst also die folgende Gleichung:

$$f = \text{cond}(\mathcal{B}[\![b]\!], f \circ \mathcal{C}[\![c]\!], \text{id}_\Sigma).$$

Problem: Es kann viele Fixpunkte (Lösungen der Gleichungen) geben.

Beispiel: $w =$ **while true do skip od**

Dann erhalten wir die Gleichung

$$\begin{aligned} f &= \mathrm{cond}(\mathcal{B}[\![\textbf{true}]\!], f \circ \mathcal{C}[\![\textbf{skip}]\!], \mathrm{id}_\Sigma) \\ &= \mathrm{cond}(\mathcal{B}[\![\textbf{true}]\!], f, \mathrm{id}_\Sigma) \\ &= f \end{aligned}$$

Wir definieren den folgenden Operator:

$$\Gamma_{b,c} : (\Sigma \rightarrow_p \Sigma) \rightarrow (\Sigma \rightarrow_p \Sigma)$$

durch

$$\Gamma_{b,c}(f) = \mathrm{cond}(\mathcal{B}[\![b]\!], f \circ \mathcal{C}[\![c]\!], \mathrm{id}_\Sigma).$$

Wir sehen $\Gamma_{b,c}$ als Funktion an in der auf der Grundmenge $(\Sigma \rightharpoonup_p \Sigma)$ definierten CPO mit $f \sqsubseteq g$ genau dann, wenn aus „$f(x)$ definiert“ folgt, dass auch $g(x)$ definiert ist und $f(x) = g(x)$ gilt.

Offenbar ist $\Gamma_{b,c}(\mathcal{C}_{\text{sos}}[\![w]\!]) = \mathcal{C}_{\text{sos}}[\![w]\!]$, d.h. $\mathcal{C}_{\text{sos}}[\![w]\!]$ ist Fixpunkt von $\Gamma_{b,c}$.

Wir definieren nun $\mathcal{C}[\![w]\!]$ als den kleinsten Fixpunkt von $\Gamma_{b,c}$.

Um die Existenz des kleinsten Fixpunkts aus Kleenes Fixpunksatz ableiten zu können, benötigen wir:

Lemma 9

$\Gamma_{b,c}$ ist stetig.

Beweis: Wir zeigen:

(a) $\Gamma_{b,c}$ ist monoton.

(b) Für eine gerichtete Menge $D = \{f_i \mid i \in I\} \subseteq (\Sigma \to_p \Sigma)$ gilt:

$$\Gamma_{b,c}(\sqcup D) = \sqcup\{\Gamma_{b,c}(f_i) \mid i \in I\}.$$

Zu (a): Sei $f \sqsubseteq g$, und sei $\Gamma_{b,c}(f)(\sigma)$ definiert.

Ist $\mathcal{B}[\![b]\!]\sigma = \textbf{false}$, dann gilt $\Gamma_{b,c}(f)(\sigma) = \mathrm{id}_\Sigma(\sigma) = \Gamma_{b,c}(g)(\sigma)$.

Ist $\mathcal{B}[\![b]\!]\sigma = \textbf{true}$, dann gilt $\Gamma_{b,c}(f)(\sigma) = (f \circ \mathcal{C}[\![c]\!])(\sigma)$.

Also muss $\mathcal{C}[\![c]\!](\sigma)$ definiert sein, etwa $\mathcal{C}[\![c]\!](\sigma) = \sigma'$, und dann muss $f(\sigma')$ definiert sein.

Wegen $f \sqsubseteq g$ muss $g(\sigma')$ definiert sein und $f(\sigma') = g(\sigma')$.

Es folgt $\Gamma_{b,c}(g)(\sigma) = (g \circ \mathcal{C}[\![c]\!])(\sigma) = g(\sigma') = f(\sigma')$.

Also ist $\Gamma_{b,c}(g)(\sigma)$ definiert und $\Gamma_{b,c}(g)(\sigma) = \Gamma_{b,c}(f)(\sigma)$.

Zu (b): Sei $D = \{f_i \mid i \in I\} \subseteq \Sigma \to_p \Sigma$ eine gerichtete Menge von Funktionen.

Sei $f = \sqcup D$ (f existiert!).

Zu zeigen ist $\Gamma_{b,c}(f) = \sqcup\{\Gamma_{b,c}(f_i) \mid i \in I\}$.

Wegen $f_i \sqsubseteq f$ und der Monotonie von $\Gamma_{b,c}$ gilt $\Gamma_{b,c}(f_i) \sqsubseteq \Gamma_{b,c}(f)$.

Also $\sqcup\{\Gamma_{b,c}(f_i) \mid i \in I\} \sqsubseteq \Gamma_{b,c}(f)$.

Das heißt: Wo beide Seiten definiert sind, sind sie gleich, und wenn die linke Seite definiert ist, ist es auch die rechte.

Bleibt also zu zeigen: Wenn die rechte Seite definiert ist, dann auch die linke.

Oder äquivalent: Wenn $\Gamma_{b,c}(f)(\sigma)$ definiert ist, dann existiert ein i mit $\Gamma_{b,c}(f_i)(\sigma)$ definiert.

Falls $\mathcal{B}[\![b]\!]\sigma = \textbf{false}$ gilt, ist $\Gamma_{b,c}(g)(\sigma)$ für alle g definiert.

Falls $\mathcal{B}[\![b]\!]\sigma = \textbf{true}$ gilt, gilt $\Gamma_{b,c}(f)(\sigma) = (f \circ \mathcal{C}[\![c]\!])(\sigma)$.

Also existiert ein $\sigma' \in \Sigma$ mit $\mathcal{C}[\![c]\!]\sigma = \sigma'$ und $f(\sigma')$ definiert.

Da $f = \sqcup\{f_i \mid i \in I\}$, gibt es ein i mit $f_i(\sigma')$ definiert.

Also ist auch $\Gamma_{b,c}(f_i)(\sigma) = (f_i \circ \mathcal{C}[\![c]\!])(\sigma)$ definiert.

Da $\Gamma_{b,c}$ stetig ist, wissen wir, dass der kleinste Fixpunkt existiert.

Wir definieren für $w = \textbf{while } b \textbf{ do } c \textbf{ od}$: $\mathcal{C}[\![w]\!] = \text{fix}(\Gamma_{b,c})$.

Lemma 10

Es gilt für alle $\sigma \in \Sigma$: $\mathcal{C}[\![w]\!]\sigma = \mathcal{C}_{sos}[\![w]\!]\sigma$.

Beweis:

Wir wissen bereits, dass $\mathcal{C}_{\text{sos}}[\![w]\!]$ Fixpunkt von $\Gamma_{b,c}$ ist.

Da $\mathcal{C}[\![w]\!]$ der kleinste Fixpunkt von $\Gamma_{b,c}$ ist, gilt $\mathcal{C}[\![w]\!] \sqsubseteq \mathcal{C}_{\text{sos}}[\![w]\!]$.

D.h. falls $\mathcal{C}[\![w]\!]\sigma$ definiert ist, dann auch $\mathcal{C}_{\text{sos}}[\![w]\!]\sigma$ und es gilt

$$\mathcal{C}[\![w]\!]\sigma = \mathcal{C}_{\text{sos}}[\![w]\!]\sigma.$$

Noch zu zeigen: Wenn $\mathcal{C}_{\text{sos}}[\![w]\!]\sigma$ definiert ist, dann auch $\mathcal{C}[\![w]\!]\sigma$.

„$\mathcal{C}_{\text{sos}}[\![w]\!]\sigma$ definiert“ bedeutet, es gibt ein $\sigma' \in \Sigma$ und einen Weg $(w, \sigma) \to_1^* \sigma'$ im Transitionssystem.

Sei n die Länge dieses Wegs.

Wir können induktiv annehmen, dass für alle σ_1 mit $(w, \sigma_1) \to_1^* \sigma'$ auf kürzerem Weg (Länge $q < n$) gilt, dass $\mathcal{C}[\![w]\!]\sigma_1$ definiert ist.

Falls $\mathcal{B}[\![b]\!]\sigma = \textbf{false}$ gilt, so gilt

$$\mathcal{C}[\![w]\!]\sigma = \text{fix}(\Gamma_{b,c})(\sigma) = \Gamma_{b,c}(\text{fix}(\Gamma_{b,c}))(\sigma) = \sigma.$$

Also ist $\mathcal{C}[\![w]\!]\sigma$ definiert.

Falls $\mathcal{B}[\![b]\!]\sigma = \textbf{true}$ gilt, gibt es ein $\sigma_1 \in \Sigma$ mit

$$(w, \sigma) \to_1 (c; w, \sigma) \to_1^* (w, \sigma_1) \to_1^* \sigma' \quad \text{und} \quad (c, \sigma) \to_1^* \sigma_1.$$

Nach Induktionsvoraussetzung ist $\mathcal{C}[\![w]\!]\sigma_1$ definiert und es gilt $\mathcal{C}[\![c]\!]\sigma = \sigma_1$.

Damit folgt

$$\begin{aligned}
\mathcal{C}[\![w]\!]\sigma &= \mathrm{fix}(\Gamma_{b,c})(\sigma) \\
&= \Gamma_{b,c}(\mathcal{C}[\![w]\!])(\sigma) \\
&= (\mathcal{C}[\![w]\!] \circ \mathcal{C}[\![c]\!])(\sigma) \\
&= \mathcal{C}[\![w]\!](\mathcal{C}[\![c]\!]\sigma) \\
&= \mathcal{C}[\![w]\!]\sigma_1.
\end{aligned}$$

Also ist $\mathcal{C}[\![w]\!]\sigma$ definiert.

Also: $\mathcal{C}_{\mathrm{SOS}}[\![w]\!]\sigma$ definiert genau dann, wenn $\mathcal{C}[\![w]\!]\sigma$ definiert ist, und dann haben beide den gleichen Wert.

Satz 11

Die operationale Semantik $\mathcal{C}_{SOS}$ und die denotationale Semantik $\mathcal{C}$ für **IMP** *stimmen überein.*

Beispiel: Definiere das Fakultätsprogramm

$$\begin{aligned} \mathrm{faku} &\equiv Y := 1; w \quad \text{mit} \\ w &\equiv \textbf{while } X > 1 \textbf{ do } Y := Y \cdot X;\ X := X - 1 \textbf{ od} \end{aligned}$$

Setze **Loc** $= \{X, Y\}$, d.h. $\Sigma = \mathbb{N}^2$.

Wir interpretieren $\sigma = (m, n)$ durch $\sigma(X) = m$, $\sigma(Y) = n$.

Definiere $f : \Sigma \to \Sigma$ und $g : \Sigma \to \Sigma$ durch

$$\begin{aligned} f(m, n) &= (\min(1, m), m!) \\ g(m, n) &= (\min(1, m), (m!) \cdot n). \end{aligned}$$

Behauptung: $\mathcal{C}[\![w]\!] = g$

Hieraus folgt leicht $\mathcal{C}[\![\mathrm{faku}]\!] = f$.

Sei $b \equiv (X > 1)$ und $c \equiv (Y := Y \cdot X; X := X - 1)$, d.h.
$w =$ **while** b **do** c **od**.

Wir wollen zeigen, dass g ein Fixpunkt von $\Gamma_{b,c}$ ist, d.h. $\Gamma_{b,c}(g) = g$.

Hieraus folgt $\mathcal{C}[\![w]\!] \sqsubseteq g$.

Wir berechnen also $\Gamma_{b,c}(g)(\sigma)$ für $\sigma = (m, n)$.

1. Fall: $m \leq 1$. Dann ist $\mathcal{B}[\![b]\!]\sigma =$ **false**.

Es folgt $\Gamma_{b,c}(g)(\sigma) = \mathrm{id}_\Sigma(\sigma) = \sigma$ und $g(\sigma) = (m, n) = \sigma$.

Also $\Gamma_{b,c}(g)(\sigma) = g(\sigma)$.

2. Fall: $m > 1$. Dann ist $\mathcal{B}[\![b]\!]\sigma =$ **true** und es folgt:

$$\begin{aligned}
\Gamma_{b,c}(g)(\sigma) &= (g \circ \mathcal{C}[\![c]\!])(\sigma) \\
&= (g \circ \mathcal{C}[\![X := X - 1]\!] \circ \mathcal{C}[\![Y := Y \cdot X]\!])(\sigma) \\
&= (g \circ \mathcal{C}[\![X := X - 1]\!])(m, n \cdot m) \\
&= g(m - 1, n \cdot m) \\
&= g(m, n)
\end{aligned}$$

Wir erhalten in jedem Fall $\Gamma_{b,c}(g)(\sigma) = g(\sigma)$.

Also ist g Fixpunkt von $\Gamma_{b,c}$ und es gilt $\mathcal{C}[\![w]\!] \sqsubseteq g$.

Wenn $\mathcal{C}[\![w]\!] \neq g$ wäre, müsste es ein σ geben, so dass $g(\sigma)$ definiert ist, aber $\mathcal{C}[\![w]\!]\sigma$ nicht.

Die while-Schleife müsste also unendlich oft ausgeführt werden.

Das ist nicht möglich, da X zu Beginn einen endlichen Wert hat und in jedem Durchlauf herabgesetzt wird.

Formal: Wenn $\sigma = (m, n)$, dann terminiert die while-Schleife nach maximal $m - 1$ Durchläufen.

Also gilt $\mathcal{C}[\![w]\!] = g$ und daher $\mathcal{C}[\![\text{faku}]\!] = f$.

Weiteres Beispiel:

Achtung: Wir betrachten hier als Grundbereich $\mathbb{Z}$ und rechnen mit der ganzzahligen Subtraktion!

Betrachte die while-Schleife $w \equiv$ **while** b **do** c **od** mit

$$\begin{aligned} b &\equiv (X \neq 0) \\ c &\equiv (X := X - Y\,;\, Y := Y - X) \end{aligned}$$

Sei $f : \mathbb{Z} \times \mathbb{Z} \to_p \mathbb{Z} \times \mathbb{Z}$ mit $f(X, Y) = \mathcal{C}[\![w]\!](X, Y)$.

Es sieht so aus, als berechne f den ggT.

Dies stimmt aber nur sehr bedingt, wie wir gleich sehen werden.

Sei $g : \mathbb{Z} \times \mathbb{Z} \to_p \mathbb{Z} \times \mathbb{Z}$ mit $g(X, Y) = (0, \mathrm{ggT}(X, Y))$.

Wir zeigen zunächst $\Gamma_{b,c}(g) = g$, d.h. g ist ein Fixpunkt.

Es sei $\sigma = (m, n)$, wobei $\sigma(X) = m$, $\sigma(Y) = n$.

1. Fall: $\mathcal{B}[\![b]\!](m, n) = \text{false}$.

Dann ist $m = 0$ und damit

$$\Gamma_{b,c}(g)(0, n) = (0, n) = (0, \text{ggT}(0, n)) = g(0, n).$$

2. Fall: $\mathcal{B}[\![b]\!](m, n) = \text{true}$.

Dann ist $m \neq 0$ und damit

$$\begin{aligned} \Gamma_{b,c}(g)(m, n) &= g(m - n, 2n - m) \\ &= (0, \text{ggT}(m - n, 2n - m)) \\ &= (0, \text{ggT}(m, n)) \\ &= g(m, n). \end{aligned}$$

Dies legt die Vermutung $f = g$ nahe, da $f = \mathcal{C}[\![w]\!]$ und $\Gamma_{b,c}(g) = g$ gilt.

Aber Vorsicht: Gezeigt wurde nur $f \sqsubseteq g$.

Die while-Schleife terminiert aber nicht immer.

Betrachte etwa $m > 0$ und $n \leq 0$.

Dann gilt auch $m - n > 0$ und $2n - m \leq 0$.

Sie terminiert auch nicht für $m < 0$ und $n \geq 0$.

Insbesondere terminiert sie nicht für $(m, 0)$ mit $m \neq 0$.

5. Axiomatische Semantik

5.1 Grundidee

Wir beschreiben die Bedeutung eines Programms, indem wir eine Vor- und eine Nachbedingung angeben:

$$\{A\}\ c\ \{B\}$$

Hierbei ist

- A die Vorbedingung,
- $c \in \mathbf{Cmd}$ und
- B die Nachbedingung.

Die Bedeutung von $\{A\}\ c\ \{B\}$ sei:

Wenn vor Ausführung von c die Bedingung A erfüllt ist und c terminiert, dann ist nach Ausführung von c die Bedingung B erfüllt.

Als Beispiel verwenden wir wieder das Fakultätsprogramm faku:

$$
\begin{array}{rll}
\{X = N\} & Y := 1; & \{X = N \wedge Y = 1\} \\
\{Y \cdot X! = N!\} & \textbf{while } X > 1 \textbf{ do} & \\
\{Y \cdot X! = N!\} & \quad Y := Y \cdot X; & \{Y \cdot X! = N! \cdot X\} \\
\{Y \cdot X! = N! \cdot X\} & \quad X := X - 1 & \{Y \cdot X! = N!\} \\
 & \textbf{od} & \{Y \cdot X! = N! \wedge X \leq 1\} \\
 & & \{Y = N!\}
\end{array}
$$

Die Aussage $\{A\}\ c\ \{B\}$ ist eine partielle Korrektheitsaussage, da wir sagen nur etwas über den Fall, dass c terminiert.

Analog nennt man $[A]\ c\ [B]$ eine Korrektheitsaussage mit der Bedeutung:

Wenn A erfüllt ist und c ausgeführt wird, dann terminiert c und anschließend ist B erfüllt.

Wir müssen festlegen, welche Art von Aussagen für A und B in Frage kommen.

Wir definieren zunächst die Menge **Aexp**$'$ von erweiterten arithmetischen Ausdrücken:

$$a \ ::= \ n \mid X \mid i \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \cdot a_1,$$

wobei $n \in \mathbb{N}$, $X \in$ **Loc**, $i \in$ **Intvar**.

Intvar ist eine Menge von Integer-Variablen, z.B. **Intvar** $= \{i_1, i_2, i_3, \ldots\}$.

Wir definieren die Menge **Assn** von Zusicherungen (englisch: assertions):

$$\begin{aligned} A \ ::= \quad & \textbf{true} \mid \textbf{false} \mid a_0 = a_1 \mid a_0 < a_1 \mid \\ & a_0 \leq a_1 \mid a_0 > a_1 \mid a_0 \geq a_1 \mid \neg A \mid \\ & A_1 \wedge A_2 \mid A_1 \vee A_2 \mid A_1 \Rightarrow A_2 \mid \\ & A_1 \Leftrightarrow A_2 \mid \forall i : A \mid \exists i : A, \end{aligned}$$

wobei $a_1, a_2 \in$ **Aexp**$'$ und $i \in$ **Intvar**.

Klammerungen werden gesetzt, wo immer es notwendig oder erwünscht ist.

Die Menge der Variablen ist nun $\mathbb{V} = \mathbf{Loc} \cup \mathbf{Intvar}$.

In einer Zusicherung $A \in \mathbf{Assn}$ kann ein $i \in \mathbf{Intvar}$ gebunden oder frei vorkommen.

Wir definieren die Menge der freien Integer-Variablen $\mathrm{FV}(A) \subseteq \mathbf{Intvar}$ einer Zusicherung A wie üblich.

$\mathrm{FV}(\textbf{true}) = \mathrm{FV}(\textbf{false}) = \emptyset$

$\mathrm{FV}(a_0 = a_1) =$ alle in a_0 oder a_1 frei vorkommenden Integer-Variablen

(analog für $<$, $\leq$, $>$, $\geq$)

$\mathrm{FV}(\neg A) = \mathrm{FV}(A)$

$\mathrm{FV}(A_1 \wedge A_2) = \mathrm{FV}(A_1) \cup \mathrm{FV}(A_2)$

(analog für $\vee$, $\Rightarrow$, $\Leftrightarrow$)

$\mathrm{FV}(\forall i : A) = \mathrm{FV}(A) \setminus \{i\}$

$\mathrm{FV}(\exists i : A) = \mathrm{FV}(A) \setminus \{i\}$

Beispiel: $\mathrm{FV}((\forall i : i > 0) \vee (i < 2)) = \{i\}$

Das einzige freie Vorkommen von i in dieser Zusicherung ist das in $i < 2$.

Gegeben sei $A \in \textbf{Assn}$, $\sigma \in \Sigma$ und es gelte $\mathrm{FV}(A) = \emptyset$.

Dann kann man A unter σ auswerten.

Wir schreiben $\sigma(A) = \textbf{true}$ bzw. $\sigma(A) = \textbf{false}$, falls A unter σ wahr bzw. falsch ist.

Für Zusicherungen mit freien Variablen werden diese zunächst interpretiert, d.h. es wird ihnen eine natürliche Zahl als Wert zugeordnet, durch die Interpretation $I : \textbf{Intvar} \to \mathbb{N}$.

Wir schreiben $\sigma \models^I A$ oder alternativ $\sigma(I(A)) = \textbf{true}$ falls die durch I interpretierte Zusicherung A im Speicherzustand σ wahr ist,

Wir schreiben $\sigma \models A$, falls $\sigma \models^I A$ für alle Interpretationen I gilt.

Beispiel: Wir konstruieren eine Formel $A \in \mathbf{Assn}$, die eine freie Variable i enthält und die genau dann erfüllt ist, wenn i eine Primzahl ist:

$$\sigma \models^I A \quad \Leftrightarrow \quad I(i) \text{ ist Primzahl}$$

Sei $A \equiv \ \forall j : ((j \leq 1) \vee (j \geq i) \vee (\forall k : j \cdot k \neq i))$.

Wähle $I(i) = n \in \mathbb{N}$ beliebig. Dann gilt

$$I(A) \equiv \forall j : ((j \leq 1) \vee (j \geq n) \vee (\forall k : j \cdot k \neq n)).$$

1. Fall: n ist Primzahl. Sei $j \in \mathbb{N}$ beliebig.

Falls ein k existiert mit $j \cdot k = n$, dann gilt $j = 1$ oder $j = n$.

Falls kein solches k existiert, ist $\forall k : j \cdot k \neq n$ wahr.

Also gilt $\forall j : (...)$ und daher $\sigma \models^I A$.

2. Fall: n ist keine Primzahl, etwa $n = a \cdot b$ mit $1 < a < n$.

Für $j = a$ gilt $\neg(j \leq 1) \wedge \neg(j \geq n) \wedge \exists k : j \cdot k = n$.

Also ist $\sigma \models^I A$ falsch.

Beispiel: Wir konstruieren eine Formel KGV $\in$ **Assn** mit freien Variablen i, j, k, die wahr ist genau dann, wenn i das kleinste gemeinsame Vielfache von j und k ist:

$$\sigma \models^I \text{KGV} \;\Leftrightarrow\; \text{kgv}(I(j), I(k)) = I(i)$$

Lösung:

$$\begin{aligned}\text{KGV} \;\equiv\; & \exists j' : j' \cdot j = i \;\wedge\; \exists k' : k' \cdot k = i \;\wedge \\ & \forall i' : (((\exists j' : j' \cdot j = i') \wedge (\exists k' : k' \cdot k = i')) \;\Rightarrow\; i \leq i')\end{aligned}$$

Zu zeigen:

- Wenn $I(i) = n, I(j) = m, I(k) = r$ und $I(\text{KGV})$ erfüllt, dann ist $n = \text{kgv}(m, r)$.
- Wenn $I(i) = n, I(j) = m, I(k) = r$ und $n = \text{kgv}(m, r)$, dann ist $I(\text{KGV})$ erfüllt.

Die Bedingung $\sigma(I(A)) = \textbf{true}$ kann wie folgt formalisiert werden:

- $\sigma(I(n)) = n$ für $n \in \mathbb{N}$
- $\sigma(I(X)) = \sigma(X)$ für $X \in \textbf{Loc}$
- $\sigma(I(i)) = I(i)$ für $i \in \textbf{Intvar}$
- $\sigma(I(a_0 + a_1)) = \sigma(I(a_0)) + \sigma(I(a_1))$, etc.
- $\sigma(I(\textbf{true})) = \textbf{true}$
- $\sigma(I(\textbf{false})) = \textbf{false}$
- $\sigma(I(a_0 = a_1)) = \textbf{true}$, falls $\sigma(I(a_0)) = \sigma(I(a_1))$, **false**, sonst; etc.
- $\sigma(I(\neg A)) = \neg\sigma(I(A))$
- $\sigma(I(A_1 \vee A_2)) = \sigma(I(A_1)) \vee \sigma(I(A_2))$, etc.

Etwas komplizierter wird es bei den Formeln der Form $\forall i : A$ und $\exists i : A$:

$$\sigma(I(\forall i : A)) = \begin{cases} \textbf{true}, & \text{falls für alle } n \in \mathbb{N} \text{ gilt: } \sigma(I[n/i](A)) = \textbf{true}. \\ \textbf{false} & \text{sonst} \end{cases}$$

Dabei ist $I[n/i]$ definiert durch $I[n/i](i) = n$ und $I[n/i](j) = I(j)$ für $j \neq i$.

Analog wird $\sigma(I(\exists i : A))$ definiert.

Wir haben zwei Arten eingeführt, n für i einzusetzen:

In einem erweiterten arithmetischen Ausdruck kann die freie Variable i durch n ersetzt werden:

$$a[n/i]$$

In einer Interpretation $I : \textbf{Intvar} \to \mathbb{N}$ kann der Wert von i durch n ersetzt werden:

$$I[n/i] : \textbf{Intvar} \to \mathbb{N}$$

Es gilt dabei

$$\sigma(I[n/i](a)) = \sigma(I(a[n/i]))$$

Beweis: Strukturelle Induktion über die Erzeugung von a.

Auch auf der Ebene der Formeln in **Assn** können wir die Einsetzung von n für i definieren; für $A \in \mathbf{Assn}$ erhalten wir

$$A[n/i] \in \mathbf{Assn}$$

Dabei werden nur freie Vorkommen von i ersetzt.

Es gilt dann:

$$(\sigma \models^I \forall i : A) \Leftrightarrow (\sigma \models^I A[n/i] \text{ für alle } n \in \mathbb{N})$$

Tatsächlich kann man sogar zeigen:

$$\sigma(I[n/i](A)) = \sigma(I(A[n/i]))$$

für alle $\sigma \in \Sigma$, $i \in \mathbf{Intvar}$, $n \in \mathbb{N}$, $A \in \mathbf{Assn}$, $I : \mathbf{Intvar} \to \mathbb{N}$.

Wir wollen nicht ständig unterscheiden zwischen Definiertheit und Nichtdefiniertheit.

Daher betrachten wir wieder Funktionen von $\Sigma_\perp \to \Sigma_\perp$, die $\perp$ auf $\perp$ abbilden, anstelle der partiellen Funktionen $\Sigma \to_p \Sigma$.

Konvention: $\perp \models^I A$ ist wahr für alle Interpretationen I und alle Zusicherungen A, d.h. $\perp(I(A)) = \textbf{true}$.

Insbesondere gilt also auch $\perp \models \textbf{false}$.

Gültigkeit der partiellen Korrektheitsaussage

Für $\sigma \in \Sigma_\perp$ und $I : \mathbf{Intvar} \to \mathbb{N}$ ist die Aussage $\{A\}c\{B\}$ mit $A, B \in \mathbf{Assn}$ und $c \in \mathbf{Cmd}$ wahr, falls gilt:

$$(\sigma \models^I A) \;\Rightarrow\; (\mathcal{C}[\![c]\!]\sigma \models^I B)$$

Wir schreiben dann $\sigma \models^I \{A\}c\{B\}$.

Ferner schreiben wir $\models^I \{A\}c\{B\}$ falls $\forall \sigma \in \Sigma_\perp : \sigma \models^I \{A\}c\{B\}$.

Schließlich schreiben wir $\models \{A\}c\{B\}$ falls $\models^I \{A\}c\{B\}$ für alle $I : \mathbf{Intvar} \to \mathbb{N}$ gilt.

In diesem Fall sagen wir, dass die partielle Korrektheitsaussage $\{A\}c\{B\}$ gültig ist.

Beispiele:

1. $\{i < X\}\ X := X - 1\ \{i \leq X\}$ ist gültig.
2. Für alle $A, B \in$ **Assn** und $w =$ **while true do skip od** gilt

$$\models \{A\} w \{B\}.$$

3. Die Aussage $\{X = 1\}\, p\, \{Y = 5050\}$ ist gültig, wenn p das folgende Programm ist:

```
Y:=0;
while X ≤ 100 do
      Y := Y+X;
      X := X+1;
od
```

5.2 Hoare-Logik

Die Hoare-Logik definiert eine Menge ableitbarer Aussagen der Form $\{A\}c\{B\}$ für $A, B \in \mathbf{Assn}$ und $c \in \mathbf{Cmd}$ durch folgende Regeln:

1. **skip**:

$$\frac{}{\{A\}\ \mathbf{skip}\ \{A\}}$$

für alle $A \in \mathbf{Assn}$.

2. Zuweisung:

$$\frac{}{\{A[a/X]\}\ X := a\ \{A\}}$$

für alle $A \in \mathbf{Assn}$, $a \in \mathbf{Aexp}$, $X \in \mathbf{Loc}$.

3. Sequenz:

$$\frac{\{A\}\ c_1\ \{C\} \quad \{C\}\ c_2\ \{B\}}{\{A\}\ c_1; c_2\ \{B\}}$$

für alle $A, B, C \in \mathbf{Assn}$, $c_1, c_2 \in \mathbf{Cmd}$.

4. **if**:

$$\frac{\{A \wedge b\}\ c_1\ \{B\} \quad \{A \wedge \neg b\}\ c_2\ \{B\}}{\{A\}\ \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2\ \textbf{fi}\ \{B\}}$$

für alle $A, B \in \textbf{Assn}$, $b \in \textbf{Bexp}$, $c_1, c_2 \in \textbf{Cmd}$.

5. **while**:

$$\frac{\{A \wedge b\}\ c\ \{A\}}{\{A\}\ \textbf{while}\ b\ \textbf{do}\ c\ \textbf{od}\ \{A \wedge \neg b\}}$$

für alle $A \in \textbf{Assn}$, $b \in \textbf{Bexp}$, $c \in \textbf{Cmd}$.

6. Konsequenz:

$$\frac{\models (A \Rightarrow A') \quad \{A'\}\ c\ \{B'\} \quad \models (B' \Rightarrow B)}{\{A\}\ c\ \{B\}}$$

für alle $A, A', B, B' \in \textbf{Assn}$, $c \in \textbf{Cmd}$.

Definition

Wir schreiben $\vdash \{A\}c\{B\}$, falls sich die Aussage $\{A\}c\{B\}$ aus den Regeln der Hoare-Logik ableiten lässt.

Beispiel: Für $w \equiv$ **while true do skip od** gilt $\vdash \{\textbf{true}\}w\{\textbf{false}\}$.

Beweis:

$\vdash \{\textbf{true}\}\textbf{skip}\{\textbf{true}\}$

Daher $\vdash \{\textbf{true} \wedge \textbf{true}\}\textbf{skip}\{\textbf{true}\}$

Weiter $\vdash \{\textbf{true}\}w\{\textbf{true} \wedge \textbf{false}\}$

Schließlich $\vdash \{\textbf{true}\}w\{\textbf{false}\}$

Weiteres Beispiel: $\vdash \{X = 3\}\ X := 8 + X\ \{X = 11\}$

Denn aus der Zuweisungsregel folgt

$$\vdash \{(X = 11)[8 + X/X]\}\ X := 8 + X\ \{X = 11\}$$

Also $\vdash \{8 + X = 11\}\ X := 8 + X\ \{X = 11\}$ und mit der Konsequenzregel

$$\vdash \{X = 3\}\ X := 8 + X\ \{X = 11\}.$$

Satz 12

Die Regeln der Hoare-Logik erzeugen nur korrekte Aussagen, d.h.

$$\vdash \{A\}c\{B\} \;\Rightarrow\; \models \{A\}c\{B\}$$

Beweis: Es sei $\{A\}c\{B\}$ herleitbar: $\vdash \{A\}c\{B\}$.

Hieraus folgt leicht $\vdash \{I(A)\}c\{I(B)\}$ für alle I : **Intvar** $\to \mathbb{N}$
(in einer Ableitung kann überall i durch $I(i)$ ersetzt werden).

Wir zeigen, dass hieraus $\models \{I(A)\}c\{I(B)\}$ folgt.

Da dies für alle I gilt, folgt $\models \{A\}c\{B\}$.

Also bleibt zu zeigen:

Seien $A, B \in \mathbf{Assn}$ ohne freie Variablen, und sei $\{A\}c\{B\}$ herleitbar. Dann gilt $\models \{A\}c\{B\}$.

Unter den gemachten Voraussetzungen ist also nachzuprüfen, dass für alle $\sigma \in \Sigma_\perp$ mit $\sigma' = \mathcal{C}[\![c]\!]\sigma$ gilt

$$\sigma(A) = \mathbf{true} \;\Rightarrow\; \sigma'(B) = \mathbf{true}.$$

1. Fall: $\{A\}c\{B\}$ ist durch die **skip**-Regel entstanden, d.h. $c = \mathbf{skip}$.

Dann ist $\sigma' = \sigma$ und $B = A$, also ist die Behauptung trivial.

2. Fall : $\{A\}c\{B\}$ ist durch die Zuweisungs-Regel entstanden, d.h. $c = (X := a)$ für $X \in \mathbf{Loc}$ und $a \in \mathbf{Aexp}$.

Dann ist $A = B[a/X]$, und es gilt $\sigma' = \mathcal{C}[\![c]\!]\sigma = \sigma[n/X]$ mit $n = \sigma(a)$.

Offensichtlich gilt $\sigma(A) = \sigma(B[a/X]) = \sigma(B[n/X]) = \sigma[n/X](B)$.
(Beweis durch Induktion über die Erzeugung von B).

Also gilt $\sigma(A) = \mathbf{true} \;\Leftrightarrow\; \sigma'(B) = \sigma[n/X](B) = \mathbf{true}$.

Insbesondere: $\sigma(A) = \mathbf{true} \;\Rightarrow\; \sigma'(B) = \mathbf{true}$.

Also haben wir die Gültigkeit der Zusicherung $\{A\}c\{B\}$ gezeigt.

3. Fall: $\{A\}c\{B\}$ ist durch die Sequenz-Regel entstanden, d.h. $c = c_1; c_2$.

Dann gibt es also eine Zusicherung C ohne freie Variablen (Beweis: Übung!) so, dass $\{A\}c_1\{C\}$ und $\{C\}c_2\{B\}$ herleitbar sind.

Induktiv folgt:

- $\sigma(A) = \textbf{true} \;\Rightarrow\; \mathcal{C}[\![c_1]\!]\sigma(C) = \textbf{true}$ und
- $\sigma'(C) = \textbf{true} \;\Rightarrow\; \mathcal{C}[\![c_2]\!]\sigma'(B) = \textbf{true}$.

Wir wählen $\sigma' = \mathcal{C}[\![c_1]\!]\sigma$ und $\sigma' = \mathcal{C}[\![c_2]\!]\sigma'$.

Dann gilt:

$$\sigma'' = \mathcal{C}[\![c_2]\!]\sigma' = \mathcal{C}[\![c_2]\!] \circ \mathcal{C}[\![c_1]\!]\sigma = \mathcal{C}[\![c_1; c_2]\!]\sigma$$

Also erhalten wir: $\sigma(A) = \textbf{true} \;\Rightarrow\; \sigma'(C) = \textbf{true} \;\Rightarrow\; \sigma''(B) = \textbf{true}$.

Die zeigt die Gültigkeit von $\{A\}c\{B\}$.

4. Fall: $\{A\}c\{B\}$ ist durch die **if**-Regel enstanden, d.h. $c = \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ fi}$.

Also sind $\{A \wedge b\}c_1\{B\}$ und $\{A \wedge \neg b\}c_2\{B\}$ ableitbar.

Induktiv folgt:

- $\sigma(A \wedge b) = \textbf{true} \;\Rightarrow\; \mathcal{C}[\![c_1]\!]\sigma(B) = \textbf{true}$ und
- $\sigma(A \wedge \neg b) = \textbf{true} \;\Rightarrow\; \mathcal{C}[\![c_2]\!]\sigma(B) = \textbf{true}$.

Sei $\sigma(b)$ wahr. Dann gilt $\mathcal{C}[\![c]\!]\sigma = \mathcal{C}[\![c_1]\!]\sigma$, und daher

$$\sigma(A) = \textbf{true} \;\Rightarrow\; \mathcal{C}[\![c]\!]\sigma(B) = \textbf{true}.$$

Sei $\sigma(b)$ falsch. Dann gilt $\mathcal{C}[\![c]\!]\sigma = \mathcal{C}[\![c_2]\!]\sigma$, und daher

$$\sigma(A) = \textbf{true} \;\Rightarrow\; \mathcal{C}[\![c]\!]\sigma(B) = \textbf{true}.$$

Diese Implikation gilt also in jedem Fall.

5. Fall: $\{A\}c\{B\}$ ist durch die **while**-Regel enstanden, d.h. $c = \textbf{while } b \textbf{ do } c_1 \textbf{ od}$.

Dann muss $\{A \wedge b\}c_1\{A\}$ herleitbar sein, und es gilt $B = A \wedge \neg b$.

Nach der denotationellen Semantik der while-Schleife gilt

$$\sigma' = \mathcal{C}[\![c]\!]\sigma = \text{fix}(\Gamma_{b,c_1})(\sigma)$$

mit $\Gamma_{b,c_1}(f) = \text{cond}(\mathcal{B}[\![b]\!], f \circ \mathcal{C}[\![c_1]\!], \text{id}_\Sigma)$ für $f \in \Sigma_\perp \to \Sigma_\perp$.

Sei $g_n = \Gamma^n_{b,c_1}(\perp)$.

Beachte: $g_0 \sqsubseteq g_1 \sqsubseteq g_2 \sqsubseteq g_3 \sqsubseteq \cdots$.

Den Fixpunkt erhalten wir als Supremum der Menge $\{g_n \mid n \geq 0\}$.

Behauptung: Für alle $n \geq 0$ und alle $\sigma, \sigma' \in \Sigma_\perp$ gilt:

$$g_n(\sigma) = \sigma' \text{ und } \sigma(A) = \textbf{true} \;\Rightarrow\; \sigma'(A \wedge \neg b) = \textbf{true}.$$

Beweis der Behauptung: Induktion über $n \geq 0$

Sei $n = 0$.

Es gilt $g_0(\bot) = \mathrm{id}_\Sigma(\bot) = \bot$, d.h. $\sigma' = \bot$ und damit $\sigma'(A \wedge \neg b) = \textbf{true}$.

Sei $n = 1$.

Dann gilt

$$g_1(\sigma) = \begin{cases} \sigma & \text{falls } \sigma(b) = \textbf{false} \\ \bot & \text{falls } \sigma(b) = \textbf{true}. \end{cases}$$

Sei nun $g_1(\sigma) = \sigma'$.

Falls $\sigma(b) = \textbf{true}$ gilt, gilt also $\sigma' = \bot$ und damit $\sigma'(A \wedge \neg b) = \textbf{true}$.

Falls $\sigma(b) = \textbf{false}$ gilt, gilt $\sigma' = \sigma$ und damit:

$$\sigma(A) = \textbf{true} \;\Rightarrow\; \sigma'(A \wedge \neg b) = \textbf{true}$$

Nun sei $n \geq 1$.

Sei $\sigma' = g_{n+1}(\sigma)$.

Falls $\sigma' = \bot$ gilt $\sigma'(A \wedge \neg b) = \textbf{true}$ und wir sind fertig.

Gelte nun also $\sigma \neq \bot \neq \sigma'$.

Beachte: $g_{n+1} = \Gamma^{n+1}_{b,c_1}(\bot) = \Gamma^{n}_{b,c_1}(g_1) = \Gamma_{b,c_1}(g_n)$.

Fall A: $\sigma(b) = \textbf{false}$.

Dann gilt $g_1(\sigma) = \sigma$ und folglich auch $\sigma' = g_{n+1}(\sigma) = \sigma$.

Also: $\sigma(A) = \textbf{true} \;\Rightarrow\; \sigma'(A \wedge \neg b) = \textbf{true}$.

Fall B: $\sigma(b) = \textbf{true}$.

Es folgt

$$\sigma' = g_{n+1}(\sigma) = \Gamma_{b,c_1}(g_n)(\sigma) = g_n \circ \mathcal{C}[\![c_1]\!]\sigma = g_n(\sigma'')$$

mit $\sigma'' = \mathcal{C}[\![c_1]\!]\sigma$.

Da $\{A \wedge b\} c_1 \{A\}$ herleitbar ist, gilt nach Induktion

$$\sigma(A \wedge b) = \textbf{true} \;\Rightarrow\; \sigma''(A) = \textbf{true}.$$

Damit erhalten wir (mit Induktion für $g_n(\sigma'') = \sigma'$):

$$\begin{aligned} \sigma(A) = \textbf{true} \quad &\Rightarrow \quad \sigma''(A) = \textbf{true} \\ &\Rightarrow \quad \sigma'(A \wedge \neg b) = \textbf{true} \end{aligned}$$

Damit ist die Behauptung gezeigt.

Wir wissen also: Wenn $\mathcal{C}[\![c]\!]\sigma \neq \bot$, dann folgt aus $\sigma(A) = \textbf{true}$ auch $\mathcal{C}[\![c]\!]\sigma(A \wedge \neg b) = \textbf{true}$.

Wenn aber $\mathcal{C}[\![c]\!]\sigma = \bot$ nicht definiert ist, dann ist die Zusicherung $\{A\}c\{B\}$ für dieses σ ohnehin erfüllt.

Das komplettiert den Beweis für die while-Schleife.

6. Fall: $\{A\}c\{B\}$ ist durch die Konsequenz-Regel enstanden.

Es seien also $A \Rightarrow A'$ und $B' \Rightarrow B$ wahr (unter allen Belegungen $\sigma \in \Sigma$) und $\{A'\}c\{B'\}$ herleitbar.

Nach Induktionsannahme ist $\{A'\}c\{B'\}$ gültig (für alle σ).

Dann ist $\{A\}c\{B\}$ auch gültig, denn es gilt für alle $\sigma \in \Sigma$:

$$\begin{aligned} \sigma(A) = \textbf{true} \quad &\Rightarrow \quad \sigma(A') = \textbf{true} \\ &\Rightarrow \quad \mathcal{C}[\![c]\!]\sigma(B') = \textbf{true} \\ &\Rightarrow \quad \mathcal{C}[\![c]\!]\sigma(B) = \textbf{true}. \end{aligned}$$

Beispiele:

1. Das Plateau-Problem: Eingabe sei ein Feld

$$a[1..n], \text{ sortiert.}$$

Als Ausgabe wollen wir die Zahl p berechnen, wobei

$$p = \max\{q \mid \exists i : \ a[i] = a[i+1] = \cdots = a[i+q-1]\}$$

Beispiele:

1. Das Plateau-Problem: Eingabe sei ein Feld

$$a[1..n], \text{ sortiert.}$$

Als Ausgabe wollen wir die Zahl p berechnen, wobei

$$p = \max\{q \mid \exists i : \ a[i] = a[i+1] = \cdots = a[i+q-1]\}$$

Lösung:

Beispiele:

1. Das Plateau-Problem: Eingabe sei ein Feld

$$a[1..n], \text{ sortiert.}$$

Als Ausgabe wollen wir die Zahl p berechnen, wobei

$$p = \max\{q \mid \exists i : \ a[i] = a[i+1] = \cdots = a[i+q-1]\}$$

Lösung:

```
plateau ≡
j := 1;
p := 1;
while j<n do
      j := j+1;
      if a[j] = a[j-p] then
           p := p+1
      else skip
      fi
od
```

Beweis der Korrektheit:
Wir formulieren die Bedingung $mp(j, p)$:

Auf dem Teilfeld `a[1..j]` hat das längste Plateau die Länge p.

Wir wollen die Herleitbarkeit von

$$\{\textbf{true}\}\ \texttt{plateau}\ \{mp(n, p)\}$$

zeigen. Als Vorbedingung verwenden wir nicht $\{\textbf{true}\}$, sondern

$$V = (mp(1, 1) \text{ und „a ist sortiert“}).$$

Es gilt offenbar, dass V erfüllt ist (unter den Voraussetzungen $n \geq 1$ und a ist sortiert).

Durch zweimaliges Anwenden der Zuweisungsregel und einmaliges Anwenden der Sequenz-Regel erhalten wir:

$\{mp(1,1),\ \texttt{a sortiert}\}\ \texttt{j} := 1; \texttt{p} := 1$

$\{mp(j,p),\ \texttt{a sortiert}\}.$

Mit nochmaliger Verwendung der Sequenz-Regel müssen wir nur noch zeigen, dass

$$\{mp(j,p),\ \texttt{a sortiert}\}\ w\ \{mp(n,p)\}$$

für $w =$ while-Schleife aus dem Programm `plateau` herleitbar ist. Von jetzt an kürzen wir die Bedingung „a sortiert“ ab durch A. Sicher sind die beiden folgenden Aussagen herleitbar:

$$\{mp(j,p+1) \wedge A\}\ \texttt{p} := \texttt{p} + 1\ \{mp(j,p) \wedge A\}$$

und

$$\{mp(j,p) \wedge A\}\ \texttt{skip}\ \{mp(j,p) \wedge A\}.$$

Damit behaupten wir

$$\{mp(j-1,p) \wedge A\}\ \texttt{if...fi}\ \{mp(j,p) \wedge A\}.$$

Denn: Wir unterscheiden zwei Fälle.

1. Fall: `a[j]=a[j-p]` (d.h. b wahr):

$$\{mp(j-1,p) \wedge A \wedge b\} \Rightarrow \{mp(j,p+1) \wedge A \wedge b\}.$$

Also folgt aus

$$\{mp(j,p+1) \wedge A \wedge b\}\ \mathtt{p} := \mathtt{p}+1\ \{mp(j,p) \wedge A \wedge b\}$$

mit der Konsequenzregel

$$\{mp(j-1,p) \wedge A \wedge b\}\ \mathtt{p} := \mathtt{p}+1\ \{mp(j,p) \wedge A\}.$$

Denn: Wir unterscheiden zwei Fälle.

1. Fall: `a[j]=a[j-p]` (d.h. b wahr):

$$\{mp(j-1,p) \wedge A \wedge b\} \Rightarrow \{mp(j,p+1) \wedge A \wedge b\}.$$

Also folgt aus

$$\{mp(j,p+1) \wedge A \wedge b\}\ \texttt{p} := \texttt{p} + 1\ \{mp(j,p) \wedge A \wedge b\}$$

mit der Konsequenzregel

$$\{mp(j-1,p) \wedge A \wedge b\}\ \texttt{p} := \texttt{p} + 1\ \{mp(j,p) \wedge A\}.$$

2. Fall: `a[j]` $\neq$ `a[j-p]` (d.h. b falsch):

Es folgt ähnlich

$$\{mp(j-1,p) \wedge A \wedge \neg b\}\ \texttt{skip}\ \{mp(j,p) \wedge A\}.$$

Denn: Wir unterscheiden zwei Fälle.

1. Fall: `a[j]=a[j-p]` (d.h. b wahr):

$$\{mp(j-1,p) \wedge A \wedge b\} \Rightarrow \{mp(j,p+1) \wedge A \wedge b\}.$$

Also folgt aus

$$\{mp(j,p+1) \wedge A \wedge b\}\ \texttt{p} := \texttt{p} + 1\ \{mp(j,p) \wedge A \wedge b\}$$

mit der Konsequenzregel

$$\{mp(j-1,p) \wedge A \wedge b\}\ \texttt{p} := \texttt{p} + 1\ \{mp(j,p) \wedge A\}.$$

2. Fall: `a[j]` $\neq$ `a[j-p]` (d.h. b falsch):

Es folgt ähnlich

$$\{mp(j-1,p) \wedge A \wedge \neg b\}\ \texttt{skip}\ \{mp(j,p) \wedge A\}.$$

In jedem Fall erhalten wir

$$\{mp(j-1,p) \wedge A\}\ \texttt{if...fi}\ \{mp(j,p) \wedge A\}.$$

Also mit der Sequenzregel:

$$\{mp(j,p) \wedge A\}\texttt{j} := \texttt{j} + 1;\ \texttt{if...fi}\ \{mp(j,p) \wedge A\}.$$

Also erhalten wir für die gesamte while-Schleife w die Herleitbarkeit von

$$\{mp(j,p) \wedge A\}\ w\ \{mp(j,p) \wedge A \wedge (j = n)\},$$

und daher mit der Sequenzregel für das gesamte Plateau-Programm:

$$\{\textbf{true}\}\ \texttt{plateau}\ \{mp(j,p) \wedge A \wedge (j = n)\}$$

und mit der Konsequenzregel

$$\{\textbf{true}\}\ \texttt{plateau}\ \{mp(n,p)\},$$

wie gewünscht.

2. Potenzieren:

```
Z := 1;
while ¬(Y=0) do
      while even(Y) do
            X := X·X;
            Y := Y/2
      od;
      Z := Z·X;
      Y := Y-1
od
```

2. Potenzieren:

```
Z := 1;
while ¬(Y=0) do
      while even(Y) do
            X := X·X;
            Y := Y/2
      od;
      Z := Z·X;
      Y := Y-1
od
```

Behauptung: Für obiges Programm c gilt $\{A\}\ c\ \{B\}$, wobei

$$A = \{X = m \ \wedge \ Y = n\},$$

$$B = \{Z = m^n\}.$$

2. Potenzieren:

```
Z := 1;
while ¬(Y=0) do
      while even(Y) do
            X := X·X;
            Y := Y/2
      od;
      Z := Z·X;
      Y := Y-1
od
```

Behauptung: Für obiges Programm c gilt $\{A\}\ c\ \{B\}$, wobei

$$A = \{X = m \ \wedge \ Y = n\},$$

$$B = \{Z = m^n\}.$$

(Achtung: B ist so nicht in **Assn** definiert!)

Beweis: Sei w die äußere while-Schleife in c, d.h.

$$c \;=\; (Z := 1;\; w).$$

Wir zeigen

$$\{X = m \;\wedge\; Y = n\}\; Z := 1\; \{X = m \;\wedge\; Y = n \;\wedge\; Z = 1\}$$

und

$$\{X = m \;\wedge\; Y = n \;\wedge\; Z = 1\}\; w\; \{Z = m^n\},$$

indem wir zeigen, dass

$$\{Z \cdot X^Y \;=\; m^n\}$$

eine Invariante für beide while-Schleifen ist.
Die Termination des Programms ist klar. Also ist das Programm korrekt.

Zur Vollständigkeit: Unser Wunsch wäre ein Algorithmus, der für gegebenes Programm $c \in \mathbf{Cmd}$ und gegebene Zusicherungen $A, B \in \mathbf{Assn}$ die Frage der Gültigkeit von $\{A\}\ c\ \{B\}$ entscheidet.
Etwas bescheidener: Aufzählung aller gültigen Aussagen der Form $\{A\}\ c\ \{B\}$.
Beides ist unmöglich.
Den Beweis dieser Behauptung kann man auf zwei Arten führen.

1. Betrachte die Aussage

$$\{\textbf{true}\}\ \textbf{skip}\ \{B\}.$$

Diese ist wahr genau dann, wenn $B \in \textbf{Assn}$ eine wahre Aussage ist. Ein Verfahren zur Entscheidung oder Aufzählung aller gültigen Aussagen der Hoare-Logik ergäbe also insbesondere ein Aufzählungsverfahren für alle wahren Aussagen der Arithmetik.

1. Betrachte die Aussage

$$\{\textbf{true}\}\ \textbf{skip}\ \{B\}.$$

Diese ist wahr genau dann, wenn $B \in \textbf{Assn}$ eine wahre Aussage ist. Ein Verfahren zur Entscheidung oder Aufzählung aller gültigen Aussagen der Hoare-Logik ergäbe also insbesondere ein Aufzählungsverfahren für alle wahren Aussagen der Arithmetik.
Nach dem

Gödelschen Unvollständigkeitssatz

existiert ein solches Verfahren nicht!
2. Betrachte die Aussage

$$\{\textbf{true}\}\ c\ \{\textbf{false}\}.$$

Diese ist wahr genau dann, wenn c auf allen Speicherzustände nicht terminiert. Gäbe es hierfür ein Aufzählungsverfahren, so wäre das Halteproblem entscheidbar.

Wir wissen aber, dass das nicht der Fall ist, also ist die Menge

$$\{(A, B, c) \mid \quad \models \{A\}\ c\ \{B\}\ \}$$

nicht rekursiv aufzählbar.

Wir wissen aber, dass das nicht der Fall ist, also ist die Menge

$$\{(A, B, c) \mid \quad \models \{A\}\ c\ \{B\}\ \}$$

nicht rekursiv aufzählbar.
Wir beobachten nun allerdings, dass die Konsequenzregel einen nicht-effektiven Schritt erlaubt:

$$\frac{\{\mathbf{true}\}\ \mathbf{skip}\ \{\mathbf{true}\} \quad \mathbf{true} \Rightarrow B}{\{\mathbf{true}\}\ \mathbf{skip}\ \{B\}}.$$

Die Regel ist „anwendbar“, wenn B wahr ist. Also sind alle Regeln der Form

$$\{\mathbf{true}\}\ \mathbf{skip}\ \{B\}$$

für wahre B herleitbar, obwohl wir diese Herleitung im allgemeinen nicht finden können.

Daher ist es kein Widerspruch, wenn wir nun zeigen, dass aus der Gültigkeit

$$\models \{A\}\ c\ \{B\}$$

die Herleitbarkeit

$$\vdash \{A\}\ c\ \{B\}$$

folgt, d.h. eine Aussage ist genau dann gültig, wenn sie im Hoare-Kalkül herleitbar ist.
Die Tatsache, dass wir trotzdem keine formale Überprüfbarkeit der Gültigkeit von Aussagen $\{A\}\ c\ \{B\}$ bekommen, wird angedeutet, indem man von

relativer Vollständigkeit

spricht.

Schwächste Vorbedingungen

Schwächste Vorbedingungen

(englisch: weakest (liberal) precondition)

Schwächste Vorbedingungen
(englisch: weakest (liberal) precondition)
Wir wollen wissen, welche $\sigma \in \Sigma_\perp$ dazu führen, dass unter Interpretation I nach Ausführung von c die Zusicherung B erfüllt ist:

$$wp^I[\![c, B]\!] = \{\sigma \in \Sigma_\perp \mid \mathcal{C}[\![c]\!]\sigma \models^I B\}.$$

Also ist $wp^I[\![c, B]\!]$ eine Teilmenge von $\Sigma_\perp$.
Beachte: Wenn c auf Speicherzustand σ nicht terminiert, gilt auch $\sigma \in wp^I[\![c, B]\!]$.

Wir können auch direkt einer Zusicherung $A \in \mathbf{Assn}$ unter einer Interpretation I eine Menge von Speicherzustände zuordnen:

$$A^I = \{\sigma \in \Sigma_\perp \mid \sigma \models^I A\}.$$

Nun gilt offenbar:

$$\models^I \{A\}\ c\ \{B\} \ \Leftrightarrow\ A^I \subseteq wp^I[\![c, B]\!].$$

Wir können auch direkt einer Zusicherung $A \in \mathbf{Assn}$ unter einer Interpretation I eine Menge von Speicherzustände zuordnen:

$$A^I = \{\sigma \in \Sigma_\perp \mid \sigma \models^I A\}.$$

Nun gilt offenbar:

$$\models^I \{A\}\ c\ \{B\} \ \Leftrightarrow\ A^I \subseteq wp^I[\![c, B]\!].$$

Beweis: Es gelte $\models^I \{A\}\ c\ \{B\}$, und sei $\sigma \in A^I$, d.h. $\sigma \models^I A$. Wegen $\models^I \{A\}\ c\ \{B\}$ folgt $\mathcal{C}[\![c]\!]\sigma \models^I B$, d.h. $\sigma \in wp^I[\![c, B]\!]$.
Die Umkehrung ist ähnlich zu zeigen.

Angenommen $A_0 \in \textbf{Assn}$ ist eine Zusicherung mit

$$\forall I : \textbf{Intvar} \to \mathbb{N} : A_0^I = wp^I[\![c, B]\!],$$

d.h.

$$\forall I : \textbf{Intvar} \to \mathbb{N}\, \forall \sigma \in \Sigma_\perp : \sigma \models^I A_0 \Leftrightarrow \mathcal{C}[\![c]\!]\sigma \models^I B.$$

Wir erhalten dann für alle $I : \textbf{Intvar} \to \mathbb{N}$ und alle $A \in \textbf{Assn}$:

$$\begin{aligned}
& \models^I \{A\}\ c\ \{B\} \\
\Leftrightarrow\ & \forall \sigma \in \Sigma_\perp : \sigma \models^I A \ \Rightarrow\ \mathcal{C}[\![c]\!]\sigma \models^I B \\
\Leftrightarrow\ & \forall \sigma \in \Sigma_\perp : \sigma \models^I A \ \Rightarrow\ \sigma \models^I A_0 \\
\Leftrightarrow\ & \models^I (A \Rightarrow A_0)
\end{aligned}$$

also

$$(\models \{A\}\ c\ \{B\}) \Leftrightarrow (\models (A \Rightarrow A_0)).$$

Daher nennen wir A_0 eine schwächste Vorbedingung für das Paar (c, B).
Wir wollen nun zeigen, dass für alle $c \in \textbf{Cmd}$ und alle $B \in \textbf{Assn}$ ein $A \in \textbf{Assn}$ existiert so, dass A eine schwächste Vorbedingung für (c, B) ist.

Lemma 5.2.3 Das Prädikat $\beta\colon \mathbb{N}^4 \to \mathbb{B}$ sei definiert durch

$$\beta(a, b, i, x) \Leftrightarrow x = a \mod (1 + b(i + 1)).$$

Seien $k, n_0, \ldots, n_k \in \mathbb{N}$. Dann gibt es Zahlen $n, m \in \mathbb{N}$ so, dass für alle $j \in \{0, 1, \ldots, k\}$ und alle $x \in \mathbb{N}$ gilt:

$$\beta(n, m, j, x) \Leftrightarrow x = n_j.$$

Beweis: Setze

$$m = (\max\{k, n_0, n_1, \ldots, n_k\})!.$$

Zeige: $\mathrm{ggT}(1 + m(i + 1), 1 + m(j + 1)) = 1$ (für $0 \leq i < j \leq k$). Mit dem Chinesischen Restsatz folgt die Existenz einer Zahl n, für die $n = n_i \mod (1 + m(i + 1))$ für $i \in \{0, \ldots, k\}$ gilt.

Satz 5.2.4 Es sei $c \in \mathbf{Cmd}$ und $B \in \mathbf{Assn}$. Dann existiert eine Zusicherung $A \in \mathbf{Assn}$ mit

$$wp^I[\![c, B]\!] = A^I$$

für alle $I : \mathbf{Intvar} \to \mathbb{N}$.

Satz 5.2.4 Es sei $c \in \mathbf{Cmd}$ und $B \in \mathbf{Assn}$. Dann existiert eine Zusicherung $A \in \mathbf{Assn}$ mit

$$wp^I[\![c, B]\!] = A^I$$

für alle $I : \mathbf{Intvar} \to \mathbb{N}$.

Beweis: Wir müssen zeigen, dass es ein A gibt mit

$$\forall \sigma \in \Sigma_\perp : \ (\sigma \models^I A) \Leftrightarrow (\mathcal{C}[\![c]\!]\sigma \models^I B).$$

Wir führen den Beweis mittels struktureller Induktion über die Erzeugung von c.

a) $c = \textbf{skip}$:

Wegen $\mathcal{C}[\![\textbf{skip}]\!]\sigma = \sigma$ für alle $\sigma \in \Sigma_\perp$ gilt:

$$(\sigma \models^I B) \Leftrightarrow (\mathcal{C}[\![c]\!]\sigma \models^I B).$$

Wir wählen also $A = B$.

a) $c = \textbf{skip}$:

Wegen $\mathcal{C}[\![\textbf{skip}]\!]\sigma = \sigma$ für alle $\sigma \in \Sigma_\perp$ gilt:

$$(\sigma \models^I B) \Leftrightarrow (\mathcal{C}[\![c]\!]\sigma \models^I B).$$

Wir wählen also $A = B$.

b) $c = (X := a)$:

Es gilt $\mathcal{C}[\![c]\!]\sigma = \sigma[n/X]$, wobei $n = \mathcal{A}[\![a]\!]\sigma$, und

$$(\sigma \models^I B[a/X]) \Leftrightarrow (\mathcal{C}[\![c]\!]\sigma \models^I B).$$

Wir wählen daher $A = B[a/X]$.

c) $c = (c_1; c_2)$:

Es sei A' so gewählt, dass

$$\forall \sigma \in \Sigma_\perp : \ (\sigma \models^I A') \Leftrightarrow (\mathcal{C}[\![c_2]\!]\sigma \models^I B)$$

und A so, dass

$$\forall \sigma \in \Sigma_\perp : \ (\sigma \models^I A) \Leftrightarrow (\mathcal{C}[\![c_1]\!]\sigma \models^I A').$$

Damit gilt für alle $\sigma \in \Sigma_\perp$:

$$\begin{aligned} \sigma \models^I A \ &\Leftrightarrow \\ \mathcal{C}[\![c_1]\!]\sigma \models^I A' \ &\Leftrightarrow \\ \mathcal{C}[\![c_2]\!] \circ \mathcal{C}[\![c_1]\!]\sigma \models^I B \ &\Leftrightarrow \\ \mathcal{C}[\![c_1; c_2]\!]\sigma \models^I B \end{aligned}$$

Dieses A erfüllt also unsere Anforderungen.

d) $c = (\textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ fi})$:

Es sei A_1 so gewählt, dass

$$(\sigma \models' A_1) \Leftrightarrow (\mathcal{C}[\![c_1]\!]\sigma \models' B)$$

und A_2 so, dass

$$(\sigma \models' A_2) \Leftrightarrow (\mathcal{C}[\![c_2]\!]\sigma \models' B).$$

Wähle $A = (A_1 \ \wedge \ b) \vee (A_2 \ \wedge \ \neg b)$.
Sei $\sigma(b) = \textbf{true}$ (d.h. $\mathcal{B}[\![b]\!]\sigma = \textbf{true}$). Dann gilt:

$$(\sigma \models^I A)$$

genau dann, wenn

$$\sigma \models^I A_1 \ \wedge \ \sigma(b) = \textbf{true} \text{ oder } \sigma \models^I A_2 \ \wedge \ \sigma(b) = \textbf{false}$$

genau dann, wenn

$$\sigma \models^I A_1$$

genau dann, wenn

$$\mathcal{C}[\![c_1]\!]\sigma \models^I B$$

genau dann, wenn

$$\mathcal{C}[\![c]\!]\sigma \models^I B$$

Sei nun $\sigma(b) = \textbf{false}$. Dann erhalten wir ebenso

$$(\sigma \models^I A) \Leftrightarrow (\sigma \models^I A_2),$$

und daher

$$\begin{aligned}(\sigma \models^I A) &\Leftrightarrow (\mathcal{C}[\![c_2]\!]\sigma \models^I B)\\ &\Leftrightarrow (\mathcal{C}[\![c]\!]\sigma \models^I B).\end{aligned}$$

Also erfüllt in beiden Fällen A unsere Anforderungen.

e) $c = (\textbf{while } b \textbf{ do } c_1 \textbf{ od})$:

Es gilt

$$wp^I[\![c, B]\!] = \{\sigma \mid \mathcal{C}[\![c]\!]\sigma \models^I B\},$$

also

$$\sigma \in wp^I[\![c, B]\!]$$

genau dann, wenn $\mathcal{C}[\![c]\!]\sigma$ undefiniert ist oder es $k \geq 0$ und $\sigma_0, \ldots, \sigma_k \in \Sigma$ gibt mit

e) $c = (\textbf{while } b \textbf{ do } c_1 \textbf{ od})$:

Es gilt

$$wp^I[\![c, B]\!] = \{\sigma \mid \mathcal{C}[\![c]\!]\sigma \models^I B\},$$

also

$$\sigma \in wp^I[\![c, B]\!]$$

genau dann, wenn $\mathcal{C}[\![c]\!]\sigma$ undefiniert ist oder es $k \geq 0$ und $\sigma_0, \ldots, \sigma_k \in \Sigma$ gibt mit

1) $\sigma = \sigma_0$

e) $c = (\textbf{while } b \textbf{ do } c_1 \textbf{ od})$:

Es gilt

$$wp^I[\![c, B]\!] = \{\sigma \mid \mathcal{C}[\![c]\!]\sigma \models^I B\},$$

also

$$\sigma \in wp^I[\![c, B]\!]$$

genau dann, wenn $\mathcal{C}[\![c]\!]\sigma$ undefiniert ist oder es $k \geq 0$ und $\sigma_0, \ldots, \sigma_k \in \Sigma$ gibt mit

1) $\sigma = \sigma_0$
2) $\forall i \in \{0, \ldots, k-1\}$:

$$(\sigma_i \models^I b) \wedge (\mathcal{C}[\![c_1]\!]\sigma_i = \sigma_{i+1})$$

e) $c = (\textbf{while } b \textbf{ do } c_1 \textbf{ od})$:

Es gilt

$$wp^I[\![c, B]\!] = \{\sigma \mid \mathcal{C}[\![c]\!]\sigma \models^I B\},$$

also

$$\sigma \in wp^I[\![c, B]\!]$$

genau dann, wenn $\mathcal{C}[\![c]\!]\sigma$ undefiniert ist oder es $k \geq 0$ und $\sigma_0, \ldots, \sigma_k \in \Sigma$ gibt mit

1) $\sigma = \sigma_0$
2) $\forall i \in \{0, \ldots, k-1\}$:

$$(\sigma_i \models^I b) \wedge (\mathcal{C}[\![c_1]\!]\sigma_i = \sigma_{i+1})$$

3) $\sigma_k \models^I B \wedge \neg b$

Im Fall $\mathcal{C}[\![c]\!]\sigma$ undefiniert existiert entweder ein i so, dass $\mathcal{C}[\![c_1]\!]\sigma_i$ undefiniert ist, oder es gibt eine unendliche Kette $\sigma_0, \sigma_1, \sigma_2, \ldots$ mit

$$\sigma = \sigma_0, \quad \sigma_i \models^I b, \quad \mathcal{C}[\![c_1]\!]\sigma_i = \sigma_{i+1}.$$

Nun fassen wir beide Fälle zusammen:

Nun fassen wir beide Fälle zusammen:

$$\sigma \in wp^I[\![c, B]\!] \quad \Leftrightarrow$$
$$\forall k \;\; \forall \sigma_0, \ldots, \sigma_k \in \Sigma :$$
$$\Big[(\sigma = \sigma_0 \;\wedge\; \forall i \in \{0, \ldots, k-1\}$$
$$[\sigma_i \models^I b, \;\; \mathcal{C}[\![c_1]\!]\sigma_i = \sigma_{i+1}]\Big]$$
$$\Rightarrow \;\; [\sigma_k \models^I B \vee b]$$

Wir brauchen eine Beschreibung der Bedingung für $\sigma \in wp^I[\![c, B]\!]$ als Element von **Assn**. Das heißt, die Speicherzustände σ_i müssen aus dem Ausdruck eliminiert werden. Außerdem brauchen wir einen festen Ausdruck (ohne Konstruktionen wie $\forall \sigma_0, \ldots, \sigma_k$), denn k kann beliebig groß sein so, dass man die Pünktchen so nicht in einer einzigen Zusicherung reflektieren kann.

Seien $X_1, \ldots, X_l$ die in B oder c vorkommenden Programmvariablen.
Seien $s_1, \ldots, s_l$ so gewählt, dass für $i \in \{1, \ldots, l\}$ gilt:

$$\sigma(X_i) = s_i.$$

Wir schreiben $\tilde{X}$ für $X_1, \ldots, X_l$ und $\tilde{s}$ für $s_1, \ldots, s_l$. Dann folgt:

$$(\sigma \models^I A) \quad \Leftrightarrow \quad \models^I A[\tilde{s}/\tilde{X}].$$

Dabei steht $[\tilde{s}/\tilde{X}]$ für

$$[s_1/X_1][s_2/X_2] \ldots [s_l/X_l].$$

(Beweis der obigen Äquivalenz per Induktion über die Struktur von A.)

Im folgenden steht $\tilde{s_i}$ immer als Abkürzung für $s_{i,1}, \ldots, s_{i,l}$, wobei für alle $j \in \{1, \ldots, l\}$ gelte: $s_{i,j} = \sigma_i(X_j)$. Also beschreibt $\tilde{s_i}$ genau die Speicherzustand σ_i.

Es gilt nun $\sigma \in wp^I[\![c, B]\!]$ genau dann, wenn:

$$\begin{aligned}
&\forall k \;\; \forall \tilde{s_0}, \ldots, \tilde{s_k} \in \mathbb{N}^l \\
&\quad \Big[\sigma \models^I (\tilde{X} = \tilde{s_0}) \quad \wedge \quad \forall i \in \{0, \ldots, k-1\} \\
&\qquad \models^I b[\tilde{s_i}/\tilde{X}] \quad \wedge \quad \models^I (A_1 \wedge \neg A_2)[\tilde{s_i}/\tilde{X}]\Big] \\
&\quad \Rightarrow \quad \models^I (B \vee b)[\tilde{s_k}/\tilde{X}].
\end{aligned}$$

Dabei sei A_1 so gewählt, dass für alle $\tau \in \Sigma$ gilt

$$(\tau \models^I A_1) \quad \Leftrightarrow \quad (\mathcal{C}[\![c_1]\!]\tau \models^I (\tilde{s_{i+1}} = \tilde{X}))$$

und A_2 so, dass für alle $\tau \in \Sigma$ gilt

$$(\tau \models^I A_2) \quad \Leftrightarrow \quad (\mathcal{C}[\![c_1]\!]\tau \models^I \textbf{false})$$

Die $l \cdot (k+1)$ Zahlen $\tilde{s_0}, \ldots, \tilde{s_k}$ können wir gemäß Lemma 5.2.3 mit dem β-Prädikat in zwei Zahlen m, n verschlüsseln.
Das Endergebnis schreiben wir hier der Einfachheit halber nur für den Fall $l = 1$ auf. Es gilt dann, dass eine Speicherzustand σ zu $wp^l[\![c, B]\!]$ genau dann gehört, wenn unter σ die folgende Assertion wahr ist:

$$
\begin{aligned}
&\forall k \forall m \forall n : \\
&\quad \beta(n, m, 0, X) \ \wedge \\
&\quad \forall i \in \{0, \ldots, k-1\} \\
&\qquad \big[\forall y : \ \beta(n, m, i, y) \ \Rightarrow \ b[y/X]\big] \ \wedge \\
&\qquad \big[\forall x, y : \ (\beta(n, m, i, x) \ \wedge \ \beta(n, m, i+1, y)) \\
&\qquad\qquad \Rightarrow \ (A_1 \wedge \neg A_2)[x/X]\big] \\
&\quad \Rightarrow \ \big(\forall x : \ \beta(n, m, k, x) \ \Rightarrow \ (B \vee b)[x/X]\big)
\end{aligned}
$$

$$
\begin{array}{l}
\forall k \forall m \forall n : \\
\quad \beta(n, m, 0, X) \ \wedge \\
\quad \forall i \in \{0, \ldots, k-1\} \\
\qquad \big[\forall y : \ \beta(n, m, i, y) \ \Rightarrow \ b[y/X]\big] \ \wedge \\
\qquad \big[\forall x, y : \ (\beta(n, m, i, x) \ \wedge \ \beta(n, m, i+1, y)) \\
\qquad\qquad \Rightarrow \ (A_1 \wedge \neg A_2)[x/X]\big] \\
\quad \Rightarrow \ \big(\forall x : \ \beta(n, m, k, x) \ \Rightarrow \ (B \vee b)[x/X]\big)
\end{array}
$$

Dabei ist A_1 so gewählt, dass

$$(\tau \models^I A_1) \ \Leftrightarrow \ (\mathcal{C}[\![c_1]\!]\tau \models^I (X = y))$$

und A_2 so, dass

$$(\tau \models^I A_2) \ \Leftrightarrow \ (\mathcal{C}[\![c_1]\!]\tau \models^I \textbf{false})$$

Man beachte, dass das X in der Formel eine Programmvariable (stellvertretend für eine endliche Menge von Programmvariablen!) darstellt. Die Formel ist bei gegebenem Speicherzustand σ also so auszuwerten, dass X durch $\sigma(X)$ ersetzt wird.
Die logische Struktur ist dabei so zu verstehen, dass erstens $\beta(n, m, 0, X)$ gelten muss, das heißt also $\sigma(X) = n_0$ (was der Anfangsbedingung $\sigma = \sigma_0$ der ursprünglichen Darstellung entspricht), und dass zweitens aus der Gültigkeit der Zeilen 2 bis 5 (unter σ) die Gültigkeit der letzten Zeile folgen muss.
Die letzte Zeile bedeutet: Wenn $n_k = x$ gilt, dann muss die Bedingung $(B \vee b)[x/X]$ wahr sein, also anders ausgedrückt, im Endzustand soll die Zusicherung $B \vee b$ erfüllt sein.

Satz 5.2.4 besagt, dass die (genauer gesagt: eine) schwächste Vorbedingung für ein Paar (c, B) mit $c \in \mathbf{Cmd}$ und $B \in \mathbf{Assn}$ selbst als Zusicherung aus **Assn** formulierbar ist, insbesondere existiert also für alle Paare (c, B) eine Zusicherung $A \in \mathbf{Assn}$ mit

$$\models \{A\}c\{B\}$$

und

$$\models \{A'\}c\{B\} \;\Rightarrow\; (A' \Rightarrow A).$$

Wir wollen nun zeigen, dass für jede schwächste Vorbedingung A von (c, B) sogar gilt:

$$\vdash \{A\}c\{B\}.$$

Satz 5.2.5 Es sei $c \in \mathbf{Cmd}$ und $B \in \mathbf{Assn}$. Es sei ferner A eine schwächste Vorbedingung für (c, B). Dann gilt

$$\vdash \{A\}c\{B\}.$$

Satz 5.2.5 Es sei $c \in \textbf{Cmd}$ und $B \in \textbf{Assn}$. Es sei ferner A eine schwächste Vorbedingung für (c, B). Dann gilt

$$\vdash \{A\}c\{B\}.$$

Beweis: Wir brauchen jeweils nur für eine fest gewählte schwächste Vorbedingung A zu zeigen, dass $\vdash \{A\}c\{B\}$ gilt. Denn für jede andere schwächste Vorbedingung A' gilt dann $A \Leftrightarrow A'$, insbesonderen also $A' \Rightarrow A$. Aus der Konsequenzregel erhalten wir dann auch $\vdash \{A'\}c\{B\}$. Den Beweis führen wir durch Induktion über den Aufbau der Anweisung c.

Satz 5.2.5 Es sei $c \in \textbf{Cmd}$ und $B \in \textbf{Assn}$. Es sei ferner A eine schwächste Vorbedingung für (c, B). Dann gilt

$$\vdash \{A\}c\{B\}.$$

Beweis: Wir brauchen jeweils nur für eine fest gewählte schwächste Vorbedingung A zu zeigen, dass $\vdash \{A\}c\{B\}$ gilt. Denn für jede andere schwächste Vorbedingung A' gilt dann $A \Leftrightarrow A'$, insbesonderen also $A' \Rightarrow A$. Aus der Konsequenzregel erhalten wir dann auch $\vdash \{A'\}c\{B\}$. Den Beweis führen wir durch Induktion über den Aufbau der Anweisung c.
a) $c = \textbf{skip}$:

Satz 5.2.5 Es sei $c \in \textbf{Cmd}$ und $B \in \textbf{Assn}$. Es sei ferner A eine schwächste Vorbedingung für (c, B). Dann gilt

$$\vdash \{A\}c\{B\}.$$

Beweis: Wir brauchen jeweils nur für eine fest gewählte schwächste Vorbedingung A zu zeigen, dass $\vdash \{A\}c\{B\}$ gilt. Denn für jede andere schwächste Vorbedingung A' gilt dann $A \Leftrightarrow A'$, insbesonderen also $A' \Rightarrow A$. Aus der Konsequenzregel erhalten wir dann auch $\vdash \{A'\}c\{B\}$.
Den Beweis führen wir durch Induktion über den Aufbau der Anweisung c.
a) $c = \textbf{skip}$:
Aus dem Beweis von Satz 5.2.4 wissen wir, dass B eine schwächste Vorbedingung für $(\textbf{skip}, B)$ ist. Wir brauchen also nur die Herleitbarkeit von $\{B\}\textbf{skip}\{B\}$ zu zeigen. Das ist aber gerade die **skip**-Regel des Hoare-Kalküls.

b) $c = (X := a)$:

b) $c = (X := a)$:
Hier kennen wir aus dem Beweis von Satz 5.2.4 die schwächste Vorbedingung $B[a/X]$. Es genügt uns also $\vdash \{B[a/X]\}\ X := a\ \{B\}$. Das ist aber genau die Zuweisungsregel des Hoare-Kalküls.

b) $c = (X := a)$:
Hier kennen wir aus dem Beweis von Satz 5.2.4 die schwächste Vorbedingung $B[a/X]$. Es genügt uns also $\vdash \{B[a/X]\}\ X := a\ \{B\}$. Das ist aber genau die Zuweisungsregel des Hoare-Kalküls.
c) $c = (c_1; c_2)$:

b) $c = (X := a)$:
Hier kennen wir aus dem Beweis von Satz 5.2.4 die schwächste Vorbedingung $B[a/X]$. Es genügt uns also $\vdash \{B[a/X]\}\ X := a\ \{B\}$. Das ist aber genau die Zuweisungsregel des Hoare-Kalküls.
c) $c = (c_1; c_2)$:
Wir wählen wie im Teil c) des Beweises von Satz 5.2.4 die Zusicherungen A' und A, wobei A' schwächste Vorbedingung für (c_2, B) ist und A schwächste Vorbedingung für (c_1, A'). Dann ist A schwächste Vorbedingung für (c, B).
Nach Induktionsvoraussetzung gilt $\vdash \{A'\}\ c_2\ \{B\}$ und $\vdash \{A\}\ c_1\ \{A'\}$. Mit der Sequenzregel des Hoare-Kalküls erhalten wir $\vdash \{A\}\ c_1; c_2\ \{B\}$, also

$$\vdash \{A\}\ c\ \{B\}.$$

d) $c =$ **if** b **then** c_1 **else** c_2 **fi**:

d) $c =$ **if** b **then** c_1 **else** c_2 **fi**:
Wir wählen wie im Teil d) von 5.2.4 A_1 als schwächste Vorbedingung für (c_1, B) und A_2 als schwächste Vorbedingung für (c_2, B). Dann ist $A = (A_1 \wedge b) \vee (A_2 \wedge \neg b)$ schwächste Vorbedingung für (c, B). Es gilt

$$(A \wedge b) \;\Rightarrow\; A_1$$

und

$$(A \wedge \neg b) \;\Rightarrow\; A_2,$$

und nach Induktionsvoraussetzung $\vdash \{A_1\}\ c_1\ \{B\}$ und $\vdash \{A_2\}\ c_2\ \{B\}$.
Durch Anwendung der Konsequenzregel erhalten wir $\vdash \{A \wedge b\}\ c_1\ \{B\}$ und $\vdash \{A \wedge \neg b\}\ c_2\ \{B\}$.
Aus diesen beiden Aussagen ist schließlich mit der **if**-Regel des Hoare-Kalküls die Aussage

$$\vdash \{A\}\ c\ \{B\}$$

herleitbar.

e) $c =$ **while** b **do** c_1 **od**:

e) $c = \textbf{while } b \textbf{ do } c_1 \textbf{ od}$:
Hier können wir nicht direkt auf die bei Satz 5.2.4 konstruierte schwächste Vorbedingung zurückgreifen, da deren Definition nicht so handlich ist wie bei den Fällen a) bis d). Wir gehen daher einen anderen Weg:

Sei A eine schwächste Vorbedingung für (c, B). Wir wollen zuerst zeigen, dass $\{A \wedge b\}\ c_1\ \{A\}$ eine gültige Aussage ist:
Sei σ ein Speicherzustand, I eine Interpretation, und sei $\sigma \models^I A \wedge b$. Wir müssen zeigen, dass gilt $\mathcal{C}[\![c_1]\!]\sigma \models^I A$. Zunächst folgt aus $\sigma \models^I A \wedge b$ insbesondere $\sigma \models^I A$, und da A schwächste Vorbedingung für (c, B) ist, gilt auch $\mathcal{C}[\![c]\!]\sigma \models^I B$. Nun ist

$$\mathcal{C}[\![c]\!] = \mathcal{C}[\![\textbf{if } b \textbf{ then } c_1; c \textbf{ else skip fi}]\!],$$

da σ unter I auch b wahr macht, haben wir also $\mathcal{C}[\![c]\!]\sigma = \mathcal{C}[\![c_1; c]\!]\sigma$. Es gilt also $\mathcal{C}[\![c_1; c]\!]\sigma \models^I B$, d.h. $\mathcal{C}[\![c]\!](\mathcal{C}[\![c_1]\!]\sigma) \models^I B$, und daher $\mathcal{C}[\![c_1]\!]\sigma \models^I A$, wie gewünscht.

Als zweites wollen wir zeigen, dass $(A \wedge \neg b) \Rightarrow B$ eine gültige Aussage ist: Sei σ ein Speicherzustand, I eine Interpretation, und sei $\sigma \models^I A \wedge \neg b$. Wir müssen zeigen, dass gilt $\sigma \models^I B$. Aus $\sigma \models^I A$ folgt aber $\mathcal{C}[\![c]\!]\sigma \models^I B$. Mit $\sigma \models^I \neg b$ und der Darstellung

$$\mathcal{C}[\![c]\!] = \mathcal{C}[\![\textbf{if } b \textbf{ then } c_1; c \textbf{ else skip fi}]\!]$$

folgt $\mathcal{C}[\![c]\!]\sigma = \sigma$ und daher bereits $\sigma \models^I B$, wie gewünscht.

Als zweites wollen wir zeigen, dass $(A \wedge \neg b) \Rightarrow B$ eine gültige Aussage ist: Sei σ ein Speicherzustand, I eine Interpretation, und sei $\sigma \models^I A \wedge \neg b$. Wir müssen zeigen, dass gilt $\sigma \models^I B$. Aus $\sigma \models^I A$ folgt aber $\mathcal{C}[\![c]\!]\sigma \models^I B$. Mit $\sigma \models^I \neg b$ und der Darstellung

$$\mathcal{C}[\![c]\!] = \mathcal{C}[\![\textbf{if } b \textbf{ then } c_1; c \textbf{ else skip fi}]\!]$$

folgt $\mathcal{C}[\![c]\!]\sigma = \sigma$ und daher bereits $\sigma \models^I B$, wie gewünscht.
Nun haben wir also die Gültigkeit von $\{A \wedge b\}\ c_1\ \{A\}$ und $(A \wedge \neg b) \Rightarrow B$. Sei A' eine schwächste Vorbedingung für (c_1, A). Dann gilt $(A \wedge b) \Rightarrow A'$ und nach Induktionsvoraussetzung ist $\{A'\}\ c_1\ \{A\}$ herleitbar. Mit der Konsequenzregel ist also auch $\{A \wedge b\}\ c_1\ \{A\}$ herleitbar. Nach der **while**-Regel ist dann auch $\{A\}\ c\ \{A \wedge \neg b\}$ herleitbar. Durch nochmaliges Anwenden der Konsequenzregel erhalten wir $\vdash \{A\}\ c\ \{B\}$.

Als zweites wollen wir zeigen, dass $(A \wedge \neg b) \Rightarrow B$ eine gültige Aussage ist: Sei σ ein Speicherzustand, I eine Interpretation, und sei $\sigma \models^I A \wedge \neg b$. Wir müssen zeigen, dass gilt $\sigma \models^I B$. Aus $\sigma \models^I A$ folgt aber $\mathcal{C}[\![c]\!]\sigma \models^I B$. Mit $\sigma \models^I \neg b$ und der Darstellung

$$\mathcal{C}[\![c]\!] = \mathcal{C}[\![\textbf{if } b \textbf{ then } c_1; c \textbf{ else skip fi}]\!]$$

folgt $\mathcal{C}[\![c]\!]\sigma = \sigma$ und daher bereits $\sigma \models^I B$, wie gewünscht.
Nun haben wir also die Gültigkeit von $\{A \wedge b\}\ c_1\ \{A\}$ und $(A \wedge \neg b) \Rightarrow B$. Sei A' eine schwächste Vorbedingung für (c_1, A). Dann gilt $(A \wedge b) \Rightarrow A'$ und nach Induktionsvoraussetzung ist $\{A'\}\ c_1\ \{A\}$ herleitbar. Mit der Konsequenzregel ist also auch $\{A \wedge b\}\ c_1\ \{A\}$ herleitbar. Nach der **while**-Regel ist dann auch $\{A\}\ c\ \{A \wedge \neg b\}$ herleitbar. Durch nochmaliges Anwenden der Konsequenzregel erhalten wir $\vdash \{A\}\ c\ \{B\}$.
Damit ist Satz 5.2.5 bewiesen.

Nun erhalten wir leicht:

Nun erhalten wir leicht:

Satz 5.2.6 Es gelte $\models \{A\}\ c\ \{B\}$ für $c \in \mathbf{Cmd}$ und $A, B \in \mathbf{Assn}$. Dann gilt auch

$$\vdash \{A\}\ c\ \{B\}.$$

Das heißt, der Hoare-Kalkül ist als Beweissystem für partielle Korrektheit relativ vollständig.

Nun erhalten wir leicht:

Satz 5.2.6 Es gelte $\models \{A\}\ c\ \{B\}$ für $c \in \mathbf{Cmd}$ und $A, B \in \mathbf{Assn}$. Dann gilt auch

$$\vdash \{A\}\ c\ \{B\}.$$

Das heißt, der Hoare-Kalkül ist als Beweissystem für partielle Korrektheit relativ vollständig.

Beweis: Es gelte

$$\models \{A\}\ c\ \{B\}$$

und es sei A' eine schwächste Vorbedingung für (c, B). Dann ist nach Satz 5.2.5 die Aussage $\{A'\}\ c\ \{B\}$ herleitbar. Andererseits gilt (vgl. Seite 141) auch $A \Rightarrow A'$. Mit der Konsequenzregel erhalten wir

$$\vdash \{A\}\ c\ \{B\}.$$