4 Semantik von Programmiersprachen

Spezifizieren - Implementieren - Verifizieren

- Datenstrukturen sind Algebren: Signatur + Interpretation.
- Programmierbarkeit über einer Algebra: A.
- Programme $\in$ Programmiersprache: α.
- Semantik: Denotational, Operational.
- Partielle Korrektheit: $A \models \{\varphi\}\alpha\{\psi\}$.
- Kalkül von Hoare.

Benötigt wird:

- Sprachen für **Signaturen** (Funktionen, Prädikate, Stelligkeit).
- **Algebren** zur Signatur (Interpretationen).
- Sprache zur Beschreibung von **Eigenschaften**.
  Prädikatenlogik: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists, =$
- Sprache zur Beschreibung (Bezeichnung) der **Programme**.
  while , **do** , **end**, **if** , **then** , **else** ...
- **Zustände** zur Beschreibung der Wirkung der Programme.

4.1 Datenstrukturen/Algebren

4.1 Definition

Eine **Signatur** ist ein Paar (S, Σ) mit einer endlichen Menge S von **Sortensymbolen** (Typbezeichnern) und einer endlichen Menge Σ von **(Operationssymbol -) Deklarationen** der Form

$c :\longrightarrow s \qquad f : s_1 \times \cdots \times s_n \longrightarrow s \qquad p : s_1 \times \cdots \times s_n$

Mit $n > 0$, $s, s_1, \ldots, s_n \in S$.

- c heißt **Konstantensymbol**.
- f heißt **Funktionssymbol** der **Stelligkeit** n.
- p heißt **Relationssymbol** (Prädikatssymbol) der **Stelligkeit** n.

Keines der Zeichen c, f, p darf in verschiedenen Deklarationen vorkommen (kein „overloading“).

Eine **Variablenmenge** V über der Signatur (S, Σ) besteht aus **Variablendeklarationen** $X : s$ mit einem **Variablenbezeichner** X und einer Sorte $s \in S$. Kein Variablenbezeichner darf in zwei verschiedenen Variablendeklarationen vorkommen.

$X : s \in V$ $\quad X$ Variable vom Typ s.

Interpretationen von Signaturen (S, Σ): Algebren

4.2 Definition Algebren (Relationalstrukturen)
Eine (S, Σ)-**Algebra** A ist eine Abbildung, die jeder Sorte $s \in S$ eine nichtleere Menge s_A, jedem Konstantensymbol $c :\longrightarrow s$ in Σ eine Konstante $c_A \in s_A$, jedem Funktionssymbol $f : s_1 \times \cdots \times s_n \longrightarrow s$ in Σ eine totale Funktion $f_A : s_{1A} \times \cdots \times s_{nA} \longrightarrow s_A$ und jedem Relationssymbol $p : s_1 \times \cdots \times s_n$ in Σ eine Relation $p_A \subseteq s_{1A} \times \cdots \times s_{nA}$ zuordnet.

s_A heißt **Grundbereich** der Sorte s der Algebra A ($s \in S$).

Mehrsortige Algebren. Schreibe

$$A = (s_A(s \in S), c_A(c \in \Sigma), f_A(f \in \Sigma), p_A(p \in \Sigma))$$

4.3 Definition Zustände - Belegung von Variablen
Sei V eine endliche Variablenmenge über (S, Σ) und A eine (S, Σ)-Algebra. Ein **Zustand** z über A und V ist eine Funktion z, die jeder Variablen X mit $X : s$ in V einen Wert $z(X)$ in der Menge s_A zuordnet.

Bezeichnungen: $z : V \longrightarrow A$ (genauer in $\bigcup_{s \in S} s_A$).

Für $X : s$ in V, $a \in s_A$ sei **z(X/a)** der Zustand über A und V der X den Wert a und allen $Y \neq X$ den Wert $z(Y)$ zuordnet.

Entsprechend ist $z(X_1/a_1, \ldots, X_n/a_n)$ (kurz $z(\vec{X}/\vec{a})$) auf paarweise verschiedenen Variablen $X_1, \ldots, X_n$ definiert.

Beispiele: Algebren

4.4 Beispiel

- N::
  Signatur $(\{nat\}, \{0 :\rightarrow nat, succ : nat \rightarrow nat\})$
  Interpretation:

nat	$\rightarrow$	$nat_N = \mathbb{N}$
0		$0_N = 0 \in \mathbb{N}$
$succ$		$succ_N(x) = x + 1$

- Nat::
  Signaturerweiterung der Signatur von N um

$$+ : nat \times nat \rightarrow nat$$
$$* : \ nat \times nat \rightarrow nat$$
$$< : \ nat \times nat$$

$+_{Nat}(x, y)$	$=$	$x + y \ (+ \ in \ \mathbb{N})$
$*_{Nat}(x, y)$	$=$	$x * y \ (* \ in \ \mathbb{N})$
$<_{Nat} \ (x, y)$	gdw	$x < y \ (< \ in \ \mathbb{N})$

- $Boolean$::
  Signatur
  $(\{b\}, \{\text{true, false} :\rightarrow b, \text{not} : b \rightarrow b, \text{and, or} : b \times b \rightarrow b\})$
  Standard-Interpretation: z. B.
  $b_{Boolean} = \{W, F\}$
  $\text{true}_{Boolean} = W$ und $\text{false}_{Boolean} = F$
  $\text{or}_{Boolean}(F, F) = F$
  $\text{or}_{Boolean}(x, y) = W$ für $(x, y) \neq (F, F)$
  ...

Beispiele: Algebren (Forts.)

- $set_d(A)$::
  Für A eine (S, Σ)-Algebra $d \in S$.
  Intention: Sorte, die endliche Teilmengen der Menge d_A beschreibt:
  Signaturerweiterung von (S, Σ) um Sorte set_d und Funktionssymbole $\varnothing :\longrightarrow set_d$ und $insert : set_d \times d \rightarrow set_d$
  Interpretation A erweitert um
  $(set_d)_{set_d(A)} :=$ alle endlichen Teilmengen von d_A
  $\varnothing_{set_d(A)} :=$ leere Menge
  $insert_{set_d(A)}(M, a) := M \cup \{a\}$ für M endlich und $a \in d_A$.
- $set_{nat}(Nat)$::
  Variablenmenge V: $X, Y : nat$ und $X_1, X_2 : set_{nat}$ dann sind z_1 und z_2 mit
  $z_1(X) = 0, \quad z_1(Y) = 3, \quad z_1(X_1) = \varnothing,$
  $z_1(X_2) = \{0, 1, 3\}$ und
  $z_2(X) = 1, \quad z_2(Y) = 0, \quad z_2(X_1) = \{0, 1, 3\},$
  $z_2(X_2) = \varnothing$ Zustände.

  $z_1(X/1, Y/0, X_1/\{0, 1, 3\}, X_2/\varnothing) = z_2$

4.2 Sprache zur Beschreibung von Eigenschaften in Algebren

Prädikatenlogik (erster Stufe) als Spezifikationssprache.

Terme als Bezeichner für Objekte einer (S, Σ)-Algebra.

4.5 Definition $\mathbf{Term(S, \Sigma, V)}$

Terme über einer Signatur und Variablenmenge V mit jeweiligem Typ sind induktiv wie folgt definiert:

- Ist $X : s$ eine Variable in V, so ist X ein Term vom Typ s.
- Ist $c :\longrightarrow s$ in Σ, so ist c ein Term vom Typ s.
- Ist $f : s_1 \times \cdots \times s_n \longrightarrow s$ in Σ, t_i ein Term über $(S, \Sigma), V$ vom Typ s_i für $i = 1, \ldots, n$, so ist $f(t_1, \ldots, t_n)$ ein Term über (S, Σ) und V vom Typ s.

Termkalkül für $Term(S, \Sigma, V)$ Menge der Terme über (S, Σ) und V und Definition von $Typ : Term(S, \Sigma, V) \longrightarrow S$

$$\frac{}{X}\ (X : s \in V) \qquad Typ(X) = s$$

$$\frac{}{c}\ (c :\longrightarrow s \in \Sigma) \qquad Typ(c) = s$$

$$\frac{t_1, \ldots, t_n}{f(t_1, \ldots, t_n)} \qquad (Typ(t_i) = s_i, f : s_1 \ldots s_n \longrightarrow s \in \Sigma).$$
$$Typ(f(t_1, \ldots, t_n)) = s.$$

Beispiele

4.6 Beispiel Beachte: Der Termkalkül ist eindeutig, d.h. jeder Term wird eindeutig aus den Teiltermen aufgebaut. Somit ist auch Typ eine wohldefinierte Funktion auf $Term$.

a) Signatur von N Variablenmenge:: $V\ X : nat$

$$0 \quad X \quad succ^n(0) \quad succ^n(X) \quad (n \in \mathbb{N})$$

b) Signatur von Nat $X, Y : nat$

Terme sind:

$(X + 0), (X * Y), (X + succ(Y)), succ((X + Y))$

eigentlich

$+(X, 0), *(X, Y), +(X, succ(Y)), succ(+(X, Y))$

Beachte Infixnotation und Klammerungsregelung. Wird für die Operationssymbole Infix - Notation gewählt, so sind äußere Klammern zu verwenden um die Eindeutigkeit der Zerlegung eines Terms in Teiltermen sicherzustellen. Zur besseren Lesbarkeit werden oft äußere Klammern unterdrückt und Prioritäten (Bindungsstärken) zwischen den Operationen vereinbart.

$succX + 0 * Y$ steht für $+(succ(X), *(0, Y))$.

Priorität $succ$, $*$, $+$.

Keine Terme sind:

$X+, *succ(0), \ldots$

Die Sprache der Prädikatenlogik - Formeln

4.7 Definition $\mathbf{Form(S, \Sigma, V)}$
(S, Σ) Signatur, V Variablenmenge über (S, Σ).

Boolesche Formeln über $(\mathbf{S}, \mathbf{\Sigma})$, $\mathbf{V}$ sind definiert durch:

- $p(t_1, \ldots, t_n)$ ist **atomare Boolesche-Formel** für $p : s_1 \times \cdots \times s_n \in \Sigma$, t_i Term vom Typ s_i $(i = 1, \ldots, n)$.
- $t_1 = t_2$ ist **Gleichung** für t_1, t_2 Terme über (S, Σ) und V vom gleichen Typ.
- Sind φ und ψ Boolesche-Formeln über (S, Σ), V, so auch die Folgenden:
  $\neg\varphi$ -nicht φ-, $(\varphi \rightarrow \psi)$ -φ impliziert ψ-,
  $(\varphi \wedge \psi)$ -φ und ψ-, $(\varphi \vee \psi)$ -φ oder ψ-
  $(\varphi \leftrightarrow \psi)$ -φ äquivalent ψ-

Prädikatenlogische Formeln über $(\mathbf{S}, \mathbf{\Sigma})$, $\mathbf{V}$

- Eine Boolesche-Formel ist eine PL-Formel.
- Ist φ Formel, so auch $\neg\varphi$.
  Sind φ, ψ Formeln, so auch $(\varphi * \psi)$ $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
- φ Formel, $X : s \in V$, so auch $\forall X \varphi$, $\exists X \varphi$.
  φ ist der Wirkungsbereich vom Quantor $\forall X$ bzw. $\exists X$.

Ein Vorkommen einer Variablen X in einer Formel heißt **frei**, sofern es nicht im Wirkungsbereich eines Quantors $\forall X$ oder $\exists X$ auftritt. Andernfalls heißt das Vorkommen **gebunden**.

Eindeutiger Kalkül zur Erzeugung der Formeln $\mathbf{Form(S, \Sigma, V)}$

- $\dfrac{}{p(t_1, \ldots, t_n)}$ $\quad (Typ(t_i) = s_i, p : s_1 \times \cdots \times s_n \in \Sigma)$
- $\dfrac{}{t_1 = t_2}$ $\quad (Typ(t_1) = Typ(t_2))$
- $\dfrac{\varphi}{\neg\varphi}$, $\quad \dfrac{\varphi, \psi}{(\varphi * \psi)}$ $\quad (* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\})$
- $\dfrac{\varphi}{\forall X \varphi}$ -für alle-, $\quad \dfrac{\varphi}{\exists X \varphi}$ -es gibt- für X Variablenbezeichner.

4.8 Beispiel Nat

1. $\exists Y succ(Y) = X$
2. $(X + succ(Y)) = succ(X + Y)$
3. $\forall X \forall Y (X + succ(Y)) = succ(X + Y)$ („Gleichungsaxiom“)
4. $(\exists Y succ(Y) = X \wedge (X < succ(X) \wedge Y = 0))$

In

1. Y kommt nur gebunden vor, X nur frei.
2. Alle Vorkommen von X und Y sind frei.
3. Alle Vorkommen von X und Y sind gebunden.(„Abgeschlossen“)
4. Alle Vorkommen von X sind frei.
   Erstes Vorkommen von Y ist gebunden.
   Zweites Vorkommen von Y ist frei.

Abgeschlossene Formeln - Substitution

4.9 Definition Eine Formel heißt **abgeschlossen**, falls sie keine freien Vorkommen von Variablen enthält.

Im Beispiel 4.8: 3. ist abgeschlossen (auch **Satz** oder **Sentence**).

4.10 Definition Substitution

Eine **Substitution** σ über $(S, \Sigma), V$ ist eine typtreue Abbildung der Menge der Variablenbezeichner in $Term(S, \Sigma, V)$, die nur an endlich vielen Stellen von der Identität verschieden ist.

Sie kann somit durch die Menge $\{X_1/s_1, \ldots, X_m/s_m\}$ ($(\vec{X}/\vec{s})$ als Vektor) beschrieben werden: Hierbei sind

- $X_1, \ldots, X_m$ Variablenbezeichner, paarweise verschieden.
- $s_1, \ldots, s_m$ sind Σ-Terme über V.
- X_i und s_i sind verschieden und vom selben Typ.

Fortsetzung von σ auf $Term(S, \Sigma, V)$:

$t\sigma$ für $t \in$ Term(S, Σ, V) ist definiert als:

- $X_i\sigma = s_i \quad 1 \leq i \leq m$
- $Y\sigma = Y \quad Y \in V \backslash \{X_1, \ldots, X_m\}$
- $c\sigma = c$
- $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$

Substitution (Forts.)

4.11 Lemma

σ ist wohldefiniert und total auf $Term(S, \Sigma, V)$.

Beweis: einfache strukturelle Induktion.

4.12 Beispiel

$\mathsf{and}(X, Y)\{X/Y, Y/\mathsf{true}\} = \mathsf{and}(Y, \mathsf{true}) \qquad X \neq Y$

$(X + succ(Y))\{X/succ(0), Y/succ(X)\} =$
$(succ(0) + succ(succ(X)))$

$succ(X)\{X/succ(X + Y)\} = succ(succ(X + Y))$

Beachte:

1. $t\sigma$ hängt nur von den Werten der in t vorkommenden Variablen ab.
2. σ ist Homomorphismus der "Termalgebra" $Term(S, \Sigma, V)$ in sich selbst.

3. Enthält t keine Variablen (d.h. t ist Grundterm), so $t\sigma = t$ für jede Substitution σ.

Bewertung von Termen

Frage: Welche Werte haben Terme im Zustand z?

4.13 Definition Werte von Terme im Zustand z.
Sei (S, Σ) eine Signatur, V eine Variablenmenge über (S, Σ), A eine (S, Σ)-Algebra, z Zustand über A und V.

Für $t \in Term(S, \Sigma, V)$ sei der Wert von t in Algebra A und Zustand z, kurz $val_{A,z}(t)$, induktiv wie folgt definiert:

- $val_{A,z}(X) = z(X)$ für $X \in V$
- $val_{A,z}(c) = c_A$ für $c \in \Sigma$
- $val_{A,z}(f(t_1, \ldots, t_n)) = f_A(val_{A,z}(t_1), \ldots, val_{A,z}(t_n))$

4.14 Beispiel

$z = (X/5, Y/3)$, d. h. $z(X) = 5, z(Y) = 3$

$$\begin{aligned} val_{Nat,z}(X + succ(Y)) &= 5 +_{Nat} val_{Nat,z}(succ(Y)) \\ &= 5 +_{Nat} (3 +_{Nat} 1) = 9 \end{aligned}$$

$$\begin{aligned} val_{N,z}(succ^5(0)) &= val_{N,z}(succ^4(0)) + 1 \\ &= val_{N,z}(succ^3(0)) + 1 + 1 = \cdots \\ &= 5 \end{aligned}$$

Bewertung von Termen - Zustände und Substitutionen

4.15 Lemma

a) $val_{A,z}$ ist wohldefiniert und $val_{A,z}(t) \in Typ(t)_A$.

b) Ist z' ein weiterer Zustand über A und V mit $z(X) = z'(X)$ für alle in t vorkommenden Variablen X, so ist $val_{A,z}(t) = val_{A,z'}(t)$.

D. h. der Wert von t hängt nur von den Werten der in t vorkommenden Variablen ab. Enthält der Term t keine Variablen (Grundterm), so hängt der Wert nicht vom Zustand z ab.

Beweis: Eindeutigkeit des Termkalküls und rekursive Definition von val mithilfe der Werte der Teilterme.

4.16 Lemma Substitutionslemma für Terme

A eine (S, Σ)-Algebra, V Variablenmenge, z Zustand über A, V.

$X \in V$, $r, t \in Term(S, \Sigma, V)$.

Sei $a = val_{A,z}(r)$. Dann gilt

$$val_{A,z}(t\{X/r\}) = val_{A,z(X/a)}(t)$$

Eine Substitution kann also bei der Termauswertung durch eine Zustandsmodifikation simuliert werden.

Substitutionslemma für Terme (fort.)

Das Lemma lässt sich auf Simultansubstitution mehrerer Var $\vec{X}$ durch Terme $\vec{r}$ verallgemeinern.

Beweis:

Strukturelle Induktion über Aufbau der Terme (Kalkül).

$t =:$

1. X:
   $$\begin{aligned} val_{A,z}(t\{X/r\}) &= val_{A,z}(r) = a(= val_{A,z}(r)) \\ &= val_{A,z(X/a)}(X) = val_{A,z(X/a)}(t) \end{aligned}$$
2. Y von X verschieden:
   $$\begin{aligned} val_{A,z}(t\{X/r\}) &= val_{A,z}(Y) \\ &= z(Y) = z(X/a)(Y) \\ &= val_{A,z(X/a)}(Y) = val_{A,z(X/a)}(t) \end{aligned}$$
3. c:
   $$val_{A,z}(t\{X/r\}) = c_A = val_{A,z(X/a)}(t)$$
4. $f(t_1, \ldots, t_n)$:
   $$\begin{aligned} val_{A,z}(t\{X/r\}) &= val_{A,z}(f(t_1\{X/r\}, \ldots, t_n\{X/r\})) \\ &= f_A(val_{A,z}(t_1\{X/r\}, \ldots, \\ &\qquad val_{A,z}(t_n\{X/r\})) \\ &= f_A(val_{A,z(X/a)}(t_1), \ldots, val_{A,z(X/a)}(t_n)) \\ &= val_{A,z(X/a)}(f(t_1, \ldots, t_n)) \\ &= val_{A,z(X/a)}(t) \end{aligned}$$

4.3 Bewertung und Gültigkeit von Formeln

4.17 Definition Gültigkeit im Zustand

A (S, Σ)-Algebra, V Variablenmenge, z Zustand über A, V.

Sei $\xi \in Form(S, \Sigma, V)$ Formel und X mit $Typ(X) = s$.

ξ **gilt in der Algebra** A **im Zustand** z: $A \models_z \xi$:
(Schreibe $A \not\models_z \xi$ für $A \models_z \xi$ gilt nicht).

Wird induktiv definiert durch

$A \models_z p(t_1, \ldots, t_n)$ gdw $(val_{A,z}(t_1), \ldots, val_{A,z}(t_n)) \in p_A$
$(A \not\models_z p(t_1, \ldots, t_n)$ gdw $(val_{A,z}(t_1), \ldots, val_{A,z}(t_n)) \notin p_A)$
$A \models_z t_1 = t_2$ gdw $val_{A,z}(t_1) = val_{A,z}(t_2)$
$A \models_z \neg\varphi$ gdw $A \not\models_z \varphi$
$A \models_z (\varphi \wedge \psi)$ gdw $A \models_z \varphi$ und $A \models_z \psi$
$A \models_z (\varphi \vee \psi)$ gdw $A \models_z \varphi$ oder $A \models_z \psi$
$A \models_z (\varphi \rightarrow \psi)$ gdw $A \not\models_z \varphi$ oder $A \models_z \psi$
$A \models_z (\varphi \leftrightarrow \psi)$ gdw $(A \models_z \varphi$ und $A \models_z \psi)$ oder
$(A \not\models_z \varphi$ und $A \not\models_z \psi)$
$A \models_z \exists X\varphi$ gdw $A \models_{z(X/a)} \varphi$ für ein $a \in s_A$
$A \models_z \forall X\varphi$ gdw $A \models_{z(X/a)} \varphi$ für alle $a \in s_A$

Beachte: Für jede Formel ξ gilt entweder $A \models_z \xi$ oder $A \not\models_z \xi$.

Insbesondere entweder $A \models_z \xi$ oder $A \models_z \neg\xi$.

4.18 Definition Gültigkeit

$A \models \varphi$ (φ gilt in A oder φ ist gültig in A) gdw $A \models_z \varphi$ für alle Zustände z über A, V.

Beispiele

4.19 Beispiel In **Boolean:**

$$\begin{aligned} A \quad &\models \mathsf{and}(X, Y) = \mathsf{and}(Y, X) \\ &\models \mathsf{or}(X, \mathsf{not}(X)) = \mathsf{true} \\ &\models \mathsf{and}(X, \mathsf{not}(X)) = \mathsf{false} \\ &\models \mathsf{not}(\mathsf{and}(\mathsf{not}(X), \mathsf{not}(Y))) = \mathsf{or}(X, Y) \end{aligned}$$

In jeder Struktur $\mathbf{A}$, beliebige Formeln φ, ψ über Signatur von A.

$$\begin{aligned} A \quad &\models (\varphi \vee \neg\varphi) \\ &\models \varphi \leftrightarrow \neg\neg\varphi \\ &\models (\varphi \vee \psi) \leftrightarrow (\neg\varphi \to \psi) \\ &\models (\varphi \wedge \psi) \leftrightarrow \neg(\varphi \to \neg\psi) \end{aligned}$$

Gilt für Formeln φ, ψ: $A \models \varphi \leftrightarrow \psi$, so heißen sie **logisch äquivalent in** A.

Eigenschaft: Jede Formel φ lässt sich effektiv in eine logisch äquivalente Formel ψ, die nur die Operationen $\neg$, $\to$ enthält, transformieren.

In $\mathbf{N}$:

$$N \models \forall Y \exists X \quad X = succ(Y)$$

Frage: Gilt auch

$$N \overset{?}{\models} \forall X \exists Y \quad X = succ(Y) \text{ oder}$$
$$N \overset{?}{\models} \exists X \forall Y \quad X = succ(Y)?$$

Beispiele (Forts.)

Behauptung: Nein, dafür
finde Zustand z mit

$$N \not\models_z \exists Y \quad X = succ(Y) \quad z.B.\ z(X) = 0$$

bzw.
für alle Zustände z gilt $N \not\models_z \forall Y\ X = succ(Y)$.

Sei $z(X)$ beliebig aber fest, dann liefert $z(Y) := z(X)$ ein „Gegenbeispiel".

Beachte:

$A \models (\forall X \varphi \leftrightarrow \neg \exists X \neg \varphi)$
$A \models (\exists X \varphi \leftrightarrow \neg \forall X \neg \varphi)$
$N \models (\exists X\ X = succ(Y) \leftrightarrow \exists Z\ Z = succ(Y))$

Anwendung von Substitutionen auf Formeln

$N \models \forall Y \exists X \ \ X = succ(Y)$

Die Ersetzung von Y durch einen beliebigen Term sollte eine Formel liefern, die in N gilt.

Vorsicht:

$\{Y/0\} ::\ \ N \models_z \exists X \ \ X = succ(0)$ (bel. z)
$\{Y/X\} :: N \not\models_z \exists X \ \ X = succ(X)$ (bel. z)

Anwendung von Substitutionen auf Formeln

Problem: Im Wirkungsbereich des Quantors $\exists X$ wird ein Term substituiert der X enthält, d. h. freies Vorkommen von Y wird gebundenes Vorkommen von X.

Lösung: Umbenennung gebundener Variablen.

4.20 Definition

Sei $\sigma = \{X_1/s_1, \ldots, X_m/s_m\}$ eine Substitution.
Induktiv über die Struktur der Formel φ sei $[\varphi]\sigma$ wie folgt definiert:

$$
\begin{array}{ll}
[p(t_1, \ldots, t_n)]\sigma & = p(t_1\sigma, \ldots, t_n\sigma) \\
[t_1 = t_2]\sigma & = t_1\sigma = t_2\sigma \\
[\neg\varphi]\sigma & = \neg[\varphi]\sigma \\
[(\varphi * \psi)]\sigma & = ([\varphi]\sigma * [\psi]\sigma),\ * \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \\
[QX\varphi]\sigma & = QY[[\varphi]\{X/Y\}]\sigma, \quad Q \in \{\forall, \exists\}
\end{array}
$$

Wobei Y eine „frische“ Variable ist d. h. Y kommt **nicht** in QX, φ, σ vor.

Dabei kommt eine Variable in Substitution σ vor, falls sie in $\{X_1, \ldots, X_m\}$ oder $\{s_1, \ldots, s_m\}$ vorkommt.

4.21 Beispiel

In N: $[\exists X\ X = succ(Y)]\{Y/X\}$ Z „neu“

$$
\begin{aligned}
&= \exists Z[Z = succ(Y)]\{Y/X\} \\
&= \exists Z\ Z = succ(X)
\end{aligned}
$$

Anwendung von Substitutionen auf Formeln (Forts.)

In Nat: Sei $\sigma : \{X/(X+Y), Y/(Y+Z), Z/0\}$

$\varphi :: \quad \forall X \forall Y (X + succ(Y)) = succ(X + Y)$

$[\varphi]\sigma =$

$\varphi' :: \quad \forall Y (X + succ(Y)) = succ(X + Y)$

$[\varphi']\sigma =$

$\varphi'' :: \quad (X + succ(Y)) = succ(X + Y)$

$[\varphi'']\sigma =$

Substitutionslemma für Formeln

4.22 Lemma

Sei A eine (S, Σ)-Algebra, φ eine Formel, $\vec{X}$ Variablen, $\vec{t}$ Terme vom selben Typ und z ein Zustand, der auf allen freien Variablen von $[\varphi]\{\vec{X}/\vec{t}\}$ definiert ist. Es sei $\vec{a} = val_{A,z}(\vec{t})$.

Dann ist $z(\vec{X}/\vec{a})$ auf allen freien Variablen von φ definiert und es gilt

$$A \models_z [\varphi]\{\vec{X}/\vec{t}\} \text{ gdw } A \models_{z(\vec{X}/\vec{a})} \varphi$$

Beweis: Induktion über Aufbau von φ

(Beachte z ist o.B.d.A. auf allen Variablen der t_i definiert und somit ist $z(\vec{X}/\vec{a})$ auf allen Variablen der t_i und der Variablen in $\vec{X}$ definiert.)

Fall $\varphi = \forall Z\psi$, Y sei eine Variable, die in ψ, Z und $\{\vec{X}/\vec{t}\}$ nicht vorkommt. Dann

$$A \models_z [\forall Z\psi]\{\vec{X}/\vec{t}\} \text{ gdw } A \models_z \forall Y[[\psi]\{Z/Y\}]\{\vec{X}/\vec{t}\}$$

$$\begin{array}{ll}
\text{gdw } A \models_{z(Y/b)} [[\psi]\{Z/Y\}]\{\vec{X}/\vec{t}\} & \text{für alle } b \text{ in Typ}(Y)_A \\
\overset{(IV)}{\text{gdw}} A \models_{z(Y/b)(\vec{X}/\vec{a})} [\psi]\{Z/Y\} & \text{für alle } b \\
\overset{(IV)}{\text{gdw}} A \models_{z(Y/b)(\vec{X}/\vec{a})(Z/b)} \psi & \text{für alle } b \\
\text{gdw } A \models_{z(\vec{X}/\vec{a})(Z/b)} \psi & \text{für alle } b \\
\text{gdw } A \models_{z(\vec{X}/\vec{a})} \forall Z\psi &
\end{array}$$

Substitutionslemma für Formeln (Forts.)

4.23 Folgerung

Für alle Algebren A, Zustände z, Formeln φ, Variablen X und Terme t vom selben Typ gilt:

$$A \models_z (\forall X \varphi \rightarrow [\varphi]\{X/t\})$$

Also ist $(\forall X \varphi \rightarrow [\varphi]\{X/t\})$ „universell“ gültig, „allgemein gültig“.

Beachte Literatur:
Andere Definitionen und Schreibweisen üblich,

z. B. - „erlaubte Substitutionen“, - φ_t^X für $[\varphi]\{X/t\}$ oder $\varphi_{\vec{t}}^{\vec{x}}$

4.24 Lemma Koinzidenzlemma für Formeln

Seien A, φ, V gegeben. Sind z und z' Zustände über V mit $z(X) = z'(X)$ für alle freien Variablen X von φ, dann gilt

$$A \models_z \varphi \ \ \mathrm{gdw} \ \ A \models_{z'} \varphi$$

Die Bewertung einer Formel (ob φ im Zustand z gilt oder nicht gilt) hängt nur von den Werten der in φ **frei** vorkommenden Variablen ab.

Insbesondere gilt dies für abgeschlossene Formeln, für solche gilt entweder $A \models \varphi$ oder $A \models \neg\varphi$.

Beachte dies muss nicht für Formeln mit freien Variablen gelten.

4.4 Programme über einer Signatur (S, Σ)

Zuweisung, Verzweigung, Iteration sind wesentliche Konstrukte jeder „universellen“ Programmiersprache.

Programm:: Mittel zur Beschreibung eines effektiven Prozesses.

4.25 Definition $\mathbf{Prog(S,\Sigma, V)}$

Sei (S, Σ) eine Signatur, V endliche Variablenmenge. Die Menge der **While-Programme** über $(S, \Sigma), V$ sei durch folgenden Kalkül definiert.

$\overline{\varepsilon}$ ε **leeres Programm**

$\overline{X := t;}$ $X : s \in V, t \in \mathsf{Term}(S, \Sigma, V)$, $\mathsf{Typ}(t) = s$ **Zuweisung**

$\dfrac{\beta, \gamma}{\underline{\textbf{if}}\ B\ \underline{\textbf{then}}\ \beta\ \underline{\textbf{else}}\ \gamma\ \underline{\textbf{end}};}$ B Boolesche-Formel über $(S, \Sigma), V$ **Test**

$\dfrac{\beta}{\underline{\textbf{while}}\ B\ \underline{\textbf{do}}\ \beta\ \underline{\textbf{end}};}$ B Boolesche-Formel über $(S, \Sigma), V$ **Schleife**

$\dfrac{\alpha, \beta}{\alpha\beta}$ (als Konkatenation von Zeichenreihen) **Komposition**

Eine **Anweisung** ist entweder eine Zuweisung, ein Test oder eine Schleife.

Beachte:

Jedes Programm ist entweder ε oder fängt mit einer Anweisung an.

Beispiele (1)

4.26 Beispiel While-Programm α über Signatur von Nat.

$\alpha :: Y := 0;\ Z := 0;$
while $\neg Y = X$ **do**
$Z := succ(Z + (Y + Y));\ Y := succ(Y);$
end;

While-Programme β und γ über Signatur von N.

$\beta :: Z := X;\ Z' := 0;$
while $\neg Y = Z'$ **do**
$Z := succ(Z);\ Z' := succ(Z');$
end;

$\gamma :: Z := 0;\ Z' := 0;$
while $\neg Y = Z'$ **do**
$\beta\{X/Z, Y/X, Z'/Z''\};\ Z' := succ(Z');$
end;

Makros: Hierbei steht $\beta\{X/Z, Y/X, Z'/Z''\}$ für Programm welches durch Substitution der entsprechenden Variablen entsteht, d. h.

$Z := Z;\ Z'' := 0;$
while $\neg X = Z''$ **do**
$Z := succ(Z);\ Z'' := succ(Z'');$
end;