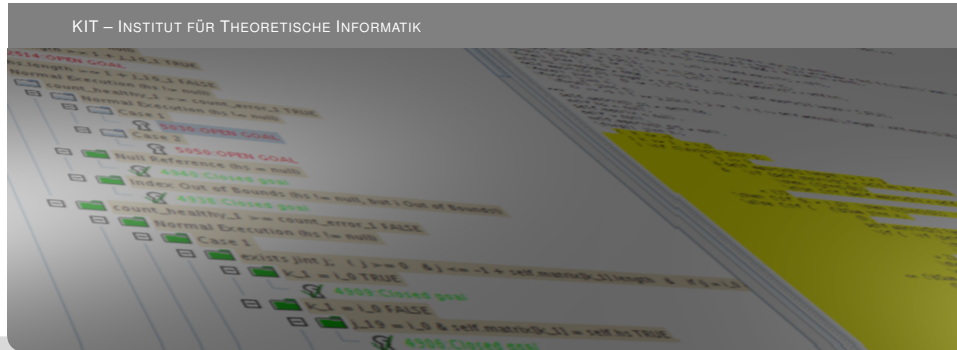


Formale Systeme

Prädikatenlogik: Syntax
Prof. Dr. Peter H. Schmitt

KIT – INSTITUT FÜR THEORETISCHE INFORMATIK



Einführendes Beispiel

Der Klassiker

Alle Menschen sind sterblich.

Sokrates ist ein Mensch.

Also ist Sokrates sterblich.

Prädikatenlogische Formalisierung:

$\forall x(\text{Mensch}(x) \rightarrow \text{sterblich}(x))$

$\text{Mensch}(\text{Sokrates})$

$\text{sterblich}(\text{Sokrates})$

Logische Zeichen: $\forall x, \rightarrow$

Anwendungsabhängiges Vokabular:

$\text{Mensch}(\cdot), \text{sterblich}(\cdot), \text{Sokrates}$

Einführendes Beispiel 2

```

1   int max = 0;
2   if ( a.length > 0 ) max = a[0];
3   int i = 1;
4   while ( i < a.length ) {
5       if ( a[i] > max ) max = a[i];
6       ++i;
7   }
  
```

Nachbedingung:

$$(\forall \text{ int } j; (j \geq 0 \ \& \ j < a.length \rightarrow \text{max} \geq a[j]))$$

&

$$\begin{aligned}
 & (a.length > 0 \rightarrow \\
 & \quad \exists \text{ int } j; (j \geq 0 \ \& \ j < a.length \ \& \ \text{max} = a[j])) \\
 &)
 \end{aligned}$$

Einführendes Beispiel 3

API Spezifikation

Die Java Card Platform Specification v2.2.1
(siehe <http://java.sun.com/products/javacard/specs.html>)
enthält u.a. die Klasse

```
public class Util  
extends Object
```

mit der Methode `arrayCompare`:

Method Summary	
static byte	<code>arrayCompare</code> (byte[] src, short srcOff, byte[] dest, short destOff, short length) Compares an array from the specified source array, beginning at the specified position, with the specified position of the destination array from left to right.

```
public static final byte arrayCompare (byte[] src,  
                                       short srcOff,  
                                       byte[] dest,  
                                       short destOff,  
                                       short length)  
                                       throws ArrayIndexOutOfBoundsException,  
                                       NullPointerException
```

Compares an array from the specified source array, beginning at the specified position, with the specified position of the destination array from left to right.

Returns the ternary result of the comparison :
less than(-1), equal(0) or greater than(1).

- ▶ If `srcOff` or `destOff` or `length` parameter is negative an `ArrayIndexOutOfBoundsException` exception is thrown.
- ▶ If `srcOff+length` is greater than `src.length` (the length of the `src` array) a `ArrayIndexOutOfBoundsException` exception is thrown.
- ▶ If `destOff+length` is greater than `dest.length`, the length of the `dest` array an `ArrayIndexOutOfBoundsException` exception is thrown.
- ▶ If `src` or `dest` parameter is `null` a `NullPointerException` exception is thrown.

Formale JML-Spezifikation (Ausschnitt)

```
public normalbehaviour
  ensures \result >= -1 && \result <= 1;
  ensures (\exists int i; 0<=i && i<length &&
    src[srcOff+i] < dest[destOff+i] &&
    (\forall int j; 0<=j && j<i ==>
      src[srcOff+j] == dest[destOff+j]))
    ==> \result == -1
  ...
```

Logische Zeichen:

	\forall	\exists	\wedge	\vee	\rightarrow	$\dot{=}$
entsprechen	\forall	\exists	\wedge	\vee	\rightarrow	$\dot{=}$

\Rightarrow *JML wird später Thema sein.*

Formalisierung: Normales Verhalten

$$s \neq \text{null} \quad \wedge \quad sO \geq 0 \quad \wedge \quad sO + l \leq \text{size}(s) \quad \wedge$$
$$d \neq \text{null} \quad \wedge \quad dO \geq 0 \quad \wedge \quad dO + l \leq \text{size}(d) \quad \wedge \quad l \geq 0$$

→

$$\neg \text{excThrown}(E) \quad \wedge$$
$$(\text{res} = -1 \quad \vee \quad \text{res} = 0 \quad \vee \quad \text{res} = 1) \quad \wedge$$
$$(\text{subSeq}(s, sO, sO + l) = \text{subSeq}(d, dO, dO + l) \rightarrow \text{res} = 0) \quad \wedge$$
$$(\exists i:\text{Int}(0 \leq i \wedge i < l \wedge (\text{at}(s, sO + i) < \text{at}(d, dO + i) \quad \wedge$$
$$\vee j:\text{Int}(0 \leq j \wedge j < i \rightarrow \text{at}(s, sO + j) = \text{at}(d, dO + j)))) \quad \wedge$$
$$) \rightarrow \text{res} = -1) \quad \wedge$$
$$(\exists i:\text{Int}(0 \leq i \wedge i < l \wedge (\text{at}(s, sO + i) > \text{at}(d, dO + i) \quad \wedge$$
$$\vee j:\text{Int}(0 \leq j \wedge j < i \rightarrow \text{at}(s, sO + j) = \text{at}(d, dO + j)))) \quad \wedge$$
$$) \rightarrow \text{res} = 1)$$

Logisches Vokabular in rot

<i>s</i>	für	<i>src</i>	<i>d</i>	für	<i>dest</i>
<i>sO</i>	für	<i>srcOff</i>	<i>dO</i>	für	<i>destOff</i>
<i>l</i>	für	<i>length</i>			
<i>E</i>	für	<i>java :: lang :: Exception</i>			
<i>NPE</i>	für	<i>java :: lang :: NullPointerException</i>			
<i>OBE</i>	für	<i>java :: lang :: ArrayIndexOutOfBoundsException</i>			

$$\neg excThrown(E) \quad \checkmark$$
$$excThrown(NPE) \wedge (s = null \vee d = null) \quad \checkmark$$
$$excThrown(OBE) \wedge$$
$$(sO < 0 \vee dO < 0 \vee l < 0 \vee sO + l > size(s) \vee dO + l > size(d))$$

Prädikatenlogik erster Stufe

Syntax

Definition: Logische Zeichen

Wie in der Aussagenlogik:

$\neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)$

Neu:

\forall Allquantor

\exists Existenzquantor

v_i Individuenvariablen, $i \in \mathbb{N}$

\doteq objektsprachliches Gleichheitssymbol

, Komma

Mit *Var* bezeichnen wir die zur Verfügung stehenden Variablen.

Prädikatenlogik erster Stufe

Signatur

Definition: Signatur

Eine *Signatur* ist ein Tripel $\Sigma = (F_\Sigma, P_\Sigma, \alpha_\Sigma)$ mit:

- ▶ F_Σ, P_Σ sind endliche oder abzählbar unendliche Mengen
- ▶ F_Σ, P_Σ und die Menge der Sondersymbole sind paarweise disjunkt
- ▶ $\alpha_\Sigma : F_\Sigma \cup P_\Sigma \rightarrow \mathbb{N}$.

$f \in F_\Sigma$ heißt **Funktionssymbol**,

$p \in P_\Sigma$ heißt **Prädikatssymbol**.

f ist **n -stelliges** Funktionssymbol, wenn $\alpha_\Sigma(f) = n$;

p ist **n -stelliges** Prädikatssymbol, wenn $\alpha_\Sigma(p) = n$; Ein

nullstelliges Funktionssymbol heißt auch **Konstantensymbol**
 oder kurz **Konstante**,

ein nullstelliges Prädikatsymbol ist ein *aussagenlogisches Atom*.

Definition: Terme

Term_Σ , die Menge der *Terme über* Σ , ist induktiv definiert durch

1. $\text{Var} \subseteq \text{Term}_\Sigma$
2. Mit $f \in F_\Sigma$,
 $\alpha_\Sigma(f) = n$,
 $t_1, \dots, t_n \in \text{Term}_\Sigma$

ist auch $f(t_1, \dots, t_n) \in \text{Term}_\Sigma$

Ein Term heißt *Grundterm*, wenn er keine Variablen enthält.

Definition: Atomare Formeln

At_{Σ} , die Menge der *atomaren Formeln* über Σ :

$$At_{\Sigma} := \{s \doteq t \mid s, t \in Term_{\Sigma}\} \cup \{p(t_1, \dots, t_n) \mid p \in P_{\Sigma}, \alpha_{\Sigma}(p) = n, t_i \in Term_{\Sigma}\}$$

Definition: Formeln

For_{Σ} , die Menge der *Formeln über Σ* , ist induktiv definiert durch

1. $\{\mathbf{1}, \mathbf{0}\} \cup At_{\Sigma} \subseteq For_{\Sigma}$
2. Mit $x \in Var$ und $A, B \in For_{\Sigma}$ sind ebenfalls in For_{Σ} :

$$\neg A, (A \wedge B), (A \vee B), (A \rightarrow B), (A \leftrightarrow B), \forall xA, \exists xA$$

Quiz

Welche der folgenden Zeichenketten sind korrekt gebildete Formeln?

p, q einstellige Prädikatszeichen r zweistellige Prädikatszeichen
 c, d Konstantensymbole x, y, z Variablen

- | | | |
|---|---|-------------|
| 1 | $\forall x(\forall y(\forall z(r(x, y) \wedge r(y, z) \rightarrow r(x, z))))$ | <i>ja</i> |
| 2 | $\forall xq(x) \wedge \exists x\neg q(x)$ | <i>ja</i> |
| 3 | $\forall x(\forall y(p(x) \wedge q(y) \rightarrow \exists r(r(x, y))))$ | <i>nein</i> |
| 4 | $\exists z(c(z))$ | <i>nein</i> |
| 5 | $\forall xp(c) \wedge \forall y\exists yq(y)$ | <i>ja</i> |
| 6 | $p(c) \wedge p(d) \rightarrow p(r(c, d))$ | <i>nein</i> |

Definition

- ▶ Hat eine Formel A die Gestalt $\forall xB$ oder $\exists xB$, so heißt B der **Wirkungsbereich** des *Präfixes* $\forall x$ bzw. $\exists x$ von A .
- ▶ Ein Auftreten einer Variablen x in einer Formel A heißt **gebunden**, wenn es innerhalb des Wirkungsbereichs eines Präfixes $\forall x$ oder $\exists x$ einer Teilformel von A stattfindet.
- ▶ Ein Auftreten einer Variablen x in einer Formel A heißt **frei**, wenn es nicht gebunden ist und nicht unmittelbar rechts neben einem Quantor stattfindet.

$$\forall x(p_0(x, y) \rightarrow \forall z(\exists y p_1(y, z) \vee \forall x p_2(f(x), x)))$$

gebundene Vorkommen

freie Vorkommen

Definition

Es sei $A \in For_{\Sigma}$ und $t \in Term_{\Sigma}$.

$Bd(A) := \{x \mid x \in Var, x \text{ tritt gebunden in } A \text{ auf}\}$

$Frei(A) := \{x \mid x \in Var, x \text{ tritt frei in } A \text{ auf}\}$.

$Var(A) := Frei(A) \cup Bd(A)$

$Var(t) := \{x \mid x \in Var, x \text{ kommt in } t \text{ vor}\}$

Definition

A heißt *geschlossen*, wenn $\text{Frei}(A) = \{\}$.

Ist $\text{Frei}(A) = \{x_1, \dots, x_n\}$, so heißt

$\forall x_1 \dots \forall x_n A$ *Allabschluss*

$\exists x_1 \dots \exists x_n A$ *Existenzabschluss*

von A .

Abkürzend schreiben wir $C\forall A$ bzw. $C\exists A$.

Ist A geschlossen, dann gilt also $C\forall A = C\exists A = A$.

Definition: Substitutionen

Eine *Substitution* ist eine Abbildung

$$\sigma : \text{Var} \rightarrow \text{Term}_{\Sigma}$$

mit $\sigma(x) = x$ für fast alle $x \in \text{Var}$.

Notation für Substitutionen

Gilt

- ▶ $\{x \mid \sigma(x) \neq x\} \subseteq \{x_1, \dots, x_m\}$,
- ▶ und ist $\sigma(x_i) = s_i$ für $i = 1, \dots, m$,

so geben wir σ auch an in der Schreibweise

$$\{x_1/s_1, \dots, x_m/s_m\}.$$

σ heißt **Grundsubstitution**, wenn für alle x mit $\sigma(x) \neq x$ der Funktionswert $\sigma(x)$ ein Grundterm ist.

Mit *id* bezeichnen wir die **identische Substitution** auf *Var*, d.h. $id(x) = x$ für alle $x \in Var$.

Definition

Sei σ eine Substitution, t ein Term, ϕ eine Formel.

- ▶ $\sigma(t)$
entsteht aus t durch simultane Ersetzung aller Variablenvorkommen x durch $\sigma(x)$.
- ▶ $\sigma(\phi)$
entsteht aus ϕ durch simultane Ersetzung aller freien Variablenvorkommen x durch $\sigma(x)$.

Definition durch Beispiele

1. Für $\sigma = \{x/f(x, y), y/g(x)\}$ gilt

$$\sigma(f(x, y)) = f(f(x, y), g(x)).$$

2. Für $\mu = \{x/c, y/d\}$ gilt

$$\mu(\exists y p(x, y)) = \exists y p(c, y).$$

3. Für $\sigma_1 = \{x/f(x, x)\}$ gilt

$$\sigma_1(\forall y p(x, y)) = \forall y p(f(x, x), y).$$

4. Für $\mu_1 = \{x/y\}$ gilt

$$\mu_1(\forall y p(x, y)) = \forall y p(y, y).$$

Definition: kollisionsfreie Substitutionen

Eine Substitution σ heißt *kollisionsfrei* für eine Formel A , wenn für jede Variable z und jede Stelle freien Auftretens von z in A gilt:

Diese Stelle liegt nicht im Wirkungsbereich eines Präfixes $\forall x$ oder $\exists x$, wo x eine Variable in $\sigma(z)$ ist.

$$\mu_1 = \{x/y\} \text{ ist nicht kollisionsfrei für } \forall yp(x, y)$$

Definition: Komposition von Substitutionen

Sind σ, τ Substitutionen, dann definieren wir die Komposition von τ mit σ durch

$$(\tau \circ \sigma)(x) = \tau(\sigma(x)).$$

Man beachte, daß auf der rechten Seite τ als die Anwendung der Substitution τ auf den Term $\sigma(x)$ verstanden werden muß.

Theorem

1. Gilt für $t \in \text{Term}_\Sigma$ und Substitutionen σ, τ , die Gleichung $\sigma(t) = \tau(t)$,
dann $\sigma(s) = \tau(s)$ für jeden Teilterm s von t .

Beweis

1. Strukturelle Induktion nach t .
 - ▶ Ist $t \in \text{Var}$, dann ist t selbst sein einziger Teilterm.
 - ▶ Sei $t = f(t_1, \dots, t_n)$. Dann gilt

$$\begin{aligned}\sigma(f(t_1, \dots, t_n)) &= f(\sigma(t_1), \dots, \sigma(t_n)) \\ \tau(f(t_1, \dots, t_n)) &= f(\tau(t_1), \dots, \tau(t_n)).\end{aligned}$$

Theorem

1. Gilt für $t \in \text{Term}_\Sigma$ und Substitutionen σ, τ , die Gleichung $\sigma(t) = \tau(t)$,
dann $\sigma(s) = \tau(s)$ für jeden Teilterm s von t .

Beweis

1. Strukturelle Induktion nach t .
 - ▶ Ist $t \in \text{Var}$, dann ist t selbst sein einziger Teilterm.
 - ▶ Sei $t = f(t_1, \dots, t_n)$. Dann gilt auch

$$f(\sigma(t_1), \dots, \sigma(t_n)) = f(\tau(t_1), \dots, \tau(t_n)).$$

und es folgt $\sigma(t_i) = \tau(t_i)$ für $i = 1, \dots, n$. Da jeder Teilterm s von t entweder mit t identisch oder Teilterm eines t_i ist, folgt 1. nach Induktionsvoraussetzung.

Theorem

1. *Gilt für $t \in \text{Term}_\Sigma$ und Substitutionen σ, τ , die Gleichung $\sigma(t) = \tau(t)$,
dann $\sigma(s) = \tau(s)$ für jeden Teilterm s von t .*
2. *Wenn $\sigma(t) = t$, dann $\sigma(s) = s$ für jeden Teilterm s von t .*

Beweis

1. Strukturelle Induktion nach t .
2. Spezialfall von 1.

Theorem

Gilt für Substitutionen σ, τ , daß $\tau \circ \sigma = id$, dann ist σ eine Variablenumbenennung.

Beweis

Es ist $\tau(\sigma(x)) = x$ für jedes $x \in Var$, woraus folgt: $\sigma(x) \in Var$.

Ferner haben wir:

Wenn $\sigma(x) = \sigma(y)$, dann $x = \tau(\sigma(x)) = \tau(\sigma(y)) = y$.

Definition

Es sei $T \subseteq \text{Term}_\Sigma$, $T \neq \{\}$, und σ eine Substitution über Σ .

σ **unifiziert** T ,

oder: σ ist **Unifikator von T** ,

genau dann, wenn $\#\sigma(T) = 1$.

T heißt **unifizierbar**, wenn T einen Unifikator besitzt.

Insbesondere sagen wir für zwei Terme s, t daß s *unifizierbar* sei *mit* t , wenn

$$\sigma(t) = \sigma(s).$$

$$\{f(g(a, x), g(y, b)), f(z, g(v, w)), f(g(x, a), g(v, b))\}$$

wird unifiziert durch

$$\{x/a, y/v, z/g(a, a), w/b\}.$$

$$\{f(g(a, x), g(y, b)), f(z, g(v, w)), f(g(x, a), g(v, b))\}$$

wird unifiziert durch

$$\{x/a, y/v, z/g(a, a), w/b\}.$$

$$\{f(g(a, x), g(y, b)), f(z, g(v, w)), f(g(x, a), g(v, b))\}$$

wird unifiziert durch

$$\{x/a, y/v, z/g(a, a), w/b\}.$$

$$\{f(g(a, a), g(v, b)), f(g(a, a), g(v, b)), f(g(a, a), g(v, b))\}$$

$$\{f(g(a, x), g(y, b)), f(z, g(v, w)), f(g(x, a), g(v, b))\}$$

wird unifiziert durch

$$\{x/a, y/v, z/g(a, a), w/b\}.$$

$$\begin{aligned} & f(g(a, a), g(v, b)), \\ & \{ f(g(a, a), g(v, b)), \} \\ & f(g(a, a), g(v, b)) \end{aligned}$$

1. Jeder Term ist mit sich selbst unifizierbar mittels *id*.
2. Zwei Terme der Gestalt

$$f(s_1, \dots, s_n), g(t_1, \dots, t_n)$$

(mit verschiedenem Kopf) sind nicht unifizierbar.

3. Zwei Terme der Gestalt

$$f(s_1, \dots, s_n), f(t_1, \dots, t_n)$$

(mit demselben Kopf) sind genau dann unifizierbar, wenn es eine Substitution σ gibt mit $\sigma(s_i) = \sigma(t_i)$ für $i = 1, \dots, n$.

4. Sei $x \in \text{Var}$ und t ein Term. Dann sind

x und t

genau dann unifizierbar, wenn x **nicht** in t vorkommt.

$$\{f(x, g(y)), f(g(a), g(z))\}$$

wird unifiziert durch

$$\sigma = \{x/g(a), z/y\} \quad \text{Ergebnis } f(g(a), g(y)),$$

aber auch durch

$$\tau = \{x/g(a), y/a, z/a\} \quad \text{Ergebnis } f(g(a), g(a)).$$

σ ist allgemeiner als τ – oder τ spezieller als σ –

$$\tau = \{y/a\} \circ \sigma.$$

Definition

Es sei $T \subseteq \text{Term}_\Sigma$.

Ein *allgemeinster Unifikator* oder mgu (*most general unifier*) von T ist eine Substitution μ mit

1. μ unifiziert T
2. Zu jedem Unifikator σ von T gibt es eine Substitution σ' mit $\sigma = \sigma' \circ \mu$.

Eindeutigkeit des allgemeinsten Unifikators

Theorem

Es sei T eine unifizierbare, nichtleere Menge von Termen. Dann ist der allgemeinste Unifikator von T bis auf Variablenumbenennung eindeutig bestimmt, d. h.:

Sind μ, μ' allgemeinste Unifikatoren von T mit

$$\mu(T) = \{t\} \text{ und } \mu'(T) = \{t'\},$$

dann gibt es eine Umbenennung π der Variablen von t mit

$$t' = \pi(t).$$

Nach der Definition gibt es Substitutionen σ, σ' mit

- ▶ $\mu' = \sigma\mu$
- ▶ $\mu = \sigma'\mu'$

Daraus folgt

$$\begin{aligned}\mu(T) = t &= \sigma'\mu'(T) = \sigma'\sigma\mu(T) = \sigma'\sigma(t) \\ \text{d.h. } t &= \sigma'\sigma(t)\end{aligned}$$

Das kann nur sein, wenn für jede Variable $x \in \text{Var}(t)$ gilt

$$\sigma'\sigma(x) = x.$$

Daraus folgt insbesondere, daß für jedes $x \in \text{Var}(t)$ $\sigma(x)$ wieder eine Variable sein muß und für $x, y \in \text{Var}(t)$ mit $x \neq y$ auch $\sigma(x) \neq \sigma(y)$ gilt.

Unifikationsalgorithmus von Robinson

Positionsnotation

Definition

Zu $t \in \text{Term}_\Sigma$ und $i \in \mathbb{N}$ sei

$t^{(i)}$ = der an Position i in t (beim Lesen von links nach rechts) beginnende Teilterm von t , wenn dort eine Variable oder ein Funktionssymbol steht
undefiniert sonst.

Unifikationsalgorithmus von Robinson

Differenzenmenge

Definition

Für $T \subseteq \text{Term}_\Sigma$ ist die *Differenz* von T , $D(T) \subseteq \text{Term}_\Sigma$, wie folgt definiert

1. $D(T) := T$ falls $\#T \leq 1$
2. Falls $\#T \geq 2$, sei j die kleinste Zahl, so daß sich zwei Terme aus T an der Position j unterscheiden.
Setze $D(T) := \{t^{(j)} \mid t \in T\}$.

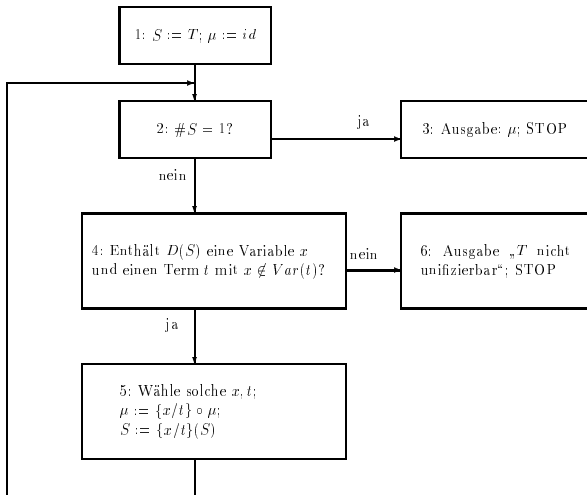
•

Beispiel

$$T = \{f(g(a, x), g(y, b)), f(z, g(v, w)), f(g(x, a), g(v, b))\}$$
$$D(T) = \{g(a, x), z, g(x, a)\}$$

Algorithmus von Robinson

Gegeben sei $T \subseteq Term_{\Sigma}$, T endlich und $\neq \emptyset$.



Theorem

1. *Der Algorithmus von ROBINSON terminiert für jedes endliche, nichtleere $T \subseteq \text{Term}_\Sigma$.*
2. *Wenn T unifizierbar ist, liefert er einen allgemeinsten Unifikator von T .*
3. *Wenn T nicht unifizierbar ist, liefert er die Ausgabe „ T nicht unifizierbar“.*

Wir zeigen

1. Der Algorithmus terminiert.
2. Wenn er eine Substitution μ ausgibt, dann ist μ Unifikator von T .
3. Ist σ ein beliebiger Unifikator von T , dann gibt es μ, σ' so daß
 - ▶ der Algorithmus mit Ausgabe μ terminiert,
 - ▶ $\sigma = \sigma' \circ \mu$
 - ▶ *Somit:* μ ist mgu von T .
4. Wenn der Algorithmus ausgibt „ T nicht unifizierbar“, dann ist T nicht unifizierbar.

Beweis

Notation

Unter einem *Schleifendurchlauf* in dem obigem Algorithmus verstehen wir einen *vollständigen* Durchlauf der Befehlsfolge 2–4–5.

Wir setzen

- ▶ $S_0 := T, \mu_0 := id$
- ▶ $S_{k+1} :=$ Wert von S nach dem $(k + 1)$ -ten Schleifendurchlauf
- ▶ $\mu_{k+1} := \mu$ nach dem $(k + 1)$ -ten Schleifendurchlauf
- ▶ x_k, t_k die im $(k + 1)$ -ten Durchlauf gewählten x, t .

Im $(k + 1)$ -ten Schleifendurchlauf gilt

$$S_{k+1} = \{x_k/t_k\}(S_k).$$

Dabei kommt x_k nicht in t_k vor.

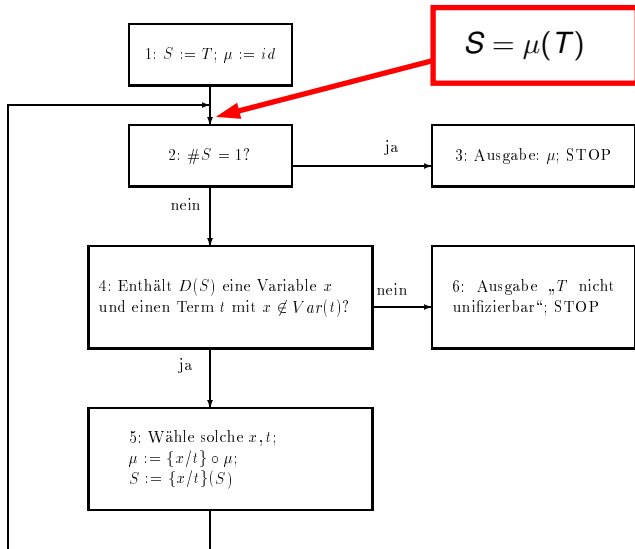
Nach Anwendung der Substitution $\{x_k/t_k\}$ gibt es in S_{k+1} kein Vorkommen der Variable x_k mehr.

Da t_k selbst ein Term in S_k war, werden durch die Substitution auch keine neuen Variablen eingeführt.

Also: Beim Übergang von S_k zu S_{k+1} vermindern sich die Variablen in S_{k+1} genau um x_k . Die Schleife terminiert nach endlichen vielen Durchläufen.

Algorithmus von Robinson

1. Schleifeninvariante



Die Invariante besagt für alle $k \leq m$:

$$S_k = \mu_k(T)$$

Hält das Programm mit Ausgabe einer Substitution μ an, dann hat μ den Wert μ_m , und es ist

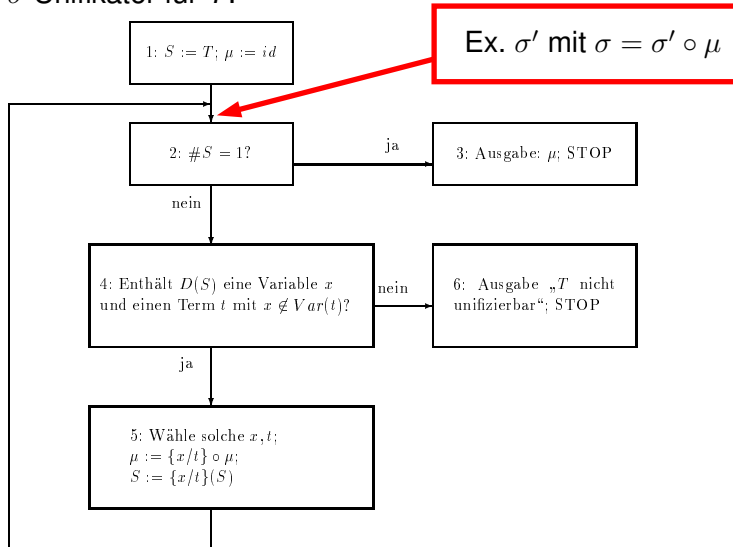
$$\#\mu_m(T) = \#S_m = 1.$$

μ_m ist Unifikator von T .

Algorithmus von Robinson

2. Schleifeninvariante

σ Unifikator für T .



Beweis

Ausgabe ist allgemeiner als vorgegebener Unifikator σ

Es sei σ ein Unifikator von T .

Behauptung: Für alle k gibt es σ_k mit $\sigma = \sigma_k \circ \mu_k$.

$k = 0$: Setze $\sigma_0 := \sigma$.

$k + 1$: Nach Induktionsannahme existiert σ_k mit $\sigma = \sigma_k \circ \mu_k$.

Wir haben

$$\#\sigma_k(S_k) = \#\sigma_k(\mu_k(T)) = \#\sigma(T) = 1$$

da σ Unifikator von T ist.

Im $(k + 1)$ -ten Durchlauf wird Test 2 mit „Nein“ und Test 4 mit „Ja“ verlassen: es ist $\#S_k \geq 2$, und in $D(S_k)$ gibt es x_k, t_k mit $x_k \notin \text{Var}(t_k)$.

Da σ_k Unifikator von S_k ist, muß gelten $\sigma_k(x_k) = \sigma_k(t_k)$.

Beweis (Forts.)

Wir setzen

$$\sigma_{k+1}(x) = \begin{cases} \sigma_k(x) & \text{falls } x \neq x_k \\ x_k & \text{falls } x = x_k. \end{cases}$$

Falls $x \neq x_k$ $\sigma_{k+1}(\{x_k/t_k\}(x)) =$
 $\sigma_{k+1}(x) =$
 $\sigma_k(x),$

Falls $x = x_k$ $\sigma_{k+1}(\{x_k/t_k\}(x)) =$
 $\sigma_{k+1}(\{x_k/t_k\}(x_k)) =$
 $\sigma_{k+1}(t_k) =$
 $\sigma_k(t_k) =$ da $x_k \notin \text{Var}(t_k)$
 $\sigma_k(x_k) = \sigma_k(x).$

Somit $\sigma_{k+1} \circ \{x_k/t_k\} = \sigma_k.$

Es folgt: $\sigma_{k+1} \circ \mu_{k+1} = \sigma_{k+1} \circ \{x_k/t_k\} \circ \mu_k$
 $= \sigma_k \circ \mu_k = \sigma.$ d. h. (*).

Insbesondere gilt: $\sigma = \sigma_m \circ \mu_m.$

σ_m unifiziert S_m (da σ T unifiziert).

Also muß $D(S_m)$ eine Variable x und einen Term t enthalten mit $x \notin \text{Var}(t)$

Die Antwort auf Test 2 muß also „Ja“ sein.