

RECHENSTRUKTUREN

Kurzbeschreibung

Rechenstrukturen sind das Informatik-Gegenstück zur Algebra und führen zu den Termen und Termersetzungssystemen. Es werden die Terme der Aussagen- und der Prädikatenlogik behandelt.

Schlüsselwörter

Algebra, Rechenstruktur, Signatur, Natürliche Zahlen, Ganze Zahlen, Boolesche Algebra, NAT, BOOL, Term, Grundterm, Termersetzung, Aussagenlogik, Prädikatenlogik, Quantoren

Lernziele

1. Die Bedeutung von Rechenstrukturen als ein zentrales Verbindungselement zwischen der Mathematik und der Informatik wird verstanden.
2. Der Umgang mit Termen und mit Systemen zur Termersetzung wird beherrscht.
3. Die Grundlagen der mathematischen Logik und der daraus hervorgehenden Rechenstrukturen der Aussagenlogik und der Prädikatenlogik können nachvollzogen werden.

Hauptquellen

- Manfred Broy: Informatik – Eine grundlegende Einführung, Band 1: Programmierung und Rechenstrukturen, Springer Verlag 1998.

Inhaltsverzeichnis

1	AUFBAU UND BEISPIELE VON RECHENSTRUKTUREN	2
1.1	Definition einer Rechenstruktur	3
1.2	Signatur und Signaturdiagramm	5
1.3	Grundterme und Termalgebra	7
1.4	Identifikatoren	9
2	TERMERSETZUNG	13
2.1	Termersetzungsregel und -algorithmus	14
2.2	Termersetzungssystem	16
3	MATHEMATISCHE LOGIK	18
3.1	Aussagenlogik	20
3.2	Prädikatenlogik	24
	VERZEICHNISSE	29
	Abkürzungen und Glossar	29
	Index	30
	Informationen und Interaktionen	30
	Literatur	31

- AUFBAU UND BEISPIELE VON RECHENSTRUKTUREN
 - Signatur, Operationen, Kantorowitsch-Baum
- TERMERSSETZUNG
 - Termersetzungsregel, Termersetzungs-system, Algorithmus, Korrektheit
- MATHEMATISCHE LOGIK
 - Formel, Aussagenlogik, Prädikatenlogik

Information 1: RECHENSTRUKTUREN

1 AUFBAU UND BEISPIELE VON RECHENSTRUKTUREN

Die Rechenstruktur ist das Gebilde, das die Algebra und die darin festgelegten Strukturen in der Informatik verankert. Mittels Rechenstrukturen wird somit ein Übergang von der Algebra hin zu den Algorithmen und Programmen geschaffen [Br98].

- Algorithmen
 - arbeiten über Datenelementen, den Trägermengen
 - bestehen aus Operationen auf den Trägermengen
- Trägermengen und Operationen lassen sich zu Rechenstrukturen zusammenfassen
 - entsprechen dem Begriff der Algebra
- Beispiele, die als Rechenstruktur formuliert werden können
 - Taschenrechner
 - leistungsstarke Rechenanlage
- Definition: Rechenstruktur
Seien S und F Mengen von Bezeichnungen. Eine Rechenstruktur A besteht aus einer Familie $\{s^A: s \in S\}$ von Trägermengen s^A und einer Familie $\{f^A: f \in F\}$ von Abbildungen f^A zwischen diesen Trägermengen.

Information 2: AUFBAU UND BEISPIELE VON RECHENSTRUKTUREN – Definition einer Rechenstruktur

Die Operationen, die in einem Algorithmus genutzt werden, hängen unmittelbar mit den Trägermengen zusammen, da sie Abbildungen zwischen diesen schaffen. Für die Begriffe der Operationen bzw. der Abbildungen benutzen Mathematiker und Informatiker auch beide den Begriff der Funktion.

Das Gebilde der Rechenstruktur ist äußerst mächtig. Es ermöglicht die Beschreibung beliebig komplexer Sachverhalte und Systeme. So lassen sich einfachere Systeme wie ein

Taschenrechner genauso mittels einer Rechenstruktur beschreiben wie auch das komplexeste Rechensystem.

In Information 2 werden eine formale Definition einer Rechenstruktur und eine gewisse Notation eingeführt.

www.cm-tm.uka.de/info1
Info1-Team (Prof. Abeck)

- Sorten S sind Bezeichnungen für Trägermengen
- Die Elemente $f \in F$ heißen Funktionssymbole oder Operationssymbole
- Einführung eines speziellen Elements \perp ("Bottom"-Element)
 - repräsentiert nicht definierte Funktionswerte, wodurch partielle Abbildungen vermieden werden
 - eine Menge M , die nicht \perp enthält, wird um das Bottom-Element ergänzt

$$M^\perp =_{\text{def}} M \cup \{\perp\}$$

- Eine Funktion $f: M_1^\perp \times M_2^\perp \times \dots \times M_n^\perp \rightarrow M_{n+1}^\perp$ heißt strikt, wenn gilt:
Ist eines der Argumente \perp , so ist auch das Resultat der Funktion \perp

Information 3: Strikte Funktion

Die Trägermengen-Elemente s werden auch als Sorten bezeichnet, die Funktionselemente f als Funktionssymbole.

Eine besondere Aufmerksamkeit ist bei den Rechenstrukturen dem Fall zu widmen, dass eine Funktion an gewissen Stellen kein Ergebnis liefert, also undefiniert ist. Zu diesem Zweck wird das Zeichen \perp ("Bottom") eingeführt, dessen Semantik „undefinierter Wert“ ist. Auf diese Problematik wird auch im Kontext mit der Programmierung in der Kurseinheit IMPERATIVE PROGRAMMIERUNG [C&M-IP] eingegangen.

Formal lässt sich eine beliebige Trägermenge M um dieses Zeichen ergänzen und entsprechend durch ein hochgestelltes \perp anzeigen. Angewendet auf Rechenstrukturen, können dadurch partielle Funktionen vermieden werden.

Im Zusammenhang mit der Einführung des Zeichens \perp steht eine Eigenschaft von Rechenstrukturen, die als strikt bezeichnet wird. Diese Eigenschaft stellt eine bestimmte Anforderung an Argumente und das Ergebnis im Hinblick auf das Undefiniert-Zeichen, nämlich dass aus der Undefiniertheit von nur einem Argument die Undefiniertheit des Ergebnisses folgt.

1.1 Definition einer Rechenstruktur

Die Definition einer Rechenstruktur geschieht in der Weise, dass in einem ersten Schritt die drei relevanten Mengen (Sorten, Funktionssymbole und Trägermengen) festgelegt werden und anschließend die Funktionssymbole näher beschrieben werden.

- Menge S der Sorten: $S = \{\text{bool}\}$
- Menge F der Funktionssymbole: $F = \{\text{true, false, } \neg, \vee, \wedge\}$
- Zuordnung Sorten zu Trägermengen $\text{bool}^{\text{Bool}} = \mathcal{B}^{\perp}$

- Zuordnung von Funktionen zu den Funktionssymbolen
($a, b \in \mathcal{B}$) Stelligkeit Schreibweise

$\text{true}^{\text{Bool}}: \rightarrow \mathcal{B}^{\perp}$	$\text{true}^{\text{Bool}} = \text{L},$	<u> </u> <u> </u>
$\text{false}^{\text{Bool}}: \rightarrow \mathcal{B}^{\perp}$	$\text{false}^{\text{Bool}} = \text{O},$	<u> </u> <u> </u>
$\neg^{\text{Bool}}: \mathcal{B}^{\perp} \rightarrow \mathcal{B}^{\perp}$	$\neg^{\text{Bool}} b = \mathbb{C} b,$	<u> </u> <u> </u>
$a \vee^{\text{Bool}} b: \mathcal{B}^{\perp} \times \mathcal{B}^{\perp} \rightarrow \mathcal{B}^{\perp}$	$a \vee^{\text{Bool}} b = a \vee b$	<u> </u> <u> </u>
$a \wedge^{\text{Bool}} b: \mathcal{B}^{\perp} \times \mathcal{B}^{\perp} \rightarrow \mathcal{B}^{\perp}$	$a \wedge^{\text{Bool}} b = a \wedge b$	<u> </u> <u> </u>

- Die in der Rechenstruktur BOOL festgelegten Funktionen sind strikt

Interaktion 1: Rechenstruktur BOOL der booleschen Werte

Schritt 1: Festlegung der Sorten, der Funktionssymbole und der Trägermengen der Rechenstruktur

Die Schreibweise $\text{bool}^{\text{Bool}}$ ist eine konkrete Ausprägung der eingeführten Notation s^A zur Festlegung der Trägermengen von Rechenstrukturen

Schritt 2: Beschreibung der Funktionssymbole durch Festelegung der Definitions- und Wertemenge und Zuordnung der Funktionen

Es werden die im Zusammenhang mit der booleschen Algebra \mathcal{B} auftretenden Funktionssymbole genutzt. Die Stelligkeit und die Schreibweise (Präfix, Infix, Postfix) ist in Interaktion 1 zu jeder der in der Rechenstruktur BOOL gehörenden Funktionen anzugeben.

- Menge S der Sorten: $S = \{\text{bool, nat}\}$
- Menge F der Funktionssymbole: $F = \{\text{true, false, } \neg, \wedge, \vee, \text{zero, succ, pred, add, mult, sub, div, } \leq, \neq\}$
- Zuordnung Sorten zu Trägermengen: $\text{bool}^{\text{NAT}} = \mathcal{B}^{\perp}, \text{nat}^{\text{NAT}} = \mathbb{N}^{\perp}$
- Zuordnung von Funktionen zu den Funktionssymbolen ($x, y \in \mathbb{N}$):

$\text{zero}^{\text{NAT}}: \rightarrow \mathbb{N}^{\perp}$ $\text{succ}^{\text{NAT}}: \rightarrow \mathbb{N}^{\perp} \rightarrow \mathbb{N}^{\perp}$ $\text{pred}^{\text{NAT}}: \rightarrow \mathbb{N}^{\perp} \rightarrow \mathbb{N}^{\perp}$ $\text{add}^{\text{NAT}}: \mathbb{N}^{\perp} \times \mathbb{N}^{\perp} \rightarrow \mathbb{N}^{\perp}$ $\text{mult}^{\text{NAT}}: \mathbb{N}^{\perp} \times \mathbb{N}^{\perp} \rightarrow \mathbb{N}^{\perp}$ $\text{sub}^{\text{NAT}}: \mathbb{N}^{\perp} \times \mathbb{N}^{\perp} \rightarrow \mathbb{N}^{\perp}$ $\text{div}^{\text{NAT}}: \mathbb{N}^{\perp} \times \mathbb{N}^{\perp} \rightarrow \mathbb{N}^{\perp}$ $\leq^{\text{NAT}}: \mathbb{N}^{\perp} \times \mathbb{N}^{\perp} \rightarrow \mathcal{B}^{\perp}$ $\neq^{\text{NAT}}: \mathbb{N}^{\perp} \times \mathbb{N}^{\perp} \rightarrow \mathcal{B}^{\perp}$	$\text{zero}^{\text{NAT}} = 0,$ $\text{succ}^{\text{NAT}}(x) = x+1,$ $\text{pred}^{\text{NAT}}(x) = x-1, \text{ falls } x \geq 1$ $\text{pred}^{\text{NAT}}(0) = \perp,$ $\text{add}^{\text{NAT}}(x, y) = x+y$ $\text{mult}^{\text{NAT}}(x, y) = x*y$ $\text{sub}^{\text{NAT}}(x, y) = x-y,$ $\text{div}^{\text{NAT}}(x, y) = x \div y, \text{ falls } y > 0$ $\text{div}^{\text{NAT}}(x, 0) = \perp,$ $0 \leq^{\text{NAT}} x = \text{L}, (x+1 \leq^{\text{NAT}} 0) = \text{O}$ $(x+1 \leq^{\text{NAT}} y+1) = (x \leq^{\text{NAT}} y),$ $(x \neq^{\text{NAT}} y) = (x \leq^{\text{NAT}} y) \wedge^{\text{NAT}} (y \leq^{\text{NAT}} x)$
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Information 4: Rechenstruktur NAT der natürlichen Zahlen

Die Rechenstruktur NAT benutzt die Sorte `bool` und fügt dieser die Sorte `nat` hinzu. Die Sorte `bool` wird im Zusammenhang mit den letzten zwei Funktionssymbolen `≤` und `=` benötigt, da diese einen booleschen Wahrheitswert als Ergebnis liefern.

Die in der Rechenstruktur auftretenden Funktionssymbole stützen sich auf die bekannten Operatoren, die auf natürlichen Zahlen angewendet werden können.


Bei der Operation `pred`, die den Vorgänger (*Predecessor*) zum Eingabeargument liefert, muss der Bereich, in dem die Funktion definiert ist, explizit bei der Spezifikation angegeben werden, indem zur 0 als Vorgänger \perp zugewiesen wird. Entsprechendes gilt für die Division durch 0.

Durch die Angabe der Gleichungen wird die Wirkungsweise und somit die Semantik einer Operation beschrieben. Die Operationen werden auf entsprechende Funktionen der genutzten Algebren zurückgeführt. Das gilt z.B. für die 0-stellige Operation `zero` genauso wie für die 1-stellige Operation `succ` oder die 2-stellige Operation `add`.

Für alle Funktionen gilt: Falls nur ein Eingabeargument \perp (also undefiniert) ist, so ist auch das Ergebnis \perp .

1.2 Signatur und Signaturdiagramm

Durch eine Signatur wird festgelegt, in welcher Weise die Funktionssymbole mit den Sorten sinnvoll verknüpft werden können.

www.cm-tm.uka.de/info1
Info1-Team (Prof. Abeck) 

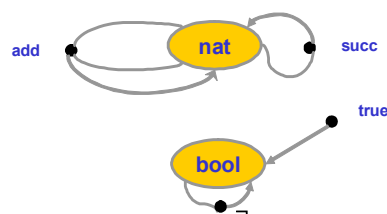
- In Anlehnung an den Signatur-Begriff von Algebren wird die Signatur zu einer Rechenvorschrift als ein Paar (S, F) von Mengen S und F eingeführt
 - S wird als die Menge der Sorten bezeichnet
 - F wird als die Menge der Funktionssymbole bezeichnet
- $fct\ f \in S^+$ beschreibt die Funktionalität zu jedem $f \in F$
 - Schreibweise: $fct\ f = (s_1, \dots, s_n) s_{n+1}$
 - [Mathematische Schreibweise](#)

Interaktion 2: Signatur einer Rechenvorschrift

Eine Funktion ist dabei Element eines über den Sorten gebildeten n -Tupels. Das in Interaktion 2 verwendete Symbol S^+ drückt aus, dass $n \geq 1$ ist. Im Falle von $n = 1$ handelt es sich um eine 0-stellige (konstante) Funktion. Die in Verbindung mit `fct` benutzte Funktionsschreibweise entspricht einer mathematischen Schreibweise, die in Interaktion 2 nachgefragt wird.



- Beispiel: Signatur der Rechenstruktur NAT
 - $S_{\text{NAT}} = \{\text{bool}, \text{nat}\}$
 - $F_{\text{NAT}} = \{\text{true}, \text{false}, \neg, \wedge, \vee, \text{zero}, \text{succ}, \text{pred}, \text{add}, \text{mult}, \text{sub}, \text{div}, \leq, =\}$
 - $\text{fct true} = \text{bool}$ $\text{fct } \neg = (\text{bool}) \text{ bool}$
 - $\text{fct succ} = (\text{nat}) \text{ nat}$ $\text{fct add} = (\text{nat}, \text{nat}) \text{ nat}$
 - ...
- Signaturen lassen sich graphisch in Form von Signaturdiagrammen übersichtlich darstellen
 - Beispiel: Signaturdiagramm zur Rechenstruktur NAT (zu ergänzen)



Interaktion 3: Signatur und Signaturdiagramm zur Rechenstruktur NAT

Die umfangreiche textuelle Beschreibung lässt sich graphisch durch ein Signaturdiagramm darstellen. Es handelt sich hierbei um einen speziellen Graphen mit zwei Arten von Ecken zur Repräsentation der an der Rechenstruktur beteiligten Sorten und Funktionen. Weiterführende Beschreibungen zu Graphen finden sich in der Kurseinheit ALGEBRAISCHE GRUNDLAGEN [C&M-AG].

Die Funktionalitäten zu jeder beteiligten Funktion werden durch die Kanten dargestellt, wie in Interaktion 3 am Beispiel von 0-, 1- und 2-stelligen Funktionen verdeutlicht wird. Die in dieser Interaktion gestellte Aufgabe besteht darin, das Signaturdiagramm gemäß der Signatur von NAT zu vervollständigen.

- Die Signatur INT der ganzen Zahlen stimmt bis auf Umbenennung von Sorten mit der Signatur NAT der natürlichen Zahlen überein
 - $S = \{\text{bool}, \text{int}\}$
 - $\text{int}^{\text{INT}} = \mathbb{Z}$
 - $F = \{\text{succ}, \text{pred}, \text{zero}, \dots\}$
 - $\text{pred}^{\text{INT}}(z) = z-1$
 - für alle z
 - beachte: $\text{pred}^{\text{NAT}}(x) = x-1$, falls $x \geq 1$
 - übrige Funktionen wie Rechenstruktur NAT
 - die Gleichungen, durch die die Wirkungsweise beschrieben ist, sind entsprechend anzupassen
- Die Signaturen von NAT und INT machen deutlich, dass sich mit der gleichen Signatur unterschiedliche Rechenstrukturen verbinden lassen
 - eine Signatur charakterisiert eine Rechenstruktur nicht eindeutig

Information 5: Rechenstruktur der ganzen Zahlen INT


Die für die natürlichen Zahlen NAT erstellte Signatur und das dazugehörige Signaturdiagramm lassen sich ohne strukturelle Änderungen – d.h. bis auf Umbenennungen – zur Beschreibung der Rechenstruktur der ganzen Zahlen INT verwenden. NAT und INT besitzen die gleichen syntaktischen Strukturen.

Der einzige (nicht an der Signatur erkennbare) Unterschied tritt bei der Vorgängerfunktion pred auf. Hier muss jetzt nicht mehr das Eingabeargument auf 0 hin überprüft werden, da ganze Zahlen immer einen Vorgänger haben. Die Bedingungen, wie sie für die Vorgänger-Funktion im Zusammenhang mit den natürlichen Zahlen definiert wurde, kann somit ersatzlos gestrichen werden.

Die korrekten aus einer Signatur ableitbaren Ausdrücke werden als Terme bezeichnet. Im Folgenden werden die Terme genauer betrachtet, weil hieraus eine für die Informatik besonders wichtige Struktur, die Termalgebra, resultiert.

1.3 Grundterme und Termalgebra

Die Menge der Grundterme zu allen Sorten einer Signatur Σ wird mit W_Σ bezeichnet.

www.cm-tm.uka.de/info1
Info1-Team (Prof. Abeck) 

- Zu einer Signatur $\Sigma = (S, F)$ ist die Menge der Grundterme W_Σ^s der Sorte s mit $s \in S$ definiert durch
 - (1) jedes nullstellige Funktionssymbol $f \in F$ mit $\text{fct } f = s$ bildet einen Grundterm der Sorte s
 - (2) jede Zeichenreihe $f(t_1, \dots, t_n)$ mit $f \in F$ und $\text{fct } f = (s_1, \dots, s_n)$ s ist ein Grundterm der Sorte s , falls für alle i , $1 \leq i \leq n$, t_i ein Grundterm der Sorte s_i ist
- Es ist ein Grundterm zur Signatur der Rechenstruktur NAT zu formulieren, in dem die Funktionen
zero, true, \neg , succ, add
vorkommen

- Welche Schreibweise wurde benutzt? _____

Interaktion 4: Grundterme

Die Menge W_Σ zerfällt in disjunkte Teilmengen W_Σ^s der Sorte s mit $s \in S$. In Interaktion 4 wird das Bildungsgesetz eines Grundterms beschrieben. Gemäß diesem Bildungsgesetz ist ein die angegebenen Anforderungen erfüllender Grundterm zu formulieren und die hierzu verwendete Schreibweise ist anzugeben.



- Grundterme lassen sich in einer Rechenstruktur A mit Signatur Σ interpretieren
 - $I^A: W_\Sigma \rightarrow \{a \in s^A: s \in S\}$
 - $I^A[t]$ (oder kurz t^A) bezeichnet die Interpretation des Grundterms t in A
 - $I^A[f(t_1, \dots, t_n)] = f^A(I^A[t_1], \dots, I^A[t_n]) = f^A(t_1^A, \dots, t_n^A)$
- Beispiel: Interpretation von Grundtermen über der Signatur NAT
 - $I^{\text{NAT}}[\text{pred}(\text{succ}(\text{zero}))] =$ _____

 - $I^{\text{NAT}}[\text{pred}(\text{zero})] =$ _____

Interaktion 5: Interpretation von Grundtermen

Beim Grundterm t wird davon ausgegangen, dass es sich um eine Repräsentation der Sorte s handelt, wobei s ein Element der Sortenmenge S der Rechenstruktur A ist.

Im folgenden Beispiel in Interaktion 5 ist das Vorgehen der Interpretation auf die zwei Grundterme $\text{pred}(\text{succ}(\text{zero}))$ und $\text{pred}(\text{zero})$ über der Signatur NAT anzuwenden. Durch die Interpretation I wird ein Grundterm der Sorte nat auf ein Element der Trägermenge abgebildet.



- Eine Rechenstruktur A heißt termerzeugt, wenn für jedes Element a der Trägermengen von A eine Termrepräsentation existiert
 - d.h. ein Grundterm der Sorte s mit $t^A = a$
 - die Interpretationsabbildung ist in diesem Fall surjektiv
- Ist die Rechenstruktur NAT termerzeugt? ja nein
- Falls ja, ist eine Termrepräsentation anzugeben

Interaktion 6: Termerzeugte Rechenstruktur

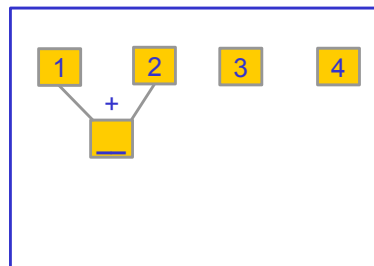
Eine Rechenstruktur wird als termerzeugt bezeichnet, wenn es für jedes Element a einer Trägermenge (s^A) einen Grundterm zur Sorte s gibt, der durch Interpretation auf dieses Element abgebildet wird ($t^A = a$). Diese Eigenschaft besagt gerade, dass das Bild der Interpretationsfunktion, also die Trägermenge vollständig erfasst wird und somit kein Element der Trägermenge ausgespart wird. Die Anforderung wird gerade von den surjektiven Abbildungen erfüllt.

In Interaktion 6 wird die Aufgabe gestellt, eine solche surjektive Abbildung für die Rechenstruktur NAT zu finden.

- Die Interpretation, also der Wert eines Grundterms, lässt sich entsprechend seiner Termstruktur ausrechnen
- Geeignetes Hilfsmittel zur Berechnung sind Formulare
 - graphische Darstellung für die Berechnung der Interpretation
 - Einführung von Rechtecken, die jeweils die Interpretation eines Teilterms beinhalten
 - Verknüpfen der Rechtecke durch Kanten entlang des Termaufbaus

- Beispiel: Grundterm $((1 + 2) \cdot 3) - 4$

Das Formular ist zu vervollständigen



Interaktion 7: Formulare

Mit der Interpretations-Abbildung ist eine Wertezuordnung verbunden. Es stellt sich die Frage, wie man diese Wertezuordnung durch ein systematisches Berechnungsprinzip durchführen kann. Eine Antwort liefern die in Interaktion 7 eingeführten Formulare.

Ein Formular ist formal betrachtet ein Baum, also ein Graph ohne Zyklen mit einer ausgezeichneten Wurzel. Die Knoten werden als Rechtecke gezeichnet und repräsentieren entsprechende Teilterme bzw. Wurzeln von Teilbäumen, die ihrerseits Teilterme repräsentieren.

1.4 Identifikatoren

Bei den gerade behandelten Beispiel-Grundtermen waren die in den Termen auftretenden Trägermengen-Elemente mit festem Wert vorgegeben. Nur deshalb konnte man die Berechnung im Formular überhaupt durchführen.

- Identifikatoren sind Platzhalter für Terme (oder Elemente), die später an der entsprechenden Stelle eingesetzt werden können
 - Identifikatoren heißen auch Bezeichner, Variable oder Unbekannte
 - sie können als Namen für erst später genau bezeichnete Terme (oder Elemente) verstanden werden
- Ergänzung der Signatur $\Sigma = (S, F)$ um eine Menge $X = \{X_s: s \in S\}$ von Identifikatoren
 - führt zu einer um X erweiterten Termalgebra $W_{\Sigma}(X)$
 - wird auch als $W_{\Sigma'}$ bezeichnet mit $\Sigma' = (S, F \cup \{x \in X_s: s \in S\})$ und $\text{fct } x = s \text{ für } x \in X_s$
- Beispiele:
 - Gleichung mit Unbekannten z.B. $ax^2 + bx + c = 0$
 - Funktionsdefinition z.B. $f: \mathbb{N} \rightarrow \mathbb{N}$ mit $f(x) = 2x + 1$

Information 6: Terme mit (freien) Identifikatoren

In vielen Anwendungsfällen wünscht man sich bezüglich der Zuordnung von Werten zu den im Term auftretenden Elementen mehr Flexibilität. Das führt zu den Identifikatoren, die Platzhalter für Terme oder Elemente darstellen.

Formal bedeutet die Einführung von Identifikatoren eine Ergänzung der Signatur um eine Menge von Identifikatoren X_s der Sorte s . Die resultierende Termalgebra wird als $W_{\Sigma'}$ bezeichnet. Zu einer Funktion lässt sich dann entsprechend ein Identifikator hinzufügen. Die Erweiterung auf mehrere Identifikatoren ist problemlos möglich.

Ein Beispiel einer Funktion mit freiem Identifikator ist die in Information 6 angegebene Formel. Terme mit freien Identifikatoren werden auch als Polynome bezeichnet. Das zweite Beispiel zeigt einen Term $2x + 1$ innerhalb einer Funktionsdefinition.

- Identifikatoren in Termen können durch andere Terme ersetzt werden
 - diese Abbildung wird als Substitution bezeichnet
 - $t[r/x]$ bezeichnet den Term, der sich ergibt, wenn der Identifikator x in t durch r ersetzt wird
- Substitution ist induktiv über den Aufbau der Terme beschrieben
 - $x[t/x] = _$
 - $y[t/x] = _ \quad \text{Annahme: } y \text{ und } x \text{ sind verschiedene Identifikatoren}$
 - $f(t_1, \dots, t_n)[t/x] = f(t_1[t/x], \dots, t_n[t/x])$
 - wobei $f \in F$ mit $\text{fct } f = (s_1, \dots, s_n) s_{n+1}$
 - Terme t_i haben Sorten s_i
- Ein Term r heißt Instanz des Terms t , wenn r durch Substitution gewisser (freier) Identifikatoren aus t erhältlich ist

Interaktion 8: Substitution in Termen

Durch die Substitution wird ein formaler Mechanismus geschaffen, um mit den Identifikatoren zu arbeiten, insbesondere um diese mit Werten zu belegen.

Die Formulierung „induktiv über den Termaufbau“ besagt, dass eine Definition entlang der Bildungsgesetze, die für den Aufbau eines Terms bestehen, erfolgt. Die zwei Bildungsgesetze wurden in Interaktion 4 angegeben und werden bei der Definition der Substitution ausgenutzt.

Da der Termaufbau mit einfachen Identifikatoren beginnt, geht die Beschreibung der Substitution ebenfalls zunächst von Identifikatoren, hier x und y , aus (siehe Interaktion 8):

- $x[t/x]$ bezeichnet dabei den Term, der sich ergibt, wenn x im "Term" x durch den Term t ersetzt wird. Da es sich bei diesem Term um einen einzelnen Identifikator handelt, wird das x gegen t ersetzt, d.h. das Ergebnis ist t .
- Falls es sich um einen Identifikator handelt, der von x verschieden ist, dann gäbe es kein zu substituierendes x und die Substitution würde gar nichts bewirken.
- Es besteht im Zusammenhang mit der Substitution noch eine weitere spezielle Schreibweise, die zugleich die dritte Regel ist, durch die die Substitution („induktiv über den Termaufbau“) beschrieben wird: $\{t_1/x_1, \dots, t_n/x_n\}$ bezeichnet den Term, der durch die simultane Substitution der Identifikatoren x_i durch die t_i aus t entsteht. Hierbei muss gefordert werden, dass die Identifikatoren x_i paarweise disjunkt sind.

Ein Term, der durch Substitution aus einem Term hervorgeht, heißt Instanz zu diesem Term. Das Vorgehen der Substitution und der Instanz-Begriff wird im Folgenden an einem Beispiel veranschaulicht.

- Term t sei definiert als $\text{mult}(\text{add}(\text{succ}(x), y), z)$
- Der Identifikator x ist mit dem Wert 0, y mit 1 und z mit 2 zu belegen

- Gesucht ist eine Instanz zu t

Interaktion 9: Beispiel einer Substitution

Durch Interaktion 9 soll verdeutlicht werden, dass die Substitution dazu genutzt werden kann, in einem Term auftretende freie Variablen mit Werten zu belegen.

- Belegung von X in A
 - Vorgehen: jedem Identifikator x in X der Sorte s wird ein Element a der Trägermenge s^A zur Sorte s zugeordnet
 - Formal: $\beta: \{x \in X_s: s \in S\} \rightarrow \{a \in s^A: s \in S\}$
- Die Interpretation eines belegten Terms ist durch folgende zwei Gleichungen definiert
 - (1) $I_\beta^A[x] = \beta(x)$
 - (2) $I_\beta^A[f(t_1, \dots, t_n)] = f^A(I_\beta^A[t_1], \dots, I_\beta^A[t_n])$
- Punktweise Änderung von Belegungen

$$\beta[m/x](z) = \begin{cases} m & \text{falls } z = x \\ \beta(z) & \text{falls } z \neq x \end{cases}$$

Information 7: Interpretation von Termen

In Information 7 ist A eine Rechenstruktur mit $\Sigma = (S, F)$ und X eine Familie von Mengen von Identifikatoren. Der Begriff "Familie" besagt, dass zu jeder einzelnen Sorte $s \in S$ eine (Teil-)Menge von Identifikatoren X_s existiert. Diese Teilmengen X_s von Identifikatoren werden zur Festlegung einer Abbildung benötigt, die Belegung genannt wird und mit β bezeichnet ist.

Gesucht ist die Interpretation $I_\beta^A[t]$ eines Terms t mit freien Identifikatoren aus X . Falls t lediglich ein Identifikator x aus X ist, so ist die Interpretation von t die Belegung von x .

Gemäß den Gesetzen des Termaufbaus ist ein komplexerer Term aufgebaut aus einer in der Funktionenmenge F der Signatur $\Sigma = (S, F)$ der Rechenstruktur A enthaltenen Funktion f und entsprechenden Termen t_1 bis t_n . In diesem Fall sind diese Terme zu interpretieren und die Ergebnisse sind als Argumente der Funktion f^A zuzuführen.

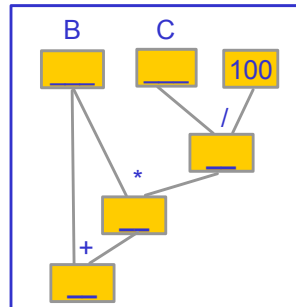
Es gibt Situationen, in denen man eine Belegung an einem Punkt abzuändern und an allen anderen Punkten zu übernehmen ist. Hierfür dient eine in Information 7 angegebene Notation, die an die Substitution angelehnt ist.



- Terme mit freien Identifikatoren können als Rechenformulare gedeutet werden, in denen noch nicht alle Werte festgelegt sind

- Beispiel: $A = B + B * (C/100)$

Was wird berechnet?



- Ein Term mit freien Identifikatoren definiert ein so genanntes Berechnungsschema
- Das Berechnungsschema hat die Form eines zyklensfreien gerichteten Graphen, falls gewisse Identifikatoren mehrfach auftreten

Interaktion 10: Terme als Formulare

In Interaktion 10 werden nochmals die Formulare aufgegriffen, da sich diese zur Behandlung von Termen mit freien Identifikatoren besonders gut eignen. Die freien Identifikatoren stellen im Formular Felder dar, deren Werte im Zusammenhang mit der Belegung festgelegt werden müssen. Anhand des Terms $A = B + B * (C / 100)$ wird der Sachverhalt verdeutlicht.

Eins solches Formular mit freien Identifikatoren wird als Berechnungsschema bezeichnet. Komplexere Berechnungsschemata findet man u.a. im Verwaltungsbereich, wie z.B. in der Lohnsteuerberechnung. Ein Berechnungsschema kann die Form eines Baumes aufweisen. Die Baumform ergibt sich genau dann, wenn Identifikatoren nur genau einmal auftreten. Tritt ein Identifikator mehrmals auf, so ergibt sich ein zyklensfreier gerichteter Graph, der auch als Hasse-Diagramm bezeichnet wird.

2 TERMERSETZUNG

Durch die Terme und die darauf aufbauenden Termersetzungssysteme wird eine im Vergleich zu den in der Kurseinheit ALGEBRAISCHE GRUNDLAGEN [C&M-AG] behandelten Textersetzungssystemen klarere und aussagekräftigere Möglichkeit zur Beschreibung von Algorithmen bereitgestellt [Br98].

- Die Beschreibung von Algorithmen als Textersetzungssysteme wurde bereits behandelt
- Die Ersetzung von Termen anstelle einzelner Zeichen führt zu einer klareren Beschreibungsmöglichkeit von Algorithmen
 - Terme lassen sich über gegebenen Signaturen nach festen Regeln aufbauen
 - Interpretationen wird über einer Rechenstruktur angegeben
 - semantische Äquivalenz auf Termen ist vorgegeben
- Aufstellen von Termersetzungsregeln, die die semantische Äquivalenz berücksichtigen
 - Ausnutzen der speziellen Struktur der Terme, die sich aus ihrem Aufbau ergibt

Information 8: TERMERSETZUNG - Motivation

Der Weg von den Textersetzungssystemen zu den Termersetzungs-systemen führt über die Festlegung, was ein Term ist, d.h. welche Syntax und Semantik ihm zugrunde gelegt werden soll.

Die Syntax von Termen ist durch den Termaufbau gegeben, der durch die Signatur und die zwei zuvor in Interaktion 4 eingeführten Regeln festgelegt ist. Die Regeln wurden bereits an verschiedenen Stellen zur Definition und zum Nachweis von Termeigenschaften benutzt. Die Semantik von Termen führt über die Interpretationsfunktion. Durch die Interpretation (über einer Rechenstruktur oder Algebra) wird eine semantische Äquivalenz auf Termen vorgegeben.

Es lassen sich Regeln zur Umformung von Termen so gestalten, dass Terme stets in semantisch äquivalente Terme überführt werden können. Das führt unmittelbar zu den Termersetzungs-systemen.

2.1 Termersetzungsregel und -algorithmus

Zunächst wird mit den Termersetzungsregeln der Kern eines Termersetzungs-systems behandelt.

- Termersetzungsregel
 - Paar von Termen (t, r)
 - Regel-Schreibweise: $t \rightarrow r$
 - Terme t und r dabei
 1. von gleicher Sorte (zu einer gegebenen Signatur)
 2. mit freien Identifikatoren aus einer gegebenen Menge X von Identifikatoren
- Forderung: Alle Identifikatoren, die im Term r auftreten, müssen auch im Term t auftreten
- Instanz der Regel $t \rightarrow r$: $t[t_1/x_1, \dots, t_n/x_n] \rightarrow r[t_1/x_1, \dots, t_n/x_n]$
 - Ersetzen gewisser Identifikatoren x_1, \dots, x_n in t und r durch Terme t_1, \dots, t_n passender Sorten

Information 9: Termersetzungsregeln

Termersetzungsregeln sind Paare von Termen (t, r) mit den in Information 9 genannten Eigenschaften. Häufig wird die Forderung gestellt, dass die Menge der in t und r auftretenden Identifikatoren übereinstimmt. Es sei angemerkt, dass es auch Termersetzungs-systeme gibt, bei denen eine Regel angewendet werden darf, auch wenn nicht sämtliche in der Regel auftretende Identifikatoren in beiden Termen vorhanden sind.

Ausgehend von einer Termersetzungsregel $t \rightarrow r$ gelangt man zu einer konkreten Ausprägung, genannt Instanz, indem festgelegt wird, welche Identifikatoren durch welche Terme in t und r zu ersetzen sind.

- Betrachtete Regel $t \rightarrow r$
 - $\text{pred}(\text{succ}(x)) \rightarrow x$
- Gesucht sind mindestens zwei Regelinstanzen $t[t_1/x] \rightarrow r[t_1/x]$

- _____
- _____
- _____
- _____

Interaktion 11: Beispiel von Instanzen zu einer Regel

Das Vorgehen wird in Interaktion 11 an einem konkreten Beispiel verdeutlicht. Anhand der in diesem Beispiel betrachteten Regel $\text{pred}(\text{succ}(x)) \rightarrow x$ kann man sich die im Vergleich zu Zeichenersetzungsregeln höhere Aussagekraft von Termersetzungsregeln klarmachen.

- $c[t/x] \rightarrow c[r/x]$ heißt Anwendung einer Regel
 - $t \rightarrow r$ eine Instanz der Regel
 - c sei ein Term, in dem der Identifikator x frei vorkommt
 - t heißt Redex
 - das Auftreten von x in c heißt die Anwendungsstelle
- Beispiel:
 - Regelinstanz $t \rightarrow r$
 - $\text{pred}(\text{succ}(\text{zero})) \rightarrow \text{zero}$
 - Anwendung der Regel auf den Term $\text{succ}(\text{succ}(\text{pred}(\text{succ}(\text{zero}))))$
 - $\text{succ}(\text{succ}(\text{pred}(\text{succ}(\text{zero})))) \rightarrow \text{succ}(\text{succ}(\text{zero}))$
- Eine Menge von Termersetzungsgesetzen bildet einen Termersetzungsalgorithmus
 - analog zu den Textersetzungssystemen

Information 10: Regelanwendung und Termersetzungsalgorithmus

Regelinstanzen kommen bei der Anwendung einer Regel auf einen Term zum Einsatz, wie Information 10 zeigt. Durch die Anwendung einer Regel wird ein so genannter Termersetzungsschritt ausgeführt. Das beschriebene Vorgehen der Anwendung einer Regel, also die Ausführung eines Termersetzungsschritts, wird an einem Beispiel verdeutlicht.

Wie bei den Textersetzungssystemen führen die Termersetzungsgesetze zum Algorithmusbegriff.

2.2 Termersetzungssystem

Bevor der Termersetzungsalgorithmus vertieft wird, muss zunächst das Termersetzungssystem formal eingeführt werden. Außerdem ist die Frage nach der Korrektheit von Termersetzungssystemen zu stellen.

- Ein Termersetzungssystem über einer Signatur Σ ist eine im Allgemeinen endliche Menge R von Termersetzungsgesetzen
- Berechnung ausgehend vom Term t_0
 - Folge von Termen t_0, t_1, \dots, t_n , für die gilt:
 - $t_i \rightarrow t_{i+1}$ ist Anwendung einer Regel aus dem Termersetzungssystem R
 - t heißt terminal, wenn es keinen Term r gibt, so dass $t \rightarrow r$ gilt
- Normalform
 - Menge der terminalen Grundterme zu einem Termersetzungssystem
 - zu einem vorgegebenen Term t wird ein terminaler Term r als Normalform zugeordnet
 - liefert ein durch das Termersetzungssystem induziertes Normalformsystem

Information 11: Termersetzungssystem

Zu den in Information 11 beschriebenen Termersetzungssystemen gelangt man über die Termersetzungsregeln. Es wird im Allgemeinen mit einer endlichen Anzahl von solchen Regeln gearbeitet.

Eine Folge von Regelanwendungen, die von einem Ausgangsterm (hier t_0) ausgeht, wird als Berechnung bezeichnet. Die Berechnung endet in natürlicher Weise, wenn ein Term erreicht wird, auf den keine der vorhandenen Regeln angewendet werden kann. Ein solcher Term (hier t_n) wird als terminal und die Berechnung als terminierend bezeichnet. Dieser Term heißt Ergebnis oder Ausgabe der Berechnung.

Auf der Basis der terminierenden Berechnung bzw. der daraus resultierenden terminalen Terme lassen sich die so genannten Normalformen definieren. Als Normalform werden alle terminalen Grundterme zu dem Termersetzungssystem R bezeichnet. Man sagt auch, ein Term ist in Normalform bezüglich R .

Die Normalformen können dazu verwendet werden, einem beliebigen Term t eine Normalform r zuzuordnen, wobei r dann das Ergebnis einer Berechnung mit Eingabe t ist. Hierdurch wird aus dem Termersetzungssystem ein so genanntes Normalformsystem induziert.

www.cm-tm.uka.de/info1
Info1-Team (Prof. Abeck)



- Ein Termersetzungssystem R mit einem Grundterm t als Eingabe definiert mittels der folgenden zwei Vorschriften einen Algorithmus:
 - (1) Enthält R eine Ersetzungsregel mit Anwendung $t \rightarrow r$, dann wird der Algorithmus mit dem Term r statt t fortgesetzt.
 - (2) Enthält R keine Ersetzungsregel mit Anwendung $t \rightarrow r$, so endet der Algorithmus mit r als Resultat.

- Beispiel: Termersetzungssystem zur Signatur der Rechenstruktur NAT
 - Sorte: nat
 - Funktionen: $\text{fct zero} = \text{nat}$ $\text{fct succ} = (\text{nat}) \text{ nat}$ $\text{fct pred} = (\text{nat}) \text{ nat}$
 - Normalformen: $\text{succ}(\dots(\text{succ}(\text{zero})\dots))$ für Terme der Sorte nat
 - Beispiel für einen Termersetzungsalgorithmus
 - Reduktionsregel für pred: $\text{pred}(\text{succ}(x)) \rightarrow x$
 - Wirkung: Elimination des Funktionssymbols pred in Grundtermen
 - Berechnung: $\text{succ}(\text{pred}(\text{pred}(\text{succ}(\text{succ}(\text{pred}(x)))))) \rightarrow$

Interaktion 12: Termersetzungsalgorithmus

Die wiederholte Anwendung von Termersetzungsregeln definiert einen Algorithmus. Man gelangt zu diesem Termersetzungsalgorithmus über die in Interaktion 12 genannten zwei einfachen Vorschriften. Die erste Vorschrift behandelt dabei den Fall, dass es eine Ersetzungsregel gibt. Andernfalls gilt die zweite Vorschrift und der Algorithmus endet (bzw. terminiert). Der Anfangsterm t heißt entsprechend Eingabe und der Endeterm r heißt Resultat oder Ausgabe.

Bei dem angegebenen Beispiel handelt sich um ein Termersetzungssystem für die Signatur der Rechenstruktur NAT mit der Sorte nat und den drei Funktionen zero, succ und pred. Die Normalformen bestehen offensichtlich aus einer beliebigen Anzahl von verschachtelten succ-Funktionen, die auf die 1-stellige Funktion zero angewendet werden.

Im beispielhaft angegebenen Termersetzungsalgorithmus wird durch die angegebene Regel $\text{pred}(\text{succ}(x)) \rightarrow x$ erreicht, dass in einem beliebigen (nicht undefinierten) Grundterm gewisse pred -Funktionen eliminiert werden.

Zu beachten ist, dass syntaktisch bildbare Terme wie $\text{pred}(\text{zero})$ mit der durch die natürlichen Zahlen gegebenen Interpretation einen undefinierten Wert (\perp) hat. Entsprechend ist der Term nicht in Normalform und es existiert auch kein semantisch äquivalenter Term in Normalform.

www.cm-tm.uka.de/info1
Info1-Team (Prof. Abeck)

- Partielle Korrektheit
 - R: Termersetzungssystem; A: Rechenstruktur der Signatur
 - Eine Termersetzungregel $t \rightarrow r$ über der Signatur heißt partiell korrekt bzgl. der Rechenstruktur A, falls für jede Belegung β in A gilt:

$$I_{\beta}^A[t] = I_{\beta}^A[r]$$
 - R heißt partiell korrekt bezüglich der Rechenstruktur A, falls jede Regel partiell korrekt bezüglich A ist
- Totale Korrektheit
 - Ein partiell korrektes Termersetzungssystem R heißt total korrekt bezüglich der Rechenstruktur A, wenn
 - (a) für Grundterme t mit $t^A \neq \perp$ keine nichtterminierenden Berechnungen existieren
 - (b) für bezüglich R terminale, verschiedene Grundterme t_1, t_2 mit $t_1^A \neq \perp$ und $t_2^A \neq \perp$ stets gilt $t_1^A \neq t_2^A$

Information 12: Korrektheit von Termersetzungssystemen

Termersetzungsgesetze entsprechen syntaktischen Ersetzungsgesetzen auf der Ebene der Terme. So sucht eine Regel wie $\text{pred}(\text{succ}(x)) \rightarrow x$ das Muster $\text{pred}(\text{succ}(\dots))$ in dem vorgegebenen Term und ersetzt dieses entsprechend der Regel.

Die Bedeutung eines Terms innerhalb der Rechenstruktur A wird bekanntlich durch die Interpretationsfunktion I_{β}^A beschrieben, wobei β die Belegung der freien Identifikatoren im Term darstellt. Eine Termersetzungsgesetz $t \rightarrow r$ ist genau dann partiell korrekt bzgl. A, wenn $t = r$ eine in A gültige Aussage ist.

Umfassender als die partielle Korrektheit ist die vollständige Korrektheit, die zu der partiellen Korrektheit zwei weitere in Information 12 genannte Forderungen stellt.

Partielle und totale Korrektheit weisen hinsichtlich eines Aspekts eine grundsätzlich verschiedene Charakteristik auf: Während die partielle Korrektheit eine Eigenschaft der einzelnen Regeln ist (und damit auch für die einzelnen Regeln unabhängig gezeigt werden kann), ist die totale Korrektheit eine Eigenschaft des gesamte Termersetzungssystems.

3 MATHEMATISCHE LOGIK

Die mathematische Logik ist in der Informatik ein wichtiges Werkzeug, um Algorithmen zu beschreiben und zu verifizieren. Der so genannten Logikprogrammierung werden Systeme logischer Formeln als Algorithmenbeschreibungen aufgefasst. In diesem Kapitel wird der Aufbau aussagen- und prädikatenlogischer Formeln sowie der Ableitungsregeln (Inferenzregeln) behandelt [Br98].

- Die mathematische Logik dient in der Informatik als ein Werkzeug zur Beschreibung von Algorithmen
- Im Folgenden werden der Aufbau logischer Formeln und der logischen Regelsysteme behandelt
- Prinzip des „Tertium non datur“
 - es werden nur elementare Aussagen betrachtet, die entweder den Wert L oder den Wert O besitzen
 - ein dritter Wert (\perp) ist nicht zulässig
 - wird durch Beschränkung auf bestimmte Terme der Sorte `bool` sichergestellt
 - so genannte starke Terme
 - werden Formeln genannt
- Grundterme, die Formeln darstellen, heißen elementare Aussagen

Information 13: MATHEMATISCHE LOGIK - Einführung

Die Vermeidung von \perp wird dadurch erreicht, dass nur solche Terme der Sorte `bool` zugelassen werden, für die aufgrund ihrer äußeren Gestalt sichergestellt ist, dass sie als Interpretation nur die Werte L und O besitzen, nicht aber den dritten Wert \perp . Das wird dadurch sichergestellt, dass in dem Term nur starke Funktionen verwendet werden. Eine Funktion heißt stark (*strong*), wenn sie für beliebige Argumente stets die Resultate L oder O liefern, also niemals \perp als Resultat hat. Entsprechend spricht man von einem starken Term bzw. einer Formel.

www.cm-tm.uka.de/info1
Info1-Team (Prof. Abeck)



- Rechenstruktur A, Sorte m; Terme t und r Grundterme der Sorte m
- Starke Gleichheit
- Schwache Gleichheit

$$(t = r)^A = \begin{cases} L & \text{falls } t^A = r^A \\ O & \text{falls } t^A \neq r^A \end{cases}$$

$$(t =? r)^A = \begin{cases} L & \text{falls } t^A \neq \perp \text{ und } r^A \neq \perp \text{ und } t^A = r^A \\ O & \text{falls } t^A \neq \perp \text{ und } r^A \neq \perp \text{ und } t^A \neq r^A \\ \perp & \text{falls } t^A = \perp \text{ oder } r^A = \perp \end{cases}$$

Ist $t_1 = t_2$ eine Formel?

Ist $t_1 =? t_2$ eine Formel?

Interaktion 13: Starke und schwache Gleichheit

Am Beispiel der starken und schwachen Gleichheit soll in Interaktion 13 die Frage verdeutlicht werden, ob ein Term der Sorte `bool` eine Formel ist oder nicht. Beide angegebenen Terme $t_1 = t_2$ und $t_1 =? t_2$ sind Grundterme der Sorte `bool`.

- Gegenstand sind Formeln der Sorte bool , die
 - als Interpretation den Wert L besitzen
 - im Sinne der mathematischen Logik wahr sind
- Mathematische Logik behandelt das Formalisieren des Schlussfolgerns
 - Formeln: (zweiwertige) Terme der Sorte bool
 - elementare Aussage: Grundterme der Sorte bool
 - Ziel: Ableitung von Formeln aus einer Menge gegebener und als wahr angenommener Formeln
- In der Aussagenlogik sind die Formeln einfache Terme der Sorte bool
- Die Prädikatenlogik erweitert die Aussagenlogik, so dass auch auf andere Sorten bzw. Trägermengen Bezug genommen werden kann

Information 14: Inhalt und Ziel der mathematischen Logik

In der mathematischen Logik stehen solche Terme im Mittelpunkt der Untersuchung, deren Interpretation den Wert L ergibt. Diese Terme werden als wahr bezeichnet.

Häufig werden gewisse Formeln von vornherein als wahr vorausgesetzt. Diese Formeln werden auch als Hypothesen oder Axiome bezeichnet. Ausgehend von einer Hypothesen- oder Axiomenmenge möchte man in der mathematischen Logik nun wissen, welche weiteren Formeln hieraus ableitbar sind und entsprechend als wahr beweisen kann. Die abgeleiteten Formeln sind dann nicht Hypothesen oder Axiome, sondern so genannte Theoreme.

Das Mittel, auf dessen Basis die Ableitung („Formel-Beweis“) vorgenommen wird, ist ein Regelsystem, bestehend aus so genannten Ableitungs- oder Inferenzregeln. Die Regeln erzeugen also aus einer Menge von Axiomen ein Theorem, was als Beweis dieser abgeleiteten Formel anzusehen ist.

Das Vorgehen der Einführung eines Regelsystems liegt sowohl der Aussagenlogik als auch der Prädikatenlogik zugrunde. Die Prädikatenlogik kann dabei als Erweiterung der Aussagenlogik verstanden werden, da sie aufbauend auf den Termen der Sorte bool (Aussagenlogik) auf andere Sorten bzw. Trägermengen Bezug nimmt.

Die Logik ist ein Teilgebiet der Mathematik, das aufgrund der darin eingeführten Regelsysteme für die Informatik von hoher Relevanz ist.

3.1 Aussagenlogik

Zunächst werden die Aussagenlogik und der darin auftretende zentrale Ableitungsbegriff betrachtet.



- $H \vdash t$
 - heißt: Die Formel t ist aus der Formelmenge H ableitbar
- Ableitungsregeln der Aussagenlogik

(1) $\vdash t \vee \neg t$ Tertium non datur

(2) $\{t_1, \neg t_1 \vee t_2\} \vdash t_2$ Modus Ponens

Darstellung mit Implikationspfeil

(3) Ist $t_1 = t_2$ Anwendung einer Regel der semantischen Äquivalenzen, der Booleschen Algebra oder der Booleschen Terme, so gilt auch $\{t_1\} \vdash t_2$ Anwendung Gleichheitsgesetze

Interaktion 14: Aussagenlogik

Es wird ein spezielles Ableitungssymbol \vdash eingeführt. Wie in Information 14 ausgeführt wurde, besteht das Ziel darin, aus einer Menge wahrer Formeln H gemäß gewisser Ableitungsregeln eine Formel t abzuleiten und damit als wahr zu beweisen. Die Axiomenmenge H kann dabei auch leer sein, was durch $\vdash t$ beschrieben ist. In der Aussagenlogik bestehen die in Interaktion 14 angegebenen drei Ableitungsregeln.

- Lemma
Gilt $H \vdash t_a \Rightarrow t_b$, dann gilt auch $H \cup \{t_a\} \vdash t_b$

Beweis

$H \vdash t_a \Rightarrow t_b$	Voraussetzung
$H \vdash \neg t_a \vee t_b$	Definition der Implikation
$\{t_a, \neg t_a \vee t_b\} \vdash t_b$	Modus Ponens
$H \cup \{t_a\} \vdash t_b$	w.z.b.w.

- Begriff der Ableitung formalisiert das Konzept des mathematischen Beweisens
 - Formales System oder Theorie
= Axiomenmenge + Menge der Ableitungsregeln
 - Theoreme = Menge der ableitbaren Formeln

Information 15: Zusammenhang zwischen Implikation und Ableitbarkeit

Am Beispiel des in Information 15 angegebenen Lemmas, welches besagt, dass aus einer Implikation von t_b aus einem Term t_a (d.h. t_a impliziert t_b) auch dessen Ableitbarkeit folgt (also $H \cup \{t_a\} \vdash t_b$), soll das Vorgehen des Beweisens in der Aussagenlogik verdeutlicht werden.

- Hierbei wird im ersten Schritt von der Voraussetzung ausgegangen.
- Der zweite Beweisschritt nutzt die Definition der Implikation aus, die gleichwertig zum aussagenlogischen Term $\neg t_a \vee t_b$ ist. Unter der Annahme der Voraussetzung wurde

hierdurch die Menge der aus der Hypothesenmenge H ableitbaren Formeln um den (starken) Term $t_2 = \neg t_a \vee t_b$ ergänzt.

- Die letzten zwei Schritte stellen den eigentlichen Beweis des Lemmas dar, indem durch die Formulierung des zum Regelsystem gehörenden Modus Ponens die Herleitung der zu beweisenden Aussage möglich wird.

Der exemplarische Beweis verdeutlicht das grundsätzliche Vorgehen, dass die in einem früheren Schritt des Beweises abgeleiteten Aussagen zur Ableitung weiterer Aussagen verwendet werden. Das galt im obigen Fall für die Aussage $\neg t_a \vee t_b$, die in Schritt 2 erzeugt und in Schritt 4 dann verwendet wurde.

Der Begriff der Ableitung formalisiert das Konzept des mathematischen Beweises. Den Ausgangspunkt liefern die Axiome und Schlussregeln (heißen formales System oder Theorie). Durch Anwendung der Schlussregeln lassen sich neue Formeln, die Theoreme, ableiten.

www.cm-tm.uka.de/info1
Info1-Team (Prof. Abeck)



- Eine Theorie heißt inkonsistent, falls jede Formel ableitbar ist
- Theorem: Ist in einer Theorie der Aussagenlogik $false$ ableitbar, so ist jede Aussage t ableitbar, da gilt: $\{false\} \vdash t$
 - Beweis

$\vdash t \vee \neg t$	Tertium non datur
$\vdash (t \vee t) \vee \neg t$	Idempotenz Disjunktion
$\vdash t \vee (t \vee \neg t)$	Assoziativität Disjunktion
$\vdash (t \vee \neg t) \vee t$	Kommutativität Disjunktion
$\vdash \neg \neg(t \vee \neg t) \vee t$	Involutionsgesetz
$\vdash \neg \neg true \vee t$	Gesetz für Boolesche Terme
$\vdash \neg false \vee t$	Gesetz für Boolesche Terme
$\vdash false \Rightarrow t$	Gesetz für Boolesche Terme
$\{false\} \vdash t$	Zuvor bewiesenes Lemma mit

Interaktion 15: Inkonsistenz einer Theorie

In Interaktion 15 wird die wichtige Eigenschaft der Konsistenz bzw. Inkonsistenz einer Theorie (also eines Axiomensystems sowie der dazugehörigen Schlussregeln) behandelt.

Falls in einer Theorie jede Formel ableitbar ist, so lässt sich zeigen, dass eine Theorie nicht konsistent sein kann, also inkonsistent ist. Zu "jeder Formel" gehören auch Formeln wie $\neg a \wedge a$ oder auch gleichbedeutend damit $false$. Falls jede Formel ableitbar ist, sind somit insbesondere auch aussagenlogische Widersprüche erzeugbar.

Nun gilt nicht nur, dass aus einer inkonsistenten Theorie $false$ abgeleitet werden kann. Auch umgekehrt gilt: Für jede Theorie, aus der $false$ abgeleitet werden kann (und somit $false$ in der Axiomenmenge enthalten ist), kann bewiesen werden, dass diese Theorie inkonsistent ist.

Der Beweis hierzu ist in Interaktion 15 angegeben. Für den letzten Schritt des Beweises wird das zuvor bewiesene Lemma ausgenutzt, was im Rahmen der Interaktion zu zeigen ist.

Eine Theorie heißt korrekt, wenn alle ableitbaren Formeln für beliebige Belegungen der darin auftretenden freien Identifikatoren wahr sind. Eine Formel t ist dann wahr, wenn für alle Belegungen β gilt, dass die Interpretationsfunktion $I_{\beta}[t] = L$ ist.

www.cm-tm.uka.de/info1
Info1-Team (Prof. Abeck)

- Theorem
Jeder aus der leeren Menge von Axiomen ableitbaren Aussage t mit freien Identifikatoren x_1, \dots, x_n ($n \in \mathbb{N}$) der Sorte `bool` wird für beliebige Belegungen von x_1, \dots, x_n durch L oder O der Wert L zugeordnet

- Beweis
 1. Aus der leeren Axiomenmenge ist unmittelbar nur $(t \vee \neg t)$ ableitbar
 - erfüllt die geforderte Eigenschaft
 2. Modus Ponens erlaubt aus wahren Aussagen nur die Ableitung wahrer Aussagen

- Ein formales System heißt vollständig, wenn für jede elementare Aussage t (jede Formel ohne freie Identifikatoren) die Formel t oder $(\neg t)$ ableitbar ist

Information 16: Korrektheit und Vollständigkeit

In Information 16 wird gezeigt, dass das formale System (die Theorie) mit der leeren Axiomenmenge und den drei eingeführten Ableitungsregeln der Aussagenlogik korrekt ist.

Eine noch umfassendere Anforderung als die Korrektheit wird an eine Theorie durch die Eigenschaft der Vollständigkeit gestellt. Diese Eigenschaft bedeutet, dass in der Theorie alle korrekten Formeln abgeleitet werden können.

www.cm-tm.uka.de/info1
Info1-Team (Prof. Abeck)

- Theorem
 - Wird eine aussagenlogische Formel t , die keine elementaren atomaren Aussagen enthält, mit freien Identifikatoren x_1, \dots, x_n der Sorte `bool` für jede Belegung von x_1, \dots, x_n mit Wahrheitswerten der Wert L zugeordnet, so ist t ableitbar

- Beweisidee
 - Es ist zu zeigen, dass einer Formel t genau dann der Wert L für alle Belegungen β zugeordnet wird, wenn t auf `true` reduzierbar ist

Information 17: Vollständigkeit der Aussagenlogik

Ein Term t der Sorte `bool` heißt aussagenlogische Formel, wenn t nur aus

- den Termen `true`, `false`
- atomaren Aussagen (eine nicht weiter zerlegbare Aussage t_i , die wahr oder falsch sein kann)
- den logischen Operatoren
- Identifikatoren x_1, \dots, x_n der Sorte `bool` aufgebaut ist.

Für jede Belegung von x_1, \dots, x_n mit Wahrheitswerten kann einer aussagenlogischen Formel, die keine elementaren atomaren Aussagen enthält, mittels der Interpretationsfunktion ein Wahrheitswert L oder O zugeordnet werden.

Die Aussagenlogik stellt u.a. eine wichtige Grundlage für Programmiersprachen dar, da in praktisch jeder Programmiersprache die Wahrheitswerte und die behandelten Abbildungen auftreten.

3.2 Prädikatenlogik

Der Aussagenlogik liegt eine Signatur zugrunde, in der nur Terme der Sorte bool auftreten. Die Prädikatenlogik stellt eine Erweiterung der Aussagenlogik dar.

www.cm-tm.uka.de/info1
Info1-Team (Prof. Abeck)



- n-stelliges Prädikat, n-stellige boolesche Funktion
 - eine Abbildung $p: M_1 \times \dots \times M_n \rightarrow \mathcal{B}$ von Tupeln über gegebenen Mengen in die Menge der Wahrheitswerte
 - jede Anwendung des Prädikats p bezogen auf Elemente $a_i \in M_i$ liefert einen Wahrheitswert $p(a_1, \dots, a_n)$

- Beispiele Funktionschreibweise
 - einstellig: x studiert Informatik _____
 - zweistellig: $\leq, \geq, =, \neq$ für Zahlen _____
 - \in für Mengen _____

- Nahe liegende Möglichkeiten, ein Prädikat p in eine elementare Aussage zu verwandeln:
 - (1) für alle $x \in M$ gilt $p(x)$
 - (2) für (mindestens) ein $x \in M$ gilt $p(x)$

Interaktion 16: Prädikatenlogik

Die Form von Abbildungen, die in der Prädikatenlogik betrachtet werden, bezeichnet man als Prädikate. Wie in Interaktion 16 dargestellt ist, kommen mit den Prädikaten neue nicht-boolesche Sorten bzw. Mengen zur Rechenstruktur hinzu. Durch das Prädikat werden diese auf einen booleschen Wahrheitswert abgebildet.

Je nach Anzahl n der als Argumente in das Prädikat eingehenden Argumente spricht man von einem n -stelligen Prädikat. Ein Beispiel eines Prädikats mit einem Eingangsargument, also ein einstelliges Prädikat, ist x studiert Informatik. Entsprechend sind die verschiedenen Gleichheitsoperatoren (z.B. ungleich) oder der Enthaltenseins-Operator Beispiele für zweistellige Prädikate.

Das Ergebnis der so konstruierten Prädikate ist selbstverständlich von den jeweiligen Werten abhängig, die die Eingangsargumente gerade besitzen. Offensichtlich lassen sich durch die zwei Konstrukte für alle und es gibt Aussagen treffen, die den gesamten Wertebereich der Eingangsargumente einbeziehen.

Nachfolgend werden Aussagen der obigen Form formal als Terme der Prädikatenlogik eingeführt.

- $W_{\Sigma}^{\text{bool}}(X)$ mit $\Sigma = (S, F)$ beschreibt die Menge der booleschen Terme
 - Σ ist eine Signatur, die u.a. die Sorte `bool` und die booleschen Operatoren $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ enthält
 - Operationssymbole aus F
 - freie Identifikatoren aus der Familie von Mengen von Identifikatoren X

- Prädikatenlogische Formeln
 - $\forall m x: t$ (Allquantor)
 - für alle Elemente $x (\neq \perp)$ der Sorte m gilt die Aussage t
 - $\exists m x: t$ (Existenzquantor)
 - es existiert ein Element $x (\neq \perp)$ der Sorte m , für das die Aussage t gilt

- Bei Formeln der obigen Form spricht man von Quantifizierung

Information 18: Bildung prädikatenlogischer Formeln

Die Informatik ist insbesondere auch an der (Term-) Algebra interessiert, die sich aus den über einer Algebra gebildeten Termen ergibt (siehe auch die Ausführungen in der Kurseinheit ALGEBRAISCHEN GRUNDLAGEN [C&M-AG]). $W_{\Sigma}^{\text{bool}}(X)$ bezeichnet die Menge der booleschen Terme, auf deren Grundlage die Termalgebra der Prädikatenlogik gebildet wird. Die Termalgebra beinhaltet die korrekt gebildeten Formeln (*well formed formula* wff).

Aufbauend auf einer Formel t , die Identifikatoren der Sorte m der Trägermenge M enthält, gelangt man durch die Einführung von zwei neuen Symbolen zu den prädikatenlogischen Formeln. Anstelle der zuvor angegebenen Begriffe für alle bzw. es gibt sind die Schreibweisen $\forall x \in M: t$ bzw. $\exists x \in M: t$ üblich. Die Symbole heißen Quantoren, weshalb im Zusammenhang mit der Nutzung dieser Symbole auch von einer Quantifizierung gesprochen wird.



- In Quantifizierungen treten Identifikatoren nicht frei, sondern durch den All- bzw. Existenzquantor gebunden auf
 - in einer Substitution eines gebundenen Identifikators durch einen Term wird dieser nicht ersetzt
 - ausgedrückt durch folgende Regeln

$$(\forall m x: t) [t'/x] = \forall m x: t$$

$$(\exists m x: t) [t'/x] = \exists m x: t$$

- Beispiele prädikatenlogischer Ausdrücke mit Quantoren

- | | wahr | falsch |
|-------------------------------------------------------|--------------------------|--------------------------|
| $\forall \text{bool } x: x \vee \neg x$ | <input type="checkbox"/> | <input type="checkbox"/> |
| $\exists \text{nat } m: \forall \text{nat } n: n < m$ | <input type="checkbox"/> | <input type="checkbox"/> |
| $\forall \text{nat } n: \exists \text{nat } m: n < m$ | <input type="checkbox"/> | <input type="checkbox"/> |

- Gesetze zu den Quantoren

- $(\forall m x: t) = (\neg \exists m x: \neg t)$

- $(\exists m x: t) = (\neg \forall m x: \neg t)$



Interaktion 17: Eigenschaften von Quantifizierungen

Die Semantik des in Interaktion 17 eingeführten Begriffs "gebunden" wird im Zusammenhang mit der Substitution deutlich.

An den Beispielen wird deutlich, dass man wahre und falsche Aussagen mittels Quantoren bilden kann. Am zweiten und dritten Beispiel lässt sich leicht klarmachen, dass sich durch Änderung der Reihenfolge der Quantoren der Wahrheitswert der Aussage ändern kann.

Durch die zwei in Interaktion 17 angegebenen Quantoren-Gesetze wird deutlich, dass sich der Allquantor in Form einer Negation durch den Existenzquantor ausdrücken lässt und umgekehrt. Die Gesetze sind von ihrem Sinn her leicht nachvollziehbar. So besagt Gesetz (1), dass man kein x finden kann, für das $\neg t$ gilt, falls für alle x der Term t gilt. Das lässt sich graphisch so veranschaulichen, dass die Menge der Identifikatoren, für die $\neg t$ gilt, leer ist.

Sei M die Trägermenge zur Sorte m und $M \neq \emptyset$

- Gilt $(\Sigma, H) \vdash t$ und ist x nicht frei in den Formeln in H und nicht Funktionssymbol in Σ , so gilt:
 $(\Sigma, H) \vdash \forall m x: t$
- Sei t_1 ein Term der Sorte m , wobei $t_1 \neq \perp$; gilt $(\Sigma, H) \vdash \forall m x: t$, so gilt:
 $(\Sigma, H) \vdash t[t_1/x]$
- Gilt $(\Sigma, H) \vdash t[t_1/x]$ für einen Term t_1 der Sorte m , wobei $t_1 \neq \perp$, so gilt:
 $(\Sigma, H) \vdash \exists m x: t$
- Gegeben die Signatur $\Sigma = (S, F)$ mit $\text{fct } x = m$ und $\Sigma' = (S, F \setminus \{x\})$; gilt $(\Sigma', H) \vdash \exists m x: t$, dann gilt, falls x nicht frei in H :
 $(\Sigma, H) \vdash t$

Information 19: Ableitungsregeln der Prädikatenlogik

Ableitungsregeln wurden bereits im Zusammenhang mit der Aussagenlogik eingeführt. Wie Information 19 zeigt, bestehen insgesamt vier Regeln zur Ableitung von prädikatenlogischen Formeln über der Signatur $\Sigma = (S, F)$.

Auf die in prädikatenlogischen Termen möglichen Umformungen der Umbenennung und der Substitution wird in Information 20 eingegangen.

www.cm-tm.uka.de/info1
Info1-Team (Prof. Abeck)

- Umbenennung von gebundenen Identifikatoren
 - $(\forall m x: t) = \forall m y: (t[y/x])$ falls y nicht frei in t vorkommt
 - $(\exists m x: t) = \exists m y: (t[y/x])$ falls y nicht frei in t vorkommt
- Substitutionsregel für quantifizierte prädikatenlogische Terme
 - (1) $(\forall m x: t)[t'/x] = \forall m x: t$
 - (2) $(\forall m x: t)[t'/y] = \forall m x: (t[t'/y])$
 - falls x und y verschiedene Identifikatoren sind und x nicht frei in t' vorkommt
- Wertezuordnung zu einer prädikatenlogischen Formel (Semantik)

$$I_{\mathfrak{g}}[\forall m x: t] = \begin{cases} L & \text{falls für alle } a \in M \text{ (mit } a \neq \perp\text{): } I_{\mathfrak{g}[a/x]}[m x: t] = L \\ 0 & \text{sonst} \end{cases}$$

Information 20: Umbenennung und Substitution

Eine Umbenennung ist möglich, wenn die Forderung der Gebundenheit des umzubennenden Identifikators erfüllt ist. Wie in Information 20 ausgeführt ist, gilt das für den Allquantor und den Existenzquantor in gleicher Weise.

Bei der Substitution eines Identifikators ist zu unterscheiden, ob dieser Identifikator im prädikatenlogischen Term gebunden oder frei ist. Die Beschreibung der Regel erfolgt in Form der zwei in Information 20 angegebenen Gleichungen.

Die Interpretationsfunktion wird dazu genutzt, um die Semantik der Quantoren festzulegen, wie am Beispiel des Allquantors in Information 20 gezeigt wird. Die Interpretation einer prädikatenlogischen Formel mit dem Existenzquantor, also $\exists m x: t$, erfolgt analog, wobei für alle gegen für ein zu ersetzen ist.

Das in prädikatenlogischen Termen verwendete Konzept der Bindung von Identifikatoren und die auftretenden Umbenennungsregeln finden sich in analoger Form auch in Programmiersprachen wieder.



- Eine Struktur A ist ein Paar (U_A, I_A) mit einer nichtleeren Menge U_A und einer Abbildung I_A , die
 - jedem k -stelligem Prädikatsymbol P ein k -stelliges Prädikat über U_A zuordnet
 - jedem k -stelligen Funktionssymbol f eine k -stellige Funktion auf U_A zuordnet
 - jeder Variablen x ein Element aus U_A zuordnet
- Eine Struktur A ist ein Modell für eine Formel F , falls F in der Struktur A wahr ist
- Eine Formel F ist erfüllbar, wenn sie mindestens ein Modell besitzt
- Beispiel: $U_A = \mathbb{N}$ $I_A(P) = \{(m, n) \mid m, n \in \mathbb{N}, m < n\}$
 - Ist die Formel $F = \exists x \exists y \exists z (P(x, y) \wedge P(z, y) \wedge P(x, z) \wedge \neg P(z, x))$ ein Modell?

Interaktion 18: Struktur und Modell

Abschließend soll in Interaktion 18 die Belegung, die der Interpretationsvorschrift I zugrunde liegt, näher beleuchtet werden, was zu dem Begriff der Struktur führt. Eine Struktur besteht aus zwei Elementen, einer Interpretation I sowie einer Menge U . Um zu verdeutlichen, dass I und U zu der Struktur A gehören, erscheint das A als Index (also I_A und U_A). U als Bezeichnung der Menge steht dabei für den Begriff des Universums.

Wie die Definition der Struktur verdeutlicht, ordnet die Interpretation Prädikatsymbole, Funktionssymbole und Variablenbezeichner bestimmten Prädikaten, Funktionen und Variablen des Universum zu. Die Menge U_A muss selbstverständlich die geforderten Prädikate und Funktionen mit der geforderten Stelligkeit enthalten.

Die Prädikate, Funktionen und Variablen treten sich in entsprechenden prädikatenlogischen Formeln auf. Auf der Basis einer Struktur-Definition, wie sie oben gegeben ist, lassen sich mittels der Interpretationsvorschrift die in einer bestimmten Formel F enthaltenen Prädikate, Funktionen und Variablen des Universums einsetzen und der Wert der Formel bestimmen.

Wenn es möglich ist, eine Struktur anzugeben, die ein Modell für eine Formel F ist, wird F erfüllbar genannt. Entsprechend ist F nicht erfüllbar, falls gezeigt werden kann, dass es kein Modell geben kann. Die Begriffe "Modell" und "Struktur" werden in Interaktion 18 anhand eines einfachen Beispiels verdeutlicht.

Die in der vorliegenden Kurseinheit behandelten Rechenstrukturen bilden die formale Grundlage für abstrakten Datentypen, die zentraler Bestandteil der Programmierung sind und in der Kurseinheit OBJEKTORIENTIERTE PROGRAMMIERUNG [C&M-OP] im Detail behandelt werden.

VERZEICHNISSE

Abkürzungen und Glossar

Abkürzung oder Begriff	Langbezeichnung und/oder Begriffserklärung
\perp	<i>Bottom-Element</i> Symbolisiert im Zusammenhang mit Zuständen den gedachten „Endzustand“ nichtterminierender Programme. Synonymer Begriff: undefiniert-Element
Grundterm	Ein Term, der ausgehend von den 0-stelligen Funktionen einer Signatur induktiv über den Termaufbau gebildet werden kann.
Modus Ponens	Bezeichnung einer Ableitungsregel der Aussagenlogik, die besagt, dass bei Kenntnis der Prämisse eines zulässigen Schlusses auch das Resultat dieses Schlusses abgeleitet werden kann.
Rechenstruktur	Zusammenfassung von Trägermengen und Operationen. Äquivalenter mathematischer Begriff: Algebra
Signatur	Legt fest, in welcher Weise die Funktionssymbole einer Rechenstruktur mit den Sorten sinnvoll verknüpft werden können.
Signaturdiagramm	Graphische Darstellung der Signatur einer Rechenvorschrift.
stark	Eigenschaft einer Funktion, die für beliebige Argumente stets die Resultate L oder O liefert, also niemals \perp als Resultat hat. Englischer Begriff: <i>strong</i>
strikt	Eigenschaft einer Rechenstruktur, die sicherstellt, dass aus der Undefiniertheit von nur einem Argument die Undefiniertheit des Ergebnisses der Rechenstruktur folgt.
Substitution	Formaler Mechanismus zum Arbeiten mit den Identifikatoren (insbesondere zur Wertebelegung).
Term	Ein korrekt gemäß einer vorgegebenen Signatur gebildeter Ausdruck.
Termersetzungssystem	Ein über einer Signatur Σ im Allgemeinen endliche Menge R von Termersetzungsregeln.
Tertium non datur	Bezeichnung einer Ableitungsregel der Aussagenlogik, die besagt, dass es einen dritten Wert (neben O und L) nicht gibt.
wff	well formed formula Formel, die vorgegebenen Termaufbaugesetzen genügt.

Index

\perp	3	stark	19
Grundterme	7	strikt	3
Modus Ponens	22	Substitution	11
Rechenstruktur	2, 3	Terme	7
Signatur	5	Termersetzungssystem	16
Signaturdiagramm	6	well formed formula	25

Informationen und Interaktionen

Information 1: RECHENSTRUKTUREN	2
Information 2: AUFBAU UND BEISPIELE VON RECHENSTRUKTUREN – Definition einer Rechenstruktur	2
Information 3: Strikte Funktion.....	3
Information 4: Rechenstruktur NAT der natürlichen Zahlen.....	4
Information 5: Rechenstruktur der ganzen Zahlen INT.....	6
Information 6: Terme mit (freien) Identifikatoren	10
Information 7: Interpretation von Termen	12
Information 8: TERMERSETZUNG - Motivation	14
Information 9: Termersetzungsregeln	15
Information 10: Regelanwendung und Termersetzungsalgorithmus	16
Information 11: Termersetzungssystem	16
Information 12: Korrektheit von Termersetzungssystemen	18
Information 13: MATHEMATISCHE LOGIK - Einführung.....	19
Information 14: Inhalt und Ziel der mathematischen Logik	20
Information 15: Zusammenhang zwischen Implikation und Ableitbarkeit.....	21
Information 16: Korrektheit und Vollständigkeit.....	23
Information 17: Vollständigkeit der Aussagenlogik	23
Information 18: Bildung prädikatenlogischer Formeln.....	25
Information 19: Ableitungsregeln der Prädikatenlogik.....	26
Information 20: Umbenennung und Substitution.....	27
Interaktion 1: Rechenstruktur BOOL der booleschen Werte.....	4
Interaktion 2: Signatur einer Rechenvorschrift	5
Interaktion 3: Signatur und Signaturdiagramm zur Rechenstruktur NAT	6
Interaktion 4: Grundterme.....	7
Interaktion 5: Interpretation von Grundtermen	8
Interaktion 6: Termerzeugte Rechenstruktur.....	8
Interaktion 7: Formulare.....	9
Interaktion 8: Substitution in Termen	10
Interaktion 9: Beispiel einer Substitution.....	11
Interaktion 10: Terme als Formulare.....	13
Interaktion 11: Beispiel von Instanzen zu einer Regel.....	15
Interaktion 12: Termersetzungsalgorithmus.....	17
Interaktion 13: Starke und schwache Gleichheit.....	19
Interaktion 14: Aussagenlogik	21
Interaktion 15: Inkonsistenz einer Theorie.....	22
Interaktion 16: Prädikatenlogik.....	24
Interaktion 17: Eigenschaften von Quantifizierungen.....	26
Interaktion 18: Struktur und Modell.....	28

Literatur

- [Br98] Manfred Broy: Informatik – Eine grundlegende Einführung, Band 1: Programmierung und Rechstrukturen, Springer Verlag 1998.
- [C&M-AG] Cooperation&Management: ALGEBRAISCHE GRUNDLAGEN, Kursdokument zur Vorlesung "INFORMATIK I", <http://www.cm-tm.uka.de/info1>, Universität Karlsruhe (TH), C&M (Prof. Abeck).
- [C&M-IP] Cooperation&Management: IMPERATIVE PROGRAMMIERUNG, Kursdokument zur Vorlesung "INFORMATIK I", <http://www.cm-tm.uka.de/info1>, Universität Karlsruhe (TH), C&M (Prof. Abeck).
- [C&M-OP] Cooperation&Management: OBJEKTORIENTIERTE PROGRAMMIERUNG, Kursdokument zur Vorlesung "INFORMATIK I", <http://www.cm-tm.uka.de/info1>, Universität Karlsruhe (TH), C&M (Prof. Abeck).
- [C&M-PG] Cooperation&Management: IMPERATIVE PROGRAMMIERUNG, Kursdokument zur Vorlesung "INFORMATIK I", <http://www.cm-tm.uka.de/info1>, Universität Karlsruhe (TH), C&M (Prof. Abeck).
- [Go95] Gerhard Goos: Vorlesungen über Informatik, Band 1: Grundlagen und funktionales Programmieren, Springer Verlag 1995.