Grundlagen der Programmierung

Studiengang „Informatik“, „Angewandte Informatik“ und „WIWI-Inf.“
SS'02

Prof. Dr. Madlener
Universität Kaiserslautern

Vorlesung:
Di 08.15-09.45 42/115 Fr 08.15-09.45 42/115

- Informationen
  www-madlener.informatik.uni-kl.de/ag-madlener/teaching/ss2002/gdp/gdp.html.
- Grundlage der Vorlesung: Buch
  Sperschneider/Hammer: Theoretische Informatik. Eine problemorientierte Einführung (ZBIB:LINF31), Springer Verlag.
- Bewertungsverfahren:
  Übungen: max. 10 Bonuspunkte (5 bis Aufsichtsarbeit 5 danach)
  Aufsichtsarbeit: max. 30 Punkte,
  Abschlussklausur: max. 70 Punkte.
- Aufsichtsarbeit: Sa 08.06 (12 - 15) Ort: Mensa
- Erste Abschlussklausur: 22.07 (Beachte Anmeldeschluss 16.06)
- Übungen (Termine und Anmeldung): Siehe WWW-Seite GDP

Inhaltsverzeichnis

1 Einleitung

Methoden zur Lösung von Problemen mit Hilfe von Rechnern

Probleme (technischen, mathematischen, formalen Sinne)

Formalisierung ($\equiv$ Festlegung $\equiv$ "Spezifikation")

Beispiele
Optimierungsprobleme

1. Rundreise minimaler Länge (Kosten minimal).
   Formalisierung z. B. Karte, Städte, Verbindungsabstand.
   Graph: Knoten $\longleftrightarrow$ Städte,
   Kanten (Gewicht) $\longleftrightarrow$ Verbindungen
   Eingabe: Menge von Knoten: V, Gewichtskanten E.
   Ausgabe: Rundreise mit minimalen Kosten, die alle Knoten aus V enthält.
   $\rightsquigarrow$ Rundreise Problem (Travelling Salesman) **TSP**
2. Profit-Maximierung bei Warenauswahl.
   Eingabe: Objekte mit Gewicht, Profit, Maximalgewicht W.
   Ausgabe: (Anteile) Objekte, die Maximalgewicht nicht überschreiten und Profit maximieren. D. h.
   $\sum x_i w_i \leq W$, $\sum x_i p_i$ max.
   $\rightsquigarrow$ Rucksackproblem (Varianten $x_i \in 0/1$ RP, x_i rational RP).
   Knapsack Problem. 0/1-KP, rat-KP.

3. **Weitere Optimierungsprobleme**
   - Optimale Bandspeicherung (Zugriffszeiten minimal)
   - Verschnittminimierung
   - Bin Packing
4. **Probleme aus der Zahlentheorie oder allgemeinem Theorembeweisen**

   Kryptologie: Große Primzahlen.

   Ist $1\,2\,3\,4\,5\,6\,7\,8\,9\,1\,2\,3\,4\,5\,6\,7\,8\,1\,2\,3\,4\,5\,6\,7$ Primzahl?

   Probleme aus der Zahlentheorie:

   z. B. „Jede natürliche Zahl lässt sich als Summe von vier Quadratzahlen schreiben“.

   Formalisiert: Sprache der Prädikatenlogik (**PL1**):

   $\forall X \exists A \exists B \exists C \exists D \qquad X = A^2 + B^2 + C^2 + D^2$

   „Interpretation“: $\mathbb{N}$ natürliche Zahlen, $+, ^2$ wie üblich interpretiert.

   oder **Fermat´s Problem**

   $\exists N \exists X \exists Y \exists Z \qquad (N > 2 \wedge X^N + Y^N = Z^N).$

   Allgemeiner aus **Hilbert´s Problemliste**:

   Gesucht “algorithmische Lösung” von

   Eingabe: Satz φ aus PL1 über $\mathbb{N}$.

   Ausgabe: Ja, falls φ Theorem.
   Nein, falls φ kein Theorem.

   Angenommen es gibt ein Programm $Prog_{\mathbb{N}}$ dafür.

   Frage: Hält $Prog_{\mathbb{N}}$ für jede Eingabe φ?

   $\rightsquigarrow$ Entscheidungsprobleme, Aufzählungsprobleme.

Entscheidungsprobleme

5. Sei P Menge von Programmen in PS (z.B. JAVA).
   Problem: Für $x \in A$ Eingabe für $p \in P$.
   Frage: Hält p bei Eingabe x?
   $\rightsquigarrow$ Halteproblem für p.
   Varianten:
   Problem: Für $p \in P$.
   Frage: Hält p für alle erlaubten Eingaben $x \in A$?
   $\rightsquigarrow$ Uniformes Halteproblem für P.("Ist p total definiert").

6. Wortpuzzle: (Gödel, Post, ..., Hofstädter)
   Alphabet Σ, $\Sigma^* = \{$endliche Folgen von Buchstaben aus $\Sigma\}$.
   $w = a_1 a_2 \dots a_n \in \Sigma^*$, $|w| = n$, $n = 0$ erlaubt $\varepsilon \equiv$ leeres Wort.
   Eingabe: Endlich viele Wortpaare
   $(u_1, w_1), (u_2, w_2) \dots (u_n, w_n)$
   $u_i, w_i \in \Sigma^+ = \Sigma^* - \{\varepsilon\}$.
   Frage: $\exists k > 0\ \exists n_1, \dots, n_k\ n_i \in \{1, 2, \dots, n\}$:
   $u_{n_1} u_{n_2} \dots u_{u_k} = w_{n_1} w_{n_2} \dots w_{n_k}$
   $\rightsquigarrow$ Post'sches Korrespondenzproblem PCP (Emil Post).

Wichtige Fragestellungen

- Was heißt es diese Probleme sind algorithmisch lösbar?
- Was nutzt es solche Probleme algorithmisch zu lösen?
- Was nutzt es „gute“ Lösungen für TSP zu haben?
- Wie schwierig sind diese Probleme?
- Wie hängen sie zusammen?
- Wie erkennt und beweist man, dass es keine („gute“) algorithmische Lösungen geben kann!

Nötig: Konzepte, Fakten, Methoden, Theoreme, Theorien

$\rightsquigarrow$ **Berechnungsmodelle/Programmiersprachen**.

Algorithmische Unlösbarkeit?

Problem Spezifikation	prinzipielle Lösbarkeit $\downarrow$ effiziente Lösbarkeit $\downarrow$ algorithmischer Entwurf $\downarrow$ P: Programm einer HPS

Problem: Syntaktische und semantische Verifikation von P.

Schwerpunkte

- **Syntaxanalyse**
  Formale Sprachen: Chomsky-Hierarchie
  Kontextfreie Sprachen
  Grammatiken/Erzeugungsprozess
  Automaten (endliche, Keller-)/Erkennungsprozess
- **Berechenbarkeitsmodelle**
  Semantik von Programmiersprachen: WHILE-Programme
  Maschinenmodelle: Turing- und Registermaschinen
  Rekursive und Partiel-Rekursive Funktionen
- **Programmverifikation**
  Tut P auch was erwartet wird.
  Beweisverpflichtungen: Zusicherungen, Invarianten, Terminierungsbedingungen
  Beweise: Siehe Logik Vorlesung

2 Mengen, Relationen, Funktionen

- A endliche Menge, $|A|$ Anzahl der Elemente, **Kardinalität**.
- A, B Mengen, $|A| = |B|$ gdw
  es gibt eine Bijektion $h : A \to B$ (injektiv+surjektiv).
  $|\{0, \ldots, n-1\}| := n \qquad A$ **endlich** $\mathrm{gdw}\ \exists n : |A| = n.$
- A heißt **abzählbar**, falls A endlich ist oder $|A| = |\mathbb{N}|$, d. h. es gibt eine Abzählungsfunktion f für A.
  f ist Bijektion von
  a) $f : \{0, 1, \ldots, |A| - 1\} \to A$ $\qquad A$ endlich.
  b) $f : \mathbb{N} \to A$ $\qquad A$ unendlich.

2.1 Satz Satz von Cantor: Es gibt nicht abzählbare Mengen.

z. B. $\mathbb{R}$.

Diagonalisierungstechnik:

$A = \{f : \mathbb{N} \to \mathbb{N} \mid dom(f) = \mathbb{N}\}.$

Angenommen A ist abzählbar. Da A nicht endlich ist, gibt es eine Bijektion von $\mathbb{N}$ auf A. Sei diese Abzählung $f_0, f_1, f_2, \ldots$

Definiere $\bar{f}(x) := f_x(x) + 1$ für $x \in \mathbb{N}$.

Offenbar $\bar{f} \notin \{f_i \mid i \in \mathbb{N}\} = A$, da $\bar{f}(x) \neq f_x(x)$, aber $\bar{f}$ ist ganz auf $\mathbb{N}$ definiert, d. h. $\bar{f} \in A$ ↯

Beachte: jede Teilmenge einer abzählbaren Menge ist abzählbar.

Relationen und Funktionen

Kartesisches Produkt von Mengen $A \times B$. Tupelschreibweise: $\vec{a} = (a_1, \ldots, a_n) \in A_1 \times \cdots \times A_n$

- **Relationen**: $R \subseteq A_1 \times \cdots \times A_n$
  $(a_1, \ldots, a_n) \in R$ schreibe $Ra_1 \ldots a_n$ oder $R(a_1, \ldots, a_n)$
  $n = 2$ Infix Notation $a_1 R a_2$ (z.B. $a \leq b$)
- **Funktionen als Relationen**:
  $f : A \to B$: **(A,B, graph(f))**, wobei $graph(f) \subseteq A \times B$, für jedes $a \in A$ gibt es höchstens ein $b \in B$ mit $(a, b) \in graph(f)$.
  Schreibe $f(a) = b$ für $(a, b) \in graph(f)$ und für
  ein b mit $(a, b) \in graph(f) : \mathbf{f(a)} \downarrow$ ($f(a)$ ist definiert),
  kein b mit $(a, b) \in graph(f) : \mathbf{f(a)} \uparrow$ ($f(a)$ nicht definiert).
- Sind $(A, B, graph(f)), (B, C, graph(g))$ Funktionen.
  $g \circ f$ (**Verkettung oder Komposition**) $g \circ f : A \to C$
  $(A, C, \{(a, g(f(a)))\})$.
- **Definitionsbereich**:
  $f : A \to B : \mathbf{dom(f)} = \{\mathbf{a} \in \mathbf{A} : \mathbf{f(a)} \downarrow\}$.
- **Wertebereich**:
  $f : A \to B : \mathbf{im(f)} = \{\mathbf{f(a)} \in \mathbf{B} : \mathbf{a} \in \mathbf{A}, \mathbf{f(a)} \downarrow\}$.
- $f : A \to B$ gilt $dom(f) = A$ so ist $f : A \to B$ **total**.
- $f : A \to A$ Funktion: Die **Iteration** $f^n : A \to A$. $\{(a, b) \in A^2 : f(f(\cdots(f(a)\cdots)) \downarrow, b = f(f \cdots f(a) \cdots))\}$
  Für n = 0 ist $f^n = id$, d.h. die Identitätsfunktion auf A.

Funktionen (Forts.), Alphabete, Sprachen

- **Charakteristische Funktion von** $R \subseteq A$:
  $\chi_R : A \to \{0, 1\}$ totale Funktion mit
  $$(\chi_R(a) = 1 \text{ gdw } a \in R).$$
- **Alphabet** ist eine abzählbare Menge $\mathbf{\Sigma}$ von Buchstaben.
- $\mathbf{\Sigma}^*$ Menge der endlichen Folgen von Buchstaben (**Wörter**).
- $a \in \Sigma^*$ $a = a_1 \cdots a_n \quad n \geq 0 \quad n$ **Länge** von a,
  $|a| = n \quad n = 0 \quad \varepsilon$ leeres Wort.
- $a = a_1 \cdots a_n, b = b_1 \cdots b_m$ **Konkatenation**
  $ab = a_1 \cdots a_n b_1 \cdots b_m$ mit $|ab| = n + m$
- Präfix, Suffix, Teilwort.
- **Wortfunktionen**: $f : \Sigma^* \to \Sigma^*$,
  z. B. Spiegelung: $a = a_1 \cdots a_n \to a^{\text{mi}} = a_n \cdots a_1$.
- **Sprachen** sind Teilmengen von Σ^*.
  $L, M \subseteq \Sigma^* : L \cup M$ Vereinigung.
  $LM = \{uv : u \in L, v \in M\}$ Konkatenation.
  $L^* = \{v_1 \cdots v_n : n \in \mathbb{N}, v_i \in L, i = 1, \ldots, n\}$
  $= \bigcup_{n \geq 0} L^n$ Iteration.
- **Beachte**: Σ^* ist abzählbar.
  Programme sind Wörter (Zeichenreihen) über Alphabet $\rightsquigarrow$ Menge der Programme einer PS ist stets abzählbar.
  Also gibt es stets Funktionen, die nicht von Programmen berechnet werden können.

Versuch einer Definition von algorithmisch lösbar

Informelle Präzisierung für $f : A \longrightarrow B$ „berechenbar“.

2.2 Definition

a) Ein effektiver Prozess zur Lösung eines Problems (einer Klasse von Problemstellungen) hat folgende Eigenschaften:
   1. Der Prozess besitzt eine endliche Beschreibung.
   2. Er besteht aus Einzelschritten, die mechanisch in endlicher Zeit ausführbar sind, d. h. jeder solche Schritt kann z. B. nur von endlich vielen Daten abhängen.
   3. Er ist deterministisch, d. h. der jeweils nächste Schritt ist immer eindeutig bestimmt (falls er überhaupt existiert) kein Raten oder auswählen.
   4. Lässt die Problemstellung eine Antwort zu, so liefert der Prozess die korrekte Antwort und terminiert nach Ausführung endlich viele Einzelschritte.
      Lässt die Problemstellung keine Antwort zu, so liefert der Prozess die Antwort „?“ oder er terminiert nicht.

b) Ein **Algorithmus** ist eine endliche Beschreibung eines effektiven Prozesses (Repräsentation).

c) Eine Funktion $f : A \longrightarrow B$ heißt **effektiv berechenbar**, falls es einen effektiven Prozess gibt, der für $x \in A$ (Instanz der Problemstellung).
   1. Stoppt mit Antwort $f(x)$, falls $x \in dom(f)$.
   2. Stoppt nicht, falls $x \notin dom(f) (f(x) \uparrow)$.

3 Kalküle

Erzeugung (Generierung) von Mengen, Relationen, Funktionen

Kalkül besteht aus **(Axiome, Regeln)**

Erzeugung syntaktischer Objekte:

Sprachen, Formeln, Terme, ..., Bilder, Graphen (Wörter über Alphabet Σ)

3.1 Definition

Regel: Vorschrift um Objekt K zu erzeugen (Konklusion der Regel) sofern Objekte $\Pi_1, \ldots, \Pi_n$ $(n \geq 0)$ (Prämissen) bereits vorhanden.

Schreibweise: $R :: \dfrac{\Pi_1, \ldots, \Pi_n}{K}$.

(d. h. z. B.: $R \in ((\Sigma^*)^n \times \Sigma^*)$ für ein $n \in \mathbb{N}$).

$n = 0$: Regel ohne Prämissen ist **Axiom**.

(Axiome erlauben die Erzeugung ihrer Konklusion ohne Voraussetzungen. Initialisierung des Generierungsprozesses).

$n > 0$: **Echte Regel**.

Kalkül ist Menge von Regeln

$$K \subseteq \bigcup_{n \geq 0} (A^n \times A)$$

A Objektmenge (z.B. Σ^*, Menge von Bildern etc.)

Ableitungen

3.2 Beispiel

1. $\Sigma = \{a_1, \ldots, a_n\}$ Regelmenge: $\frac{}{\varepsilon}$, $\frac{u}{ua_1}$, $\cdots$, $\frac{u}{ua_n}$ $u \in \Sigma^*$ (unendlich viele Regeln - Regelschema -, u als Wortvariable aufgefasst).
2. mu-Kalkül: für alle Wörter $X, Y \in \{i, u, m\}^*$
   Regeln:
$$\left\{ \frac{Xi}{Xiu}, \frac{mY}{mYY}, \frac{XiiiY}{XuY}, \frac{XuuY}{XY} \right\}$$

Frage: kann man aus mi das Wort mu ableiten?

3.3 Definition

a) Sei K ein Kalkül. Eine **Ableitung** in K ist eine Folge $(\varphi_1, \varphi_2, \ldots, \varphi_n)$ von Objekten, so dass für alle $i = 1, \ldots, n$ φ_i die Konklusion einer Regel von K ist, deren Prämissen alle in $\{\varphi_1, \ldots, \varphi_{i-1}\}$ enthalten sind.

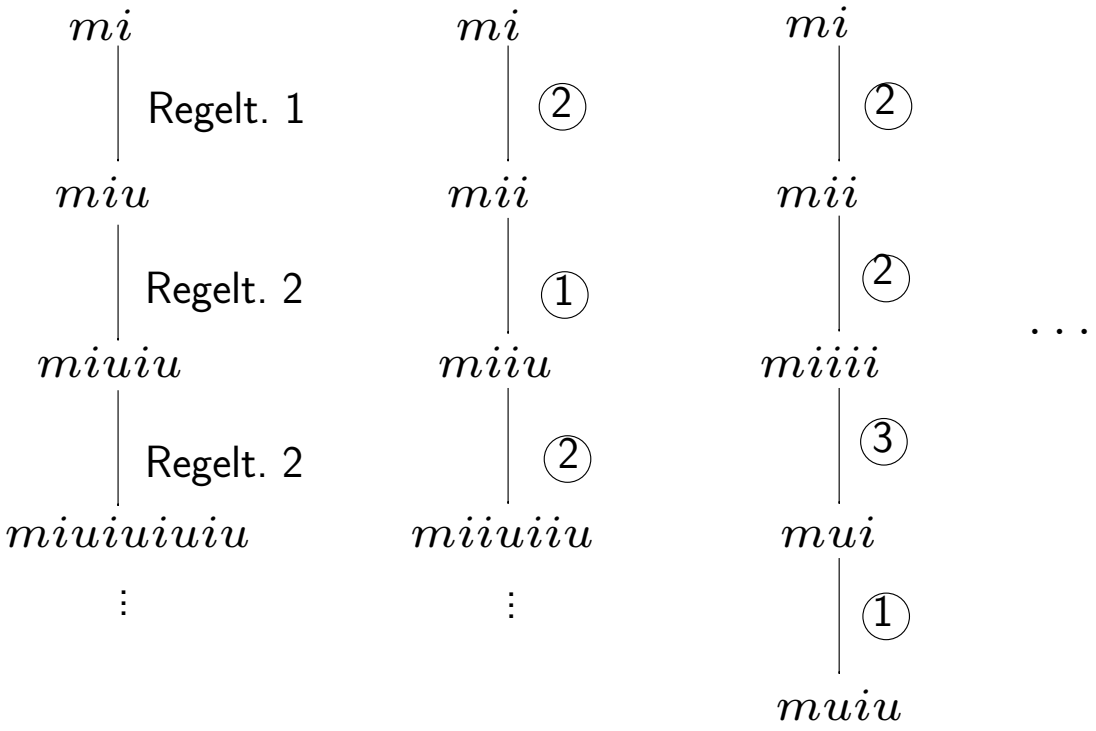
b) Ein Objekt φ ist in K **ableitbar**, falls es eine Ableitung in K mit letztem Objekt φ gibt.
   Schreibweise: $\underset{K}{\vdash} \varphi$.

c) Ein Objekt φ ist in K **aus einer Menge** M von Objekten ableitbar, falls es eine Ableitung in $K(M)$ mit letztem Objekt φ gibt, wobei $K(M)$ die Erweiterung von K um die Axiome $\frac{}{K}$ $(K \in M)$ ist. Schreibweise: $M \underset{K}{\vdash} \varphi$.

Beispiel 3.2 (Fort.)

a) Ableitbar sind $\varepsilon, a_1, a_2, \ldots, a_n, \ldots a_i a_j, \ldots$, d. h. alle Wörter aus Σ^*.

b) Ableitbar aus mi sind z. B.



$\{mi, miu, miuiu, \ldots, mii, mui, \ldots\}$

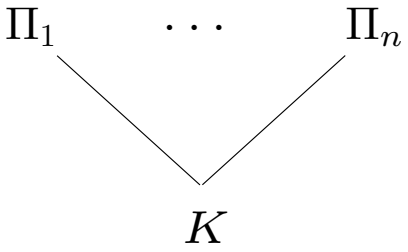
Frage: Liegt mu in dieser Menge?

Beachte: Es können stets mehrere Regeln anwendbar sein.

Darstellung von Ableitungen

Ableitungen können als Bäume dargestellt werden.
Blätter: Prämissen, Wurzel: Konklusion.

Regel:



Ableitung : Blätter: Konklusionen von Axiomen (Annahmen).
$(\varphi_1, \ldots, \varphi_n)$: Innere Knoten: Regel.
Wurzel: φ_n.

Ableitungsbäume $\rightsquigarrow$ Fragen: Tiefe, Eindeutigkeit usw.

3.4 Lemma Kompaktheitssatz für Kalküle

Sei $M \subset A$. Dann gilt:
$M \underset{K}{\vdash} \varphi$ genau dann wenn es eine endliche Teilmenge $F \subset M$ gibt mit $F \underset{K}{\vdash} \varphi$.

In dieser Tatsache liegt die vielfältige Verwendung von Kalkülen für die Fundierung zahlreicher Begriffe und Methoden der Informatik begründet.

Kalküle definieren Hüllenoperatoren

Allgemeiner Hüllenoperator für Teilmengen einer Menge bildet Teilmengen in Teilmengen ab, z.B. Transitive Hülle, Folgerungshülle usw.

$\Gamma_K : \mathcal{P}(A) \to \mathcal{P}(A)$ (Teilmengen von A werden in Teilmengen abgebildet).

Mit $\Gamma_K(M) := \{\varphi \in A : M \underset{K}{\vdash} \varphi\}$ für $M \subset A$.

Wichtige Eigenschaften für Hüllenoperatoren:
Einbettung, Monotonie, Abgeschlossenheit.

Es gilt für Γ_K:

Einbettung: für alle M : $M \subseteq \Gamma_K(M)$.

Monotonie: $M \subseteq M'$ so $\Gamma_K(M) \subseteq \Gamma_K(M')$
($M \underset{K}{\vdash} \varphi$ so auch $M' \underset{K}{\vdash} \varphi$).

Abgeschlossenheit: $\Gamma_K(\Gamma_K(M)) = \Gamma_K(M)$.

Der Ableitbarkeitsbegriff ist **transitiv**: aus $M \underset{K}{\vdash} \varphi$ und $M \cup \{\varphi\} \underset{K}{\vdash} \psi$ folgt $M \underset{K}{\vdash} \psi$.

(Die Verwendung eines ableitbaren Objekts als Voraussetzung (Blatt) in einer Ableitung kann stets eliminiert werden, d.h. Blatt wird ersetzt durch Ableitungsbaum mit entsprechender Wurzel).

Schrittweise Konstruktion von $\Gamma_K(.)$

„Konstruktive“ Sicht der Menge der ableitbaren Objekte in K:
$K \subseteq A^* \times A$ $(:= \bigcup_{n \geq 0} A^n \times A)$.

$\Gamma_K(B) := \bigcup_{i \geq 0} B_i$ mit $B_0 = B$, $B_{i+1} := B_i \cup \Gamma^1_K(B_i)$, wobei

$$\Gamma^1_K(B_i) := \{\varphi \in A \mid \text{es gibt } n \geq 0, \Pi_1, \ldots, \Pi_n \in B_i, ((\Pi_1, \ldots, \Pi_n), \varphi) \in K\}$$

„Einschritt-Ableitungen aus B_i“

i ist Maß für die „Tiefe“ des Ableitungsbaums für $\varphi \in B_i$.

Spezialfall: $A = \Sigma^*$. Zeichenreihen.

Wortersetzungssysteme (Semi-Thue-Systeme 1914).

3.5 Definition

Ein **Wortersetzungssystem** ist ein Paar (Σ, Π) mit einem endlichen Alphabet Σ und einer endlichen Menge Π von Produktionen über Σ. Eine Produktion über Σ ist eine Zeichenreihe der Form

$$l ::= r$$

(oft auch $l \longrightarrow r$ „Regel“) mit $l \neq \varepsilon, l, r \in \Sigma^*$.

Der durch (Σ, Π) definierte Kalkül $K(\Sigma, \Pi)$ auf Σ^* besteht aus allen Regeln

$$\frac{ulv}{urv} \quad l ::= r \in \Pi, u, v \in \Sigma^*$$

Wortersetzungssysteme (Fort.)

Beachte ∞-viele Regeln, endlich viele Regelschemata.

Ableitbarkeit im Wortersetzungssystem (Σ, Π):

$$x \underset{\Pi}{\vdash} y \ \text{ gdw } \ \{x\} \underset{K(\Sigma,\Pi)}{\vdash} y$$

Äquivalente Darstellung: Ableitbarkeit in Schritten $\overset{n}{\underset{\Pi}{\vdash}}$.

$x \overset{1}{\underset{\Pi}{\vdash}} y$ gdw es gibt $l ::= r \in \Pi, u, v \in \Sigma^*$ mit

$$x = ulv \text{ und } y = urv$$

$x \overset{n}{\underset{\Pi}{\vdash}} y$ gdw es gibt $z_0, \ldots, z_n \in \Sigma^*$ mit

$$x = z_0,\ z_i \overset{1}{\underset{\Pi}{\vdash}} z_{i+1}\ (i < n),\ z_n = y.$$

$n = 0$ liefert $x \overset{0}{\underset{\Pi}{\vdash}} y$ gdw $x = y$.

3.6 Lemma

$x \underset{\Pi}{\vdash} y$ gdw es gibt $n \in \mathbb{N}$ mit $x \overset{n}{\underset{\Pi}{\vdash}} y$

Beispiele

3.7 Beispiel

1. Wortersetzungssysteme

   $$aba ::= baab$$

   Regel: $\dfrac{uabav}{ubaabv}$

   Dann ist
   $aba \overset{1}{\underset{\Pi}{\vdash}} baab$ nur endlich viele Wörter ableitbar aus aba und

   $$a\underline{aba} \overset{1}{\underset{\Pi}{\vdash}} \underline{aba}ab \overset{1}{\underset{\Pi}{\vdash}} ba\underline{aba}b \overset{1}{\underset{\Pi}{\vdash}} b\underline{aba}abb \overset{1}{\underset{\Pi}{\vdash}} bba\underline{aba}bb \overset{1}{\underset{\Pi}{\vdash}} \cdots$$

   unendlich viele Wörter ableitbar.

   $$ababa \begin{array}{l} \overset{1}{\underset{\Pi}{\not\vdash}} baabba \\ \overset{1}{\underset{\Pi}{\not\vdash}} abbaab \end{array}$$

2. Grammatiken als Wortersetzungssysteme

leftside::=rightside		(EBNF) Schreibweise für Produktionen
Statement Expression:		$A ::= B$
Assignment	$\rightsquigarrow$	$A ::= C$
MethodInvocation		$A ::= D$
ClassInstanceCreationExpression		⋮
⋮		aus A ableitbar

Beispiele (Forts.)

3. $\Pi :: S \to \varepsilon,\ S \to aSbb$
   Ableitbare „Sprache“ $L = \{w \in \{a,b\}^* \mid S \underset{\Pi}{\vdash} w\}$

$$
\begin{array}{ccccccc}
S & \overset{1}{\underset{\Pi}{\vdash}} & aSbb & \overset{1}{\underset{\Pi}{\vdash}} & aaSbbbb \vdash \cdots & \overset{1}{\underset{\Pi}{\vdash}} & a^nSb^{2n} \\
\top & & \top & & \top & & \top \\
\varepsilon & & abb & & a^2b^4 & & a^nb^{2n}
\end{array}
$$

   Offenbar gilt: $L = \{a^nb^{2n} : n \in \mathbb{N}\}$.

   Wie zeigt man dies? **Induktionsbeweise**.

Nachweis von Eigenschaften ableitbarer Objekte

Induktionsprinzipien

Erinnerung: Induktionsprinzip in den natürlichen Zahlen:
Sei A eine Aussage über natürliche Zahlen: $\mathbb{N}$

A soll für alle $n \in \mathbb{N}$ richtig sein:
Methode:

Zeige: A trifft für 0 zu: **Induktionsanfang**.

Unter der Annahme, dass A für n gilt,

Zeige: A trifft auch für $n+1$ zu: **Induktionsschritt**.

Analog für Σ^*: Anfang: ε
Schritt: $|u| = n \rightsquigarrow |w| = n+1$

3.8 Satz Strukturelle Induktion (Induktion über Aufbau)

Sei A Menge, $K \subseteq \bigcup_{n>0} (A^n \times A)$ Kalkül, $B \subseteq A$.

Um eine Eigenschaft P für alle aus B in K ableitbaren Elemente (d.h. aus $\Gamma_K(B)$) zu beweisen genügt es folgendes nachzuweisen:

a) Alle Elemente in B haben die Eigenschaft P.
b) Haben $\Pi_1, \ldots, \Pi_n \in A$ die Eigenschaft P und ist $((\Pi_1, \ldots, \Pi_n), \varphi) \in K$, dann hat auch φ die Eigenschaft P.

Nachweis von Eigenschaften ableitbarer Objekte (Forts.)

Wichtige Spezialfälle:

- $B = \varnothing$, dann entfällt a).
- B endlich a) muss für endliche viele geprüft werden.

Beweismethode entspricht auch der Induktion nach Ableitungslänge.

3.9 Beispiel mu-Kalkül.

Behauptung: Angenommen $mi \underset{\Pi}{\vdash} w \rightsquigarrow 3 \not| \, |w|_i$

(3 teilt nicht die i-Länge von w (Anzahl der i-s in w)).

a) Überprüfe die Behauptung für Wort mi : $|mi|_i = 1$.

b) Regel:

$$\frac{Xi}{Xiu}, \frac{mY}{mYY}, \frac{XiiiY}{XuY}, \frac{XuuY}{XY}$$

$|Xi|_i = |Xiu|_i, 2|mY|_i = |mYY|_i,$

$|XuY|_i = |XiiiY|_i - 3, |XuuY|_i = |Y|_i$

Ist für die Prämisse die Behauptung richtig, so auch für die Konklusion.

Wegen $3 \mid |mu|_i = 0$ kann mu nicht aus mi abgeleitet werden.

Erzeugung von Funktionen: Rekursion

3.10 Beispiel 1: $S : \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+ \qquad (\mathbb{N}^+ = \mathbb{N} - \{0\})$.

$$S(x,y) = \begin{cases} x & \text{falls } x = y \\ S(x-y,y) & \text{falls } x > y \\ S(x,y-x) & \text{falls } y > x \end{cases}$$

Welche „Funktion“ soll von einer solchen „Gleichung“ definiert werden!

Semantik

Offenbar sollte z. B. $S(5,5) = 5$ und wohl $S(10,5) = S(5,5) = 5 \ldots$

Frage ist S total? Ist S “effektiv berechenbar”?

Prinzip: Initiale Werte (Rechte Seite $\downarrow$ (Axiome).

Definierte Funktionswerte führen zu neu definierten Werten.

Beispiel 2: $t : \mathbb{N} \to \mathbb{N}$ mit

$$t(n) = \begin{cases} 1 & \text{falls } n = 1 \\ t(n/2) & \text{falls } n \text{ gerade} \\ t(3n+1) & \text{falls } n \text{ ungerade} \end{cases}$$

$$\begin{aligned} t(15) &= t(46) = t(23) = t(70) = t(35) = t(106) \\ &= t(53) = t(160) = t(80) = t(20) = t(10) \\ &= t(5) = t(16) = t(8) = t(4) = t(2) = t(1) = 1 \end{aligned}$$

Gibt es totale Funktionen, die die Gleichung erfüllen?

Erzeugung von Funktionen: Rekursion (Forts.)

Wie beim Hüllenoperator ist die von einer rekursiven Gleichung definierten Funktion als die „kleinste“ Funktion, die die Gleichung erfüllt gemeint. Dabei ist für Funktionen $f, g : A \to B \qquad f \sqsubseteq g$

„f kleiner als g“ durch
$dom(f) \subseteq dom(g)$ und für alle $x \in dom(f)$ gilt $f(x) = g(x)$.

D. h. sucht man die Lösung, muss man unter den Lösungen der Gleichungen die „kleinste“, d.h. die am wenigsten definierte, bestimmen.

Beispiel 3: Gleichung für f sei $f(z) = \begin{cases} 0 & z = 0 \\ f(z-2) + \frac{1}{2}z & z \text{ gerade} \end{cases}$

Wobei $f : \mathbb{Z} \to \mathbb{Z}$.

Fange mit undefinierten Funktionen an $f_0 = \varnothing \subset \mathbb{Z} \times \mathbb{Z}$.

Setze: $f_{i+1}(z) = \begin{cases} 0 & z = 0 \\ f_i(z-2) + \frac{1}{2}z & z \text{ gerade} \neq 0 \end{cases}$

$$f_1(z) = \begin{cases} 0 & z = 0 \\ \uparrow & \text{sonst} \end{cases}$$

$$f_2(z) = \begin{cases} 0 & z = 0 \\ 1 & z = 2 \\ \uparrow & \text{sonst} \end{cases}$$

Erzeugung von Funktionen: Rekursion (Forts.)

$$f_3(z) = \begin{cases} 0 & z = 0 \\ 1 & z = 2 \\ 3 & z = 4 \\ \uparrow & \text{sonst} \end{cases} \qquad f_4(z) = \begin{cases} 0 & z = 0 \\ 1 & z = 2 \\ 3 & z = 4 \\ 6 & z = 6 \\ \uparrow & \text{sonst} \end{cases}$$

Dann ist $f_i \sqsubseteq f_{i+1}$ und $f = \bigcup_{i \geq 0} f_i$ die gesuchte Funktion.

$f_i \sqsubseteq f_{i+1}$: Induktion nach i: $i = 0$ einfach.

Induktionsschritt: Sei $i > 0$ und $z \in dom(f_i)$. Ist $z = 0$ so Beh. klar. Also ist $0 \neq z$ gerade und $f_i(z) = f_{i-1}(z-2) + \frac{1}{2}z = f_i(z-2) + \frac{1}{2}z = f_{i+1}(z)$ nach Def. von f_i, Ind. Annahme und Def. von f_{i+1}.

$dom(f) = 2\mathbb{N}$ und f erfüllt die Rekursionsgleichung und ist die kleinste (bzgl. $\sqsubseteq$) Funktion die diese Gleichung erfüllt. **Beweis!**.

Die Funktion $h(z) = \begin{cases} \sum_{i=o}^{z/2} i & z \geq 0 \text{ gerade} \\ \uparrow & \text{sonst} \end{cases}$
erfüllt die Gleichung mit selben Definitionsbereich, d.h. $f = h$.