Theoretische Informatik für Wirtschaftsinformatik und Lehramt

Motivation und Grundlagen

Priv.-Doz. Dr. Stefan Milius
stefan.milius@fau.de

Theoretische Informatik
Friedrich-Alexander Universität Erlangen-Nürnberg

SS 2016

Gliederung

Worum geht es in diesem Abschnitt? (I)

- Theoretische Informatik = wichtiges Teilgebiet der Informatik von großer praktischer Relevanz
- Zur Orientierung:
  - Teilgebiete der theoretischen Informatik
  - typische Fragestellungen in diesen Teilgebieten

Worum geht es in diesem Abschnitt? (II)

- Theoretische Informatik ist sehr formale Teildisziplin der Informatik:
  grundlegende Aussagen werden abschließend nachgewiesen.
- Betrachtete Zusammenhänge müssen formal beschrieben werden, um sie mathematischen Beweistechniken zugänglich zu machen.
- Deshalb: betrachten hier überblicksartig die relevanten mathematischen Grundbegriffe.
  (Werden später bei Benutzung nochmal kurz wiederholt.)

Lernziele

1. Teilgebiete und typische Fragestellungen der Theoretischen Informatik kennen
2. Mathematische Grundbegriffe und -techniken verstehen und anwenden können

Warum ist Theoretische Informatik relevant? (I)

- Informatik stellt eine Menge von Konzepten, Modellen, Algorithmen und Beschreibungsmethoden bereit zur:
  - Modellierung komplexer Sachverhalte und Wirkprinzipien
  - Bearbeitung, Analyse und Transformation der modellierten Strukturen und Sachverhalte durch Computer

- Beschäftigung mit Theoretischer Informatik dient dem Aufbau bzw. der Vertiefung eines informatischen Weltbildes
  - Was ist durch Modellierung, Formalisierung und Algorithmisierung im Computer *prinzipiell* möglich?
  - Wo liegen die Grenzen des Computereinsatzes?

    Warum sind mache Aufgaben auch mit den leistungsstärksten Computern überhaupt nicht (Berechenbarkeit) oder nicht effizient (Komplexität) lösbar?

Warum ist Theoretische Informatik relevant? (II)

- Beschäftigung mit Theoretischer Informatik dient dem Aufbau bzw. der Vertiefung eines informatischen Weltbildes (Fortsetzung)
  - Welche Entscheidungen kann ein Computer (ein Programm) bzgl. anderer Programme treffen (Entscheidbarkeit)?

    Gibt es Algorithmen, die folgende Fragen beantworten:
    - Wird mein Programm jemals terminieren?
    - Leistet mein Programm das Gewünschte?
- Regelungen in Prüfungsordnungen und Lehrplänen:
  - Bestandteil des Pfichtbereichs der Wirtschaftsinformatik-Ausbildung
  - Bestandteil der Ersten Staatsprüfung für das Lehramt für alle Schulformen (Klausur)
  - Bestandteil des Lehrplans (Gymnasium)
  - Relevantes Hintergrundwissen für Informatik-Lehrkräfte (alle Schulformen) und alle informatikbezogenen Berufe

Repräsentation von Information (I)

- Kernthema der Informatik:
  Verarbeitung von Daten durch Algorithmen
- Wie kann man Daten und Algorithmen in geeigneter Weise repräsentieren (darstellen, aufschreiben)?

Beispiel

Darstellungen des Datenelements „die Zahl dreizehn“

13	(Dezimaldarstellung)
1101	(Binärdarstellung)
IIIII IIIII III	(Strichdarstellung)
DREIZEHN	(Wortdarstellung)

Repräsentation von Information (II)

Welche Darstellung ist geeignet?

- Abhängig davon, was man mit den dargestellten Objekten machen will (z. B. Rechnen)
- **grundsätzlich geeignet:** alle Darstellungen durch Wörter (oder Zeichenreihen)
  (digitale Darstellungen, im Gegensatz zu Darstellungen durch Handzeichen, akustische Laute und Ähnlichem)

Kernthema der Informatik:

Verarbeitung von Daten durch Algorithmen

Daten: dargestellt durch Wörter

Beispiele

42	"ThInfWiL"	'x'	10010011

Repräsentation von Information (III)

Algorithmen (Programme): dargestellt durch Wörter

Beispiel: Euklid'scher Algorithmus zur Bestimmung des ggTs

EUCLID(a, b)
`wenn` $b = 0$
  `dann return` a
  `sonst return` EUCLID$(b, a \bmod b)$

Datenverarbeitung:

Verarbeitung von Wörtern (Datendarstellungen) durch Algorithmen/Programme (ebenfalls durch Wörter dargestellt)

Womit befasst sich die Theoretische Informatik? (I)

Berechenbarkeitstheorie: Berechenbarkeit und deren Grenzen

- Was bedeutet es, dass eine Datenverarbeitungsaufgabe (überhaupt) algorithmisch lösbar ist?
- Welche Aufgaben sind algorithmisch lösbar, welche nicht?
- Modelle für (idealisierte) Computer: endliche Automaten, Turing-Maschinen

Komplexitätstheorie:

- Lassen sich Probleme algorithmisch lösen und falls ja, wie effizient?
- Was bedeutet es, dass Datenverarbeitungsaufgaben algorithmisch „weniger“, „mehr“ oder „besonders“ abarbeitungsaufwändig lösbar sind?
- Problemklassen: Welche Aufgaben sind von welchem Aufwand?

Womit befasst sich die Theoretische Informatik? (II)

Automatentheorie: Formale Sprachen und Automatenmodelle

- Wie lassen sich Programmiersprachen syntaktisch definieren?
- Entscheidbarkeit: Wie lässt sich algorithmisch feststellen, ob ein gegebenes Programm syntaktisch korrekt ist?
- Modellierung zustandsbasierter oder reaktiver Systeme (z.B. Schaltwerke in Computer-CPUs oder eingebettete Systeme)
- Grundlage formaler Methoden in der System-/Hardware-/Software-Entwicklung

u.v.a.m.

Mathematische Grundbegriffe und -techniken

Um zu diesen (und anderen) Fragen zu fundierten Aussagen gelangen zu können, ist deren präzise Formulierung mittels mathematischer Strukturen und Techniken erforderlich, z.B.

- Mengen und Operationen auf Mengen
- Relationen und Funktionen
- Graphen und Bäume
- Induktion

Jetzt: Überblick über wesentliche Begriffe und Techniken

Mengenbegriff

Definition (= Einführung eines Konzeptes/Begriffes)

Alle unterscheidbaren Objekte aus einem Grundbereich, die eine bestimmte gemeinsame Eigenschaft haben, bilden eine Menge M.

Für jedes Objekt x muss eindeutig feststellbar sein, ob ihm diese Eigenschaft zukommt (Objekt ist Element der Menge, $x \in M$) oder nicht ($x \notin M$).

Die Objekte haben daneben mindestens eine Eigenschaft, die sie voneinander unterscheidet.

Darstellungsformen:

- verbal: Menge der Seiten eines Buches
- Aufzählung der Elemente: $M = \{3, 7, 11\}$
- Aussageform: $M = \{x \in \mathbb{R} \mid |x| < 1\}$

Mengenoperationen (I)

- **Leere Menge** $\emptyset$ $\quad x \notin \emptyset$ gilt *für alle* x
- **Durchschnitt** zweier Mengen $\quad M \cap N := \{x \mid x \in M \wedge x \in N\}$
  Konjunktion, UND, sowohl ... als auch ...
- **Vereinigung** zweier Mengen $\quad M \cup N := \{x \mid x \in M \vee x \in N\}$
  Disjunktion, einschließendes ODER
- **Differenz** zweier Mengen $\quad M \setminus N := \{x \mid x \in M \wedge x \notin N\}$
- **(kartesisches) Produkt** zweier Mengen
  $$M \times N := \{(x, y) \mid x \in M \wedge y \in N\}$$
  geordnete Paare aller Elemente von M und N
- **Potenzmenge** $\quad 2^M := \{N \mid N \subseteq M\}$ $\quad$ (auch $\wp(M)$)
  Menge aller Teilmengen von M

Mengenoperationen (II)

Beispiele (I)

$M = \{0, 1, 42, 4711\}, \qquad N = \{1, 42, 2010\}$

Durchschnitt:	$M \cap N =$	$\{1, 42\}$
Vereinigung:	$M \cup N =$	$\{0, 1, 42, 2010, 4711\}$
Differenz:	$M \setminus N =$	$\{0, 4711\}$

Mengenoperationen (III)

Beispiele (II)

$M = \{0, 42, 4711\}, N = \{1, 2010\}$

Kartesisches Produkt:	$M \times N =$	$\{(0,1),(0,2010),(42,1),$ $(42,2010),(4711,1),$ $(4711,2010)\}$
Potenzmengen:	$2^M =$	$\{\emptyset,\{0\},\{42\},\{4711\},$ $\{0,42\},\{0,4711\},\{42,4711\},$ $\{0,42,4711\}\}$
	$2^N =$	$\{\emptyset,\{1\},\{2010\},\{1,2010\}\}$

Relationen

Eine Relation setzt Elemente mehrerer Mengen zueinander in Beziehung.

Definition

Eine *n*-stellige Relation R ist eine Teilmenge des kartesischen Produkts von n Mengen $M_1, \ldots, M_n$: $R \subseteq M_1 \times \cdots \times M_n$
mit $M_1 \times \cdots \times M_n := \{(m_1, \ldots, m_n) \mid m_1 \in M_1 \wedge \cdots \wedge m_n \in M_n\}$

Ist $n = 2$, also $R \subseteq M_1 \times M_2$, spricht man von einer binären Relation.

Beispiel

Die Ordnung der natürlichen Zahlen ist eine binäre Relation:

$$\leq := \{(n, m) \in \mathbb{N} \times \mathbb{N} \mid \exists c \in \mathbb{N} \text{ mit } n + c = m\}$$

$(2, 5) \in \leq$ bzw. in der üblichen Schreibweise: $2 \leq 5$

Reflexivität, Transitivität u. Symmetrie von Relationen (I)

Definition

Eine binäre Relation R auf einer Menge M: $R \subseteq M \times M$ heißt

reflexiv	$\iff$	$\forall x \in M : x\,R\,x$
transitiv	$\iff$	$\forall x, y, z \in M : (x\,R\,y \wedge y\,R\,z \Rightarrow x\,R\,z)$
symmetrisch	$\iff$	$\forall x, y \in M : (x\,R\,y \Rightarrow y\,R\,x)$

Man beachte die übliche Schreibweise:

$$x\,R\,y \qquad \text{für } (x, y) \in R.$$

Reflexivität, Transitivität u. Symmetrie von Relationen (II)

Beispiele

- R reflexiv heißt: jedes $x \in M$ steht in Relation zu sich selbst
  Z.B. ist die „$\leq$“-Relation auf $\mathbb{N}$ reflexiv, da $x \leq x$ für alle natürlichen Zahlen gilt.
- Die „$\neq$“-Relation ist *nicht* reflexiv, da $x \neq x$ nicht gilt.
- Die „$<$“-Relation ist transitiv: aus $x < y$ und $y < z$ folgt $x < z$. Aber $<$ ist nicht reflexiv.
- Die „$\neq$“-Relation ist *nicht* transitiv; Gegenbeispiel:
  es gelten: $0 \neq 42$ und $42 \neq 0$ aber nicht $0 \neq 0$.
- Die „$=$“-Relation ist symmetrisch: aus $x = y$ folgt $y = x$.
- Die „$<$“-Relation ist *nicht* symmetrisch; Gegenbeispiel:
  es gilt $0 < 1$ aber nicht $1 < 0$.

Komposition und Potenzen einer Relation

Definition

Die Komposition (bzw. Verkettung) zweier binärer Relationen $R \subseteq M \times N$ und $S \subseteq N \times P$ ist die Relation

$$RS \subseteq M \times P \qquad \text{mit} \qquad RS := \{(a, b) \mid \exists c \in N : aRc \wedge cSb\}.$$

Potenzen einer Relation $R \subseteq M \times M$ sind wie folgt definiert:

$$\begin{aligned} R^0 &:= \{(a, a) \mid a \in M\} \\ R^1 &:= \{(a, b) \mid a, b \in M, aRb\} \\ R^2 &:= RR = \{(a, b) \mid \exists c \in M : aRc \wedge cRb\} \\ R^{n+1} &:= RR^n \quad \text{für alle } n \in \mathbb{N} \end{aligned}$$

Man schreibt:

- $R^+ = \bigcup_{n \geq 1} R^n = R^1 \cup R^2 \cup R^3 \cup \cdots$
- $R^* = \bigcup_{n \geq 0} R^n = R^0 \cup R^1 \cup R^2 \cup \cdots$

Reflexive und Transitive Hülle (I)

$R^+ = \bigcup_{n \geq 1} R^n$ heißt transitive Hülle von R.

$R^* = \bigcup_{n \geq 0} R^n$ heißt reflexive und transitive Hülle von R.

Beobachtung

Es gilt $x\ R^+\ y$ genau dann wenn $\exists n > 0, x_0, x_1, \ldots, x_n$:

$$x = x_0\ R\ x_1\ R\ x_2\ R \cdots R\ x_n = y$$

Es gilt $x\ R^*\ y$ genau dann wenn $\exists n \geq 0, x_0, x_1, \ldots, x_n$:

$$x = x_0\ R\ x_1\ R\ x_2\ R \cdots R\ x_n = y$$

(für $n = 0$, $x = x_0 = y$)

Reflexive und Transitive Hülle (II)

Beispiel

Sei $R = \{(n, n+1) \mid n \in \mathbb{N}\} \subseteq \mathbb{N} \times \mathbb{N}$, also

$$0\,R\,1, \quad 1\,R\,2, \quad 2\,R\,3, \quad \ldots$$

Dann ist R^+ die Relation $<$ auf $\mathbb{N}$,

und R^* ist die Relation $\leq$ auf $\mathbb{N}$.

Funktionen

Eine Funktion f von der Menge A in die Menge B
(Schreibweise: $f : A \to B$) ist eine Zuordnung,
die Elementen $a \in A$ einen eindeutigen Wert $f(a) \in B$ zuordnet.

Definition

Eine Funktion (oder Abbildung) $f : A \to B$ von der Menge A in die Menge B ist eine Relation $f \subseteq A \times B$, für die zusätzlich gilt:

> Jedem Wert $a \in A$ wird genau ein Wert aus B zugeordnet, der Funktionswert $f(a)$.

Auch: totale Funktion („überall definiert“).

Eine partielle Funktion erfüllt nur:

. . . höchstens ein . . . statt . . . genau ein . . .

Graphen (I)

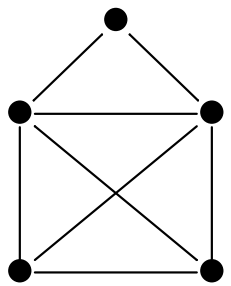
Informell: Graph besteht aus:

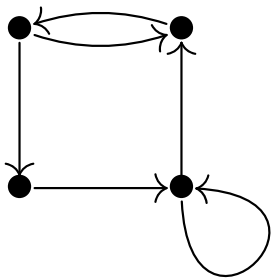
- Menge von Punkten („Knoten“)
- zwischen denen Linien („Kanten“) verlaufen

Knoten und Kanten können auch beschriftet sein („beschrifteter Graph“)

Beispiel für Graphen

- Straßen- oder Bahnnetz
- mögliche Spielzüge in einem Schachspiel
- Stammbaum der Familie





Graphen (II)

Definition

Ein Graph $G = (V, E)$ ist ein Paar bestehend aus:

- einer Menge von Knoten V („vertices“),
- einer Menge von Kanten $E \subseteq V \times V$ („edges“).

G heißt ungerichtet, falls E symmetrische Relation ist, d.h.:
für alle Knoten $a, b \in V$ gilt: aus $(a, b) \in E$ folgt $(b, a) \in E$.

Ansonsten heißt G gerichtet.

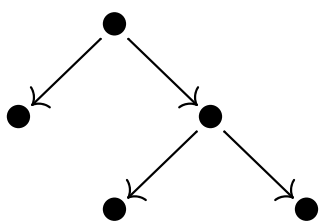
Bäume

Definition

Ein Baum $B = (V, E, v_0)$ ist ein gerichteter Graph (V, E) mit einem ausgezeichneten Knoten v_0, der Wurzel von B.

Dabei gelten noch folgende Eigenschaften:

- Die Wurzel v_0 ist Vorfahre aller Knoten, es gibt also einen Weg von v_0 zu jedem Knoten $v \in V$.
- B enthält keine Zyklen, d. h. es gibt für *keinen* Knoten $v \in V$ einen Weg der Länge > 0 von v nach v.
- Kein Knoten hat mehr als einen Vorgänger.



Vollständige Induktion (I)

Die vollständige Induktion ist eine mathematische Beweismethode, die üblicherweise eine Aussage für alle natürlichen Zahlen beweist.

Die vollständige Induktion arbeitet nach folgendem Prinzip:

- Gegeben sei P als eine Eigenschaft natürlicher Zahlen n, $P(n)$ bedeute: „n hat die Eigenschaft P“.
- Um $P(n)$ für alle $n \in \mathbb{N}$ zu beweisen, gehe wie folgt vor:
  1. **Induktionsanfang:** beweise, dass $P(0)$ gilt.
  2. **Induktionsschritt:** beweise für ein beliebiges $n \in \mathbb{N}$:

     falls $P(n)$ gilt, so gilt auch $P(n+1)$.

  3. **Induktionsschluss:** Aus den Punkten 1 und 2 folgt, dass $P(m)$ für alle $m \in \mathbb{N}$ gilt.

Vollständige Induktion (II)

Beispiel

Behauptung: Für jede natürliche Zahl n gilt:

$$\sum_{i=0}^{n} 2^i = 2^0 + 2^1 + \cdots + 2^n = 2^{n+1} - 1$$

Beweis (durch vollständige Induktion):

1. **Induktionsanfang:** Für $n = 0$ ist:
   $\sum_{i=0}^{n} 2^i = 2^0 + 2^1 + 2^2 + \cdots + 2^n = 2^0 = 1 = 2^{0+1} - 1 = 2^{n+1} - 1$
2. **Induktionsschritt:**
   Induktionsvoraussetzung: $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$
   Induktionsbehauptung: $\sum_{i=0}^{n+1} 2^i = 2^{(n+1)+1} - 1$
   Beweis (der Induktionsbehauptung):

$$\begin{aligned}\sum_{i=0}^{n+1} 2^i = 2^{n+1} + \sum_{i=0}^{n} 2^i &= 2^{n+1} + (2^{n+1} - 1) \\ &= 2 \cdot 2^{n+1} - 1 \\ &= 2^{n+2} - 1 \\ &= 2^{(n+1)+1} - 1.\end{aligned}$$

Vollständige Induktion (III)

Grund für die Wirksamkeit dieses Beweisprinzips:
Induktiver Aufbau der Menge $\mathbb{N}$ durch die Definition:

1. $0 \in \mathbb{N}$
2. Ist $n \in \mathbb{N}$, so ist auch $n + 1 \in \mathbb{N}$.
3. Außer den Elementen gemäß 1. und 2. enthält $\mathbb{N}$ keine weiteren Elemente.
   (Drittens wird im Folgenden nicht explizit angegeben!)

Definition legt Erzeugungsmechanismus für alle Elemente von $\mathbb{N}$ fest, der bei einem Induktionsbeweis der Aussage

$$\text{„}P(n) \text{ gilt für alle } n \in \mathbb{N}\text{“}$$

widergespiegelt wird.

Induktive Definition von Mengen (I)

- auch andere Mengen M sind induktiv definierbar
- **Schema:**
  1. Explizite Angabe von einigen Elementen von M
  2. Regeln zur Erzeugung weiterer Elemente $y \in M$ aus vorhandenen $x_1, ..., x_k \in M$.

Induktive Definition von Mengen (II)

Beispiel 1

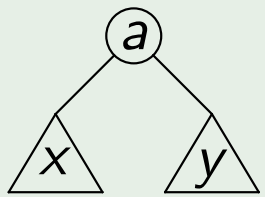
Induktive Definition einer Menge $N \subseteq \mathbb{N} \times \mathbb{N}$

1. $(0,0) \in N$ und $(1,1) \in N$.
2. Falls $(m,n) \in N$, so $(m+2,n) \in N$.
3. Falls $(m,n) \in N$, so $(m,n+2) \in N$.

Beispiel 2

Sei M eine Menge. Induktive Definition der Menge $M^{\triangle}$ der Binärbäume mit Knotenmarkierungen in M:

1. $a \in M^{\triangle}$ für alle a aus M (liefert Bäume $\textcircled{a}$)
2. Falls $a \in M$ und $x, y \in M^{\triangle}$, so $(a,x,y) \in M^{\triangle}$:



Strukturelle Induktion (I)

Mathematische Beweismethode, die eine Aussage für alle Elemente einer induktiv definierten Menge M beweist.
(*Verallgemeinerung* der vollständigen Induktion.)

Prinzip:

- Gegeben: P als Eigenschaft der Elemente von M,
  $P(x)$ bedeute: „x hat die Eigenschaft P “
- Um $P(x)$ für alle $x \in M$ zu beweisen:
  1. **Induktionsanfang:** beweise $P(x)$ für alle (in der induktiven Definition on M) explizit angegebenen Elemente $x \in M$.
  2. **Induktionsschritt:** beweise für jede Regel, mit der $y \in M$ aus $x_1, \ldots, x_k$ erzeugt werden kann:

     falls $P(x_1), \ldots, P(x_k)$ gelten, so gilt auch $P(y)$.

  3. **Induktionsschluss:** aus den Punkten 1. und 2. folgt, dass $P(z)$ für alle $z \in M$ gilt.

Strukturelle Induktion (II)

Beispiel

- Induktive Definition einer Menge $N \subseteq \mathbb{N} \times \mathbb{N}$
  1. $(0,0) \in N$ und $(1,1) \in N$.
  2. Falls $(m,n) \in N$, so $(m+2,n) \in N$.
  3. Falls $(m,n) \in N$, so $(m,n+2) \in N$.

- Behauptung: Für alle $(m,n) \in N$ gilt:

  $m+n$ ist durch 2 teilbar.

Strukturelle Induktion (III)

Beispiel, Fortsetzung

- Beweis (durch Induktion):
  1. **Induktionsanfang:**
     Element $(0,0) \in N$: Es ist $0+0=0$ durch 2 teilbar.
     Element $(1,1) \in N$: Es ist $1+1=2$ durch 2 teilbar.
  2. **Induktionsschritt:**
     Induktionsvoraussetzung:
     für $(m,n) \in N$ sei $m+n$ durch 2 teilbar.
     Dann gilt für
     Element $(m+2,n) \in N$:
     Es ist $(m+2)+n=(m+n)+2$ durch 2 teilbar.
     Element $(m,n+2) \in N$:
     Es ist $m+(n+2)=(m+n)+2$ durch 2 teilbar.

Rekursive Definitionen von Funktionen auf induktiv definierten Mengen (I)

Allgemeines Schema der rekursiven Definition einer Funktion

$$f : M \to N$$

mit einer (gemäß obigem Schema) induktiv definierten Menge M:

1. Definiere $f(x)$ für alle explizit angegebenen Elemente $x \in M$.
2. Für jede Regel, die $y \in M$ aus $x_1, \ldots, x_k \in M$ erzeugt: definiere $f(y)$ unter Verwendung von $f(x_1), \ldots, f(x_k)$.

Rekursive Definitionen von Funktionen auf induktiv definierten Mengen (II)

Beispiel 1

Fakultätsfunktion $! : \mathbb{N} \to \mathbb{N}$ mit der rekursiven Definition:

1. $0! = 1$
2. $(n+1)! = n! \cdot (n+1)$ $\qquad (n! = 1 \cdot 2 \cdot 3 \cdot \cdots \cdot n)$

Beispiel 2

Sei $M^{\triangle}$ (s.o.) die Menge der M-markierten Binärbäume.

Die Funktion $h : M^{\triangle} \to \mathbb{N}$, die jedem Binärbaum seine Höhe zuordnet, ist rekursiv definiert durch:

1. $h(a) = 0 \quad$ für alle $a \in M$.
2. $h((a, x, y)) = \max\{h(x), h(y)\} + 1$

Wörter (Zeichenreihen) (I)

Definition

Es sei Σ eine nicht-leere, endliche Menge.

- So eine Menge Σ heißt Alphabet,
  ihre Elemente heißen Symbole (oder Zeichen).
- Ein Wort (auch Zeichenreihe, Zeichenfolge, Zeichenkette)
  über Σ ist eine *endliche Folge* von Elementen aus Σ.
- Die Anzahl der Folgenglieder heißt Länge des Wortes.
- Die leere Folge (mit der Länge 0) wird mit

$$\varepsilon$$

  bezeichnet und heißt leeres Wort.

Vereinbarung: das Symbol ε ist *kein* Element von Σ.

Wörter (Zeichenreihen) (II)

- *endliche Folge* mit den Folgengliedern $a_1, a_2, \ldots, a_n$:
  - mathematisch korrekt: ein n-Tupel $(a_1, a_2, \ldots, a_n)$
  - hier schreiben wir: $a_1 a_2 \cdots a_n$
- Ist $w = a_1 a_2 \cdots a_n$ ein Wort über Σ und $b \in \Sigma$, so bezeichnet bw das Wort $b a_1 a_2 \cdots a_n$.
- Σ^* bezeichnet die Menge aller Wörter über Σ.
- Σ^+ bezeichnet die Menge $\Sigma^* \setminus \{\varepsilon\}$. (nicht leere Wörter)

Definition

Induktive Definition der Menge Σ^* (für gegebenes Σ) zusammen mit der Länge $|w| \in \mathbb{N}$ für alle $w \in \Sigma^*$:

1. $\varepsilon \in \Sigma^*$ und $|\varepsilon| = 0$.
2. Ist $b \in \Sigma$ und $w \in \Sigma^*$, so ist $bw \in \Sigma^*$ und $|bw| = |w| + 1$.

Wörter (Zeichenreihen) (III)

Beispiel

$\Sigma = \{a, b\}$ Alphabet

Wörter über Σ: a, $ababb$, ε, $a^{42} = \underbrace{aaa \cdots a}_{42 \text{ Stück}}$

$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

Rekursive Definition von Funktionen auf Wörtern

Konkatenation von Wörtern:

$$\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$$

$$\begin{aligned} \varepsilon \circ v &= v \\ (bw) \circ v &= bu, \quad \text{wobei } u = w \circ v. \end{aligned}$$

Spiegelung (Revertierung) von Wörtern:

$$R : \Sigma^* \to \Sigma^*$$

$$\begin{aligned} \varepsilon^R &= \varepsilon \\ (bw)^R &= w^R \circ b \end{aligned}$$

($R(v)$ geschrieben als v^R)

Anzahl des Vorkommens eines Zeichens in einem Wort:

$$\text{anz} : \Sigma \times \Sigma^* \to \mathbb{N} \qquad (anz(a, v) \text{ geschrieben als } |v|_a),$$

$$\begin{aligned} |\varepsilon|_a &= 0 \\ |bw|_a &= \begin{cases} |w|_a + 1, & \text{falls } a = b \\ |w|_a & \text{sonst.} \end{cases} \end{aligned}$$

Gesetze und Schreibweisen für Wörter

Satz 1.1

Für beliebige $u, v, w \in \Sigma^*$, $a \in \Sigma$ gilt:

a) $a \circ w = aw$.

b) $w \circ \varepsilon = w$.

c) $u \circ (v \circ w) = (u \circ v) \circ w$.

d) $|v \circ w| = |v| + |w|$.

e) $|v \circ w|_a = |v|_a + |w|_a$.

f) $(v \circ w)^R = w^R \circ v^R$.

Schreibweisen:

- $u \circ v \circ w = u \circ (v \circ w) = (u \circ v) \circ w$ (Assoziativität von $\circ$)
- vw für $v \circ w$ (siehe a))
- w^n für $\underbrace{ww\ldots w}_{n\text{-mal}}$ (und $w^0 = \varepsilon$) für $w \in \Sigma^*$

Teilwort

Definition

Seien $v, w \in \Sigma^*$ Wörter über Σ.

v heißt Teilwort (Teilzeichenreihe) von w, wenn es $x, y \in \Sigma^*$ gibt mit

$$w = xvy.$$

Ist $x = \varepsilon$, so heißt v auch Präfix von w.
Ist $y = \varepsilon$, so heißt v auch Postfix von w.

Beispiel

$\Sigma = \{a, \ldots, z\}$

$v = aus$ ist Teilwort/Postfix von $w = klaus$
(setze $x := kl$ und $y := \varepsilon$)

Sprachen

Definition

Sei Σ ein Alphabet.
Eine (formale) Sprache L über Σ ist eine Teilmenge von Σ^*:

$$L \subseteq \Sigma^*.$$

Die leere Teilmenge $\emptyset$ heißt leere Sprache.

Beispiel

Sei $\Sigma = \{a, b\}$. Sprachen über Σ:

$L = \{\varepsilon\} \neq \emptyset$, $L_1 = \{\varepsilon, a, ab, abab\}$, $L_2 = \{a^n \mid n \in \mathbb{N}\}$.

Entscheidungsproblem einer formalen Sprache

Zu jeder Sprache L über Σ gehört das Entscheidungsproblem:

- **Eingabe:** $w \in \Sigma^*$
- **Aufgabe:** entscheide, ob $w \in L$ gilt.

Operationen auf Sprachen (I)

- $L_1, L_2 \subseteq \Sigma^*$ seien Sprachen:

  Durchschnitt und **Vereinigung**: $L_1 \cap L_2$ bzw. $L_1 \cup L_2$.

  Komplement der Sprache $L \subseteq \Sigma^*$: $\overline{L} = \Sigma^* \setminus L$.

- **Sprachprodukt (Konkatenation von Sprachen):**

  $L_1 \circ L_2 = \{vw \mid v \in L_1, w \in L_2\}$ (Schreibweise: $L_1 L_2$)

- **Sprachpotenz:** $L^0 = \{\varepsilon\}$, $L^{n+1} = L \circ L^n$, also explizit:

  $L^n = \{w_1 w_2 \dots w_n \mid w_i \in L \text{ für } i = 1, \dots, n\}$ (für $n \in \mathbb{N}$)

- **Kleene-Stern:** $L^* = \bigcup_{n \geq 0} L^n$ (außerdem: $L^+ = \bigcup_{n \geq 1} L^n$)

- **Spiegelung:** $L^R = \{w^R \mid w \in L\}$

Operationen auf Sprachen (II)

Beispiel

Sei $\Sigma = \{a, b\}$, $L_1 = \{\varepsilon, ab, abab\}$, $L_2 = \{ba, bbb\}$.

$$L_1 \circ L_2 = \{ba, bbb, abba, abbbb, ababba, ababbbb\}$$

$$L_2^+ = \{ba, bbb, baba, babbb, bbbba, bbbbbb, \ldots\}$$

$$L_2^* = \{\varepsilon, ba, bbb, baba, babbb, bbbba, bbbbbb, \ldots\}$$

$$L_1^R = \{\varepsilon, ba, baba\}$$

Gesetze für die Operationen auf Sprachen

Satz 1.2

Für beliebige $L, L_1, L_2 \subseteq \Sigma^*$ gilt:

a) $\overline{\overline{L}} = L$.

b) $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

c) $L(L_1 \cup L_2) = LL_1 \cup LL_2$.

d) $(L^*)^* = L^*$.

e) $(L_1 \cup L_2)^R = L_1^R \cup L_2^R$.

f) $(L_1 L_2)^R = L_2^R L_1^R$.

g) $(L^*)^R = (L^R)^*$.

Zurückführen der Ausgangsfragen auf den Wortbegriff (I)

Fragen der Theoretischen Informatik	**Fragen der Theoretischen Informatik bezogen auf den Wortbegriff**
Was bedeutet es, dass eine Datenverarbeitungsaufgabe (überhaupt) algorithmisch lösbar ist?	Wie kann formal präzisiert werden, dass eine Funktion $f : \Sigma^* \to \Gamma^*$ (oder das Entscheidungsproblem einer formalen Sprache $L \subseteq \Sigma^*$) algorithmisch „berechenbar“ ist?
Welche Aufgaben sind algorithmisch lösbar, welche nicht?	Welche derartigen Funktionen sind berechenbar (welche Entscheidungsprobleme entscheidbar), welche nicht?
Was bedeutet es, dass Datenverarbeitungsaufgaben algorithmisch „weniger“, „mehr“ oder „besonders“ abarbeitungsaufwändig lösbar sind?	Wie können Berechnungsaufwände für berechenbare Funktionen klassifiziert werden?

Zurückführen der Ausgangsfragen auf den Wortbegriff (II)

Fragen der Theoretischen Informatik	**Fragen der Theoretischen Informatik bezogen auf den Wortbegriff**
Welche Aufgaben sind von welchem Aufwand?	Wie ordnen sich konkrete Funktionen/formale Sprachen in eine solche Klassifizierung ein?
Wie lassen sich Programmiersprachen syntaktisch definieren?	Wie lassen sich Mengen von Wörtern konstruktiv beschreiben (erzeugen)?
Wie lässt sich algorithmisch feststellen, ob ein gegebenes Programm syntaktisch korrekt ist?	Wie lässt sich algorithmisch feststellen, ob ein gegebenes Wort Element einer gegebenen formalen Sprache ist?

Inhalt: Motivation und Grundlagen