Theoretische Informatik

Elmar Eder

Inhalt I

Inhalt II

Inhalt III

Abschnitt 1

Einleitung

Unterabschnitt 1

Was ist Theoretische Informatik?

Was ist Theoretische Informatik?

Behandlung von Fragestellungen aus der Informatik
(Datenstrukturen, Algorithmen, . . .)
mit Methoden aus der Mathematik

Unterabschnitt 2

Teilgebiete der theoretischen Informatik

Einige Teilgebiete der theoretischen Informatik

- Sprachen
- Formale Systeme
  - Automaten
  - Chomsky-Grammatiken
  - μ-partiellrekursive Funktionen
  - Logikkalküle
  - ...
- Berechenbarkeitstheorie
- Komplexitätstheorie
- Semantik

Unterabschnitt 3

Der Begriff des Algorithmus

Der Begriff des Algorithmus

Definition

Ein Algorithmus ist eine formalisierbare Rechenvorschrift.

Ein Algorithmus kann

- zu gegebenen Eingaben (engl. input) eine
- Ausgabe (engl. output) produzieren.

Beispiel

Euklidischer Algorithmus

- Eingaben: zwei natürliche Zahlen
- Ausgabe: eine natürliche Zahl, der ggT dieser Zahlen

Formalisierungen von Algorithmen

Definition

Ein Computerprogramm ist ein in computerlesbarer Form formalisierter Algorithmus.

Definition

Eine Programmiersprache ist ein Formalismus hierfür.

Weitere Systeme zur Formalisierung von Algorithmen

- Automaten (endliche Automaten, Kellerautomaten, Turingmaschinen, ...)
- μ-partiellrekursive Funktionen
- Logikkalküle
- Chomsky-Grammatiken
- ...

Unterabschnitt 4

Fragestellungen

Fragestellungen über Algorithmen

Frage

Welche Probleme sind prinzipiell algorithmisch lösbar?
Berechenbarkeitstheorie

Frage

Mit welchem Aufwand an Ressourcen (Rechenzeit, Speicherplatz) sind sie lösbar?
Komplexitätstheorie

Frage

Wie beschreibt man das, was ein Algorithmus leistet?
Formale Semantik

Abschnitt 2

Mathematische Hilfsbegriffe

Unterabschnitt 1

Natürliche Zahlen

Natürliche Zahlen

Natürliche Zahlen

sind die Zahlen 0,1,2,3,. . .

- $\mathbb{N} = \{0, 1, 2, \dots\}$ Menge der natürlichen Zahlen
- $\mathbb{N} \setminus \{0\} = \{1, 2, 3, \dots\}$ Menge der positiven natürlichen Zahlen

Unterabschnitt 2

Logik

Logik

- Sprache zum Formulieren logischer Aussagen
- Begriff der Wahrheit
- Logisches Schließen (Beweisen)

Aussagen

Definition

Eine Aussage ist ein deutscher Satz, der entweder wahr oder falsch ist.

Beispiele

- Der Amazonas ist ein Fluss in Südamerika.
  (wahr)
- Die Zahl 5 ist gerade.
  (falsch)
- Jede gerade Zahl, die größer als 2 ist, ist die Summe zweier Primzahlen. (Goldbach'sche Vermutung)
  (nicht bekannt, ob wahr oder falsch)

Wahrheitswerte

Definition

Die Symbole w und f heißen Wahrheitswerte.

w	wahr
f	falsch

Jede Aussage hat entweder den Wahrheitswert w oder den Wahrheitswert f.

Verum und Falsum

Definition

- $\top$ (verum) bezeichnet eine Aussage, die immer wahr ist.
- $\bot$ (falsum) bezeichnet eine Aussage, die immer falsch ist.

Der Wahrheitswert von $\top$ ist w.
Der Wahrheitswert von $\bot$ ist f.

$\bot$ stellt Widerspruch dar.

Verknüpfung von Aussagen

Aus Aussagen lassen sich neue komplexere Aussagen bilden mit aussagenlogischen Verknüpfungen

Verknüpfungen werden mit Symbolen bezeichnet: Junktoren

Verknüpfung	Junktor	
nicht	$\neg$	
und	$\wedge$	
oder	$\vee$	
wenn . . . , dann . . .	$\Rightarrow$	(oder $\rightarrow$)
genau dann, wenn	$\Leftrightarrow$	(oder $\leftrightarrow$)

Verknüpfung von Aussagen

Beispiele

Sei A die Aussage „2 ist ungerade“ und sei B die Aussage „3 ist eine Primzahl“. Dann ist

$\neg A$	die Aussage „2 ist nicht ungerade“,
$A \wedge B$	die Aussage „2 ist ungerade und 3 ist eine Primzahl“,
$A \vee B$	die Aussage „2 ist ungerade oder 3 ist eine Primzahl“,
$A \Rightarrow B$	die Aussage „Wenn 2 ungerade ist, dann ist 3 eine Primzahl“,
$A \Leftrightarrow B$	die Aussage „2 ist genau dann ungerade, wenn 3 eine Primzahl ist“.

Welche dieser Aussagen sind wahr, welche falsch?

Verknüpfungen von Aussagen

Definition

Sind A und B Aussagen, so heißt

$\neg A$	die Negation von A
$A \wedge B$	die Konjunktion von A und B
$A \vee B$	die Disjunktion (oder Adjunktion) von A und B
$A \Rightarrow B$	die Implikation von B aus A
$A \Leftrightarrow B$	die Äquivalenz von A und B.

Auch Konjunktionen und Disjunktionen von mehr als zwei Aussagen sind möglich, z.B. $A \wedge B \wedge C$.

Klammerung

Bei einem Ausdruck wie $A \wedge B \vee C$ ist nicht ersichtlich welche der folgenden beiden Aussagen gemeint ist:

- Die Konjunktion von A und $B \vee C$
- Die Disjunktion von $A \wedge B$ und C.

Um solche Mehrdeutigkeiten zu vermeiden, ist ggf. zu klammern:

Beispiel

- $A \wedge (B \vee C)$
- $(A \wedge B) \vee C$.

Klammerung

Vereinbarung

- $\neg$ bindet stärker als $\wedge$ und $\vee$
- $\wedge$ und $\vee$ binden stärker als $\Rightarrow$ und $\Leftrightarrow$.

Beispiel

$\neg A \wedge B \Rightarrow C \vee D$ bedeutet dasselbe wie $((\neg A) \wedge B) \Rightarrow (C \vee D)$.

Ketten von Implikationen und Äquivalenzen

Bei Ketten von Implikationen und Äquivalenzen ist die Terminologie leider nicht einheitlich.

In der Mathematik

ist $A_1 \Rightarrow A_2 \Rightarrow \ldots \Rightarrow A_n$ eine Abkürzung für
$(A_1 \Rightarrow A_2) \wedge (A_2 \Rightarrow A_3) \wedge (A_3 \Rightarrow A_4) \wedge \cdots \wedge (A_{n-1} \Rightarrow A_n)$.

In der formalen Logik

steht $A_1 \to A_2 \to \ldots \to A_n$ für
$A_1 \to (A_2 \to (A_3 \to (\ldots (A_{n-1} \to A_n) \ldots)))$.

In der Mathematik

ist $A_1 \Leftrightarrow A_2 \Leftrightarrow \ldots \Leftrightarrow A_n$ eine Abkürzung für
$(A_1 \Leftrightarrow A_2) \wedge (A_2 \Leftrightarrow A_3) \wedge (A_3 \Leftrightarrow A_4) \wedge \cdots \wedge (A_{n-1} \Leftrightarrow A_n)$.

Wahrheitswerttafeln

Der Wahrheitswert einer zusammengesetzten Aussage errechnet sich aus den Wahrheitswerten der Teilaussagen aufgrund der folgenden Wahrheitswerttafeln.

A	$\neg A$
w	f
f	w

A	B	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
w	w	w	w	w	w
w	f	f	w	f	f
f	w	f	w	w	f
f	f	f	f	w	w

Prädikate

Prädikate sind Eigenschaften oder Relationen, die auf jeweils eine feste Anzahl von Gegenständen aus einer gegebenen Menge von Gegenständen (Individuenbereich) bezogen werden können.

Beispiele (für Prädikate, die sich auf natürliche Zahlen beziehen)

- $P(x)$ bedeute „x ist eine Primzahl“. Dann ist P ein einstelliges Prädikat. $P(4)$ bedeutet „4 ist eine Primzahl“ und ist falsch.
- $Q(x, y)$ bedeute „$x < y$“. Dann ist Q ein zweistelliges Prädikat.
- $R(x, y, z)$ bedeute „x hat den Rest y modulo z“.
  Dann ist R ein dreistelliges Prädikat.
  $R(9, 1, 4)$ ist wahr, weil $9 \mod 4 = 1$.

Prädikate

Für ein n-stelliges Prädikat P und Gegenstände $a_1, \ldots, a_n$ muss $P(a_1, \ldots, a_n)$ entweder wahr oder falsch sein.

Notation

Für $P(a_1, \ldots, a_n)$ schreiben wir gelegentlich einfach $Pa_1 \ldots a_n$.

Extensionalität

Definition

Zwei n-stellige Prädikate P und Q heißen einander extensional gleich, wenn für alle Gegenstände $a_1, \ldots, a_n$ gilt

$$P(a_1, \ldots, a_n) \iff Q(a_1, \ldots, a_n) \,.$$

In der Logik werden extensional gleiche Prädikate meist als identisch betrachtet.

Variablen und Aussageformen

Der Satz

x ist der Vater von y.

ist keine Aussage, da die Zeichen x und y keine feste Bedeutung haben und daher an sich nicht feststeht, ob dieser Satz wahr oder falsch ist.

Definition

x und y heißen Variablen.

Definition

Ein Satz, der Variablen enthalten darf und bei Ersetzung der Variablen durch Bezeichnungen von Gegenständen in eine Aussage übergeht, heißt eine Aussageform.

Komprehension

Eine Aussageform F, die höchstens die Variablen $x_1, \ldots, x_n$ enthält, definiert ein n-stelliges Prädikat P durch

$$P(x_1, \ldots, x_n) \iff F \ .$$

Beispiel

Die Aussageform „x ist Vater von y“ definiert das Prädikat P durch

$$P(x, y) \iff x \text{ ist Vater von } y \ .$$

Quantifizierung

Ist F eine Aussageform und ist x eine Variable, so können daraus die folgenden neuen Aussageformen gebildet werden:

$\forall x\, F$ *Dies bedeutet „Für alle x gilt F“.*

$\exists x\, F$ *Dies bedeutet „Es gibt ein x so, dass F gilt“.*

Beispiel

$\forall x\,(x^2 > y)$ ist eine Aussageform über dem Individuenbereich der reellen Zahlen. Sie ist wahr für negative Werte von y und falsch sonst.

Komplexe Aussagenformen

Mit Hilfe der Junktoren und Quantoren können aus einfachen Aussageformen komplexere Aussagenformen aufgebaut werden.

Beispiel

$$\forall x_0 \forall \epsilon(\epsilon > 0 \Rightarrow \exists \delta(\delta > 0 \wedge \forall x(|x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \epsilon)))$$

drückt aus, dass die Funktion f stetig ist.

Eingeschränkte Quantifizierung

In der Mathematik und im täglichen Leben schränkt man Quantifizierungen oft auf einen Teilbereich des Individuenbereichs ein.

Beispiel

1. Für alle ϵ, die größer als 0 sind, gilt $P(\epsilon)$.
2. Es gibt ein δ, das größer als 0 ist, sodass $Q(\delta)$ gilt.

Schreibweise

1. $\forall \epsilon > 0 \colon P(\epsilon)$
2. $\exists \delta > 0 \colon Q(\delta)$

Schreibweise (mit vergrößertem $\wedge$- bzw. $\vee$-Zeichen)

1. $\bigwedge_{\epsilon > 0} P(\epsilon)$
2. $\bigvee_{\delta > 0} Q(\delta)$

Eingeschränkte Quantifizierung

Kein neues sprachliches Ausdrucksmittel, sondern nur Schreibersparnis:

$$\bigwedge_{\epsilon>0} P(\epsilon) \quad \text{ist Abkürzung für} \quad \forall\epsilon(\epsilon > 0 \Rightarrow P(\epsilon))$$

$$\bigvee_{\delta>0} Q(\delta) \quad \text{ist Abkürzung für} \quad \exists\delta(\delta > 0 \wedge Q(\delta))$$

Z.B. ist

$$\bigwedge_{x_0} \bigwedge_{\epsilon>0} \bigvee_{\delta>0} \bigwedge_{x} (|x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \epsilon)$$

eine Abkürzung für die Stetigkeitsaussage

$$\forall x_0 \forall\epsilon(\epsilon > 0 \Rightarrow \exists\delta(\delta > 0 \wedge \forall x(|x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \epsilon)))$$

Noch kürzer: $$\bigwedge_{x_0} \bigwedge_{\epsilon>0} \bigvee_{\delta>0} \bigwedge_{\substack{x \\ |x-x_0|<\delta}} |f(x) - f(x_0)| < \epsilon$$

Beweise

Um nachzuweisen, dass eine Aussage wahr ist oder dass eine Aussageform gültig, d.h. für alle Werte der Variablen wahr ist, verwendet man

Das Prinzip des logischen Schließens

Axiome sind Aussagen oder Aussageformen, die als wahr oder gültig angenommen werden.

Schlussregeln sind Regeln, mit denen aus bereits als wahr oder gültig nachgewiesenen Aussagen oder Aussageformen neue Aussagen oder Aussageformen als wahr oder gültig nachgewiesen werden.

Beweise

Axiome werden einfach so hingeschrieben.
Schlussregeln werden folgendermaßen geschrieben.

$$\frac{A_1 \quad \dots \quad A_n}{B}$$

bedeutet: Wenn Aussagen oder Aussagenformen $A_1, \dots, A_n$ bereits bewiesen sind, dann kann man auf die Gültigkeit von B schließen.
$A_1, \dots, A_n$ heißen die Prämissen des Schlusses, und B heißt die Konklusion des Schlusses.

Der indirekte Beweis

Um A zu beweisen, nimmt man $\neg A$ an und leitet daraus einen Widerspruch ab.

Gleichheit

In der Logik mit Gleichheit wird ein zweistelliges Prädikat, das Gleichheitsprädikat $=$ gesondert behandelt. Es wird in Infixschreibweise geschrieben und genügt den folgenden Axiomen:

Gleichheitsaxiome

- $x = x$
- $x = y \Rightarrow (Fx \Rightarrow Fy)$

Logik höherer Stufe

In der Logik erster Stufe

- dürfen Prädikate nur auf Objekte angewendet werden.
- darf nur über Objekte quantifiziert werden.

In der Logik zweiter Stufe darf auch über Prädikate quantifiziert werden.

Beispiel (Das Axiom der vollständigen Induktion)

$$\forall P\Big(P(0) \wedge \forall x\big(P(x) \Rightarrow P(f(x))\big) \Rightarrow \forall x\, P(x)\Big)$$

In Logiken höherer Stufen gibt es auch Prädikatenprädikaten, usw.

Der Beweis durch vollständige Induktion

Satz

Für alle $n \in \mathbb{N}$ gilt $P(n)$.

Beweis durch vollständige Induktion nach n

- Induktionsanfang $P(0)$
  Beweis: ...
- Induktionsvoraussetzung $P(n)$
- Induktionsbehauptung $P(n')$
  Beweis: ... (darf Induktionsvoraussetzung benutzen)

Nach dem Prinzip der vollständigen Induktion gilt daher $P(n)$ für alle $n \in \mathbb{N}$. □

Der Beweis durch vollständige Induktion (Beispiel)

Satz

Für alle $n \in \mathbb{N}$ *gilt* $\sum_{i=0}^{n} 1 = n + 1$.

Beweis durch vollständige Induktion nach n

- Induktionsanfang $\sum_{i=0}^{0} 1 = 0 + 1$
  Beweis: $\sum_{i=0}^{0} 1 = 1$ und $0 + 1 = 1$.
- Induktionsvoraussetzung $\sum_{i=0}^{n} 1 = n + 1$
- Induktionsbehauptung $\sum_{i=0}^{n'} 1 = n' + 1$
  Beweis: $\sum_{i=0}^{n'} 1 = \sum_{i=0}^{n} 1 + 1 \overset{\text{I.V.}}{=} n' + 1$

Nach dem Prinzip der vollständigen Induktion gilt daher $\sum_{i=0}^{n} 1 = n + 1$ für alle $n \in \mathbb{N}$. □

Definition

Einführung eines neuen Zeichens oder einer neuen Notation als Abkürzung für ein Objekt, eine Funktion, ein Prädikat, usw.

Explizite Definition

1. $a := t$
2. $f(x_1, \ldots, x_n) := t$ mit Komprehension
3. $P(x_1, \ldots, x_n) :\Leftrightarrow F$ mit Komprehension

Beispiele

1. $a := \int_{x=0}^{c} f(x)\mathrm{d}x$
2. $f(x, y) := axy + bx + cy + d$
3. $P(x) :\Leftrightarrow x$ ist Primzahl

Definition

Implizite Definition

Definiere X durch eine Aussageform, die für genau ein X erfüllt ist.

Beispiele

- $a \in \mathbb{R}$ sei definiert durch
  $a^2 = 2 \wedge a > 0$.
- $f : \mathbb{R} \to \mathbb{R}$ sei definiert durch
  $f(x)^3 = x$.
- $\arctan : \mathbb{R} \to \mathbb{R}$ sei definiert durch
  $\forall x(\arctan'(x) = \frac{1}{1+x^2}) \wedge \arctan(0) = 0$.
- Das einstellige Prädikat P auf $\mathbb{N}$ sei definiert durch
  $P(0) \wedge \neg P(1) \wedge \forall x(P(x) \Leftrightarrow P(x+2))$.

Definition

Implizite nicht-eindeutige Definition

Definiere X durch eine Aussageform, die für mindestens ein X erfüllt ist.

Beispiele

1. Sei a eine Wurzel von $1 + i$, d.h. $a^2 = 1 + i$.
2. Für jede komplexe Zahl x sei $f(x)$ eine komplexe Zahl mit $f(x)^2 = x$.

Eine Definition wie 2. erfordert im Allgemeinen die Annahme des Auswahlaxioms der Mengenlehre.

Unterabschnitt 3

Mengen

Der Mengenbegriff

Definition

Eine Menge ist eine Zusammenfassung von wohlunterscheidbaren Dingen.

Definition

Diese Dinge heißen die Elemente der Menge.

Eine Menge kann ein Element nur einmal enthalten.

Notationen

Notationen

$x \in A$	Das Ding x ist Element der Menge A.
$x \notin A$	$\neg\, x \in A$
$\{x_1, \ldots, x_n\}$	die Menge mit den Elementen $x_1, \ldots, x_n$.
$\emptyset$	die leere Menge $\{\}$.
$\{x \mid P(x)\}$	die Menge, deren Elemente diejenigen Dinge x sind, für die $P(x)$ gilt. Komprehension

Es gilt:

$$\{x_1, \ldots, x_n\} = \{x \mid x = x_1 \vee \cdots \vee x = x_n\}$$
$$\emptyset = \{x \mid \bot\}$$

Teilmengen

Definition

Eine Menge A heißt Teilmenge einer Menge B, in Zeichen $A \subseteq B$, wenn gilt

$$\forall x(x \in A \Rightarrow x \in B) \ .$$

Definition

Die Menge der Teilmengen einer Menge A heißt Potenzmenge von A. Sie wird mit $\mathcal{P}(A)$ oder 2^A bezeichnet.

Operationen auf Mengen

Operationen auf Mengen

$$A \cap B := \{x \mid x \in A \wedge x \in B\} \quad \text{Durchschnitt}$$
$$A \cup B := \{x \mid x \in A \vee x \in B\} \quad \text{Vereinigung}$$
$$A \setminus B := \{x \mid x \in A \wedge x \notin B\} \quad \text{Differenz}$$

Operationen auf Mengen von Mengen

$$\bigcap A := \{x \mid \bigwedge_{X \in A} x \in X\} \quad \text{Durchschnitt}$$
$$\bigcup A := \{x \mid \bigvee_{X \in A} x \in X\} \quad \text{Vereinigung}$$

Paare

Definition

- Ein Paar ist eine Zusammenfassung (a, b) von zwei Dingen a und b.
- Diese Dinge heißen Komponenten des Paares.

Gegensatz zu Mengen

- Ein Paar kann ein Ding zweimal enthalten: (a, a)
- Es kommt auf die Reihenfolge an: Im Allgemeinen ist $(a, b) \neq (b, a)$.

Tupel

Definition

Ein n-Tupel ist eine Zusammenfasung $(a_1, \ldots, a_n)$ von n Dingen $a_1, \ldots, a_n$.

Ein Paar ist also ein 2-Tupel.

Definition

Das kartesische Produkt von Mengen $A_1, \ldots, A_n$ ist definiert als

$$A_1 \times \cdots \times A_n := \{(a_1, \ldots, a_n) \mid a_1 \in A_1 \wedge \cdots \wedge a_n \in A_n\}$$

$A^n := A \times \cdots \times A \qquad (n\ A\text{'s}).$

Relationen, Prädikate

Definition

- Eine n-stellige Relation ist eine Teilmenge eines kartesischen Produkts $A_1 \times \cdots \times A_n$.
- Eine n-stellige Relation auf einer Menge A ist eine Teilmenge von A^n.

Definition

In der Logik ist

- ein Individuenbereich eine nichtleere Menge D.
- ein n-stelliges Prädikat auf D eine n-stellige Relation auf D.

Funktionen

Definition

Eine Funktion f ist eine Zuordnung:

- Jedem Element a einer Menge A wird ein und nur ein Element einer Menge B zugeordnet.
- Man schreibt dann $f : A \to B$.
- Das dem a zugeordnete Element der Menge B bezeichnet man mit $f(a)$.

Definition

Der Graph von $f : A \to B$ ist die Menge $\{(a, f(a)) \mid a \in A\}$.

Funktionen

In der Mengenlehre wird meist eine Funktion mit ihrem Graphen gleichgesetzt. Dann gilt die

Definition

Eine Funktion oder Abbildung ist eine Menge f von Paaren, sodass gilt

$$\forall x \forall y_1 \forall y_2 \left((x, y_1) \in f \wedge (x, y_2) \in f \Rightarrow y_1 = y_2 \right) .$$

Funktionen

Definition

Unter einer *n*-stelligen Funktion auf einer Menge D verstehen wir eine Funktion $f : D^n \to D$.

Definition

- Der Definitionsbereich einer Funktion $f : A \to B$ ist die Menge A, also $\{a \mid \exists b\ (a, b) \in f\}$.
- Der Wertebereich oder Bildbereich von f ist die Menge aller Werte $f(a)$, also $\{b \mid \exists a\ (a, b) \in f\}$

Injektive und surjektive Funktionen

Definition

- Eine Funktion $f : A \to B$ heißt injektiv, wenn

$$\bigwedge_{x,y \in A} (f(x) = f(y) \Rightarrow x = y).$$

- Eine Funktion $f : A \to B$ heißt surjektiv, wenn

$$\bigwedge_{y \in B} \bigvee_{x \in A} f(x) = y.$$

- Eine Funktion $f : A \to B$ heißt bijektiv, wenn f injektiv und surjektiv ist.

Mächtigkeit von Mengen

Definition

- Eine Menge A heißt höchstens gleichmächtig zu einer Menge B, wenn es eine injektive Abbildung $f : A \to B$ gibt. Wir schreiben dann $|A| \leq |B|$.
- Zwei Mengen A und B heißen gleichmächtig, wenn es eine bijektive Abbildung $f : A \to B$ gibt. Wir schreiben dann $|A| = |B|$.

Satz

- *„höchstens gleichmächtig zu“ ist eine Quasiordnung (engl. preorder), d.h. eine reflexive transitive Relation, auf der Klasse der Mengen.*
- *„gleichmächtig“ ist eine Äquivalenzrelation auf der Klasse der Mengen.*

Mächtigkeit von Mengen

Satz (Cantor, Bernstein, Schröder)

Wenn $|A| \leq |B|$ *und* $|B| \leq |A|$, *dann* $|A| = |B|$.

Satz

Eine Menge A *ist genau dann höchstens gleichmächtig wie eine Menge* B, *wenn* $A = \emptyset$ *ist oder es eine surjektive Abbildung* $f : B \to A$ *gibt.*

Definition

Wenn $|A| \leq |B|$, aber nicht $|A| = |B|$ ist, schreibt man $|A| < |B|$ und sagt B ist mächtiger als A.

Abzählbare Mengen

Definition

Eine Menge A heißt abzählbar, wenn A höchstens gleichmächtig zu $\mathbb{N}$ ist.

Beispiel

Jede endliche Menge ist abzählbar.

Satz

Eine Menge A ist genau dann abzählbar,
wenn A leer ist oder eine surjektive Abbildung $\theta\colon \mathbb{N} \to A$ existiert.

Abzählbare Mengen

Beispiel ($\mathbb{N}$ ist abzählbar)

Surjektive Abzählungsfunktion $\theta\colon \mathbb{N} \to \mathbb{N}$:

$$\theta(n) := n.$$

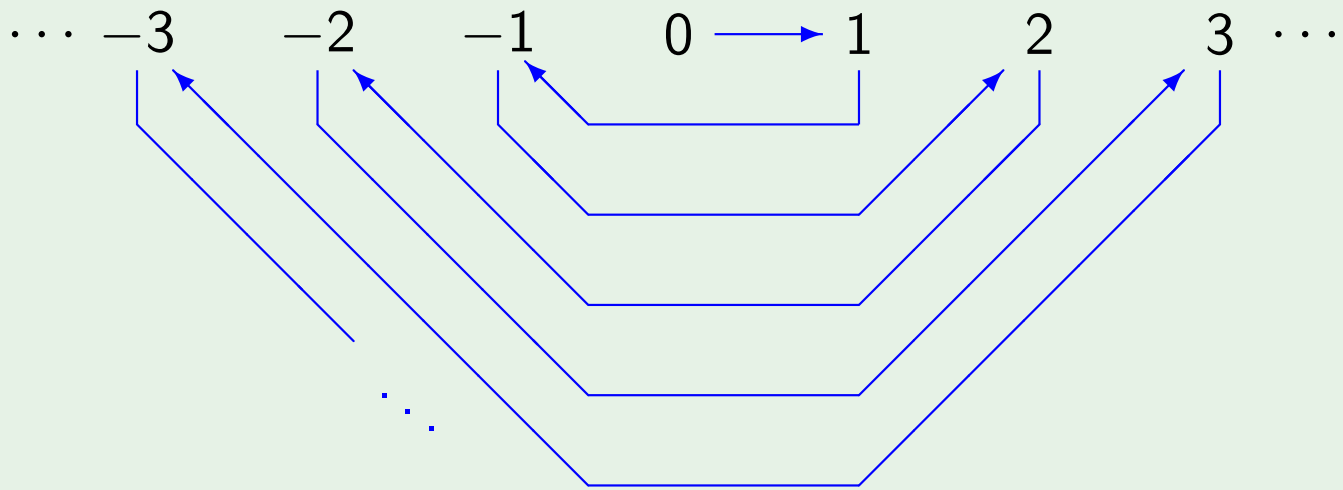
$$0 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4 \longrightarrow \cdots$$

Abzählbare Mengen

Beispiel ($\mathbb{Z}$ ist abzählbar)

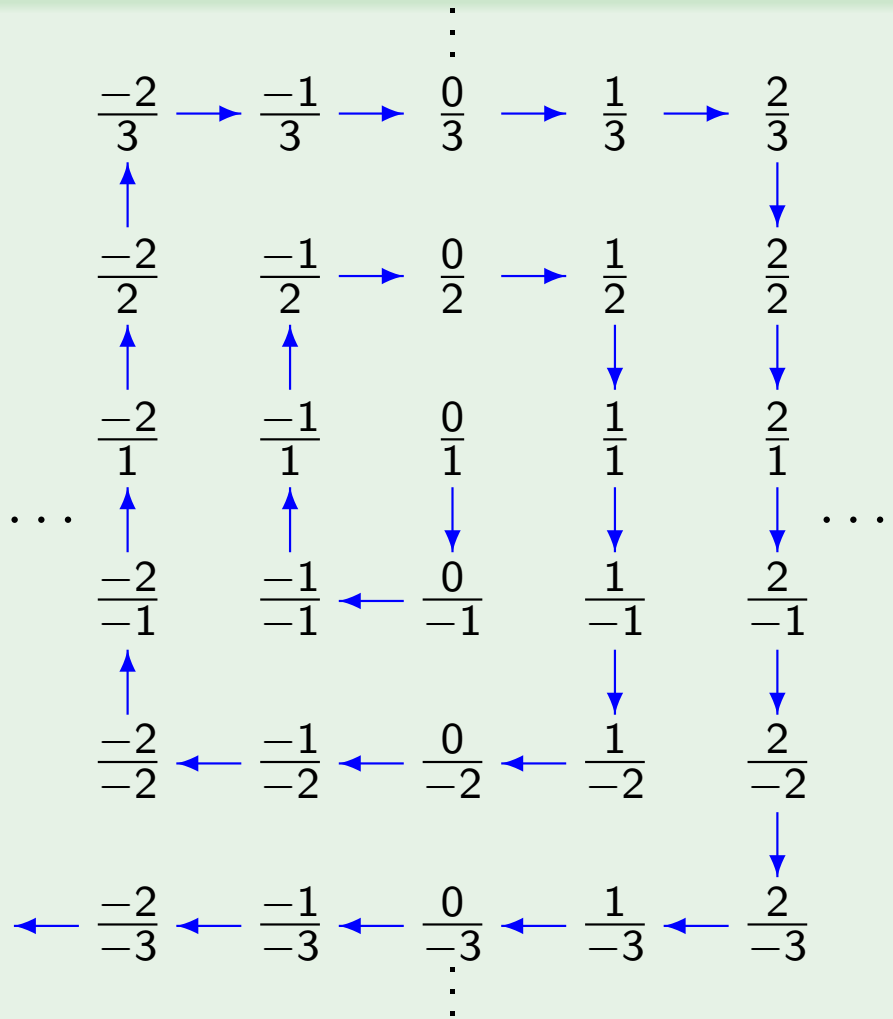
Surjektive Abzählungsfunktion $\theta\colon \mathbb{N} \to \mathbb{Z}$:

$$\theta(2n) := -n$$
$$\theta(2n+1) := n+1.$$



Abzählbare Mengen

Beispiel ($\mathbb{Q}$ ist abzählbar)



Abzählbare Mengen

Satz

- *Jede Teilmenge einer abzählbaren Menge ist abzählbar.*
- *Die Vereinigung zweier abzählbaren Mengen ist abzählbar.*
- *Die Vereinigung einer abzählbaren Menge von abzählbaren Mengen ist abzählbar.*
- *Das kartesische Produkt $A_1 \times \cdots \times A_n$ von endlich vielen abzählbaren Mengen $A_1, \ldots, A_n$ ist abzählbar.*

Das Cantor'sche Diagonalverfahren

Satz (Cantor)

Die Menge der reellen Zahlen ist nicht abzählbar.

Das Cantor'sche Diagonalverfahren

Beweis mit dem Cantor'schen Diagonalverfahren, indirekt

- Angenommen, $\mathbb{R}$ wäre abzählbar.
- Dann wäre auch die Teilmenge $[0, 1)$ abzählbar.
- Abzählung
$$\begin{aligned} &0, a_{11}\, a_{12}\, a_{13} \ldots \\ &0, a_{21}\, a_{22}\, a_{23} \ldots \\ &0, a_{31}\, a_{32}\, a_{33} \ldots \\ &\quad\vdots \end{aligned}$$
- Diagonale $a_{11}\, a_{22}\, a_{33} \ldots$

  abgeändert zu $0, b_1\, b_2\, b_3 \ldots$ $\qquad b_k := \begin{cases} 3, & \text{falls } a_{kk} = 6 \\ 6 & \text{sonst} \end{cases}$
- Dieser Dezimalbruch kommt in der Abzählung nicht vor.
- Zahl aus $[0, 1)$. Kommt in Abzählung nicht vor. Widerspruch

Das Cantor'sche Diagonalverfahren

Beispiel

- Abzählung

$$
\begin{aligned}
&0,\mathbf{6}589732\ldots\\
&0,1\mathbf{0}92845\ldots\\
&0,32\mathbf{1}1691\ldots\\
&0,244\mathbf{8}123\ldots\\
&0,567\mathbf{7}980\ldots\\
&0,448276\mathbf{1}\ldots\\
&0,500179\mathbf{9}\ldots\\
&\quad\vdots
\end{aligned}
$$

- Diagonale $\quad 6018969\ldots$
  abgeändert zu $\quad 0,3666636\ldots$

Das Cantor'sche Diagonalverfahren

Satz

Die Menge $\mathbb{N}^{\mathbb{N}}$ der Funktionen $f : \mathbb{N} \to \mathbb{N}$ ist nicht abzählbar.

Das Cantor'sche Diagonalverfahren

Beweis mit dem Cantor'schen Diagonalverfahren, indirekt

- Angenommen, $\mathbb{N}^{\mathbb{N}}$ wäre abzählbar.
- Dann gäbe es eine Abzählung $f_0, f_1, f_2, \ldots$ von $\mathbb{N}^{\mathbb{N}}$.
- $$\begin{array}{llll} f_0(0) & f_0(1) & f_0(2) & \\ f_1(0) & f_1(1) & f_1(2) & \ldots \\ f_2(0) & f_2(1) & f_2(2) & \\ & \vdots & & \ddots \end{array}$$
- Veränderte Diagonale $f(k) := f_k(k) + 1$

$$f_0(0) + 1 \quad f_1(1) + 1 \quad f_2(2) + 1 \quad \ldots$$

Das Cantor'sche Diagonalverfahren

Beweis (Fortsetzung)

- Die Funktion f ist keine der Funktionen f_n der Abzählung, da sonst
$$f(n) = f_n(n) + 1 = f(n) + 1.$$
- Widerspruch zur Annahme, dass $f_0, f_1, f_2, \ldots$ eine Abzählung von $\mathbb{N}^{\mathbb{N}}$ ist. □

Mengen von Mengen

Mengen dürfen auch selbst Elemente von Mengen sein:

$$\{\emptyset\}$$
$$\{\{\emptyset\}\}$$
$$\{\emptyset, \{\emptyset\}\}$$

sind Mengen.

Mengen von Mengen

In der Mengenlehre kommt man mit dem Mengenbildungsbegriff allein aus um praktisch alle mathematischen Begriffe zu definieren. Z.B. die natürlichen Zahlen:

$$
\begin{aligned}
0 &:= \emptyset \\
1 &:= \{0\} \\
2 &:= \{0, 1\} \\
3 &:= \{0, 1, 2\} \\
&\vdots \\
\omega = \mathbb{N} &= \{0, 1, 2, \dots\}
\end{aligned}
$$

Auch Paare definiert man in der Mengenlehre als Mengen:
$(a, b) := \{\{a\}, \{a, b\}\}$.

Satz von Cantor

Satz (Cantor)

Für jede Menge A gilt $|A| < |\mathcal{P}(A)|$.

Beweis.

indirekt und mit dem Cantor'schen Diagonalverfahren:

- Andernfalls gäbe es eine Surjektion $f : A \to \mathcal{P}(A)$.
- Sei $B := \{x \in A \mid x \notin f(x)\}$. (1)
- Dann ist $B \in \mathcal{P}(A)$.
- Da f surjektiv ist, gäbe es ein $a \in A$ mit $f(a) = B$. (2)
- $a \in B \overset{(1)}{\iff} a \notin f(a) \overset{(2)}{\iff} a \notin B$. Widerspruch

□

Eine Antinomie der Mengenlehre

Sei $A := \{x \mid x \notin x\}$.

$$A \in A \Leftrightarrow A \notin A$$

Widerspruch

Man darf nicht erlauben, dass eine Menge sich selbst enthält.

Ausweg

Unterscheidung zwischen **Klassen** und **Mengen**

- Eine Klasse ist eine Zusammenfassung von Mengen.
- Eine Menge ist eine „kleine“ Klasse.

Nur Mengen sind zugelassen als Elemente von Klassen oder von Mengen.

Die Axiome der Mengenlehre

Extensionalitätsaxiom

$\forall x(x \in A \Leftrightarrow x \in B) \Rightarrow A = B$

Komprehensionsaxiom

$a \in \{x \mid Fx\} \Leftrightarrow Fa \wedge a$ ist Menge

Paarmengenaxiom

Für alle Mengen A und B ist auch $\{A, B\}$ eine Menge.

Die Axiome der Mengenlehre

Vereinigungsmengenaxiom

Für jede Menge A ist auch $\bigcup A$ eine Menge.

Unendlichkeitsaxiom

Es gibt eine Menge N so, dass $\emptyset \in N$ und für jedes $n \in N$ auch $n' \in N$.

Ersetzungsaxiom

Wenn f eine Funktion ist, deren Definitionsbereich eine Menge ist, dann ist auch ihr Bildbereich eine Menge.

Die Axiome der Mengenlehre

Fundierungsaxiom

$\forall x(x \neq \emptyset \Rightarrow \exists y(y \in x \wedge x \cap y = \emptyset))$

Potenzmengenaxiom

Die Potenzmenge einer Menge ist eine Menge.

Auswahlaxiom

Für jede Menge x gibt es eine Menge y, die eine Funktion ist, derart, dass für jede Menge z, die eine nichtleere Teilmenge von x ist, gilt: $y(z) \in z$.

Unterabschnitt 4

Graphen und Bäume

Graphen

Definition

- Ein gerichteter Graph ist ein Paar (V, E) von Mengen mit $E \subseteq V \times V$.
- Die Elemente von V heißen Knoten (nodes, vertices).
- Die Elemente von E heißen Kanten (edges).

Definition

Wenn (V, E) ein Graph ist und $(v, w) \in E$, dann heißt v Vorgänger von w und w Nachfolger von v.

Wege

Definition

- Ein Weg in einem gerichteten Graphen (V, E) ist eine endliche oder unendliche Folge $(v_0, v_1, \dots)$ von Knoten derart, dass für alle $j > 0$ gilt $(v_{j-1}, v_j) \in E$.
- Ein Weg $(v_0, \dots, v_n)$ heißt ein Weg von v_0 nach v_n.

Definition

In einem Graphen heißt ein Knoten w von einem Knoten v aus erreichbar, wenn es einen Weg von v nach w gibt.

Bäume

Definition

Ein Baum ist ein gerichteter Graph (V, E) derart, dass ein Knoten w (genannt die Wurzel) existiert, von dem aus zu jedem Knoten des Graphen genau ein Weg existiert.

Definition

- Ein Blatt eines Baumes ist ein Knoten ohne Nachfolger.
- Ein innerer Knoten eines Baumes ist ein Knoten, der kein Blatt ist.
- Ein Ast eines Baumes ist ein Weg, der von der Wurzel ausgeht und der entweder in einem Blatt endet oder unendlich ist.

Teilbäume

Definition

Der durch zu einem Knoten v gehörige Teilbaum oder Unterbaum eines Baumes B ist die Menge aller von v aus erreichbaren Knoten zusammen mit allen Kanten von B zwischen diesen Knoten.

Lemma von König

Definition

Ein Baum heißt endlich verzweigt, wenn jeder Knoten nur endlich viele Nachfolger hat.

Satz (Lemma von König)

Ein endlich verzweigter Baum, in dem jeder Ast endlich ist, ist selbst endlich.

Beweisidee (indirekt):

- Angenommen, er wäre unendlich.
- Dann gäbe es unendliche Teilbäume $B_0, B_1, B_2, \ldots$ mit
- B_0 ist der gegebene Baum.
- B_{n+1} ist unmittelbarer Unterbaum von B_n.
- Die Wurzeln der B_n bilden einen unendlichen Ast. Widerspruch

Unterabschnitt 5

Erzeugungssysteme

Erzeugungssysteme

Definition

- Ein Erzeugungssystem ist ein System von Regeln zur Erzeugung einer Menge / Klasse.
- Dabei sagt jede Regel aus, dass ein Objekt K herleitbar ist, wenn alle Objekte aus einer Menge $\mathbf{P}$ herleitbar sind.

Definition

- Das Paar $(\mathbf{P}, K)$ heißt Instanz der Regel.
- Die Elemente von $\mathbf{P}$ heißen Prämissen der Regelinstanz.
- K heißt Konklusion der Regelinstanz.
- Wenn $\mathbf{P} = \emptyset$ ist, dann heißt K ein Axiom.
- Eine Regel ohne Prämissen heißt Axiomenschema.

Notation

Wenn $\mathbf{P} = \{P_1, \ldots, P_n\}$ *ist, dann schreiben wir auch*

$$\frac{P_1 \quad \ldots \quad P_n}{K}.$$

und statt $\frac{}{K}$ *schreiben wir einfach* K*.*

Beispiel (für Regeln eines Erzeugungssystems zur Herleitung wahrer logischer Aussagen, also eines Logikkalküls)

Axiomenschemata: $A \to A$

$\ldots$

Regeln: $$\frac{A \qquad A \to B}{B}$$

$\ldots$

Herleitung

Definition

Eine Herleitung ist eine endliche oder unendliche Folge $(x_1, x_2, \dots)$ derart, dass für jedes j ein Menge $\mathbf{P} \subseteq \{x_1, \dots, x_{j-1}\}$ existiert, sodass $(\mathbf{P}, x_j)$ eine Instanz einer Regel des Erzeugungssystems ist.

Definition

Ein Herleitungsbaum ist ein Baum, bei dem die Nachfolger eines mit K markierten Knotens mit den Elementen von $\mathbf{P}$ markiert sind für eine Instanz $(\mathbf{P}, K)$ einer Regel des Erzeugungssystems.

Beispiel (Erzeugungssystem auf den natürlichen Zahlen)

Axiome: 3

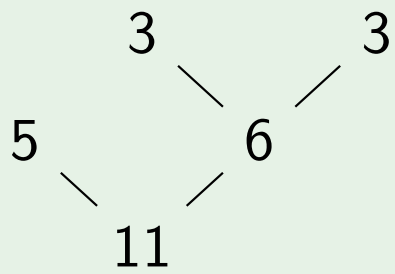
5

Regel: $\dfrac{x \qquad y}{x + y}$

Beispiel (Herleitung)

$$(3, 5, 6, 11)$$

Beispiel (Herleitungsbaum)



Abschnitt 3

Sprachen

Unterabschnitt 1

Wörter

Wörter

Definition

- Ein Alphabet ist eine endliche Menge Σ von Zeichen.
- Ein Wort über Σ ist eine endliche Folge von Zeichen aus Σ.
- Σ^* ist die Menge der Wörter über Σ.

Beispiele

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

- $ababbc$
- bac
- b
- das leere Wort, bezeichnet mit ε

Wörter

Frage

Warum verwendet man Wörter?

- Ausdrucksstärke
  Alles, was im Computer dargestellt werden kann, kann als Wort kodiert werden.
- mathematische Einfachheit

Operationen für Wörter

- Einbettung $\iota : \Sigma \to \Sigma^*$
  $\iota(a)$ auch kurz als a geschrieben
- Konkatenation $\cdot : \Sigma^{*2} \to \Sigma^*$
  $u \cdot v$ auch kurz als uv geschrieben
- Länge $|\cdot| : \Sigma^* \to \mathbb{N}$
  $|w|$
- n-tes Zeichen $\pi : D \to \Sigma$,
  wobei $D := \{(n, w) \mid n \in \mathbb{N} \setminus \{0\} \wedge w \in \Sigma^* \wedge n \leq |w|\}$
- n-te Potenz $\mathrm{pot} : \mathbb{N} \times \Sigma^* \to \Sigma^*$
  $\mathrm{pot}(n, w)$ kurz als w^n geschrieben
- Spiegelung $.^R : \Sigma^* \to \Sigma^*$
  w^R

Unterabschnitt 2

Sprachen

Sprachen

Definition

Eine Sprache über einem Alphabet Σ ist eine Menge von Wörtern über Σ.

Beispiele

für Sprachen über dem Alphabet $\Sigma = \{a, b, c\}$:

- $\emptyset$
- Σ^*
- $\{a, abc, cbaa\}$
- $\{\varepsilon\}$
- $\{a\}$
- $\{a^n \mid n \in \mathbb{N}\} = \{\varepsilon, a, aa, aaa, \dots\}$

Sprachen

Beispiele

- $\Sigma := \{0, 1, +, *, (,)\}$.
  Die Sprache L sei die Menge der arithmetischen Ausdrücke über Σ.
- Σ sei die Menge der ASCII-Zeichen.
  Die Sprache L sei die Menge aller syntaktisch korrekten C-Programme.
- $\Sigma := \{1, x, y, z, +, *, =, (,)\}$.
  Die Sprache L sei die Menge der arithmetischen Gleichungen über Σ, die eine Lösung in $\mathbb{N}$ besitzen.
  - $(1+1)*(1+1+1) = 1+1+1+1+1+1 \in L$
  - $x + x = x * x \in L$
  - $(x+1)*(x+1)*(x+1) + (y+1)*(y+1)*(y+1) = z*z*z \notin L$

Verwendung von Sprachen

- als Mittel, um Informationen auszudrücken:
  Programmiersprachen, ...
- Zur Formulierung von Entscheidungsproblemen:
  Gibt es einen Algorithmus, der entscheiden kann,
  ob ein gegebenes Wort in der Sprache liegt?

Operationen auf Sprachen

- Komplement $\overline{L} = \Sigma^* \setminus L$
- Vereinigung $L_1 \cup L_2$
- Durchschnitt $L_1 \cap L_2$
- Konkatenation $L_1 L_2 = \{uv \mid u \in L_1 \wedge v \in L_2\}$
- Kleene-Stern $L^* = \{w_1 \cdots w_n \mid n \in \mathbb{N} \wedge w_1, \ldots, w_n \in L\}$
- $L^+ = \{w_1 \cdots w_n \mid n \in \mathbb{N} \setminus \{0\} \wedge w_1, \ldots, w_n \in L\}$

Abschnitt 4

Reguläre Ausdrücke und reguläre Sprachen

Reguläre Ausdrücke

Definition

Sei Σ ein Alphabet.
Ein regulärer Ausdruck über Σ ist ein spezielles Wort über dem Alphabet $\Sigma \cup \{O, e, ^*, |, (,)\}$.
Induktive Definition:

- O ist ein regulärer Ausdruck.
- e ist ein regulärer Ausdruck.
- Für jedes Zeichen $a \in \Sigma$ ist das Wort a ein regulärer Ausdruck.
- Wenn A und B reguläre Ausdrücke sind, dann auch $(A|B)$.
- Wenn A und B reguläre Ausdrücke sind, dann auch (AB).
- Wenn A ein regulärer Ausdruck ist, dann auch A^*.

Reguläre Ausdrücke

Beispiele

- a
- (ab)
- $(a|b)$
- (a^*b^*)
- $(ab)^*$
- $(a^*|b^*)$
- $(a|b)^*$

Klammern werden oft weggelassen, wenn dies nicht zu Missverständnissen führt.

Die durch einen regulären Ausdruck bezeichnete Sprache

Definition

Die durch einen regulären Ausdruck A bezeichnete Sprache L_A ist folgendermaßen definiert.

- $L_0 = \emptyset$
- $L_e = \{\varepsilon\}$
- $L_a = \{a\}$ für $a \in \Sigma$
- $L_{(A|B)} = L_A \cup L_B$
- $L_{(AB)} = L_A L_B$
- $L_{A^*} = {L_A}^*$

Definition

Eine durch einen regulären Ausdruck bezeichnete Sprache heißt reguläre Sprache.

Die durch einen regulären Ausdruck bezeichnete Sprache

Beispiele

- $L_a = \{a\}$
- $L_{(ab)} = \{ab\}$
- $L_{(a|b)} = \{a, b\}$
- $L_{(a^*b^*)} = \{\varepsilon, b, bb, \ldots, a, ab, abb, \ldots\}$
- $L_{(ab)^*} = \{\varepsilon, ab, abab, \ldots\}$
- $L_{(a^*|b^*)} = \{\varepsilon, a, aa, \ldots, b, bb, \ldots\}$
- $L_{(a|b)^*} = \{\varepsilon, a, b, aa, ab, ba, bb, \ldots\}$

Reguläre Sprachen

Man kann die regulären Sprachen über Σ auch induktiv charakterisieren:

- Die leere Sprache $\emptyset$ ist regulär.
- Die Sprache $\{\varepsilon\}$ ist regulär.
- Für $a \in \Sigma$ ist die Sprache $\{a\}$ regulär.
- Wenn L und L' regulär sind, dann ist auch $L \cup L'$ regulär.
- Wenn L und L' regulär sind, dann ist auch LL' regulär.
- Wenn L regulär ist, dann ist auch L^* regulär.

Reguläre Sprachen

Wenn L und L' reguläre Sprachen sind, so sind die folgenden Sprachen auch regulär.

- $L \cup L'$
- LL'
- L^*
- $\overline{L}$ (Beweis auf Umweg über endliche Automaten)
- $L \cap L'$ (Beweis auf Umweg über endliche Automaten)
- $L \setminus L'$ (folgt aus den vorigen beiden Aussagen)

Abschnitt 5

Endliche Automaten

Unterabschnitt 1

Deterministische endliche Automaten

Deterministische endliche Automaten

Definition

Ein deterministischer endlicher Automat ist ein Quintupel $(Q, \Sigma, \delta, q_0, F)$, das gegeben ist durch

- eine endliche Menge Q von Zuständen
- ein Alphabet Σ
- eine Übergangsfunktion $\delta\colon Q \times \Sigma \to Q$
- einen Anfangszustand (Startzustand) $q_0 \in Q$
- eine Menge $F \subseteq Q$ von Endzuständen

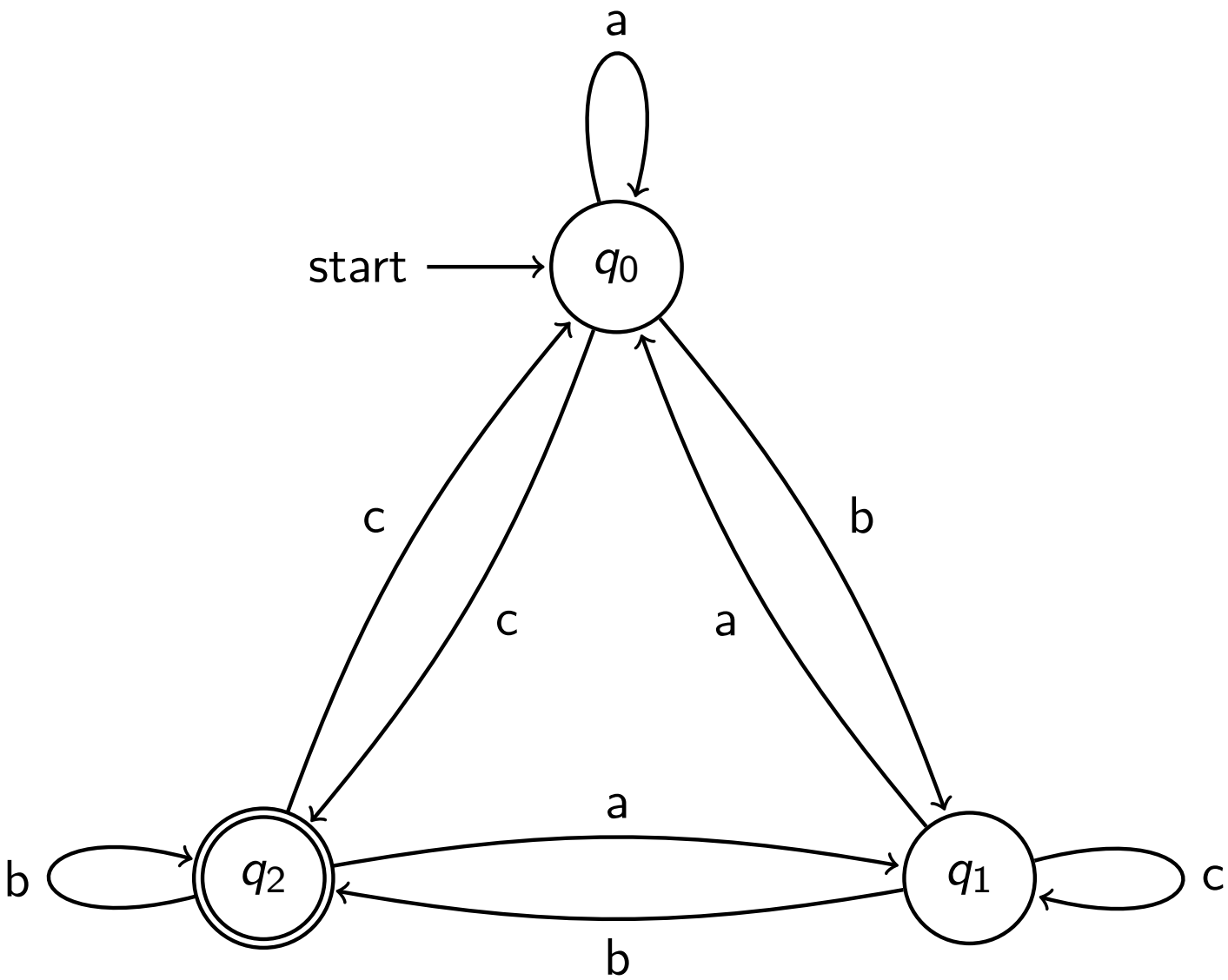
Deterministische endliche Automaten

Beispiel

Ein deterministischer endlicher Automat $(Q, \Sigma, \delta, q_0, F)$

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b, c\}$
- $\delta(q_0, a) = q_0 \quad \delta(q_0, b) = q_1 \quad \delta(q_0, c) = q_2$
  $\delta(q_1, a) = q_0 \quad \delta(q_1, b) = q_2 \quad \delta(q_1, c) = q_1$
  $\delta(q_2, a) = q_1 \quad \delta(q_2, b) = q_2 \quad \delta(q_2, c) = q_0$
- $F = \{q_2\}$

Darstellung eines endlichen Automaten als Graph



Die von einem Automaten akzeptierte Sprache

Definition

Ein deterministischer endlicher Automat über einem Alphabet Σ akzeptiert ein Wort $w = a_1 \dots a_n$ aus Σ^*, wenn es eine endliche Folge $(q_0, \dots, q_n)$ von Zuständen gibt, sodass gilt

- q_0 ist der Anfangszustand des Automaten
- $q_i = \delta(q_{i-1}, a_i)$ für $i = 1, \dots, n$
- $q_n \in F$.

Definition

Die von einem deterministischen endlichen Automaten akzeptierte Sprache ist die Menge der von ihm akzeptierten Wörter.

Konfigurationen

Definition

Eine Konfiguration eines deterministischen endlichen Automaten $(Q, \Sigma, \delta, q_0, F)$ ist ein Paar (q, w), wobei $q \in Q$ und $w \in \Sigma^*$ ist.

Gibt den momentanen Zustand und den noch zu lesenden Teil des Eingabewortes an.

Berechnungen

Definition

Ein deterministischer endlicher Automat $(Q, \Sigma, \delta, q_0, F)$ führt eine Konfiguration (q, w) in eine Konfiguration (q', w') über, wenn es ein Zeichen $a \in \Sigma$ gibt mit

$$w = aw' \wedge q' = \delta(q, a).$$

Berechnungen

Definition

Eine Berechnung durch einen deterministischen endlichen Automaten $(Q, \Sigma, \delta, q_0, F)$ ist eine endliche Folge $(k_0, \ldots, k_n)$ von Konfigurationen derart, dass gilt:

- Der Automat führt die Konfiguration k_{i-1} in die Konfiguration k_i über für $i = 1, \ldots, n$.
- k_0 hat die Form (q_0, w).
- k_n hat die Form (q, ε).

Die Berechnung heißt akzeptierend, wenn $q \in F$ ist.

Von DEA akzeptierte Sprachen

Frage

Für welche Sprachen L gibt es einen deterministischen endlichen Automaten, der L akzeptiert?

Sei Σ ein Alphabet.

Satz

Seien L und L' von deterministischen endlichen Automaten akzeptierte Sprachen. Dann werden auch $\overline{L}$, $L \cap L'$, $L \cup L'$ und $L \setminus L'$ von deterministischen endlichen Automaten akzeptiert.

Wir sagen auch, die Menge der Sprachen L über Σ, zu denen es einen deterministischen endlichen Automaten gibt, der sie akzeptiert, ist abgeschlossen gegenüber Komplementbildung, endlichem Durchschnitt und Vereinigung und Mengendifferenz.

Von DEA akzeptierte Sprachen

Definition

$L[u] := \{w \in \Sigma^* \mid uw \in L\}$ für $u \in \Sigma^*$

Von DEA akzeptierte Sprachen

Für einen deterministischen endlichen Automaten:

Definition

- q_u sei der Zustand, der nach Lesen des Wortes u erreicht ist.
- L_q sei die Menge aller Wörter, die den Automaten vom Zustand q aus in einen Endzustand überführen.

Lemma

Wenn ein deterministischer endlicher Automat die Sprache L akzeptiert, dann gilt für ihn $L[u] = L_{q_u}$ für alle $u \in \Sigma^*$.

Beweis.

$w \in L[u] \iff uw \in L \iff$ Der Automat akzeptiert $uw \iff w \in L_{q_u}$ □

Von DEA akzeptierte Sprachen

$L[u] = L_{q_u}$. Also $L[u] \in \{L_q \mid q \in Q\}$.

Satz

- *Es gibt genau dann einen deterministischen endlichen Automaten, der L akzeptiert, wenn es nur endlich viele verschiedene Sprachen $L[u]$ gibt.*
- *In diesem Fall lässt sich ein solcher deterministischer endlicher Automat mit einer minimalen Anzahl von Zuständen folgendermaßen konstruieren.*
  - *Zu jedem $L[u]$ gibt es einen Zustand q_u.*
  - *Ein Zeichen a führt vom Zustand q_u in den Zustand q_{ua}.*
  - *Anfangszustand ist q_ε.*
  - *Endzustände sind alle q_u mit $\varepsilon \in L[u]$.*
- *Er hat so viele Zustände, wie es Sprachen $L[u]$ gibt.*

Satz von Myhill-Nerode

Definition

Nerode-Relation $\sim_L$ einer Sprache $L \subseteq \Sigma^*$:

$$u \sim_L v : \Longleftrightarrow \bigwedge_{w \in \Sigma^*} (uw \in L \iff vw \in L)$$

$$u \sim_L v \iff L[u] = L[v]$$

Satz (Myhill, Nerode)

- *Zu einer Sprache L existiert genau dann ein DEA, der sie akzeptiert, wenn der Index $|\Sigma^* / \sim_L|$ ihrer Nerode-Relation endlich ist.*
- *In diesem Fall ist die Anzahl der Zustände eines minimalen deterministischen endlichen Automaten, der L akzeptiert, gleich diesem Index.*

Das Pumping Lemma für DEA

Satz (Pumping Lemma)

Sei L eine von einem deterministischen endlichen Automaten akzeptierte Sprache. Dann existiert ein $n \in \mathbb{N}$ so, dass sich jedes Wort $w \in L$ mit $|w| \geq n$ als Konkatenation $w = xyz$ schreiben lässt mit

1. $y \neq \epsilon$
2. $|xy| \leq n$
3. *Für alle $k \geq 0$ ist $xy^k z \in L$.*

Beweis.

- Sei n die Anzahl der Zustände des Automaten.
- $w = a_1 \dots a_m$ mit $m \geq n$.
- Für $i = 0, \dots, m$ sei q_i der Zustand nach Lesen von $a_1 \dots a_i$.
- Nach dem Schubfachprinzip gibt es i und j mit $i < j \leq n$ und $q_i = q_j$.
- $x := a_1 \dots a_i, \quad y := a_{i+1} \dots a_j, \quad z := a_{j+1} \dots a_m.$ □

Beispiele für Sprachen die nicht von einem DEA akzeptiert werden

- $\{a^n b^n \mid n \in \mathbb{N}\}$
  Alle $L[a^n]$ sind verschieden. $a^n b^n$ kann nicht gepumpt werden.
- Menge der wohlgeformten Klammerausdrücke
  Alle $L[(^n]$ sind verschieden. $(^n)^n$ kann nicht gepumpt werden.

Unterabschnitt 2

Nichtdeterministische endliche Automaten

Nichtdeterministische endliche Automaten

Definition

Ein nichtdeterministischer endlicher Automat ist ein Quintupel $(Q, \Sigma, \Delta, q_0, F)$, das gegeben ist durch

- eine endliche Menge Q von Zuständen
- ein Alphabet Σ
- eine Übergangsrelation $\Delta \subseteq Q \times \Sigma \times Q$
- einen Anfangszustand (Startzustand) $q_0 \in Q$
- eine Menge $F \subseteq Q$ von Endzuständen

Die vom Automaten akzeptierte Sprache

Definition

Ein nichtdeterministischer endlicher Automat über einem Alphabet Σ akzeptiert ein Wort $w = a_1 \dots a_n$ aus Σ^*, wenn es eine endliche Folge $(q_0, \dots, q_n)$ von Zuständen gibt, sodass gilt

- q_0 ist der Anfangszustand des Automaten
- $(q_{i-1}, a_i, q_i) \in \Delta$ für $i = 1, \dots, n$
- $q_n \in F$.

Definition

Die von einem nichtdeterministischen endlichen Automaten akzeptierte Sprache ist die Menge der von ihm akzeptierten Wörter.

Konfigurationen

Definition

Eine Konfiguration eines nichtdeterministischen endlichen Automaten $(Q, \Sigma, \Delta, q_0, F)$ ist ein Paar (q, w), wobei $q \in Q$ und $w \in \Sigma^*$ ist.

Gibt den momentanen Zustand und den noch zu lesenden Teil des Eingabewortes an.

Berechnungen

Definition

Ein nichtdeterministischer endlicher Automat $(Q, \Sigma, \Delta, q_0, F)$ führt eine Konfiguration (q, w) in eine Konfiguration (q', w') über, wenn es ein Zeichen $a \in \Sigma$ gibt mit

$$w = aw' \wedge (q, a, q') \in \Delta.$$

Berechnungen

Definition

Eine Berechnung durch einen nichtdeterministischen endlichen Automaten $(Q, \Sigma, \Delta, q_0, F)$ ist eine endliche Folge $(k_0, \ldots, k_n)$ von Konfigurationen derart, dass gilt:

- Der Automat führt die Konfiguration k_{i-1} in die Konfiguration k_i über für $i = 1, \ldots, n$.
- k_0 hat die Form (q_0, w).
- k_n hat die Form (q, ε).

Die Berechnung heißt akzeptierend, wenn $q \in F$ ist.

Simulation eines nichtdeterministischen endlichen Automaten durch einen deterministischen

Gegeben ein nichtdeterministischer endlicher Automat $(Q, \Sigma, \Delta, q_0, F)$.

Konstruktion eines deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert

Automat $(Q', \Sigma, \delta', q_0', F')$:

- $Q' = \mathcal{P}(Q)$
- $\delta'(p', a) = \{q \mid \bigvee_{p \in p'} (p, a, q) \in \Delta\}$
- $q_0' = \{q_0\}$
- $F' = \{R \subseteq Q \mid \bigvee_{r \in R} r \in F\}$

Unterabschnitt 3

Automaten mit Wort-Übergängen

Nichtdeterministische endliche Automaten mit Wortübergängen

Definition

Ein nichtdeterministischer endlicher Automat mit Wortübergängen ist ein Quintupel $(Q, \Sigma, \Delta, q_0, F)$, das gegeben ist durch

- eine endliche Menge Q von Zuständen
- ein Alphabet Σ
- eine Übergangsrelation $\Delta \subseteq Q \times \Sigma^* \times Q$
- einen Anfangszustand (Startzustand) $q_0 \in Q$
- eine Menge $F \subseteq Q$ von Endzuständen

Die vom Automaten akzeptierte Sprache

Definition

Ein nichtdeterministischer endlicher Automat mit Wortübergängen über einem Alphabet Σ akzeptiert ein Wort $w \in \Sigma^*$, wenn es Wörter $w_1, \ldots, w_n \in \Sigma^*$ und Zustände $q_0, \ldots, q_n$ gibt, sodass gilt

- q_0 ist der Anfangszustand des Automaten
- $(q_{i-1}, w_i, q_i) \in \Delta$ für $i = 1, \ldots, n$
- $q_n \in F$
- $w = w_1 \ldots w_n$.

Definition

Die von einem nichtdeterministischen endlichen Automaten mit Wortübergängen akzeptierte Sprache ist die Menge der von ihm akzeptierten Wörter.

Konfigurationen

Definition

Eine Konfiguration eines nichtdeterministischen endlichen Automaten mit Wortübergängen $(Q, \Sigma, \Delta, q_0, F)$ ist ein Paar (q, w), wobei $q \in Q$ und $w \in \Sigma^*$ ist.

Gibt den momentanen Zustand und den noch zu lesenden Teil des Eingabewortes an.

Berechnungen

Definition

Ein nichtdeterministischer endlicher Automat mit Wortübergängen $(Q, \Sigma, \Delta, q_0, F)$ führt eine Konfiguration (q, w) in eine Konfiguration (q', w') über, wenn es ein Wort $v \in \Sigma^*$ gibt mit

$$w = vw' \wedge (q, v, q') \in \Delta.$$

Berechnungen

Definition

Eine Berechnung durch einen nichtdeterministischen endlichen Automaten mit Wortübergängen $(Q, \Sigma, \Delta, q_0, F)$ ist eine endliche Folge $(k_0, \ldots, k_n)$ von Konfigurationen derart, dass gilt:

- Der Automat führt die Konfiguration k_{i-1} in die Konfiguration k_i über für $i = 1, \ldots, n$.
- k_0 hat die Form (q_0, w).
- k_n hat die Form (q, ε).

Die Berechnung heißt akzeptierend, wenn $q \in F$ ist.

Automaten mit epsilon-Übergängen

Definition

Ein nichtdeterministischer endlicher Automat mit Wortübergängen $(Q, \Sigma, \Delta, q_0, F)$ heißt nichtdeterministischer endlicher Automat mit ε-Übergängen, wenn für alle $(q, v, q') \in \Delta$ gilt: $|v| \leq 1$.

Unterabschnitt 4

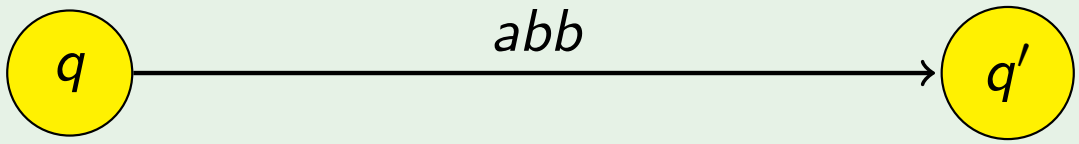
Simulation eines Automaten mit Wortübergängen durch einen deterministischen Automaten

Simulation von Wortübergängen durch Zeichenübergänge

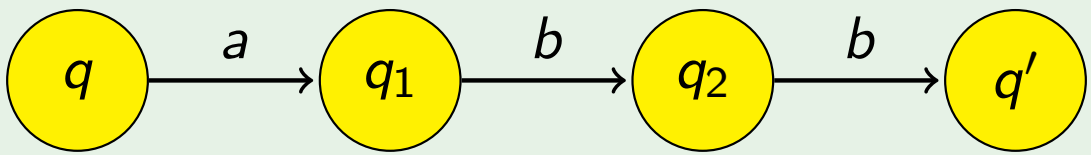
Ein Übergang $(q, w, q') \in \Delta$ mit $|w| > 1$ kann unter Verwendung von Zwischenzuständen durch mehrere Zeichenübergänge ersetzt werden.

Beispiel

$(q, abb, q') \in \Delta$



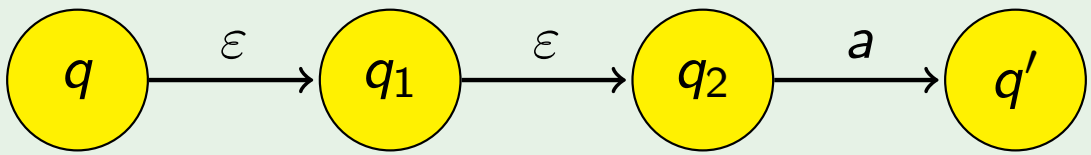
wird ersetzt durch



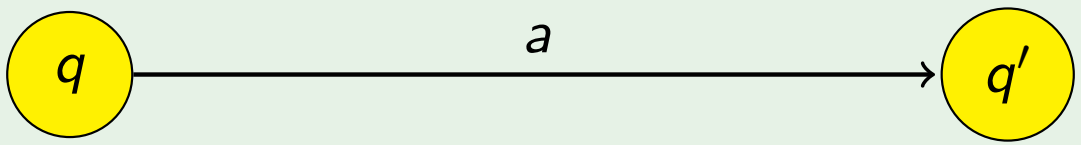
Elimination von epsilon-Übergängen

Epsilon-Übergänge können eliminiert werden, indem jede Folge von ε-Übergängen gefolgt von einem Zeichenübergang ersetzt wird durch einen Zeichenübergang.

Beispiel



wird ersetzt durch



Die Menge der Endzustände muss dann noch angepasst werden.

Konstruktion eines deterministischen Automaten

Der letzte Schritt ist die Simulation des so erhaltenen nichtdeterministischen endlichen Automaten durch einen deterministischen, die wir bereits kennen.

Abschnitt 6

Äquivalenz von regulären Ausdrücken und endlichen Automaten

Unterabschnitt 1

Jede von einem endlichen Automaten akzeptierten Sprache ist regulär

Jede von einem endlichen Automaten akzeptierten Sprache ist regulär

Satz

- *Sei L die von einem deterministischen oder nichtdeterministischen endlichen Automaten akzeptierte Sprache.*
- *Dann ist L regulär.*

Beweis

- Für $p, r \in Q$ und $J \subseteq Q$ sei $L[p, J, r]$ die Menge der Wörter, die im endlichen Automaten von p aus durch die Menge J nach r führen.
- Dann ist

$$L = \bigcup_{q \in F} L[q_0, Q, q]$$

Jede von einem endlichen Automaten akzeptierten Sprache ist regulär

Beweis (Fortsetzung)

- $L[p, \emptyset, r]$ ist endlich und daher regulär.
- Es gilt

$$L[p, J \cup \{q\}, r] = L[p, J, r] \cup L[p, J, q] L[q, J, q]^* L[q, J, r].$$

- Durch vollständige Induktion nach der Anzahl der Elemente von J folgt, dass $L[p, J, r]$ regulär ist.
- Also ist die Sprache L regulär. □

Unterabschnitt 2

Jede reguläre Sprachen wird von einem endlichen Automaten akzeptiert

Von endlichen Automaten akzeptierte Sprachen

Definition

- Sei Σ ein Alphabet.
- Dann bezeichnen wir mit $\mathbb{L}_{\mathrm{endA}}(\Sigma)$ die Menge der von endlichen Automaten akzeptierten Sprachen über Σ.

Von endlichen Automaten akzeptierte Sprachen

Lemma

Sei Σ ein Alphabet. Dann gilt:

- $\emptyset \in \mathbb{L}_{\mathrm{endA}}(\Sigma)$.
- $\{\varepsilon\} \in \mathbb{L}_{\mathrm{endA}}(\Sigma)$.
- $\{\iota(a)\} \in \mathbb{L}_{\mathrm{endA}}(\Sigma)$ *für alle* $a \in \Sigma$.
- *Wenn* $L, L' \in \mathbb{L}_{\mathrm{endA}}(\Sigma)$, *dann* $L \cup L' \in \mathbb{L}_{\mathrm{endA}}(\Sigma)$.
- *Wenn* $L, L' \in \mathbb{L}_{\mathrm{endA}}(\Sigma)$, *dann* $LL' \in \mathbb{L}_{\mathrm{endA}}(\Sigma)$.
- *Wenn* $L \in \mathbb{L}_{\mathrm{endA}}(\Sigma)$, *dann* $L^* \in \mathbb{L}_{\mathrm{endA}}(\Sigma)$.
- *Wenn* $L \in \mathbb{L}_{\mathrm{endA}}(\Sigma)$, *dann* $\overline{L} \in \mathbb{L}_{\mathrm{endA}}(\Sigma)$.

Beweisidee

Setze die endlichen Automaten für L und L' mithilfe von ε-Übergängen geeignet zusammen!

Jede reguläre Sprachen wird von einem endlichen Automaten akzeptiert

Folgerung (1)

$\mathbb{L}_{\mathrm{endA}}(\Sigma)$ *ist abgeschlossen nicht nur gegenüber Konkatenation, Mengenvereinigung, Kleene-Stern-Operation und Komplementbildung, sondern auch gegenüber Bildung von Mengendurchschnitt und Mengendifferenz.*

Folgerung (2)

Jede reguläre Sprache über Σ *ist in* $\mathbb{L}_{\mathrm{endA}}(\Sigma)$.

Reguläre Sprachen und endliche Automaten

Satz

Sei L eine Sprache. Dann sind die folgenden Aussagen zueinander äquivalent:

- *L ist regulär.*
- *L wird von einem deterministischen endlichen Automaten akzeptiert.*
- *L wird von einem nichtdeterministischen endlichen Automaten akzeptiert.*
- *L wird von einem nichtdeterministischen endlichen Automaten mit ε-Übergängen akzeptiert.*
- *L wird von einem nichtdeterministischen endlichen Automaten mit Wort-Übergängen akzeptiert.*

Abschnitt 7

Primitivrekursive Funktionen

Unterabschnitt 1

Zahlentheorie

Zahlentheorie

Die Zahlentheorie

Theorie der natürlichen Zahlen $0, 1, 2, 3, \ldots$

Sie gehört zu den schwierigsten Gebieten der Mathematik.

Beispiel (Fermat'sche Vermutung)

Es gibt keine vier natürlichen Zahlen $a, b, c > 0$ und $n > 2$ mit $a^n + b^n = c^n$.

Beispiel (Goldbach'sche Vermutung)

Jede gerade Zahl > 2 ist Summe zweier Primzahlen.

Zahlentheorie

Beispiel (Zehntes Hilbert'sches Problem)

Lösung diophantischer Gleichungen:

- Gegeben zwei Polynome $P[x_1, \ldots, x_n]$ und $Q[x_1, \ldots, x_n]$ mit Koeffizienten aus $\mathbb{N}$ über Variablen $x_1, \ldots, x_n$.
- Frage: Hat die Gleichung

$$P[x_1, \ldots, x_n] = Q[x_1, \ldots, x_n]$$

eine Lösung für $x_1, \ldots, x_n$ in $\mathbb{N}$?

Es hat zwanzig Jahre gedauert, bis man zeigen konnte, dass dieses Problem unentscheidbar ist.

Warum natürliche Zahlen in der Informatik?

- Jeder Datensatz, der im Computer dargestellt werden kann, lässt sich als natürliche Zahl codieren.
- Natürliche Zahlen gehören zu den einfachsten Objekten.
- Rekursionsprozesse lassen sich deshalb an natürlichen Zahlen am einfachsten studieren.
- Durch einfache Rekursionen auf natürlichen Zahlen lassen sich die komplexesten Algorithmen beschreiben.
- Daher sind natürliche Zahlen und Rekursion auf natürlichen Zahlen geeignet, auf vergleichsweise einfache Weise Einsichten in den Begriff des Algorithmus zu gewinnen.

Die Peano-Axiome

Peano-Axiome

1. 0 ist eine natürliche Zahl.
2. Jeder natürlichen Zahl n ist eine und nur eine natürliche Zahl n' zugeordnet, die der Nachfolger von n genannt wird.
3. 0 ist kein Nachfolger.
4. Für natürliche Zahlen m und n gilt:

$$m \neq n \Rightarrow m' \neq n'.$$

5. Enthält eine Menge P von natürlichen Zahlen die Zahl 0 und folgt aus $n \in P$ stets $n' \in P$, so besteht P aus allen natürlichen Zahlen. (Vollständige Induktion)

Die Peano-Axiome

5' Gilt $P(0)$ und folgt aus $P(n)$ stets $P(n')$, so gilt $P(n)$ für alle natürlichen Zahlen n.

Aufbau der natürlichen Zahlen

Grundoperationen

- 0 Null
- N Nachfolgerfunktion. $N(x) = x' = x + 1.$

$$3 = 0''' = N(N(N(0)))$$

Unterabschnitt 2

Primitivrekursive Funktionen

Zahlentheoretische Funktionen

Definition

- Eine *n*-stellige zahlentheoretische Funktion ist eine Funktion $f : \mathbb{N}^n \to \mathbb{N}$.
- Die natürliche Zahl n heißt die Stelligkeit von f.

Zahlentheoretische Funktionen

Beispiele (für einstellige zahlentheoretische Funktionen)

- Nachfolgerfunktion $N = x \mapsto x + 1$
- Verdoppelungsfunktion $x \mapsto 2x \colon \mathbb{N} \to \mathbb{N}$
- Quadratfunktion $x \mapsto x^2 \colon \mathbb{N} \to \mathbb{N}$
- Fakultätsfunktion $\mathrm{fak} = x \mapsto x! \colon \mathbb{N} \to \mathbb{N}$
- Vorgängerfunktion $V \colon \mathbb{N} \to \mathbb{N}$ mit

$$V(x) = \begin{cases} x - 1, & \text{wenn } x > 0 \\ 0, & \text{wenn } x = 0. \end{cases}$$

Zahlentheoretische Funktionen

Beispiele (für zweistellige zahlentheoretische Funktionen)

- $\mathrm{add}(a, b) = a + b$
- $\mathrm{mul}(a, b) = ab$
- $\mathrm{pot}(a, b) = a^b$
- $\mathrm{diff}(a, b) = a \dot{-} b = \begin{cases} a - b, & \text{wenn } a \geq b \\ 0 & \text{sonst.} \end{cases}$ modifizierte Differenz

Beispiele (für n-stellige zahlentheoretische Funktionen)

- $K_c^n(x_1, \ldots, x_n) = c$ Konstantenfunktion
- $P_i^n(x_1, \ldots, x_n) = x_i$ Projektionsfunktion

Komposition

Definition

Sei f eine m-stellige zahlentheoretische Funktion ($m > 0$).
Seien $g_1, \ldots, g_m$ n-stellige zahlentheoretische Funktionen.
Dann heißt die durch

$$h(x_1, \ldots, x_n) := f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$$

definierte n-stellige zahlentheoretische Funktion h die Komposition $(fg_1 \ldots g_m)$ von f mit $g_1, \ldots, g_m$.

Primitive Rekusion

Definition

Sei f eine n-stellige zahlentheoretische Funktion.
Sei g eine $(n+2)$-stellige zahlentheoretische Funktion.
Dann heißt die durch

$$h(x_1, \ldots, x_n, 0) := f(x_1, \ldots, x_n)$$
$$h(x_1, \ldots, x_n, y') := g(h(x_1, \ldots, x_n, y), x_1, \ldots, x_n, y)$$

definierte $(n+1)$-stellige zahlentheoretische Funktion h die durch primitive Rekursion aus f und g gebildete Funktion (Rfg).

Primitivrekursive Funktionen

Definition (induktive Definition des Begriffs der primitivrekursiven Funktionen)

- O ist eine primitivrekursive Funktion.
- N ist eine primitivrekursive Funktion.
- P_i^n ist eine primitivrekursive Funktion für $1 \leq i \leq n$.
- Wenn f eine m-stellige primitivrekursive Funktion ist ($m > 0$) und $g_1, \ldots, g_m$ n-stellige primitivrekursive Funktionen sind, dann ist $(fg_1 \ldots g_m)$ eine primitivrekursive Funktion.
- Wenn f eine n-stellige primitivrekursive Funktion ist und g eine $(n+2)$-stellige primitivrekursive Funktion ist, dann ist (Rfg) eine primitivrekursive Funktion.

Primitivrekursive Funktionen

Beispiele

- Additionsfunktion add, Multiplikationsfunktion mul, Potenzierungsfunktion pot, Vorgängerfunktion V, Konstantenfunktionen K_c^n sind primitivrekursiv (siehe Übungsaufgaben).
-

$$\mathrm{diff} = (RP_1^1(VP_1^3))$$
$$\mathrm{sg} = (ROK_1^2)$$
$$\overline{\mathrm{sg}} = (\mathrm{diff}\, K_1^1 P_1^1)$$

Summe und Produkt

Definition

Für eine $(n+1)$-stellige Funktion f sei

$$(\Sigma f) := (Rg(+P_1^{n+2}h))$$
$$(\Pi f) := (Rg(*P_1^{n+2}h)),$$

wobei

$$g := (fP_1^n \dots P_n^n K_0^n)$$
$$h := (fP_2^{n+2} \dots P_{n+1}^{n+2}(NP_{n+2}^{n+2}))).$$

Summe und Produkt

Dann gilt:

$$(\Sigma f)(x_1, \ldots, x_n, y) = \sum_{z=0}^{y} f(x_1, \ldots, x_n, z)$$

$$(\Pi f)(x_1, \ldots, x_n, y) = \prod_{z=0}^{y} f(x_1, \ldots, x_n, z)$$

Primitivrekursive Terme

Definition (primitivrekursive Terme über Variablen $x_1, \ldots, x_n$, induktive Definition)

- Jedes x_i mit $i = 1, \ldots, n$ ist ein primitivrekursiver Term.
- Die Dezimaldarstellungen natürlicher Zahlen sind primitivrekursive Terme.
- Wenn f ein Symbol für eine k-stellige primitivrekursive Funktion ist und $t_1, \ldots, t_k$ primitivrekursive Terme sind, dann ist das Wort $f(t_1, \ldots, t_k)$ ein primitivrekursiver Term.

Komprehension mittels primitivrekursiver Terme

Satz

Eine n-stellige zahlentheoretische Funktion h sei folgendermaßen definiert.

$$h(x_1, \ldots, x_n) := t.$$

Dabei seien $x_1, \ldots, x_n$ paarweise verschiedene Variablen und t sei ein primitivrekursiver Term über den Variablen $x_1, \ldots, x_n$.
Dann ist h primitivrekursiv.

Beispiel

Seien f und g zwei einstellige primitivrekursive Funktionen und

$$k(x) := f(x) \cdot \overline{\mathrm{sg}}(x) + g(x) \cdot \mathrm{sg}(x).$$

Dann ist auch k primitivrekursiv. (Fallunterscheidung)

Methoden zum Nachweis, dass eine Funktion h primitivrekursiv ist

Methode

Wenn h durch Induktion nach ihrem letzten Argument definiert ist, dann Ansatz: $h = (Rfg)$

Methoden zum Nachweis, dass eine Funktion h primitivrekursiv ist

Beispiel

$$\begin{aligned} \mathrm{diff}(x, 0) &= x \\ \mathrm{diff}(x, y') &= V(\mathrm{diff}(x, y)) \end{aligned}$$

Ansatz: $\mathrm{diff} = (Rfg)$. Dann wähle

$$\begin{aligned} f(x) &:= x \\ g(u, x, y) &:= V(u) \end{aligned}$$

Methoden zum Nachweis, dass eine Funktion h primitivrekursiv ist

Methode

Wenn h durch Induktion nach einem anderen Argument definiert ist, dann vertausche die Argumente.

Beispiel

$h(x, y)$ durch Induktion nach x definiert.
Sei $k(y, x) := h(x, y)$.
Ansatz: $k = (Rfg)$
$h = (kP_2^2 P_1^2)$

Methoden zum Nachweis, dass eine Funktion h primitivrekursiv ist

Methode

Wenn h sich aus einfacheren Funktionen zusammensetzen lässt, dann verwende den Satz über Komprehension.

Beispiel

$$g(u, x, y) = V(u)$$

Dann ist $g = (VP_1^3)$.

Unterabschnitt 3

Primitivrekursive Funktionale

Primitivrekursive Funktionale

Die formalen Ausdrücke für primitivrekursive Funktionen nennt man primitivrekursive Funktionale

Primitivrekursive Funktionale sind spezielle Wörter über der Zeichenmenge

$$\{\mathtt{O}, \mathtt{N}, (,), \mathtt{R}\} \cup \{\mathtt{P}_i^n \mid 1 \leq i \leq n\}$$

Primitivrekursive Funktionale

Definition

- $\mathtt{O}$ ist ein 0-stelliges primitivrekursives Funktional.
- $\mathtt{N}$ ist ein 1-stelliges primitivrekursives Funktional.
- $\mathtt{P}_i^n$ ist ein n-stelliges primitivrekursives Funktional für $1 \leq i \leq n$.
- Wenn ϕ ein m-stelliges primitivrekursives Funktional ist $(m > 0)$ und $\psi_1, \ldots, \psi_m$ n-stellige primitivrekusive Funktionale sind, dann ist $(\phi\psi_1 \ldots \psi_m)$ ein n-stelliges primitivrekursives Funktional.
- Wenn ϕ ein n-stelliges primitivrekursives Funktional ist und ψ ein $(n+2)$-stelliges primitivrekursives Funktional ist, dann ist $(\mathtt{R}\phi\psi)$ ein $(n+1)$-stelliges primitivrekusives Funktional.

Unterabschnitt 4

Nicht alle zahlentheoretischen Funktionen sind primitivrekursiv

Frage

Ist jede zahlentheoretische Funktion primitivrekursiv?

Antwort

Nein, denn es gibt nur abzählbar viele primitivrekursive Funktionen, aber überabzählbar viele zahlentheoretische Funktionen.

Abzählbare Mengen

Satz

Sei Σ ein Alphabet. Dann ist Σ^* abzählbar.

Beweisidee

- Das Alphabet habe n Zeichen.
- Zu jedem $k \in \mathbb{N}$ gibt es n^k Wörter der Länge k: $w_{k1}, \ldots, w_{kn^k}$.
- Abzählung: $w_{01}, w_{11}, \ldots, w_{1n}, w_{21}, \ldots, w_{2n^2}, w_{31}, \ldots, w_{3n^3}, \ldots$

Abzählbare Mengen

Satz

Sei Σ eine abzählbare Menge von Zeichen. Dann ist die Menge Σ^* der Wörter über Σ abzählbar.

Beweisidee

- Sei $\Sigma = \{z_0, z_1, z_2, \dots\}$.
- Kodiere ein Zeichen $z_n \in \Sigma$ als ein z mit n Strichen: $z'^{\cdots'}$.
- Hierdurch wird ein Wort aus Σ^* kodiert durch ein Wort aus $\{z,'\}^*$.
- $\{z,'\}^*$ ist abzählbar nach dem vorigen Satz.

Abzählbare Mengen

Folgerung

Die Menge der primitivrekursiven Funktionen ist abzählbar.

Beweis.

- Die Zeichenmenge $\{\mathtt{O}, \mathtt{N}, (,), \mathtt{R}\} \cup \{\mathtt{P}_i^n \mid 1 \leq i \leq n\}$ ist abzählbar.
- Daher ist $(\{\mathtt{O}, \mathtt{N}, (,), \mathtt{R}\} \cup \{\mathtt{P}_i^n \mid 1 \leq i \leq n\})^*$ abzählbar.
- Die Menge der primitivrekursiven Funktionale ist eine Teilmenge hiervon und daher abzählbar.

Das Cantor'sche Diagonalverfahren

Satz

Die Menge $\mathbb{N}^{\mathbb{N}}$ der Funktionen $f : \mathbb{N} \to \mathbb{N}$ ist nicht abzählbar.

Beweis mit dem Cantor'schen Diagonalverfahren, indirekt

- Sonst gäbe es eine Abzählung $f_0, f_1, f_2, \ldots$ von $\mathbb{N}^{\mathbb{N}}$.
- $$\begin{array}{cccc} f_0(0) & f_0(1) & f_0(2) & \\ f_1(0) & f_1(1) & f_1(2) & \ldots \\ f_2(0) & f_2(1) & f_2(2) & \\ & \vdots & & \ddots \end{array}$$
- Veränderte Diagonale $f(k) := f_k(k) + 1$
- f ist keines der f_n, da sonst $f(n) = f_n(n) + 1 = f(n) + 1$.
- Widerspruch dazu, dass $f_0, f_1, f_2, \ldots$ eine Abzählung von $\mathbb{N}^{\mathbb{N}}$ ist. □

Folgerungen

Folgerung

Sei $n \geq 1$. Dann sind die Menge der n-stelligen zahlentheoretischen Funktionen und die Menge der n-stelligen Funktionen über der Menge der Wörter über einem Alphabet nicht abzählbar.

Folgerung

Zu jeder Stelligkeit außer 0 gibt es zahlentheoretische Funktionen, die nicht primitivrekursiv sind.

Unterabschnitt 5

Konstruktion neuer primitivrekursiver Funktionen

Zahlentheoretische Prädikate

Definition

Unter einem *n-stelligen zahlentheoretischen Prädikat* versteht man ein n-stelliges Prädikat auf der Menge $\mathbb{N}$ der natürlichen Zahlen.

Definition

Für ein n-stelliges zahlentheoretisches Prädikat P wird die charakteristische Funktion χ_P von P definiert durch

$$\chi_P(x_1, \ldots, x_n) := \begin{cases} 1, & \text{wenn } P(x_1, \ldots, x_n) \text{ gilt} \\ 0 & \text{sonst} \end{cases}$$

für alle $x_1, \ldots, x_n \in \mathbb{N}$.

Zahlentheoretische Prädikate

Definition

Ein zahlentheoretisches Prädikat heißt primitivrekursiv, wenn seine charakteristische Funktion primitivrekursiv ist.

Definition

Sei n eine natürliche Zahl und seien P und Q zwei n-stellige zahlentheoretische Prädikate. Dann werden die n-stelligen zahlentheoretischen Prädikate $\neg P$, $P \wedge Q$ und $P \vee Q$ folgendermaßen definiert.

$$
\begin{aligned}
(\neg P)(x_1, \ldots, x_n) &:\Longleftrightarrow \neg P(x_1, \ldots, x_n) \\
(P \wedge Q)(x_1, \ldots, x_n) &:\Longleftrightarrow P(x_1, \ldots, x_n) \wedge Q(x_1, \ldots, x_n) \\
(P \vee Q)(x_1, \ldots, x_n) &:\Longleftrightarrow P(x_1, \ldots, x_n) \vee Q(x_1, \ldots, x_n)
\end{aligned}
$$

Primitivrekursive zahlentheoretische Prädikate

Lemma

Ein n-stelliges zahlentheoretisches Prädikat P ist genau dann primitivrekursiv, wenn es eine n-stellige primitivrekursive Funktion f gibt mit

$$\bigwedge_{x_1,\ldots,x_n \in \mathbb{N}} \big(P(x_1,\ldots,x_n) \iff f(x_1,\ldots,x_n) = 0\big).$$

Lemma (Aufgabe)

Sei n eine natürliche Zahl und seien P und Q zwei n-stellige primitivrekursive Prädikate. Dann sind die Prädikate $\neg P$, $P \wedge Q$ und $P \vee Q$ auch primitivrekursiv.

Komprehension

Satz

Ein zahlentheoretisches Prädikat P, das mittels Komprehension

$$P(x_1, \ldots, x_n) :\Longleftrightarrow A$$

definiert ist, wobei der Ausdruck A aufgebaut ist aus

- *Variablen $x_1, \ldots, x_n$*
- *Dezimaldarstellungen natürlicher Zahlen*
- *primitivrekursiven Funktionen und Prädikaten*

durch

- *Funktions- und Prädikatsanwendung*
- *Negation $\neg$, Konjunktion $\wedge$ und Disjunktion $\vee$,*

ist primitivrekursiv.

Komprehension

Beispiel

- Das Prädikat $\leq$ ist primitivrekursiv, da für alle $x, y \in \mathbb{N}$
  $x \leq y \iff \mathrm{diff}(x, y) = 0$ gilt
  und da diff primitivrekursiv ist.
- $x \neq y \iff \neg(x \leq y \wedge y \leq x)$.
- Daher ist auch $\neq$ primitivrekursiv.

Beschränkte Quantifizierung

Definition

Sei P ein $(n+1)$-stelliges zahlentheoretisches Prädikat. Dann werden die $(n+1)$-stelligen zahlentheoretischen Prädikate

- $(\forall_b P)$ (beschränkte Allquantifizierung von P)
- $(\exists_b P)$ (beschränkte Existenzquantifizierung von P)

folgendermaßen definiert:

$$(\forall_b P)(x_1, \ldots, x_n, y) :\Longleftrightarrow \bigwedge_{z \leq y} P(x_1, \ldots, x_n, z)$$

$$(\exists_b P)(x_1, \ldots, x_n, y) :\Longleftrightarrow \bigvee_{z \leq y} P(x_1, \ldots, x_n, z)$$

Beschränkte Quantifizierung

Lemma (Aufgabe)

Sei P ein $(n+1)$-stelliges zahlentheoretisches Prädikat. Dann gilt

$$\chi_{(\forall_b P)} = (\Pi \chi_P)$$
$$\chi_{(\exists_b P)} = (\mathrm{sg}(\Sigma \chi_P))$$

Folgerung (Aufgabe)

Wenn P ein $(n+1)$-stelliges primitivrekusives zahlentheoretisches Prädikat ist, dann sind auch die Prädikate $(\forall_b P)$ und $(\exists_b P)$ primitivrekursiv.

Beschränkte Minimalisierung

Definition

Für ein $(n+1)$-stelliges zahlentheoretisches Prädikat P ist die $(n+1)$-stellige zahlentheoretische Funktion $(\mu_b P)$ definiert durch

$$(\mu_b P)(x_1, \ldots, x_n, y) := \begin{cases} \min\{z \leq y \mid P(x_1, \ldots, x_n, z)\}, & \text{falls so ein } z \text{ existiert} \\ 0 & \text{sonst.} \end{cases}$$

Die Funktion $(\mu_b P)$ heißt die beschränkte Minimalisierung von P oder die durch Anwendung des beschränkten μ-Operators μ_b aus P entstandene Funktion.

Beschränkte Minimalisierung

Definition

Für eine $(n+1)$-stellige zahlentheoretische Funktion f ist die $(n+1)$-stellige zahlentheoretische Funktion $(\mu_b f)$ definiert durch

$$(\mu_b f)(x_1, \ldots, x_n, y) := \begin{cases} \min\{z \leq y \mid f(x_1, \ldots, x_n, z) = 0\}, & \text{falls so ein } z \text{ existiert} \\ 0 & \text{sonst.} \end{cases}$$

Die Funktion $(\mu_b f)$ heißt die beschränkte Minimalisierung von f oder die durch Anwendung des beschränkten μ-Operators μ_b aus f entstandene Funktion.

Beschränkte Minimalisierung

Satz

Wenn P ein $(n+1)$-stelliges primitivrekursives Prädikat ist, dann ist $(\mu_b P)$ eine $(n+1)$-stellige primitivrekursive Funktion.

Beweis.

Es ist $(\mu_b P) = (RK_0^n g)$, wobei

$$g(u, x_1, \ldots, x_n, y) := \begin{cases} y+1, & \text{falls } \neg \bigvee_{z \leq y} P(x_1, \ldots, x_n, z) \wedge P(x_1, \ldots, x_n, y+1) \\ u & \text{sonst.} \end{cases}$$

g ist primitivrekursiv (Fallunterscheidung, Komprehension, beschränkte Quantifizierung), also auch $(\mu_b P)$. □

Beschränkte Minimalisierung

Satz

Wenn f eine $(n+1)$-stellige primitivrekursive Funktion ist, dann ist $(\mu_b f)$ eine $(n+1)$-stellige primitivrekursive Funktion.

Beweis.

Es ist $(\mu_b f) = (\mu_b P)$, wobei

$$P(x_1, \ldots, x_n, y) :\Longleftrightarrow\ f(x_1, \ldots, x_n, y) = 0.$$

P ist primitivrekursiv. Behauptung folgt aus vorigem Satz. □

Einige primitivrekursive Funktionen

Beispiel (ganzzahlige Division)

$$\mathrm{div}(x, y) = \mu d \leq x \colon (d + 1)y > x.$$

das kleinste $d \leq x$ mit $(d + 1)y > x$, sonst 0

$$\mathrm{div}(x, y) = (\mu_b P)(x, y, x) \quad \text{mit} \quad P(x, y, d) :\Longleftrightarrow (d + 1)y > x.$$

Beispiel (kleinstes gemeinsames Vielfaches)

$$\mathrm{kgV}(x, y) = \mu z \leq xy \colon (x|z \wedge y|z \wedge z > 0)$$

das kleinste $z \leq xy$ mit $x|z \wedge y|z \wedge z > 0$, sonst 0

$$\mathrm{kgV}(x, y) = (\mu_b P)(x, y, xy)$$
$$\text{mit} \quad P(x, y, z) :\Longleftrightarrow (x|z \wedge y|z \wedge z > 0)$$

Einige primitivrekursive Funktionen

Beispiel (größter gemeinsamer Teiler)

$$\mathrm{ggT}(x,y) = \mu t \leq x+y\colon (t|x \wedge t|y \wedge \neg \bigvee_{u \leq x+y} (u > t \wedge u|x \wedge u|y))$$

das kleinste $t \leq x+y$ mit $t|x \wedge t|y \wedge \neg \bigvee_{u \leq x+y}(u > t \wedge u|x \wedge u|y)$

$$\mathrm{ggT}(x,y) = (\mu_b P)(x,y,x+y)$$

$$\text{mit} \quad P(x,y,t) :\Longleftrightarrow (t|x \wedge t|y \wedge \neg \bigvee_{u \leq x+y} (u > t \wedge u|x \wedge u|y))$$

Unterabschnitt 6

Kodierungen

Die Cantor'sche Paarfunktion

Definition

Die Paarfunktion $(x, y) \mapsto \langle x, y \rangle : \mathbb{N}^2 \to \mathbb{N}$ ist definiert durch

$$\langle x, y \rangle = \frac{(x + y)(x + y + 1)}{2} + y.$$

Einige Eigenschaften der Paarfunktion

Lemma

Die Paarfunktion ist bijektiv und streng monoton steigend und für alle $x, y \in \mathbb{N}$ *gilt:*

$$x \leq \langle x, y \rangle$$
$$y \leq \langle x, y \rangle.$$

Lemma

Die Paarfunktion ist primitivrekursiv.

Die n-Tupel-Funktion

Definition

Für $n \in \mathbb{N}_+$ ist die n-Tupel-Funktion
$(x_1, \ldots, x_n) \mapsto \langle x_1, \ldots, x_n \rangle : \mathbb{N}^n \to \mathbb{N}$ definiert durch

$$\langle x \rangle := x$$

$$\langle x_1, \ldots, x_{n+1} \rangle := \langle \langle x_1, \ldots, x_n \rangle, x_{n+1} \rangle.$$

Einige Eigenschaften der n-Tupel-Funktion

Bemerkung

Die 2-Tupel-Funktion ist die Paarfunktion.

Lemma

Die n-Tupel-Funktion ist bijektiv und streng monoton steigend und für alle $x_1, \ldots, x_n \in \mathbb{N}$ gilt:

$$x_i \leq \langle x_1, \ldots, x_n \rangle \qquad \text{für } i = 1, \ldots, n.$$

Lemma

Die n-Tupel-Funktion ist primitivrekursiv.

Die Umkehrungen der n-Tupel-Funktionen

Definition

Die Funktion $\pi_k^n \colon \mathbb{N} \to \mathbb{N}$ für $k, n \in \mathbb{N}$ mit $1 \leq k \leq n$ ist definiert durch

$$\pi_k^n(\langle x_1, \ldots, x_n \rangle) := x_k.$$

Die Umkehrungen der n-Tupel-Funktionen

Satz

Die Funktionen π_k^n sind primitivrekursiv.

Beweis.

$\pi_k^n(x) = \mu x_k \leq x$:

$$\bigvee_{x_1 \leq x} \cdots \bigvee_{x_{k-1} \leq x} \bigvee_{x_{k+1} \leq x} \cdots \bigvee_{x_n \leq x} x = \langle x_1, \ldots, x_n \rangle.$$

- n-Tupelfunktion ist primitivrekursiv.
- π_k^n ergibt sich daraus mit beschränkter Quantifizierung und beschränkter Minimalisierung.
- π_k^n ist daher selbst primitivrekursiv. □

Iteration

Definition

Für eine einstellige zahlentheoretische Funktion f ist die Iteration $(\mathrm{It}\, f)$ von f eine zweistellige zahlentheoretische Funktion, die gegeben ist durch

$$(\mathrm{It}\, f)(x, y) = \underbrace{f(f(\ldots f(f}_{y \text{ mal das } f}(x))\ldots)).$$

Lemma

Wenn f eine einstellige primitivrekursive Funktion ist, dann ist $(\mathrm{It}\, f)$ auch primitivrekursiv.

Iteration

Definition

Für eine Funktion $\vec{f}\colon \mathbb{N}^n \to \mathbb{N}^n$ ist die Iteration $(\mathrm{It}\,\vec{f})$ von $\vec{f}$ die Funktion von $\mathbb{N}^{n+1}$ in $\mathbb{N}^n$, die gegeben ist durch

$$(\mathrm{It}\,\vec{f})(x_1,\ldots,x_n,y) = \underbrace{\vec{f}(\vec{f}(\ldots\vec{f}(\vec{f}}_{y \text{ mal das } f}(x_1,\ldots,x_n))\ldots)).$$

Lemma

Wenn $\vec{f}\colon \mathbb{N}^n \to \mathbb{N}^n$ primitivrekursiv ist, dann ist $(\mathrm{It}\,\vec{f})$ auch primitivrekursiv.

Gödels Kodierung von endlichen Folgen

Kodierung von Zeichen, Zeichenfolgen, usw.

- Kodierung von Zeichen, nichtleeren endlichen Folgen von Zeichen, nichtleeren endlichen Folgen von nichtleeren endlichen Folgen von Zeichen, usw. als positive natürliche Zahlen.
- Kodiere Zeichen als ungerade Zahlen
- Kodiere eine nichtleere endliche Folge $(F_1, \ldots, F_n)$ als die Zahl

$$2^{k_1} \cdot 3^{k_2} \cdots p_n^{k_n},$$

wobei k_i die Kodierung von F_i ist und p_i die i-te Primzahl ist.

Abschnitt 8

Berechenbare nicht-primitivrekursive Funktionen

Unterabschnitt 1

Die Peter'sche Funktion

Die Peter'sche Funktion

Definition (Peter'sche Funktion p)

$$p(0, y) := y + 1$$
$$p(x + 1, 0) := p(x, 1)$$
$$p(x + 1, y + 1) := p(x, p(x + 1, y))$$

Satz

Zu jeder n-stelligen primitivrekursiven Funktion f gibt es ein $k \in \mathbb{N}$, sodass für alle $x_1, \ldots, x_n \in \mathbb{N}$ gilt $f(x_1, \ldots, x_n) \leq p(k, x_1 + \cdots + x_n)$.

Die Peter'sche Funktion

Bemerkung

p ist berechenbar.

Satz

p ist nicht primitivrekursiv.

Beweis.

- Angenommen, p wäre primitivrekursiv.
- Sei $f(x) := p(x, x) + 1$ für alle $x \in \mathbb{N}$.
- Dann wäre auch f primitivrekursiv.
- Also gäbe es ein $k \in \mathbb{N}$, sodass $f(x) \leq p(k, x)$ für alle $x \in \mathbb{N}$.
- Also $f(k) \leq p(k, k)$. Widerspruch zu $f(k) = p(k, k) + 1$. □

Unterabschnitt 2

Das Cantor'sche Diagonalverfahren

Das Cantor'sche Diagonalverfahren

Notation

Jedes n-stellige primitivrekursive Funktional ϕ definiert eine n-stellige zahlentheoretische Funktion, die wir mit $[\phi]$ bezeichnen wollen.

Abzählung der 1-stelligen Funktionale

- Die Menge der einstelligen primitivrekursiven Funktionale ist abzählbar.
- Abzählung $(\phi_0, \phi_1, \phi_2, \dots)$
- Zu gegebenem k lässt sich ϕ_k explizit berechnen.

Das Cantor'sche Diagonalverfahren

Cantor'sches Diagonalverfahren

Für alle $x \in \mathbb{N}$ sei $f(x) := [\phi_x](x) + 1$.

Bemerkung

f ist berechenbar.

Satz

f ist nicht primitivrekursiv.

Beweis.

- Andernfalls gäbe es ein k mit $f = [\phi_k]$.
- $f(k) = [\phi_k](k) + 1 = f(k) + 1$. Widerspruch. □

Kein formales System für totale zahlentheoretische Funktionen erfasst alle berechenbaren zahlentheoretischen Funktionen.

Satz

Sei Σ ein Alphabet und L eine Sprache über Σ. Jedem Wort ϕ aus L sei eine zahlentheoretische Funktion $[\phi]$ zugeordnet mit den folgenden Eigenschaften.

- *Es ist entscheidbar, ob ein Wort aus Σ^* in L liegt.*
- *Zu einem Wort ϕ aus L ist die Stelligkeit von $[\phi]$ berechenbar.*
- *Zu $\phi \in L$ und $(x_1, \ldots, x_n) \in \mathbb{N}^n$ (mit $n =$ Stelligkeit von $[\phi]$) ist $[\phi](x_1, \ldots, x_n)$ berechenbar.*

Dann gibt es eine berechenbare zahlentheoretische Funktion, die keinem Wort aus L zugeordnet ist.

Cantor'sches Diagonalverfahren

Beweis mit dem Cantor'schen Diagonalverfahren

- Abzählung von Σ^* berechenbar
- Abzählung von $\{\phi \in L \mid [\phi] \text{ ist 1-stellig}\}$ berechenbar:
- $(\phi_0, \phi_1, \phi_2, \dots)$
- Für alle $x \in \mathbb{N}$ sei $f(x) := [\phi_x](x) + 1$.
- f ist berechenbar.
- Es gibt kein $\phi \in L$ mit $f = [\phi]$.
- Denn andernfalls gäbe es $k \in \mathbb{N}$ mit $f = [\phi_k]$.
- $f(k) = [\phi_k](k) + 1 = f(k) + 1$. Widerspruch. □

Unterabschnitt 3

Unentscheidbarkeit der Terminierung von Programmen

Unentscheidbarkeit der Terminierung von Programmen

Satz

Für eine Programmiersprache, in der jede berechenbare zahlentheoretische Funktion implementierbar ist, ist es unentscheidbar, ob ein gegebenes Programm bei gegebener Eingabe terminiert.

Unentscheidbarkeit der Terminierung von Programmen

Beweis

- Angenommen, die Terminierung wäre entscheidbar.
- Sei Σ das Alphabet der Programmiersprache.
- Sei L_n die Menge der Programme mit Eingabe aus $\mathbb{N}^n$ und Ausgabe aus $\mathbb{N}$ für $n \in \mathbb{N}$ und sei $L := \bigcup_{n \in \mathbb{N}} L_n$.
- Für $\phi \in L_n$ und $x_1, \ldots, x_n \in \mathbb{N}$ sei $[\phi](x_1, \ldots, x_n)$ die Ausgabe des Programms ϕ bei Eingabe $(x_1, \ldots, x_n)$, falls ϕ eine Ausgabe liefert, und 0 andernfalls.
- Dann ist entscheidbar, ob ein Wort aus Σ^* in L liegt.
- Zu einem Wort ϕ aus L ist die Stelligkeit von $[\phi]$ berechenbar.
- Zu $\phi \in L$ und $(x_1, \ldots, x_n) \in \mathbb{N}^n$ (mit $n =$ Stelligkeit von $[\phi]$) ist $[\phi](x_1, \ldots, x_n)$ berechenbar. Begründung:

Unentscheidbarkeit der Terminierung von Programmen

Beweis (Fortsetzung)

- Nach Annahme ist die Terminierung entscheidbar. Lasse Algorithmus laufen, der entscheidet, ob das Programm ϕ bei Eingabe von $(x_1, \ldots, x_n)$ terminiert. Wenn er mit „ja“ antwortet, lasse ϕ mit Eingabe $(x_1, \ldots, x_n)$ laufen und liefere die Ausgabe dieses Laufes. Andernfalls liefere die 0 als Antwort.
- Nach dem vorigen Satz gäbe es also eine berechnbare zahlentheoretische Funktion f, die nicht ein $[\phi]$ wäre.
- Nach Voraussetzung wäre f aber durch ein Programm ϕ implementierbar.
- Das Programm ϕ würde daher für jede Eingabe aus $\mathbb{N}^n$ terminieren.
- Daher wäre $f = [\phi]$. Widerspruch. □

Abschnitt 9

μ-partiellrekursive Funktionen

Unterabschnitt 1

Partielle zahlentheoretische Funktionen

Partielle zahlentheoretische Funktionen

Definition

- Eine n-stellige partielle zahlentheoretische Funktion ist eine Funktion $f : A \to \mathbb{N}$ mit $A \subseteq \mathbb{N}^n$.
- Wir schreiben $f : \mathbb{N}^n \to_p \mathbb{N}$.
- Wenn $A = \mathbb{N}^n$, dann sprechen wir von einer totalen Funktion.

Idee

Für $(x_1, \ldots, x_n) \in \mathbb{N}^n \setminus A$ ist $f(x_1, \ldots, x_n)$ undefiniert.
Dem entspricht Nichtterminierung eines Algorithmus.

Komposition

Definition

Sei f eine m-stellige partielle zahlentheoretische Funktion ($m > 0$) und seien $g_1, \ldots, g_m$ n-stellige partielle zahlentheoretische Funktionen.
Dann ist die Komposition $(fg_1 \ldots g_m)$ von f mit $g_1, \ldots, g_m$ eine n-stellige partielle zahlentheoretische Funktion.

$$(fg_1 \ldots g_m)(x_1, \ldots, x_n) := f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)),$$

wenn $g_i(x_1, \ldots, x_n)$ für alle $i = 1, \ldots, m$ definiert ist und $f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$ definiert ist.
Andernfalls ist $(fg_1 \ldots g_m)(x_1, \ldots, x_n)$ undefiniert.

Primitive Rekursion

Definition

Sei f eine n-stellige partielle zahlentheoretische Funktion und sei g eine $(n+2)$-stellige partielle zahlentheoretische Funktion.
Dann ist die durch primitive Rekursion aus f und g gebildete Funktion (Rfg) eine $(n+1)$-stellige partielle zahlentheoretische Funktion.

$$(Rfg)(x_1, \ldots, x_n, 0) := f(x_1, \ldots, x_n),$$

wenn $f(x_1, \ldots, x_n)$ definiert ist.
Andernfalls ist $(Rfg)(x_1, \ldots, x_n, 0)$ undefiniert.

$$(Rfg)(x_1, \ldots, x_n, y+1) := g((Rfg)(x_1, \ldots, x_n, y), x_1, \ldots, x_n, y),$$

wenn $(Rfg)(x_1, \ldots, x_n, y)$ und $g((Rfg)(x_1, \ldots, x_n, y), x_1, \ldots, x_n, y)$ definiert sind. Andernfalls ist $(Rfg)(x_1, \ldots, x_n, y+1)$ undefiniert.

Minimalisierung oder μ-Operator

Definition

Sei f eine $(n+1)$-stellige partielle zahlentheoretische Funktion.
Dann ist die Minimalisierung von f oder die durch Anwendung des μ-Operators aus f gebildete Funktion (μf) eine n-stellige partielle zahlentheoretische Funktion.

$$(\mu f)(x_1, \ldots, x_n) := y,$$

wenn $f(x_1, \ldots, x_n, z)$ definiert und > 0 ist für alle $z < y$ und $f(x_1, \ldots, x_n, y)$ definiert und $= 0$ ist.
Wenn es kein solches y gibt, dann ist $(\mu f)(x_1, \ldots, x_n)$ undefiniert.

Unterabschnitt 2

μ-partiellrekursive Funktionen

μ-partiellrekursive Funktionen

Definition (μ-partiellrekursive Funktionen)

- Jede der Grundfunktionen O, N und P_i^n ist μ-partiellrekursiv.
- Eine Komposition von μ-partiellrekursiven Funktionen ist μ-partiellrekursiv.
- Eine durch primitive Rekursion aus μ-partiellrekursiven Funktionen gebildete Funktion ist μ-partiellrekursiv.
- Die Minimalisierung einer μ-partiellrekursiven Funktion ist μ-partiellrekursiv.

Definition (μ-partiellrekursive Funktionale)

- $\mathtt{O}$, $\mathtt{N}$, $\mathtt{P}_i^n$
- $(\phi\psi_1 \dots \psi_m)$
- $(\mathtt{R}\phi\psi)$
- $(\mu\phi)$

Einige μ-partiellrekursive Funktionen

- Die Peter'sche Funktion
- Die nicht primitivrekursive berechenbare Funktion f, die wir mit dem Cantor'schen Diagonalverfahren konstruiert haben:

$$f(x) := [\phi_x](x) + 1$$

- Jede partielle zahlentheoretische Funktion, die in einem der bekannten formalen Systeme (Programmiersprachen, μ-partiellrekursive Funktionale, Turingmaschinen, Chomskygrammatiken, Logikkalküle, λ-Kalkül, Kombinatorenkalkül, u.s.w.) dargestellt werden kann

Abschnitt 10

Intuitive Berechenbarkeit und verwandte Begriffe

Algorithmus

Dieser Abschnitt stützt sich auf den intuitiven und nicht mathematisch formalisierten Begriff des **Algorithmus**. Die Wörter „Definition“, „Satz“, „Beweis“ sind hier daher **nicht** in einem formalen Sinne, etwa **im Sinne der Mathematik**, zu verstehen!

Ein Algorithmus $\mathcal{A}$ zur Berechnung einer Funktion erwartet eine Eingabe $x \in X$ und produziert eine Ausgabe $y \in Y$.

Der Datenbereich X bzw. Y ist jeweils die Menge aller Dinge eines bestimmten Datentyps. Im Folgenden seien X und Y Datenbereiche.

Mit $\mathcal{A}(x)$ bezeichnen wir den Lauf des Algorithmus bei Eingabe x.

Algorithmus

Beispiel

Sei $\mathcal{A}$ der euklidische Algorithmus.
$X = \mathbb{N}^2$
$Y = \mathbb{N}$
$\mathcal{A}((a, b))$ liefert als Ausgabe $\mathrm{ggT}(a, b)$.

Statt $\mathcal{A}((a, b))$ schreiben wir kurz $\mathcal{A}(a, b)$.

Berechenbarkeit

Sei f eine partielle Funktion von X nach Y.
Also $f : A \to Y$ mit $A \subseteq X$.

Definition

f heißt (partiell)berechenbar, wenn es einen Algorithmus $\mathcal{A}$ gibt so, dass gilt

$$\mathcal{A}(x) \text{ liefert } \begin{cases} \text{die Ausgabe } f(x), & \text{wenn } x \in A \\ \text{keine Ausgabe}, & \text{wenn } x \in X \setminus A\,. \end{cases}$$

Beispiel

$\mathcal{A}(x)$: $n := 0$; while $2n \neq x$ do $n := n + 1$; return n
berechenet die 1-stellige partielle zahlentheoretische Funktion
$f : \mathbb{N} \to_p \mathbb{N}$, die gegeben ist durch

$$f(x) \begin{cases} = \frac{x}{2}, & \text{wenn } x \text{ gerade} \\ \text{undefiniert}, & \text{wenn } x \text{ ungerade}\,. \end{cases}$$

Entscheidbarkeit

Sei $A \subseteq X$.

Definition

A heißt entscheidbar, wenn es einen Algorithmus $\mathcal{A}$ gibt so, dass gilt

$$\mathcal{A}(x) \text{ liefert die Ausgabe } \begin{cases} \text{„ja“}, & \text{wenn } x \in A \\ \text{„nein“}, & \text{wenn } x \in X \setminus A\,. \end{cases}$$

$\mathcal{A}$ ist Entscheidungs-Algorithmus für A

Semi-Entscheidbarkeit

Sei $A \subseteq X$.

Definition

A heißt semi-entscheidbar, wenn es einen Algorithmus $\mathcal{A}$ gibt so, dass gilt

$$\mathcal{A}(x) \begin{cases} \text{terminiert}, & \text{wenn } x \in A \\ \text{terminiert nicht}, & \text{wenn } x \in X \setminus A\,. \end{cases}$$

$\mathcal{A}$ ist Semientscheidungs-Algorithmus für A

Semi-Entscheidbarkeit

Satz

Jede entscheidbare Menge ist semi-entscheidbar.

Beweis.

- Sei $\mathcal{A}$ ein Entscheidungs-Algorithmus für A.
- $\mathcal{B}(x)$:
  Lasse $\mathcal{A}(x)$ laufen. Ausgabe sei y.
  Wenn $y =$„ja“, brich ab. Sonst gehe in eine Endlosschleife.
- Dann ist $\mathcal{B}$ ein Semientscheidungs-Algorithmus für A.

Aufzählbarkeit

Ein Algorithmus kann 0, 1 oder mehr, auch unendlich viele Ausgaben liefern.

Beispiel

$n := 0$; while true do $\ulcorner$ print n^2; $n := n + 1$ $\urcorner$
liefert als Ausgaben alle Quadratzahlen.

Aufzählbarkeit

Sei $A \subseteq X$.

Definition

- A heißt aufzählbar, wenn es einen Algorithmus gibt, der 0, 1 oder mehr, möglicherweise auch unendlich viele Ausgaben $a_1, a_2, a_3, \ldots$ liefert so, dass $A = \{a_1, a_2, a_3, \ldots\}$.
- Wir sagen dann, der Algorithmus zählt die Menge A auf.

Nicht verwechseln mit „abzählbar“!

Aufzählbarkeit

Satz

A aufzählbar $\Rightarrow$ *A semi-entscheidbar.*

Beweis.

- Sei $\mathcal{A}$ ein Algorithmus, der A aufzählt.
- $\mathcal{B}(x)$: Lass $\mathcal{A}$ laufen, bis es die Ausgabe x liefert.
- Dann ist $\mathcal{B}$ ein Semientscheidungsalgorithmus für A, d.h.
  - Wenn $x \in A$, dann terminiert $\mathcal{B}(x)$.
  - Wenn $x \in X \setminus A$, dann terminiert $\mathcal{B}(x)$ nicht.

Aufzählbarkeit

Satz

A semi-entscheidbar $\Rightarrow$ A aufzählbar.

Beweis.

- Sei $\mathcal{A}$ ein Semientscheidungsalgorithmus für A.
- X ist aufzählbar: $X = \{x_0, x_1, x_2, \dots\}$.
- Paarfunktion $(i,j) \mapsto \langle i,j \rangle : \mathbb{N}^2 \to \mathbb{N}$.
- Projektionen π_1, π_2: $\quad \pi_1(\langle i,j \rangle) = i, \quad \pi_2(\langle i,j \rangle) = j$.
- $\mathcal{B}$: $\quad n := 0$; while true do
  ⌜ wenn $\mathcal{A}(x_{\pi_1(n)})$ nach genau $\pi_2(n)$ Schritten terminiert,
  dann gib $x_{\pi_1(n)}$ aus; $\quad n := n+1$ ⌟
- Dann zählt der Algorithmus $\mathcal{B}$ die Menge A auf.

□

Aufzählbarkeit

Satz

Sei $f : X \to_p Y$.
Dann sind die folgenden Aussagen zueinander äquivalent.

1. *f ist eine partiellberechenbare Funktion*
2. *$\mathrm{Graph}(f)$ ist semi-entscheidbar*
3. *$\mathrm{Graph}(f)$ ist aufzählbar*

Beweis.

- 1.⇒2.: Algorithmus $\mathcal{A}$ berechne f.
  $\mathcal{B}(x, y)$: Lasse $\mathcal{A}(x)$ laufen. Ausgabe sei z. Wenn $y = z$, dann terminiere, sonst gehe in Endlosschleife.
- 2.⇒3. wurde bereits gezeigt.
- 3.⇒1.: Algorithmus $\mathcal{A}$ zähle $\mathrm{Graph}(f)$ auf.
  $\mathcal{B}(x)$: Lasse $\mathcal{A}$ laufen, bis er ein (x, y) ausgibt. Gib y aus.

□

Aufzählbarkeit

Satz

Sei $A \subseteq X$. Dann sind die folgenden Aussagen zueinander äquivalent.

1. *A ist semi-entscheidbar*
2. *A ist aufzählbar*
3. *Die einzige Funktion $f : A \to \{0\}$ ist partiellberechenbar*
4. *A ist Definitionsbereich einer partiellberechenbaren Funktion*
5. *A ist Bildbereich einer partiellberechenbaren Funktion*
6. *$A = \emptyset$ oder A ist Bildbereich einer totalen berechenbaren Funktion*

Beweis.

- 2.$\Leftrightarrow$1.$\Leftrightarrow$3.$\Leftrightarrow$4. und 6.$\Rightarrow$5.: einfach
- 2.$\Rightarrow$6.: A unendlich, $\mathcal{A}$ Aufzählungsalgorithmus: $f(n) :=$ n-te Ausgabe
- 5.$\Rightarrow$2.: $A = \mathrm{Bild}(f)$; $\mathrm{Graph}(f)$ aufzählbar, Algorithmus $\mathcal{A}$.
  $\mathcal{B}$: Lasse $\mathcal{A}$ laufen. Bei jeder Ausgabe (x, y) von $\mathcal{A}$ gib y aus.

Entscheidbarkeit

Satz

Sei $A \subseteq X$. Dann sind die folgenden Aussagen zueinander äquivalent.

1. *A ist entscheidbar*
2. *A und $X \setminus A$ sind semi-entscheidbar*
3. *Die charakteristische Funktion χ_A von A ist eine totale berechenbare Funktion.*

Beweis.

- 1.$\Leftrightarrow$3. und 1.$\Rightarrow$2.: trivial
- 2.$\Rightarrow$1.: Lasse Entscheidungsalgorithmen für A und für $X \setminus A$ gleichzeitig oder zeitlich verzahnt laufen.

Nichtdeterministischer Algorithmus

Definition

- Ein nichtdeterministischer Algorithmus ist eine rein mechanische Rechenvorschrift.
- Es ist nicht notwendig in jedem Schritt genau festgelegt, was geschieht, sondern es können mehrere Wahlmöglichkeiten offen sein.

Definition

Ein nichtdeterministischer Algorithmus akzeptiert eine Eingabe $x \in X$, wenn es eine terminierende Berechnung zu dieser Eingabe gibt.

Ein deterministischer Algorithmus (oder einfach: ein Algorithmus) ist ein Spezialfall eines nichtdeterministischen Algorithmus:
keine Wahl zwischen verschiedenen Möglichkeiten

Nichtdeterministischer Algorithmus

Satz

Sei $A \subseteq X$. Dann sind die folgenden Aussagen zueinander äquivalent.

1. *A ist aufzählbar.*
2. *A wird von einem (deterministischen) Algorithmus akzeptiert.*
3. *A wird von einem nichtdeterministischen Algorithmus akzeptiert.*

Beweis.

- 1.⇒2.: Sei A aufzählbar. Dann ist A semi-entscheidbar, also gilt 2.
- 2.⇒3.: Jeder (deterministische) Algorithmus ist auch ein nichtdeterministischer Algorithmus.

□

Nichtdeterministischer Algorithmus

Beweis.

- 3.⇒1.:
  - Sei $\mathcal{A}$ ein nichtdeterministischer Algorithmus, der $A \subseteq X$ akzeptiert.
  - Sei B der Datenbereich der endlichen Berechnungen durch $\mathcal{A}$.
  - Wenn b eine Berechnung durch $\mathcal{A}$ zur Eingabe x ist, dann ist $(b, x) \in B \times X$.
  - Sei $(b_1, x_1), (b_2, x_2), (b_3, x_3), \ldots$ eine Aufzählung von $B \times X$.
  - $\mathcal{B}$: $i := 1$; while true do
    ⌜ if b_i ist eine Berechnung zu $\mathcal{A}(x_i)$ then print x_i;
    i:=i+1 ⌟
  - Dann zählt der Algorithmus $\mathcal{B}$ die von $\mathcal{A}$ akzeptierte Menge A auf.

Existenzdarstellung aufzählbarer Mengen

Satz

Sei $A \subseteq X$. Sei Y unendlich. Dann sind die folgenden Aussagen zueinander äquivalent.

1. *A ist aufzählbar.*
2. *Es gibt ein entscheidbares $C \subseteq X \times Y$ so, dass für alle $x \in X$ gilt:*

$$x \in A \Leftrightarrow \bigvee_{y \in Y} C(x, y)$$

3. *Es gibt ein aufzählbares $C \subseteq X \times Y$ so, dass für alle $x \in X$ gilt:*

$$x \in A \Leftrightarrow \bigvee_{y \in Y} C(x, y)$$

Existenzdarstellung aufzählbarer Mengen

Beweis.

- 1.⇒2.: Sei $\mathcal{A}$ ein Algorithmus, der A aufzählt. Es gibt eine bijektive Aufzählung $(y_1, y_2, \dots)$ von Y. Sei

$$C(x, y_n) :\Leftrightarrow \mathcal{A} \text{ gibt } x \text{ im } n\text{-ten Schritt aus .}$$

  Dann ist C entscheidbar.
- 2.⇒3.: trivial
- 3.⇒1.: Sei $\mathcal{A}$ ein Algorithmus, der C aufzählt. Lasse $\mathcal{A}$ laufen. Wenn $\mathcal{A}$ ein (x, y) ausgibt, gib x aus.

Unterabschnitt 1

Kurze Zusammenfassung

Definitionen

- f (partiell)berechenbar: existiert Berechnungsalgorithmus
- A entscheidbar: existiert Entscheidungsalgorithmus
- A semientscheidbar: existiert Semientscheidungsalgorithmus
- A aufzählbar: existiert Aufzählungsalgorithmus

(Dabei Algorithmus jeweils als deterministisch zu verstehen)

Entscheidbarkeit

A entscheidbar $\Rightarrow$ A semientscheidbar

Die folgenden Aussagen sind äquivalent:

- A entscheidbar
- A und $X \setminus A$ semientscheidbar
- χ_A berechenbar

Semientscheidbarkeit

Die folgenden Aussagen sind äquivalent:

- A ist semientscheidbar
- A ist aufzählbar
- A ist Definitionsbereich einer partiellberechenbaren Funktion
- A ist Wertebereich einer partiellberechenbaren Funktion
- $A = \emptyset$ oder A ist Wertebereich einer totalen berechenbaren Funktion
- A wird von einem deterministischen Algorithmus akzeptiert
- A wird von einem nichtdeterministischen Algorithmus akzeptiert
- A wird von einem Erzeugungssystem mit entscheidbaren Regeln erzeugt
- Es gibt ein entscheidbares C, sodass $x \in A \Leftrightarrow \exists y C(x, y)$
- Es gibt ein semientscheidbares C, sodass $x \in A \Leftrightarrow \exists y C(x, y)$

Berechenbarkeit

Die folgenden Aussagen sind äquivalent:

- f partiellberechenbar
- $\mathrm{Graph}(f)$ semientscheidbar

Die folgenden Aussagen sind äquivalent

- f total und berechenbar
- f total und $\mathrm{Graph}(f)$ semientscheidbar
- f total und $\mathrm{Graph}(f)$ entscheidbar

Church'sche These

- Es wurden verschiedene mathematisch formale Systeme zur Beschreibung des Berechenbarkeits-/Semientscheidbarkeits-Begriffs aufgestellt.
- Alle diese Systeme beschreiben den gleichen Berechenbarkeits- und Semientscheidbarkeits-Begriff (sind mathematisch äquivalent).

Church'sche These

Diese Systeme beschreiben genau die Klasse der partiellberechenbaren Funktionen und die Klasse der semi-entscheidbaren Mengen.

Berechenbarkeit

Mathematisch formaler Begriff	Intuitiver Begriff
Partiellrekursive Funktion	Partiellberechenbare Funktion
Rekursive Funktion	Totale berechenbare Funktion
Rekursiv aufzählbare Menge	semi-entscheidbare Menge
Rekursive Menge	Entscheidbare Menge

Englisch für „rekursiv aufzählbar“:
recursively enumerable r.e.

Abschnitt 11

Die Church'sche These

Unterabschnitt 1

Formale Systeme

Formale Systeme

Formale Systeme

- μ-partiellrekursive Funktionen
- Logikkalküle
- SLD-Resolution (Prolog)
- Chomsky-Grammatiken
- Turing-Maschinen
- λ-Kalkül
- Kombinatoren-Kalkül
- Termersetzungssysteme
- Programmiersprachen
- ...

Partiell berechenbare Funktionen und aufzählbare Mengen

Bemerkung

Jedes dieser Systeme beschreibt eines der folgenden Dinge:

- Partiell berechenbare Funktionen $f : X \to_p Y$
- Aufzählbare Teilmengen eines Datenbereichs X.

Bemerkung

Die Begriffe der partiellen Berechenbarkeit und der Aufzählbarkeit sind aufeinander zurückführbar.

- Eine Menge ist genau dann aufzählbar, wenn sie der Definitionsbereich / Wertebereich einer partiell berechenbaren Funktion ist.
- Eine partielle Funktion ist genau dann partiell berechenbar, wenn ihr Graph aufzählbar ist.

Folgerung

Jedes dieser formalen Systeme beschreibt jedes der folgenden Dinge:

- *Eine Klasse von partiell berechenbaren Funktionen*
- *Eine Klasse von aufzählbaren Mengen.*

Satz

All diese formalen Systeme beschreiben dieselbe Klasse von partiell berechenbaren Funktionen und dieselbe Klasse von aufzählbaren Mengen. (Eventuell nach Umkodierung der Datenbereiche)

Beweisidee

Simuliere ein System im anderen.

Partiellrekursive Funktionen und rekursiv aufzählbare Mengen

Definition

- Eine partielle Funktion heißt partiellrekursiv, wenn sie in einem dieser formalen Systeme darstellbar ist.
- Sie heißt rekursiv, wenn sie partiellrekursiv und total ist.

Definition

- Eine Teilmenge eines Datenbereichs heißt rekursiv aufzählbar, wenn sie in einem dieser formalen Systeme darstellbar ist.
- Sie heißt rekursiv, wenn ihre charakteristische Funktion rekursiv ist.

Folgerung

- Eine partielle zahlentheoretische Funktion ist genau dann partiellrekursiv, wenn sie μ-partiellrekursiv ist.
- Eine totale zahlentheoretische Funktion ist genau dann rekursiv, wenn sie μ-rekursiv ist.

Satz

Eine Menge $A \subseteq X$ ist genau dann rekursiv, wenn A und $X \setminus A$ rekursiv aufzählbar sind.

Satz

Eine Teilmenge A von $\mathbb{N}^n$ bzw. von $\mathbb{N}$ ist genau dann
rekursiv aufzählbar,

- wenn A der Definitionsbereich einer μ-partiellrekursiven Funktion ist.
- wenn A der Wertebereich einer μ-partiellrekursiven Funktion ist.
- wenn A der Wertebereich einer μ-rekursiven Funktion ist.
- wenn A der Wertebereich einer primitivrekursiven Funktion ist.
- wenn es ein $(n+1)$-stelliges primitivrekursives Prädikat P gibt mit

$$A(x_1, \ldots, x_n) \iff \exists y\, P(x_1, \ldots, x_n, y)$$

Unterabschnitt 2

Church'sche These

Church'sche These

Church'sche These (Church-Turing-These

Jede partiell berechenbare Funktion ist partiellrekursiv.

Bemerkung

Die Church'sche These sagt also aus, dass die obengenannten formalen Systeme alle partiell berechenbaren Funktionen erfassen.

Folgerung (aus der Church'schen These)

- *Eine partielle Funktion ist genau dann partiell berechenbar, wenn sie partiellrekursiv ist.*
- *Eine totale Funktion ist genau dann berechenbar, wenn sie rekursiv ist.*
- *Eine Menge ist genau dann semi-entscheidbar, wenn sie rekursiv aufzählbar ist.*
- *Eine Menge ist genau dann entscheidbar, wenn sie rekursiv ist.*

Bemerkung

- Die Church'sche These bezieht sich auf den intuitiven Begriff der Berechenbarkeit, der sich wiederum auf den intuitiven Begriff des Algorithmus stützt.
- Sie besagt, dass die obengenannten Formalen Systeme adäquate mathematische Formalisierungen dieses intuitiven Begriffs darstellen.
- Daher ist die Church'sche These keine mathematische Aussage und lässt sich mathematisch nicht beweisen.

Abschnitt 12

Unentscheidbarkeitssätze der Logik

Die Zahlentheorie ist nicht formalisierbar

Satz (Kurt Gödel)

Zu jedem korrekten formalen System der Zahlentheorie gibt es eine geschlossene zahlentheoretische Formel H so, dass weder H noch $\neg H$ in dem System beweisbar ist.

Beweisidee

- Sei $u_1 := \ulcorner x_1 \urcorner$. Definiere eine Formel Fx_1 durch

$$Fx_1 := \neg \mathrm{Bew}(\mathrm{sb}(x_1{}^{u_1}_{\mathrm{zahl}(x_1)})).$$

- Für eine Formel Gx_1 und für $n := \ulcorner Gx_1 \urcorner$ ist dann

$$F\underline{n} \;=\; \neg \mathrm{Bew}(\mathrm{sb}(n^{u_1}_{\mathrm{zahl}(n)})).$$

- $\mathrm{sb}(n^{u_1}_{\mathrm{zahl}(n)})$ ist aber die Gödelnummer von $G\underline{n}$, also von $G\underline{\ulcorner Gx_1 \urcorner}$.
- $F\underline{\ulcorner Gx_1 \urcorner}$ besagt also, dass $G\underline{\ulcorner Gx_1 \urcorner}$ nicht beweisbar ist.
- Wählt man $Gx_1 := Fx_1$, so erhält man

$$F\underline{\ulcorner Fx_1 \urcorner} \iff F\underline{\ulcorner Fx_1 \urcorner} \text{ ist nicht beweisbar.}$$

- Sei $H := F\underline{\ulcorner Fx_1 \urcorner}$.
- Dann besagt H, dass H nicht beweisbar ist.

Beweisidee (Fortsetzung)

- Die Formel H besagt also in etwa:
  „Ich bin nicht beweisbar.“
- H muss wahr sein. Andernfalls wäre H falsch und damit beweisbar im Widerspruch zur Korrektheit des Kalküls.
- Also ist H wahr und daher nicht beweisbar.
- Also ist $\neg H$ falsch und daher wegen der Korrektheit des Kalküls nicht beweisbar.

Folgerung

Für die Zahlentheorie gibt es keinen Kalkül, der korrekt und vollständig ist.

Dies steht im Gegensatz zur reinen Prädikatenlogik erste Stufe, wo der vorgestellte Logikkalkül ein korrekter und vollständiger Kalkül ist. In der reinen Prädikatenlogik erster Stufe gilt jedoch

Satz

Die Menge aller natürlichen Zahlen, die nicht Gödelnummer einer allgemeingültigen Formel sind, ist nicht allgemeingültigkeitsdefinierbar: Es gibt keine Formel Ax_1 der Prädikatenlogik erster Stufe, die nur x_1 als freie Variable enthält, so, dass für alle geschlossenen Formeln G gilt

$$\models A\underline{\ulcorner G\urcorner} \quad \Longleftrightarrow \quad \not\models G. \qquad *$$

Beweisidee indirekt (angenommen *)

- Die „Diagonalfunktion“ $d : \ulcorner Fx_1\urcorner \mapsto \ulcorner F\underline{\ulcorner Fx_1\urcorner}\urcorner$ ist primitivrekursiv.
- Es gibt daher eine Formel Bx_1, sodass für alle Fx_1 gilt

$$\begin{aligned} \models B\underline{\ulcorner Fx_1\urcorner} \quad &\Longleftrightarrow \quad \models A\underline{\ulcorner F\underline{\ulcorner Fx_1\urcorner}\urcorner} \\ &\Longleftrightarrow \quad \not\models F\underline{\ulcorner Fx_1\urcorner}. \qquad \text{(wegen *)} \end{aligned}$$

- Setze $Fx_1 :\equiv Bx_1$. Dann $\models B\underline{\ulcorner Bx_1\urcorner} \Longleftrightarrow \not\models B\underline{\ulcorner Bx_1\urcorner}$. Widerspruch

Satz

Die Widerspruchsfreiheit eines korrekten hinreichend ausdrucksstarken Kalküls der Zahlentheorie ist in dem Kalkül selbst nicht beweisbar.

Abschnitt 13

Unentscheidbarkeit der Terminierung von Programmen

Programmiersprachen

Im Folgenden nehmen wir an, dass $\mathcal{S}$ eine Programmiersprache ist, die unter anderen die folgenden Strukturen und Konstrukte bietet:

- Datentyp der Wörter (Strings)
- Fallunterscheidung
- Endlosschleife

Definition

- Seien $X_1, \ldots, X_n$ Datenbereiche von $\mathcal{S}$.
- Sei $A \subseteq X_1 \times \cdots \times X_n$.

Dann sagen wir, A sei in $\mathcal{S}$ entscheidbar, wenn es in $\mathcal{S}$ ein Programm Q gibt derart, dass für alle $x_1 \in X_1, \ldots, x_n \in X_n$ gilt: Der Aufruf

$$Q(x_1, \ldots, x_n) \text{ liefert } \begin{cases} \text{„ja“}, & \text{wenn } (x_1, \ldots, x_n) \in A \\ \text{„nein“} & \text{sonst .} \end{cases}$$

Definition

Ein Programm P heiße selbstterminierend, wenn P als Eingabe ein Wort erwartet und bei Eingabe von P terminiert.

Definition

- Wenn P ein Wort als Eingabe erwartet und bei Eingabe des Wortes w terminiert, heißt (P, w) ein terminierendes Paar.
- Sei $\mathcal{T}$ die Menge der terminierenden Paare.

Satz

$\mathcal{T}$ ist in $\mathcal{S}$ unentscheidbar.

Beweis

- Angenommen, $\mathcal{T}$ wäre in $\mathcal{S}$ entscheidbar.
- Dann wäre die Menge der selbstterminierenden Programme in $\mathcal{S}$ entscheidbar.
- Dann gäbe es ein Programm ST derart, dass für alle Wörter P gilt:

$$\mathrm{ST}(P) \text{ liefert } \begin{cases} \text{„ja“}, & \text{wenn } P \text{ selbstterminierend ist} \\ \text{„nein“} & \text{sonst} \end{cases}$$

Beweis (Fortsetzung)

- Wir definieren ein Programm P_0 folgendermaßen:
  $P_0(P)$:
  - Wenn $\mathrm{ST}(P)$ „ja“ liefert, dann gehe in Endlosschleife.
  - Sonst brich ab.
- Dann gilt:

$$\begin{aligned} P_0 \text{ ist selbstterminierend} &\iff P_0(P_0) \text{ terminiert} \\ &\iff \mathrm{ST}(\mathrm{P_0}) \text{ liefert nicht „ja“} \\ &\iff P_0 \text{ ist nicht selbstterminierend.} \end{aligned}$$

- Widerspruch □

Turing-vollständige Programmiersprachen

Definition

Eine Programmiersprache heißt universell oder Turing-vollständig, wenn man darin jede partiellrekursive Funktion implementieren kann.

Beispiele

- Prolog
- Lisp
- Haskell
- C
- Ada
- Java

Unentscheidbarkeit der Terminierung von Programmen

Satz

- *Nehmen wir die Church-Turing-These an.*
- *Nehmen wir weiter an, die Programmiersprache $\mathcal{S}$ sei universell.*
- *Dann ist die Terminierung von Programmen von $\mathcal{S}$ ein unentscheidbares Problem.*

Beweis.

$\mathcal{T}$ Menge der terminierenden Paare (P, w).

- Nach dem vorigen Satz ist $\mathcal{T}$ nicht in $\mathcal{S}$ entscheidbar.
- Da $\mathcal{S}$ universell ist, ist $\mathcal{T}$ daher nicht rekursiv.
- Nach der Church-Turing-These ist $\mathcal{T}$ daher nicht entscheidbar.
- Also ist die Terminierung von Programmen von $\mathcal{S}$ unentscheidbar. □

Abschnitt 14

Der Normalformsatz von Kleene

Der Normalformsatz von Kleene

Satz

Es gibt eine 1-stellige primitivrekursive Funktion U und zu jedem $r \in \mathbb{N}$ eine $r+2$-stellige primitivrekursive Funktion T_r so, dass für jede r-stellige partiellrekursive zahlentheoretische Funktion f eine natürliche Zahl e existiert so, dass

$$f(x_1, \ldots, x_r) = \begin{cases} (U(\mu T_r))(e, x_1, \ldots, x_r), & \text{falls dies definiert ist} \\ \text{undefiniert} & \text{sonst} \end{cases}$$

für alle $(x_1, \ldots, x_r) \in \mathbb{N}^r$.

Definition

Diese Zahl e nennt man einen Index der partiellen zahlentheoretischen Funktion f. Man schreibt dann oft $f = \{e\}^r$.

Bemerkung

- $(U(\mu T_r))$ ist eine universelle partiellrekursive Funktion. Durch sie kann man jede partiellrekursive Funktion berechnen.
- Entsprechend gibt es auch für Programmiersprachen ein universelles Programm, einen sogenannten Interpreter, der jedes Programm der Programmiersprache simulieren kann.
- Auch für den Formalismus der Turingmaschinen (s. später) gibt es eine universelle Turingmaschine, die jede beliebige Turingmaschine simulieren kann.
- Entsprechendes gilt auch für alle anderen Formalismen zur Beschreibung partiell berechenbarer Funktionen.

Abschnitt 15

Chomsky-Grammatiken

- hauptsächlich zur Beschreibung der
  Syntax von natürlichen Sprachen und von Programmiersprachen
- Dazu geeignet kontextfreie Chomsky-Grammatiken.
  ermöglichen Syntaxanalyse und Konstruktion von Strukturbäumen.
- Bequemere Darstellung von kontextfreien Grammatiken durch Backus-Naur-Form. Auch reguläre Ausdrücke auf der rechten Seite einer Regel erlaubt.
- Allgemeine Chomsky-Grammatiken
  können beliebige rekursiv aufzählbare Sprachen erzeugen.

Abschnitt 16

Turing-Maschinen

Turing-Maschinen

- Turing-Maschinen definieren ein einfaches und doch universelles Maschinen-Modell.
- Turing-Maschinen erlauben die Berechnung beliebiger partiellrekursiver Funktionen.
- Turing-Maschinen erlauben die Beschreibung beliebiger rekursiv aufzählbarer Sprachen.
- Turing-Maschinen werden in der Komplexitätstheorie zur Definition und Untersuchung der Komplexität von Berechnungen verwendet.

Turing-Berechenbarkeit

Definition

Wenn eine Turingmaschine T eine Konfiguration $u\underset{S}{a}v$ im Startzustand in eine Konfiguration $w\underset{H}{b}x$ im Haltezustand überführt, also $u\underset{S}{a}v \Rightarrow_T^* w\underset{H}{b}x$, so schreiben wir $u\underline{a}v \Rrightarrow_T w\underline{b}x$.

Definition

Sei $f : (\Delta^*)^r \to_p \Delta^*$. Wir sagen, f werde berechnet durch eine Turingmaschine $T = (Q, \Sigma, S, H, \delta)$, wenn für alle $u_1, \ldots, u_r \in \Delta^*$ gilt

- Wenn $(u_1, \ldots, u_r) \in \mathrm{Def}(f)$, dann $u_1\# \ldots \#u_r\underline{\#} \Rrightarrow_T f(u_1, \ldots, u_r)\underline{\#}$.
- Andernfalls gibt es keine Berechnung, die mit $u_1\# \ldots \#u_r\underset{S}{\#}$ beginnt und hält, d.h. im Zustand H endet.

Man sagt dann, f sei turingberechenbar.

Definition

Eine partielle zahlentheoretische Funktion heißt turingberechenbar, wenn die entsprechende partielle Funktion auf dem einelementigen Alphabet $\{|\}$ turingberechenbar ist, wobei eine Zahl n als Wort $\{|^n\}$ kodiert wird.

Bemerkung

- Man kann Turing-Maschinen zusammensetzen und damit zeigen, dass eine Komposition von turingberechenbaren Funktionen wieder turingberechenbar sind.
- Ähnlich lassen sich Schleifen simulieren.
- So lässt sich zeigen, dass jede μ-partiellrekursive Funktion turingberechenbar ist.
- Wie für das Terminierungsproblem von Programmiersprachen zeigt man, dass das Halteproblem für Turingmaschinen, d.h. die Frage, ob eine Turingmaschine nach endlich vielen Schritten anhält, unentscheidbar ist.

Akzeptierte Sprache

Definition

- Eine Turingmaschine akzeptiert ein Wort w, wenn sie bei Eingabe von w nach endlich vielen Schritten hält.
- Die von einer Turingmaschine akzeptierte Sprache ist die Menge der von ihr akzeptierten Wörter.

Satz

Eine Sprache wird genau dann von einer Turingmaschine akzeptiert, wenn sie rekursiv aufzählbar ist.

Universelle Turingmaschine

- Codiere Turingmaschine T als Wort $[T]$ über $\{0,1\}$.
- Codiere Eingabe w als Wort $[w]$ über $\{0,1\}$.
- Codiere Paar (T,w) als Wort $[T,w]$ über $\{0,1\}$.
- Es gibt eine universelle Turingmaschine T_u, die bei Eingabe der Codierung von $[T,w]$ die Turingmaschine T bei Eingabe w simuliert.
- T_u akzeptiert die universelle Sprache

$$L_u = \{[T,w] \mid T \text{ akzeptiert } w\}$$

Probleme und Sprachen

- Probleme in der Informatik werden oft als ja/nein-Fragen gestellt. Einem solchen Problem entspricht dann eine Sprache und umgekehrt.

Beispiel

- Halteproblem: Gegeben eine Turingmaschine T und ein Eingabewort w. Hält T bei Eingabe w nach endlich vielen Schritten?
- Sprache: $L_u = \{[T, w] \mid T \text{ hält bei } w\}$

Beispiel

- Erfüllbarkeitsproblem: Gegeben eine Formel F. Ist F erfüllbar?
- Sprache: Menge der Codierungen erfüllbarer Formeln

Probleme und Sprachen

Beispiel

- Travelling Salesman-Problem: Gegeben ein Netz G von Städten und Straßen und eine Länge l. Gibt es eine Rundroute mit Länge $\leq l$, die jede Stadt besucht?
- Sprache: Menge der Codierungen von Paaren (G, l), sodass es eine solche Rundroute gibt.

Von Turingmaschinen akzeptierte Sprachen

Satz

Eine Sprache wird genau dann von einer Turingmaschine akzeptiert, wenn sie rekursiv aufzählbar ist.

Cantor'sches Diagonalverfahren

Lemma

- *Sei* $L := \{[T] \mid T \text{ hält bei Eingabe } [T] \text{ nicht}\}$
- *Dann gibt es keine Turingmaschine, die* L *akzeptiert.*

Beweis.

- Angenommen, T_0 würde L akzeptieren.
- Dann würde für alle $w \in \{0,1\}^*$ gelten:

$$T_0 \text{ akzeptiert } w \iff w \in L$$

- Dann

$$\begin{aligned} T_0 \text{ akzeptiert } [T_0] &\iff [T_0] \in L \\ &\iff T_0 \text{ akzeptiert } [T_0] \text{ nicht} \end{aligned}$$

- Widerspruch

Halteproblem

Satz

Das Halteproblem für Turingmaschinen ist nicht Turing-entscheidbar.

Church-Turing-These $\Rightarrow$ nicht entscheidbar.