Theoretische Informatik

Julian Schubert

30. Juni 2021

Inhaltsverzeichnis

1 Wichtige Vermutungen

Definition 1: Goldbachsche Vermutung

Jede natürliche gerade Zahl größer 2 ist Summe zweier Primzahlen

Definition 2: Collaz-Problem (3n +1)-Vermutung

- Beginne mit irgendeiner natürlichen Zahl $n > 0$
- Ist n gerade, so nimm als nächstes $n//2$ (abrundende Division)
- ist n ungerade, so nimm als nächstes $3n + 1$
- Wiederhole das Vorgehen mit der erhaltenen Zahl

Vermutung: Jede so konstruierte Zahlenfolge mündet in den Zyklus 4, 2, 1, egal mit welcher natürlichen zahl $n > 0$ beginnt

Definition 3: Ackermann-Funktion

Frage: Gilt LOOP $= \{f \in \text{WHILE} \mid f \text{ ist total}\}$?
Die folgende Funktion (auch **Ackermann-Funktion** genannt) $a : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ ist total und While-berechenbar, aber nicht Loop-berechenbar:

$$a(n,m) = \begin{cases} m+1 & \text{falls } n = 0 \\ a(n-1,1) & \text{falls } n > 0 \text{ und } m = 0 \\ a(n-1,a(n,m-1)) & \text{falls } n > 0 \text{ und } m > 0 \end{cases}$$

$\Rightarrow$ Die Ackermann-Funktion ist eine totale Funktion in $WHILE - LOOP$

Definition 4: Hauptsatz der Algorithmentheorie

$RAM = WHILE = MINIWHILE = TM$

Definition 5: Curch-Turing These

Auch: These von Church:
Turing-Berechenbarkeit erfasst den intitiven Begriff der Berechenbarkeit.

2 Elementare Begriffe

2.1 Komplexitätsklassen

$$ALL \subset P \subset NP$$

- **ALL:** Alle Probleme
- **NP:** Probleme, deren Lösungen schnell übrprüft weden können (effizient überprüfbare Probleme)
- **P:** Probleme, die isch in polynomieller Zeit lösen lassen (effizient lösbare Probleme)

2.2 Funktionen

Definition 6: Funktionen

Seien $f : A \to B$ und $g : B \to C$ Funktionen

- **Definitionsbereich** von f:
  $D_f = \{a \in A |$ es existiert ein $b \in B$ mit $f(a) = b\}$
  $\Rightarrow$ Alles was etwas im Wertebereich trifft
- **Wertebereich** von f:
  $D_f = \{a \in A |$ es existiert ein $a \in A$ mit $f(a) = b\}$
  $\Rightarrow$ alles was von etwas im Definitionsbereich getroffen wird
- **Total:** $D_f = A$
- **Surjektiv:** $W_f = B$
- **Injektiv:** aus $a_1, a_2 \in D$ und $a_1 \neq a_2$ folgt $f(a_1) \neq f(a_2)$
- **Bijektiv:** f ist total, surjektiv und injektiv
- ist f injektiv, so existiert die **Umkehrfunktion** $f^{-1} : B \to A$ mit $f^{-1}(b) =$ dasjenige $a \in A$ mit $f(a) = b$

2.3 Binärdarstellung

Definition 7

Jede natürliche Zahl $n \geqslant 1$ ist in genau einer Weise darstellbar als

$$n = \sum_{i=0}^{m} a_i \cdot 2^i$$

mit $m \in \mathbb{N}, a_m = 1$ und $a_0, \ldots, a_{m-1} \in \{0, 1\}$.

Eigenschaft 1: Binärdarstellung

$$bin(2n + a) = bin(n)a \text{ für } n \geqslant 1 \text{ und } a \in \{0, 1\}$$

2.4 Listencodierung

Liste von Binärzahlen: $\langle x_1, \ldots, x_n \rangle$
Anwendung: Bits verdoppeln, 10 alss Anfangs-, Trenn- und Enmarkierung
Beispiele:
$\langle\rangle = bin^{-1}(10) = 2$
$\langle 2 \rangle = bin^{-1}(10110010) = 178$
$\langle 5, 3, 2 \rangle = bin^{-1}(1011001110111110110010) = 2944946$

3 While-Programme

Definition 8: While-Berechenbarkeit

Eine Funktion ist dann **While-Berechenbar**, falls es ein While-Programm gibt, sodass der Definitionsbereich von beiden identisch ist und der Wert für alle Eingaben übereinstimmt.

Definition 9: Loop-Programm

ein **Loop-Programm** ist ein While-Programm mit folgenden Eigenschaften:

- Das Programm enthält keine While-Schleifen
- Aus einer Funktion können nur weiter oben deklarierte Funktionen aufgerufen werden. Insbesondere sind keine Selbstaufrufe erlaubt
- Das Programm enhält nur Funktionsdeklarationen mit Initialiserung
- Das Programm ist für alle Eingaben definiert

$\Rightarrow$ Alle Loop-berechenbaren Funktionen sind total.

3.1 Berechnende Funktion bestimmen

1. Schauen für welche Eingabe(n) die Schleife(n) wie oft ausgeführt werden
2. Schauen was sich mit jedem Schleifendurchlauf verändert

4 Ram-Programme

Definition 10: modifizierte Differenz

$$x \dot{-} y = md(x, y) \begin{cases} x - y & \text{falls } x > y \\ 0 & \text{sonst} \end{cases}$$

5 Alphabete und Wörter

Definition 11: Alphabete und Wörter

- Ein **Alphabet** ist eine endliche, nichtleere Menge
- Die Elemente eines Alphabets werden **Buchstaben** oder **Symbole** genannt
- Ein **Wort über einem Alphabet** Σ ist eine endliche Folge von 0 oder mehr Elementen aus Σ
- das **leere Wort** (d.h. das Wort, das aus 0 Buchstaben) besteht bezeichnen wir mit ε

Definition 12: Mengen von Wörtern

Sei Σ ein Alphabet, $n \geqslant 0$ und $a_1, a_2, \dots a_n \in \Sigma$

- Die **Länge eines Wortes** w $a_1 a_2 \dots a_n$ ist $|w| = n$
- **Menge aller Wörter mit Länge n:**
  $\Sigma^n = \{w | w \text{ ist ein Wort über } \Sigma \text{ mit } |w| = n\}$
  Es gilt $\Sigma^0 = \{\varepsilon\}$
- **Menge aller Wörter:**
  $\Sigma^* = \{w | w \text{ ist ein Wort über } \Sigma\} = \bigcup_{u \geqslant 0} \Sigma^n$ und $\Sigma^+ = \Sigma^* \backslash \{\varepsilon\}$
- eine **formale Sprache über** Σ ist eine Teilmenge von Σ^*
- Das **Entscheidungsproblem** einer formalen Sprache $L \subseteq \Sigma^*$ ist folgende Aufagbe:
  Eingabe: $w \in \Sigma^*$
  Ausgabe:
  1, falls $w \in L$
  0, falls $w \notin L$

Definition 13: Dyadische dartstellung

dya: $\mathbb{N} \to \{1,2\}^*$ ist definiert durch

- dya(0) $= \varepsilon$
- day(n) $= a_m \dots a_0$ falls $n \geqslant 1, n = \sum_{i=0}^{m} a_i \cdot 2^i$ und $a_0, \dots, a_m \in \{1,2\}$

Eigenschaft 2: k-adische Darstellung

Sei $k \geqslant 2$

1. $\mathrm{ad}_k(\mathrm{kn} + \mathrm{a}) = \mathrm{ad}_k(\mathrm{n})\mathrm{a}$ für $n \geqslant 0$ und $a \in \{1, \dots k\}$
2. $\mathrm{ad}_k{}^{-1}(\mathrm{xa}) = \mathrm{k} \cdot \mathrm{ad}_k{}^{-1}(\mathrm{x}) + \mathrm{a}$ für $x \in \{1, \dots, k\}^*, a \in \{1, \dots, k\}$

6 Turing-Maschinen

Definition 14: Turing Maschiene

Sei $k \geqslant 1$. Eine **k-Band-Turing-Maschine** ist ein Quintupel (Σ, Z, f, z_0, z_1) mit

- Σ ist eine endliche Menge (Alphabet)
- Z ist eine endliche Menge (Zustandsmenge)
- $f(Z \backslash \{z_1\}) \times \Sigma^k \to Z \times \Sigma^k \times \{L, O, R\}^k$ ist eine totale Funktion (Überführungsfunktion)
- $z_0 \in Z$ (Startzustand)
- $z_1 \in Z$ (Stoppzustand)

$M(z, a_1 \dots a_m)$: Wort das auf Band 1 steht, alle anderen Bänder leer, und $a \in \Sigma \backslash \{\text{Leersymbol}\}$

Definition 15: Palindrom

Ein wort $a_1 \dots a_n$ heißt symmetrisch oder auch **Palindrom**, falls $a_1 \dots a_n = a_n \dots a_1$

7 Laufzeit von Algorithmen

Definition 16: Länge einer Zahl

$$|x| = |dya(abs(x))|$$

8 Entscheidbarkeit und Aufzählbarkeit

Definition 17: Entscheidungsalgorithmus

Entscheidungsalgorithmus für eine Menge A:

$$\text{Eingabe x} \Rightarrow \text{Ausgabe} \begin{cases} 1 \text{ (ja)}, & \text{falls } x \in A \\ 0 \text{ (nein)}, & \text{falls } x \notin A \end{cases}$$

Dies ist die berechnung der **charakteristischen Funktion** von A ($c_A(x)$).

Semicharakteristische Funktion:
Wie characteristische Funktion, nur n.d. falls $x \notin A$ ($\chi_A(x)$)

Definition 18: Entscheidbarkeit

Seien $n \geqslant 0$ und $t : \mathbb{N} \to \mathbb{N}$ eine totale Funktion:

- $A \subseteq \mathbb{N}^n$ heißt **entscheidbar** $\Leftrightarrow$ c_A ist berechenbar
- $A \subseteq \mathbb{N}^n$ heißt **semientscheidbar** $\Leftrightarrow$ $\chi_A(x)$ ist berechenbar
- **REC** $= \{A | \exists n \geqslant 0$ mit $A \subseteq \mathbb{N}^n$ und A ist entscheidbar$\}$ (recursive languages), also alle berechenbaren Mengen
- Ein Algorithmus **M entscheidet A** $\subseteq \mathbb{N}^n$ **in der Zeit t (bzw. O(t))** $\Leftrightarrow$ M berechnet c_A in der Zeit t (bzw. O(t))

Eigenschaft 3

Fär $A \subseteq \mathbb{N}^n$ gilt:

A entscheidbar $\Leftrightarrow$ A und $\bar{A}$ semientscheidbar
A entscheidbar $\Leftrightarrow$ A und $\bar{A}$ aufzählbar
A aufzählbar $\Leftrightarrow$ $B \subseteq \mathbb{N}^n \times \mathbb{N}$ mit $A = Pr(B)$

Definition 19: Aufzählbarkeit

$A \subseteq \mathbb{N}^n$ mit $n \geqslant 0$ heißt **rekursiv aufzählbar** (kurz: aufzählbar) $\Leftrightarrow$ $A = \varnothing$ oder es gibt ein berechenbares, totales $f : \mathbb{N} \to \mathbb{N}^n$ mit $W_f = A$
RE Alle Mengen die Aufzählbar sind

Eigenschaft 4

Für $m, n \geqslant 0$ gilt:
$f : \mathbb{N}^m \to \mathbb{N}^n$ berechenbar, total $\Rightarrow W_F$ ist aufzählbar

Eigenschaft 5

Für A $\subseteq \mathbb{N}^n$ sind folgende Aussagen äquivalent

1. A ist aufzählbar
2. A ist semientscheidbar
3. A ist Definitionsbereich einer berechenbaren Funktion $f : \mathbb{N}^n \to \mathbb{N}^m$ mit $m \geqslant 0$
4. A ist Wertebereich einer berechenbaren Funktion $g : \mathbb{N}^m \to \mathbb{N}^n$ mit $m \geqslant 0$

Definition 20: Projektion

Die **Projektion** einer Menge
$B \subseteq \mathbb{N}^n \times \mathbb{N}$ ist definiert als $Pr(B) = \{x \in \mathbb{N}^n | \exists y \in \mathbb{N}[(x, y) \in B]\}$

Definition 21: Reduzierbarkeit

Seien $A \subseteq \mathbb{N}^n$ und $B \subseteq \mathbb{N}^n$.
A ist reduzierbar auf B $\Leftrightarrow$ es gibt ein totales, berechenbares $f : \mathbb{N}^m \to \mathbb{N}^n$ sodass für alle $x \in \mathbb{N}^m$ gilt:

$$x \in A \Leftrightarrow f(x) \in B$$

Die Äquivalenz ist gleichbedeutend mit den Aussagen $c_A = c_b \circ f$ und $\chi_A = \chi_B \circ f$

Eigenschaft 6

Seien $A \subseteq \mathbb{N}^m$ und $B \subseteq \mathbb{N}^n$. Falls A reduzierbar auf B ist, so gelten folgende Implikationen:
$B \in REC \Rightarrow B \in REC$
$B \in RE \Rightarrow A \in RE$

Definition 22: Gödelisierung

Skritp ab Seite 172, Rams werden als Liste codiert.

Definition 23: Halteproblem

$K_0 = \{x | M_x$ hält bei Eingabe x$\}$ **spezielles Halteproblem**
$K = \{(x, y) | M_x$ hält bei Eingabe y$\}$ **allgemeines Halteproblem**
$\Rightarrow$ wir geben der Maschiene ihren eigenen Quellcode als Eingabe
K_0 ist aufzählbar, aber nicht entscheidbar

Definition 24: Satz von Rice

Die Frage, ob die von einem gegebenen Quelltext berechnete Funktion eine Eigenschaft S hat, lässt sich nicht Algorithmisch lösen

Definition 25

Seien $\mathbb{G}$ eine Grundmenge, $A \subseteq \mathbb{G}$ und $t : \mathbb{N} \to \mathbb{N}$ eine totale Funktion

- A heißt **entscheidbar** $\Leftrightarrow c_A : \mathbb{G} \to \{0, 1\}$ ist berechenbar
- A heißt **semientscheidbar** $\Leftrightarrow \chi_A : \mathbb{G} \to \{0, 1\}$ ist berechenbar
- A heißt **rekursiv aufzählbar** (kurz: aufzählbar) $\Leftrightarrow A = \varnothing$ oder es gibt ein berechenbares, totales $f : \mathbb{N} \to \mathbb{G}$ mit $W_f = A$

9 Endliche Automaten

Definition 26: Deterministischer endlicher Automat

Ein **deterministischer endlicher Automat (DEA)** ist ein Quintupel

$(\Sigma, Z, \delta, z_0, F)$ mit folgenden Eigenschaften:

- Σ ist eine endliche, nichtleere Menge (Eingabealphabet)
- Z ist eine endliche Menge (Zustandsmenge)
- δ ist eine totale Funktion $Z \times \Sigma \to Z$ (Überführungsfunktion)
- $z_0 \in Z$ (Startzustand)
- $F \subseteq Z$ (Menge der akzeptierenden Zustände)

Definition 27: Erweiterte Überführungsfunktion

Die **erweiterte Überführungsfunktion** eines DEA A $= (\Sigma, Z, \delta, z_0, F)$ ist die wie folg definierte Abbildung $\bar{\delta} : Z \times \Sigma^* \to Z$.
(IA) $\bar{\delta}(z, \epsilon) = z$ für alle $z \in Z$
(IS) $\bar{\delta} = \delta(\bar{\delta}(z, w), a)$ für alle $z \in Z, w \in \Sigma^*, a \in \Sigma$

Damit gilt: $\bar{\delta}(z, w) =$ Zustand, den der DEA erreicht wenn er in z startet und das Wort w einliest.

Definition 28: Akzeptierung von Sprachen durch DEAs

- Ein wort $w \in \Sigma^*$ heißt **von A akzeptiert** $\Leftrightarrow \bar{\delta}(z_0, w) \in F$
- Die von **A akzeptierte Sprache** ist

$$\mathbf{L(A)} = \{w \in \Sigma^* | \text{ w wird von A akzeptiert}\}$$

- Die **Menge der von DEAs akzeptierten Sprachen ist**

$$\mathbf{EA} = \{L(A) | \text{ A ist ein DEA}\}$$

$\Rightarrow$ eine $L \in EA$ sind **entscheidbar**

10 Nichtdeterministische endliche Automaten

Unterschied DEA:
δ ist eine totale Funktion $Z \times \Sigma \to P(Z)$, also eine Abbildung auf die Potenzmenge.
Kann in mehreren Zuständen gleichzeitig sein.

Erweiterte Überführungsfunktion: $\bar{\delta}(z, w) =$ Menge der Zustände, die der NEA gleichzeitig erreicht, wenn er in z startet und das Wort w einliest.

Die von NEAS akzeptierte Sprache heißt **L(A)**

Definition 29: Potenzmengenkonstruktion

Aus DEA einen NEA machen

Definition 30: Konkatenation von Sprachen

Seien L, L' $\subseteq \Sigma^*$

$$\begin{aligned} L \cdot L' &= \{uv | u \in L \text{ und } v \in L'\} \\ L^0 &= \{\epsilon\} \\ L^{k+1} &= L \cdot L^k \text{ für } k \geqslant 0 \\ L^* &= \bigcup_{k \geqslant 0} L^k = \{u_1 u_2 \ldots u_m | m \geqslant 0 \text{ und } u_1, \ldots u_m \in L\} \end{aligned}$$

Definition 31: Abschlusseigenschaften von EA

$$L, L' \in EA \Rightarrow \bar{L}, L \cup L', L \cap L', L \cdot L', L^* \in EA$$

11 Reguläre Ausdrücke

Definition 32: Syntax und Semantik regulärer Ausdrücke

Sei Σ ein Alphabet. Wir definieren **reguläre Ausdrücke** γ und die durch sie beschriebenen Sprachen $L(\gamma)$.

- (IA) $\varnothing, \epsilon$ und **a** sind reguläre Ausdrücke (wobei $a \in \Sigma$).
  Semantik: $L(\varnothing) = \varnothing, L(\epsilon) = \{\epsilon\}, L(a) = \{a\}$
- (IS) sind α, β reguläre ausdrücke, so auch $(\alpha + \beta), (\alpha \cdot \beta), \alpha^*$
  Semantik:
  $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
  $L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$
  $L(\alpha^*) = L(\alpha)^*$

1. Unnötige Klammern un $\cdot$ können weggelassen werden

2. Bindungsreihensfolge: * -> $\cdot$ -> +
   $0 + 01^*$ steht also für $(0 + (0 \cdot (1^*)))$

3. Bezeichnung reg. Ausdrücke durch griechische Kleinbuchstaben

Definition 33: Reguläre Sprachen

- Eine Sprache L heißt **regulär** $\Leftrightarrow$ es existiert ein regulärer Ausdruck α mit $L = L(\alpha)$
- **REG** $= \{L|L \text{ ist regulär}\}$

Eigenschaft 7

Seien $L, A, B \subseteq \Sigma^*$ mit $\epsilon \notin A$. Falls $L = A \cdot L \cup B$, so gilt $L = A^*B$

12 Pumping-Lemma

Eigenschaft 8

Für jede reguläre Sprache L existiert ein $n \in \mathbb{N}^+$ mit folgender Eigenschaft: Jedes Wort $w \in L$ mit $|w| \geqslant n$ lässt sich zerlegen in $w = xyz$, sodass $y \neq \epsilon, |xy| \leqslant n$ und $\forall_{i \geqslant 0} xy^i z \in L$

Definition 34: Äquivalenz von Zuständen

Sei $A = (\Sigma, Z, \delta, z_0, F)$ ein DEA. Zwei Zustände $z_1, z_2 \in Z$ heißen **äquivalent**, falls für alle $w \in \Sigma^*$ gilt:

$$\bar{\delta}(z_1, w) \in F \Leftrightarrow \bar{\delta}(z_2, w) \in F$$

Definition 35: unterscheidbare Zustandspaare

Sei $A = (\Sigma, Z, \delta, z_0, F)$ ein DEA.

- (IA) falls $z_1 \in F \Leftrightarrow z_2 \in F$ so sind z_1 und z_2 **unterscheidbar**
- (IS) Sind $a \in \Sigma$, $\delta(z_1, a) = z_1', \delta(z_2, a) = z_2'$ sowie z_1' und z_2' unter-

scheidbare Zustände, so sind z_1 und z_2 **unterscheidbar**

Eigenschaft 9

Sei $A = (\Sigma, Z, \delta, z_0, F)$ ein DEA. Sind zwei Zustände $z_1, z_2 \in Z$ nicht unterscheidbar $\Leftrightarrow$ so sind sie äquivalent

Definition 36: Äquivalente DEA

EquivalentDEA $= \{(A_1, A_2)|A_1, A_2$ sind DEAS mit gleichem Eingabealphabet und $L(A_1) = L(A_2)\}$

1. Wir fassen A_1 und A_2 als einen DEA auf, indem wir beide Automaten nebeneinander zeichen und als Startzustand den von A_1 wählen
2. Bestimme die unterscheidbaren Zustände mit Hilfe des Algorithmus auf seite 234
3. A_1 und A_2 sind genau dann äquivalent, wenn die beiden Start- zustände nicht unterscheidbar sind

Eigenschaft 10

Sei $A = (\Sigma, Z, \delta, z_0, F)$ ein DEA. Die Äquivalenz von Zuständen ist eine Äquivalenzrelation auf Z, d.h. es gilt:

1. Jeder Zustand ist sich selbst äquivalent (Reflexivität)
2. Ist p äquivalent zu q so auf q zu p (Symmetrie)
3. Ist p äquivalent zu q und q äquivalent zu r, so auf p zu r (Transitivität)

Definition 37

Es ist entscheidbar, ob ein DEA die leere Sprache bzw. eine endliche Sprache akzeptiert.
EmptyDEA $= \{A|A$ ist ein DEA mit $L(A) = \varnothing\}$
FiniteDEA $= \{A|A$ ist ein DEA mit und $L(A)$ ist endlich$\}$
EmptyDEA und FiniteDEA sind entscheidbar.

13 Formale Sprachen

Definition 38: generative Grammatik

Eine **generative Grammatik** ist ein Quadrupel $G = (\Sigma, N, S, R)$ mit folgenden Eigenschaften

- Σ ist eine enldiche, nichtleere Menge (Terminalsymbole)
- N ist eine endliche, nichtleere Menge mit $\Sigma \cap N = \varnothing$ (Nichtterminalsymbole)
- $S \in N$ (Startsymbol)
- $R \subseteq (\Sigma \cup N)^+ \times (\Sigma \cup N)^*$ ist eine endliche Menge (Menge der Erzeugungsregeln)
  Für $(v, w) \in R$ schreiben wir auch $v \to w$

Konvention:

- Terminale = Kleinbuchstaben
- Nichtterminale = Großbuchstaben

Definition 39: Erzeugung von Sprachen durch generative Grammatik

Seien $G = (\Sigma, N, S, R)$ eine Grammatik, $v, w \in (\Sigma \cup N)^*$ und $t \geqslant 0$

- $v \underset{G}{\Rightarrow} w \Leftrightarrow$ es existieren u_1, u_2, x, y mit $(x, y) \in R, v = u_1 x u_2$ und $w = u_1 y u_2$
  (G erzeugt w aus v in einem Schritt)
- $v \overset{t}{\underset{G}{\Rightarrow}} w \Leftrightarrow$ es existieren $w_0, \ldots, w_t \in (\Sigma \cup N)^*$ mit
  $v = w_0 \underset{G}{\Rightarrow} w_1 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} w_t = w$
  (G erzeugt w aus v in t Schritten)
- $v \overset{*}{\underset{G}{\Rightarrow}} w \Leftrightarrow$ es existiert $t' \geqslant 0$ mit $v \overset{*}{\underset{G}{\Rightarrow}} w$
  (G erzeugt w aus v)
- $L(G) = \{z \in \Sigma^* | S \overset{*}{\underset{G}{\Rightarrow}} z\}$
  (Die von G erzeugte Sprache)

Beachte:

- $w \overset{*}{\underset{G}{\Rightarrow}} w$ gilt für alle $w \in (\Sigma \cup N)^*$
- L(G) besteht nur aus Terminalsymbolwörtern
- Nichtterminalsymbole sind Hilfszeiczhen, die für die Erzeugung eines $v \in \Sigma^*$ benötigt werden. Bis zum Ende der Erzeugung von v müssen alle Nichtterminalsymbole eliminiert sein

Definition 40: Äquivalenz von Grammatiken

Zei Grammatiken G und G' heißen äquivalent $\Leftrightarrow$ L(G) = L(G')

Definition 41: Typen von Grammatik

Sei $G = (\Sigma, N, S, R)$ eine Grammatik

- G heißt **Grammatik vom Typ 0**
- G heißt **Grammatik vom Typ 1** oder **kontextsensitive Grammatik** $\Leftrightarrow$ jede Regel hat die Form $u_1 A u_2 \to u_1 w u_2$
mit $A \in N, u_1, u_2, w \in (\Sigma \cup N)^*$ und $w \neq \epsilon$
- G heißt **Grammatik vom Typ 2** oder **kontextfreie Grammatik** $\Leftrightarrow$ jede Regel hat die Form $A \to w$ mit $A \in N, w \in (\Sigma \cup N)^*$ und $w \neq \epsilon$
(Ersetzung von A durch w ohne Beachtung des Kontextes)
Nimmt man das ϵ hinzu führt das nicht aus der Klasse hinaus, ist formal dann jedoch keine kontextfreie sprache mehr.
- G heißt **Grammatik vom Typ 3** oder **rechtslineare Grammatik** $\Leftrightarrow$ jede Regel hat die Form $A \to aB$ oder $A \to a$ mit $A, B \in N$ und $a \in \Sigma$

Definition 42

Seien $L \subset \Sigma^*$ und $i \in \{0, 1, 2, 3\}$

- L heißt **Sprache vom Typ i** $\Leftrightarrow$ es existiert eine Grammatik G vom Typ i mit $L = L(G)$ oder $L = L(G) \cup \{\epsilon\}$
- L heißt **kontextsensitiv** $\Leftrightarrow$ L ist vom Typ 1
- L heißt **kontextfrei** $\Leftrightarrow$ L ist vom Typ 2

- $L_i = \{L |$ L ist vom Typ i$\}$
- Die **Chomsky-Hiererachie** besteht aus L_0, L_1, L_2, L_3

Es gilt:

$$L_3 \subsetneq L_2 \subsetneq L_1 \subsetneq L_0$$

Eigenschaft 11

Für $L \subseteq \Sigma^*$ sind folgende Aussagen äquivalent:

1. L ist Regulär
2. L ist vom Typ 3

Definition 43: Chomsky-Normalform

Eine Grammatik $G = (\Sigma, N, S, R)$ ist in **Comsky-Normalform** $\Leftrightarrow$ G hat nur Regelnd der Form $A \to BC$ oder $A \to a$ mit $A, B, C \in N$ und $a \in \Sigma$
Jede kontextfreie Grammatik besitzt eine äquivalente Grammatik in Chomsky-Normalform.

Umwandlen einer kontextfreien Grammatik in Chomsky-Normalform

1. G_1 entsteht aus G wie folgt
   - wähle neues Nichtterminal D_a für jedes $a \in \Sigma$
   - ersetze in allen Regeln a durch D_a
   - füge neue Regel $D_a \to a$ hinzu
2. G_2 entsteht aus G_1 wie folgt
   - wenn $A_1 \underset{G}{\Rightarrow} A_2 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} A_k \underset{G}{\Rightarrow} a$ mit $k \geqslant 2$, so füge $A_1 \to a$ hinzu
   - wenn $A_1 \underset{G}{\Rightarrow} A_2 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} B_1 \ldots B_m$ mit $m, k \geqslant 2$ so füge $A_1 \to B_1 \ldots B_m$ hinzu
   - entferne alle Regeln der Form $A \to B$
3. G_3 entsteht aus G_2 wie folgt
   - ersetze $A \to B_1 \ldots B_m$ mit $m \geqslant 3$ durch $A \to B_1 E_2, E_2 \to B_2 E_3, \ldots, E_{m-1} \to B_{m-1} B_m$ (wobei E_i neue Nichtterminale sind)

Definition 44: Pumping-Lemma für kontextfreie Sprachen

Für jede kontextfreie Sprache L existiert ein $n \in \mathbb{N}^+$ mit folgender Eigenschaft:
Jedes Wort $z \in L$ mit $|z| \geqslant n$ lässt sich zerlegen in $z = uvwxy$, sodass $vx \neq \epsilon$, $|vwx| \leqslant n$ und $\forall_{i \geqslant 0} uv^i wx^i y \in L$

Eigenschaft 12

Im Gegensatz zu L_3 ist L_2 nicht mehr unter Durchschnitt und Komplenent abgeschlossen.

1. L_2 ist abgeschlossen unter $\cup, \cdot, *$
2. L_2 ist nicht abgeschlossen unter $\cap$ bzw Komplement

Eigenschaft 13

Jede kontextfreie Sprache kann durch ein Python-Programm in der Zeit $O(n^3)$ entschieden werden

Definition 45: Akzeptierung Nichtdeterministischer Algorithmus

Sei M ein Nichtdeterministischer Algorithmus

- **M akzeptiert x** $\Leftrightarrow$ es gibt einen Rechenweg von M bei eingabe x, der den Wert 1 berechnet
- Die **von M akzeptierte Menge ist**

$$\mathbf{L(M)} = \{x | M \text{ akzeptiert } x\}$$

14 Kellerautomaten

Definition 46: nichtdeterministischer Kellerautomat

Ein **nichtdeterministischer Kellerautomat** ist ein Quintupel $M = (\Sigma, \Delta, Z, f, z_0)$ mit folgenden Eigenschaften

- Σ ist endliche, nichtleere Menge mit $\square \notin \Sigma$ (Eingabealphabet)

- Δ ist endliche, nichtleere Menge mit $\square \notin \Delta$ (Kelleralphabet)
- Z ist eine enliche Menge (Zustandsmenge)
- f ist eine totale Funktion $Z \times (\Sigma \cup \{\epsilon\}) \times \Delta \to P_{fin}(Z \times \Delta^*)$ (Überführungsfunktion)
  - $P_{fin}(A) = \{B \subseteq A | B \text{ ist endlich } \}$ = Menge der endlichen Teilmengen von A
- $z_0 \in Z$ (Startzustand)

Schreibweise: statt $(z', u) \in f(z, a, A)$ schreiben wir auch $zaA \to z'u$
Akzeptierung von Wörtern:
Ein Kellerautomat akzeptiert ein Wort falls am Ende eines Rechenweges der Keller leer ist (und das gesamte Eingabewort gelesen wurde)
L(M) $= \{w \in \Sigma^* |$ M akzeptiert w$\}$

Möglichkeiten für Nichtakzeptierung

- Keller leer bevor Eingabeende erreicht (Stopp)
- kein Befehl für aktzellen Zustand / Eingabesymbol / Topsymbol vorhanden (Stopp)
- Einabeende nicht erreicht und nor folgen von ϵ-Befehelen möglich, d.h. keine Möglichkeit das nächste Zeichen zu lesen (Endlosschleife)
- Eingabeende zwar erreicht, aber keine Möglichkeit den Keller zu leeren (Stopp oder Endlosschleife)

Eigenschaft 14

Für $L \subseteq \Sigma^*$ sind folgende Aussagen äquivalent:

1. L ist kontextfrei
2. L wird von einem Nichtdeterministischen Kellerautomaten akzeptiert.

Definition 47: Nichtverkürzende Grammatik

Eine Grammatik von Typ 0 heißt **nichtverkürzend**, falls sie nur Regeln der Form $v \to w$ mit $|v| \leqslant |w|$ besitzt

Eigenschaft 15

Für $L \in \Sigma^*$ sind folgende Aussagen äquivalent

1. L kann von einer kontextsensitiven Grammatik erzeugt werden
2. L kann von einer nichtverkürzenden Grammatik erzeugt werden

Definition 48: Linear beschränkter Automat

Ein **linear beschränkter Automat (LBA)** ist eine Nichtdeterministische 1-Band-TM, die während ihrer Arbeit nur die vom Eingabewort belegten Fekder und deren Nachbarfelder benutzt

Eigenschaft 16

Für $L \subseteq \Sigma^*$ sind folgende Aussagen äquivalent:

1. L ist kontextsensitiv
2. L kann von einem LBA akzeptiert werden

Eigenschaft 17

Jede kontextsensitive Sprache ist in der Zeit $2^{O(n)}$ entscheidbar

Eigenschaft 18

L_1 ist abgeschlossen unter $\cup, \cap, \cdot, *$ und Komplement

Eigenschaft 19

Für $L \subseteq \Sigma^*$ und $k \geqslant 1$ sind folgende Aussagen äquivalent:

1. L ist vom Typ 0
2. L wird von einer Nichtdeterministischen k-Band-TM akzeptiert
3. L ist aufzählbar

Eigenschaft 20

Abschlusseigenschaften der Klasse L_0

- L_0 ist abgeschossen unter $\cup, \cap, \cdot$ und $*$
- L_0 ist nicht abgeschlossen unter komplement

15 Komplexität

Definition 49: Polynomialzeit und Exponentialzeit

Sei M ein Algorithmus und f eine Funktion

- M berechent f in **Polynomialzeit**, falls es ein Polynom $p(n) = \sum_{i=0}^{m} a_i n^i$ mit $m, a_0, a_1, \ldots, a_m \in \mathbb{N}$ gibt, sodass f von M in der Zeit p berechnet wird.
- M berechent f in **Exponentialzeit**, falls es ein Polynom $p(n) = \sum_{i=0}^{m} a_i n^i$ mit $m, a_0, a_1, \ldots, a_m \in \mathbb{N}$ gibt, sodass f von M in der Zeit 2^p berechnet wird.

Wir können uns auf Polynome der Form $p(n) = n^k + k$ mit $k \in \mathbb{N}$ beschränken.

Definition 50: Wichtige Komplexitätsklassen

- **FP** $= \{f | f : \mathbb{N} \to \mathbb{N}$ ist eine totale Funktion, die in Polynomialzeit berechenbar ist
- **P** $= \{A | A \subseteq \mathbb{N}^n$ ist eine Menge, die in Polynomialzeit entscheidbar ist $\}$
- **EXP** $= \{A | A \subseteq \mathbb{N}^n$ ist eine Menge, die in Exponentialzeit entscheidbar ist $\}$

  Für jedes $A \subseteq \mathbb{N}^n$ gilt

$$A \in P \Leftrightarrow c_a \in FP$$

Wir betrachten FP und P als die klassen der effizient berechenbaren Funktionen bzw. der effizient lösbaren Probleme

Eigenschaft 21: Terminologie

- Problem = Menge
- Problem A lösen = Menge A entscheiden = Berechnung von c_a
- Komplexitätsklasse = Klasse von Funktionen oder Mengen, die durch eine vorgegebene, maximale Berechnungskomplexität definiert ist

Eigenschaft 22

1. Kontextfreie Sprachen sind in Polynomialzeit entscheidbar
2. kontextsensitive Sprachen sind in Exponentialzeit entscheidbar
3. Es gibt Sprachen aus $L_1 - L_2$ die in Polynomialzeit entscheidbar sind

Eigenschaft 23

Ist $f \in FP$, so existiert ein Polynom p mit

- f ist in der Zeit p berechenbar und
- $|f(x)| \leqslant p(|x|)$