Informatik III

Ein Aufschrieb der Vorlesung *Informatik III* an der Uni Karlsruhe in den Wintersemestern 1997/98 bzw. 1999/00, gelesen von Prof. Dr. H. Prautzsch.

GeTEXt von Tobias Dussa (s_dussa@ira.uka.de), überarbeitet von Andreas Klöckner (ak@ixion.net). Kommentare und Berichtigungen gehen an eine dieser Adressen.

LATEX-Lauf am 23. Februar 2006.

Inhaltsverzeichnis

Teil I

Formale Sprachen

1 Grammatiken und Sprachen

1.1 Grundbegriffe

Definition 1.1 Alphabet

Ein Alphabet ist eine endliche nichtleere Menge V von Zeichen.

Definition 1.2 Wort

Ein Wort oder eine Zeichenkette w über einem Alphabet V ist eine geordnete Menge $w = a_1 \dots a_n$ mit $a_i \in V$. Worte werden gemeinhin mit Kleinbuchstaben bezeichnet.

Die Menge aller Zeichenketten schreibt man V^*, die Menge aller nichtleeren Zeichenketten $V^+ := V^* \setminus \{\lambda\}$.

Ein Wort v ist genau dann ein Teilwort des Wortes w, wenn es $b, c \in V^*$ gibt mit $w = bvc$.

Definition 1.3 Sprache

Eine Sprache L ist eine Teilmenge $L \subseteq V^*$.

Beispiel Sprache und Alphabet

Eine Sprache über dem Alphabet $V := \{0, 1\}$ ist z.B. $L := \{0^n \mid n \in \mathbb{N}_0\}$.

Definition 1.4 Produktion

Eine Produktion oder Ersetzungsregel ist ein Zweitupel $(\alpha, \beta) \in V^* \times V^*$, auch geschrieben $\alpha \to \beta$. Ein Ersetzungssystem oder Semi-Thue-System (V, P) ist ein geordnetes Paar, bestehend aus einem Alphabet V und einer Produktionsmenge P.

Definition 1.5 Ableitung

Seien nun $x, y \in V^*$. y heißt direkt ableitbar aus x (Schreibweise: $x \Rightarrow y$) $:\Leftrightarrow$

$$(\exists(\alpha, \beta) \in P)(\exists u, v \in V^*)(x = u\alpha v \wedge y = u\beta v)$$

y heißt (indirekt) ableitbar aus x (Schreibweise: $x \Rightarrow^* y$) :$\Leftrightarrow$

$$(\exists x_1, \ldots, x_n \in V^*)(x \Rightarrow x_1 \Rightarrow \cdots \Rightarrow x_n \Rightarrow y)$$

Bemerkung

Das Problem festzustellen, ob $x \Rightarrow^* y$ für gegebene x, y gilt, ist im allgemeinen unentscheidbar.

1.2 Grammatiken

Definition 1.6 Grammatik

Ein Viertupel $G = (V, T, S, P)$ mit den Alphabeten V und T sowie dem Startsymbol S und der Produktionsmenge P heißt eine Grammatik, falls gilt

(1) $V \cap T = \emptyset$

(2) $S \in V$

(3) $|P| < \infty$ und für P gilt

$$(\forall p = u \to v \in P)(u, v \in (V \cup T)^+, u \cap V \neq \emptyset)$$

T heißt das Terminalalphabet, V das Variablenalphabet oder Nichtterminalalphabet. Ein beliebiges Element $q \in T$ heißt Terminalzeichen, ein beliebiges Element $r \in V$ heißt Variable oder Nichtterminal.

Sei nun $G = (V, T, S, P)$ eine Grammatik. Dann ist $L(G)$ die von G erzeugte Sprache, definiert durch

$$L(G) := \{w \in T^* \mid S \Rightarrow^* w\}$$

Worte über Grammatiken sind Elemente aus T^*, während für eine Satzform u über G ein $u \in (V \cup T)^*$ gilt.

Beispiel Grammatik

Sei $V = \{S, A, B\}$ und $T = \{0, 1\}$. Die Menge P der Produktionen sei

$$P := \{S \to 0B|1A, A \to 0|0S|1AA, B \to 1|1S|0BB\}$$

Dann gilt

$$L(G) = \{w \in \{0, 1\}^+ \mid \text{Anzahl der Nullen gleich Anzahl der Einsen}\}$$

Definition 1.7 Äquivalenz von Grammatiken

Offensichtlich kann man für ein- und dieselbe Sprache mehrere Grammatiken angeben. Zwei Grammatiken G_1, G_2 heißen äquivalent :$\Leftrightarrow$ $L(G_1) = L(G_2)$.

1.3 Die Chomsky-Hierarchie

Definition 1.8 Chomsky-Klasse

Sei $G = (V, T, S, P)$ eine Grammatik. Dann heißt G

- kontextsensitiv :$\Leftrightarrow$ die Produktionen haben die Form
  - $yAz \to ywz$ mit $A \in V, w \neq \lambda, y, w, z \in (V \cup T)^*$
  - $S \to \lambda$, aber nur, wenn S auf keiner rechten Seite einer Produktionsregel vorkommt.

  Dieser Grammatiktyp zeichnet sich dadurch aus, daß Worte niemals kürzer werden (außer bei der trivialen Produktionsregel).
- kontextfrei :$\Leftrightarrow$ die Produktionen haben die Form $A \to w$ mit $A \in V, w \in (V \cup T)^*$.
- rechtslinear :$\Leftrightarrow$ die Produktionen haben die Form
  - $A \to bC$ mit $A, C \in V, b \in T$
  - $A \to C$ mit $A, C \in V$

  Die Linkslinearität wird analog definiert.

Einer Grammatik wird nun auch eine Typnummer zugeordnet: G ist vom Typ

- 0 :$\Leftrightarrow$ die Produktionen sind beliebig
- 1 :$\Leftrightarrow$ sie ist kontextsensitiv (kts)
- 2 :$\Leftrightarrow$ sie ist kontextfrei (ktf)
- 3 :$\Leftrightarrow$ sie ist rechts- oder linkslinear

Sei L_i die Klasse aller Grammatiken des Typs i. Dann gilt:

$$L_0 \supsetneqq L_1 \supsetneqq L_2 \supsetneqq L_3$$

Eine Sprache heißt vom Typ i :$\Leftrightarrow$ sie wird durch eine Grammatik des Typs i erzeugt.

1.4 Abgeschlossenheit

Definition 1.9 Operationen auf Sprachen

Seien L, M Sprachen über V. Dann definiert man

- Vereinigung: $L \cup M$ auf naheliegende Art und Weise.

- Schnitt: $L \cap M$ auf naheliegende Art und Weise.
- Konkatenation: $L \cdot M := LM := \{\alpha\beta \mid \alpha \in L, \beta \in M\}$ Daher auch $L^0 := \{\lambda\}$, $L^i := L^{i-1} \cdot L$.
- Sternoperation: $L^* := \{\alpha_1 \cdots \alpha_n \mid \alpha_i \in L, i, n \in \mathbb{N}_0\}$, deswegen auch $L^* = \bigcup_{n \geq 0} L^n$
- Komplement: $L^c := T^* \setminus L$

Satz 1.1 Abgeschlossenheit

Die Mengen L_i sind abgeschlossen bezüglich der Operationen $\cup, \cdot, *$.

Beweis zu Satz 1.1

Seien $G_1 = (V_1, T, S_1, P_1), G_2 := (V_2, T, S_2, P_2)$ Grammatiken vom Typ i. Setze $L_j := L(G_j)$ für $j = 1, 2$. O.B.d.A. gelte $V_1 \cap V_2 = \emptyset$.

Vereinigung Setze $P := \{S \to S_1 | S_2\} \cup P_1 \cup P_2$, $V := V_1 \cup V_2$. Dann gilt $L(G) = L_1 \cup L_2$ mit $G := (V, T, S, P)$.

Konkatenation ($i \neq 3$) Ein Ansatz der Art $P := \{S \to S_1 S_2\} \cup P_1 \cup P_2$ ist nicht ausreichend, da auf dem Wege über das identische Terminalalphabet Produktionen aus P_1 auf Teilworte angewandt werden könnten, die eigentlich aus L_2 "stammen".

Daher legen wir zwei verschiedene Kopien T_1, T_2 des Terminalalphabets T als Variablenalphabete an. Die Produktionen müssen natürlich ebenfalls entsprechend angepasst werden. So seien Q_i die Produktionenmengen nach den erforderlichen Umbenennungen, ergänzt um Produktionen, die aus den Möchtegern-Terminalen in T_i wieder echte Terminale machen.

Seien weiter $P := \{S \to S_1 S_2\} \cup Q_1 \cup Q_2$ sowie $V := V_1 \cup V_2 \cup T_1 \cup T_2 \cup \{S\}$.

Dann ist $G := (V, T, S, P)$ wieder vom Typ i und $L(G) = L_1 L_2$.

Konkatenation ($i = 3$) Wir nehmen an, die Grammatik sei o.B.d.A. rechtslinear. Dann setzen wir $P := P_1 \cup \{A \to bS_2 \mid A \to b \in P_1\} \cup P_2$. Dann ist $G := (V, T, S_1, P)$ rechtslinear und $L(G) = L_1 \cdot L_2$.

Sternoperation ($i \neq 3$) Definiere

$$P := P_1 \cup \{S \to \lambda | S_1 | S_1 S_1 | S_1 S_2 S_1, S_2 \to S_2 S_1 S_2\}$$

*** FIXME Hä? dabei sind $G_1 = G_2$, daher $S_1 = S_2$. Dann ist $G := (V_1 \cup \{S\}, T, S, P)$ wieder vom gleichen Typ wie $G_1 = G_2$ und $L(G) = L_1^*$.

Sternoperation ($i = 3$) Die Grammatik sei o.B.d.A. rechtslinear. Setze

$$P := P_1 \cup \{S_1 \to \lambda\} \cup \{A \to bS_1 \mid A \to b \in P_1\}$$

so ist $G := (V_1, T, S_1, P)$ rechtslinear und $L(G) = L_1^*$. □

Satz 1.2

Rechtslineare Sprachen sind abgeschlossen bezüglich der Komplement- und Durchschnittbildung. (Beweis später)

2 Reguläre Sprachen

Definition 2.10 Reguläre Sprache

Rechts- oder linkslineare Sprachen werden als reguläre Sprachen bezeichnet.

2.1 Finite Akzeptoren

Definition 2.11 Deterministischer finiter Akzeptor

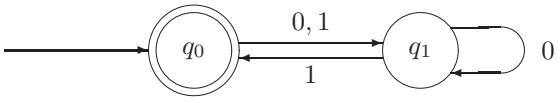
Ein deterministischer finiter Akzeptor/Automat (DFA) M ist ein Quintupel $M = (X, Q, \delta, q_0, F)$ mit dem Eingabealphabet X, der Zustandsmenge Q, der Zustandsübergangsfunktion $\delta : Q \times X \to Q$, dem Anfangszustand $q_0 \in Q$ und der Endzustandsmenge F.

Man definiert rekursiv eine mehrstufige Übergangsfunktion $\delta^* : Q \times X^* \to Q$ mit $\delta^*(q, \lambda) := q$ und $\delta^*(q, wx) := \delta(\delta^*(q, w), x)$.

Die von einem Automaten M akzeptierte Sprache

$$L(M) := \{w \in X^* \mid \delta^*(q_0, w) \in F\}$$

Beispiel Ein einfacher Automat



Hierbei ist $Q := \{q_0, q_1\}$, $X := \{0, 1\}$ sowie $F := \{q_0\}$. Die Zustandsübergangstabelle lautet

δ	0	1
q_0	q_1	q_1
q_1	q_1	q_0

Die von diesem Automaten akzeptierte Sprache ist $L = ((0+1)0^*1)^*$. (Notation siehe Definition 2.13.)

Satz 2.3

Voraussetzung: $M = (X, Q, \delta, q_0, F)$ DFA

Dann ist $L(M)$ rechtslinear.

Beweis zu Satz 2.3

Wir legen einfach die Funktionsweise eines Automaten in Form von Produktionen dar. Setze

$$P := \{q \to xq' \mid \delta(q,x) = q', q, q' \in Q, x \in X\} \cup \{q \to x \mid \delta(q,x) \in F, q \in Q, x \in X\}$$

Dann ist $G := (Q, X, q_0, P)$ rechtslinear. □

Definition 2.12 Nichtdeterministischer finiter Akzeptor

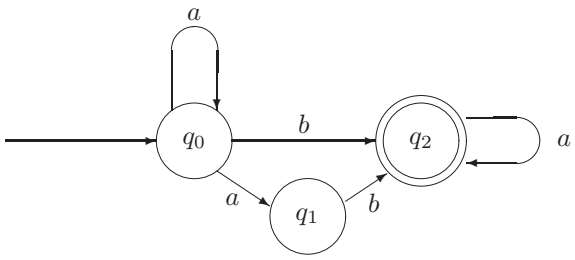
Ein Quintupel $M = (X, Q, \delta, q_0, F)$ heißt ein nichtdeterministischer finiter Akzeptor (NFA) :⇔ X, Q, q, F sind wie bei einem DFA und die Zustandsübergangsfunktion ist $\delta : Q \times X \to \mathcal{P}(Q)$.

Rekursiv definiert man wieder $\delta^* : Q \times X^* \to \mathcal{P}(Q)$ mit $\delta^*(q, \lambda) := \{q\}$ und $\delta^*(q, wx) := \delta(\delta^*(q, w), x)$. (Beachte: Ergebnis von δ^* in vorigem Ausdruck ist bereits eine Menge!)

Die von M akzekptiere Sprache ist

$$L(M) := \{w \in X^* \mid \delta^*(q_0, w) \cap F \neq \emptyset\}$$

Beispiel Ein einfacher NFA



Die zugehörige Grammatik ist rechtslinear:

$q_0 \to aq_0|aq_1|bq_2|b$
$q_1 \to bq_2|b$
$q_2 \to aq_2|a$

Satz 2.4

Voraussetzung: L rechtslinear

Dann existiert ein NFA M mit $L(M) = L$.

Satz 2.5

Voraussetzung: $M = (X, Q, \delta, q_0, F)$ NFA

Es existiert ein DFA M' mit $L(M) = L(M')$.

Beweis zu Satz 2.5

Man definiert schlicht

$$M' := (X, \mathcal{P}(Q), \delta', \{q_0\}, F')$$

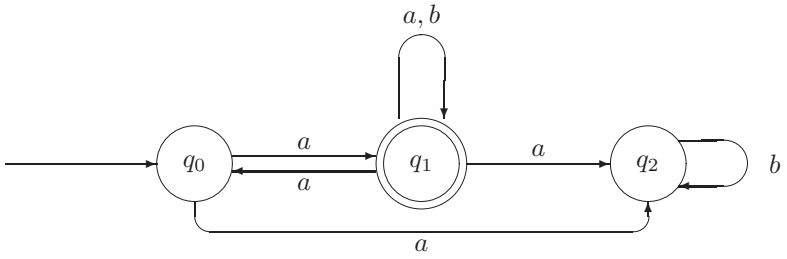
mit

$$\delta'(\{q_1, \ldots, q_n\}, x) := \delta(q_1, x) \cup \ldots \cup \delta(q_n, x)$$

und $F' := \{p \in \mathcal{P}(Q) \mid p \cap F \neq \emptyset\}$. Dieser Automat ist ein DFA und erkennt dieselbe Sprache wie M. □

Beispiel Umwandlung NFA → DFA

Wir geben zunächst einen NFA M an:

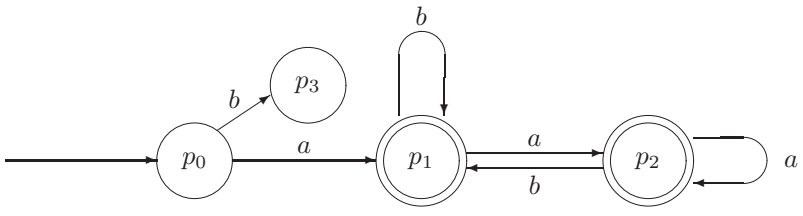


Die Zustandsübergangstabelle für diesen NFA lautet wie folgt:

$p_0 := \{q_0\}$
$p_1 := \{q_1, q_2\}$
$p_2 := \{q_0, q_1, q_2\}$
$p_3 := \emptyset$

δ'	p_0	p_1	p_2
a	p_1	p_2	p_2
b	p_3	p_1	p_1

Ein zugehöriger DFA M' ist also



Korollar 2.6

Voraussetzung: L rechtslineare Sprache

Es existiert ein DFA M mit $L(M) = L$.

2.2 Das Pumping-Lemma

Lemma 2.7 Pumping-Lemma

Voraussetzung: $M = (X, Q, \delta, q_0, F)$ DFA, $|Q| = n$

Dann gilt für alle $z \in L(M)$ mit $|z| > n$

$$(\exists u, v, w \in T^*)(z = uvw, v \neq \lambda, uv^*w \in L(M))$$

Beweis zu Lemma 2.7

Sei $z = a_1 \dots a_k$, $|z| = k > n$. Dann existieren $q_1, \dots, q_n \in Q$ mit $\delta(q_{i-1}, a_i) = q_i$ für $i = 1, \dots k$.

Wegen $k > n$ existieren nun mehr Zustände des DFA als Zeichen im Wort, es existieren also $i < j \in \{1, \dots, k\}$ mit $q_i = q_j$. Dann erfüllt $v := a_{i+1} \dots a_j$ die Behauptung, denn der Zyklus $q_i \dots q_j = q_i$ akzeptiert v beliebig oft. □

Bemerkung Folgerung

Die Sprache $\{a^n b^n \mid n \in \mathbb{N}\}$ ist nicht rechtslinear, aber kontextfrei. Wäre sie rechtslinear, müsste für sie auch Lemma 2.7 gelten. Offensichtlich kann dies aber nicht der Fall sein, da kein Teilwort v von $a^n b^n$ die geforderte Bedingung erfüllen kann.

Satz 2.8

Voraussetzung: K, L rechtslinear

Dann sind auch K^c und $K \cap L$ rechtslinear.

Beweis zu Satz 2.8

Wir zeigen beide Teile der Behauptung:

Komplement: Betrachte den zu K gehörigen DFA $M := (X, Q, \delta, q_0, F)$. Dann ist auch $M^c := (X, Q, \delta, q_0, Q \setminus F)$ ein DFA, und er akzeptiert genau K^c. Also ist auch K^c rechtslinear.

Schnitt: Man mache sich klar, dass gilt

$$K \cap L = (K^c \cup L^c)^c$$

womit sich die Aussage aus dem Vorangegangenen ergibt. □

2.3 Reguläre Ausdrücke

Definition 2.13 Regulärer Ausdruck

R ist ein regulärer Ausdruck über einem Alphabet T, falls

1. $R = \emptyset, \lambda, a$ mit $a \in T$
2. $R = (R_1 + R_2)$ mit regulären Ausdrücken R_1, R_2 (Vereinigung)
3. $R = (R_1 \cdot R_2)$ mit regulären Ausdrücken R_1, R_2 (Konkatenation)
4. $R = (R_1)^*$ mit regulärem Ausdruck R_1 (Sternoperation)

Eine Sprache L heißt regulär, falls sie von einem regulären Ausdruck erzeugt wird.

Beispiel Reguläre Ausdrücke

Z.B. $\emptyset, \lambda, a^*, (ab + bb)^*$.

Bemerkung

Mit den Ergebnissen aus 1.4 erhalten wir: Jede reguläre Sprache ist rechtslinear.

Satz 2.9

Voraussetzung: $M = (Q, X, \delta, q_0, F)$ ein DFA, $Q = \{q_0, \ldots, q_n\}$
Dann ist $L(M)$ regulär.

Beweis zu Satz 2.9

Seien $i, j \in \{0, \ldots, n\}$ beliebig. Setze $R_{ij} := \{w \mid \delta^*(q_i, w) = q_j\}$. Ist $w \in R_{ij}$ und $q_i, q_{i_1}, \ldots, q_{i_l}, q_j$ die Abfolge der Zustände bei der Akzeption von w, so definiere $Q(i, j, w) := \{q_{i_1}, \ldots, q_{i_l}\}$. Definiere dann weiterhin

$$R_{ij}^k := \{w \in R_{ij} \mid Q(i, j, w) \cap \{q_k, \ldots, q_n\} = \emptyset\}$$

Wir zeigen durch vollständige Induktion über k: $R_{ij}^{n+1} = R^{ij}$ ist rechtslinear. Für den Induktionsanfang (R_{ij}^0) ist dies klar, denn es gilt $R_{ij}^0 \subseteq X$.

$k \curvearrowright k + 1$: Es gilt offensichtlich $R_{ij}^{k+1} = R_{ij}^k + R_{ik}^k (R_{kk}^k)^* R_{kj}^k$. Weiterhin ist dieser Ausdruck nach Bildung und Induktionsvoraussetzung regulär. □

Bemerkung Linearität

Da die Argumentation von Satz 2.9 auch auf linkslineare Sprachen anwendbar ist, erhält man:

$$\{\text{rechtslineare Sprachen}\} = \{\text{reguläre Sprachen}\} = \{\text{linkslineare Sprachen}\}$$

2.4 Eine Äquivalenzrelation

Definition 2.14 Index einer Äquivalenzrelation

Sei "$\sim$" eine Äquivalenzrelation. Dann nennt die Anzahl der von $\sim$ erzeugten Äquivalenzklassen auch den Index von $\sim$, geschrieben $\# \sim$.

Definition 2.15 Charakteristische Äquivalenzrelation

Sei $L \subseteq T^*$ eine Sprache. Dann definiert man für $x, y \in T^*$

$$x \sim_L y :\Leftrightarrow (\forall z \in T^*)(xz \in L \Leftrightarrow yz \in L)$$

Man kann leicht nachprüfen, dass diese Relation symmetrisch, reflexiv und transitiv, also eine Äquivalenzrelation ist.

Sei $M = (Q, X, \delta, q_0, F)$ ein DFA. Dann definiert man für $x, y \in X^*$

$$x \sim_M y :\Leftrightarrow \delta^*(q_0, x) = \delta^*(q_0, y)$$

Auch diese Relation ist eine Äquivalenzrelation, ihr Index ist gerade $|Q|$.

Satz 2.10

Voraussetzung: $L \subseteq T^*$ Sprache

Dann gilt: L regulär $\Leftrightarrow$ Index von $\sim_L < \infty$.

Beweis zu Satz 2.10

"$\Longrightarrow$" Sei $M = (Q, T, \delta, q_0, F)$ ein DFA mit $L = L(M)$. Dann gilt für alle $x \in T^*$: $x \sim_M y \implies x \sim_L y$. In diesem Sinne ist $\sim_M$ eine Verfeinerung von $\sim_L$, also gilt $\infty > \# \sim_M > \# \sim_L$.

"$\Longleftarrow$" Seien $[w_1], \ldots, [w_n]$ die Äquivalenzklassen von $\sim_L$. Wir definieren dann $M := (Q, X, \delta, q_0, F)$ mit $Q := \{[w_1], \ldots, [w_n]\}$, $\delta([w_i], x) := [wx]$, $q_0 := [\lambda]$, $F := \{[w] \mid w \in L\}$. Die Definition von F ist repräsentantenunabhängig, denn sei $w_1 \sim_L w_2, w_1 \in L$. Dann folgt nach Def. $w_1\lambda \in L \Leftrightarrow w_2\lambda \in L$. Insgesamt erhält man $w \in L(M) \Leftrightarrow w \in L$. □

Beispiel

Wir geben zwei Beispiele für dieses Kriterium an.

1. Sei $L := \{a^n b^n \mid n \leq 1\}$. Einige Äquivalenzklassen von $\sim_L$ sind dann zum Beispiel
   - $[\lambda] = \{\lambda\}$

- $[a] = \{a, a^2b, a^3b^2, a^4b^3, \dots\}$
- $[a^2] = \{a^2, a^3b, a^4b^2, a^5b^3, \dots\}$
- $[a^3] = \{a^3, a^4b, a^5b^2, a^6b^3, \dots\}$

Der Index von $\sim_L$ ist offensichtlich unendlich. Folglich ist L nicht regulär.

2. Sei nun $L := \{w \in \{0,1\}^+ \mid w \text{ endet mit } 00\}$. Dann sind die sämtlichen Äquivalenzklassen von $\sim_L$:

   - $[\lambda] = \{w \in \{0,1\}^+ \mid w \text{ endet nicht mit } 0\}$
   - $[0] = \{w \in \{0,1\}^+ \mid w \text{ endet mit } 0, \text{ aber nicht mit } 00\}$
   - $[00] = \{w \in \{0,1\}^+ \mid w \text{ endet mit } 00\}$

   Folglich ist $\{0,1\}^+ = [\lambda] \cup [0] \cup [00]$, der Index von $\sim_L$ ist also gleich 3, L also regulär.

2.5 Der Minimalautomat

Definition 2.16 Minimalautomat

Sei L eine Sprache. Dann ist der Minimalautomat für diese Sprache derjenige DFA $M = (X, Q, \delta, q_0, F)$ mit minimalem $|Q|$, für den noch gilt $L(M) = L$.

Bemerkung Folgerungen aus Abschnitt 2.4

Unmittelbar kann man beobachten:

- Sei n der Index der Äquivalenzrelation $\sim_L$. Dann hat jeder endliche DFA M mit $L = L(M)$ mindestens n Zustände.
- Der Äquivalenzklassenautomat aus dem Beweis zu Satz 2.10 ist minimal.
- Es gibt bis auf Isomorphie nur einen Minimalautomaten für L.
- Ein DFA kann durch Verschmelzen von Zuständen in den Minimalautomaten überführt werden.

Zwei Zustände p, q sind nicht verschmelzbar, falls falls es ein $w \in X^*$ gibt mit $\delta^*(p, w) \in F$, aber $\delta^*(q, w) \notin F$ beziehungsweise falls $\delta^*(p, w)$ und $\delta^*(q, w)$ nicht verschmelzbar sind. Daraus ergibt sich sofort der folgende Minimierungsalgorithmus, der von der Grundsatzidee nur "Paare von nicht verschmelzbaren Zuständen anhäuft".

Algorithmus Minimierung eines DFA

Gegeben: $M = (X, Q, \delta, q_0, F)$ DFA.

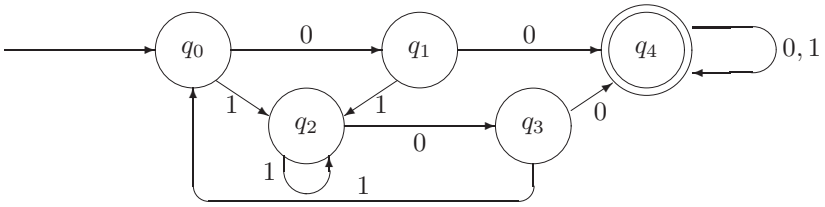
Gesucht: Zugehöriger Minimalautomat $\tilde{M} = (X, \tilde{Q}, \tilde{\delta}, q_0, F)$

Ablauf:

- Entferne alle Zustände, die nicht von q_0 aus erreichbar sind.
- Markiere alle Paare $(p, q) \in F \times F^c \cup F^c \times F$.
- Wiederhole, bis keine Markierungen mehr hinzukommen:
  - Für alle unmarkierten Zustände $(p, q) \in Q^2$:
    - Ist $(\delta(p, a), \delta(q, a))$ für ein $a \in X$ markiert, so markiere (p, q).
- Verschmelze alle unmarkierten Zustandspaare miteinander.

Beispiel Minimierung eines Beispielautomaten

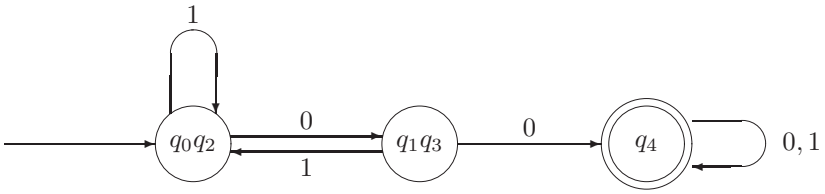
Gegeben sei der folgende Automat



Im ersten Schritt werden die Paare $(q_0, q_4), (q_1, q_4), (q_2, q_4), (q_3, q_4)$ markiert, also alle Paare, bei denen ein Zustand ein Finalzustand ist, der andere aber nicht.

Im zweiten Schritt werden die Paare $(q_0, q_1), (q_0, q_3), (q_1, q_2), (q_2, q_3)$ markiert, da sie auf Zustände führen, die als Paar schon markiert sind.

Der dritte Schritt bringt keine weiteren Markierungen, also sind wir fertig mit Aussortieren. Wir können nun beginnen, die verbleibenden Zustandspaare zu verschmelzen. Wir kombinieren (q_0, q_2) zu q_0q_2 und (q_1, q_3) zu q_1q_3. Der resultierende Minimalautomat ist



Erkannt wird in beiden Fällen die Sprache $L = \{w \in \{0, 1\}^+ \mid w \text{ enthält } 00\}$.

2.6 Entscheidbarkeit

Definition 2.17 Produktautomat

Seien $M_1 = (X, Q_1, \delta_1, q_0, F)$ und $M_2 = (X, Q_2, \delta_2, q_0', F)$ DFA'en. Dann heißt der Automat

$$M := (X, Q_1 \times Q_2, (\delta_1, \delta_2), (q_0, q_0'), F \times F')$$

der Produktautomat von M_1 und M_2.

Es gilt $L(M) = L(M_1) \cap L(M_2)$.

Bemerkung Algorithmische Entscheidbarkeit

Algorithmisch entscheidbar sind:

- Wortproblem: $w \in L$?
- Leerheit: $L = \emptyset$? Ist ein Endzustand von q_0 aus erreichbar?
- Äquivalenz: $L_1 = L_2$? Minimalautomaten vergleichen.
- Schnitt: $L_1 \cap L_2 = \emptyset$? Betrachte hierfür den Produktautomaten.
- Endlichkeit: $|L| = \infty$? Hierbei gilt wegen des Pumping-Lemmas (mit dem dortigen $n = |Q|$ des DFA)

  $$|L| = \infty \Leftrightarrow (\exists z \in L)(n < |z| \leq 2n)$$

  Wir prüfen also nur noch alle Worte der Länge $l \in (n, 2n]$ auf Zugehörigkeit zu L.

3 Kontextfreie Grammatiken

3.1 Syntaxbäume

Beispiel Kontextfreie Grammatik

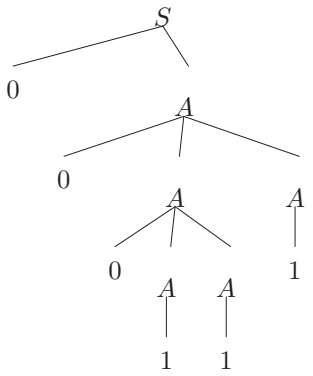
Wir betrachten eine Grammatik mit den Produktionen

$$S \to 0A$$
$$A \to 0AA|1$$

Die kontextfreie Ableitung

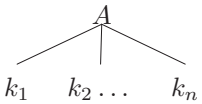
$$S \Rightarrow 0A \Rightarrow 00AA \Rightarrow 000AAA \Rightarrow^* 000111$$

hat den Syntax oder Ableitungsbaum



Bemerkung Beobachtungen am Syntaxbaum

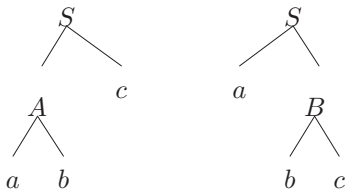
- S ist die Wurzel
- Blätter sind Elemente von $T \cup \{\lambda\}$
- Knoten sind Elemente von V
- Ist

  ein Teilbaum, dann gilt $A \to k_1 k_2 \dots k_n \in P$.
- Zu einem gegebenen Ableitungsbaum ist die Reihenfolge der Ableitungen i.A. nicht eindeutig.
- Zu einem gegebenen Wort aus $L(G)$ mit einer Grammatik G ist die Ableitung i.A. nicht eindeutig. Grammatiken, auf die dieses zutrifft, heißen mehrdeutig.

Beispiel Verschiedene Bäume

Die beiden letzten Sachverhalte veranschaulicht das folgende Beispiel:



Definition 3.18 Linksableitung

Eine Links- bzw. Rechtsableitung ist eine Ableitung, bei der die am weitesten links beziehungsweise rechts stehende Variable zuerst ersetzt wird.

3.2 Das Pumping-Lemma

Lemma 3.11 Baumhöhenlemma

Voraussetzung: $G = (V, T, S, P)$ eine kontextfreie Grammatik, $m := \max\{|n| \mid (\exists A \in V)(A \to n \in P)\}$, $z \in L(G)$, Ableitungsbaum von z hat die Höhe k

Dann ergibt sich sofort aus dem Ableitungsbaum: $|z| \leq m^k$.

Satz 3.12 Pumping-Lemma (auch $uvwxy$-Theorem)

Voraussetzung: $G = (V, T, S, P)$ kontextfreie Grammatik

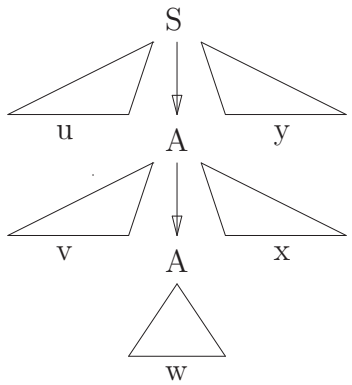
Dann existiert ein $p \in \mathbb{N}$ so, dass für alle $z \in L(G)$ mit $|z| \geq p$ gilt:

$$(\exists u, v, w, x, y \in T^*)(z = uvwxy, \\ |vwx| \leq p, vx \neq \lambda, (\forall i \in \mathbb{N}_0)(uv^iwx^iy \in L(G)))$$

Beweis zu Satz 3.12

Setze wieder $m := \max\{|n| \mid (\exists A \in V)(A \to n \in P)\}$, $z \in L(G)$. Setze $p := m^{|V|+1}$, sei $z \in L(G)$ nun ein Wort mit $|z| \geq p$. Wegen Lemma 3.11 kann der Ableitungsbaum von z keine Höhe $h \leq |V| + 1$ haben, da z sonst gar nicht so lang sein könnte.

Da nun $h > |V|$, muss im Ableitungsbaum ein Nichtterminal doppelt vorkommen, wähle dasjenige Nichtterminal, welches als letztes doppelt vorkommt, dieses sei A. Betrachte folgende Abbildung:



Beachte: A kann in der ersten Ableitung $S \Rightarrow^* A$ auch schon vorkommen, wähle daher v und x so, dass in der Ableitung $A \Rightarrow^* vAx$ kein weiteres A mehr vorkommt, weiterhin so, dass $vx \neq \lambda$ (dies kann o.B.d.A. verlangt werden, da eine Ableitung $A \Rightarrow^* A$ nichts zur Länge beigetragen hat, d.h. unser Wort erfüllt die Voraussetzungen des Satzes auch nach Entfernung dieser vergeblichen Ableitung noch)

Nun gilt: $|vwx| \leq p$ aufgrund der Wahl von v, x. Offensichtlich sind nun auch uv^iwx^iy mit $i \in \mathbb{N}_0 \in L(G)$. □

Beispiel Folgerung

Die Sprache $L := \{a^nb^nc^n \mid n > 0\}$ ist nicht kontextfrei.

Wähle $n > p$. Angenommen $uvwxy = a^nb^nc^n$ mit $|vwx| \leq p < n$ sowie $vx \neq \lambda$ und p beliebig. Dann folgt

$$vwx \in a^*b^* \text{ oder } b^*c^*$$

und somit $uv^2wx^2y \notin L$. Daraus folgt nach dem Pumping-Lemma, dass L nicht kontextfrei ist.

L ist sehr wohl kontextsensitiv, denn die Produktionen

$$\begin{aligned} S &\to aBC|aSBC \\ CB &\to BC \\ aB &\to ab \\ bB &\to bb \\ bC &\to bc \\ cC &\to CC \end{aligned}$$

erzeugen L und sind *erweiternd*. Daher ist L auch kontextsensitiv.

3.3 λ- und kontextfreie Sprachen

Satz 3.13

Voraussetzung: $G = (V, T, S, P)$ kontextfrei

$L(G)$ kontextsensitiv.

Beweis zu Satz 3.13

Einziges Problem ist hier, dass bei kontextfreien Sprachen Produktionen der Form $A \to \lambda$ zugelassen sind, bei kontextsensitiven jedoch nicht.

Seien $P^* := \{A_1 \to \lambda, \ldots, A_k \to \lambda\}$ alle λ-Produktionen in P. Setze $P' := P \setminus P^*$.

Dann müssen wir lediglich für jede Produktion in P', in der mindestens eines von $A_1, \ldots, A_k$ auf einer rechten Seite auftritt, alle Möglichkeiten "durchrattern", die es gibt, die $A_1, \ldots, A_k$ wegzulassen. Für jeden solchen "Ratterschritt" erzeugen wir eine neue Produktion. Dann ist die so erzeugte neue Produktionenmenge P'' gleichwertig mit P, aber λ-frei.

Setze weiterhin

$$G' := \begin{cases} (V, T, S, P'') & \text{falls } \lambda \notin L(G) \\ (V \cup \{S'\}, T, S', P'' \cup \{S' \to \lambda | S\}) & \text{sonst} \end{cases}$$

Dann ist G' kontextsensitiv mit $L(G) = L(G')$. □

3.4 Abschlusseigenschaften

Satz 3.14

L_2 ist *nicht* abgeschlossen bezüglich des Schnittes.

Beweis zu Satz 3.14

Durch ein Gegenbeispiel. $L_1 := \{a^m b^n c^n \mid m, n \in \mathbb{N}\}$ und $L_2 := \{a^n b^n c^m \mid m, n \in \mathbb{N}\}$ sind kontextfrei, $L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ allerdings nicht. □

Bemerkung Abgeschlossenheit ggü. Komplementbildung

Als direkte Folge von Satz 3.14 und aufgrund der DeMorgan-Regel sind kontextfreie Sprachen bezüglich des Komplementbildung i.A. nicht abgeschlossen.

3.5 Die Chomsky-Normalform

Definition 3.19 Chomsky-Normalform

Eine kontextfreie Grammatik ist in *Chomsky-Normalform* (CNF), wenn sie λ-frei ist und alle ihre Produktionen von folgender Form sind:

- $S \to \lambda$ oder
- $A \to BC$ oder
- $A \to a$,

mit $A, B, C \in V$ sowie $a \in T$.

Satz 3.15

Voraussetzung: $L = L(G)$ mit $G = (V, T, S, P)$ kontextfrei

Dann existiert eine Grammatik G' mit $L(G') = L(G)$, wobei G' Chomsky-Normalform besitzt.

Beweis zu Erzeugung der Chomsky-Normalform

Durch Angabe eines Algorithmus, wir erzeugen $G' = (V', T, S, P')$. *Ablauf*:

- Mache die Grammatik λ-frei.
- Erzeuge ein neues Nichtterminal $A_a \in V'$ für jedes Terminal $a \in T$, Ergänze P' um $A_a \to a$.
- Entferne alle überflüssigen Produktionen der Art $A \to A$ aus P'.
- Eliminiere alle Zyklen: Für jeden Zyklus $A_1 \to A_2 \to \cdots \to A_k \to A_1$ eliminiere $A_2, \ldots, A_k$, ändere deren Produktionen so, dass sie statt dessen von A_1 ausgehen.
- Eliminiere alle Umwege: Befinden sich noch Produktionen der Art $A \to B \in P'$, wobei $A, B \in V'$, so existieren auch Produktionen $B \to \cdots \to C \to |\alpha|$ mit $|\alpha| \geq 2$. Nimm für alle auf diese Art erzeugbaren $\alpha \in (V \cup T)^*$ die Produktion $A \to w$ in P' auf, wirf dafür die Einer-Produktionen weg.
- Eliminiere alle Produktionen $A \to B_1 \ldots B_m$ mit $m \geq 3$: Nimm einfach statt ihrer die neuen Nichtterminale $C_1, \ldots, C_{m-2}$ in V' auf, erzeuge folgende Produktionen:

$$\begin{aligned} A &\to B_1 C_1 \\ C_1 &\to B_2 C_2 \\ &\cdots \\ C_{m-2} &\to B_{m-1} B_m \end{aligned}$$

Dann ist G' offensichtlich in CNF. □

Beispiel Chomsky-Normalform einer Grammatik

Wir betrachten das folgende Beispiel:

$$\begin{aligned} S &\to AB \\ A &\to aA|a \\ B &\to bBc|bc \end{aligned}$$

Wir formen die Grammatik um und erhalten

$$\begin{aligned} S &\to AB \\ A' &\to a \\ B' &\to b \\ C' &\to c \\ A &\to A'A|a \\ B &\to B'C'|B'D \\ D &\to BC' \end{aligned}$$

3.6 Der Cocke-Kasami-Younger-Algorithmus

Algorithmus Cocke-Kasami-Younger-Algorithmus

Gegeben: $G = (V, T, S, P)$ in CNF, $w \in T^*$

Gesucht: $w = x_1 \dots x_n \in L(G)$?

Ablauf:

- Lege eine Pyramide mit den Zeilen $Y[1,1] \dots Y[1,n]$, $\dots Y[n,1]$ an, wobei $Y[i,j] \in \mathcal{P}(V)$. $Y[i,j]$ wird im Laufe des Algorithmus all diejenigen Nichtterminale enthalten, aus denen $x_j \dots x_{j+i-1}$ erzeugt werden kann.
- Initialisierung: Schreibe in $Y[1,j]$ jeweils alle Nichtterminale V, aus denen x_j $(j = 1, \dots, n)$ erzeugt werden kann.
- Für $i = 2, \dots, n$:
  - Für $j = 1, \dots, i$:
    - Für $m = 1, \dots, i-1$:
      - Untersuche alle Produktionen der Form $A \to BC \in P$: Ist $B \in Y[m,j]$ ($\implies$ m lang, gleicher Offset) und $C \in Y[i-m, j+m]$ ($\implies$ $i-m$ lang, um m nach rechts verschobener Offset), so ergänze $Y[i,j]$ um A.
- Ist nun $S \in Y[n,1]$, so gilt $w \in L(G)$.

Aufwand: $O(n^3)$

Bemerkung

Der CKY-Algorithmus ist ein Bottom-up-Algorithmus.

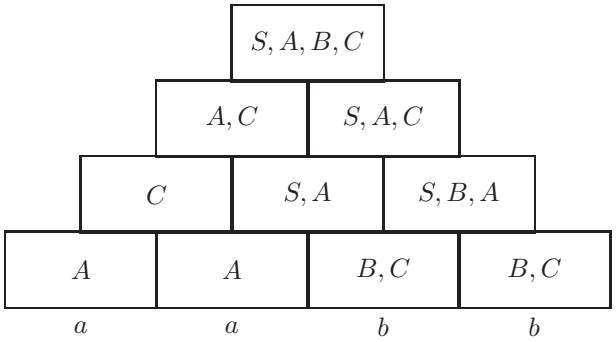
Aus der vom CKY-Algorithmus erzeugten Pyramide kann man sämtliche möglichen Ableitungsbäume eines Wortes ablesen.

Beispiel

Wir betrachten als Beispiel die Grammatik mit den Produktionen

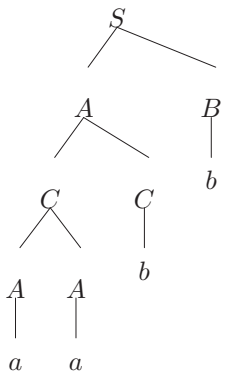
$$\begin{aligned} S &\rightarrow AB|BB \\ A &\rightarrow AB|CC|a \\ B &\rightarrow CA|BB|b \\ C &\rightarrow BA|AA|b \end{aligned}$$

und fragen, ob $x_1x_2x_3x_4 := aabb \in L(G)$ gilt. Dazu bauen wir die Pyramide auf:



Es ist zu sehen, daß gilt $S \in$ Zelle $(4, 1)$. Es folgt also, daß $aabb \in L(G)$!

Im Beispiel gibt es nur einen Syntaxbaum:



3.7 Entscheidbarkeit

Bemerkung

Sei $G = (V, T, S, P)$ eine kontextfreie Grammatik in CNF. Dann sind in polynomieller Zeit lösbar:

Wortproblem: Gilt $w \in L(G)$? Diese Frage kann der CKY-Algorithmus beantworten.

Leerheit: Gilt $L(G) = \emptyset$? Der folgende Algorithmus bestimmt alle Variablen, aus denen Worte ableitbar sind: *Ablauf*:

- $U := \emptyset \subseteq V$
- $U' := \{A \in V \mid (\exists t \in T)(A \to t \in P)\} \subseteq V$
  (alle Nichtterminale, die in einem Schritt Terminale erzeugen)
- Solange $U \neq U'$:
  - $U := U'$
  - $U' := \{A \in V \mid (\exists B, C \in U)(A \to BC \in P)\}$
- $S \in U \Leftrightarrow L(G) \neq \emptyset$

Die Nichtterminale in $V \setminus U$ sind nutzlos und können rückstandsfrei entfernt werden.

Endlichkeit: Gilt $|L(G)| = \infty$? Enthalte V keine nutzlosen Variablen. Der folgende Algorithmus bestimmt alle Variablen, die nur endliche Worte $w \in T^*$ erzeugen. *Ablauf*:

- $U := \emptyset \subseteq V$
- Setze

$$U' := \{A \in V \mid (\exists t \in T)(A \to t \in P)\} \cap \{A \in V \mid (\forall B, C \in V)(A \to BC \notin P)\} \subseteq V$$

  (alle Nichtterminale, die in einem Schritt *nur* Terminale erzeugen)
- Solange $U \neq U'$:
  - $U := U'$
  - $U' := U \cup \{A \in V \mid (\exists B, C \in U)(A \to BC \in P)\}$
- $S \in U \Leftrightarrow |L(G)| = \infty$

Erläuterung: Der Trick an diesem Algorithmus ist, dass er in U keine Nichtterminale aufnimmt, die unendliche Wörter erzeugen, also Produktionen wie $A \to BC, C \to BA$ mit $B \to b$, und zwar deswegen, weil der Algorithmus nur strikte "Bottom-up"-Konstruktion zulässt, ein Nichtterminal, das Dinge produziert, die noch nicht bekannt (d.h. $\in U$) sind, wird nicht zugelassen (d.h. in U gesteckt).

Beispiel

Wir wollen nun als Beispiel eine Grammatik mit den Produktionen

$$\begin{aligned} S &\to AD|BC \\ B &\to BC \\ D &\to FE|b \\ F &\to AD \\ A &\to a \\ E &\to b|c \end{aligned}$$

Die Produktionen $S \to BC$ und $B \to BC$ sind nutzlos, da das Nichtterminal C nicht in ein Terminal umgewandelt werden kann. Die beiden Produktionen können also gestrichen werden.

Nutztest: Wir wollen nun bestimmen, ob G überhaupt eine Sprache erzeugt. Dazu wenden wir Algorithmus 1 an. Wir notieren die Ergebnisse in Tabellenform.

Durchlauf	U
0	$\emptyset$
1	$\{A, D, E\}$
2	$\{S, F, A, D, E\}$
3	$\{S, F, A, D, E\}$

Nach Ablauf des Algorithmus ist das Startsymbol S in U enthalten. Es folgt also $L \neq \emptyset$.

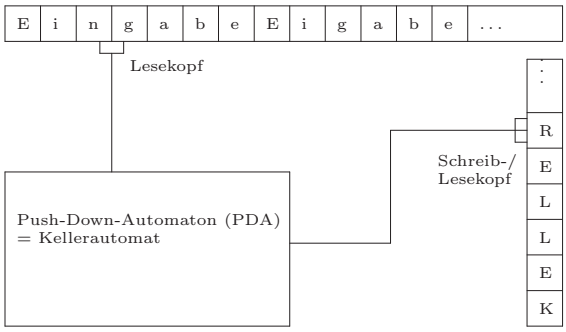
Endlichkeitstest: Weiterhin betrachten wir den Index von L. Hierzu verwenden wir Algorithmus 2; hierbei ist zu beachten, daß nutzlose Produktionen aus der Grammatik zu entfernen sind, wie wir es getan haben.

Durchlauf	U
0	$\emptyset$
1	$\{A, E\}$
2	$\{A, E\}$

Hier ist nach Beendigung des Algorithmus das Startsymbol S nicht in U enthalten. Es folgt also $|L| = \infty$.

3.8 Einfache Kellerautomaten

Definition 3.20 Einfacher Kellerautomat



Seien T ein Eingabe-, $\Gamma \supseteq T$ ein Kelleralphabet, $S \in \Gamma$ und $\delta : T \cup \{\lambda\} \times \Gamma \to \mathcal{P}(\Gamma^*)$ eine Übergangsfunktion, deren Ergebnismenge immer höchstens endlich sein darf.

Dann nennt man $M = (T, \Gamma, \delta, S)$ einen einfachen Kellerautomaten. Ein Tupel $K := (a_i \dots a_n, A_k \dots A_1) \in T^* \times \Gamma^*$ heißt Konfiguration des Automaten.

Eine Konfiguration K eines einfachen Kellerautomaten kann mehrere Nachfolgekonfiguationen haben, einfache Kellerautomaten sind demzufolge nach Definition nichtdeterministisch. Ausgehend von dieser Konfiguration werden aufgrund der Zustandsübergangsfunktion δ mögliche Nachfolgekonfigurationen ermittelt:

- $(a_{i+1} \dots a_n, A_{k-1} \dots A_1)$ ist mögliche Nachfolgekonfiguration, falls $\lambda \in \delta(a_i, A_k)$
- $(a_i \dots a_n, \alpha A_{k-1} \dots A_1)$ ist mögliche Nachfolgekonfiguration, falls $\alpha \in \delta(\lambda, A_k)$ mit $\alpha \in \Gamma^*$.

Für die Überführung von Konfigurationen in andere schreibt man wie gehabt $K \Rightarrow K', K \Rightarrow^* K'$.

Die vom Kellerautomaten akzeptierte Sprache ist

$$L(M) := \{w \in T^* \mid (w, S) \Rightarrow^* (\lambda, \lambda)\}$$

Satz 3.16

Voraussetzung: $G = (V, T, S, P)$ kontextfreie Grammatik

Dann existiert ein einfacher Kellerautomat $M = (T, \Gamma, \delta, S)$ mit $L(M) = L(G)$.

Beweis zu Satz 3.16

Betrachte die obige Definition mit $\Gamma := V \cup T$ und die direkte Analogie zwischen den Möglichkeiten für die δ-Funktion und den Möglichkeiten für Produktionen in kontextfreien Sprachen. □

Satz 3.17

Voraussetzung: $M = (T, \Gamma, \delta, S)$ einfacher Kellerautomat

$L(M)$ ist kontextfreie Sprache.

Beweis zu Satz 3.17

Betrachte die obige Definition mit $V := \Gamma \setminus T$ und die direkte Analogie zwischen den Möglichkeiten für Produktionen in kontextfreien Sprachen und den Möglichkeiten für die δ-Funktion. □

Bemerkung

Im Gegensatz zum CKY-Algorithmus führt ein einfacher Kellerautomat eine Top-Down-Analyse durch.

Bemerkung Greibach-Normalform

Jede kontextfreie Grammatik kann in eine äquivalente Grammatik umgeformt werden mit Produktionen der Form

$$A \to aB_1 \dots B_k, a \in T \cup \{\lambda\}, A, B_1, \dots, B_k \in V$$

Ist $\lambda \notin L(G)$, so kann G auch in Greibach-Normalform überführt werden. Alle Produktionen sind dann von der Form

$$A \to aB_1 \dots B_k, a \in T, A, B_1, \dots, B_k \in V$$

3.9 Allgemeine Kellerautomaten

Definition 3.21 Allgemeiner Kellerautomat

Sei Q eine endlichen Zustandsmenge, T ein Eingabealphabet, Γ ein Kelleralphabet, $\delta : Q \times T \cup \{\lambda\} \times \Gamma \to \mathcal{P}(Q) \times \mathcal{P}(\Gamma^*)$ eine Zustandsübergangsfunktion, deren Ergebnismenge höchstens endlich sein darf, $q_0 \in Q$ ein Anfangszustand und $S \in \Gamma$ ein Start-Kellerzeichen.

Dann nennt man $M = (Q, T, \Gamma, \delta, q_0, S)$ einen allgemeinen Kellerautomaten. Ein Tripel $K := (q, a_i \dots a_n, A_k \dots A_1) \in Q \times T^* \times \Gamma^*$ heißt Konfiguration des Automaten.

Eine Konfiguration K eines einfachen Kellerautomaten kann mehrere Nachfolgekonfiguationen haben, einfache Kellerautomaten sind demzufolge nach Definition nichtdeterministisch. Ausgehend von dieser Konfiguration werden aufgrund der Zustandsübergangsfunktion δ mögliche Nachfolgekonfigurationen ermittelt:

- $(q', a_{i+1} \dots a_n, A_{k-1} \dots A_1)$ ist mögliche Nachfolgekonfiguration, falls $(q', \lambda) \in \delta(q, a_i, A_k)$
- $(q', a_i \dots a_n, \alpha A_{k-1} \dots A_1)$ ist mögliche Nachfolgekonfiguration, falls $(q', \alpha) \in \delta(q, a_i, A_k)$ mit $\alpha \in \Gamma^*$.

Für die Überführung von Konfigurationen in andere schreibt man wie gehabt $K \Rightarrow K', K \Rightarrow^* K'$.

Die vom Kellerautomaten akzeptierte Sprache ist

$$L(M) := \{w \in T^* \mid (q_0, w, S) \Rightarrow^* (q, \lambda, \lambda)(q \in Q)\}$$

Satz 3.18

Voraussetzung: $M = (Q, T, \Gamma, \delta, q_0, S)$ PDA

Dann ist $L(M)$ kontextfrei.

Beweis zu Satz 3.18

Sei $V := \{{}^pA^q \mid A \in \Gamma, p, q \in Q\} \cup \{S'\}$. Die Menge der Produktionen gewinnt man so:

$$P := \{S' \to {}^{q_0}S^q \mid q \in Q\} \cup \{{}^pA^q \to a^p B_1^{p_1 p_1} B_2^{p_2} \dots {}^{p_k}B_i^q \mid \\ p_1, \dots, p_k \in Q, (q, B_1 B_2 \dots B_i) \in \delta(p, a, A)\}$$

Dann ist mit $G := (V, T, S', P)$ gerade $L(G) = L(M)$. □

Bemerkung

Damit ergibt sich, dass PDAs mit einfachen Kellerautomaten gleichwertig sind. Sie sind eine Verallgemeinerung, bringen allerdings keine größeren Möglichkeiten.

Man kann allerdings zusätzliche Endzustände einführen:

$$L(M) := \{w \in T^* \mid (q_0, w, S) \Rightarrow^* (q, \lambda, \lambda) \text{ mit } q \in F \subseteq Q\}$$

Es gibt Sprachen, die durch deterministische Kellerautomaten *mit* F, aber nicht *ohne* F erkannt werden. PDA mit F erkennen also nicht zwingend alle kontextfreien Sprachen.

4 Typ 1-und Typ 0-Sprachen

4.1 Erweiternde Grammatiken

Definition 4.22 Erweiternde Grammatik

Sei $G = (V, T, S, P)$ eine Grammatik. G heißt erweiternd :⇔ für $u \to v \in P$ gilt $|u| \leq |v|$. Die Produktion $S \to \lambda$ ist zulässig, falls S auf keiner rechten Seite einer Produktion vorkommt.

Satz 4.19

Voraussetzung: L Sprache

Es existiert eine kontextsensitive Grammatik $G = (V, T, S, P)$ mit $L(G) = L \Leftrightarrow$ es existiert eine erweiternde Grammatik $G' = (V', T, S, P')$ mit $L(G') = L$

Beweis zu Satz 4.19

Die Richtung " $\Longrightarrow$ " ist nach Definition klar. Wir zeigen "$\Leftarrow$":

Das Problem ist, dass bei einer kontextsensitiven Grammatik nur immer ein Nichtterminal in etwas aus $(V \cup T)^*$ wechseln darf, während bei erweiternden Grammatiken nur Aussagen über die Länge getroffen sind. Zunächst ersetze alle Terminale $a \in T$ durch neue Nichtterminale A_a. Schreibe V aus V' entsprechend um, ergänze $P := P' \cup \{A_a \to a \mid a \in T\}$. Dies behebt mit dem Holzhammer die Möglichkeit für Produktionen der Art $abc \to cde$ (d.h. nur Terminale), die im kts Fall ebenfalls nicht zulässig sind.

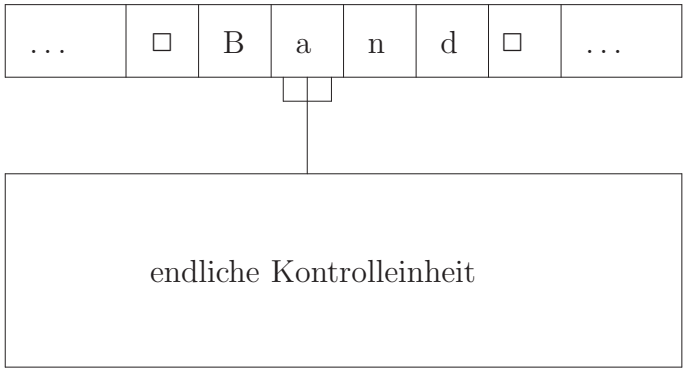
Um nun eine beliebige erweiternde Produktion $A_1 \dots A_m \to B_1 \dots B_n \in P$ in eine kontextsensitive Produktion umzuwandeln, führe neue Nichtterminale $C_1, \dots, C_{m-1}$ ein und ersetze die alte Produktion durch die folgenden neuen:

$$\begin{aligned}
A_1 \dots A_m &\to C_1 A_2 \dots A_m \\
C_1 A_2 \dots A_m &\to C_1 C_2 A_3 \dots A_m \\
&\dots \\
C_1 \dots C_{m-1} A_m &\to C_1 \dots C_{m-1} B_m B_{m+1} \dots B_n \\
C_1 \dots C_{m-1} B_m B_{m+1} \dots B_n &\to C_1 \dots C_{m-2} B_{m-1} B_m B_{m+1} \dots B_n \\
C_1 \dots C_{m-2} B_{m-1} B_m B_{m+1} \dots B_n &\to C_1 \dots C_{m-3} B_{m-2} B_m B_{m-1} \dots B_n \\
&\dots \\
C_1 B_2 \dots B_n &\to B_1 \dots B_n
\end{aligned}$$

Damit ist die Behauptung gezeigt. □

4.2 Turingmaschinen

Definition 4.23 Turingmaschine



Sei Q eine Zustandsmenge, $F \subseteq Q$ eine Endzustandsmenge, $q_0 \in Q$ der Anfangszustand, Γ ein Bandalphabet, $X \subseteq \Gamma$ ein Eingabealphabet, $\square \in \Gamma$ das

Leer- bzw. Bandzeichen und $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$ eine Zustandsübergangsfunktion.

Dann heißt das Septupel $M = (Q, F, q_0, \Gamma, X, \square, \delta)$ eine Turingmaschine. Ein Tripel $K := (\alpha, q, \beta) \in \Gamma^* \times Q \times \Gamma^*$ heißt eine Konfiguration der Turingmaschine, die gleiche Konfiguration wird auch durch $(\square\alpha, q, \beta\square)$ beschrieben. Wie üblich verwenden wir die Symbolik $K \Rightarrow K'$, $K \Rightarrow^* K'$ für den Konfigurationsübergang.

Sei z.B. $\delta(q, x) = (p, y, L)$. Dann wird auf dem Band an die Stelle des x ein y geschrieben, in den Zustand p gewechselt und der Kopf eine Stelle nach links gesetzt.

Die von einer Turingmaschine akzeptierte Sprache ist

$$L(M) := \{w \in X^* \mid (\lambda, q_0, w) \Rightarrow^* (\alpha, q, \beta), q \in F\}$$

Das Band einer Turingmaschine darf beliebig lang werden, die anfängliche Eingabe muss jedoch endlich sein. Turingmaschinen gibt es in den Geschmacksrichtungen deterministisch bzw. nichtdeterministisch und unbeschränkt bzw. linear beschränkt. In letzterem Fall heißt die Maschine auch LBA wie "linear bounded automaton".

Satz 4.20

Voraussetzung: $G = (V, T, S, P)$ kontextsensitive Grammatik

$L(G)$ wird durch einen LBA erkannt.

Beweis zu Satz 4.20

O.B.d.A. $\lambda \notin L(G)$. Setze $\Gamma := T \cup V$, Wir programmieren die Turingmaschine so, dass sie sich (nichtdeterministisch) beliebig ein v mit $u \to v \in P$ auf dem Band auswählt, dieses durch u ersetzt und im Falle $|u| < |v|$ die Eingabe nach rechts zusammenschiebt. Die Turingmaschine geht in einen Endzustand, falls am Schluss nur noch S auf dem Band steht. Wegen der Eigenschaft $|u| \leq |v|$ für $u \to v \in P$ ist die Turingmaschine linear beschränkt.

Bemerkung

Analog zeigt man, dass Typ 0-Sprachen von unbeschränkten Turingmaschinen erkannt werden.

Satz 4.21

Voraussetzung: $M = (Q, F, q_0, \Gamma, X, \square, \delta)$ LBA

Dann ist $L(M)$ kontextsensitiv.

Beweis zu Satz 4.21

Zu einem gegebenen LBA geben wir eine Grammatik G an mit $L(M) = L(G)$
Setze $T := X$,

$$V := \{S, A\} \cup (\Gamma^2) \cup (Q\square\square\Gamma^2) \cup (\square\square Q\Gamma^2) \cup (Q\Gamma^2\square\square) \cup (\Gamma^2 Q\square\square) \cup (\square\square\Gamma^2) \cup (\Gamma^2\square\square)$$

Dabei dienen die speziellen $(*\square*)$-Nichtterminale (anhand ihrer Länge von den anderen unterscheidbar) lediglich dazu, den linken und rechten Rand für den die "Pseudo-Lesekopfposition" anzeigenden Zustand unüberwindlich zu machen.

Wir erzeugen folgende Arten von Produktionen:

- Produktionen, um jede beliebige Anfangskonfiguration den Turingmaschine zu erzeugen:

  $$\{S \to A(aa\square\square) \mid a \in X\}$$
  $$\{A \to A(\square\square qaa) | A(aa) \mid q \in Q, a \in X\}$$

- Produktionen, um jeden möglichen δ-Übergang zu simulieren.

  $$\begin{aligned}
  \{(\square\square qxa) &\to (p\square\square ya),\\
  (\square\square zb)(qxa) &\to (\square\square pzb)(ya),\\
  (zb)(qxa) &\to (pzb)(ya),\\
  (zb)(qxa\square\square) &\to (pzb)(ya\square\square) \mid \delta(q,x) = (p,y,L), z \in \Gamma\} \cup\\
  \{(xaq\square\square) &\to (pxa\square\square) \mid \delta(q,\square) = (p,y,L)\}
  \end{aligned}$$

  wobei die rechts- und neutralläufigen Übergänge analog gebildet werden.

- Produktionen, um das schlussendlich akzeptierte Wort aus dem Backup an den jeweils zweiten Stellen zu rekonstruieren. Wir stellen sicher, dass nur dann ein Terminalwort entsteht, wenn ein Endzustand vorliegt, indem wir nur in diesem Fall die Umwandlung des Nichtterminals mit dem gegenwärtigen "Lesekopf" zulassen:

  $$\begin{aligned}
  \{(q\square\square xa) &\to a,\\
  (\square\square qxa) &\to a,\\
  (qxa) &\to a,\\
  (qxa\square\square) &\to a,\\
  (xaq\square\square) &\to a,\\
  (xa) &\to a,\\
  (\square\square xa) &\to a,\\
  (xa\square\square) &\to a \mid q \in F, x, a \in X\}
  \end{aligned}$$

Insgesamt ergibt sich dann mit $G = (V, X, S, P)$ mit obiger Produktionenmenge P eine kontextsensitive Grammatik mit $L(G) = L(M)$. □

Bemerkung

Indem man in obigem Beweis auch das Schreiben über die Bandgrenzen hinaus zulässt, und die entsprechenden Backup-Bandzeichen dann zu λs umwandelt, erhält man, dass allgemeine Turingmaschinen Sprachen des Typs 0 erkennen.

4.3 Deterministische Turingmaschinen

Satz 4.22

Voraussetzung: $M = (Q, F, q_0, \Gamma, X, \square, \delta)$ nichtdet. Turingmaschine

Dann existiert eine deterministische Turingmaschine M' mit $L(M) = L(M')$.

Beweis zu Satz 4.22

Indem man die deterministische Turingmaschine in einer Art Timesharing für jede Konfiguration der nichtdeterministsichen Maschine jeweils Einzelschritte durchführen lässt. □

Bemerkung

Es ist unbekannt, ob nichtdeterministische LBA in deterministische LBA umgewandelt werden können.

4.4 Abschlussseigenschaften

Bemerkung Zusammenfassung

Es sind abgeschlossen unter folgenden Operationen:

	$\cdot, \cup, *$	$\cap$	^{c}
Typ 0	$\surd$	$\surd$	$\times$
Typ 1	$\surd$	$\surd$	$\surd$
Typ 2	$\surd$	$\times$	$\times$
Typ 3	$\surd$	$\surd$	$\surd$

4.5 Das Halteproblem

Definition 4.24 Gödelnummer

Zum Beispiel durch zeichengebundene Kodierung der formalen Festlegung einer Turingmaschine in einem n-adischen System und anschließende Darstellung der

Kodierung $a_0, \ldots, a_j$ als Zahl

$$q := \sum_{k=0}^{j} a_k n^k$$

erhält man eine Nummer für jede Turingmaschine, genannt deren Gödelnummer bzgl. eines festen Kodierungssystems.

Diese Zuordnung ist offensichtlich nicht zwingend surjektiv (d.h. es existieren Zahlen, zu denen es keine Turingmaschine gibt) und auch nicht zwingend eindeutig (es exisitieren mehrere Kodierungen für eine Turingmaschine), aber injektiv.

Surjektivität und Eindeutigkeit stellt man durch folgende zwei Konventionen wieder her:

- Als das Urbild einer Zahl, der auf bisherige Weise keine Turingmaschine zugeordnet war, betrachten wir die triviale endlos laufende Turingmaschine.
- Man definiert sich eine gewisse Normalform für die Kodierung von Turingmaschinen.

Da unsere Zuordnung mit diesen Veränderungen bijektiv ist, definieren wir M_i als die Turingmaschine mit der Nummer i.

Definition 4.25 Berechenbarkeit

Eine partielle Funktion $f : \mathbb{N}_0 \to \mathbb{N}_0$ heißt berechenbar :$\Leftrightarrow$

Es existiert ein $i \in \mathbb{N}_0$ so, dass für M_i folgendes gilt: Betrachte das erste Eingabezeichen x_i von M_i nach dem Bandzeichen. Man lässt M_i mit der Eingabe

$$\underbrace{x_i x_i \ldots x_i}_{n\text{-mal}}$$

laufen, und es muss gelten:

- M_i hält an mit genau k x_i's auf dem Band $\Leftrightarrow$ $f(n) = k$ und
- M_i hält nicht an $\Leftrightarrow$ $f(n)$ ist undefiniert ($f(n) = \perp$).

Schreibweise: $f = \varphi_i$.

Bei Funktionen, die "Ja/Nein"=0/1-Antworten liefern sollen, spricht man auch von Entscheid- oder Lösbarkeit.

Definition 4.26 Die Funktion halt

Wir definieren

$$\text{halt}(i) := \begin{cases} 1 & \varphi_i(i) \downarrow \\ 0 & \varphi_i(i) \uparrow \end{cases}$$

(Notation: $\downarrow$ = "TM beendet mit Ausgabe", $\uparrow$ = "TM läuft endlos")

Satz 4.23

halt ist nicht berechenbar.

Beweis zu Satz 4.23

Annahme: halt berechenbar. Definiere:

$$\delta(i) := \begin{cases} \bot & \varphi_i(i) \downarrow \\ 1 & \varphi_i(i) \uparrow \end{cases}$$

Elementar (auf der Ebene von Turingmaschinen) überlegt man sich, dass dann auch δ berechen bar ist. Es existiert also ein $j \in \mathbb{N}_0$ so, dass $\varphi_j \equiv \delta$, also für $\varphi_j(j)$ gilt

$$\varphi_j(j) = \begin{cases} \bot & \varphi_j(j) \downarrow \\ 1 & \varphi_j(j) \uparrow \end{cases}$$

Widerspruch! □

4.6 Das Wortproblem

Satz 4.24

Das Wortproblem ist für Sprachen des Typs 0 nicht entscheidbar.

Beweis zu Satz 4.24

Wir können das Halteproblem auf das Typ 0-Wortproblem reduzieren. □

Satz 4.25

Das Wortproblem ist für Sprachen des Typs 1 entscheidbar.

Beweis zu Satz 4.25

$G = (V, T, S, P)$ sei eine Typ 1-Grammatik. Anhand des folgenden Algorithmus kann das Wortproblem für das Wort w entschieden werden: *Ablauf*:

- $M := \emptyset, M' := \{S\}$
- Solange $M \neq M'$
  - $M := M'$

- $M' := M \cup \{\beta \mid \alpha \Rightarrow \beta, \alpha \in M, |\beta| \leq |w|\}$

- $w \in L(G) \Leftrightarrow w \in M$.

□

4.7 Das Post'sche Korrespondenzproblem

Definition 4.27 Post'sches Korrespondezproblem

Ein Postsystem über einem endlichen Alphabet X ist eine Menge

$$\left\{ \binom{x_1}{y_1}, \ldots, \binom{x_n}{y_n} \right\} \subset X^* \times X^*$$

$i_1, \ldots, i_k$ ist eine Lösung des Postsystems, falls gilt

$$x_{i_1} x_{i_2} \ldots x_{i_k} = y_{i_1} y_{i_2} \ldots y_{i_k}$$

Das Postsche Korrespondenzproblem, auch Postian Correspondence Problem oder *PCP*, besteht darin, eine solche Lösung zu finden.

Beispiel Zwei Beispiel-Postsysteme

Das Postsystem

$$\left\{ \binom{1}{01}, \binom{\lambda}{11}, \binom{0101}{1}, \binom{111}{\lambda} \right\} \subset X^* \times X^*$$

hat eine Lösung $x_2x_1x_1x_3x_2x_4 = 110101111 = y_2y_1y_1y_3y_2y_4$.

Die kürzeste Lösung für das Postsystem

$$\left\{ \binom{001}{0}, \binom{01}{011}, \binom{01}{101}, \binom{10}{001} \right\} \subset X^* \times X^*$$

besteht aus 66 Indizes.

Satz 4.26

Voraussetzung: $G = (V, T, S, P)$ Grammatik

Dann existiert für jedes Wort $w \in L(G)$ ein Postsystem $K(w)$ mit

$$K(w) \text{ lösbar} \Leftrightarrow w \in L(G) \text{ lösbar}$$

Beweis zu Satz 4.26

Wir konstruieren K über $X := V \cup V' \cup T \cup T' \cup \{*, *', [,]\}$, wobei y' eine (unterscheidbare) Kopie von y bezeichnet. K bestehe aus

$$\left\{ \binom{[S*}{[}, \binom{]}{*'w]}, \binom{]}{*w']}, \binom{a}{a'}, \binom{a'}{a}, \binom{v}{u'}, \binom{v'}{u} \mid a \in X, u \to v \in P \right\}$$

Jede Lösung hat zwingend folgende Gestalt:

$$\left(\begin{array}{c|c|c|c|c|c|c} [S* & w_1' & *' & w_2 & \dots & * & w' \\ [& S & * & w_1' & \dots & *' & w_n \end{array} \left| \begin{array}{c}] \\ (*w'|\,*'\,w)] \end{array} \right.\right)$$

d.h. man kann aus ihr eine Ableitungskette

$$S \Rightarrow w_1' \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w$$

ablesen. □

Bemerkung Berechenbarkeit von Lösungen des PCP

Wäre das Postsche Korrespondenzproblem lösbar, so wäre auch das Wortproblem lösbar. Das Wortproblem ist nicht lösbar, also ist auch das PCP nicht entscheidbar.

4.8 Entscheidbarkeit bei Grammatiken

Satz 4.27

Voraussetzung: $G = (V, T, S, P)$ eine Typ-2-Grammatik

Es ist nicht entscheidbar, ob G nur eindeutige Ableitungen hat.

Beweis zu Satz 4.27

Wir beweisen die Aussage durch Reduktion des PCP auf das Eindeutigkeitsproblem. Sei nun

$$\left\{ \binom{x_1}{y_1}, \dots, \binom{x_n}{y_n} \right\}$$

ein Postsystem über K. Wir betrachten weiter die kontextfreie Grammatik

$$\begin{array}{lcl} S & \rightarrow & A|B \\ A & \rightarrow & x_1A1|x_2A2|\dots|x_nAn|\lambda \\ B & \rightarrow & y_1B1|y_2B2|\dots|y_nBn|\lambda \end{array}$$

Die erzeugten Worte sind von der Form

$$x_{i_1} \dots x_{i_k} i_k \dots i_1 \text{ oder } y_{i_1} \dots y_{i_k} i_k \dots i_1$$

G hat mehr als eine Ableitung für ein einziges Wort genau dann, wenn das PCP lösbar ist. Das PCP ist aber nicht entscheidbar, also ist auch das Eindeutigkeitsproblem entscheidbar. □

Satz 4.28

Das Schnittproblem ist für kontextfreie Sprachen unentscheidbar.

Beweis zu Satz 4.28

Wir beweisen die Aussage durch Reduktion des PCP auf das Schnittproblem. Sei nun

$$\left\{\binom{x_1}{y_1}, \ldots, \binom{x_n}{y_n}\right\}$$

ein Postsystem über K. Wir betrachten weiter die kontextfreie Grammatik

$$\begin{array}{lcl} S & \rightarrow & A|B \\ A & \rightarrow & x_1A1|x_2A2|\ldots|x_nAn|x_11|x_22|\ldots x_nn \\ B & \rightarrow & y_1B1|y_2B2|\ldots|y_nBn|y_11|y_22|\ldots y_nn \end{array}$$

Dann sind

$$L_1 := \{\text{aus } A \text{ ableitbare Worte}\}, L_2 := \{\text{aus } B \text{ ableitbare Worte}\}$$

kontextfreie Sprachen, und Elemente aus $L_1 \cap L_2$ sind eine Lösung des PCP. Wäre das Schnittproblem entscheidbar, dann auch das PCP. Dies ist nicht der Fall, also ist das Schnittproblem nicht entscheidbar. □

Bemerkung Kontextfreie Sprachen mit kontextfreien Komplementen

Bei kontextfreien Sprachen mit kontextfreien Komplementen ist das Schnittproblem ebensowenig entscheidbar, da die Grammatik in obigem Beweis eine Sprache mit kontextfreiem Komplement erzeugt.

Bemerkung Folgerung

Das Schnittproblem ist für Sprachen des Typs $i \leq 2$ nicht entscheidbar.

Satz 4.29

Voraussetzung: $G = (V, T, S, P)$ kontextsensitiv

$L(G) = \emptyset$ ist nicht entscheidbar.

Beweis zu Satz 4.29

Wir reduzieren das Schnittproblem für kontextsensitive Grammatiken auf die Leerheit einer kontextsensitiven Grammatik und nehmen ohne Beweis an, dass der Schnitt zweier kontextsensitiven Sprachen wieder kontextsensitiv ist. Dann gilt

$$L(G) = \emptyset \Leftrightarrow L(G) \cap L(G) = \emptyset$$

Das Schnittproblem ist nicht entscheidbar, also ist auch die Leerheit nicht entscheidbar. □

Satz 4.30

Voraussetzung: $G_1 = (V_1, T, S_1, P_1)$,$G_2 = (V_2, T, S_2, P_2)$ kontextfrei

$L(G_1) = L(G_2)$ ist nicht entscheidbar.

Beweis zu Satz 4.30

Seien L_1, L_2 kontextfreie Sprachen mit kontextfreiem Komplement. (deren Schnittproblem ist, wie oben gezeigt, nicht entscheidbar) Wäre das Gleichheitsproblem entscheidbar, so könnte man den Schnitt aber wegen $L_1 \cap L_2 = \emptyset \Leftrightarrow L_1 \cup L_2^c = L_2^c$ entscheiden. Also ist die Gleichheit für ktf Sprachen mit ktf Komplementen nicht entscheidbar, deswegen erst recht nicht für allgemeine ktf Sprachen.

5 Fixpunkttheorie

Definition 5.28 Halbordnung

Eine Relation "$\leq$" auf einer Menge M heißt Halb- oder Teilordnung bzw. partielle Ordnung :$\Leftrightarrow$

(1) $(\forall x \in M)(x \leq x)$ (Reflexivität)

(2) $(\forall x, y, z \in M)(x \leq y, y \leq z \implies x \leq z)$ (Transitivität)

(3) $(\forall x, y, z \in M)(x \leq y, y \leq x \implies x = y)$ (Antisymmetrie)

$(M, \leq)$ heißt dann eine halbgeordnete Menge.

Definition 5.29 Kleinstes Element

Sei $(M, \leq)$ eine halbgeordnete Menge. Dann heißt $k \in M$ kleinstes Element von $M :\Leftrightarrow k \leq x$ für alle $x \in M$.

Definition 5.30 Obere Schranke

Sei $(M, \leq)$ eine halbgeordnete Menge. Dann heißt $o \in M$ obere Schranke von $T \subseteq M :\Leftrightarrow x \leq o$ für alle $x \in T$.

Definition 5.31 Supremum

$s \in M$ heißt Supremum von $T \subseteq M :\Leftrightarrow s$ ist kleinste obere Schranke von T.

Bemerkung Eindeutigkeit

Suprema sind eindeutig.

Definition 5.32 Kette

Sei $(M, \leq)$ eine halbgeordnete Menge. Eine Folge $(x_i)_{i=1}^{\infty} \subseteq M$ heißt Kette $:\Leftrightarrow$ $x_i \leq x_{i+1}$ für $i \geq 1$.

Definition 5.33 Vollständige halbgeordnete Menge

Sei $(M, \leq)$ eine halbgeordnete Menge. M heißt vollständig $:\Leftrightarrow$

(1) Es existiert ein kleinstes Element in M.

(2) Jede Kette in M hat ein Supremum in M.

Definition 5.34 Stetige Funktion

Seien $(A, \leq), (B, \leq)$ vollständige halbgeordnete Mengen. Dann heißt $f : A \to B$ stetig, $\Leftrightarrow$ für jede Kette $(x_i)_{i=1}^{\infty} \subseteq A$ gilt

$$\sup_{i=1}^{\infty} f(x_i) = f(\sup_{i=1}^{\infty} x_i)$$

Bemerkung

Im analytischen Sinne sind unsere stetigen Funktionen monoton wachsend und linksseitig stetig auf abgeschlossenen Intervallen.

Definition 5.35 Monotone Funktion

Seien $(A, \leq), (B, \leq)$ halbgeordnete Mengen. Dann heißt $f : A \to B$ monoton, $\Leftrightarrow$ für alle $x, y \in A$ mit $x \leq_A y$ gilt auch $f(x) \leq_B f(y)$

Lemma 5.31

Voraussetzung: $(A, \leq), (B, \leq)$ vollständige halbgeordnete Mengen, $f : A \to B$ stetig

Dann ist f monoton.

Beweis zu Lemma 5.31

Sei $a \leq_A b$ in A. Betrachte $x_1 := a$, $x_2 := x_3 \ldots := b$.

f ist stetig, woraus folgt $\sup_n f(x_n) = f(\sup_n x_n) = f(b)$ Also gilt $f(a) = f(x_1) \leq f(b)$. □

Definition 5.36 Fixpunkt

Sei $(A, \leq)$ eine vollständige halbgeordnete Menge und $f : A \to A$ eine Abbildung. $a \in A$ heißt Fixpunkt von $f :\Leftrightarrow f(a) = a$.

Satz 5.32 Kleene'scher Fixpunktsatz

Voraussetzung: $(A, \leq)$ vollständige halbgeordnete Menge, $f : A \to A$ stetig

Sei m das kleinste Element von A. Dann ist $\sup_{n\in\mathbb{N}} f^n(m)$ der kleinste Fixpunkt von f. Insbesondere existiert mindestens ein Fixpunkt.

Beweis zu Satz 5.32

Wir zeigen zunächst, dass $\sup f^n(m)$ existiert:

$$m \leq f(m) \implies f(m) \leq f^2(m) \implies \ldots \implies f^n(m) \leq f^{n+1}(m)$$

Also ist $(f^n(m))_{n\in\mathbb{N}}$ eine Kette. Es folgt direkt die Existenz des Supremums.

Weiter zeigen wir: Das Supremum von $f^n(m)$ ist ein Fixpunkt:

$$f(\sup(f^n(m))) = \sup(f^{n+1}(m)) = \sup(f^n(m))$$

Und schließlich zeigen wir, dass $\sup f^n(m)$ kleinster Fixpunkt ist, angenommen $c \in A$ sei ein kleinerer Fixpunkt:

$$m \leq c \implies f(m) \leq f(c) = c \implies \ldots \implies f^n(m) \leq f(c) = c$$

□

5.1 Der Fixpunktsatz

Bemerkung Kurzschreibweise von Wortmengen

Analog zu den regulären Ausdrücken schreiben wir z.B.

$$a\{b, c\}d := \{abd, acd\}$$

Beachte:

$$a\emptyset d := \emptyset$$

Definition 5.37 Funktion einer kontextfreien Grammatik

Sei $G = (V, T, A_1, P)$ mit $V = \{A_1, \ldots, A_n\}$ und

$$P = \{A_i \to a_{i1}, \ldots, a_{ik_i} \mid i = 1, \ldots, n, a_{ij} \in (T \cup V)^*\}$$

eine kontextfreie Grammatik.

Definiere

$$g_i : \mathcal{P}(T^*)^n \to \mathcal{P}(T^*)$$

mit $g_i(M_1, \ldots, M_n) := \overline{a_{i1}} \cup \ldots \cup \overline{a_{ik_i}}$. Ist z.B. $a_{ik} = \alpha A_j \beta$, so ist $\overline{a_{ik}} := \alpha M_j \beta$. Beachte: a_{ik} ist eine Zeichenkette, $\overline{a_{ik}}$ ist eine Menge.

Dann heißt $g := (g_1, \ldots, g_n)$ die Funktion der kontextfreien Grammatik G.

Konvention für Kapitel 5

Ab hier betrachten wir als Teilordnung die Ordnung "$\subseteq$" auf Mengen $M \in \mathcal{P}(T^*)$ über einem Alphabet T: $M \leq N :\Leftrightarrow M \subseteq N$.

Für Tupel $(M_1, \ldots, M_n) \in \mathcal{P}(T^*)^n$ von solchen Mengen gelte dabei die Ordnung

$$(M_1, \ldots, M_k) \leq (N_1, \ldots, N_k) :\Leftrightarrow (\forall i \in \{1, \ldots, k\})(M_i \leq N_i)$$

Lemma 5.33

Eine zusammengesetzte Funktion ist stetig $\Leftrightarrow$ ihre Komponentenfunktionen sind stetig.

Lemma 5.34

Die Funktion einer kontextfreien Grammatik ist stetig.

Beweis zu Lemma 5.34

Wegen Lemma 5.33 muss nur noch die Stetigkeit einer Komponentenfunktion g_i gezeigt werden. Sei $(M^{(k)})_{k=1}^{\infty} = (M_1^{(k)}, \ldots, M_n^{(k)})_{k=1}^{\infty}$ eine Kette. Zu zeigen ist $g_i(\sup_k M^{(k)}) = \sup_k g_i(M^{(k)})$. Mit den Bezeichnungen aus der Definition von g gilt:

$$\begin{aligned}
w \in g_i(\sup_k M^{(k)}) &\Leftrightarrow w \in g_i(\sup_k (M_1^{(k)}, \ldots, M_n^{(k)})) \\
&\Leftrightarrow w \in g_i(\sup_k M_1^{(k)}, \ldots, \sup_k M_n^{(k)}) \\
&\Leftrightarrow (\exists j \in \{1, \ldots, k_i\})(w \in \overline{a_{ij}} \overset{\text{z.B.}}{=} \alpha(\sup_k M_l^{(k)})\beta) \\
&\Leftrightarrow (\exists k \in \mathbb{N})(\exists j \in \{1, \ldots, k_i\})(w \in \overline{a_{ij}} \overset{\text{z.B.}}{=} \alpha(M_l^{(k)})\beta) \\
&\Leftrightarrow (\exists k \in \mathbb{N})(w \in g_1(M^{(k)})) \Leftrightarrow w \in \sup_k g_1(M^{(k)})
\end{aligned}$$

□

Satz 5.35

Voraussetzung: $G = (V, T, S, P)$ eine kontextfreie Grammatik, g die Funktion dazu

Sei dann

$$L_i := \{w \in T^* \mid A_i \Rightarrow^* w, A_i \in V\} \qquad i = 1, ldots, n$$

also z.B. $L_1 = L(G)$, falls $A_1 = S$. Dann ist $(L_1, \ldots, L_n)$ der kleinste Fixpunkt von g.

Beweis zu Satz 5.35

Offenbar ist $\emptyset \in \mathcal{P}(T^*)$ das Minimum von $\mathcal{P}(T^*)$ bezüglich der Teilmengenordnung. Weiter ist

$$\sup_k g(\emptyset, \ldots, \emptyset) = (L_1, \ldots, L_n)$$

und damit ergibt sich nach Satz 5.32 sofort die Behauptung. □

5.2 Gleichungssysteme

Satz 5.36

Voraussetzung: $A, B \in \mathcal{P}(T^*)$, T Alphabet

Für das Gleichungssystem $X = AX + B$ ist A^*B die kleinste Lösung.

Beweis zu Satz 5.36

Wir betrachten einfach $g(X) = AX + B$. Der kleinste Fixpunkt ist

$$\sup_k g^k(\emptyset) = B + AB + A^2B + A^3B + \cdots = A^*B$$

Hallelujah! □

Beispiel

Sei $G := (\{A_1, A_2\}, \{a, b\}, A_1, P)$ mit

$$P := A_1 \to aA_1|bA_2|a, A_2 \to aA_1|aA_2|b$$

Wir haben also gegeben

$$\begin{aligned} x_1 &= ax_1 + bx_2 + a \\ x_2 &= ax_1 + ax_2 + b \end{aligned}$$

mit $x_1, x_2 \in \mathcal{P}((a + b)^*)$. Es folgt nach Satz Satz 5.36

$$x_1 = a^*(bx_2 + a),$$

woraus weiter folgt

$$x_2 = a^+(bx_2 + a) + ax_2 + b = a^+bx_2 + ax_2 + aa^+b = (a^+b + a)x_2 + aa^+ + b$$

Erneutes Anwenden von Satz Satz 5.36 ergibt

$$x_2 = (a^+b + a)^*(aa^+ + b)$$

Einsetzen führt auf

$$x_1 = a^*(b(a^+b + a)^*(aa^+ + b) + a) = a^*b(a^+b + a)^*(aa^+ + b) + a^+$$

Also folgt

$$L(G) = a^*b(a^+b + a)^*(aa^+ + b) + a^+$$

6 Syntaxanalyse

A GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five). C. State on the Title page the name of the publisher of the Modified Version, as the publisher. D. Preserve all the copyright notices of the Document. E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice. H. Include an unaltered copy of this License. I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. K. In any section entitled "Acknowledgements" or "Dedications", preserve the section’s title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version. N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Index

Sätze und Definitionen

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

V

W

Z