Eine Mitschrift zu der Vorlesung

„Theoretische Informatik 2“

gehalten an der

Johann Wolfgang von Goethe Universität Frankfurt

von

Herr Prof. Dr. Wotschke

Fassung vom 19. Oktober 2000

Autoren:

Frank Bergmann (bergmann@informatik.uni-frankfurt.de),

Jörn Gersdorf (gersdorf@informatik.uni-frankfurt.de),

Martin Klossek (klossek@informatik.uni-frankfurt.de),

Fabian Wleklinski (wleklins@informatik.uni-frankfurt.de)

1 Inhaltsverzeichnis

2 Vorlesung vom 11. April 2000

2.1 Mengen, Mengenoperationen und Relationen

2.1.1 Funktion, Umkehrfunktionen und Abbildungen

Eine Funktion ist eine eindeutige Zuordnung der Elemente einer Menge *A* zu den Elementen einer Menge *B*. Jedem Element von *A* darf höchstens ein Element von *B* zugeordnet sein, verschiedenen Elementen von *A* kann aber dasselbe Element von *B* zugeordnet sein. Es braucht nicht jedem Element von *A* ein Element aus *B* zugeordnet zu sein.

Für beliebige Funktionen gilt:

$$\begin{aligned} &A, B \text{ Mengen}, A \subseteq S_1, B \subseteq S_2, f : S_1 \to S_2 \\ &f(A) \underset{\text{Df}}{=} \{ f(x) \mid x \in A \} \\ &f'(B) \underset{\text{Df}}{=} \{ x \mid x \in S_1, f(x) \in B \} \end{aligned} \tag{2.1}$$

$$\begin{aligned} &A \subseteq f'(f(A)) \\ &B \supseteq f(f'(B)) \end{aligned} \tag{2.2}$$

$$\begin{aligned} &f(A_1 \cup A_2) = f(A_1) \cup f(A_2) \\ &f(A_1 \cap A_2) \subseteq f(A_1) \cap f(A_2) \end{aligned} \tag{2.3}$$

Anmerkungen dazu:

Es ist zu beachten, dass *f'()* keinesfalls die Umkehrfunktion ist – zu dieser wird es erst durch die bijektive Eigenschaft.

Bei der obigen Betrachtung von *A*, *B*, S_1, S_2 und *f()* existieren drei Sonderfälle. 1) Ein Element aus *A* kann nicht nach *B* abgebildet werden, weil f() für dieses Element nicht definiert ist. 2) Für ein bestimmtes Element aus *B* gibt es kein Element aus *A*, das auf dieses Element abgebildet wird. 3) Zwei oder mehr Elemente aus *A* werden auf ein Element in *B* abgebildet. (Die vierte, logische Kombination, nämlich dass ein Element aus *A* auf mehrere Elemente aus *B* abgebildet wird, verbietet der Funktionsbegriff.)

2.1.2 Injektivität, Surjektivität und Bijektivität

Die Begriffe Injektivität, Surjektivität und Bijektitivät entstammen dem mathematischen Sprachgebrauch und müssen daher hier nicht näher erläutert werden. Bedeutsam sind aber die Auswirkungen auf die oben gemachten Aussagen:

$$\begin{aligned} &f \text{ injektiv} \Rightarrow A = f'(f(A)) \\ &f \text{ injektiv} \Rightarrow f(A_1 \cap A_2) = f(A_1) \cap f(A_2) \end{aligned} \tag{2.4}$$

$$\begin{aligned} &f \text{ surjektiv} \Rightarrow B = f(f'(B)) \\ &f \text{ surjektiv} \Rightarrow f'(B_1 \cap B_2) = f'(B_1) \cap f'(B_2) \end{aligned} \tag{2.5}$$

$$f \text{ bijektiv} \Rightarrow f^{-1} \text{ (Inverse Funktion) existiert mit } \mathit{Identität} = f \circ f' = f' \circ f \tag{2.6}$$

Generell gilt, es existiert eine bijektive Abbildung zwischen zwei Mengen genau dann, wenn ihre Mächtigkeit identisch ist:

$$|A| = |B| \Leftrightarrow \exists \text{ eine bijektive Funktion } f \text{ mit } f : A \to B \tag{2.7}$$

2.1.3 Relationen

Eine binäre Relation *R* auf einer Menge *M* ist eine Teilmenge von *M*x*M*. Sind $a,b \in M$ und gilt $(a,b) \in R$, so sagt man: „*a* und *b* stehen in der Relation *R*.“ Gelegentlich schreibt man statt $(a,b) \in R$ auch aRb. Eine Relation kann über eine oder mehrere der folgenden Eigenschaften verfügen:

1. Reflexitivität: $\forall a \in M : (a,a) \in R$
2. Transitivität: $\forall a,b,c \in M : (a,b) \in R \wedge (b,c) \in R \Rightarrow (a,c) \in R$
3. Symmetrie: $\forall a,b \in M : (a,b) \in R \Rightarrow (b,a) \in R$
4. Antisymmetrie: $\forall a,b \in M : (a,b) \in R \wedge (b,a) \in R \Rightarrow a = b$

Beachte: Symmetrie und Transitivität implizieren nicht Reflexitivität! Zur Bewertung einer Relation muss jede dieser vier Aussagen einzeln überprüft werden.

2.1.4 Äquivalenzrelationen

Man spricht von einer „*Äquivalenzrelation*“, wenn eine Relation reflexiv, transitiv und symmetrisch ist, und von einer Ordnung, wenn eine Relation reflexiv, transitiv und antisymmetrisch ist. Zum Beispiel ist die normale Vergleichsoperation auf der Menge der reellen Zahlen („=“) eine Äquivalenzrelation, und die kleiner-gleich-Operation („≤“) eine Ordnung.

2.1.5 Äquivalenzklassen

Alle Elemente *b* einer Menge *M*, die zu einem gegebenen Element *a* äquivalent sind, bilden zusammen die „*Äquivalenzklasse*“ des Elementes *a*:

$$[a] \underset{Df}{=} \{b \mid b \in A \wedge {}^\wedge b \,\square\, a\} \tag{2.8}$$

Ist die Schnittmenge zweier Äquivalenzklassen nicht leer, so sind die Äquivalenzklassen identisch:

$$[a] \cap [b] \neq \varnothing \Rightarrow [a] = [b] \tag{2.9}$$

Bei den Äquivalenzklassen kommt die reflexive Eigenschaft der Relationen zum Tragen, da ohne diese Eigenschaft ein Element nicht in seiner eigenen Äquivalenzklasse enthalten wäre.

Es besteht ein Zusammenhang zwischen den Indizes der Äquivalenzklassen und der Anzahl der Zustände eines Automaten.

2.1.6 Kollektionen

Eine Kollektion $C = \{C_i \mid i \in I\}$ ist eine Menge von Elementen C_i, die über einen Index *i* indiziert werden können, der aus der Indiziermenge *I* stammt.

2.1.7 Partitionen

Eine Kollektion von Mengen C_i heißt *Partition* von *A* genau dann wenn 1) alle Mengen C_i disjunkt oder gleich sind, 2) jede Menge C_i eine Teilmenge von *A* ist, und 3) die Vereinigung aller Mengen C_i die Menge *A* ergibt:

$$C \text{ ist Partition von } A \underset{Df}{\Leftrightarrow} \begin{array}{l} 1)\ \forall i,j \in I : C_i = C_j \vee (C_i \cap C_j = \varnothing) \\ 2)\ \forall i \in I : C_i \subseteq A \\ 3)\ \bigcup\limits_{i \in I} C_i = A \end{array} \tag{2.10}$$

Die Menge der Äquivalenzklassen aller Elemente einer gegebenen Menge A ist eine Partition von A:
$C = \{[a] | a \in A\}$ ist eine Partition von A.

2.2 Algebraische Strukturen

2.2.1 Die algebraische Struktur

Die algebraische Struktur ist wichtig, da sich jeder endlicher Automat als algebraische Struktur darstellen lässt. Die algebraische Struktur besteht aus einer Menge, und aus einer Rechenoperation auf dieser Menge:

$$\langle S, \circ \rangle \text{ ist eine algebraische Struktur} \underset{Df}{\Leftrightarrow} \begin{array}{l} 1)\ S \text{ ist eine Menge} \\ 2)\ \circ : SxS \to S \end{array} \tag{2.11}$$

Schreibweise: $\circ(a,b)$ oder $a \circ b$.

2.2.2 Die Halbgruppe

Die Halbgruppe ist eine algebraische Struktur, für deren Operation das Assoziativgesetz gilt:

$$\langle S, \circ \rangle \text{ ist eine Halbgruppe} \underset{Df}{\Leftrightarrow} \begin{array}{l} 1)\ \langle S, \circ \rangle \text{ ist eine algebraische Struktur} \\ 2)\ \forall s_1, s_2, s_3 \in S : \circ(\circ(s_1, s_2), s_3) = \circ(s_1, \circ(s_2, s_3)) \end{array} \tag{2.12}$$

2.2.3 Das Monoid

Das Monoid ist eine Halbgruppe mit Einselement:

$$\langle S, \circ \rangle \text{ ist ein Monoid} \underset{Df}{\Leftrightarrow} \begin{array}{l} 1)\ \langle S, \circ \rangle \text{ ist eine Halbgruppe} \\ 2)\ \exists e \in S\ \forall s \in S : e \circ s = s \circ e = s \end{array} \tag{2.13}$$

2.2.4 Das freie Erzeugendensystem, das freie Monoid

G ist ein freies Erzeugendensystem für $\langle S, \circ \rangle$, wenn 1) jedes Element aus S als Kombination von Elementen aus G darstellbar ist, und 2) jede dieser Darstellungen eindeutig ist.

$$\begin{array}{l} G \text{ ist ein freies Erzeugendensystem für } \langle S, \circ \rangle \underset{Df}{\Leftrightarrow} \\ 1)\ \forall s \in S \setminus \{e\} \exists g_1, ..., g_n \in G : s = g_1 \circ ... \circ g_n \\ 2)\ s = g_{i_1} \circ ... \circ g_{i_n} \wedge s = g_{j_1} \circ ... \circ g_{j_n} \Rightarrow n = m \wedge g_{i_k} = g_{j_k} \quad 1 \le k \le n \end{array} \tag{2.14}$$

Schreibweise: „G ist ein freies Erzeugendensystem für $\langle S, \circ \rangle$“ oder „$\langle S, \circ \rangle$ ist ein freies Monoid über G“.

Beispiele:

$\langle N, + \rangle$ ist ein freies Monoid über {1}, $\langle N, + \rangle$ ist kein freies Monoid über {1,2}

2.2.5 Der Homomorphismus

Ein *Homomorphismus* ist die Kombination zweier algebraischer Strukturen $\langle S_1, \circ \rangle$, $\langle S_2, \bullet \rangle$ und einer Funktion *h()*, die die Eigenschaft besitzt, kommutativ bezüglich der Verknüpfungsoperationen (der algebraischen Strukturen) zu sein:

$$\begin{array}{l} h \text{ ist ein Homomorphismus} \Leftrightarrow \\ 1)\ h : S_1 \to S_2 \\ 2)\ \forall a_1, a_2 \in S_1 : h(a_1 \circ a_2) = h(a_1) \bullet h(a_2) \end{array} \tag{2.15}$$

2.2.6 Der Isomorphismus

Ein *Isomorphismus* ist ein bijektiver Homomorphismus. Wenn zwei Mengen isomorph zueinander sind, bedeutet das, dass sie sich (von den Bezeichnern ihrer Elemente abgesehen) nicht voneinander unterscheiden lassen. Ein minimaler Automat ist „bis auf Isomorphismen exakt bestimmt".

$$\begin{aligned} &h \text{ ist ein Isomorphismus} \Leftrightarrow \\ &\quad 1)\ h : \langle S_1, \circ \rangle \to \langle S_2, \bullet \rangle \text{ ist ein Homomorphismus} \\ &\quad 2)\ h : S_1 \to S_2 \text{ ist bijektiv} \end{aligned} \tag{2.16}$$

Beispiele:

$h(n)=2^n$: h ist Homomorphismus, aber nicht Isomorphismus

2.2.7 Abzählbarkeit

Eine Menge M heißt „*abzählbar unendlich*", wenn es eine bijektive Abbildung ihrer Elemente auf die Menge der natürlichen Zahlen gibt:

$$M \text{ ist abzählbar unendlich} \Leftrightarrow \exists\, f : M \to N \,|\, f \text{ ist bijektiv} \tag{2.17}$$

$$M \text{ ist überabzählbar unendlich} \Leftrightarrow \neg\, M \text{ ist abzählbar unendlich} \tag{2.18}$$

Anmerkung: Wie es der Name sagt, bedeutet „Abzählbarkeit", dass die Elemente einer Menge zwar zahlreich oder unendlich sind, dass sie aber dennoch abzählbar sind.

Beispiel:

Die Menge der rationalen Zahlen ist abzählbar unendlich, die Menge der irrationalen Zahlen (und damit auch die Menge der reellen Zahlen) ist überabzählbar unendlich.

2.2.8 Konkatenation

Die Konkatenation zweier Mengen S_1 und S_2 ist die Menge aller „zusammengesetzten" Elemente dieser beiden Mengen:

$$S_1 \bullet S_2 \underset{Df}{=} \left\{ uv \,\middle|\, u \in S_1, v \in S_2 \right\} \tag{2.19}$$

2.3 Sprachen

Sei Σ eine endliche Menge von Symbolen, und $\bullet$ der Operator für die Konkatenation. Eine Sprache L ist definiert als Teilmenge der Menge aller Wörter über diesem Symbolalphabet:

$$\begin{aligned} &\Sigma^0 \underset{Df}{=} \{e\} && \text{(das leere Wort)} \\ &\Sigma^1 \underset{Df}{=} \Sigma && \text{(Wörter der Länge 1)} \\ &\vdots \\ &\Sigma^n \underset{Df}{=} \Sigma^{n-1} \bullet \Sigma && \text{(Wörter der Länge n)} \end{aligned} \tag{2.20}$$

$$\begin{aligned} &\Sigma^* \underset{Df}{=} \bigcup_{n=0}^{\infty} \Sigma^n && \text{(Alle Wörter über } \Sigma\text{)} \\ &\Sigma^+ \underset{Df}{=} \bigcup_{n=1}^{\infty} \Sigma^n && \text{(Alle Wörter über } \Sigma \text{ außer dem leeren Wort)} \end{aligned} \tag{2.21}$$

$$L \text{ ist eine Sprache über dem Alphabet } \Sigma \underset{Df}{\Leftrightarrow} L \subseteq \Sigma^* \tag{2.22}$$

$\langle \Sigma^*, \bullet \rangle$ ist das freie Monoid über Σ, wobei die Konkatenation („$\bullet$") die Operation ist. Σ dient hier als freies Erzeugendensystem.

Die Menge aller Wörter über einem Alphabet Σ (entspricht Σ^*) ist abzählbar unendlich; die Menge aller Sprachen über Σ ist überabzählbar unendlich.

2.4 Grammatiken

Die zulässige Form der Sätze einer Sprache nennt man *Syntax*, die Bedeutung der Sätze wird durch die *Semantik* beschrieben. Zur Festlegung der Syntax einer Spracheverwendet man Grammatiken. Eine Grammatik ist eine Menge von Regeln, die bestimmen, welche Sätze zur Sprache gehören, und welche nicht. Man spricht von *Chomsky-Grammatiken*, wenn eine Grammatik durch vier Angaben definiert wird: durch eine Menge von *Terminalsymbolen*, eine Menge von *Nichtterminalsymbolen*, eine Menge von Grammatikregeln und durch ein Startsymbol.

Von einer Grammatik wird gefordert, dass sie ein endlich langer Beschreibungsmechanismus für eine Sprache ist. Solche endlich langen Beschreibungsmechanismen können der Länge nach, und innerhalb einer Länge alphabetisch geordnet werden. Somit ist die Menge der Grammatiken abzählbar unendlich. Weil die Menge der Sprachen aber überabzählbar unendlich ist, folgt daraus, dass es Sprachen gibt, die keine Grammatik haben können.

Wie viele Sprachen gibt es, die eine Grammatik haben?

Weil es insgesamt nur abzählbar unendlich viele Grammatiken gibt, ist auch die Menge aller Grammatiken für eine beliebige Sprache über Σ nur abzählbar unendlich. Daraus folgt, dass die Menge der Sprachen mit Grammatik höchstens ebenfalls abzählbar unendlich ist. „Höchstens“ deswegen, weil u.U. mehrere Grammatiken die selbe Sprache beschreiben.

Wie viele Sprachen gibt es, die keine Grammatik haben?

Es gibt überabzählbar unendlich viele Sprachen, die keine Grammatik haben. Der Beweis gestaltet sich recht einfach. Wir haben bereits festgestellt, dass es nur abzählbar unendlich viele Sprachen gibt, die eine Grammatik haben. Gäbe es ebenfalls nur abzählbar unendlich viele Sprachen, die keine Grammatik haben, so würden auch insgesamt nur abzählbar unendlich viele Sprachen existieren. Wir haben aber bereits festgestellt, dass überabzählbar unendlich viele Sprachen existieren (Widerspruch!).

2.5 Klassifizierung von Sprachen

Noam Chomsky hat 1956 eine Klassifizierung der künstlichen Sprachen vorgeschlagen, die sich bis heute erhalten hat. Sie sieht vor, Grammatiken in insgesamt vier verschiedene Gruppen zu kategorisieren. Jede Gruppe verfeinert die vorhergehende Gruppe, bzw. beinhaltet sie. Zu diesen vier Gruppen hat sich im Laufe der Zeit eine fünfte Gruppe gesellt:

Klassifizierung nach Chomsky	Verschiedene Methoden (Klassen), Automaten (Klassen)	Grammatiken
Chomsky-3	Endliche Automaten - deterministisch - nichtdeterministisch	Reguläre Grammatiken
Chomsky-2	Pushdown-Automaten (Kellerautomaten)	Kontext-freie Grammatiken
	Deterministische Pushdown-Automaten	LR(z) – Grammatiken
Chomsky-1	Linear beschränkte Automaten	Kontext-sensitive Grammatiken
Chomsky-0	Turing-Maschinen	Rekursiv aufzählbare Mengen

3 Vorlesung vom 13. April 2000

3.1 Abzählbarkeit von Mengen, Diagonalisierung

3.1.1 Mächtigkeit von Mengen

Mengen sind endliche oder unendliche Ansammlungen von Elementen. Die Anzahl dieser Elemente einer Menge wird *Mächtigkeit* oder auch *Kardinalität* genannt. Die formale Notation dafür ist:

$$|M| = \text{Mächtigkeit der Menge M} \tag{3.1}$$

Haben zwei Mengen M_1 und M_2 die gleiche Mächtigkeit und existiert zwischen den beiden Menge eine bijektive Abbildung, gilt also

$$\begin{aligned} &|M_1| = |M_2| \\ &f: M_1 \to M_2 \,|\, \text{f ist bijektiv} \end{aligned} \tag{3.2}$$

so heißen die Mengen *gleichmächtig*. Für den Spezialfall, dass eine Menge M gleichmächtig der Menge der natürlichen Zahlen $\mathbb{N}$ ist, so ist die Menge M *abzählbar unendlich*. Endliche Mengen sind immer *abzählbar*. Dementsprechend wird eine Menge, deren Mächtigkeit größer der Menge der natürlichen Zahlen ist, für die es also keine bijektive Abbildung zwischen den beiden Mengen gibt und die somit nicht abzählbar ist, *überabzählbar* genannt.

Einfacher ausgedrückt sind Mengen genau dann abzählbar, wenn ihre Elemente durchnumeriert werden können. Dann existiert eine surjektive Abbildung der natürlichen Zahlen auf die betrachtete Menge, so dass jedes Element der Menge einer natürlichen Zahl zugeordnet ist.

3.1.2 one-to-one-Mapping

Die im vorangegangenen Abschnitt beschriebene Gleichmächtigkeit zweier Mengen kann auch mit dem Schlagwort *one-to-one-mapping* illustriert werden. Seien beispielsweise folgende Mengen gegeben:

$$S_1 = \{\text{gerade Zahlen}\} \text{ und } S_2 = \{\text{alle ganzen Zahlen}\} \tag{3.3}$$

Dann kann S_1 eindeutig auf S_2 abgebildet werden mit der Funktion

$$\begin{aligned} &S_1 \to S_2 \\ &f(2i) \to i \end{aligned} \tag{3.4}$$

Es liegt eine bijektive Abbildung zwischen den beiden Mengen vor, so dass die Mengen von gleicher Mächtigkeit (Kardinalität) sind. Man kann den Begriff one-to-one-mapping verwenden, da jedes Element der einen Menge genau einem Element der anderen Menge zugeordnet wird und umgekehrt.

Ein weiteres Beispiel sei mit folgenden Mengen gegeben:

$$S_1 = \{\text{ganze, positive Zahlen}\} \text{ und } S_2 = \{\text{ganze, negative Zahlen}\} \tag{3.5}$$

Auch hier liegt eine eindeutige Abbildung von S_1 auf S_2 vor:

$$\begin{aligned} &S_1 \to S_2 \\ &f(i) \to -i \end{aligned} \tag{3.6}$$

Folglich sind S_1 und S_2 gleichmächtig, was noch einmal durch eine grafische Diagonalisierung verdeutlicht werden kann.

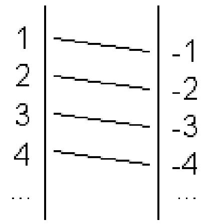


Abbildung 1 - Grafische Zuordnung von Mengenelementen

Zur Motivation ist anzumerken, dass die detaillierte Beschreibung dieses Themas als sinnvoll zu erachten ist, da viele Beweise der kommenden Wochen in ihrer Basis auf diesem Vorgehen basieren und die Begriffe so zu den Essentials der Veranstaltung gehören.

3.1.3 Abzählbarkeit rationaler Zahlen

Ein Beispiel für einen Beweis der Abzählbarkeit einer Menge ist die Abzählbarkeit der rationalen Zahlen. Die rationalen Zahlen ist die Menge der Bruchzahlen, die Vereinigung einer Zählermenge A und einer Nennermenge B mit A und B gleich Menge der ganzen Zahlen $\mathbb{Z}$.

$$p = \frac{a}{b} \text{ mit } a, b \in \mathbb{Z} \tag{3.7}$$

Dies läßt sich leicht mit einer einfachen Darstellung und der aus der Mathematik bekannten Gesetzmäßigkeit beweisen, dass die Vereinigung abzählbar vieler abzählbarer Mengen (wie die der ganzen Zahlen) wieder eine abzählbare Menge ist.

Jedes Element der Zählermenge besitzt eine abzählbare Menge von Elementen im Nenner, die Menge der ganzen Zahlen eben. Ebenso gibt es eine abzählbare Menge von Zählerelementen, so dass man folgendes unendliches Schema anwenden kann, wobei die Pfeile eine lineare Numerierung bedeuten, beginnend mit der linken oberen Ecke. Folglich ist die Menge der rationalen Zahlen abzählbar unendlich.

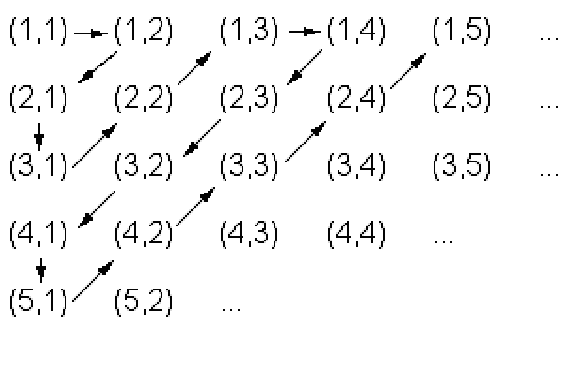


Abbildung 2 - Quadratisch unendliches Schema der Bruchzahlen

Formal läßt sich eine Zeile des Schemas folgendermaßen ausdrücken:

$$P_z := \left\{ \frac{z}{k} \mid k \in \mathbb{Z} \setminus \{0\} \right\} \text{ für } z \in \mathbb{Z} \tag{3.8}$$

Da alle P_z abzählbar sind, ist auch die Vereinigung abzählbar vieler solcher Mengen abzählbar und damit die Menge der rationalen Zahlen abzählbar:

$$\mathbb{Q} = \bigcup_{z \in \mathbb{Z}} P_z \qquad (3.9)$$

3.1.4 Überabzählbarkeit reeller Zahlen

Die Menge der reellen Zahlen ist die Menge der Dezimalzahlen mit endlichen und unendlicher Periode. Man kann die Zahlen im Bereich]0,1[in einem Schema anordnen und dann eine *Cantorsches Diagonalverfahren* genannte Methode anwenden. Ist bereits dieses Intervall nicht abzählbar, so ist die gesamte Menge der reellen Zahlen überabzählbar.

Nehmen wir zunächst an, das Intervall sei abzählbar und es existiere dementsprechend eine surjektive Abbildung $f : N \to \left]0,1\right[$. Sei x_n eine Zahl aus diesem Intervall, die sich aus Stellen $a_1, a_2, a_3, \ldots$ zusammensetzt:

$$\begin{aligned} x_1 &= 0, a_{11} a_{12} a_{13} \ldots \\ x_2 &= 0, a_{21} a_{22} a_{23} \ldots \\ x_3 &= 0, a_{31} a_{32} a_{33} \ldots \\ &\ldots \end{aligned} \qquad (3.10)$$

Sei nun c eine weitere Zahl aus dem Intervall mit

$$c = 0, c_1 c_2 c_3 \ldots \qquad (3.11)$$

wobei sich c_i folgendermaßen zusammensetzt, also Stellen aus x_n neu zusammensetzt (diagonal im Schema wegen a_{ii}):

$$c_i = \begin{cases} 5 \text{ für } a_{ii} \neq 5 \\ 4 \text{ für } a_{ii} = 5 \end{cases} \qquad (3.12)$$

Somit ist gewährleistet, dass c_i niemals gleich a_{ii} ist und mindestens in einer Stelle verschieden. Damit ist c ungleich jedem beliebigen x_n. Wenn das Intervall abzählbar ist, müßte ein x_j mit $x_j = c$ existieren. Das ist jedoch ein Widerspruch und somit ist das Intervall der nicht abzählbar. Daraus folgt, dass die reellen Zahlen ebenfalls nicht abzählbar, sondern überabzählbar sind.

Würde man diesen Beweis auf die Menge der rationalen Zahlen anwenden, könnte man leicht den Fehler machen und übersehen, dass es sich bei x_n um Dezimalzahlen mit unendlicher Periode handelt. So ist bei den rationalen Zahlen das c ebenfalls ungleich jedem beliebigen x_n. Allerdings ist c zugleich auch nicht mehr Element der rationalen Zahlen (sondern der reellen). Die Bedingung der Abgeschlossenheit ist damit nicht mehr erfüllt und somit nichts über die Abzählbarkeit der Menge der rationalen Zahlen ausgesagt.

3.2 Transitive und reflexive Hülle

3.2.1 Definition

Das *Relationenprodukt* zweier Relationen R und S wird folgendermaßen definiert:

$$RS = R \circ S = \{(x, y) \mid \exists z \text{ mit } xRz \wedge zSy\} \qquad (3.13)$$

Dabei wird gewissermaßen die Transitivität über zwei Relationen hinweg ausgedrückt. Insbesondere heißt eine Relation transitiv, wenn das Relationenprodukt mit sich selbst wieder Teilmenge der Relation ist:

$$R \circ R \subseteq R \qquad (3.14)$$

$$R \circ R = \{(x, y) \mid \exists z \text{ mit } xRz \wedge zRy\} \qquad (3.15)$$

Bestimmt man dieses Relationenprodukt unendlich oft und vereinigt anschließend die Relationenprodukte, so erhält man den transitiven Abschluß (auch transitive Hülle genannt):

$$R^{+} = tra(R) = \bigcup_{n\geq 1} R^{n} = R \cup R \circ R \cup R \circ R \circ R \cup R^{4} \cup R^{5} ... \qquad (3.16)$$

Für R^0 definiert man die identische, reflexive Abbildung auf der Menge A:

$$R^{0} = \{(x,x) \mid x \in A\} \qquad (3.17)$$

Die Vereinigung von transitiver Hülle und der identischen Abbildung R0 wird reflexive Hülle genannt:

$$R^{*} = R^{+} \cup R^{0} = \bigcup_{n\geq 0} R^{n} = R^{0} \cup R^{1} \cup R^{2} \cup R^{3} \cup R^{4} ... \qquad (3.18)$$

Ein interessanter Anwendungsfall der transitiven Hülle in Verbindung mit einer Relationenmatrix ist das Wege-Problem, bei dem die kürzeste Entfernung zwischen zwei Punkten über andere Punkte hinweg gesucht (*traveling salesman problem*)

3.2.2 Beispiele

Sei R eine Relation mit

$$R = \{(1,2),(2,2),(2,3)\} \qquad (3.19)$$

auf der Menge A mit

$$A = \{1,2,3\} \qquad (3.20)$$

Dann ergibt sich die transitive Hülle R^+ von R folgendermaßen

$$R^{+} = \{(1,2),(1,3),(2,2),(2,3)\} \qquad (3.21)$$

und die relfexive Hülle R* entsprechend

$$R^{*} = \{(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)\} \qquad (3.22)$$

3.3 Endliche Automaten (FSM, DFA)

3.3.1 Veranschaulichung und Darstellung

Ein *Automat* kann als eine Blackbox, beispielsweise als Modell eines elektrischen Schaltwerks betrachtet werden. Füttert man den Automaten mit einer Eingabe, kann er diese akzeptieren oder nicht akzeptieren. Damit lassen sich Eingaben in die Gruppe der erkannten und die Gruppe der nicht erkannten Wörter einordnen, wobei die Menge der erkannten Wörter die Sprache ist, auf die der Automat hält.

Mit Hilfe eines Automaten läßt sich also eine Sprache definieren, genau wie bei der Definition einer Grammatik, nur dass bei der Grammatik die Wörter der Sprache gewissermaßen als Produkt entstehen, während der Automat die Eingabewörter auf ihre Zugehörigkeit zu der betrachteten Sprache prüft.

Wie beschrieben, erkennt ein Automat eine Sprache $L \subseteq \Sigma^{*}$. Das läuft so ab, dass der Automat ein beliebigen Wort $x_1x_2...x_n$ als Eingabe erhält und Zeichen für Zeichen gemäß Übergangsfunktion abarbeitet. Er wechselt dabei so lange den Zustand, bis das letzte Eingabezeichen erreicht ist. Befindet sich der Automat dann in einem Endzustand, dann gehört das Eingabewort zur Sprache L. Befindet er sich in einem anderen Zustand, gehört das Wort nicht zur Sprache L.

Verdeutlicht wird das Prinzip durch eine bildhafte Darstellung wie folgt. Dabei ließt der Automat über einen Lesekopf die Eingabe vom Eingabeband und wechselt entsprechend der Übergangsfunktion zwischen den Zuständen. Tritt der Automat in einen Endzustand, so wird dies hier durch ein Signal vermerkt.

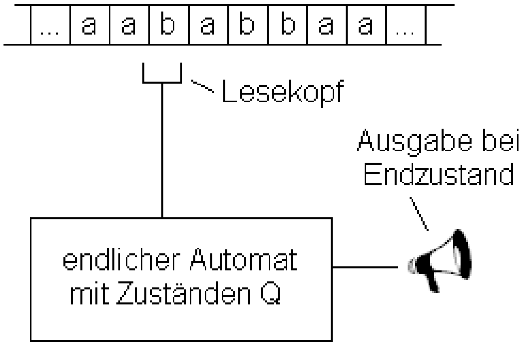


Abbildung 3 - Prinzip eines endlichen Automaten

Die übliche anschauliche Notation eines endlichen Automaten ist der Graph. Eine Kante verbindet Knoten, die den Zuständen entsprechen. Für jedes Zeichen des Eingabealphabets gibt es eine eigene Kante, die vom jeweiligen Zustand zu einem Folgezustand führt.

3.3.2 Definition

Ein *deterministischer endlicher Automat* auch *Finite State Maschine* (FSM) oder *Deterministic Finite Automation* (DFA) genannt, setzt sich aus fünf charakteristischen Eigenschaften zusammen, so dass ein solcher Automat ein 5-Tupel darstellt:

$$M = (Q, \Sigma, \delta, q_0, F) \tag{3.23}$$

Dabei bezeichnet Q die Menge der *Zustände*, Σ das *Eingabealphabet*, δ die *Übergangsfunktion*, q_0 den *Startzustand* und F die Menge der *Endzustände*. Ausführlicher haben die 5 Elemente folgende Funktionen:

- Q ist eine endliche Menge von Zuständen (daher der Name!)
- Σ ist eine endliche Menge von Eingabesymbolen, das Eingabealphabet
- $\delta : Q \times \Sigma \rightarrow Q$ ist eine Übergangsfunktion (kartesisches Produkt von Zustand und Eingabesymbol), die einem Zustand und Element des Eingabealphabets einen Folgezustand zuordnet
- q_0 ist der Startzustand des Automaten mit $q_0 \in Q$
- F ist eine Menge von Endzuständen (akzeptierende Zustände) mit $F \subseteq Q$

Charakteristisch für diesen Automatentyp (es gibt auch weitere wie wir später sehen werden) ist, dass jeder Kombination aus Zustand und Eingabesymbol ein eindeutiger Folgezustand zugeordnet wird. Verschiedene Kombinationen können jedoch auf den gleichen Folgezustand zeigen.

3.3.3 Erweiterung von $\delta : Q \times \Sigma \rightarrow Q$ auf $\delta^* : Q \times \Sigma^* \rightarrow Q$

Um Wörter auf ihre Sprachzugehörigkeit zu prüfen, ist es sinnvoll, als Eingabe für eine Übergangsfunktion nicht nur einzelne Zeichen sondern ganze Wörter zuzulassen. Diese neue Übergangsfunktion wird dann δ^* genannt. Da sie auf dem freien Monoid Σ arbeitet, kann sie auch das leere Wort verarbeiten. Prinzipiell handelt es sich bei δ^* um eine rekursive Mehrfachausführung des einfachen δ.

Es gelten folgende Axiome:

$$\delta^*(q, e) = q \tag{3.24}$$

$$\delta^*(q, a) = \delta(q, a) \quad \forall q \in Q, \forall a \in \Sigma, \tag{3.25}$$

$$\delta^*(q, a_1, a_2, ..., a_n) = \delta(\delta^*(q, a_1, a_2, ..., a_{n-1}), a_n) \tag{3.26}$$

Zur Vereinfachung der Notation legt man fest, dass

$$\delta^{*} \square \delta \tag{3.27}$$

Daraus folgt, dass die *akzeptierte Sprache* des Automaten $M = (Q, \Sigma, \delta, q_0, F)$ definiert ist als

$$T(M) := \left\{ x \mid x \in \Sigma^{*} \ \wedge \ \delta(q_0, x) \in F \right\} \tag{3.28}$$

3.3.4 Beispiel eines endlichen Automaten

Ein Beispiel eines endlichen, deterministischen Automaten sei ein Automat, der eine Sprache erkennt, die aus einer beliebig langen Folge von ab besteht:

$$T(M) := \left\{ (ab)^{n} \mid n \geq 1 \right\} \tag{3.29}$$

Mit Hilfe eines Induktionsbeweises, der ausführlich am nächsten Vorlesungstag behandelt wird, kann die Korrektheit des folgenden Graphen bewiesen werden:

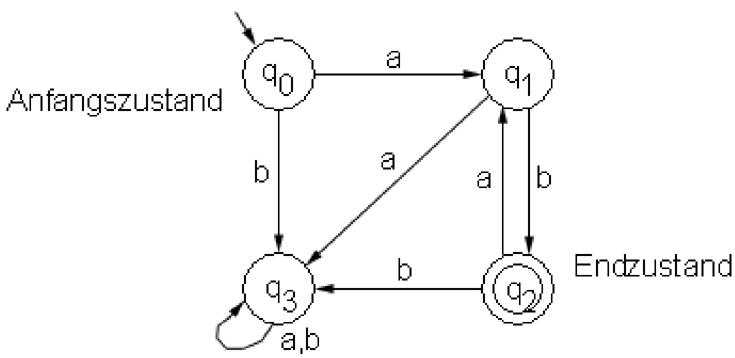


Abbildung 4 - Beispiel eines endlichen Automaten

4 Vorlesung vom 18. April 2000

4.1 Beweis eines DFA (deterministischer endlicher Automat)

Bereits im letzten Protokoll wurde unser Beispielautomat vorgestellt. Er erkennt die folgende Sprache:

$$T(M) := \left\{ (ab)^n \mid n \geq 1 \right\} \subseteq \Sigma^* \tag{4.1}$$

Der Automat selbst ist in der folgenden Abbildung graphisch dargestellt:

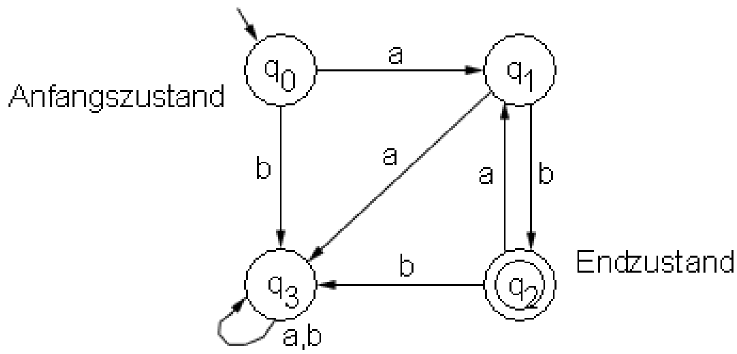


Abbildung 5 - Beispiel eines endlichen Automaten

4.2 Vorgehensweise

Was haben wir nun zu tun, wenn wir beweisen wollen, dass der gezeigte endliche Automat wirklich die Sprache T(M) akzeptiert? Wir müssen zeigen, dass der Automat wirklich nur für die Zeichenfolgen im Endzustand $q_2 \in F = \{q2\}$ landet, die in der Sprachspezifikation (4.1) angegeben wurden.

Die *allgemeine Vorgehensweise* lautet für einen solchen Beweis dann wie folgt:

1. Aufstellen von Induktionsbehauptungen für jeden Zustand des endlichen Automaten, also für jedes $q \in Q$. Die Induktionsbehauptungen haben die allgemeine Form

$$\delta(q_0, x_1, \ldots, x_n) = q_i \Leftrightarrow \{\text{Bedingung für Erreichen des Zustandes}\} \tag{4.2}$$

2. Beweis der Behauptungen per Induktion
3. Beweis der gewünschten Spracheigenschaft durch folgende Äquivalenz. Mit „gewünschter Sprachspezifikation“ ist eine Beschreibung von $x_1 \ldots x_n$ in der Form gemeint, dass $x_1 \ldots x_n$ die gewünschte Sprache darstellen.

$$x_1 \ldots x_n \in T(M) \Leftrightarrow \{\text{gewünschte Sprachspezifikation}\} \tag{4.3}$$

Bei oberflächlicher Betrachtungsweise fragt man sich leicht, wieso es nicht reicht, die Induktionsbehauptungen für die Endzustände zu beweisen. Dies liegt daran, dass die Endzustände i. d. R. nur über andere Zustände erreicht werden können, deren Eigenschaften also auch bewiesen werden müssen.

4.2.1 Beweis des Beispiels

Zu zeigen:

Wir haben zu zeigen: $T(M) = \left\{ (ab)^n \middle| n \geq 1 \right\}$

Wir stellen nun zunächst die Induktionsbehauptungen gemäß (4.2) auf. Hierzu betrachten wir im Transitionsdiagramm (Abbildung 5) jeden einzelnen Zustand und überlegen uns, wann (für welche

Eingabefolgen) dieser Zustand erreicht werden kann. Diese Überlegungen führen uns zu den folgenden Behauptungen:

4.2.2 Induktionsbehauptungen:

$$\delta(q_0, x_1 \ldots x_n) = q_1 \Leftrightarrow \text{n ungerade } \wedge x_1 \ldots x_{n-1} = (ab)^{\frac{n-1}{2}} \wedge x_n = a \tag{4.4}$$

$$\delta(q_0, x_1 \ldots x_n) = q_2 \Leftrightarrow \text{n gerade } \wedge x_1 \ldots x_n = (ab)^{\frac{n}{2}} \tag{4.5}$$

$$\delta(q_0, x_1 \ldots x_n) = q_3 \Leftrightarrow x_1 \ldots x_n \in \text{"sonst"} \tag{4.6}$$

$$\delta(q_0, x_1 \ldots x_n) \neq q_0 \tag{4.7}$$

Als Erläuterung der Induktionsbehauptungen betrachten wir (4.4): Nach dem Transitionsdiagramm kann q_1 nur dann erreicht werden, wenn – ausgehend vom Startzustand q_0 - eine „ab“-Folge mit abschließendem „a“ als Eingabezeichenkette vorliegt (z. B. „ababababa“). Die Anzahl der Zeichen ist hierbei stets ungerade.

4.2.3 Induktionsverankerung:

Wir beweisen durch Induktion über die Länge der Eingabefolgen und wählen n=1 (also eine Zeichenkette der Länge 1). Wir zeigen für die einzelnen Induktionsbehauptungen deren Richtigkeit für n=1:

zu (4.4) $\quad \delta(q_0, x_1) = q_1 \Leftrightarrow x_1 = a \Leftrightarrow x_1 = (ab)^{\frac{0}{2}} a$

zu (4.5) $\quad \delta(q_0, x_1) = q_2$ tritt nicht auf (siehe Erläuterung)

zu (4.6) $\quad \delta(q_0, x_1) = q_3 \Leftrightarrow x_1 = b \Leftrightarrow x_1 = \text{"sonst"}$

zu (4.7) $\quad \delta(q_0, x_1) \neq q_0$ (siehe Erläuterung)

Hierzu folgende Erläuterungen: Die Äquivalenzen sind natürlich dadurch zu zeigen, dass man die Implikationen sowohl von links nach rechts ($\Rightarrow$) als auch von rechts nach links ($\Leftarrow$) begründen muss.

Zum Falle von (4.5) ist zu sagen, dass zunächst („$\Rightarrow$“) der Fall, mit n=1 (nur einem Zeichen) q_2 nicht zu erreichen ist. Damit ist die Prämisse falsch, die Implikation also wahr. Die andere Richtung („$\Leftarrow$“): n ist ungerade, damit ist wiederum die Prämisse falsch und die Implikation wahr.

Zu (4.7): Nach dem Transitionsdiagramm wird q_0 bereits mit einem Eingabezeichen verlassen und kann nie wieder erreicht werden.

4.2.4 Induktionsschritt ($n \to n+1$):

Wir nehmen hier eine Fallunterscheidung vor. Solche Fallunterscheidungen sind in Induktionsbeweisen von endlichen Automaten häufig vorzunehmen. In unserem Falle bietet sich eine Fallunterscheidung nach dem Kriterium „n gerade?“ an, da dieses Kriterium in unseren Induktionsbehauptungen eine Rolle spielt.

Fall 1: n+1 gerade, n ungerade

Zu (4.4)**:**

$$\delta(q_0, x_1 \ldots x_n x_{n+1}) = q_1 \overset{\text{Def. von } \delta}{=} \delta\left(\underbrace{\delta(q_0, x_1 \ldots x_n)}_{=q_0 \vee q_2}, x_{n+1}\right) \tag{4.8}$$

Wie ist das nun zu lesen? Um nach q_1 zu kommen, muss man nach dem Transitionsdiagramm im vorletzten Schritt in q_0 oder q_2 gewesen sein.

Weil nun $n+1 \geq 2 \Rightarrow n \geq 1$ gilt, kann kann Fall q_0 schon mal nicht vorkommen. q_2 kann nicht vorkommen, weil n ungerade ist. Damit gilt die Prämisse von (4.4) nicht und die Implikation von links nach rechts ist wahr. Liest man (4.4) von rechts nach links, dann gelten q_0 / q_2 ebenfalls nicht und damit ist die Äquivalenz (4.4) insgesamt gezeigt.

Zu (4.5)**:**

$$\delta(q_0, x_1 \ldots x_n x_{n+1}) = q_2 \overset{\text{Def. von } \delta}{=} \delta\left(\underbrace{\delta(q_0, x_1 \ldots x_n)}_{=q_1}, x_{n+1}\right) \tag{4.9}$$

Wieder haben wir das Transitionsdiagramm betrachtet und sind diesmal nur auf q_1 als letzten möglichen Schritt gestoßen.

Wir betrachten nun zunächst den Weg von links nach rechts in (4.5)**:**

$$\begin{aligned}
&\delta(q_0, x_1 \ldots x_n x_{n+1}) = q_2 \overset{\text{Def. von } \delta}{=} \delta\left(\underbrace{\delta(q_0, x_1 \ldots x_n)}_{=q_1}, x_{n+1}\right) \\
&\overset{IV}{\Rightarrow} x_1 \ldots x_n = (ab)^{\frac{n-1}{2}} a \wedge x_{n+1} = b \\
&\Rightarrow x_1 \ldots x_{n+1} = (ab)^{\frac{n+1}{2}}
\end{aligned}$$

Natürlich muss auch der Rückweg gezeigt werden. Dies funktioniert aber analog:

$$\begin{aligned}
&\text{n ungerade } \wedge \mathrm{x}_1 \ldots x_{n+1} = (ab)^{\frac{\mathrm{n}+1}{2}} \\
&\Rightarrow n+1 \text{ gerade } \wedge \mathrm{x}_1 \ldots x_{n+1} = (ab)^{\frac{\mathrm{n}+1}{2}} \\
&\Rightarrow n+1 \text{ gerade } \wedge \mathrm{x}_1 \ldots x_{n+1} = (ab)^{\frac{\mathrm{n}-1}{2}} ab \\
&\overset{IV}{\Rightarrow} \delta(q_0, x_1 \ldots x_n) = q_1 \wedge x_{n+1} = b \\
&\overset{Def.\delta, IV}{\Rightarrow} \delta(q_0, x_1 \ldots x_{n+1}) = q2
\end{aligned}$$

Damit haben wir die Äquivalenz gezeigt.

Zu (4.6)**:**

Hier ergibt sich:

$$\delta(q_0, x_1 \ldots x_n x_{n+1}) = q_2 \overset{\text{Def. von } \delta}{=} \delta\left(\underbrace{\delta(q_0, x_1 \ldots x_n)}_{=q_0 \vee q_1 \vee q_2 \vee q_3}, x_{n+1}\right) \tag{4.10}$$

Wie zuvor scheidet q_0 aus, da $n \geq 1$ und q_2 scheidet aus, weil n ungerade ist.

„$\Rightarrow$“ Wir wählen zunächst den Schritt von links nach rechts:

I.1 (q1)

$$\delta(q_0, x_1 \ldots x_n) = q_1$$
$$\overset{IV}{\Rightarrow} x_1 \ldots x_n = (ab)^{\frac{n-1}{2}} a \wedge x_{n+1} = a$$
$$\Rightarrow x_1 \ldots x_n x_{n+1} \in \text{"sonst"}$$

I.2 (q3)

$$\delta(q_0, x_1 \ldots x_n) = q_3$$
$$\overset{IV}{\Rightarrow} x_1 \ldots x_n \in \text{"sonst"} \wedge (x_{n+1} = a \vee x_{n+1} = b)$$
$$\Rightarrow x_1 \ldots x_n x_{n+1} \in \text{"sonst"}$$

„$\Leftarrow$“ Jetzt beweisen wir den Rückweg und beachten zuvor die folgende Regel:

$$x_1 \ldots x_{n+1} \in \text{"sonst"}$$
$$\Rightarrow \mathrm{x}_1 \ldots x_n \in \neg\text{sonst} \;\vee\; \mathrm{x}_1 \ldots x_n \in \text{sonst}$$

II. 1 (not sonst)

$$x_1 \ldots x_n \in \neg\text{sonst}$$
$$\overset{\substack{\text{n ungerade}\\ \text{IV}}}{\Rightarrow} \mathrm{x}_1 \ldots x_n = (ab)^{\frac{n-1}{2}}$$
$$\overset{\substack{x_1 \ldots x_{n+1}\\ =\text{sonst}}}{\Rightarrow} \delta(q_0, x_1 \ldots x_n) = q_1 \wedge x_{n+1} = a$$
$$\Rightarrow \delta(q_0, x_1 \ldots x_{n+1}) = \delta\left(\underbrace{\delta(q_0, x_1 \ldots x_n)}_{q_1}, a\right) = q_3$$

II.2 (sonst)

$$x_1 \ldots x_n \in \text{"sonst"}$$
$$\overset{IV}{\Rightarrow} \delta(q_0, x_1 \ldots x_n) = q3$$
$$\overset{\substack{Def\,\delta\\ Transitionsdiagr.}}{\Rightarrow} \delta(q_0, x_1 \ldots x_{n+1}) = \delta(\delta(q_0, x_1 \ldots x_n), x_{n+1}) = q3$$

zu (4.7)**:**

$\delta(q_0, x_1 \ldots x_{n+1}) = q_0$ ist ein Widerspruch, da $n+1 \geq 2$. Damit kann dieser Fall nicht vorkommen.

Fall 2: n+1 ungerade, n gerade

Dieser Fall ist ganz analog wie Fall 1 zu behandeln:

Zu (4.4)**:**

$$\delta(q_0, x_1 \ldots x_n x_{n+1}) = q_1 \overset{Def\,\delta}{=} \delta\left(\underbrace{\delta(q_0, x_1 \ldots x_n)}_{q_0 \vee q_2}, x_{n+1}\right)$$

q_0 scheidet nach der Induktionsannahme aus, weil $n \geq 1$ gilt.

$$\overset{Def\,\delta}{\Leftrightarrow} \delta(q_0, x_1 \ldots x_n) = q_2 \wedge x_{n+1} = a$$
$$\overset{IV}{\Leftrightarrow} x_1 \ldots x_n = (ab)^{\frac{n}{2}} \wedge x_{n+1} = a$$
$$\Leftrightarrow x_1 \ldots x_n x_{n+1} = (ab)^{\frac{n}{2}} a$$

Zu (4.5)**:**

Es gilt:

$$\delta\left(q_0, x_1 \ldots x_n x_{n+1}\right) = q_2 = \delta\left(\underbrace{\delta\left(q_0, x_1 \ldots x_n\right)}_{q_1}, x_{n+1}\right)$$

$$\Rightarrow \delta\left(q_0, x_1 \ldots x_{n+1}\right) = q_1 \wedge n \text{ gerade}$$

Auch dieser Fall scheidet aus, weil n ungerade ist.

Zu (4.6)**:**

Dieser Fall ist ähnlich wie beim Fall n ungerade zuvor

zu (4.7)**:**

Ist genauso wie oben.

4.2.5 Beweis der Sprachäquivalenz

Wir müssen nun noch mit Hilfe der aus dem Induktionsbeweis gewonnenen Erkenntnis, wann die Endzustände erreicht werden, zeigen, dass der endliche Automat aus Abbildung 5 tatsächlich T(M) realisiert:

$$\begin{aligned} x_1 \ldots x_n \in T(M) \quad & \Leftrightarrow \delta\left(q_0, x_1 \ldots x_n\right) \in F \\ & \Leftrightarrow \delta\left(q_0, x_1 \ldots x_n\right) = F \\ & \overset{Ind.-Beweis}{\Leftrightarrow} \quad x_1 \ldots x_n = (ab)^{\frac{n}{2}} \wedge \text{n gerade} \end{aligned} \tag{4.11}$$

$$\Rightarrow T(M) = \left\{(ab)^n \middle| n \geq 1\right\}$$

□

4.3 Nichtdeterministische Endliche Automaten (NFA)

Neben den deterministischen endlichen Automaten (DFA) existieren auch nichtdeterministische endliche Automaten (NFA), die sich dadurch auszeichnen, dass bei einem Zustand bei demselben Eingabesymbol keine, eine oder mehr Transitionen erlaubt sind.

Die nichtdeterministischen Automaten sind insbesondere ein nützliches Konzept zum Beweis von Sätzen. Weiter wird sich zeigen, dass die NFAs äquivalent zu den DFAs sind. Insbesondere kann man auch DFAs als einen Spezialfall der NFAs auffassen, nämlich als einen, bei dem es je Zustand eine einzige Transitions für jedes Symbol gibt.

Um zu bestimmen, ob eine bestimmte Zeichenkette w von einem DEA akzeptiert wird, genügt es, diesen einen Pfad zu prüfen. Bei einem NFA kann es viele Pfade geben, die mit w markiert sind. Hier müssen also alle Pfade überprüft werden.

4.3.1 Formale Definition

Definition (NFA):

Ein nichtdeterministischer endlicher Automat ist ein Quintupel $M = \left(Q, \Sigma, \delta, q_0, F\right)$ mit:

1. Q ist eine endliche Menge von Zuständen
2. ist eine endliche Menge von Eingabesymbolen
3. $q_0 \in Q$ ist der Anfangszustand
4. $F \subseteq Q$ ist die Menge der Endzustände (= akzeptierten Zustände)
5. $\delta : Q \times \Sigma \rightarrow P(Q) = 2^Q$ (Potenzmenge) ist die Übergangsfunktion

Wichtig hierbei ist, dass die Ergebnismenge der Übergangsfunktion
$\delta'([p_1, p_2, \dots, p_j], a) = [r_1, r_2, \dots, r_k]$ nun also die Potenzmenge von Q ist. liefert jetzt also immer eine Menge von Zuständen. Dabei bedeutet $\delta(q,a)$ die Menge aller Zustände p, für die es einen mit a markierten Übergang von q nach p gibt. $\delta(q,a) = \emptyset \in P(Q)$ bedeutet, dass der Übergang für (q,a) undefiniert ist.

4.3.2 Erweiterung von des NFA

Die Funktion kann folgendermassen auf eine Funktion * ausgedehnt werden.

Definition (*):

(Erweiterung von $\delta : Q \times \Sigma \to P(Q)$ auf $\delta^* : Q \times \Sigma^* \to P(Q)$):

1. $\delta^*(q,e) := \{q\} \in 2^Q$
2. $\delta^*(q,x) := \delta(q,x) \qquad \forall x \in \Sigma$
3. $\delta(q, x_1 \dots x_n x_{n+1}) := \bigcup\limits_{p \in \delta^*(q, x_1 \dots x_n)} \delta(p, x_{n+1})$
4. Konvention: δ^* □ δ

4.3.3 Die vom NFA erkannte Sprache

Definition (vom NFA erkannte Sprache):

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein nichtdeterministischer endlicher Automat (NFA). Dann heißt
$T(M) := \left\{ x \middle| x \in \Sigma^* \wedge \left(\delta(q_0, x) \cap F \neq \emptyset \right) \right\}$ die von M erkannte Sprache.

Anmerkung: Diese Definition impliziert, dass es *wenigstens einen* Weg von Startzustand aus geben muss, mit dem man über das Wort x in einem Endzustand landet.

4.3.4 Beispiel eines NFA

Wir betrachten den NFA, der die Sprache $L = \left\{ (ab)^n \middle| n \geq 1 \right\} \cup \left\{ (abb)^n \middle| n \geq 1 \right\}$ akzeptiert:

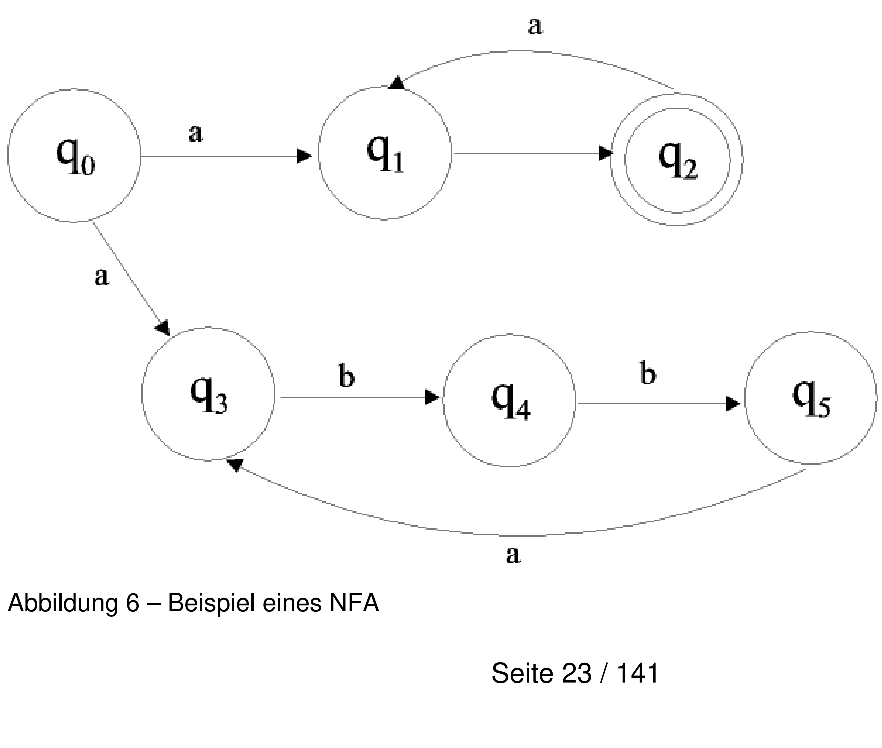


Abbildung 6 – Beispiel eines NFA

Aus diesem Transitionsdiagramm kann man nun die folgenden Transitionen erkennen:

$$\begin{aligned}
&\delta(q_0,e)=\{q_0\}\\
&\delta(q_0,a)=\{q_1,q_3\}\\
&\delta(q_0,ab)=\{q_2,q_4\}\\
&\delta(q_0,aba)=\{q_1\}\\
&\delta(q_0,abb)=\{q_5\}
\end{aligned} \tag{4.12}$$

Um diese zu erhalten, geht man wie folgt vor (wir betrachten das Beispiel $\delta(q_0,ab)\to\delta(q_0,aba)$):

Betrachte die beiden Zustände q_2 und q_4 aus $\delta(q_0,ab)$. Wohin gelangt man jeweils bei einer weiteren Eingabe von „a“? Von q_2 aus gelangt man nach q_1 (also bilde die Menge $\{q_1\}$), von q_4 aus gelangt man mit „a“ nirgendwohin (bilde die leere Menge $\{\}$). Dann vereinige die gefundenen Ergebnismengen. Es ergibt sich die Menge $\{q_1\}$. Also gilt: $\delta(q_0,aba)=\{q_1\}$.

Wir beweisen nun bei diesem NFA, dass er die Sprache $L=\{(ab)^n \mid n\ge 1\}\cup\{(abb)^n \mid n\ge 1\}$ abbildet.

Wir verwenden hierbei Induktion.

Induktionsverankerung (n <= 2):

$$\begin{aligned}
&\forall n\le 2:\delta(q_0,x_1\dots x_n)\cap F\ne\varnothing\\
&\Leftrightarrow q_2\in\delta(q_0,x_1\dots x_n)\\
&\Leftrightarrow x_1\dots x_n=ab
\end{aligned}$$

Induktionsschritt:

Man beweise durch Induktion $\forall n\ge 3$:

$$\begin{aligned}
&q_0\notin\delta(q_0,x_1\dots x_n)\\
&q_1\in\delta(q_0,x_1\dots x_n)\Leftrightarrow \text{n ungerade } \wedge x_1\dots x_n=(ab)^{\frac{n-1}{2}}a\\
&q_2\in\delta(q_0,x_1\dots x_n)\Leftrightarrow \text{n gerade } \wedge x_1\dots x_n=(ab)^{\frac{n}{2}}\\
&q_3\in\delta(q_0,x_1\dots x_n)\Leftrightarrow n=3m+1\ \wedge x_1\dots x_n=(abb)^{\frac{n-1}{3}}a\\
&q_4\in\delta(q_0,x_1\dots x_n)\Leftrightarrow n=3m+2\ \wedge x_1\dots x_n=(abb)^{\frac{n-2}{3}}ab\\
&q_5\in\delta(q_0,x_1\dots x_n)\Leftrightarrow n=3m\ \wedge x_1\dots x_n=(abb)^{\frac{n}{3}}
\end{aligned}$$

Diese Behauptung müsste nun per Induktion formal bewiesen werden.

Gesamtbeweis

Zusammenfassend mit dem gelungenen Induktionsbeweis folgt also:

$$\begin{aligned}
\forall n\ge 3:\quad & x_1\dots x_n\in T(M)\\
\overset{Def.T}{\Leftrightarrow}\quad & \delta(q_0,x_1\dots x_n)\cap F\ne\varnothing\\
\overset{Def.F}{\Leftrightarrow}\quad & q_2\in\delta(q_0,x_1\dots x_n)\vee q_5\in\delta(q_0,x_1\dots x_n)\\
\overset{Induktionsbew.}{\Leftrightarrow}\quad & x_1\dots x_n=(ab)^{\frac{n}{2}}\vee x_1\dots x_n=(abb)^{\frac{n}{3}}\\
\Rightarrow\quad & T(M)=\{(ab)^n \mid n\ge 1\}\cup\{(abb)^n \mid n\ge 1\}
\end{aligned} \tag{4.13}$$

5 Vorlesung vom 20. April 2000

5.1 Über die Äquivalenz von DFA und NFA [1]

Da jeder DFA ein NFA ist, ist es klar[2], dass die Klasse der Sprachen, welche ein NFA akzeptiert die Sprachen enthält, die von einem DFA akzeptiert werden. Es stellt sich jedoch heraus, dass auch nur diese Sprachen von NFA akzeptiert werden. Der Beweis dafür baut darauf auf, dass gezeigt werden kann, dass ein DFA einen NFA simulieren kann. (Für jeden NFA kann ein equivalenter DFA konstruiert werden.) Der DFA simuliert einen NFA indem die einzelnen Zustände des DFA den Mengen von Zuständen des NFA entsprechen. Der so konstruierte DFA überwacht all die Zustände, die der NFA durch die selbe Eingabefolge erreichen (wie der DFA) könnte. Somit gelangt man zu dem folgendem Theorem:

5.1.1 Theorem

Wenn L eine akzeptierte Sprache für einen nichdeterministischen endlichen Automaten ist, dann existiert ein deterministischer endlicher Automat, der auch L akzeptiert. In anderen Worten: Sei $L \subseteq \Sigma^*$ eine Sprache, dann sind folgende Aussagen gleichwertig:

1. $L = T(M_1)$ für einen deterministischen endlichen Automaten M_1
2. $L = T(M_2)$ für einen nichtdeterministischen endlichen Automaten M_2

5.1.2 Beweis des Buches

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein NFA, welcher die Sprache L akzeptiert. Ein DFA $M' = (Q', \Sigma, \delta', q'_0, F')$ sei dann wie folgt definiert: Die Zustände von M' sind alle Teilmengen der Menge der Zustände von M. Das heißt: $Q' = 2^Q$. M' wird alle Zustände behandeln, welche auch M behandeln kann. F' ist die Menge, der Zustände von Q', welche einen Endzustand von M enthalten. Ein Element von Q' wird dargestellt, als $[q_1, q_2, \ldots, q_i]$, wobei $q_1, q_2, \ldots, q_i$ aus Q stammen. Zu beachten ist, dass $[q_1, q_2, \ldots, q_i]$ ein einziger Zustand des DFA darstellt. Weiterhin ist $q_0 = [q_0]$.

Wir definieren nun:

$$\delta' = ([q_1, q_2, \ldots, q_i], a) = [p_1, p_2, \ldots, p_j] \tag{5.1}$$

dann und nur dann, wenn:

$$\delta = (\{q_1, q_2, \ldots, q_i\}, a) = \{p_1, p_2, \ldots, p_j\} \tag{5.2}$$

In anderen Worten, δ' angewendet auf $[q_1, q_2, \ldots, q_i]$ von Q' wird berechnet, durch anwenden von δ auf jeden Zustand von Q, welcher durch $[q_1, q_2, \ldots, q_i]$ repräsentiert wird. Durch das Anwenden von δ auf alle $q_1, q_2, \ldots, q_i$ und vereinigen der Ergebnisse, erhalten wir eine neue Menge von Zuständen, $p_1, p_2, \ldots, p_j$. Diese neue Menge hat einen Vertreter $[p_1, p_2, \ldots, p_j]$ in Q' und dieser ist der Wert von $\delta' = ([q_1, q_2, \ldots, q_i], a)$.

Es lässt sich nun durch Induktion über die Länge eines Eingabestrings x zeigen, dass

$$\delta'(q'_0, x) = [q_1, q_2, \ldots, q_i] \tag{5.3}$$

[1] Vgl.: INTRODUCTION TO AUTOMATA THEORY, LANGUAGES AND COMPUTATION by John E. HOPECROFT and Jeffrey D. ULLMAN pages 22f

[2] Ein DFA ist ein Spezialfall des NFA, bei dem es für jeden Zustand eine einzigartige Übergangsfunktion für jedes Symbol gibt.

dann und nur dann gilt, wenn:

$$\delta\left(q_0, x\right) = \{q_1, q_2, \ldots, q_i\} \tag{5.4}$$

Induktionsverankerung:

Man sieht leicht, dass für $|x| = 0$ die Behauptung gilt, da $q'_0 = [q_0]$ und x gleich ε sein muss.

Induktionsschritt:

Angenommen, die Behauptung ist wahr für Eingaben der Länge $\leq m$. Weiterhin sei xa eine Kette der Länge $m+1$ mit a in Σ. Dann gilt:

$$\delta' = \left(q'_0, xa\right) = \delta'\left(\delta'\left(q'_0, x\right), a\right) \tag{5.5}$$

Aufgrund der Behauptung ist:

$$\delta'\left(q'_0, x\right) = \left[p_1, p_2, \ldots, p_j\right] \tag{5.6}$$

dann und nur dann wahr, wenn:

$$\delta\left(q0, x\right) = \left\{p_1, p_2, \ldots, p_j\right\} \tag{5.7}$$

gilt. Doch mit der Definition von δ' gilt:

$$\delta'\left(\left[p_1, p_2, \ldots, p_j\right], a\right) = \left[r_1, r_2, \ldots, r_k\right] \tag{5.8}$$

dann und nur dann, wenn:

$$\delta\left(\left\{p_1, p_2, \ldots, p_j\right\}, a\right) = \{r_1, r_2, \ldots, r_k\} \tag{5.9}$$

gilt. Somit ist:

$$\delta'\left(q'_0, xa\right) = \left[r_1, r_2, \ldots, r_k\right] \tag{5.10}$$

dann und nur dann, wenn:

$$\delta\left(q_0, xa\right) = \{r_1, r_2, \ldots, r_k\} \tag{5.11}$$

und das unterstützt die Behauptung. Um den Beweis zu vollenden, muss nur noch hinzugefügt werden, dass $\delta'\left(q'_0, x\right)$ genau dann zu F' führt, wenn $\delta\left(q_0, x\right)$ einen Zustand von Q enthält, der in F liegt. Somit ist $L(M) = L(M')$

Soweit der Beweis des Buches, in der Vorlesung haben wir den Beweis nur geringfügig anders gehört.

5.1.3 Beweis der Vorlesung

Zur Wiederholung hier noch einmal den zu beweisenden Satz: Sei $L \subseteq \Sigma^*$ eine Sprache, dann sind folgende Aussagen gleichwertig:

1. $L = T(M_1)$ für einen deterministischen endlichen Automaten M_1
2. $L = T(M_2)$ für einen nichtdeterministischen endlichen Automaten M_2

„Hinrichtung“:

Zuerst zeigen wir hier, mit einfachen Definitionen, aus 1 folgt 2. Gegeben sei ein DFA $M_1 = \left(Q_1, \Sigma, \delta_1, q_0^{(1)}, F_1\right)$ mit $\delta_1 : Q_1 \times \Sigma \to Q_1$. Dieser Automat akzeptiert die Sprache $L = T(M_1)$ nach Definition. Nun definieren wir uns dazu einen NFA $M_2 =_{Df} \left(Q_1, \Sigma, \delta_2, q_0^{(1)}, F_1\right)$ mit einer Übergangsfunktion $\delta_2 : Q_1 \times \Sigma \to P(Q_1)$.

Diese Übergangsfunktion ist wie folgt definiert:

$$\delta_2(q,\sigma) =_{Df} \{\delta_1(q,\sigma)\} \tag{5.12}$$

Aus dieser Definition folgt nun schon, dass M_2 ein nichtdeterministischer Automat ist. Der einzige Unterschied zum DFA besteht darin, dass die Übergangsfunktion so definiert ist, dass sie eine Menge von Folgezuständen liefert. In unserem Fall sind dies jedoch alles einelementige Mengen.

Durch Induktion über alle $n \geq 0$ kann man dies zeigen:

$$\begin{array}{llll}
\multicolumn{4}{l}{\delta_2(q_0, x_1 \cdots x_n) = \{p\} \Leftrightarrow \delta_1(q_0, x_1 \cdots x_n) = p} \\
\Rightarrow & & & \\
x \in T(M_2) & \Leftrightarrow & & \delta_2(q_0, x) \cap F_1 \neq 0 \\
 & \Leftrightarrow & \exists p: & \delta_2(q_0, x) = \{p\} \subseteq F_1 \\
 & \Leftrightarrow & \exists p: & \delta_1(q_0, x) = p \in F_1 \\
 & \Leftrightarrow & x \in T(M_1) & \\
\multicolumn{4}{l}{\Rightarrow T(M_2) = T(M_1) = L}
\end{array} \tag{5.13}$$

In diesem Beweis ist ausgenutzt worden, dass ein Wort nur dann von einem NFA erkannt wird, wenn die Menge von $\{p\}$ eine Teilmenge von F_1 ist. Dies bedeutet für einen gleichwertigen DFA, dass dieses $p \in F_1$ sein muss, um akzeptiert zu werden. Dies war bei uns nach Definition der Fall.

Rückrichtung:

Nun bleibt noch zu zeigen, dass der umgekehrte Fall gilt. Sprich, wir haben einen NFA gegeben und wollen daraus einen DFA konstruieren. (Aus 2 folgt 1): Hier sei ein NFA $M_2 = \left(Q_2, \Sigma, \delta_2, q_0^{(2)}, F_2\right)$ mit einer Übergangsfunktion $\delta_2 : Q_2 x \times \Sigma \to P(Q_2)$ gegeben. Auch dieser Automat soll nach Definition die Sprache $L = T(M_2)$ akzeptieren.

Wir definieren nun einen DFA M_1 wie folgt:

$$\begin{aligned}
&M_1 =_{Df} \left(P(Q_2), \Sigma, \delta_1, \left\{q_0^{(2)}\right\}, F_1\right) \\
&F_1 =_{Df} \left\{A \middle| A \subseteq Q_2 \wedge (A \cap F_2 \neq \emptyset)\right\} \Rightarrow F_1 \subseteq P(Q_2) \\
&\delta_1 : P(Q_2) \times \Sigma \to P(Q_2) \\
&\delta_1(A,\sigma) = \bigcup_{p \in A} \delta_2(p,\sigma), \sigma \in \Sigma, A \in P(Q_2)
\end{aligned} \tag{5.14}$$

Der so definierte Automat ist ein deterministischer endlicher Automat. Nun bleibt zu zeigen, dass jeder Zustand, den der DFA erreicht, der Menge von Zuständen entspricht, die der NFA erreicht hätte:

$$\delta_1\left(\left\{q_0^{(2)}\right\}, x\right) = \delta_2\left(q_0^{(2)}, x\right) \quad \forall x \in \Sigma^* \tag{5.15}$$

Dies wird wieder durch Induktion über die Länge der Eingaben bewiesen.

Induktionsbehauptung:

$$\delta_1\left(\left\{q_0^{(2)}\right\}, x_1 \cdots x_n\right) = \delta_2\left(q_0^{(2)}, x_1 \cdots x_n\right) \tag{5.16}$$

Induktionsverankerung (n=0):

$\delta_1\left(\left\{q_0^{(2)}\right\}, \varepsilon\right) = \delta_2(q_0^{(2)}, \varepsilon)$ Man sieht leicht, dass für leere Eingaben der Definition der δ -Funktion nach das selbe Ergebnis (das Bleiben im Startzustand $q_0^{(2)}$) erreicht wird.

Induktionsschritt ($n \to n+1$):

$$\begin{aligned}
\delta_1\left(\left\{q_0^{(2)}\right\}, x_1 \cdots x_n x_{n+1}\right) &= \delta_1\left(\delta_1\left(\left\{q_0^{(2)}\right\}, x_1 \cdots x_n\right), x_{n+1}\right) \\
&= \delta_1\left(\delta_2\left(q_0^{(2)}, x_1 \cdots x_n\right), x_{n+1}\right) \\
&= \bigcup_{p \in \delta_2\left(q_0^{(2)}, x_1 \cdots x_n\right)} \delta_2\left(p, x_{n+1}\right) \\
&= \delta_2\left(q_0^{(2)}, x_1 \cdots x_n x_{n+1}\right)
\end{aligned} \tag{5.17}$$

Bei diesem Beweis haben wir folgendes Wissen angewendet. Da es ein DFA ist, nimmt man den Zustand, der mit den ersten n Symbolen (Von diesen wissen wir aufgrund der Behauptung, dass für diese die Übergangsfunktion δ_2 gleichwertig mit δ_1 ist.) erreicht hat und vereinigt diesen mit dem Zustand, der durch das $n+1$. Symbol erreicht wird. Und die Vereinigung der Teilmengen der ersten $\{1 \dots n\}$ Symbole zusammen mit dem Symbol x_{n+1} ist gleichbedeutend mit der Übergangsfunktion δ_2 angewendet auf die Symbole $\{1 \dots n+1\}$. Nun ist noch zu zeigen, dass auch die selben Wörter akzeptiert werden.

$$\begin{aligned}
\forall x \in \Sigma^* \quad x \in T\left(M_2\right) &\Leftrightarrow \delta_2\left(q_0^{(2)}, x\right) \cap F_2 \neq \varnothing \\
&\Leftrightarrow \delta_2\left(q_0^{(2)}, x\right) \in F_1 \\
&\Leftrightarrow \delta_1\left(\left\{q_0^{(2)}\right\}, x\right) \in F_1 \\
&\Leftrightarrow x \in T\left(M_1\right) \\
\Rightarrow L = T\left(M_2\right) &= T\left(M_1\right)
\end{aligned} \tag{5.18}$$

Auch dieser Beweis ist wieder mittels der Definitionen von δ_1 geführt worden.

Somit steht fest, dass es für alle nichtdeterministischen endlichen Automaten $M = \left(Q, \Sigma, \delta, q_0, F\right)$ ein deterministischer endlicher Automat $M' = \left(Q', \Sigma, \delta, q'_0, F'\right)$ existiert für den die folgenden zwei Aussagen gelten:

1. $T\left(M\right)) = T\left(M'\right)$
2. $\left|Q'\right| \leq 2^{|Q|}$

Die erste Aussage haben wir eben bewiesen. Die zweite Aussage folgt aus der Betrachtung der möglichen Potenzmenge. Hier der kurze Induktionsbeweis.

Induktionsbehauptung:

Für alle $n \geq 0$ mit $|S| = n$ gilt $\left|P\left(S\right)\right| = \left|2^S\right| = 2^{|S|}$.

Induktionsverankerung:

Für $n = 0$ gilt $P\left(S\right) = \{\varnothing\}$, $\left|P\left(S\right)\right| = 1 = 2^0$.

Induktionsschritt ($n \to n+1$):

Bei der Betrachtung der Symbole $\left\{s_1, \dots, s_n, s_{n+1}\right\}$ steht folgendes fest. Die Anzahl der Teilmengen ohne s_{n+1} beträgt 2^n. Die Anzahl der Teilmengen mit s_{n+1} beträgt 2^n. Zusammen gibt dies: $2^n + 2^n = 2 \cdot 2^n = 2^{n+1}$. Damit ist die Behauptung gezeigt.

5.1.4 Beispiel 1 (Konstruktion eines DFA aus einem NFA)

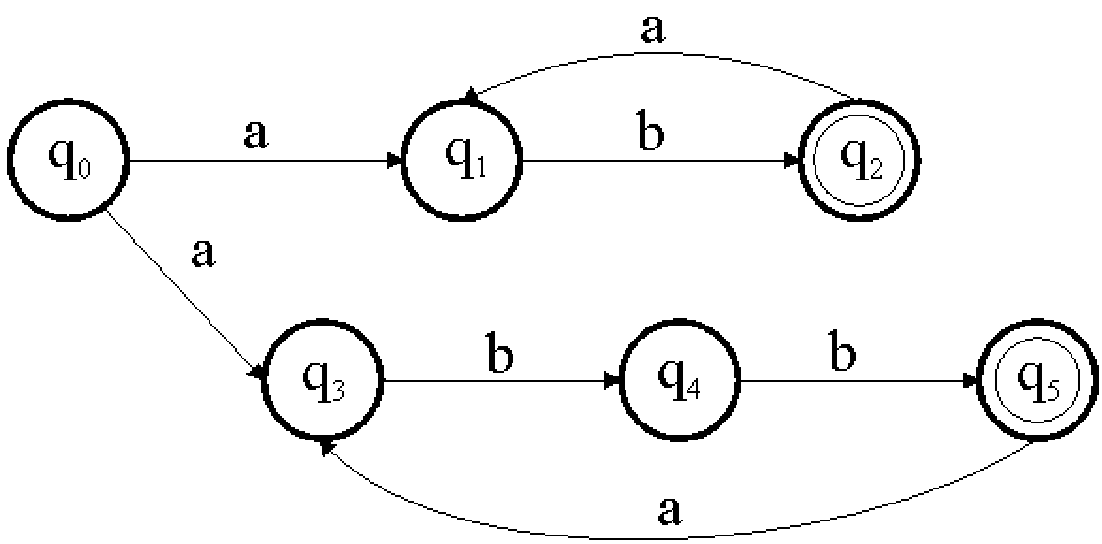


Abbildung 7 – ein nichtdeterministischer, endlicher Automat M

Dieser Automat akzeptiert offensichtlich die Sprache $T(M)=\left\{(ab)^n \middle| n\geq 1\right\}\cup\left\{(abb)^n \middle| n\geq 1\right\}$.

Ein deterministischer endlicher Automat könnte so aussehen:

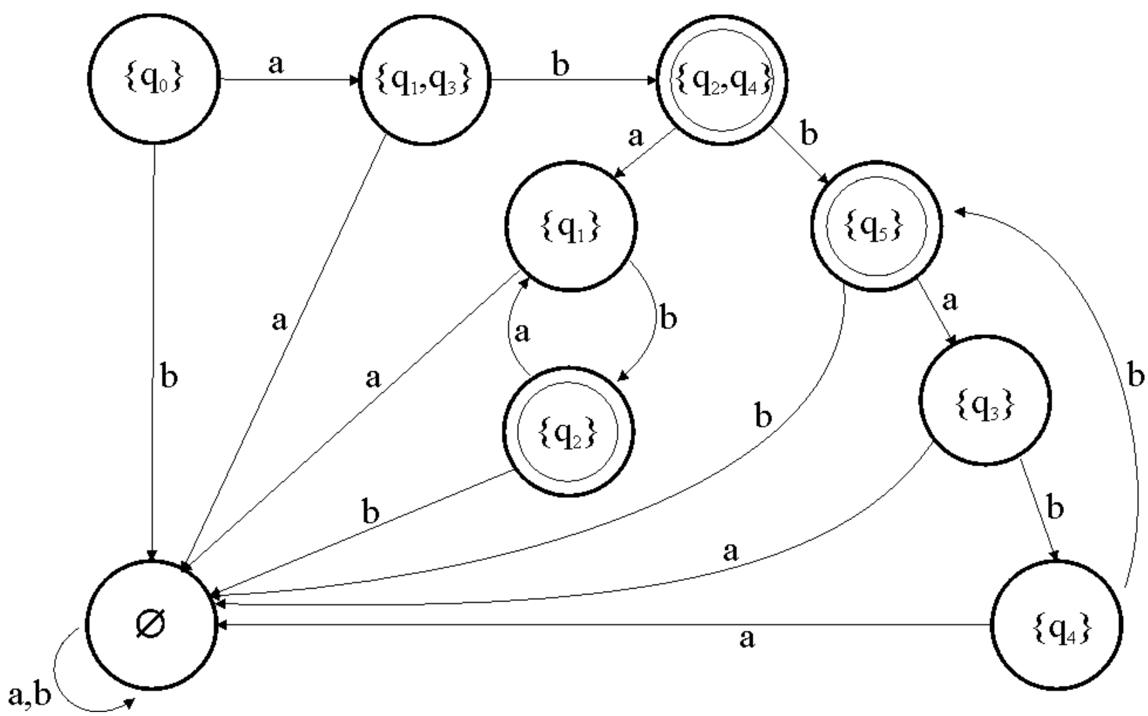


Abbildung 8 - Der DFA, der zu' M äquivalent ist

In diesem Beispiel hat der DFA M' 9 Zustände, während der NFA M 6 Zustände hatte. Und das liegt noch einiges unter den maximal 2^6 möglichen Zuständen.

5.1.5 Beispiel 2 (Konstruktion eines DFA aus einem NFA)

Hier nun ein etwas komplexeres Beispiel. Es soll ein NFA umgewandelt werden, der die Sprache $L_4 = \{0,1\}^* \cdot \{1\} \cdot \{0,1\}^3$ akzeptiert. Ein solcher NFA kann so aussehen:

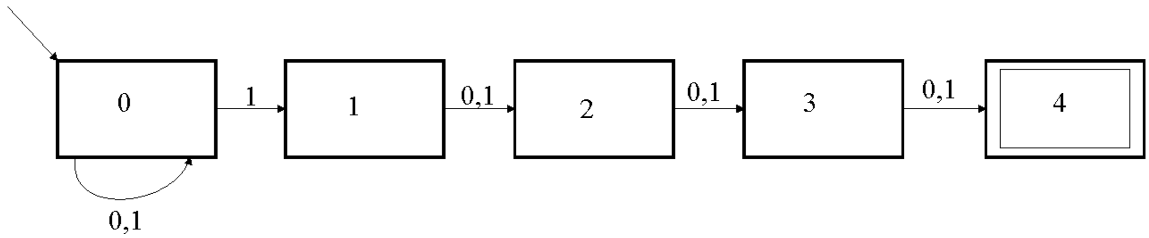


Abbildung 9: NFA zur Sprache L_4

Einen DFA für die Sprache bildet dann:

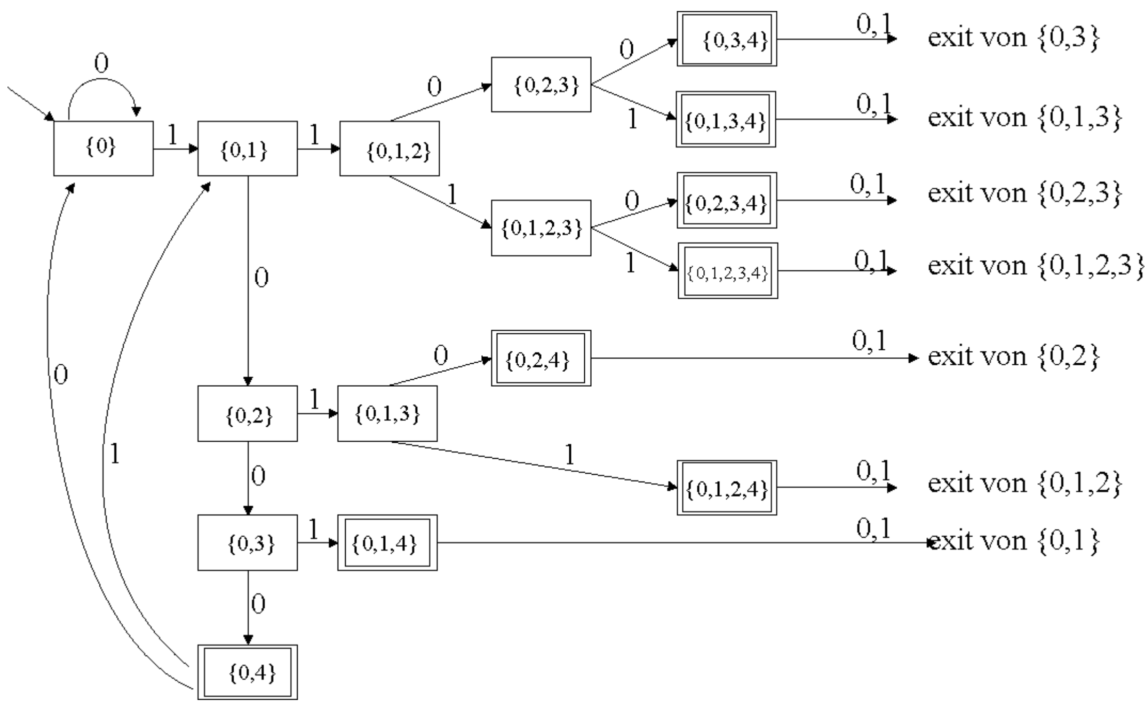


Abbildung 10: der aus dem NFA erzeugte DFA für die Sprache L_4

Bei diesem Beispiel benötigt der NFA 5 Zustände, der DFA jedoch 16 Zustände. Es stellt sich nun die Frage, ob diese Konstruktion auch wirklich minimal ist.

5.2 Satz über die Minimierung von Zuständen

Für alle $n \geq 1$ gilt: Die Sprache $L_n = \{0,1\}^* \cdot \{1\} \cdot \{0,1\}^{n-1}$ kann von einem nichtdeterministischen Automaten mit $(n+1)$ Zuständen akzeptiert werden. Dann folgt für jeden deterministischen Automaten $M = (Q, \Sigma, \delta, q_0, F)$ mit der Sprache $L_n = T(M) \Rightarrow |Q| \geq 2^n$.

Beweis:

Sei $L_n = \{0,1\}^* \cdot \{1\} \cdot \{0,1\}^{n-1}$ gegeben. Weiterhin auch ein NFA M der diese Sprache akzeptiert, also $L_n = T(M)$. Ein Graph dieses Automaten kann wie folgt aussehen:

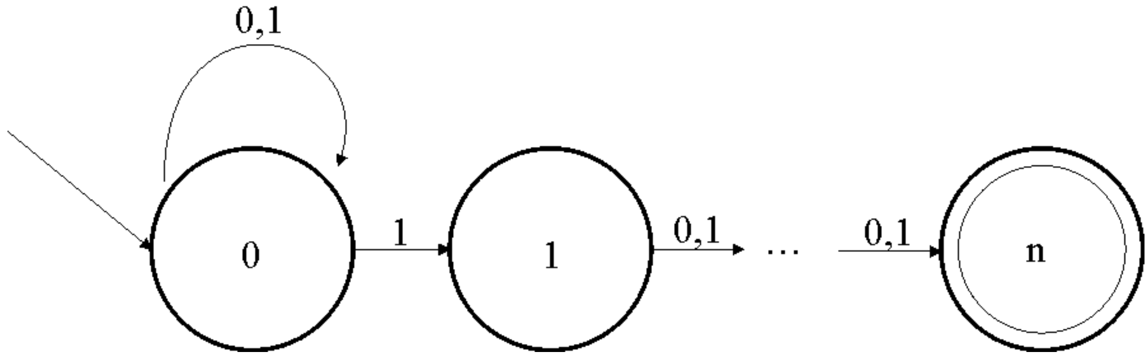


Abbildung 11: NFA für die Sprache Ln

Sei nun weiterhin $M = (Q, \Sigma, \delta, q_0, F)$ ein DFA, der die selbe Sprache akzeptiert, also $L_n = T(M)$.

Behauptung: Jedes Wort des möglichen Eingabealphabetes (von 2^n möglichen Wörtern) muss von dem Automaten unterschieden werden können.

$$\forall x, y \in \{0,1\}^n : x \neq y \Rightarrow \delta(q_0, x) \neq \delta(q_0, y) \tag{5.19}$$

Angenommen zwei gleichlange Wörter aus $\{0,1\}^n$, die unterschiedlich sind (d.h. sie unterscheiden sich nach einigen gleichen Zeichen an einer Stelle, danach folgt ein Wort z) hätten dieselben Nachfolgezustände.

$$\begin{aligned} & x, y \in \{0,1\}^n \wedge x \neq y \\ \Rightarrow \quad & x = x_1 \; 1 \; z \\ & y = y_1 \; 0 \; z \\ & |x_1| = |y_1| \\ \textit{angenommen}: \quad & \delta(q_0, x) = \delta(q_0, y) \end{aligned} \tag{5.20}$$

Dann werden entweder beide Wörter akzeptiert, oder aber keines. In beiden Fällen macht der Automat einen Fehler.

$$\begin{aligned} \Rightarrow \quad & \delta\left(q_0, x_1 \; 1 \; z \; 0^{n-|z|-1}\right) && = && \delta\left(q_0, x \; 0^{n-1-|z|}\right) \\ = \quad & \delta\left(\delta(q_0, x), 0^{n-1-|z|}\right) && = && \delta\left(\delta(q_0, y), 0^{n-1-|z|}\right) \\ = \quad & \delta\left(q_0, y \; 0^{n-1-|z|}\right) && = && \delta\left(q_0, y_1 \; z \; 0^{n-1-|z|}\right) \\ \Rightarrow \quad & x_1 \; 1 \; \underbrace{z \; 0^{n-1-|z|}}_{n-1} \in T(M) && \Leftrightarrow && y_1 \; 0 \; \underbrace{z \; 0^{n-1|z|}}_{n-1} \in T(M) \end{aligned} \tag{5.21}$$

Der Widerspruch liegt darin, dass $x_1 \; 1 \; z \; 0^{n-1-|z|}$ Element der Sprache L_n ist, $y_1 \; 0 \; z \; 0^{n-1-|z|}$ jedoch nicht, trotzdem wird für beide Wörter die selbe Entscheidung getroffen, folglich benötigt man für unterschiedliche Eingabewörter unterschiedliche Zustände. Die maximale Anzahl der Zustände beträgt, wie schon vorher gezeigt 2^n.

$$\begin{aligned} & x_1 \; 1 \; z \; 0^{n-1-|z|} \in L_n \\ & y_1 \; 0 \; z \; 0^{n-1-|z|} \notin L_n \\ & \Rightarrow \forall x, y \in \{0,1\}^n : x \neq y \Rightarrow \delta(q_0, x) \neq \delta(q_0, y) \\ & \left|\{0,1\}^n\right| = 2^n \Rightarrow |Q| \geq 2^n \end{aligned} \tag{5.22}$$

6 Vorlesung vom 25. April 2000

6.1 Nachtrag zur vorherigen Vorlesung

Gegeben ist die folgende (reguläre) Sprache:

$$T(M) = \left\{(ab)^n \middle| n \geq 1\right\} \cup \left\{(abb)^n \middle| n \geq 1\right\} \tag{6.1}$$

Korollar:

Es gibt für jede dieser Sprachen einen nichtdeterministischen, endlichen Automaten, welcher mit n+1 Zuständen auskommt:

$$\begin{aligned} &\forall n \geq 1: \\ &\exists \text{ NFA M}=(\text{Q},\Sigma,\delta,\text{q}_0,\text{F}) \quad \text{mit} \quad L_n = T(M) \wedge |Q| = n+1 \end{aligned} \tag{6.2}$$

Es gibt für jede dieser Sprachen einen deterministischen, endlichen Automaten, welcher mit weniger als 2^{n+1} Zuständen auskommt:

$$\begin{aligned} &\forall n \geq 1: \\ &\exists \text{ DFA M}=(\text{Q},\Sigma,\delta,\text{q}_0,\text{F}) \quad \text{mit} \quad L_n = T(M) \wedge |Q| < 2^{n+1} \end{aligned} \tag{6.3}$$

Jeder deterministische, endliche Automaten, der eine dieser Sprachen beschreibt, besitzt mindestens 2^n Zustände.

$$\begin{aligned} &\forall n \geq 1: \\ &\forall \text{ DFA M}=(\text{Q},\Sigma,\delta,\text{q}_0,\text{F}): \quad L_n = T(M) \Rightarrow |Q| \geq 2^n \end{aligned} \tag{6.4}$$

Satz:

Es gibt für jede dieser Sprachen einen nichtdeterministischen, endlichen Automaten, welcher mit n Zuständen auskommt:

$$\begin{aligned} &\forall n \geq 1: \\ &\exists \text{ NFA M}=(\text{Q},\Sigma,\delta,\text{q}_0,\text{F}) \quad \text{mit} \quad L_n = T(M) \wedge |Q| = n \end{aligned} \tag{6.5}$$

Jeder deterministische, endliche Automaten, der eine dieser Sprachen beschreibt, besitzt mindestens 2^n Zustände.

$$\begin{aligned} &\forall n \geq 1: \\ &\forall \text{ DFA M}=(\text{Q},\Sigma,\delta,\text{q}_0,\text{F}): \quad L_n = T(M) \Rightarrow |Q| \geq 2^n \end{aligned} \tag{6.6}$$

Beweis:

Der Beweis gestaltet sich umfangreich, deswegen wird an dieser Stelle darauf verzichtet.

6.2 Mathematischer Hintergrund

6.2.1 Erweiterung des Begriffes „Äquivalenzrelation“

Man spricht von einer „*Äquivalenzrelation*“, wenn eine Relation reflexiv, transitiv und symmetrisch ist, und von einer Ordnung, wenn eine Relation reflexiv, transitiv und antisymmetrisch ist. Zum Beispiel ist die normale Vergleichsoperation auf der Menge der reellen Zahlen („=“) eine Äquivalenzrelation, und die kleiner-gleich-Operation („≤“) eine Ordnung.

Unter dem „*Index*“ einer Äquivalenzrelation versteht man die Maximalzahl paarweise verschiedener Äquivalenzklassen.

6.2.2 Rechts-Invarianz

Eine Äquivalenzrelation $R \subseteq S \times S$ auf einer algebraischen Struktur $\langle S, \circ \rangle$ mit einer binären Verknüpfung $\circ : S \times S \rightarrow S$ heißt „*rechts-invariant*“

$$\underset{\text{Df}}{\Leftrightarrow} xRy \Rightarrow \forall z \in S : \underbrace{(x \circ z)}_{\in S} R \underbrace{(y \circ z)}_{\in S} \tag{6.7}$$

6.2.3 Links-Invarianz

Eine Äquivalenzrelation $R \subseteq S \times S$ auf einer algebraischen Struktur $\langle S, \circ \rangle$ mit einer binären Verknüpfung $\circ : S \times S \rightarrow S$ heißt „links-invariant“

$$\underset{\text{Df}}{\Leftrightarrow} xRy \Rightarrow \forall z \in S : \underbrace{(z \circ x)}_{\in S} R \underbrace{(z \circ y)}_{\in S} \tag{6.8}$$

6.2.4 Kongruenzrelation

Eine Äquivalenzrelation $R \subseteq S \times S$ auf einer algebraischen Struktur $\langle S, \circ \rangle$ mit einer binären Verknüpfung $\circ : S \times S \rightarrow S$ heißt „Kongruenzrelation“:

$$\begin{aligned} R \text{ Kongruenzrelation} \quad & \underset{\text{Df}}{\Leftrightarrow} xRy \Rightarrow \forall w, z \in S : \underbrace{(w \circ x \circ z)}_{\in S} R \underbrace{(w \circ y \circ z)}_{\in S} \Leftrightarrow \\ & \Leftrightarrow (R \text{ ist rechts-invariant}) \wedge (R \text{ ist links-invariant}) \end{aligned} \tag{6.9}$$

Eine Kongruenzrelation ist eine mit der Struktur verträgliche Äquivalenzrelation. Zum Beispiel ist auf dem Körper der reellen Zahlen die Gleichheitsoperation („=“) bezüglich der Operationen „Addition“ („+“) und „Multiplikation“ („·“) eine Kongruenzrelation:

$$a = b \Rightarrow \forall x, y \in \mathbb{R} : (x \cdot a \cdot y) = (x \cdot b \cdot y), (x + a + y) = (x + b + y) \tag{6.10}$$

6.2.5 Quotientenmenge

Eine „*Quotientenmenge*“ wird auf Basis einer Menge, und einer Kongruenzrelation definiert. Man versteht dann unter der „Quotientenmenge“ die Menge aller Äquivalenzklassen, welche (bezüglich der gegebenen Kongruenzrelation) innerhalb der gegebenen Menge existieren.

Gegeben sei die algebraische Struktur $\langle \Sigma^*, \circ \rangle$ und die Kongruenzrelation „~“. Unter der „Quotientenmenge“ von Σ^* und ~ versteht man:

$$\Sigma^* /_{\sim} \underset{\text{Df}}{=} \left\{ [x]_{\sim} \,\middle|\, x \in \Sigma^* \right\} \tag{6.11}$$

6.2.6 Quotientenmonoid

Unter dem „*Quotientenmonoid*“ versteht man die Zusammenfassung der Quotientenmenge mit einer Operation auf dieser Menge. Die Abbildung $f : \Sigma^* \rightarrow \Sigma^* /_{\sim}$ (der Elemente der ursprünglichen Menge auf die der Quotientenmenge) definiert man sinnvollerweise als einen Homomorphismus[3]. Die neue Operation für das Quotientenmonoid sei „$\bullet$“.

Das „Quotientenmonoid“ ergibt sich dann als:

$$\left\langle \Sigma^* /_{\sim}, \bullet \right\rangle \text{ bzw. } \left\langle \left\{ [x]_{\sim} \,\middle|\, x \in \Sigma^* \right\}, \bullet \right\rangle \tag{6.12}$$

[3] Bei der homomorphen Eigenschaft handelt es sich nicht um eine zwangsläufige Konsequenz, oder um eine durch das Monoid definierten Eigenschaft. Vielmehr wird der Homomorphismus (willkürlich) gefordert, um die neue, algebraische Struktur als „Ring“ zu erhalten

Die Rechenregeln für das Quotientenmonoid leiten sich direkt aus der homomorphen Eigenschaft der Abbildung her:

$$[x]\bullet[y] \underset{\mathrm{Df}}{=} [x\circ y] \tag{6.13}$$

$$\begin{aligned}
&[x]=[x'] \\
&[y]=[y'] \\
&\Downarrow \\
&x \sim x' \\
&y \sim y' \\
&\Downarrow \text{Kongruenz von } \sim \\
&x\circ y \sim x'\circ y \\
&x'\circ y \sim x'\circ y' \\
&\Downarrow \text{Transitivität von } \sim \\
&x\circ y \sim x'\circ y' \\
&\Downarrow \\
&[x\circ y]=[x'\circ y']
\end{aligned} \tag{6.14}$$

Man sieht an der obigen Herleitung, wie bedeutsam die Eigenschaften „Kongruenz“ und „Transitivität“ der Kongruenzrelation „~“ für die Quotientenmenge sind.

Beispiel 1:

Gegeben sei die Menge der natürlichen Zahlen $\mathbb{N}$ mit der Verknüpfung „·“ (Multiplikation), und ein Modul *m*. Zwei Zahlen gelten als äquivalent bezüglich „~“, wenn sie den gleichen Rest bei einer Division durch *m* abwerfen:

$$\begin{aligned}
&a,b\in\mathbb{N} \\
&a\sim b \Leftrightarrow a \bmod m = b \bmod m
\end{aligned} \tag{6.15}$$

Gemäß dieser Definition ist z.B. bei m=5 die Zahl 11 äquivalent zu 31: 11~31. Es handelt sich bei „~“ um eine Äquivalenzrelation, da die Bedingungen Reflexitivität, Transitivität und Symmetrie erfüllt sind (der Nachweis dieser Eigenschaften sollte keine Probleme bereiten).

Es handelt sich bei „~“ aber nicht nur um eine Äquivalenzrelation, sondern sogar um eine Kongruenzrelation. Der Nachweis hierfür ist etwas aufwendiger:

Nach (6.9) gilt:

$$\begin{aligned}
\sim \text{ Kongruenzrelation} \quad &\underset{\mathrm{Df}}{\Leftrightarrow} a\sim b \Rightarrow \forall x,y\in\mathbb{N}: \underbrace{(x\cdot a\cdot y)}_{\in\mathbb{N}} \sim \underbrace{(x\cdot b\cdot y)}_{\in\mathbb{N}} \Leftrightarrow \\
&\Leftrightarrow a \bmod m = b \bmod m \Rightarrow \forall x,y\in\mathbb{N}: \underbrace{(x\cdot a\cdot y)}_{\in\mathbb{N}} \bmod m = \underbrace{(x\cdot b\cdot y)}_{\in\mathbb{N}} \bmod m
\end{aligned}$$

Die abschließende Implikation sollte nach kurzem Nachdenken für jedermann ersichtlicherweise korrekt sein; ein Beweis an dieser Stelle ist nicht nötig.

Daher gelten gemäß (6.13) und (6.14) die folgenden Rechengesetze:

$$\begin{aligned}
&\forall a,b\in\mathbb{N} \\
&a \bmod m \bullet b \bmod m = a\cdot b \bmod m \\
&a \bmod m = a' \bmod m, b \bmod m = b' \bmod m \\
&\Downarrow \\
&a\cdot b \bmod m = a'\cdot b' \bmod m
\end{aligned} \tag{6.16}$$

6.3 Reguläre Ausdrücke

6.3.1 Kleenesche Hülle

Sei Σ eine endliche Menge von Symbolen und seien L, L_1 und L_2 Mengen von Zeichenketten aus Σ^*. Die Konkatenation von L_1 und L_2 – geschrieben als L_1L_2 – ist die Menge $\{xy \mid x \text{ ist aus } L_1 \text{ und } y \text{ ist aus } L_2\}$. Die Zeichenketten in L_1L_2 werden gebildet, indem man an eine aus L_1 gewählte Zeichenkette eine Zeichenkette aus L_2 anhängt und das in allen möglichen Kombinationen. Wir definieren nun:

$$
\begin{aligned}
&L^0 \underset{\text{Df}}{=} \{\in\} \\
&L^i \underset{\text{Df}}{=} LL^{i-1} \quad \forall i \geq 1 \\
&L^* = \bigcup_{i=0}^{\infty} L^i \qquad \text{(Hülle, Kleenesche Hülle)} \\
&L^+ = \bigcup_{i=1}^{\infty} L^i \qquad \text{(positive Hülle)}
\end{aligned}
\tag{6.17}
$$

L^* bezeichnet also alle Wörter, die durch die Konkatenation einer beliebigen Anzahl von Wörtern aus L entstehen.

Beispiel:

Sei $L_1 = \{10,1\}$ und $L_2 = \{011,11\}$. Dann ist $L_1L_2 = \{10011,1011,111\}$. Ebenso gilt $\{10,11\}^* = \{\in,10,11,1010,1011,1110,1111,...\}$.

6.3.2 Reguläre Ausdrücke

Die von endlichen Automaten akzeptierten Sprachen lassen sich durch einfache Ausdrücke beschreiben, die als „*reguläre Ausdrücke*“ bezeichnet werden. Auf Grund dieser Äquivalenz werden die Sprachen der endlichen Automaten auch „*reguläre Mengen*“ genannt.

Oder umgekehrt: *Reguläre Ausdrücke* bilden einen Formalismus zur Beschreibung *formaler Sprachen*.

Zwei reguläre Ausdrücke heißen „*äquivalent*“, wenn sie die gleiche formale Sprache beschreiben (z.B. [ab]*a und a[ba]*). Die Äquivalenz regulärer Ausdrücke lässt sich über algebraische Gesetze zeigen.

6.3.3 Reguläre Ausdrücke vs. endliche Automaten

Satz:

Die folgenden drei Aussagen sind äquivalent:

3. L ist reguläre Sprache/reguläre Menge
   $\Updownarrow$
4. L ist die Vereinigung von einigen Äquivalenzklassen einer rechts-invarianten Äquivalenzrelation von endlichem Index.
   $\Updownarrow$
5. Sei R_L eine Relation, die wie folgt definiert ist:

$$x \, R_L \, y \Leftrightarrow \left(\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\right) \tag{6.18}$$

   Dann ist R_L von endlichem Index.

Beweis:

$1 \Rightarrow 2$:

Wenn *L* eine reguläre Menge ist, dann existiert ein endlicher Automat (DFA/NFA), der exakt diese Menge beschreibt:

$$L \text{ ist regulär} \Leftrightarrow \exists M : M = (Q, \Sigma, \delta, q_0, F) : L = T(M) \tag{6.19}$$

Sei ρ eine Äquivalenzrelation, welche alle Wörter aus Σ*, die zu dem selben Zustand des Automaten führen, in einer Äquivalenzklasse zusammenfasst:

$$x, y \in \Sigma^* : x \;\rho\; y \underset{\text{Df}}{\Leftrightarrow} \delta(q_0, x) = \delta(q_0, y) \tag{6.20}$$

Die Äquivalenzrelation ρ ist rechts-invariant:

$$\forall x, y, z \in \Sigma^* : \delta(q_0, x) = \delta(q_0, y) \Rightarrow \delta(q_0, xz) = \delta(q_0, yz) \tag{6.21}$$

Dann lässt sich hieraus die Behauptung folgern:

$$\text{index}(\rho) \leq |Q| \tag{6.22}$$

Wieso lässt sich diese Behauptung folgern? Nun, gehen wir von dem Gegenteil aus, nämlich dass $\text{index}(\rho) > |Q|$. Das würde bedeuten, dass es mehr Wörter gibt, als es Zustände gibt, die aber dennoch alle zu verschiedenen Zuständen führen. Der Widerspruch dieser Annahme liegt auf der Hand.

Damit lässt sich die reguläre Menge *L* darstellen als Menge der Eingabewörter, welche zu einem der akzeptierenden Zustände des Automaten führt. Jeder akzeptierende Zustand des Automaten kann bezüglich der Äquivalenzrelation (definitionsgemäß) nur von einer Klasse von Eingabewörtern erreicht werden.

Die reguläre Menge *L* können wir darstellen als Vereinigung aller akzeptierenden Zustände des Automaten, und damit als Vereinigung einiger Äquivalenzklassen einer rechts-invarianten Äquivalenzrelation von endlichem Index:

$$L = \bigcup_{\delta(q_0, x) \in F} [x]_\rho \tag{6.23}$$

$2 \Rightarrow 3$:

Im vorangegangenen Teil wurde die Äquivalenzrelation ρ eingeführt, diese benutzen wir weiterhin. Darüber hinaus definieren eine (zweite) Äquivalenzrelation R_L wie folgt:

$$\begin{aligned} \forall x, y \in \Sigma^* : x \; R_L \; y \quad & \Leftrightarrow \left(\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L\right) \Leftrightarrow \\ & \Leftrightarrow \left(\forall z \in \Sigma^* : (xz \in L \wedge yz \in L) \vee (xz \notin L \wedge yz \notin L)\right) \end{aligned} \tag{6.24}$$

Die Definition der Äquivalenzrelation R_L ist etwas kompliziert. Textuell formuliert bedeutet (6.24), dass alle jenen Wörter aus Σ* in Relation stehen, die a) entweder beide in der Sprache enthalten sind, oder beide nicht in der Sprache enthalten sind; und b) die Bedingung a) auch dann noch erfüllen, wenn ihnen ein beliebiges Wort „angehängt wird“.

Die Definition dieser Äquivalenzrelation erinnert an die Zustandsminimierung via Implikationstafel. Man bezeichnet dabei alle Zustände als „äquivalent“, die bei den selben Eingabesymbolen in die selben Nachfolgezustände, oder in äquivalente Nachfolgezustände überführt werden.

Behauptung:

$$\forall x, y \in \Sigma^* : x \;\rho\; y \Rightarrow x \; R_L \; y \tag{6.25}$$

Beweis zu (6.25):

Bei dem Beweis machen wir uns die Tatsache zunutze, dass die Äquivalenzrelation ρ rechts-invariant ist:

$$\begin{aligned}
&\forall x,y\in\Sigma^*: x\ \rho\ y \Rightarrow \forall z\in\Sigma^*: xz\ \rho\ yz\\
&\Downarrow\\
&\forall x,y\in\Sigma^*: x\ \rho\ y \Rightarrow \left(\forall z\in\Sigma^*: xz\in L \Leftrightarrow yz\in L\right)\\
&\Downarrow\\
&\forall x,y\in\Sigma^*: x\ \rho\ y \Rightarrow \left(\forall z\in\Sigma^*: \left(xz\in L \wedge yz\in L\right)\vee\left(xz\notin L\wedge yz\notin L\right)\right)\\
&\Updownarrow \text{ siehe oben}\\
&\forall x,y\in\Sigma^*: x\ \rho\ y \Rightarrow x\ R_L\ y
\end{aligned} \tag{6.26}$$

Aus der Tatsache, dass jedes Paar von Eingabewörtern aus Σ*, dass in der Äquivalenzrelation ρ enthalten ist, automatisch auch in der Äquivalenzrelation R_L enthalten ist, resultiert, dass die Äquivalenzklasse eines beliebigen Eingabewortes bezüglich der Äquivalenzrelation ρ eine Teilmenge der Äquivalenzklasse bezüglich der Äquivalenzrelation R_L ist:

$$\forall x\in\Sigma^*: [x]_\rho \subseteq [x]_{R_L} \tag{6.27}$$

Daraus folgt, dass es bezüglich der Äquivalenzrelation ρ mindestens so viele (disjunkte) Äquivalenzklassen geben muss, wie bezüglich der Äquivalenzrelation R_L:

$$\text{index}(R_L) \le \text{index}(\rho) \le \text{"endlich"} \tag{6.28}$$

Daraus wiederum folgt, dass auch der Index der Äquivalenzrelation R_L endlich sein muß:

$$\text{index}(\mathrm{R_L}) \text{ ist endlich} \tag{6.29}$$

2 ⇒ 3:

Behauptung:

Die Äquivalenzrelation R_L ist rechts-invariant.

Beweis:

$$\begin{aligned}
&\forall x,y,w,z\in\Sigma^*: x\ R_L\ y\\
&\Updownarrow \text{ Definition } \mathrm{R_L}\\
&\left(x\in L \Leftrightarrow y\in L\right)\\
&\Updownarrow \text{ Determinismus des Automaten}\\
&\left(x(zw)\in L \Leftrightarrow y(zw)\in L\right)\\
&\Updownarrow \text{ Assoziativität der Konkatenation}\\
&\left((xz)w\in L \Leftrightarrow (yz)w\in L\right)\\
&\Updownarrow\\
&\forall x,y,z\in\Sigma^*: x\ R_L\ y \Leftrightarrow xz\ R_L\ yz\\
&\Downarrow\\
&R_L \text{ ist rechts-invariant}
\end{aligned} \tag{6.30}$$

6.4 Nerode Automat

6.4.1 Definition des Nerode Automaten

Mit dem bisher erlangten Wissen, ist es möglich, einen minimalen Automaten zu erstellen, der eine reguläre Sprache repräsentiert. Dieser Automat heißt „*Nerode Automat*“.

Zur Konstruktion des Nerode Automaten wird erneut die Äquivalenzrelation R_L herangezogen, die wie folgt definiert ist:

$$\begin{aligned} \forall x, y \in \Sigma^* : x\, R_L\, y \quad & \Leftrightarrow \left(\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L \right) \Leftrightarrow \\ & \Leftrightarrow \left(\forall z \in \Sigma^* : \left(xz \in L \wedge yz \in L \right) \vee \left(xz \notin L \wedge yz \notin L \right) \right) \end{aligned} \tag{6.31}$$

Die Menge der unterschiedlichen Äquivalenzklassen, die durch R_L definiert werden, wird im Folgenden mit „Q_N“ bezeichnet:

$$Q_N \underset{\text{Df}}{=} \left\{ [x]_{R_L} \middle| x \in \Sigma^* \right\} \tag{6.32}$$

Der Nerode Automat ist definiert als:

$$M_N \underset{\text{Df}}{=} \left\{ Q_N, \Sigma, \delta_N, [e]_{R_L}, F_N \right\} \qquad \text{(Nerode Automat)} \tag{6.33}$$

mit

$$F_N = \left\{ [x]_{R_L} \middle| x \in L \right\} \tag{6.34}$$

$$\delta_N \left([x], a \right) \underset{\text{Df}}{=} [xa] \tag{6.35}$$

δ_N ist damit wohldefiniert.

6.4.2 Repräsentantenunabhängigkeit des Nerode Automaten:

Im Folgenden wird gezeigt, dass die Übergangsfunktion δ_N wohldefiniert ist, d.h. ihre Definition repräsentantenunabhängig ist. Dazu ist es nötig, zu zeigen, dass die Übergangsfunktion für alle Eingabewörter, die äquivalent sind, den selben Nachfolgezustand liefert.

Zu zeigen:

$$[x]_{R_L} = [y]_{R_L} \Rightarrow \delta_N \left([x]_{R_L}, a \right) = \delta_N \left([y]_{R_L}, a \right) \tag{6.36}$$

Beweis:

$$\begin{gathered} [x]_{R_L} = [y]_{R_L} \\ \Downarrow \\ x\, R_L\, y \\ \Downarrow \\ xa\, R_L\, ya \\ \Downarrow \\ [xa]_{R_L} = [ya]_{R_L} \\ \Downarrow \\ \delta_N \left([x]_{R_L}, a \right) = \delta_N \left([y]_{R_L}, a \right) \\ \Downarrow \\ M_N \text{ ist endlicher Automat} \end{gathered} \tag{6.37}$$

6.4.3 Regularität des Nerode Automaten:

Im Folgenden wird bewiesen, dass die Sprache, die der Nerode Automat erzeugt/definiert, eine reguläre Sprache ist.

$$
\begin{aligned}
x \in T(M_N) \quad & \Leftrightarrow \delta_N([e],x) \in F_N \\
& \Leftrightarrow [x] \in F_N \\
& \Leftrightarrow x \in L
\end{aligned}
\tag{6.38}
$$

$$\Rightarrow T(M_N) = L \Rightarrow$$

$$\Rightarrow L \text{ ist regulär}$$

6.4.4 Minimalität des Nerode Automaten

Im Folgenden wird bewiesen, dass der Nerode Automat minimal ist, d.h. dass jeder endliche Automat, der die selbe Sprache L definiert, mindestens genauso viele Zustände haben muß wie M_N.

Satz:

Der Nerode Automat ist minimal.

Beweis:

Der Beweis ist ein Widerspruchsbeweis. Es wird angenommen, dass sehr wohl ein Automat existiert, der die selbe Sprache L definiert, der aber mit weniger Zuständen auskommt:

$$\exists M : M = (Q,\Sigma,\delta,q_0,F) \quad T(M) = L \quad |Q| < |Q_N| \tag{6.39}$$

Die Definition der Äquivalenzrelation ρ wird aus (6.20), (6.22) und (6.28) übernommen:

$$x, y \in \Sigma^* : x \;\rho\; y \underset{\text{Df}}{\Leftrightarrow} \delta(q_0,x) = \delta(q_0,y) \tag{6.40}$$

$$\text{index}(\rho) \le |Q| \tag{6.41}$$

$$\text{index}(R_L) \le \text{index}(\rho) \le \text{"endlich"} \tag{6.42}$$

Weil die Menge Q_N der Menge der Äquivalenzklassen entspricht, die durch die Äquivalenzrelation R_L definiert werden (siehe (6.32)), gilt:

$$|Q_N| = \text{index}(R_L) \tag{6.43}$$

Aus (6.39), (6.41), (6.42) und (6.43) folgt:

$$
\begin{gathered}
\text{index}(\rho) \le |Q| < |Q_N| = \text{index}(R_L) \le \text{index}(\rho) \\
\Downarrow \\
\text{index}(\rho) < \text{index}(\rho) \\
\Downarrow \\
\text{Widerspruch}
\end{gathered}
\tag{6.44}
$$

6.4.5 Eindeutigkeit des Nerode Automaten

Satz:

Sei $M = (Q,\Sigma,\delta,q_0,F)$ ein minimaler Automat für eine Sprache L mit $T(M) = L$.

Sei $M_N = \{Q_N,\Sigma,\delta_N,q_{0N},F_N\}$ der Nerode Automat für L mit $T(M_N) = L$. Dann ist M isomorph zu M_N. D.h. es existiert ein Isomorphismus

$$h : Q \to Q_N \quad \text{mit} \quad h(\delta(q,a)) = \delta_N(h(q),a) \tag{6.45}$$

Beweis:

Weil sowohl M als auch M_N minimal sind, müssen sie über die gleiche Anzahl an Zuständen verfügen:

$$|Q| = |Q_N| \tag{6.46}$$

Sei *h()* so definiert, dass es jeden Zustand q aus der Menge der Zustände Q auf eine Äquivalenzklasse bezüglich der Äquivalenzrelation R_L abbilde (wir erinnern uns: die Äquivalenzklassen bezüglich der Äquivalenzrelation R_L liegen dem konstruierten Nerode Automaten zugrunde).:

$$h(q) \underset{\mathrm{Df}}{=} [x]_{R_L} \Leftrightarrow \exists x : \delta(q_0, x) = q \tag{6.47}$$

7 Vorlesung vom 27. April 2000

7.1 Nerode Automat

7.1.1 Homomorphismus und Isomorphismus

Gegeben seien zwei Gruppen S_1 und S_2 mit $\langle S_1, \cdot \rangle$ und $\langle S_2, \circ \rangle$. Eine Abbildung $h: S_1 \rightarrow S_2$ heißt Homomorphismus, wenn für alle $x, y \in S_1$ gilt

$$\begin{aligned} & h(x \cdot y) = h(x) \circ h(y) \\ & \text{bzw.} \\ & \mathrm{h}(\cdot(\mathrm{x},\mathrm{y})) = \circ(\mathrm{h}(\mathrm{x}),\mathrm{h}(\mathrm{y})) \end{aligned} \tag{7.1}$$

Eine Illustration verdeutlicht das

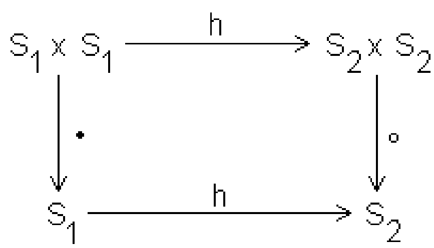


Abbildung 12 - Homomorphismus

Ist eine homomorphe Abbildung bijektiv, so wird sie isomorph genannt. Man sagt, die Struktur beider Gruppen ist gleich, sie unterscheiden sich nur in den Bezeichnungen der Elemente.

7.1.2 Minimalität und Eindeutigkeit des Nerode Automaten

Nachdem in den vorangegangenen Abschnitten gezeigt wurde, dass der Nerode Automat minimal ist, vergleichen wir ihn jetzt mit anderen minimalen Automaten. Dabei werden wir feststellen, dass alle minimalen Automaten (DFA) einer Sprache L die gleiche "Verdrahtung" ihrer Zustände haben und sich lediglich in der Bezeichnung der Zustände unterscheiden, sie also isomorph sind.

Sei L eine Sprache, die von einem minimalen Automaten M und dem Nerode Automaten M_N erkannt wird

$$\begin{aligned} M &= (Q, \Sigma, \delta, q_0, F) & L &= T(M) \\ M_N &= (Q_N, \Sigma, \delta_N, q_{0N}, F_N) & L &= T(M_N) \end{aligned} \tag{7.2}$$

dann seien M und M_N wie oben beschrieben isomorph, d. h. es existiert ein Homomorphismus

$$\begin{aligned} & h: Q \rightarrow Q_N \\ & h(\delta(q,a)) = \delta_N(h(q),a) \end{aligned} \tag{7.3}$$

Das wird in folgendem Schaubild illustriert

$$\begin{tikzcd} Q_1 \times \Sigma_1 \arrow[r, "h"] \arrow[d, "\delta_1"] & Q_2 \times \Sigma_2 \arrow[d, "\delta_2"] \\ Q_1 \arrow[r, "h"] & Q_2 \end{tikzcd}$$

Abbildung 13 - Homomorphismus zwischen Automat M und Nerode Automat

Es existiert eine Abbildung h, die alle Zustände von M in Zustände in M_N (Äquivalenzklassen) überführt. Formal ausgedrückt

$$h(q) = [x]_{R_L} \Leftrightarrow \exists x : \delta(q_0, x) = q \tag{7.4}$$

Hier stellt sich die Frage, ob h eindeutig definiert ist. Alle Wörter x und y, die den Automaten M in den gleichen Zustand überführen, müssen auch M_N in die gleiche Äquivalenzklasse überführen. Zu Zeigen ist, dass x und y der gleichen Äquivalenzklasse von R_L angehören.

$$\begin{gathered} x, y \in \Sigma^* : \delta(q_0, x) = q = \delta(q_0, y) \\ \overset{?}{\Leftrightarrow} \\ [x]_{R_L} = [y]_{R_L} \end{gathered} \tag{7.5}$$

Beweis:

Zum Beweis führen wir eine neue Äquivalenzrelation ~ ein

$$x \sim y \Leftrightarrow x, y \in \Sigma^* : \delta(q_0, x) = \delta(q_0, y) \tag{7.6}$$

Überführen zwei Wörter x und y den Automaten in den gleichen Zustand, so sind sie äquivalent. Da die Anzahl der Zustände der oben betrachteten Automaten minimal und gleich ist, gilt für den Index (die Anzahl der maximal paarweise verschiedenen Äquivalenzklassen) der ~ Relation

$$index(\sim) \le |Q| = |Q_N| \tag{7.7}$$

Weiterhin gelte

$$x, y \in \Sigma^* : \delta(q_0, x) = \delta(q_0, y) \Rightarrow x \sim y \Rightarrow x R_L y \Rightarrow [x]_{R_L} = [y]_{R_L} \tag{7.8}$$

Somit ist h als eindeutige Abbildung von Zuständen Q auf Äquivalenzklassen aus Q_N definiert.

Könnte es Zustände q in Automat M geben, für die h(q) nicht definiert ist?

$$\exists q \in Q \text{ mit } h(q) \text{ nicht definiert} \Rightarrow \neg\exists x \text{ mit } \delta(q_0, x) = q \tag{7.9}$$

Wenn ein solcher Zustand existieren würde, dann gebe es kein Wort x, dass den Automaten in diesen Zustand überführt und der Zustand somit überflüssig. Damit wäre M nicht mehr minimal und ein Widerspruch läge vor. h ist für alle Zustände q aus Q definiert.

Surjektivität von h

$$[x] \in Q_N, \exists q \in Q \text{ mit } \delta(q_0, x) = q \wedge h(q) = x \tag{7.10}$$

Injektivität von h

Wenn $q \ne p$ zwei verschiedene Zustände aus Q sind und [x] und [y] ihre Abbildungen in Q_N, dann gilt

$$\begin{aligned} h(q) &= [x]_{R_L} \text{ mit } \delta(q_0, x) = q \\ h(p) &= [y]_{R_L} \text{ mit } \delta(q_0, y) = p \end{aligned} \tag{7.11}$$

Dann sind x und y nicht äquivalent zueinander (bezüglich der ~ Relation) und damit gilt Injektivität von h ($x \sim y \Rightarrow xR_L y$ gilt immer).

Isomorphismus von h

Da h injektiv und surjektiv, ist die Abbildung bijektiv. h ist daher ein Isomorphismus. Für einen Zustand q existiert ein x, dass den Automaten M in diesen Zustand überführt.

$$q \in Q \Rightarrow \exists x \text{ mit } \delta(q_0, x) = q \tag{7.12}$$

$$\begin{aligned} \delta_N\left(h(q), a\right) &= \delta_N\left(h\left(\delta\left(q_0, x\right)\right), a\right) \\ &= \delta_N\left([x], a\right) \\ &= [xa] \\ &= h\left(\delta\left(q_0, xa\right)\right) \\ &= h\left(\delta\left(\delta\left(q_0, x\right), a\right)\right) \\ &= h\left(\delta\left(q, a\right)\right) \end{aligned} \tag{7.13}$$

Die Herleitung zeigt, dass es egal ist, ob die Konkatenation von x und a zunächst in der Urmenge (die Zustände von Q) stattfindet und dann nach Q_N abgebildet wird oder umgekehrt.

Damit ist h isomorph und als Korollar kann festgestellt werden, dass alle minimalen Automaten (DFA) isomorph sind. Anders ausgedrückt sind alle minimalen Automaten bis auf Isomorphien eindeutig bestimmt. Die Verdrahtung der Zustände ist gleich, lediglich die Zustandsbezeichnung kann variieren.

7.2 Minimierung von endlichen Automaten

7.2.1 Rekursion der k Relationen

Endliche Automaten können sich in einem nicht minimalen Zustanden befinden. Mit einiger Logik und Transformation lassen sie sich in minimale Automaten überführen, die isomorph zum Nerode Automaten sind (wenn es sich denn um DFAs handelt - bei NFAs ist die Anzahl der nicht isomorphen minimalen Automaten u. U. beliebig groß).

Für die Minimierung - auch Zustandsreduktion genannt - führen wir zunächst eine neue Äquivalenzrelation k zwischen zwei Zuständen q und q' ein. Zustände sind bezüglich k äquivalent, wenn sie Wörter x, deren Länge kleiner gleich k ist, in einen Endzustand überführen oder beide Zustände die Wörter nicht in einen Endzustand überführen.

Der Betrachtung zugrunde liegt ein endlicher Automat $M = \left(Q, \Sigma, \delta, q_0, F\right)$ mit q und q' Zuständen aus Q. Dann seien folgende Äquivalenzrelationen definiert

$$\begin{aligned} q \overset{k}{\sim} q' &\Leftrightarrow \left(\forall x \in \Sigma^*,\ |x| \le k : \delta\left(q, x\right) \in F \Leftrightarrow \delta\left(q', x\right) \in F\right) \\ q \sim q' &\Leftrightarrow \left(\forall x \in \Sigma^*,\ \delta\left(q, x\right) \in F \Leftrightarrow \delta\left(q', x\right) \in F\right) \end{aligned} \tag{7.14}$$

Scheinbar besteht zwischen der auf eine Wortlänge k beschränkten und der allgemeinen ~ Relation ein Zusammenhang. Daher behaupten wir

Behauptung:

$$q \overset{k}{\sim} q' \Leftrightarrow q \overset{k-1}{\sim} q' \wedge \left(\forall a \in \Sigma,\ \delta\left(q, a\right) \overset{k-1}{\sim} \delta\left(q', a\right)\right) \tag{7.15}$$

Beweis:

Die genau-dann-wenn Beziehung der obigen Behauptung beweisen wir in beide Richtungen. Wenn die k Relation gilt, dann muß auch die k-1 Relation gelten, da die k Relation alle Wörter enthält, die auch in k-1 enthalten sind.

Seien ein Wort $z \in \Sigma^*$ mit $|z| \le k-1$ und ein Zeichen $a \in \Sigma$ beliebig gewählt, dann gilt

$$\delta(\delta(q,a),z) \in F \Leftrightarrow \delta(q,az) \in F \Leftrightarrow \delta(q',az) \in F \Leftrightarrow \delta(\delta(q',a),z) \in F \tag{7.16}$$

und die Zustände q und q' gehören zur k-1 Relation

$$\delta(q,a) \overset{k-1}{\sim} \delta(q',a) \tag{7.17}$$

Damit ist gezeigt

$$q \overset{k}{\sim} q' \Rightarrow q \overset{k-1}{\sim} q' \tag{7.18}$$

Gilt bereits die k-1 Relation, so zeigen wir, dass die Behauptung in der anderen Richtung gilt. Für $\forall z \in \Sigma^*$ mit $|z| = k$ gilt

$$\delta(q,z) \in F \Leftrightarrow \delta(q',z) \in F \tag{7.19}$$

Sei $z \in \Sigma^*$ mit $|z| = k$ und $z = aw$ mit $a \in \Sigma \wedge |w| = k-1$. Dann läßt sich zeigen

$$\begin{aligned}
&\delta(q,a) \overset{k-1}{\sim} \delta(q',a) \\
&\Rightarrow \big(\delta(\delta(q,a),w) \in F \Leftrightarrow \delta(\delta(q',a),w) \in F\big) \\
&\Rightarrow \big(\delta(q,aw) = \delta(q,z) \in F \Leftrightarrow \delta(q',aw) = \delta(q',z) \in F\big) \\
&\Rightarrow q \overset{k}{\sim} q'
\end{aligned} \tag{7.20}$$

Somit gilt die Behauptung auch in der anderen Richtung

$$q \overset{k}{\sim} q' \Leftarrow q \overset{k-1}{\sim} q' \tag{7.21}$$

7.2.2 Verfeinerung von Äquivalenzrelationen

Zwischen Äquivalenzrelationen lassen sich Beziehungen ausdrücken, d. h. sie können kongruent oder verschieden sein. Nehmen wir die beiden Äquivalenzrelationen ρ_1 und ρ_2 die auf der Menge S liegen. Sind die ρ_1 und ρ_2 kongruent, dann

$$\begin{aligned}
&\rho_1 \equiv \rho_2 \Leftrightarrow \forall [x]_{\rho_1}\ \exists [y]_{\rho_2} : [x]_{\rho_1} = [y]_{\rho_2} \\
&\forall x, y \in S : x \rho_1 y \Leftrightarrow x \rho_2 y
\end{aligned} \tag{7.22}$$

Andernfalls ist eine Relation kleiner oder echt kleiner als die andere. Man spricht von *Verfeinerung*

$$\begin{aligned}
&\rho_1 \le \rho_2 \Leftrightarrow (x \rho_1 y \Rightarrow x \rho_2 y) \\
&\rho_1 < \rho_2 \Leftrightarrow (\rho_1 \le \rho_2 \wedge \rho_1 \not\equiv \rho_2)
\end{aligned} \tag{7.23}$$

7.2.3 Anwendung auf k Relation zur Zustandsreduktion

Betrachtet man die k Relationen, dann enthält die k Relation höchstens gleich viele Zustände wie die k-1 Relation. k ist daher eine Verfeinerung von k-1

$$\overset{k}{\sim} \le \overset{k-1}{\sim} \tag{7.24}$$

Wir behaupten zudem

$$\overset{k}{\sim} \equiv \overset{k+1}{\sim} \Rightarrow \overset{k}{\sim} \equiv \overset{k+s}{\sim} \quad \forall s \geq 0 \tag{7.25}$$

Beweis:

Die Behauptung kann man durch Induktion beweisen. Für s = 0 ist die Aussage klar, da es sich um dieselbe Relation k handelt. Für alle weiteren nehmen wir an, dass

$$\overset{k}{\sim} \equiv \overset{k+s}{\sim} \tag{7.26}$$

bereits gelte. Dann kann man auf k+s+1 gehen

$$\begin{aligned} q \overset{k+s+1}{\sim} q' &\Leftrightarrow q \overset{k+s}{\sim} q' \wedge \left(\forall a \in \Sigma : \delta(q,a) \overset{k+s}{\sim} \delta(q',a) \right) \\ q \overset{k+1}{\sim} q' &\Leftrightarrow q \overset{k}{\sim} q' \wedge \left(\forall a \in \Sigma : \delta(q,a) \overset{k}{\sim} \delta(q',a) \right) \\ q \overset{k+2}{\sim} q' &\Leftrightarrow q \overset{k+1}{\sim} q' \wedge \left(\forall a \in \Sigma : \delta(q,a) \overset{k+1}{\sim} \delta(q',a) \right) \\ &\dots \\ &\Leftrightarrow q \overset{k}{\sim} q' \end{aligned} \tag{7.27}$$

Korollar:

Der Index der k-1 Relation ist kleiner als der der k Relation

$$Index(\overset{k-1}{\sim}) \leq Index(\overset{k}{\sim}) \tag{7.28}$$

Damit haben wir als Ausgangspunkt für weitere Betrachtungen folgende Aussagen zusammengestellt

$$\overset{k}{\sim} \leq \overset{k-1}{\sim} \tag{7.29}$$

$$Index(\overset{k}{\sim}) \geq Index(\overset{k-1}{\sim}) \tag{7.30}$$

$$\overset{k}{\sim} \equiv \overset{k+1}{\sim} \Rightarrow \forall s \geq 0 : \overset{k}{\sim} \equiv \overset{k+s}{\sim} \tag{7.31}$$

8 Vorlesung vom 2. Mai 2000

8.1 Minimierung mittels Nerode Automaten

8.1.1 Satz über die Gleichwertigkeit von Relationen

Sei $M = (Q,\Sigma,\delta,q_0,F)$ ein endlicher deterministischer Automat mit der Mächtigkeit $|Q| = n$. Dann gilt: $q \sim q' \Leftrightarrow q \overset{n-1}{\sim} q'$.

Der Beweis der „Hinrichtung“. ist trivial und folgt auch schon aus dem Satz der letzten Vorlesung. Die „Rückrichtung“ ist da schon komplizierter. Der Beweis erfolgt indirekt. Angenommen die beiden Relationen währen nicht identisch:

$$q \overset{n-1}{\sim} q' \wedge q \nsim q' \Rightarrow \sim \not\equiv \overset{n-1}{\sim} \tag{8.1}$$

Grundsätzlich gilt, dass ~ eine Verfeinerung von $\overset{n}{\sim}$ ist. Damit muss $\overset{n}{\sim}$ eine echte Verfeinerung von $\overset{n-1}{\sim}$ sein, denn sonst wäre ja $\sim \equiv \overset{n-1}{\sim}$, was ein Widerspruch zur Annahme wäre.

Durch eine Betrachtung der Anzahl der Klassen der Zustände ($\overset{0}{\sim} \geq \overset{1}{\sim} \geq \overset{2}{\sim} \geq \ldots \geq \overset{n-1}{\sim} \geq \overset{n}{\sim}$) erkennt man schnell, dass es mindestens eine Klasse von Zuständen geben muss, und zwar die der akzeptierenden Zustände. Da weiterhin gilt, dass $\overset{n}{\sim}$ eine echte Verfeinerung von $\overset{n-1}{\sim}$ ist, sind alle anderen Relationen echte Verfeinerungen (also gilt $\overset{0}{\sim} > \overset{1}{\sim} > \overset{2}{\sim} > \ldots > \overset{n-1}{\sim} > \overset{n}{\sim}$). Die Anzahl der Klassen wächst durch jeden Index um eins. Führt man diesen Gedankengang fort und beachtet, dass es alles echte Verfeinerungen sind, so gelangt man für den $Index\left(\overset{n}{\sim}\right)$ zu dem Schluss, dass dieser mindestens $n+1$ Klassen enthalten muss.

$$\begin{aligned} & 1 \leq Index\left(\overset{0}{\sim}\right) < Index\left(\overset{1}{\sim}\right) < Index\left(\overset{2}{\sim}\right) < \ldots < Index\left(\overset{n-1}{\sim}\right) < Index\left(\overset{n}{\sim}\right) \\ & \Rightarrow Index\left(\overset{n}{\sim}\right) \geq n+1 \end{aligned} \tag{8.2}$$

Dies ist allerdings ein Widerspruch, da es höchstens n (mit $n = |Q|$) Äquivalenzklassen auftreten können.

8.1.2 Konstruktion eines minimalen Automaten

Sei nun ~ weiterhin wie bisher definiert. Dann gibt es zu dem endlichen Automaten $M = (Q,\Sigma,\delta,q_0,F)$ einen äquivalenten minimalen Automaten $M' = \left(Q',\Sigma,\delta',[q_0]_\sim,F'\right)$. Dabei ist:

$$\begin{aligned} & Q' =_{Df} \left\{[q]_\sim \mid q \in Q\right\} \\ & F' =_{Df} \left\{[f]_\sim \mid f \in F\right\} \\ & \delta'\left([q]_\sim,a\right) =_{Df} \left[\delta(q,a)\right]_\sim \end{aligned} \tag{8.3}$$

Dieser Automat wird wie folgt konstruiert, als Zustände nehme man die Äquivalenzklassen der Zustände. Als akzeptierte Zustände gelten die Äquivalenzklassen der Tilderelation von F.

Damit ist $M' = \left(Q',\Sigma,\delta',[q_0]_\sim,F'\right)$ ein minimaler Automat mit $T(M) = T(M')$.

Aber ist dieser Automat auch wirklich minimal, oder nur ein reduzierter Automat? Gezeigt wird dies durch den folgenden Beweis:

δ' ist mit :

$$\forall x \in \Sigma^*: \delta'([q], x) = [\delta(q, x)] \tag{8.4}$$

wohldefiniert, aus folgendem Grund: Wenn q und q' in der selben Äquivalenzklasse liegen und die Relation definiert sind, dann ist auch $\delta'([q], a)$ mit $\delta'([q'], a)$ in Relation.

Mit der Definition von F' gilt dann:

$$\begin{aligned} x \in T(M) &\Leftrightarrow \delta(q_0, x) \in F \Leftrightarrow [\delta(q_0, x)] \in F' \\ &\Leftrightarrow \delta'([q_0], x) \in F' \\ &\Leftrightarrow x \in T(M') \end{aligned} \tag{8.5}$$

Somit akzeptieren also beide Automaten die selbe Sprache. Die Frage ist nun, ob dieser Automat auch minimal ist.

Zuerst definieren wir die ρ Relation als:

$$x \quad \rho \quad y \Leftrightarrow_{Df} \delta'([q_0], x) = \delta'([q_0], y) \tag{8.6}$$

Weiterhin wissen wir über diesen Automaten, dass der $Index(\rho) = |Q'|$, da jeder Zustand erreichbar ist. Somit wissen wir, dass die ρ Relation die Nerode-Automat Relation R_L verfeinert:

$$x \quad \rho \quad y \Rightarrow x \quad R_L \quad y \tag{8.7}$$

Gilt nun auch, die Umkehrung? Folgt aus $x \quad \rho \quad y \Leftarrow x \quad R_L \quad y$? Der Beweis erfolgt durch Kontraposition.

$$x \quad \not\rho \quad y \Rightarrow x \quad \not R_L \quad y \tag{8.8}$$

Die Klassen können nicht gleich sein, da es die Elemente nicht in der ~ Relation gibt. Es gibt daher ein Element z, so dass:

$$\begin{aligned} & x \quad \not\rho \quad y \Rightarrow [\delta(q_0, x)] = \delta'([q_0], x) \neq \delta'([q_0], y) = [\delta(q_0, y)] \\ & \Rightarrow \exists \quad z: (\delta(q_0, xz) \in F \wedge \delta(q_0, yz) \notin F) \vee (\delta(q_0, xz) \notin F \wedge \delta(q_0, yz) \in F) \\ & \Rightarrow x \quad \not R_L \quad y \end{aligned} \tag{8.9}$$

Die beiden Relationen sind also gleichwertig: $x \quad \rho \quad y \Leftrightarrow x \quad R_L \quad y$. Und die Minimalzahl der Zustände ist: $|Q'| = index(\rho) = index(R_L)$ Und damit ist gezeigt, dass der Automat M' minimal ist.

8.2 „Praktische“ Minimierung von deterministischen endlichen Automaten

Nachdem wir wissen, dass die Bestimmung der Äquivalenzklassen $q \sim q'$ definiert ist für: $\forall x \in \Sigma^*$: $\delta(q, x) \in F \Leftrightarrow \delta(q', x) \in F$. Eine Konstruktion, die den (bis auf Isomorphie) eindeutig bestimmten minimalen endlichen Automaten ($M' = (Q', \Sigma, \delta, q_0, F)$ mit $Q' = \{[q]_\sim \mid q \in Q\}$) liefert benötigt die folgende Eigenschaft:

$$q \overset{k}{\sim} q' \Leftrightarrow q \overset{k-1}{\sim} q' \wedge \left(\forall a \in \Sigma: \ \delta(q, a) \overset{k-1}{\sim} \delta(q', a) \right) \tag{8.10}$$

8.2.1 Der „Algorithmus“

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein endlicher Automat. Dann gelangt man durch die folgenden Schritte zu dem minimalen Automaten:

1. Man beseitige alle nicht-erreichbaren Zustände, d.h. alle Zustände q, für die gilt $q \notin \left\{ \delta(q_0, x) \middle| x \in \Sigma^* \right\}$. (Es stellt sich natürlich die Frage, in wie weit dieses Problem algorithmisch lösbar ist. Ein möglicher Algorithmus könnte alle durch Wörter der Länge n erreichbaren Zustände betrachten. Damit kommt man durch alle erreichbaren Zustände.)

2. $\overset{0}{\sim}$ hat maximal zwei Äquivalenzklassen. Zum einen die akzeptierenden Zustände und dann noch die nicht akzeptierenden Zustände: $\left\{ q \middle| q \in Q - F \right\}_{\overset{0}{\sim}}$, $\left\{ q \middle| q \in F \right\}_{\overset{0}{\sim}}$

3. Angenommen, man hat die Äquivalenzklassen der Relation $\overset{k}{\sim}$ für $(k \geq 0)$, dann bestimmt man die Klassen der Relation $\overset{k+1}{\sim}$ wie folgt:
   Zunächst wird nun die folgende Tabelle aufgestellt:

	$a_1 \; a_2 \; \ldots$	a_j	$\ldots \; a_{\lvert\Sigma\rvert}$
q_0			
q_1			
$\vdots$			
q_i		$\left[\delta(q_i, a_j) \right]_{\overset{k}{\sim}}$	
$\vdots$			
$q_{\lvert Q\rvert-1}$			

   Alle Felder (q_i, a_j) erhalten also als Eintrag $\left[\delta(q_i, a_j) \right]_{\overset{k}{\sim}}$. Nun formt man die Klassen von $\overset{k+1}{\sim}$, indem man alle Zustände, die bereits bei $\overset{k}{\sim}$ in einer Klasse waren *und* die in obiger Liste gleiche „Zeilen“ haben in jeweils eine Klasse der Relation $\overset{k+1}{\sim}$ schreibt.

4. Wenn nun $\overset{k}{\sim}$ äquivalent zu $\overset{k+1}{\sim}$ ist (was, wie schon gezeigt, spätestens bei $k = n-1$ eintreten muss), dann bricht das Verfahren ab. Wir bilden dann die ~ Relation durch Setzen, von $\sim \equiv \overset{k}{\sim}$.

5. Der minimale Automat $M' = \left(Q', \Sigma, \delta', [q_0]_\sim, F' \right)$ wird letztlich wie oben schon beschrieben konstruiert:

$$
\begin{aligned}
Q' &= \left\{ [q]_\sim \middle| q \in Q \right\} \\
F' &= \left\{ [q]_\sim \middle| q \in F \right\} \\
\delta'\left([q]_\sim, a \right) &= \left[\delta(q, a) \right]_\sim
\end{aligned}
\tag{8.11}
$$

8.2.2 Beispiel

Wir betrachten einen Automaten $M = (Q, \Sigma, \delta, q_0, F)$, welcher die Sprache $T(M) = \{ab, ba\}^*$ akzeptieren soll. Ein Automat, der dies löst ist in folgender Grafik aufgelistet. Dabei ist zu beachten, dass der Zustand 7 nicht erreicht werden kann.

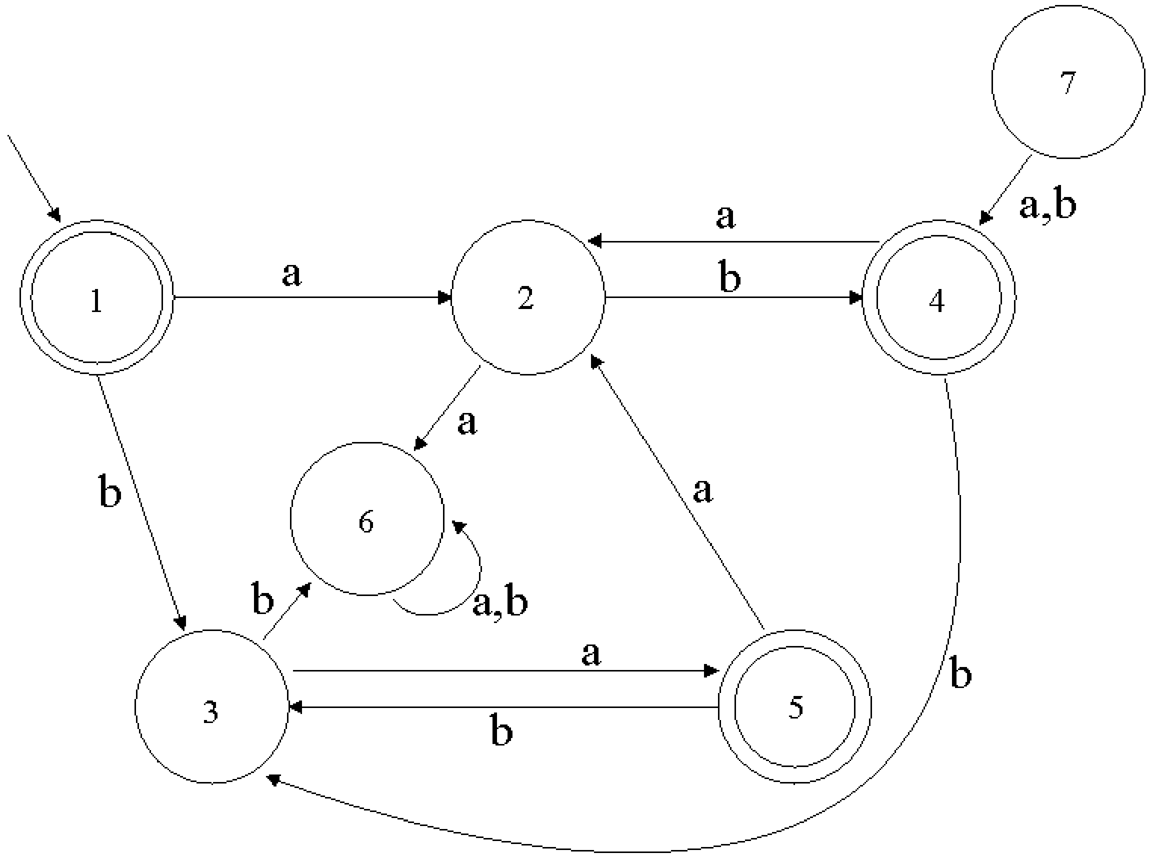


Abbildung 14: Beispiel eines nicht minimalen Automaten

Durch das Anwenden des oben beschriebenen Algorithmus gelangt man schnell zu folgender Tabelle:

$\overset{0}{\sim}$: $[2,3,6]$ **Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen.**

$\overset{1}{\sim}$:

	a	b
1.	$[2,3,6]_{\overset{0}{\sim}}$	$[2,3,6]_{\overset{0}{\sim}}$
2.	$[2,3,6]_{\overset{0}{\sim}}$	$[1,4,5]_{\overset{0}{\sim}}$
3.	$[1,4,5]_{\overset{0}{\sim}}$	$[2,3,6]_{\overset{0}{\sim}}$
4.	$[2,3,6]_{\overset{0}{\sim}}$	$[2,3,6]_{\overset{0}{\sim}}$
5.	$[2,3,6]_{\overset{0}{\sim}}$	$[2,3,6]_{\overset{0}{\sim}}$
6.	$[2,3,6]_{\overset{0}{\sim}}$	$[2,3,6]_{\overset{0}{\sim}}$

Folglich sind die Klassen von $\overset{1}{\sim}$:

$[1,4,5],[2],[3],[6]$

$\overset{2}{\sim}$:

	a	b
1.	$[2]_{\overset{1}{\sim}}$	$[3]_{\overset{1}{\sim}}$
2.	$[6]_{\overset{1}{\sim}}$	$[1,4,5]_{\overset{1}{\sim}}$
3.	$[1,4,5]_{\overset{1}{\sim}}$	$[6]_{\overset{1}{\sim}}$
4.	$[2]_{\overset{1}{\sim}}$	$[3]_{\overset{1}{\sim}}$
5.	$[2]_{\overset{1}{\sim}}$	$[3]_{\overset{1}{\sim}}$
6.	$[6]_{\overset{1}{\sim}}$	$[6]_{\overset{1}{\sim}}$

Folglich sind die Klassen von $\overset{2}{\sim}$:

$[1,4,5],[2],[3],[6]$

An dieser Stelle sind wir dann auch schon fertig, da wir sehen, dass die Klassen $\overset{1}{\sim}$ und $\overset{2}{\sim}$ schon äquivalent sind, somit ist die ~ Relation definiert durch:

$$\sim \equiv \overset{1}{\sim} \tag{8.12}$$

Es bleibt nur noch den endgültigen Automaten $M' = \left(Q', \Sigma, \delta', [q_0]_{\sim}, F'\right)$ aufzustellen. Dieser sieht dann, so aus, wie in der folgenden Abbildung gezeigt.

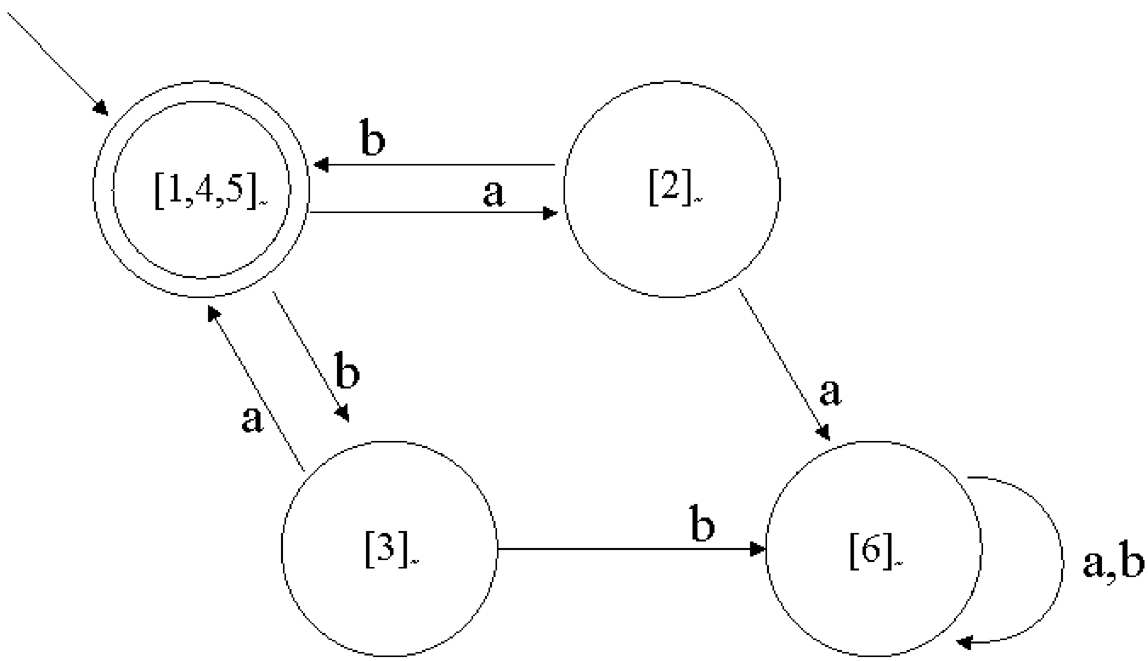


Abbildung 15 der minimierte Automat M'

9 Vorlesung vom 04. Mai 2000 Abschlußeigenschaften regulärer Mengen

9.1 Motivation

Viele Operationen auf regulären Mengen erhalten die Eigenschaft der Regularität. Dies bedeutet, daß die Operationen angewandt auf den regulären Mengen wieder zu einer regulären Menge führt.

Mit Hilfe dieser als *Abgeschlossenheit* bezeichneten Eigenschaften kann man also aus gegebenen regulären Mengen wieder neue Mengen schaffen, die von endlichen Automaten akzeptiert werden.

Man kann die Abschlußeigenschaften auch dazu verwenden, um zu zeigen, dass eine bestimmte gegebene Sprache regulär (bzw. nicht regulär) ist, indem man über die Abschlußeigenschaften auf bereits bekannte Sprachen zurückgeht.

9.2 Sammlung von Abschlußeigenschaften

Es gelten die folgenden Abschlußeigenschaften regulärer Mengen:

(9.1)	Vereinigung:	$L_1, L_2 \in REG$	$\Rightarrow L_1 \cup L_2 \in REG$
(9.2)	Konkatenation:	$L_1, L_2 \in REG$	$\Rightarrow L_1 \circ L_2 \in REG$ $L_1 \circ L_2 = \{xy \mid x \in L_1, y \in L_2\}$
(9.3)	Kleenesche Hülle:	$L \in REG$	$\Rightarrow L^* \in REG, L^* = \bigcup_{i=0}^{\infty} L^i$
(9.4)	Komplement:	$L \subseteq \Sigma^*, L \in REG$	$\Rightarrow \Sigma^* - L \in REG$
(9.5)	Schnitt:	$L_1, L_2 \in REG$	$\Rightarrow L_1 \cap L_2 \in REG$
(9.6)	Homomorphishmus:	$L_1 \subseteq \Sigma_1^*, L_1 \in REG, h: \Sigma_1^* \to \Sigma_2^*$	$\Rightarrow h(L_1) \in REG$
(9.7)	inv. Homomorphism.:	$L_1 \subseteq \Sigma_2^*, L_2 \in REG, h: \Sigma_1^* \to \Sigma_2^*$	$\Rightarrow h^{-1}(L_2) \in REG$ $h^{-1}(L_2) = \{x \mid x \in \Sigma_1^*, h(x) \in L_2\}$

9.2.1 Beweise Vereinigung, Konkatenation, Kleenesche Hülle

Zu (9.1), (9.2) und (9.3) findet man die entsprechenden Beweise im Beweis über die Äquivalenz von regulären Ausdrücken und einem NFA mit ε-Bewegungen.

9.2.2 Komplementbildung (9.4)

Zu Zeigen: $L \subseteq \Sigma^*, L \in REG \quad \Rightarrow \Sigma^* - L \in REG$

Beweis:

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DFA mit L=T(M). Dann definieren wir einen DFA M‘ wie folgt:

$$M' := (Q, \Sigma, \delta, q_0, Q - F) \qquad (9.8)$$

Es zeigt sich nun folgendes:

Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen. (9.9)

Also: $T(M') = \Sigma^* - T(M) = \Sigma^* - L$

9.2.3 Schnitt (9.5)

Zu zeigen: $L_1, L_2 \in REG \qquad \Rightarrow L_1 \cap L_2 \in REG$

Beweis:

Dies zeigt man sehr schnell über die Regeln von deMorgan. Es gilt nämlich $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ und mit der Abgeschlossenheit unter Komplement- und Vereinigungsbildung folgt automatisch die Abgeschlossenheit unter Schnittbildung.

9.2.4 Substitution

Definition:

Eine **Substitution f** ist eine Abbildung $f : \Sigma \to L, L \subseteq \Delta^*$ mit Δ ein Alphabet . F assoziiert also mit jedem Symbol aus Σ eine Sprache. Wir erweitern f auf Zeichenketten:

1. $f(\varepsilon) = \varepsilon$
2. $f(xa) = f(x) f(a)$

Beispiel:

Ist $f(0) = a$ und $f(1) = b^*$. Damit ist f(010) die reguläre Menge ab^*a .

Satz:

Die Klasse der regulären Mengen ist unter Substitution abgeschlossen.

Beweis:

Sei $R \subseteq \Sigma^*$ eine reguläre Menge und $\hat{\delta}(q,a)$. Sei weiter $f : \Sigma \to 2^{\Delta^*}$ die Substitution, die durch $f(a)=R_a$ definiert ist.

Wir wählen reguläre Ausdrücke zur Bezeichnung von R und allen R_a.

Man ersetze nun jedes Auftreten des Symbols **a** im regulären Ausdruck für R durch den regulären Ausdruck für R_a.

Um zu beweisen, dass der resultierende reguläre Ausdruck $f(R)$ beschreibt, wird verwendet, dass die Substitution einer Vereinigung, eines Produkts oder einer Hülle gleich der Vereinigung, dem Produkt bzw. der Hülle der Substitution ist (also homomorph).

Mit einer Induktion über die Anzahl der Operatoren im regulären Ausdruck vervollständigt man leicht den Beweis.

9.2.5 Homomorphismus (9.6)

Zu zeigen: $L_1 \subseteq \Sigma_1^*, L_1 \in REG, h : \Sigma_1^* \to \Sigma_2^* \qquad \Rightarrow h(L_1) \in REG$

Beweis:

Die Abgeschlossenheit unter Homomorphismen folgt sofort aus der Abgeschlossenheit unter Substitutionen, da ein Homomorphismus ein Spezialfall der Substitution ist, bei der h(a) für alle a gegeben wird.

9.2.6 Inverser Homomorphismus (9.7)

Zu zeigen: $L_1 \subseteq \Sigma_2^*, L_2 \in REG, h : \Sigma_1^* \to \Sigma_2^* \qquad \Rightarrow h^{-1}(L_2) \in REG$

Beweis:

Es seien $M_2 = (Q, \Sigma, \delta_2, q_0, F)$ ein DFA mit $T(M_2) = L_2$ und $h : \Sigma_1^* \to \Sigma_2^*$ ein Homomorphismus.

Wir konstruieren einen DFA M_1, der $h^{-1}(L)$ akzeptiert, indem er ein Symbol a aus Σ_1 liest und M_1 auf h(a) simuliert. Wir geben die Definition an:

$$\begin{aligned} &M_1 := (Q, \Sigma_1, \delta_1, q_0, F) \\ &\delta_1(q,a) := \delta_2(q, h(a)) \forall q \in Q, a \in \Sigma_1 \end{aligned} \tag{9.10}$$

Zu beachten ist, dass h(a) eine lange Zeichenkette oder aber auch das leere Wort sein kann. Dies ist aber kein Problem, da δ mittels Erweiterung auch auch Zeichenketten operiert.

Durch Induktion über die Wortlänge des Wortes x kann man leicht zeigen, dass gilt

$$\delta_1(q_0, x) = \delta_2(q_0, h(x)) \tag{9.11}$$

Daher wird x von M_1 genau dann akzeptiert, wenn h(x) von M_1 akzeptiert wird. Wir haben also einen DFA gebaut mit $T(M_1) = h^{-1}(T(M_2))$. Damit gilt auch $h^{-1}(L_2) \in REG$.

10 Vorlesung vom 9. Mai 2000

10.1 Beispiele zur Nutzung der Abschlusseigenschaften

10.1.1 Einleitung

Aus den vorhergehenden Vorlesungseinheiten sind die sogenannten „Abschlusseigenschaften“ bekannt:

$$\begin{aligned} &L_1, L_2 \in \mathrm{REG} \Rightarrow L_1 \circ L_2 \in \mathrm{REG} \quad && \circ \in \{\cap, \cup, \cdot\} \\ &L_1 \in \mathrm{REG} \Rightarrow \circ(L_1) \in \mathrm{REG} && \circ \in \{-, h, h^{-1}, *\} \end{aligned} \tag{10.1}$$

Weiterhin nehmen wir zur Kenntnis[4]:

$$L = \{a^n b^n \mid n \geq 0\} \notin \mathrm{REG} \tag{10.2}$$

In den folgenden Beispielen wird durch die Kombination des Wissens von (10.2) mit den Aussagen der Abschlusseigenschaften der Beweis für einen neuen Zusammenhang erbracht.

Behauptung:

Die folgende Aussage ist falsch: „Wenn $a^n b^n$ nicht regulär ist, kann auch eine „größere“ Sprache wie $a^* b^*$ nicht regulär sein.“ Tatsächlich gilt, dass $a^n b^n$ nicht regulär ist, und $a^* b^*$ regulär ist.

10.1.2 Beispiel 1

Zu zeigen sei:

$$L = \{a^n b^r c^n d^s \mid n \geq 0, r \geq 0, s \geq 0\} \notin \mathrm{REG} \tag{10.3}$$

Wir führen einen Widerspruchsbeweis durch, und nehmen an:

$$L \in \mathrm{REG} \tag{10.4}$$

Wir definieren eine homomorphe Funktion *h()* wie folgt:

$$h(a) := a \quad h(b) := \varepsilon \quad h(c) := b \quad h(d) := \varepsilon \tag{10.5}$$

(10.3) lässt sich mit Hilfe von (10.5) umschreiben zu:

$$h(L) = \{a^n b^n \mid n \geq 0\} \tag{10.6}$$

Gemäß (10.1) gilt:

$$L \in \mathrm{REG} \Rightarrow h(L) \in \mathrm{REG} \tag{10.7}$$

Aus (10.6) und (10.7) folgt:

$$h(L) = \{a^n b^n \mid n \geq 0\} \in \mathrm{REG} \qquad \text{Widerspruch!} \tag{10.8}$$

10.1.3 Beispiel 2

Zu zeigen sei:

$$L = \{w \mid w \in \{a,b\}^* \wedge \#_a(w) = \#_b(w)\} \notin \mathrm{REG} \tag{10.9}$$

Wir führen einen Widerspruchsbeweis durch, und nehmen an:

$$L \in \mathrm{REG} \tag{10.10}$$

[4] (10.2) ist zwar beweisbar, an dieser Stelle verzichten wir aber darauf.

Wir definieren eine zweite Sprache L_1 wie folgt:

$$L_1 := \underbrace{L}_{\substack{\text{angeblich}\\\text{reguläre}\\\text{Sprache}}} \cap \underbrace{a^*b^*}_{\substack{\text{reguläre}\\\text{Sprache}}} \tag{10.11}$$

Aus (10.1) folgt, dass die Schnittmenge zweier regulärer Mengen erneut regulär ist, daher folgt:

$$L_1 \in \text{REG} \tag{10.12}$$

(10.11) lässt sich umschreiben zu:

$$L_1 := L \cap a^*b^* = \left\{a^n b^n \,\middle|\, n \geq 0\right\} \tag{10.13}$$

Aus (10.13) und (10.10) folgt:

$$L_1 := L \cap a^*b^* = \left\{a^n b^n \,\middle|\, n \geq 0\right\} \in \text{REG} \qquad \text{Widerspruch!} \tag{10.14}$$

10.1.4 Beispiel 3

Zu zeigen sei:

$$L = \left\{a^n b^{k-n} \,\middle|\, n \geq 0\right\} \notin \text{REG} \tag{10.15}$$

Wir führen einen Widerspruchsbeweis durch, und nehmen an:

$$L \in \text{REG} \tag{10.16}$$

Wir definieren eine homomorphe Funktion *h()* wie folgt:

$$h(a) := a \quad h(b) := b^n \tag{10.17}$$

(10.15) lässt sich mit Hilfe von (10.17) umschreiben zu:

$$h^{-1}(L) = \left\{a^n b^n \,\middle|\, n \geq 0\right\} \tag{10.18}$$

Gemäß (10.1) gilt:

$$L \in \text{REG} \Rightarrow h^{-1}(L) \in \text{REG} \tag{10.19}$$

Aus (10.18) und (10.19) folgt:

$$h^{-1}(L) = \left\{a^n b^n \,\middle|\, n \geq 0\right\} \in \text{REG} \qquad \text{Widerspruch!} \tag{10.20}$$

10.1.5 Beispiel 4

Zu zeigen sei:

$$L = \left\{a^n b^m \,\middle|\, 0 \leq m \leq n\right\} \notin \text{REG} \tag{10.21}$$

Wir führen einen Widerspruchsbeweis durch, und nehmen an:

$$L \in \text{REG} \tag{10.22}$$

Wir definieren eine zweite Sprache L_1 wie folgt:

$$L_1 := \underbrace{\overline{L}}_{\substack{\text{angeblich}\\\text{reguläre}\\\text{Sprache}}} \cap \underbrace{a^*b^*}_{\substack{\text{reguläre}\\\text{Sprache}}} = \left\{a^n b^m \,\middle|\, 0 \leq n < m\right\} \tag{10.23}$$

Aus (10.1) folgt, dass das Komplement einer regulären Menge, und auch die Schnittmenge zweier regulärer Mengen erneut regulär ist, daher folgt:

$$L_1 \in \text{REG} \tag{10.24}$$

Wir definieren eine homomorphe Funktion *h()* in eine weitere, neue Sprache L_2 wie folgt:

$$\begin{aligned} &h(a) := a \quad h(b) := b \quad h(c) := c \\ &L_2 := h^{-1}(L_1) = \{a^n x \| x| = m = \#_b(x) + \#_c(x)\} \quad 0 \le n < m \end{aligned} \tag{10.25}$$

Aus (10.1) folgt, dass die umgekehrte Anwendung einer homomorphen Abbildung auf eine reguläre Menge erneut regulär ist:

$$L_2 \in \text{REG} \tag{10.26}$$

Wir definieren nun eine dritte Sprache:

$$L_3 := \underbrace{L_2}_{\substack{\text{reguläre}\\\text{Sprache}}} \cap \underbrace{a^* b^* c}_{\substack{\text{reguläre}\\\text{Sprache}}} = \{a^n b^m c \mid 0 \le n < m+1\} \tag{10.27}$$

Aus (10.1) folgt, dass die Schnittmenge zweier regulärer Menge erneut regulär ist:

$$L_3 \in \text{REG} \tag{10.28}$$

Wir definieren eine zweite, homomorphe Funktion *h()* in eine weitere, neue Sprache L_4 wie folgt:

$$\begin{aligned} &h(a) := a \quad h(b) := b \quad h(c) := \varepsilon \\ &L_4 := h(L_3) = \{a^n b^m \mid 0 \le n \le m\} \end{aligned} \tag{10.29}$$

Wir definieren nun eine fünfte Sprache:

$$L_5 := \underbrace{L}_{\substack{\text{angeblich}\\\text{reguläre}\\\text{Sprache}}} \cap \underbrace{L_4}_{\substack{\text{reguläre}\\\text{Sprache}}} = \{a^n b^n \mid n \ge 0\} \tag{10.30}$$

Aus (10.1) folgt, dass die Schnittmenge zweier regulärer Menge erneut regulär ist:

$$L_5 \in \text{REG} \qquad \text{Widerspruch!} \tag{10.31}$$

10.1.6 Beispiel 5

Wir verwenden dieses Mal ein erweitertes Alphabet Σ:

$$\Sigma := \{a, b_1, \ldots, b_k\} \qquad k \ge 1 \tag{10.32}$$

Zu zeigen sei:

$$L = \{a_n b_1^{i_1} b_2^{i_2} b_3^{i_3} \ldots b_n^{i_n} \mid 0 \le i_1 + i_2 + i_3 + \ldots + i_n \le n\} \notin \text{REG} \tag{10.33}$$

Wir führen einen Widerspruchsbeweis durch, und nehmen an:

$$L \in \text{REG} \tag{10.34}$$

Wir definieren eine homomorphe Funktion *h()* wie folgt:

$$\begin{aligned} &h(a) := a \quad h(b_j) := b \qquad 1 \le j \le n \\ &\Downarrow \\ &h(L) = \{a^n b^m \mid 0 \le m \le n\} \end{aligned} \tag{10.35}$$

Gemäß (10.1) gilt:

$$L \in \text{REG} \Rightarrow h(L) \in \text{REG} \tag{10.36}$$

Aus (10.35) und (10.36) folgt:

$$h(L) = \{a^n b^m \mid 0 \le m \le n\} \in \text{REG} \qquad \text{Widerspruch!} \tag{10.37}$$

Der Widerspruch entsteht durch die Aussage von (10.21).

10.2 Automaten mit ε-Bewegungen

10.2.1 Einleitung

Bisher sind der deterministische endliche Automat (*DFA*) und der nichtdeterministische endliche Automat (*NFA*) betrachtet worden, wobei der *DFA* lediglich ein Spezialfall des *NFA* war. Diese Automaten haben sich dadurch ausgezeichnet, dass jede Transition (jeder Zustandswechsel) durch ein Eingabesymbol $x \in \Sigma$ ausgelöst wurde.

Bei den im Folgenden betrachteten „Automaten mit ε-Bewegungen" gibt es Transitionen, die durch das Eingabesymbol „ε", d.h. das „leere Wort" ausgelöst werden.

Das Transitionsdiagramm eines derartigen Automaten kann z.B. wie folgt aussehen:

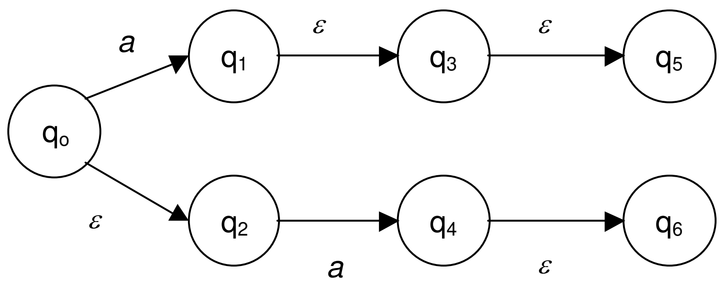


Abbildung 16 – ein Transitionsdiagramm

Bei dem obigen Automaten führt aus dem Startzustand q_0 für das Eingabezeichen a lediglich ein direkter Pfad zu dem Zustand q_1. Allerdings führen zahlreiche weitere Pfade für das Eingabesymbol ε zu den restlichen Zuständen.

Das Eingabesymbol ε unterscheidet sich von allen anderen Eingabesymbolen, und zwar insofern, als dass es der Automat „jederzeit" einlesen kann – unabhängig von den wirklichen Eingabesymbolen.

Der obige Automat kann also für das Eingabezeichen a in jeden, eingezeichneten Zustand wechseln. Um dieses Verhalten beschreibungstechnisch in den Griff zu kriegen, definiert man den Begriff der ε-Hülle (siehe folgender Abschnitt).

10.2.2 ε-Hülle

Man bezeichnet die Menge aller Konten *p*, zu denen es einen mit ε bezeichneten Weg vom Knoten *q* gibt, als ε-Hülle(q)[5].

Für den Automaten aus Abbildung 16 gilt, dass alle Knoten vom Startzustand aus erreichbar sind, also:

$$\varepsilon\text{-Hülle}(q_0) = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\} \tag{10.38}$$

Definitionsgemäß ist jeder Knoten/Zustand in seiner eigenen ε-Hülle enthalten. Man kann sich das so vorstellen, als dass von jedem Knoten ein mit ε markierter Übergang auf sich selber existiert, den man beim Zeichnen weglässt:

$$k = 1 \tag{10.39}$$

[5] Im englischen: „ε-Closure"

Auf das obige Beispiel angewendet ergibt sich die ε-Hülle wie folgt:

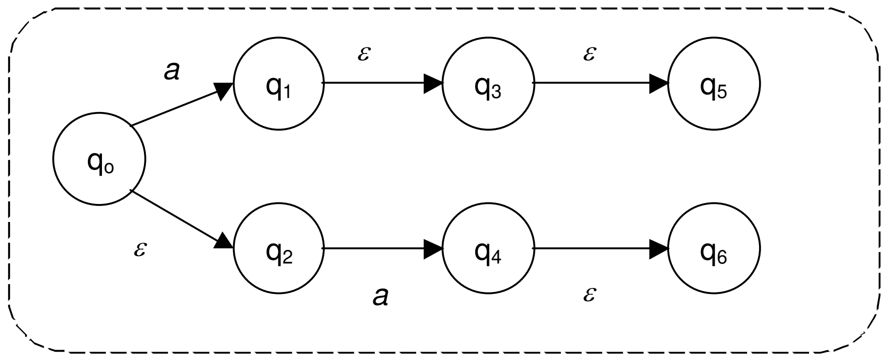


Abbildung 17 – die Epsilon-Hülle im Transitionsdiagramm

10.2.3 Formale Unterschiede zum gewöhnlichen NFA

Die Definition des Automaten mit ε-Bewegungen entspricht der des „normalen" NFA:

$$\varepsilon\text{-NFA} := (Q, \Sigma, \delta, q_0, F) \tag{10.40}$$

Der einzige Unterschied zum „normalen" NFA ist die Übergangsfunktion δ, die nun wie folgt definiert ist:

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q \tag{10.41}$$

Die Erweiterung der Übergangsfunktion δ (die nur für einzelne Symbole taugt) auf ganze Eingabewörter geschieht mit Hilfe der Übergangsfunktion $\hat{\delta}$. $\hat{\delta}(q,w)$ soll als Ergebnis die Menge aller Zustände liefern, zu denen es von q aus einen mit w markierten Pfad gibt, welcher ε-Kanten beinhalten darf:

$$\hat{\delta} : Q \times \Sigma^* \to 2^Q \tag{10.42}$$

$$\hat{\delta}(q,\varepsilon) := \varepsilon\text{-Hülle}(q) \tag{10.43}$$

$$\forall w \in \Sigma^*, a \in \Sigma : \quad \hat{\delta}(q,wa) := \varepsilon\text{-Hülle}\Big(\delta\big(\hat{\delta}(q,w),a\big)\Big) \tag{10.44}$$

$$\delta(R,a) = \bigcup_{q \in R} \delta(q,a) \tag{10.45}$$

$$\hat{\delta}(R,w) = \bigcup_{q \in R} \hat{\delta}(q,w) \tag{10.46}$$

Anmerkung: Beim Automaten mit ε-Bewegungen können die Ergebnisse von $\hat{\delta}(q,a)$ und $\delta(q,a)$ verschieden sein; zwischen $\hat{\delta}(\ )$ und $\delta(\ )$ muss also unterschieden werden.

10.2.4 Äquivalenz vom NFA mit und ohne ε-Bewegungen

Auch die ε-Bewegungen erlauben es dem Automaten nicht, mehr als reguläre Mengen zu akzeptieren. Der Automaten mit ε-Bewegungen reiht sich damit in die Klasse von NFA und DFA ein.

Wir werden uns darauf beschränken, zu zeigen, dass jeder NFA mit ε-Bewegungen durch einen „normalen NFA“ ersetzt werden kann. Der Beweis in der umgekehrten Richtung ist zwar theoretisch ebenfalls nötig, erübrigt sich aber bei näherer Betrachtung[6].

Zu einem gegebenen ε-NFA $M = (Q, \Sigma, \delta, q_0, F)$ definieren wir einen „normalen“ NFA $M' = (Q, \Sigma, \delta', q_0, F')$. Wie man an der Definition sehen kann, muss an dem Automaten nicht viel verändert werden. Die Menge der akzeptierenden Zustände *F* kann weitestgehend übernommen werden, abgesehen von der Ausnahme, dass die ε-Hülle von q_0 einen akzeptierenden Zustand enthält:

$$F' := \begin{cases} F \cup \{q_0\} & \text{falls die } \varepsilon\text{-Hülle}(q_0) \text{ einen Zustand aus F enthält} \\ F & \text{sonst} \end{cases} \tag{10.47}$$

Diese Veränderung ist nötig, weil *M’* keine ε-Transitionen kennt. Tritt nämlich der Fall ein, dass die ε-Hülle von q_0 einen akzeptierenden Zustand enthält, dann würde *M* ja beenden, ohne ein einziges Zeichen gelesen zu haben. Für *M’* lässt sich dies nur nachbilden, indem der Startzustand (q_0) selber akzeptierend ist. Und genau das erreichen wir mit der obigen Fallunterscheidung.

Nun wird noch eine Übergangsfunktion δ' benötigt. Dazu behalten wir die bereits definierte Funktion $\hat{\delta}$ bei:

$$\delta' := \hat{\delta} \tag{10.48}$$

Wie kann das sein? Fangen wir ganz vorne an. Wir möchten zwei äquivalente Automaten erhalten. Wir haben bereits definiert, dass beide Automaten auf der selben Menge von Zuständen operieren. Was ist nun also naheliegender, als auch die Übergangsfunktion komplett zu übernehmen? Genau dies tun wir laut (10.48). Der Sicherheit halber werden wir aber im folgenden beweisen, dass es auch richtig ist, was wir da tun... Wir betrachten zunächst einmal den einfachsten Fall, nämlich, dass die Eingabe ein einzelnes Symbol ist: Dann folgt aus (10.44), (10.43) und (10.45):

$$\begin{aligned}
&\forall w \in \Sigma^*, a \in \Sigma: \quad \hat{\delta}(q, wa) = \varepsilon\text{-Hülle}\left(\delta\left(\hat{\delta}(q, w), a\right)\right) \\
&\Downarrow \qquad w := \varepsilon a \\
&\hat{\delta}(q, \varepsilon a) = \varepsilon\text{-Hülle}\left(\delta\left(\hat{\delta}(q, \varepsilon), a\right)\right) \\
&\Downarrow \qquad \hat{\delta}(q, \varepsilon) := \varepsilon\text{-Hülle}(q) \\
&\hat{\delta}(q, \varepsilon a) = \varepsilon\text{-Hülle}\left(\delta\left(\varepsilon\text{-Hülle}(q), a\right)\right) \\
&\Downarrow \qquad \delta(R, a) = \bigcup_{q \in R} \delta(q, a) \\
&\hat{\delta}(q, \varepsilon a) = \varepsilon\text{-Hülle}\left(\bigcup_{p \in \varepsilon\text{-Hülle}(q)} \delta(p, a)\right)
\end{aligned} \tag{10.49}$$

In klarer Sprache bedeutet das, dass $\hat{\delta}$ angewendet auf ein einzelnes Symbol *a* (und einen Zustand *q*) nichts anderes ist, als die Vereinigung der Ergebnisse von δ für jeden einzelnen Zustand aus der ε-Hülle von *q*. Die ε-Hülle von *q* wiederum ist nichts anderes als eine Menge von Zuständen, zwischen denen der Automat indifferent ist. Genau dieses Verhalten ist auch die Forderung, die wir an die Übergangsfunktion δ' des NFA stellen. Wir haben jetzt für den Sonderfall $|x| = 1$ bewiesen, dass die Forderung, die wir an die „neue“ Funktion δ' stellen, durch die „alte“ Funktion $\hat{\delta}$ erfüllt werden.

Die Korrektheit für den Sonderfall $|x| = 0$, d.h. $x = \varepsilon$ haben wir bereits bei der Definition der akzeptierenden Zustände unter (10.47) verbal erklärt.

---

[6] Ein NFA wird ja schon dadurch zum ε-NFA, dass man von jedem Zustand einen mit „ε“ markierten Pfeil auf sich selber einzeichnet. Dadurch verändert sich das Verhalten des Automaten nicht.

Für $|x| > 1$ erbringen wir den Beweis induktiv. Da $|x| > 1$, lässt sich die Eingabe zerlegen in $x = wa \quad w \in \Sigma^*, a \in \Sigma$. Durch die Induktionsvoraussetzung gilt

$$\delta'(q_0, w) = \hat{\delta}(q_0, w) \tag{10.50}$$

Weiterhin gilt:

$$\delta'(q_0, wa) = \delta'(\delta'(q_0, w), a) \tag{10.51}$$

Aus (10.50) und (10.51) folgt:

$$\delta'(q_0, wa) = \delta'\left(\hat{\delta}(q_0, w), a\right) \tag{10.52}$$

Aus (10.46) folgt unter Zuhilfenahme der Induktionsannahme[7]:

$$\begin{aligned} &\delta'(R, a) = \bigcup_{q \in R} \delta'(q, a) \\ &\Downarrow \quad \delta'(q, a) = \hat{\delta}(q, a) \\ &\delta'(R, a) = \bigcup_{q \in R} \hat{\delta}(q, a) \end{aligned} \tag{10.53}$$

Mit dem Wissen von (10.53) lässt sich (10.52) wie folgt umschreiben:

$$\begin{aligned} &\delta'(q_0, wa) = \delta'\left(\hat{\delta}(q_0, w), a\right) \\ &\Downarrow \quad \delta'(R, a) = \bigcup_{q \in R} \hat{\delta}(q, a) \\ &\delta'(q_0, wa) = \bigcup_{p \in \hat{\delta}(q_0, w)} \hat{\delta}(p, a) \end{aligned} \tag{10.54}$$

Nach der Definition von $\hat{\delta}$ (siehe (10.44) gilt:

$$\bigcup_{p \in R} \hat{\delta}(p, a) = \hat{\delta}(R, a) \tag{10.55}$$

Damit folgt aus (10.54), (10.55) und (10.51):

$$\begin{aligned} &\delta'(q_0, wa) = \bigcup_{p \in \hat{\delta}(q_0, w)} \hat{\delta}(p, a) = \hat{\delta}\left(\hat{\delta}(q_0, w), a\right) = \hat{\delta}(q_0, wa) \\ &\Downarrow \\ &\delta'(q_0, wa) = \hat{\delta}(q_0, wa) \end{aligned} \tag{10.56}$$

[7] Hier wird die Induktionsannahme für Länge(x)=1 verwendet. Da in dem aktuellen Induktionsschritt Wörter der Länge > 1 betrachtet werden, ist dieses Vorgehen erlaubt.

11 Vorlesung vom 11. Mai 2000

11.1 Äquivalenz von endlichen Automaten und regulären Ausdrücken

11.1.1 Von regulären Ausdrücken zu endlichen Automaten

In den vorangegangen Wochen haben wir verschiedene Typen von endlichen Automaten kennengelernt. Dabei haben wir gezeigt, dass deterministische und nichtdeterministische endliche Automaten (DFA und NFA) sowie nichtdeterministische Automaten mit ε-Übergängen die gleiche Klasse von Sprachen, die regulären Sprachen, beschreiben und beliebig ineinander konvertiert werden können.

Jetzt wollen wir zeigen, dass die Sprachen, die durch reguläre Ausdrücke definiert werden, ebenfalls äquivalent zu diesen endlichen Automaten sind. Diese Sprachen werden auch reguläre Mengen genannt (nach dem *Satz von Kleene*: „Die Menge der durch reguläre Ausdrücke beschreibbaren Sprachen ist genau die Menge der regulären Sprachen.“)

Folgendes Schaubild verdeutlicht den Zusammenhang und die Abhängigkeit zwischen regulären Ausdrücken und endlichen Automaten, die wir im Anschluß zeigen wollen.

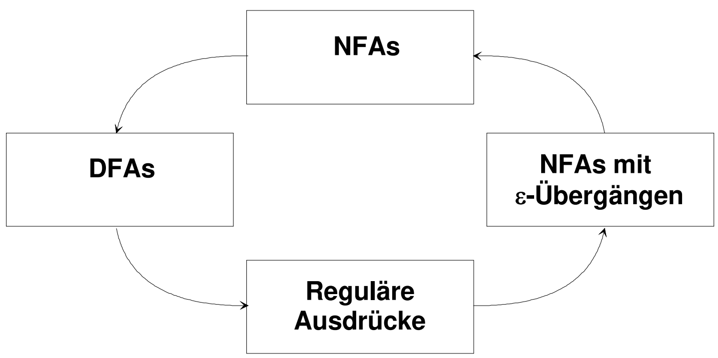


Abbildung 18 - Äquivalenzkreislauf reguläre Ausdrücke - endliche Automaten

Satz:

Sei r ein regulärer Ausdruck, dann existiert ein NFA mit ε-Übergängen, der die zum regulären Ausdruck gehörende Sprache L(r) akzeptiert.

Beweis

Für einfache reguläre Ausdrücke läßt sich die Behauptung einfach zeigen. Wenn r die leere Menge $r = \emptyset$, das leere Wort $r = \varepsilon$ oder ein einzelnes Zeichen $r = x \in \Sigma$ ist, dann liegt ein einfacher regulärer Ausdruck vor.

Zu zeigen ist, dass ein solcher regulärer Ausdruck in einen endlichen Automaten mit ε-Übergängen überführt werden kann, wir konstruieren also einen solchen Automaten gemäß der Vorgabe des regulären Ausdrucks. Diese drei ein- oder keinelementigen Ausdrücke ohne Operator stellen gewissermaßen den Induktionsanfang der Induktion über die Anzahl der Operatoren des regulären Ausdrucks vor.

Wenn wir zeigen können, dass ein beliebiger regulärer Ausdruck mit beliebig vielen Operatoren n in einen NFA mit ε-Übergängen und einem einzigen Endzustand, aus dem keine Übergänge herausführen, überführt werden kann, dann existiert für jeden regulären Ausdruck ein entsprechender Automat und die Äquivalenz ist bewiesen.

Für die drei elementaren Zustände können folgende Automaten gebaut werden, die einen einzigen Endzustand haben, der erreicht wird, wenn ein Wort der Sprache L(r) erkannt wird und von dem keine Übergänge ausgehen.

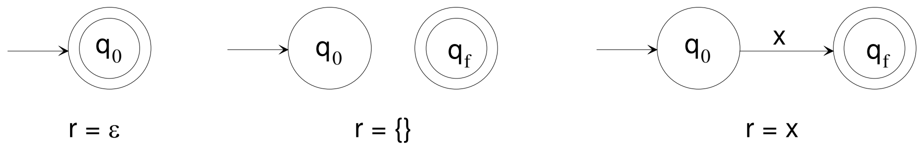


Abbildung 19 - NFAs für reguläre Ausdrücke ohne Operatoren

Für einen regulären Ausdruck, der das leere Wort erkennt, geht der Automat direkt in den Endzustand über, da ε-Übergänge erlaubt sind. Der reguläre Ausdruck leere Menge tritt niemals in den Endzustand ein und der reguläre Ausdruck für ein bestimmtes Zeichen geht genau dann in den Endzustand über, wenn dieses Zeichen erkannt wird.

Für den Induktionsschritt von i - 1 auf i Operatoren lassen sich NFAs mit ε-Übergängen für reguläre Ausdrücke mit weniger als i Operatoren bereits konstruieren (i ≥ 1). Dabei muß zwischen drei Verknüpfungsarten unterschieden werden

1. Vereinigung:

   $r = r_1 + r_2$

2. Konkatenation:

   $r = r_1 \cdot r_2$

3. Kleensche Hülle:

   $r = r_1*$

11.1.2 Vereinigung zweier regulärer Ausdrücke als NFAs

Gegeben sind zwei reguläre Ausdrücke r_1 und r_2 mit jeweils weniger als i Operatoren, die regulären Sprachen $L(r_1)$ und $L(r_2)$ beschreiben. Nach Induktionsannahme gibt es zwei entsprechende NFAs $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$ und $M_1 = (Q_2, \Sigma_2, \delta_2, q_2, \{f_2\})$, die diese Sprachen erkennen. Die beiden Automaten haben voneinander disjunkte Zustände, die unterschiedlich zu benennen sind, um sie in einer gemeinsamen Zustandmenge vereinigen zu können. Damit bilden wir einen neuen Automaten als Vereinigung von M_1 und M_2.

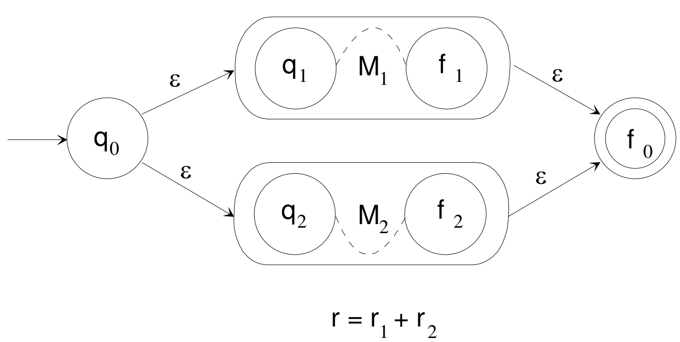


Abbildung 20 - Vereinigung von regulären Ausdrücken

Zu den Zuständen der beiden Automaten kommt ein gemeinsamer Startzustand q_0 und ein gemeinsamer Endzustand f_0, aus dem keine weiteren Übergänge herausführen. Die erzeugte reguläre Sprache L(r) soll entweder Wörter akzeptieren, die in $L(r_1)$ oder in $L(r_2)$ enthalten. Daher geht der konstruierte Automat beim Start von q_0 in einem ε-Übergang entweder in M_1 oder in M_2. Gemäß Induktionsannahme erkennt er dort ein entsprechendes Wort des jeweiligen Automaten und verläßt den Teilautomaten nicht, bevor er in den Endzustand f_1 bzw. f_2 eingetreten ist. In einem weiteren ε-Übergang geht der Automat vom Endzustand eines Teilautomaten in den gemeinsamen neuen Endzustand f_0.

Die Vereinigung zweier regulärer Ausdrücke, die sich in NFAs überführen lassen, liefert also wieder einen regulären Ausdruck, für den ein NFA existiert. Formal läßt sich der Automat, der L(r) erkennt als M ausdrücken:

$$M = \left(Q_1 \cup Q_2 \cup \{q_0, f_0\}, \Sigma_1 \cup \Sigma_2, \delta, q_0, \{f_0\}\right) \tag{11.1}$$

Die Übergangsfunktion δ ist folgendermaßen gestaltet:

$$\begin{aligned} &\delta(q_0,\varepsilon) = \{q_2, q_2\} \\ &\delta(\{f_1, f_2\},\varepsilon) = \{f_0\} \end{aligned} \tag{11.2}$$

Die Übergänge in den beiden Teilautomaten M_1 und M_2 entsprechen den Übergangsfunktionen δ_1 und δ_2. Dabei werden alle Zustände außer den beiden Endzuständen f_1 und f_2 betrachtet. Da f_1 und f_2 Endzustände von M_1 und M_2 sind, aus denen per Induktionsannahme keine Übergänge herausführen, können sie nicht in der folgenden Definition enthalten sein, sondern gehen gemäß obigem Abschnitt mit einem ε-Übergang zum gemeinsamen Endzustand f_0 über.

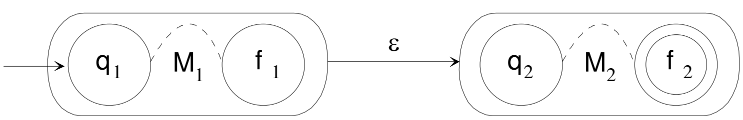
$$\begin{aligned} &\delta(q,a) = \delta_1(q,a) \quad \text{für } q \in Q_1 \setminus \{f_1\} \text{ und } a \in \Sigma_1 \cup \varepsilon \\ &\delta(q,a) = \delta_2(q,a) \quad \text{für } q \in Q_2 \setminus \{f_2\} \text{ und } a \in \Sigma_2 \cup \varepsilon \end{aligned} \tag{11.3}$$

Somit gilt der Induktionsschritt für Vereinigung:

$$L(M) = L(M_1) \cup L(M_2) \Leftrightarrow L(r) = L(r_1) \cup L(r_2) \tag{11.4}$$

11.1.3 Schnitt zweier regulärer Ausdrücke als NFAs

Die Konkatenation zweier regulärer Ausdrücke r_1 und r_2 zu einem neuen regulären Ausdruck r gestaltet sich ähnlich. Seien M_1 und M_2 NFAs, die $L(r_1)$ und $L(r_2)$ akzeptieren, dann sei M ein Automat, der die Sprachen beider Automaten konkateniert.



$r = r_1 \cdot r_2$

Abbildung 21 - Konkatenation von regulären Ausdrücken

Der Startzustand von M_1 ist gleichzeitig Startzustand von M. Der Automat durchläuft in M_1 die einzelnen Zustände, bis er ein Wort gemäß r_1 erkennt und in f_1 übergeht. In einem ε-Übergang geht der Automat von f_1 in q_2, den Startzustand von M_2, über. Hier tritt er über mögliche Zwischenzustände in f_2, den Endzustand von M_2, der auch gleichzeit Endzustand von M ist.

M akzeptiert also eine reguläre Sprache L(M), die gemäß Abschlußeigenschaften regulärer Sprachen aus der Konkatenation der per Induktionsannahme regulären Sprachen $L(M_1)$ und $L(M_2)$ entsteht. Für r existiert so ein NFA mit ε-Übergängen. Formal definiert sich M als:

$$M = \left(Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, \delta, q_1, \{f_2\}\right) \tag{11.5}$$

Die Übergangsfunktion δ für f_1 ist folgendermaßen gestaltet. Da f_1 Endzustand von M_1, führen keine Übergänge aus f_1 heraus, eine Verbindung zwischen M_1 und M_2 wird daher benötigt.

$$\delta(f_1,\varepsilon) = \{q_2\} \tag{11.6}$$

In den beiden Teilautomaten wird δ entsprechend mit δ_1 und δ_2 definiert. Der Übergang von f_1 wurde bereits definiert, f_2 verhält sich wie in M_2 und ist gleichzeitig Endzustand von M.

$$\begin{aligned} &\delta(q,a) = \delta_1(q,a) \quad \text{für } q \in Q_1 \setminus \{f_1\} \text{ und } a \in \Sigma_1 \cup \varepsilon \\ &\delta(q,a) = \delta_2(q,a) \quad \text{für } q \in Q_2 \text{ und } a \in \Sigma_2 \cup \varepsilon \end{aligned} \tag{11.7}$$

Somit ist der Induktionsschritt für Konkatenation gezeigt:

$$L(M) = L(M_1) \cdot L(M_2) \quad \Leftrightarrow \quad L(r) = L(r_1) \cdot L(r_2) \tag{11.8}$$

11.1.4 Kleensche Hülle eines regulären Ausdruckes als NFA

Bei der Hüllenbildung (auch Stern genannt) über den regulären Ausdruck r_1 kann r_1 einmal, keinmal oder beliebig oft auftreten. Dementsprechend führen wir einen Automaten M ein, der den zu r_1 passenden Automaten $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$ erweitert.

Tritt r_1 keinmal auf, so geht M vom neuen Startzustand q_0 in einem ε-Übergang direkt in den neuen Endzustand f_0 über. Andernfalls ist r_1 mindestens einmal vorhanden und M geht in den Bereich von M_1. Hier kann der reguläre Ausdruck beliebig oft wiederholt werden, da von f_1 mit einem ε-Übergang in q_1 verzweigt werden kann. Folgen keine weiteren Wiederholungen von r_1, geht M in einem weiteren ε-Übergang in den Endzustand f_0.

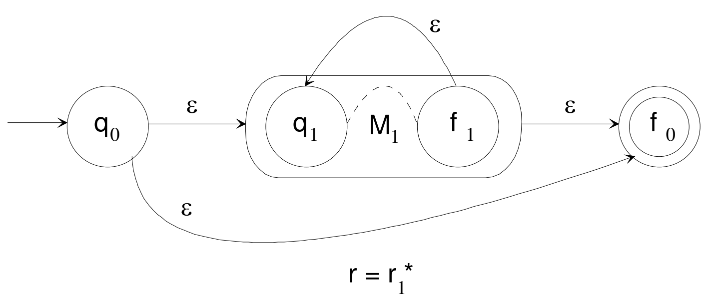


Abbildung 22 - Kleensche Hülle über regulärem Ausdruck

M akzeptiert somit reguläre Sprachen, die L(M) = L(M_1)* genügen und kann folgendermaßen definiert werden

$$M = (Q_1 \cup \{q_0, f_0\}, \Sigma_1, \delta, q_0, \{f_0\}) \tag{11.9}$$

Die zugehörige Übergangsfunktion δ entspricht im Wesentlichen δ_1, muß allerdings für die Sterneigenschaften erweitert werden

$$\begin{aligned} &\delta(q_0, \varepsilon) = \{q_1, f_0\} \\ &\delta(q, a) = \delta_1(q, a) \quad \text{für } q \in Q_1 \setminus \{f_1\} \text{ und } a \in \Sigma_1 \cup \varepsilon \\ &\delta(f_1, a) = \{q_1, f_0\} \end{aligned} \tag{11.10}$$

Somit ist der Induktionsschritt für die Kleensche Hülle (oder Stern) gezeigt. Es existiert ein NFA M, der beliebig viele Konkatenationen der von r_1 akzeptierten Sprache sowie das leere Wort akzeptiert.

$$L(M) = L(M_1)^* \quad \Leftrightarrow \quad L(r) = L(r_1)^* \tag{11.11}$$

Mit den drei Fällen Vereinigung, Konkatenation und Hüllenbildung ist die Induktion über die Anzahl der Operatoren abgeschlossen. Verschachtelte, vollständig geklammerte Ausdrücke können mit Hilfe der Abschlußeigenschaften in diese Grundformen zerlegt werden.

11.1.5 Beispiel: Konstruktion eines NFA mit ε-Übergängen für einen regulärem Ausdruck

Als Beispiel betrachten wir einen regulären Ausdruck r = 01*+1. Mit vollständiger Klammerung und Aufspaltung in die Operationen Vereinigung, Konkatenation und Stern sowie Substitution mit symbolischen Ausdrücken r_i (ihrerseits wieder reguläre Ausdrücke) erhält man

$$
\begin{aligned}
01^* + 1 &= \left(0\left(1^*\right)\right) + 1 \\
&= \left(r_2 \cdot \left(r_1^*\right)\right) + r_3 \\
&= \left(r_2 \cdot r_4\right) + r_3 \\
&= r_5 + r_3 \\
&= r
\end{aligned} \tag{11.12}
$$

So zerlegt lassen sich für jeden Ausdruck r_i wie im vorangegangenen Abschnitt beschrieben Automaten konstruieren und diese dann zusammensetzen. Dabei beginnt man mit dem am weitesten in der Klammerhierarchie innenliegenden Ausdruck r_1 und erweitert den Automaten immer um weitere Zustände für die umgebenden Ausdrücke.

Für r_1 ist die Operation Stern anzuwenden, daher konstruiert man einen Automaten mit 4 Zuständen, einem Start- und einem Endzustand q_1 und q_4, sowie zwei weiteren Zuständen q_2 und q_3. Vom Startzustand q_1 kommt man in einem ε-Übergang zu q_2 oder direkt zum Endzustand q_4 (leeres Wort). Mit einer 1 kommt man zu q_3 und von dort per ε-Übergang zu q_2 zurück (Stern) oder zum Endzustand q_4.

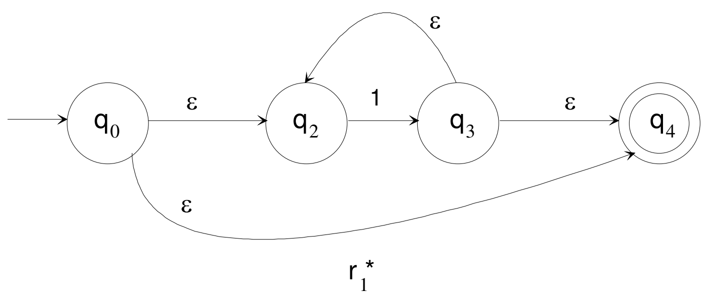


Abbildung 23 - Automat für 1*

Die Konkatenation (Und) von r_2 und r_4 benötigt lediglich zwei weitere Zustände q_5 und q_6, die vor den Automaten für r_4 geschaltet werden. Von q_5 geht der Automat nur in q_6 über, wenn eine 0 als Zeichen anliegt. Von q_6 gelangt der Automat in einem ε-Übergang in q_0, den Anfangszustand des oben beschriebenen Automaten. Dadurch ergibt sich folgender Automat

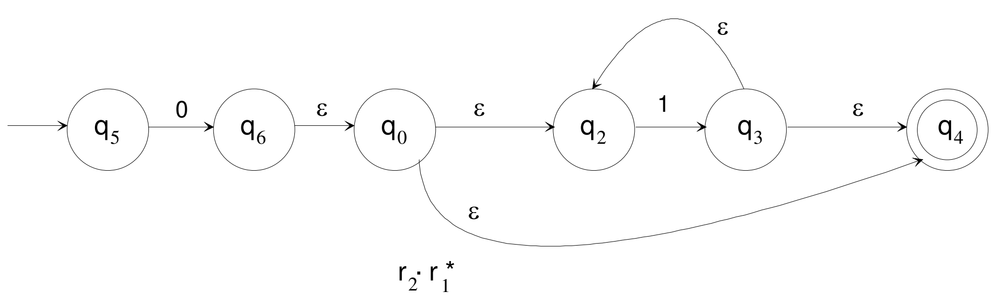


Abbildung 24 - Automat für 01*

Der noch verbleibende Ausdruck r_3 muß mit $r_5 = r_2 \cdot r_1{}^*$ in Vereinigung (Oder) aufgehen. Dazu benötigt man einen neuen Startzustand q_7 und einen neuen Endzustand q_8. Für die Konstruktion eines Automaten, der r_3 erkennt, benötigt man weitere zwei Zustände q_9 und q_{10}. Vom neuen Startzustand q_7 gehen ε-Übergänge zu den Startzuständen der beiden alternativen Automaten q_5 und q_9. Genauso gehen von den Endzuständen der Teilautomaten q_4 und q_{10} ε-Übergänge zum neuen Endzustand q_9. Weiter tritt der Automat von q_9 bei Zeichen 1 in den Zustand q_{10}.

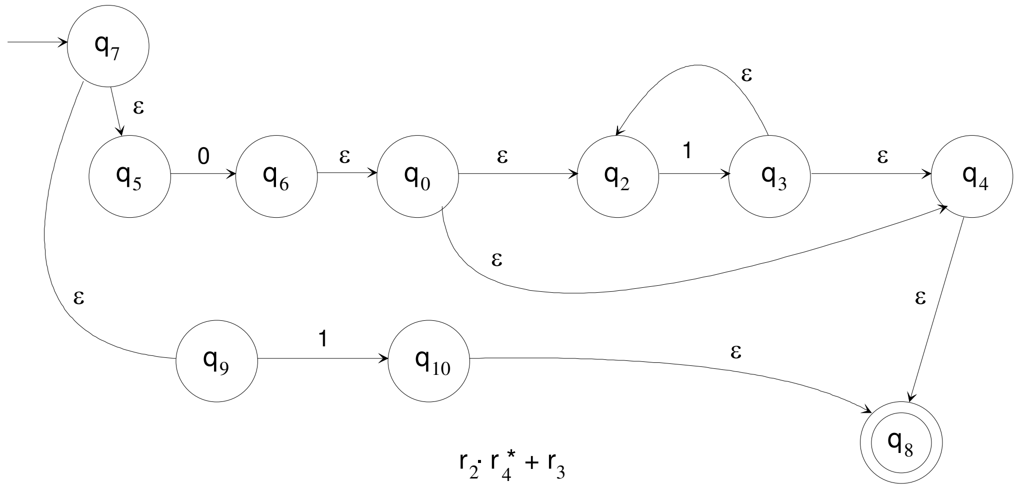


Abbildung 25 - Gesamter Automat

Für den regulären Ausdruck 01*+1 exisiert wie gezeigt ein endlicher, deterministischer Automat mit ε-Übergängen, der folgende Definition hat. Für die zehn Zustände gibt es einen Startzustand q_7 und einen Endzustand q_8. Die Übergangsfunktion entspricht dem oben aufgeführten Graphen für den gesamten Automaten.

$$M = \left(\left\{q_1, q_2, \ldots, q_{10}\right\}, \Sigma, \delta, q_7, \left\{q_8\right\}\right) \tag{11.13}$$

Auf diese Weise lassen sich für alle regulären Ausdrücke DFAs und NFAs konstruieren, die die gleiche Sprache wieder reguläre Ausdruck erkennen.

11.1.6 Von endlichen Automaten zu regulären Ausdrücken

Gemäß der Darstellung des Schaubildes zu Beginn des Protokolls, haben wir gezeigt, das sich reguläre Ausdrücke in nichtdeterministische Automaten mit ε-Übergängen umwandeln lassen. Aus vorangegangenen Protokollen ist bekannt, dass DFAs, NFAs und NFAs mit ε-Übergängen äquivalent sind. Jetzt ist der Weg zurück zu den regulären Ausdrücken zu zeigen, der letzte Pfeil im Schaubild von DFAs zu regulären Ausdrücken.

Satz:

Wenn eine Sprache L von einem DFA akzeptiert wird, dann läßt sich diese Sprache L durch einen regulären Ausdruck r beschreiben.

Beweis

Es ist zu zeigen, dass der zu bestimmende reguläre Ausdruck r die gleiche Sprache beschreibt wie der gegebene Automat M, also $L(r) = L(M)$ gilt. Sei M ein endlich deterministischer Automat, dessen Zustände von 1 bis n durchnummeriert sind und q_1 der Startzustand ist.

$$\begin{aligned} M &= \left(Q, \Sigma, \delta, q_1, F\right) \\ Q &= \left\{q_1, q_2, \ldots, q_n\right\} \end{aligned} \tag{11.14}$$

$R_{i,j}^k$ seien Sprachen mit $i, j \in \left\{1, \ldots, n\right\}$ und $k \in \left\{1, \ldots, n\right\}$, die wir durch reguläre Ausdrücke beschreiben wollen. $R_{i,j}^k$ enthält alle Wörter x, die den Automaten von Zustand q_i über verschiedene Zwischenzustände nach q_j überführen. Dabei seien q_l die besuchten Zustände[8] und es gelte $l \leq k$ für alle Wörter x.

$$R_{i,j}^k = \left\{ x \in \Sigma^* \middle| \delta(q_i, x) = q_j \text{ mit Zwischenzuständen } q_l \setminus \left\{q_i, q_j\right\} \middle| l < k \right\} \tag{11.15}$$

[8] Besuchen bezeichnet hier in den Zustand einzutreten und wieder herauszutreten

Diese Sprachen R können rekursiv definiert und per Induktion über k bewiesen werden. Für k = 0 als *Induktionsannahme* geht der Automat direkt von Zustand q_i nach q_j ohne dabei Zwischenzustände zu besuchen, da kein Zustand q_p mit $p \leq 0$ existiert. Für den Fall $i \neq j$ enthält die Sprache $R_{i,j}^0$ alle Zeichen, für die der Automat beginnend in q_i direkt in q_j übergeht.

$$R_{i,j}^0 = \left\{ a \in \Sigma \middle| \delta(q_i, a) = q_j \right\} \tag{11.16}$$

Für den Fall, dass $i = j$ gegeben ist, erkennt der Automat alle Zeichen ohne über Zwischenzustände zu gehen, bei denen er von q_i in q_i geht, also im gleichen Zustand verbleibt. Im „Zustand Verbleiben" heißt aber auch, dass der Automat keinen Zustandswechsel macht und so dass leere Wort zur Sprache gehört

$$R_{i,i}^0 = \left\{ \varepsilon \right\} \cup \left\{ a \in \Sigma \middle| \delta(q_i, a) = q_i \right\} \tag{11.17}$$

Für diese Fälle ist $R_{i,j}^0$ eine endliche Menge von Zeichen und ein regulärer Ausdruck läßt sich aufstellen. Die *Induktionsannahme* ist damit bewiesen. Ein entsprechender regulärer Ausdruck besteht aus der Veroderung der auftretenden Zeichen sowie dem leeren Wort, falls $i = j$

$$\begin{aligned} i \neq j: \quad & r_{i,j}^0 = a_1 + a_2 + \ldots + a_t \\ i = j: \quad & r_{i,i}^0 = a_1 + a_2 + \ldots + a_t + \varepsilon \end{aligned} \tag{11.18}$$

Für den *Induktionsschritt* von k auf k+1 verwenden wir die Rekursionsgleichung und setzen $R_{i,j}^k$ als gegeben voraus.

$$R_{i,j}^{k+1} = R_{i,j}^k \cup R_{i,k+1}^k \left(R_{k+1,k+1}^k \right)^* R_{k+1,j}^k \tag{11.19}$$

Wörter x, bei denen der Automat von q_i nach q_j übergeht, müssen den Zustand q_{k+1} nicht passieren. Dann wird dieser Zustand nicht benötigt und die Sprache $R_{i,j}^{k+1}$ läßt sich durch das bereits bewiesene $R_{i,j}^k$ ausdrücken. Wird der Zustand benötigt, geht der Automat von q_i in q_{k+1} über und kann innerhalb von $\left(R_{k+1,k+1}^k \right)^*$ q_{k+1} beliebig oft aufgerufen werden (Stern, Kleensche Hülle). Anschließend tritt er in q_j ein und erkennt somit Wörter, bei denen der Automat durch q_{k+1} als Zwischenzustand geht.

Ein entsprechender regulärer Ausdruck $r_{i,j}^{k+1}$ für die Sprache $R_{i,j}^{k+1}$ läßt sich aufstellen, da die Rekursionsgleichung nur die Operationen für reguläre Ausdrücke Vereinigung, Konkatenation und Hüllenbildung enthält. Nach Induktionsannahme beschreiben die regulären Ausdrücke für k bereits reguläre Sprachen. Die Anwendung der Operationen auf regulären Ausdrücke ergibt wieder reguläre Ausdrücke genauso wie die Anwendung der Operationen auf reguläre Sprachen wieder eine reguläre Sprache ergibt (siehe Abschlußeigenschaften). Damit läßt sich der folgende reguläre Ausdruck aufstellen

$$r_{i,j}^{k+1} = \left(r_{i,j}^k + r_{i,k+1}^k \left(r_{k+1,k+1}^k \right)^* r_{k+1,j}^k \right) \tag{11.20}$$

Der *Induktionsschritt* ist damit bewiesen und es lassen sich beliebige reguläre Ausdrücke aufstellen, die Teilmengen von L(M) beschreiben. Für dem Automaten M betrachten wir jetzt alle Wörter, die den Automaten vom Startzustand q_1 in einen Endzustand q_i überführen und dabei alle n Zustände des Automaten (siehe Definition am Anfang des Beweises) besuchen. Die Menge dieser Wörter ist die Sprache, die der Automat erkennt.

$$L(M) = \bigcup_{q_i \in F} R_{1,i}^n \tag{11.21}$$

Damit läßt sich analog zum vorangeganen Abschnitt ein regulärer Ausdruck konstruieren und so der Beweis abschließen. Für die möglichen Endzustände

$$F = \left\{ q_{i_1}, q_{i_2}, \ldots, q_{i_p} \right\} \tag{11.22}$$

verwenden wir die Indizes $i_1, i_2, \ldots, i_p$. Der reguläre Ausdruck ist so die Veroderung regulärer Ausdrücke, die die Wörter beschreiben, bei denen der Automat in jeweils einem gemeinsamen Endzustand hält.

$$r = r_{1,i_1}^{n} + r_{1,i_1}^{n} + \ldots + r_{1,i_p}^{n} \tag{11.23}$$

Somit wurde wurde gezeigt, dass man zu einer von einem DFA M akzeptierten Sprache L(M) einen regulären Ausdruck r angeben kann, der die gleiche reguläre Sprache L(r) = L(M) beschreibt.

Reguläre Ausdrücke, endliche deterministische, endliche nichtdeterministische sowie endliche nichtdeterministische Automaten mit ε-Übergängen sind somit wie im heutigen und den vorangegangen Protokollen gezeigt, äquivalent und beschreiben die Menge der regulären Sprachen.

12 Vorlesung vom 16. Mai 2000

12.1 Wiederholung der letzten Vorlesung

12.1.1 Abschlusseigenschaften

Für die folgenden Sprachen soll überprüft werden, ob sie zur Klasse REG der regulären Sprachen gehören.

Beispiel 1

Wir wissen, dass die Sprache $L = \left\{ a^n b^n \middle| n \geq 0 \right\}$ nicht regulär ist, folgt daraus, dass die Sprache $L' = \left\{ a^n c b^n \middle| n \geq 0 \right\}$ auch nicht regulär ist? Der argumentative Beweis nach Myhill-Nerode sieht wie folgt aus:

$$L_1 = \left\{ a^n c b^n \middle| n \geq 0 \right\} \notin REG \Leftarrow \mathrm{L}_2 = \left\{ \mathrm{a}^n \mathrm{b}^n \middle| \; n \geq 0 \right\} \notin REG \tag{12.1}$$

Es ist nun zu zeigen, dass der $Index(L_1)$ unendlich ist. Dies ist aber leicht zu sehen, wenn man die Wörter der Sprache, als Zusammensetzung aus $a^i c$ und b^i für alle i sieht. Es ist Fakt, dass es unendlich viele Wörter $a^i c$ gibt. Somit ist der Index unendlich und damit $L_1 \notin REG$.

Beispiel 2

Nun ist die Frage, ob auch die Rückrichtung gilt. In anderen Worten gilt:

$$\mathrm{L}_1 = \left\{ a^n c b^n \middle| n \geq 0 \right\} \notin REG \Rightarrow L_2 = \left\{ \mathrm{a}^n \mathrm{b}^n \middle| \; n \geq 0 \right\} \notin REG \tag{12.2}$$

Nehmen wir an $\mathrm{L}_1 = \left\{ a^n c b^n \middle| n \geq 0 \right\} \notin REG$ sei regulär. Sei weiterhin h ein Homomorphismus, welcher die folgenden Abbildungen durchführt:

$$\begin{aligned} h(a) &= a \\ h(b) &= b \\ h(c) &= \varepsilon \end{aligned} \tag{12.3}$$

Bildet man nun die Sprache:

$$L' = h^{-1}(L_2) \tag{12.4}$$

und vereinigt diese mit beliebigen Folgen der Form $a^* c b^*$ so ergibt sich die Sprache:

$$L'' = L' \cap a^* c b^* \tag{12.5}$$

Diese wäre aufgrund der Abgeschlossenheit der Regulären Sprachen immer noch regulär. Jedoch kann man diese Sprache auch umformulieren zu:

$$L'' = \left\{ a^n c b^n \middle| n \geq 0 \right\} \tag{12.6}$$

Jedoch wissen wir, von unserer Annahme, $L_2 = \left\{ \mathrm{a}^n \mathrm{b}^n \middle| \; n \geq 0 \right\}$ sei regulär, dass diese falsch ist. Somit kann auch die Sprache $L_1 = L'' = \left\{ a^n c b^n \middle| n \geq 0 \right\}$ nicht regulär sein.

Beispiel 3:

Man kann weiterhin zeigen, dass gilt:

$$L_1 = \left\{ a^n c b^n \middle| n \geq 0 \right\} \notin REG \Rightarrow L_2 = \left\{ a^n b^r c^n d^2 \middle| n, r, s \geq 0 \right\} \notin REG \tag{12.7}$$

Angenommen die Sprache L_2 wäre regulär, dann wäre auch die Sprache

$$\begin{aligned} &L' = L_2 \cap a^* b c^* d^* \\ &\Updownarrow \\ &L' = \left\{ a^n b c^n d^s \,\middle|\, n, s \geq 0 \right\} \end{aligned} \tag{12.8}$$

regulär. Nun kann man einen Homomorphismus h verwenden mit den Abbildungen:

$$\begin{aligned} h(a) &= a \\ h(b) &= c \\ h(c) &= b \\ h(d) &= \varepsilon \end{aligned} \tag{12.9}$$

Sei nun L'' die Sprache, welche erzeugt wird, indem man den Homomorphismus auf die Sprache L' anwendet:

$$L'' = h(L') = \left\{ a^n c b^n \,\middle|\, n \geq 0 \right\} \tag{12.10}$$

Diese müsste auch regulär sein, da wir jeweils nur strukturerhaltende Operationen verwendet haben. Wir wissen jedoch, dass die Sprache bestehend aus Wörtern der Form $\left\{ a^n c b^n \,\middle|\, n \geq 0 \right\}$ nicht regulär somit ergibt sich ein Widerspruch.

Beispiel 4:

Gilt:

$$L_1 = \left\{ a^n c b^n \,\middle|\, n \geq 0 \right\} \notin REG \Rightarrow L_2 = \left\{ w \,\middle|\, w \in \{a, b\}^*, \#_a(w) = 2\#_b(w) \right\} \notin REG \text{ ?} \tag{12.11}$$

Angenommen, dass L_2 regulär ist, dann ist auch die Sprache:

$$\begin{aligned} &L' = L_2 \cap a^* b^* \\ &\Updownarrow \\ &L' = \left\{ a^{2n} b^n \,\middle|\, n \geq 0 \right\} \end{aligned} \tag{12.12}$$

regulär. Wieder definieren wir einen Homomorphismus mit den Abbildungen:

$$\begin{aligned} h(a) &= aa \\ h(b) &= b \\ h(c) &= \varepsilon \end{aligned} \tag{12.13}$$

Sei nun L'' die Sprache, welche erzeugt wird, indem man den inversen Homomorphismus auf die Sprache L' anwendet:

$$\begin{aligned} &L'' = h^{-1}(L') \\ &\text{Man definiert nun:} \\ &L''' = L'' \cap a^* c b^* \\ &\Updownarrow \\ &L'' = \left\{ a^n c b^n \,\middle|\, n \geq 0 \right\} \end{aligned} \tag{12.14}$$

Diese müsste auch regulär sein, da wir jeweils wieder nur strukturerhaltende Operationen verwendet haben. Wir wissen jedoch, dass die Sprache bestehend aus Wörtern der Form $\left\{ a^n c b^n \,\middle|\, n \geq 0 \right\}$ nicht regulär somit ergibt sich ein Widerspruch.

Beispiel 5:

Zum Schluss gilt noch zu überprüfen, ob:

$$L_1 = \left\{ a^n b^n \,|\, n \geq 0 \right\} \notin REG \Rightarrow L_2 = \left\{ a^n b^m \,|\, 0 \leq m \leq n \right\} \notin REG \tag{12.15}$$

Wieder nehmen wir zunächst an, dass die Sprache L_2 regulär sei. Dann ist auch die Sprache:

$$L' = a^* b^* \cap \overline{L}_2 = \left\{ a^n b^m \,|\, 0 \leq n < m \right\} \tag{12.16}$$

regulär. Durch Konkatenation bilden wir:

$$\begin{aligned} L'' &= \{a\} \cdot L' = \left\{ a^n b^m \,|\, 0 < n \leq m \right\} \\ L''' &= L_2 \cap L'' = \left\{ a^n b^n \,|\, n \geq 1 \right\} \\ L'''' &= L''' \cap \{\varepsilon\} \end{aligned} \tag{12.17}$$

Dann sind $L'''' = L_2$. Doch wiederum gilt, da die Sprache L_1 nicht regulär ist, dass auch L_2 nicht regulär ist.

12.2 Beispiel zur Umwandlung eines Automaten dem entsprechenden Regulären Ausdruck

In diesem Abschnitt beschäftigen wir uns hauptsächlich mit dem folgenden Automaten:

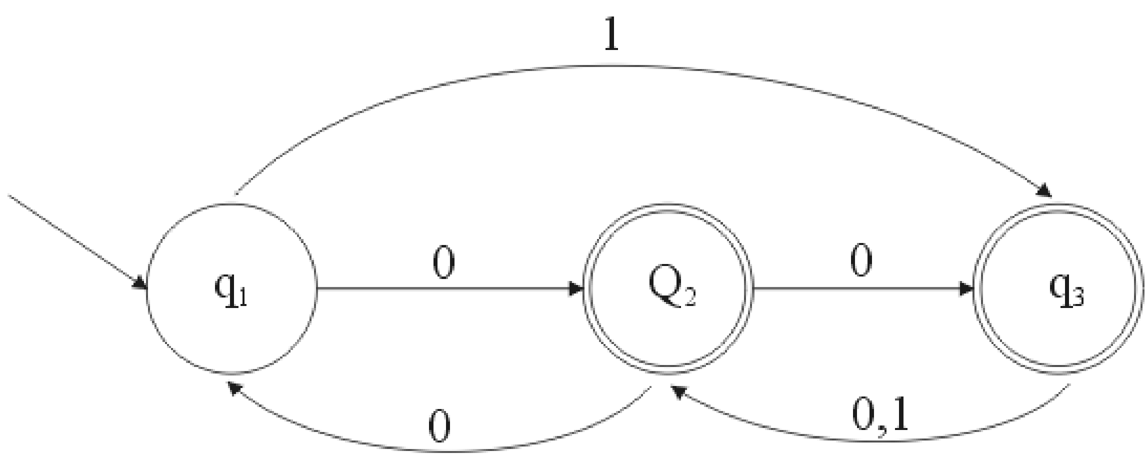


Abbildung 26: der endliche Automat des Beispiels

	k=0	k=1	k=2
r_{11}^k	ε	ε	(00)
r_{12}^k	0	0	$0(00)$
r_{13}^k	1	1	0^*1
r_{21}^k	0	0	$0(00)^*$
r_{22}^k	ε	$\varepsilon + 00$	$(00)^*$
r_{23}^k	1	$1+01$	0^*1

	k=0	k=1	k=2
r_{31}^k	$\varnothing$	$\varnothing$	$(0+1)(00)^*0$
r_{32}^k	$0+1$	$0+1$	$(0+1)(00)^*$
r_{33}^k	ε	ε	$\varepsilon+(0+1)0^*1$

Sei M der abgebildete endliche Automat. Die Wert $r_{ij}^{\;k}$ $\forall\ i$ und j, und $\text{k} = 1{,}2 \text{ oder } 3$ sind in Tabelle 1 abgebildet. Dabei wurden die regulären Ausdrücke vereinfacht, und zwar in der Form: $(r+s)t = rt+st$ oder $(\varepsilon+r)^* = r^*$. Zum Beispiel ergibt sich für $r_{22}^{\;1}$:

$$r_{22}^{\;1} = r_{21}^{\;0}\left(r_{11}^{\;0}\right)^* r_{12}^{\;0} + r_{22}^{\;0} = 0(\varepsilon)^* 0 + \varepsilon \tag{12.18}$$

Ebenso ist:

$$r_{13}^{\;2} = r_{12}^{\;1}\left(r_{22}^{\;1}\right)^* r_{23}^{\;1} + r_{13}^{\;1} = 0(\varepsilon+00)^*(1+01) \tag{12.19}$$

Bedenkt man, dass $(\varepsilon+00)^* = (00)^*$ und weiterhin gilt $1+01 = (\varepsilon+0)1$, dann ist:

$$r_{13}^{\;2} = 0(00)^*(\varepsilon+0)1+1 \tag{12.20}$$

Da weiterhin der Ausdruck $(00)^*(\varepsilon+0)$ äquivalent zu 0^* ist, ergibt sich für $0(00)^*(\varepsilon+0)1+1$ zunächst 00^*1+1 und schließlich 0^*1. Die Regulären Ausdrucke für M ergeben sich aus den akzeptierenden Zuständen. Also ist $r_{12}^{\;3} + r_{13}^{\;3}$ zu bestimmen. Mit:

$$\begin{aligned} r_{12}^{\;3} &= r_{12}^{\;2}\left(r_{33}^{\;2}\right)^* r_{32}^{\;2} + r_{12}^{\;2} \\ &= 0^*1\left(\varepsilon+(0+1)0^*1\right)^*(0+1)(00)^* + 0(00)^* \\ &= 0^*1\left((0+1)0^*1\right)^*(0+1)(00)^* + 0(00)^* \end{aligned} \tag{12.21}$$

und

$$\begin{aligned} r_{13}^{\;3} &= r_{13}^{\;2}\left(r_{33}^{\;2}\right)^* r_{33}^{\;2} + r_{13}^{\;2} \\ &= 0^*1\left(\varepsilon+(0+1)0^*1\right)^*\left(\varepsilon+(0+1)0^*1\right) + 0^*1 \\ &= 0^*1\left((0+1)0^*1\right)^* \end{aligned} \tag{12.22}$$

ergibt sich dann:

$$r_{12}^{\;3} + r_{13}^{\;3} = 0^*\left((0+1)0^*1\right)^*\left(\varepsilon+(0+1)(00)^*\right) + 0(00)^* \tag{12.23}$$

Dies stellt den Regulären Ausdruck für den Beispielautomaten dar.

12.3 Endliche 2-Wege Automaten

Dieser Abschnitt dient nur zur Information, da diese Automaten kein Thema dieser Vorlesung sein sollen. Deterministische endliche Zweiwegautomaten sind definiert als Quintupel $M = (Q, \Sigma, \delta, q_0, F)$ unterschiedlich zum „normalen“ DFA ist hier zum einem dass die Übergangsfunktion neben einem Zustand noch eine Laufrichtung zurückgibt, in welcher der nächste Zustand besucht werden soll. δ ist also auf $Q \times \Sigma^* \rightarrow Q \times \{L, R\}$ definiert. Es kann jedoch gezeigt werden, das ein und zwei-Wege endliche Automaten gleichwertig sind.

13 Vorlesung vom 23. Mai 2000

13.1 Grammatiken

13.1.1 Beispiel zu Grammatiken

Als Beispiel für eine Grammatik diene hier die Menge der booleschen Ausdrücke[9]. Das Startsymbol ist ein Nicht-Terminalsymbol, das wir „BOOL“ nennen. Als Terminalsymbole lassen wir alle Zeichen zu, die man gemeinhin für boolesche Ausdrücke verwendet. Die Produktionsregeln sind (13.1) zu entnehmen, und sollten jeden booleschen Ausdruck produzieren können:

$$
\begin{aligned}
&G := (V, \Sigma, P, S)\\
&V := \{\text{BOOL}, \text{QUANTOR}, \text{a}\ldots\text{z}, \wedge, \vee, \neg, (,)\}\\
&\Sigma := \{\text{a}\ldots\text{z}, \wedge, \vee, \neg, (,)\}\\
&P := \left\{\begin{array}{ll}
\text{BOOL} & \to (\text{BOOL}),\\
\text{BOOL} & \to \neg\text{BOOL},\\
\text{BOOL} & \to \text{BOOL} \wedge \text{BOOL},\\
\text{BOOL} & \to \text{BOOL} \vee \text{BOOL},\\
\text{BOOL} & \to \text{QUANTOR},\\
\text{QUANTOR} & \to \text{a}\ldots\text{z}
\end{array}\right\}\\
&S := \text{BOOL}
\end{aligned}
\tag{13.1}
$$

Der Ableitungsbaum eines beispielhaften booleschen Ausdruckes gemäß obiger Grammatik könnte wie folgt aussehen:

$$
\begin{aligned}
&\left(\underbrace{x}_{\text{QUANTOR}} \vee \neg \underbrace{y}_{\text{QUANTOR}}\right) \wedge \neg \left(\underbrace{a}_{\text{QUANTOR}} \wedge \underbrace{b}_{\text{QUANTOR}}\right)\\
&\left(\underbrace{\text{QUANTOR}}_{\text{BOOL}} \vee \neg \underbrace{\text{QUANTOR}}_{\text{BOOL}}\right) \wedge \neg \left(\underbrace{\text{QUANTOR}}_{\text{BOOL}} \wedge \underbrace{\text{QUANTOR}}_{\text{BOOL}}\right)\\
&\left(\text{BOOL} \vee \underbrace{\neg\text{BOOL}}_{\text{BOOL}}\right) \wedge \neg \left(\underbrace{\text{BOOL} \wedge \text{BOOL}}_{\text{BOOL}}\right)\\
&\left(\underbrace{\text{BOOL} \vee \text{BOOL}}_{\text{BOOL}}\right) \wedge \neg \underbrace{(\text{BOOL})}_{\text{BOOL}}\\
&\underbrace{(\text{BOOL})}_{\text{BOOL}} \wedge \underbrace{\neg\text{BOOL}}_{\text{BOOL}}\\
&\underbrace{\text{BOOL} \wedge \text{BOOL}}_{\text{BOOL}}\\
&\text{BOOL}
\end{aligned}
\tag{13.2}
$$

Wie man sieht, beginnt die Transformation bei einem gewöhnlichen booleschen Ausdruck, und endet bei dem Startsymbol S[10].

13.2 Die Äquivalenz von regulären Grammatiken und endlichen Automaten

Satz:

Sei $L \subseteq \Sigma^*$ eine Sprache, dann gilt:

---

[9] Für die Korrektheit übernehme ich keine Haftung ;-)

[10] Die Begriffe „starten“ und „enden“ dürfen in diesem Zusammenhang auch vertauscht werden.

$$\begin{array}{lll} 1) & \exists \text{ ein endlicher Automat } M: & L = T(M) \\ & \Updownarrow & \\ 2) & \exists \text{ eine reguläre Grammatik } G: & L = L(G) \end{array} \tag{13.3}$$

Beweis:

$1 \Rightarrow 2$:

Gegeben sei der endliche Automat

$$M = (Q, \Sigma, \delta, q_0, F) \tag{13.4}$$

Wir definieren eine neue Grammatik, die dem endlichen Automaten *M* entsprechen soll. Diese neue Grammatik benutzt das Alphabet des Automaten als Terminalalphabet, und die Zustandsmenge des Automaten als Nicht-Terminalalphabet:

$$G' := (Q \cup \Sigma, \Sigma, P, q_0) \tag{13.5}$$

Es bleibt die Frage nach der Produktionsmenge *P*.

$$P := \left\{ q \to aq' \middle| \delta(q,a) = q' \right\} \cup \left\{ q \to e \middle| q \in F \right\} \tag{13.6}$$

Wir zeigen nun, dass die durch G' definierte Menge tatsächlich die Menge ist, die von M akzeptiert wird. Für jedes Wort, das der Automat *M* akzeptiert, gilt:

$$\begin{array}{l} x \in T(M) \\ \Updownarrow \\ p := \delta(q_0, x) \in F \\ \Updownarrow \quad x = a_1 a_2 \ldots a_{|x|} \\ p = \delta\left(q_0, a_1 a_2 \ldots a_{|x|}\right) \in F \\ \Updownarrow \\ p = \delta\left(\ldots \delta\left(\delta(q_0, a_1), a_2\right) \ldots, a_{|x|}\right) \in F \\ \Updownarrow \\ \left(\delta(p_0 := q_0, a_1) = p_1\right) \wedge \left(\delta(p_1, a_2) = p_2\right) \wedge \ldots \wedge \left(\delta\left(p_{|x|-1}, a_{|x|}\right) = p\right) \wedge (p \in F) \end{array} \tag{13.7}$$

Aus (13.6) folgt:

$$\forall p_n \in Q, a_s \in \Sigma: \quad \left(\delta(p_n, a_s) = p_m\right) \Leftrightarrow \left(p_n \overset{1}{\underset{G'}{\Rightarrow}} a_s p_m\right) \tag{13.8}$$

Aus (13.7) und (13.8) folgt:

$$x \in T(M)$$
$$\Updownarrow$$
$$\left(p_0 := q_0 \overset{1}{\underset{G'}{\Rightarrow}} a_1 p_1\right) \wedge \left(p_1 \overset{1}{\underset{G'}{\Rightarrow}} a_2 p_2\right) \wedge \ldots \wedge \left(p_{|x|-1} \overset{1}{\underset{G'}{\Rightarrow}} a_{|x|} p\right) \wedge \left(p = \delta(q_0, x) \in F\right)$$
$$\Updownarrow$$
$$q_0 \overset{*}{\underset{G'}{\Rightarrow}} a_1 a_2 \ldots a_{|x|} p \tag{13.9}$$
$$\Updownarrow \quad (p \in F) \Leftrightarrow \left(p \overset{1}{\underset{G'}{\Rightarrow}} \varepsilon\right)$$
$$q_0 \overset{*}{\underset{G'}{\Rightarrow}} a_1 a_2 \ldots a_{|x|}$$
$$\Updownarrow \quad a_1, a_2, \ldots, a_{|x|} \in \Sigma \text{ (Terminalsymbole)}$$
$$x \in L(G')$$

Die Mengen sind also identisch:

$$T(M) = L(G') \tag{13.10}$$

„$\Leftarrow$“:

Gegeben sei die reguläre Grammatik

$$G = (V, \Sigma, P, S) \tag{13.11}$$

Wir definieren einen nichtdeterministischen, endlichen Automaten *M’*, welcher der regulären Grammatik entsprechen soll. Dieser neue Automat benutzt die Nicht-Terminalsymbole der Grammatik als Zustandsmenge, und die Terminalsymbole der Grammatik als Eingabealphabet. Wir definieren weiterhin den ausgezeichneten (akzeptierenden) Zustand f:

$$M' := \left((V - \Sigma) \cup \{f\}, \Sigma, \delta, S, F\right) \tag{13.12}$$

Die Menge der akzeptierenden Zustände *F* seien alle jenen Zustände (bzw. Nicht-Terminalsymbole), welche in das leere Wort übergehen können, sowie der ausgezeichnete Zustand *f*[11]:

$$F := \left\{A \middle| (A \in V - \Sigma) \wedge (A \to \varepsilon \in P)\right\} \cup \{f\} \tag{13.13}$$

Die Übergangsfunktion ist beim endlichen Automaten definiert auf dem Kreuzprodukt von Zustandsmenge und Eingabesymbol, d.h. in unserem Fall, auf dem Kreuzprodukt der Nicht-Terminalsymbole (plus „*f*“) und der Terminalsymbole. Die Übergangsfunktion liefert beim endlichen Automaten eine Teilmenge der Zustandsmenge zurück, d.h. in unserem Fall eine Teilmenge der Nicht-Terminalsymbole.

Für jeden Zustand, der einem Nicht-Terminalsymbol entspricht, welches direkt in ein Terminalsymbol übergehen kann ($A \in V - \Sigma, a \in \Sigma : A \to a \in P$), soll der neue Automat (unter anderem) in den ausgezeichnete Zustand *f* übergehen:

$$\delta(A, a) := \left\{B \middle| A \to aB \in P\right\} \cup \left\{f \middle| A \to a \in P\right\} \tag{13.14}$$

[11] Dieser, ausgezeichnete Zustand *f* ist das wichtigste Element der Menge der akzeptierenden Zustände...

Wir zeigen nun, äquivalent zum ersten Teil des Beweises, dass die von M' akzeptierte Menge tatsächlich die Menge ist, die durch G definiert wird[12]. Für jedes Wort, das unser neuer Automat *M'* akzeptiert, gilt:

$$\begin{aligned} & x \in T(M') \\ & \Updownarrow \\ & \delta(S,x) \cap F \neq \emptyset \\ & \Updownarrow \\ & \vdots \end{aligned} \tag{13.15}$$

Falls $\delta(S,x)$ einen Zustand *Q'* beinhaltet, der akzeptierend ist, so gilt gemäß (13.13):

$$\begin{aligned} & \vdots \\ & \Updownarrow \\ & \delta(S,x) \cap F \in \{A \mid (A \in V - \Sigma) \wedge (A \to \varepsilon \in P)\} \cup \{f\} \\ & \Updownarrow \\ & \vdots \end{aligned} \tag{13.16}$$

Bis hierher haben wir Umformungen vorgenommen, die eine Aussage darüber machen, wann ein Wort vom neuen Automaten akzeptiert wird, und wann nicht; in Abhängigkeit von dem Wort. Wie man an (13.16) sieht, gibt es genau zwei Möglichkeiten, warum ein Wort von dem neuen Automaten akzeptiert werden kann – von diesen zwei Möglichkeiten muss mindestens eine Möglichkeit eintreten.

Die Definition des neuen Automaten und seiner Übergangsfunktion beruht auf einzelnen Symbolen, nicht auf Wörtern. Daher zerpflücken wir an dieser Stelle die Bearbeitung eines Wortes in die Bearbeitung vieler Symbole:

$$\delta(S,x) = \bigcup_{\overset{\cdot^{\cdot^{\cdot}}}{Q_2 \in \bigcup\limits_{Q_1 \in \delta(S,a_1)} \delta(Q_1,a_2)}} \delta\left(Q_{|x|-1}, a_{|x|}\right) \tag{13.17}$$

Mit diesem Wissen lässt sich (13.16) umschreiben zu:

$$\begin{aligned} & \vdots \\ & \Updownarrow \\ & \delta\left(Q_{|x|-1}, a_{|x|}\right) \cap F \in \{A \mid (A \in V - \Sigma) \wedge (A \to \varepsilon \in P)\} \cup \{f\} \\ & \Updownarrow \\ & \vdots \end{aligned} \tag{13.18}$$

Auch bis hierher haben wir „nichts anderes", als eine Gleichung, die beschreibt, wann ein Eingabewort von unserem neuen Automaten akzeptiert wird, und wann nicht – jetzt aber nur noch in Abhängigkeit von dem letzten Symbol des Eingabewortes, sowie dem Zustand des Automaten zu diesem Zeitpunkt.

Wir haben den ausgezeichneten Zustand f genau so definiert, dass ein Übergang f ergibt, falls der ursprüngliche Zustand gemäß Grammatik in das Eingabesymbol übergehen kann (siehe (13.14)):

$$\left(f \in \delta(A,a)\right) \Rightarrow (A \to a \in P) \tag{13.19}$$

[12] Dieser Beweis wurde im Skript komplett weggelassen. Warum, ist mir nicht klar.

Mit diesem Wissen folgt:

$$
\begin{aligned}
&\vdots\\
&\Updownarrow\\
&\underbrace{\left(\cancel{\delta\left(Q_{|x|-1}, a_{|x|}\right) \cap F \in \left(A \in V - \Sigma\right)} \wedge Q_{|x|-1} \rightarrow \varepsilon \in P\right)}_{\text{Möglichkeit 1}} \vee \underbrace{\left(Q_{|x|-1} \rightarrow a_{|x|} \in P\right)}_{\text{Möglichkeit 2}}\\
&\Updownarrow\\
&\underbrace{\left(Q_{|x|-1} \rightarrow \varepsilon \in P\right)}_{\text{Möglichkeit 1}} \vee \underbrace{\left(Q_{|x|-1} \rightarrow a_{|x|} \in P\right)}_{\text{Möglichkeit 2}}\\
&\Updownarrow\\
&\vdots
\end{aligned}
\tag{13.20}
$$

Die obige, boolesche „Kürzung" durften wir vornehmen, da diese Bedingung gemäß der Definition der Übergangsfunktion immer erfüllt ist. Es gilt nun, irgendwie den „Weg zurück" zu finden vom Zustand $Q_{|x|-1}$ zum Zustand *S*. Aus (13.14) folgt:

$$
\begin{aligned}
&x \in T\left(M'\right)\\
&\Updownarrow\\
&\delta\left(S, a_1\right) \ni Q_1 \wedge\\
&\wedge\, \delta\left(Q_1, a_2\right) \ni Q_2 \wedge\\
&\wedge \ldots \wedge\\
&\wedge\, \delta\left(Q_{|x|-3}, a_{|x|-2}\right) \ni Q_{|x|-2} \wedge\\
&\wedge\, \delta\left(Q_{|x|-2}, a_{|x|-1}\right) \ni Q_{|x|-1}\\
&\Updownarrow\\
&\left(S \rightarrow a_1 Q_1 \in P\right) \wedge\\
&\wedge \left(Q_1 \rightarrow a_2 Q_2 \in P\right) \wedge\\
&\wedge \ldots \wedge\\
&\wedge \left(Q_{|x|-3} \rightarrow a_{|x|-2} Q_{|x|-2} \in P\right) \wedge\\
&\wedge \left(Q_{|x|-2} \rightarrow a_{|x|-1} Q_{|x|-1} \in P\right)\\
&\Updownarrow\\
&S \overset{*}{\underset{G}{\Rightarrow}} a_1 \ldots a_{|x|-1} Q_{|x|-1}
\end{aligned}
\tag{13.21}
$$

Oho. Wir wissen nun also, dass – falls ein Wort von unserem neuen Automaten akzeptiert wird, das zugehörige Nicht-Terminalsymbol Bestandteil der durch die Grammatik definierten Sprache ist. Weiterhin wissen wir aus (13.20), dass der „letzte Zustand" des Automaten in ein Terminalsymbol übergeht. Daraus folgt:

$$\begin{gathered} x \in T(M') \\ \Updownarrow \\ \left(S \overset{*}{\underset{G}{\Rightarrow}} a_1 \ldots a_{|x|-1} Q_{|x|-1} \right) \wedge \left(\underbrace{\left(Q_{|x|-1} \rightarrow \varepsilon \in P \right)}_{\text{Möglichkeit 1}} \vee \underbrace{\left(Q_{|x|-1} \rightarrow a_{|x|} \in P \right)}_{\text{Möglichkeit 2}} \right) \\ \Updownarrow \\ \left(S \overset{*}{\underset{G}{\Rightarrow}} a_1 \ldots a_{|x|-1} \right) \wedge \left(S \overset{*}{\underset{G}{\Rightarrow}} a_1 \ldots a_{|x|-1} a_{|x|} \right) \\ \Updownarrow \\ S \overset{*}{\underset{G}{\Rightarrow}} x \end{gathered} \tag{13.22}$$

Die Mengen sind also identisch:

$$T(M') = L(G) \tag{13.23}$$

13.3 Die Sprache „$a^n b^n$"

Satz:

Die Sprache $L := \left\{ a^n b^n \,\middle|\, n \geq 1 \right\}$ ist nicht regulär!

Beweis:

Wir führen einen Widerspruchsbeweis durch, und behaupten, *L* ist regulär. Dann existiert auch ein endlicher Automat *M*, welcher L akzeptiert:

$$\begin{aligned} & L \in \text{REG} \\ & \Downarrow \\ & \exists \text{DFA}\, M: \qquad && M = (Q, \Sigma, \delta, q_0, F) \\ & && L = T(M) \end{aligned} \tag{13.24}$$

Wir gehen davon aus, dass dieser Automat existiert und funktioniert. Wir betrachten diesen Automaten nach der „Hälfte" der Ausführungszeit, d.h. zu dem Zeitpunkt, an dem er die *a*'s abgearbeitet hat, und die *b*'s noch nicht. Wir wählen ein Eingabewort, dass genau so viele *a*'s enthält, wie der Automat Zustände besitzt. Nach der Verarbeitung aller *a*'s befindet sich der Automat in einem Zustand, den wir *q* nennen:

$$\begin{gathered} n := |Q| \\ \delta(q_0, a^n) = q \end{gathered} \tag{13.25}$$

Der Automat ist in dem Startzustand q_0 gestartet, hat n Eingabezeichen verarbeitet, und hat demzufolge n mal seinen Zustand gewechselt. Außer dem Startzustand q_0 existieren noch $(n-1)$ weitere Zustände. Daraus folgt, dass der Automat mindestens einen Zustand mehr als einmal angenommen hat:

$$\begin{aligned}
&\exists q' \in Q, \quad i,j,k \in \mathbb{N} : \\
&(j \geq 1) \quad \wedge \quad (i+j+k=n) \quad \wedge \\
&\delta(q_0, a^i) = q' \quad \wedge \\
&\underbrace{\delta(q', a^j) = q'}_{\text{"Schleife"}} \quad \wedge \\
&\delta(q', a^k) = q
\end{aligned} \tag{13.26}$$

Für den Automaten ist es logischerweise „egal", ob er die Schleife ausführt, oder nicht. Daraus folgt, dass der Automat zwischen den beiden Eingabewörtern a^i und a^{i+j} indifferent ist:

$$\delta(q_0, a^{i+j}) = \delta\left(\underbrace{\delta(q_0, a^i)}_{q'}, a^j\right) = \delta(q', a^j) = q' = \delta(q_0, a^i) \tag{13.27}$$

Das bedeutet ganz allgemein, dass man die Schleife auch weglassen kann, d.h. die a^j's aus dem Eingabewort weglassen kann, ohne dass sich am Verhalten des Automaten irgendetwas ändert:

$$\begin{aligned}
\delta(q_0, a^{i+k} b^{i+j+k}) &= \delta(\delta(q_0, a^i), a^k b^{i+j+k}) = \\
&= \delta(\delta(q_0, a^{i+j}), a^k b^{i+j+k}) = \\
&= \delta(q_0, a^{i+j+k} b^{i+j+k}) = \\
&= \delta(q_0, a^n b^n)
\end{aligned} \tag{13.28}$$

Wir haben durch (13.24) definiert, dass Eingabewörter der Form $a^n b^n$ von dem Automaten akzeptiert werden, d.h. den Automaten vom Startzustand aus in einen akzeptierenden Zustand versetzen. Das gleiche Verhalten gilt demnach für Eingabewörter der Form $a^{i+k} b^{i+j+k}$, mit $j>0$:

$$\begin{aligned}
&a^{i+j+k} b^{i+j+k} \in T(M) \\
&\Downarrow \\
&\delta(q_0, a^{i+j+k} b^{i+j+k}) \in F \\
&\Downarrow \\
&\delta(q_0, a^{i+k} b^{i+j+k}) \in F \\
&\Downarrow \\
&a^{i+k} b^{i+j+k} \in T(M) \qquad \text{Widerspruch!} \\
&\Downarrow \\
&\mathrm{L} \notin \mathrm{REG}
\end{aligned} \tag{13.29}$$

Unser Automat akzeptiert demzufolge auch Eingabewörter der Form $a^{i+k} b^{i+j+k}$, welche aber nicht Bestandteil der gegebenen Sprache L sind. Die Sprache L ist nicht regulär!

13.4 Das Pumping-Lemma

Satz:

„Wenn man einen Automaten hat, und ein akzeptiertes Wort kennt, dessen Länge größer ist als die Anzahl der Zustände des Automaten, dann ist die durch den Automaten akzeptierte Sprache unendlich."

Wir haben im vorangegangenen Beispiel gesehen, dass es einen Zusammenhang gibt, zwischen der Anzahl von Zuständen eines Automaten, der Länge eines Wortes der erzeugten Sprache, und der Anzahl der Zustandsschleifen während der Verarbeitung dieses Wortes.

Das Pumping-Lemma generalisiert diese Erkenntnis: Nach dem Pumping-Lemma gibt es für jede Sprache L eine bestimmte Wortlänge n, ab welcher Wörter, die in der Sprache enthalten sind, zwangsweise solche „Schleifen" enthalten:

$$\begin{aligned}
&\forall L \in \mathrm{REG} \quad \exists n \geq 1 \quad \forall x \in L: \\
&\left(|x| \geq n\right) \Rightarrow \exists u, v, w \in \Sigma^*: 1)\ x = uvw \\
&\qquad\qquad\qquad\qquad\qquad 2)\ 0 < |v| < n \\
&\qquad\qquad\qquad\qquad\qquad 3)\ u \underbrace{v^i}_{\text{Schleife}} w \in L \quad \forall i \geq 0
\end{aligned} \tag{13.30}$$

Beweis:

Der Beweis verläuft ähnlich wie im vorangegangenen Beispiel. Wenn L regulär ist, existiert auch ein endlicher Automat M, welcher L akzeptiert:

$$\begin{aligned}
&L \in \mathrm{REG} \\
&\Downarrow \\
&\exists \mathrm{DFA}\, M: \qquad M = \left(Q, \Sigma, \delta, q_0, F\right) \\
&\qquad\qquad\qquad\quad L = T\left(M\right)
\end{aligned} \tag{13.31}$$

Wir dürfen davon ausgehen, dass dieser Automat existiert, und dass er eine endlich Zustandsmenge besitzt:

$$n := |Q| \tag{13.32}$$

Angenommen, es existiert ein Eingabewort, welches von dem Automaten akzeptiert wird, und welches genauso viele Buchstaben besitzt, wie der Automat Zustände hat. Dann startet der Automat in dem Startzustand q_0, verarbeitet n Eingabezeichen, und wechselt demzufolge n mal seinen Zustand. Außer dem Startzustand q_0 existieren noch *(n-1)* weitere Zustände. Daraus folgt, dass der Automat mindestens einen Zustand mehr als einmal annimmt:

$$\begin{aligned}
&\exists x \in L: \quad |x| \geq n \\
&\Downarrow \\
&\delta\left(q_0, x\right) \in F \\
&\Downarrow \\
&\exists q' \in Q, \quad u, w \in \Sigma^*, \quad v \in \Sigma^+: \\
&x = uvw \quad \wedge \quad \delta\left(q_0, u\right) = q' \quad \wedge \quad \underbrace{\delta\left(q', v\right) = q'}_{\text{"Schleife"}} \quad \wedge \quad \delta\left(q', w\right) \in F
\end{aligned} \tag{13.33}$$

Für den Automaten ist es logischerweise „egal", ob, und wie oft er die Schleife ausführt. Daraus folgt, dass der Automat zwischen den Eingabewörtern der Form wv^iw für $i>0$ indifferent ist:

$$\begin{aligned}
\delta\left(q_0, uv^i w\right) &= \delta\left(\delta\left(\delta\left(q_0, u\right), v^i\right), w\right) = \\
&= \delta\left(\underbrace{\delta\left(\underbrace{\delta\left(q_0, u\right)}_{q'}, v^i\right)}_{q'}, w\right) = \\
&= \delta\left(q', w\right) \in F
\end{aligned} \tag{13.34}$$

Daraus folgt:

$$uv^i w \in L \tag{13.35}$$

13.5 Entscheidbarkeit

13.5.1 Prädikate

Der Begriff des „Prädikates“ ist der Prädikatenlogik als Teilgebiet der mathematischen Logik entnommen. Ein Prädikat ist eine Aussage, die entweder wahr oder falsch ist. Ein Prädikat wird zumeist als mathematische Funktion definiert.

13.5.2 Definition der Entscheidbarkeit

Sei *S* eine Menge, und $p : S \to \{0,1\}$ ein Prädikat mit

$$\forall s \in S:\ p(s) := \begin{cases} 1 & \text{falls } p(s) \text{ wahr} \\ 0 & \text{falls } p(s) \text{ falsch} \end{cases} \tag{13.36}$$

Es gilt:

$$\begin{aligned} &\text{Problem } \langle \text{S,p} \rangle \text{ ist entscheidbar} \\ &\Updownarrow_{\text{Df.}} \\ &\exists \text{ Algorithmus/Programm } A:\quad \forall s \in S:\quad A(s) = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \Leftrightarrow p(s) \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \end{aligned} \tag{13.37}$$

Entscheidbar sind z.B. folgende Fragen:

1. T(M)=∅ ?
2. T(M) endlich?
3. T(M) unendlich?

Die Beweise dafür erbringen wir im Folgenden.

13.5.3 Beispiel: Entscheidbarkeit der leeren Sprache

Behauptung:

Die Frage, ob *T(M)=∅*, ist entscheidbar.

Beweis:

Sei $M = (Q, \Sigma, \delta, q_0, F)$, $n := |Q|$. Es gilt:

$$T(M) = \varnothing \Leftrightarrow \left(\underbrace{\forall x \in \Sigma^* : |x| < n}_{\text{Entscheidbares Problem}} \Rightarrow x \notin T(M) \right) \tag{13.38}$$

Wieso?

„Hinrichtung“:

Diese Richtung ist trivial. Wenn der Automat überhaupt kein einziges Wort akzeptiert, dann erst recht keines, das eine bestimmte Länge hat.

Rückrichtung:

Hierfür bemühen wir einen Widerspruchsbeweis. Wir nehmen an, dass der Automat alle Wörter der beschriebenen Länge nicht akzeptiert, aber es trotzdem Wörter gibt, die der Automat existiert:

Annahme:

$$\left(\forall x \in \Sigma^* : |x| < n \Rightarrow x \notin T(M)\right) \wedge \left(T(M) \neq \varnothing\right)$$
$$\Downarrow \tag{13.39}$$
$$\exists y : \left(y \in T(M)\right) \wedge \left(|y| \geq n\right)$$

Wir betrachten nun das kürzeste dieser akzeptierten Wörter, und nennen es x:

$$x := z \in L \quad \middle| \forall z' \in L \Rightarrow \left(|z'| \geq |z|\right) \tag{13.40}$$

Gemäß Annahme (13.39) gilt für die Länge dieses Wortes:

$$|x| \geq n \tag{13.41}$$

Nach dem Pumping-Lemma gilt, dass dieses Wort in dem endlichen Automaten „Schleifen“ verursachen muss:

$$\exists x_1, x_2, x_3 : \left(x = x_1 x_2 x_3\right) \wedge \left(1 \leq |x_2| \leq n\right) \wedge \left(x_1 x_3 \in L\right)$$
$$\Downarrow$$
$$|x_1 x_3| < |x| \tag{13.42}$$
$$\Downarrow$$
$$|x_1 x_3| \geq n - 1 \qquad \text{Widerspruch!}$$

Das bedeutet, dass es automatisch auch ein Wort geben müsste, welches ebenfalls akzeptiert wird, aber gegen die Annahme (13.39) verstößt. Also gilt (13.38), und damit gilt:

$$T(M) \overset{?}{\neq} \varnothing \text{ ist entscheidbar!} \tag{13.43}$$

13.5.4 Beispiel: Entscheidbarkeit der unendlichen Sprache

Behauptung:

Die Frage, ob *T(M)* unendlich ist, ist entscheidbar.

Beweis:

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DFA, $n := |Q|$. Es gilt:

$$T(M) \text{ unendlich} \Leftrightarrow \exists x \in T(M) : \underbrace{\left(n \leq |x| \leq 2n - 1\right)}_{\text{Entscheidbares Problem}} \tag{13.44}$$

Wieso?

Rückrichtung:

Wenn ein Wort von dem Automaten akzeptiert wird, und genauso viele oder mehr Buchstaben besitzt, wie der Automat Zustände, dann ist gemäß dem Pumping-Lemma eine Schleife in dem Automatenablauf, die „aufgeblasen werden kann“:

$$\left(x \in T(M)\right) \wedge \left(|x| \geq n\right)$$
$$\Downarrow$$
$$\exists u, v, w : \left(uvw = x\right) \wedge \left(\left\{uv^i w \middle| i \geq 0\right\} \subseteq T(M)\right) \tag{13.45}$$
$$\Downarrow$$
$$T(M) \text{ unendlich}$$

„Hinrichtung“:

Wir gehen von der Menge aller Wörter aus, die von dem Automaten akzeptiert werden, und länger sind als *2n-1*. Falls diese Menge leer ist, bedeutet das, dass die erzeugte Sprache ohnehin nicht unendlich sein kann. Wir gehen davon aus, dass diese Menge nicht leer ist. Aus dieser Menge greifen wir uns dann eines der kürzesten Wörter heraus, und nennen es *y*:

$$\begin{aligned} &S := \left\{ x \middle| \left(x \in T(M) \right) \wedge \left(|x| \geq 2n \right) \right\} \\ &S \neq \varnothing ! \end{aligned} \tag{13.46}$$

$$y := z \in S \,\middle|\, \forall z' \in S : |z'| \geq |z| \tag{13.47}$$

Da dieses Wort immer noch mehr Buchstaben besitzt, als der Automat Zustände, schlägt das Pumping-Lemma zu, und wir können eine „Schleife“ aus diesem Wort entfernen:

$$\begin{aligned} &|y| \geq n \\ &\Downarrow \quad {\scriptstyle \text{Pumping-Lemma}} \\ &\exists \text{u,v,w:}(\text{uvw=y}) \wedge \left(1 \leq |v| \leq n\right) \wedge \left(uw \in T(M)\right) \\ &\Downarrow \quad {\scriptstyle 1 \leq |v|} \\ &|uw| < |y| = |uvw| \\ &\Downarrow \\ &|uw| \leq 2n-1 \end{aligned} \tag{13.48}$$

Damit haben wir schon die halbe Miete. Wir wissen nun, dass es automatisch ein akzeptiertes Wort geben muss, welches eine Länge kleiner oder gleich *2n-1* besitzt. Nun gilt es noch zu zeigen, dass dieses Wort auch eine Länge von größer oder gleich *n* besitzt:

$$\begin{aligned} &|uw| = |uvw| - |v| \geq \min\left(|uvw|\right) - \max\left(|v|\right) \\ &\Downarrow \quad {\scriptstyle |v| \leq n, |uvw| \geq 2n} \\ &|uw| \geq 2n - n \\ &\Downarrow \\ &|uw| \geq n \end{aligned} \tag{13.49}$$

Damit ist der Beweis vollbracht. Es gilt also:

$$\left(uw \in T(M)\right) \wedge \left(n \leq |uw| \leq 2n-1\right) \tag{13.50}$$

Also gilt (13.44), und daraus folgt:

$$\text{"}T(M) \text{ unendlich?" ist entscheidbar!} \tag{13.51}$$

13.5.5 Beispiel: Entscheidbarkeit der endlichen Sprache

Behauptung:

Die Frage, ob *T(M)* endlich ist, ist entscheidbar.

Beweis:

$$T(M) \text{ endlich} \quad \Leftrightarrow \quad \neg\left(T(M) \text{ unendlich}\right) \tag{13.52}$$

Somit gilt:

$$\text{"}T(M) \text{ endlich?" ist entscheidbar!} \tag{13.53}$$

13.5.6 Beispiel: Entscheidbarkeit der Schnittmenge

Behauptung:

Die Frage, ob für zwei beliebige, endliche Automaten M_1 und M_2 $T(M_1) \cap T(M_2) \neq \varnothing$ gilt, ist entscheidbar.

Beweis:

Es ist bekannt, dass die Schnittmenge zweier regulärer Automaten (bzw. Mengen) wieder regulär ist:

$$\begin{aligned} &T(M_1), T(M_2) \in \mathrm{REG} \\ &\Downarrow \\ &T(M) := T(M_1) \cap T(M_2) \in \mathrm{REG} \end{aligned} \tag{13.54}$$

Wir haben bereits bewiesen, dass es für einen einzelnen Automaten entscheidbar ist, ob er eine leere Sprache akzeptiert, oder nicht:

$$T(M) \overset{?}{\neq} \varnothing \text{ ist entscheidbar!} \tag{13.55}$$

13.5.7 Beispiel: Entscheidbarkeit der Teilmenge

Behauptung:

Die Frage, ob für zwei beliebige, endliche Automaten M_1 und M_2 $T(M_1) \subseteq T(M_2)$ gilt, ist entscheidbar.

Beweis:

Wir führen den Beweis über eine weitere Behauptung, die wir zuerst beweisen:

Behauptung α:

$$\left(T(M_1) \subseteq T(M_2)\right) \Leftrightarrow \left(T(M_1) \cap \overline{T(M_2)} = \varnothing\right) \tag{13.56}$$

Beweis α:

„Hinrichtung“:

$$\begin{aligned} &\forall x: \left(x \in T(M_1)\right) \Rightarrow \left(x \in T(M_2)\right) \\ &\Downarrow \\ &\forall x: \left(x \notin T(M_2)\right) \Rightarrow \left(x \notin T(M_1)\right) \\ &\Downarrow \\ &\forall x: \left(x \in \overline{T(M_2)}\right) \Rightarrow \left(x \notin T(M_1)\right) \\ &\Downarrow \\ &T(M_1) \cap \overline{T(M_2)} = \varnothing \end{aligned} \tag{13.57}$$

Rückrichtung:

Beweis über Contraposition ($\neg \Rightarrow \neg$):

$$\begin{aligned}
&T(M_1) \not\subseteq T(M_2) \\
&\Downarrow \\
&\exists x : \left(x \in T(M_1)\right) \wedge \left(x \notin T(M_2)\right) \\
&\Downarrow \\
&\exists x : \left(x \in T(M_1)\right) \wedge \left(x \in \overline{T(M_2)}\right) \\
&\Downarrow \\
&T(M_1) \cap T(M_2) \neq \emptyset
\end{aligned} \tag{13.58}$$

Also gilt die Behauptung α! Weiter im Beweis...

Wir konstruieren uns gedanklich einen dritten Automaten:

$$\begin{aligned}
&T(M) := \overline{T(M_2)} \\
&\Downarrow
\end{aligned} \tag{13.59}$$

Unter Zuhilfenahme von (13.56) folgt daraus:

$$\left(T(M_1) \subseteq T(M_2)\right) \Leftrightarrow \neg\left(T(M_1) \cap T(M) \neq \emptyset\right) \tag{13.60}$$

Wir haben aber bereits in „13.5.6 Beispiel: Entscheidbarkeit der Schnittmenge“ gezeigt, dass die Frage, ob die Schnittmenge der durch zwei endliche Automaten definierten Sprachen nicht leer ist, entscheidbar ist. Damit ist der Beweis erbracht:

$$T(M_1) \overset{?}{\subseteq} T(M_2) \text{ ist entscheidbar!} \tag{13.61}$$

13.5.8 Beispiel: Entscheidbarkeit der Äquivalenz

Behauptung:

Die Frage, ob für zwei beliebige, endliche Automaten M_1 und M_2 $T(M_1) = T(M_2)$ gilt, ist entscheidbar.

Beweis α:

Wir können den Beweis nun auf bereits Bewiesenes zurückführen:

$$\begin{aligned}
&T(M_1) = T(M_2) \\
&\Updownarrow \\
&\left(T(M_1) \subseteq T(M_2)\right) \wedge \left(T(M_2) \subseteq T(M_1)\right)
\end{aligned} \tag{13.62}$$

Den Beweis, dass dieser Zusammenhang entscheidbar ist, haben wir in „13.5.7 Beispiel: Entscheidbarkeit der Teilmenge“ erbracht. Daraus folgt:

$$T(M_1) \overset{?}{=} T(M_2) \text{ ist entscheidbar!} \tag{13.63}$$

Beweis β:

Wir konstruieren uns zwei minimale Automaten, die äquivalent zu den beiden gegebenen Automaten sind. Dann gilt:

$$\begin{aligned}&M_1' := \text{Nerodeautomat}(\mathrm{M}_1), M_2' := \text{Nerodeautomat}(\mathrm{M}_2)\\&\Downarrow\\&T(M_1) = T(M_2) \Leftrightarrow T(M_1') = T(M_2')\end{aligned} \tag{13.64}$$

Vorausgesetzt, dass die beiden Automaten wirklich die selbe Sprache definieren, müssen die dazugehörigen Nerode-Automaten isomorph sein, d.h.:

$$\exists \text{ Isomorphismus } h: \quad h: M_1' \to M_2' \tag{13.65}$$

Es ist nun möglich, alle denkbaren Abbildungen von M_1' nach M_2' auszuprobieren, und festzustellen, ob eine davon ein Isomorphismus ist. Es kann kein Isomorphismus dabei sein, wenn die Automaten nicht identisch sind... Auch hieraus folgt:

$$T(M_1) \stackrel{?}{=} T(M_2) \text{ ist entscheidbar!} \tag{13.66}$$

13.6 Kontext-freie Grammatiken

Definition:

$$\begin{aligned}&\text{Eine Grammatik } G = (V, \Sigma, P, S) \text{ heißt "kontextfrei"}\\&\Updownarrow_{\text{Df.}}\\&\forall p \in P: \quad p \in (V - \Sigma) \times V^*\end{aligned} \tag{13.67}$$

Im Gegensatz zur normalen Grammatik stehen hier auf „der linken Seite“ der Produktionen also lediglich Nicht-Terminalsymbole, und auch immer nur eines davon. Bei der normalen Grammatik standen hier auch Terminalsymbole, und dazu noch beliebig viele (konkateniert).

13.6.1 Beispiel zur Kontext-freien Grammatik

Gegeben sei die Grammatik:

$$\begin{aligned}&G := (V, \Sigma, P, S)\\&V := \{S, a, b\}\\&\Sigma := \{a, b\}\\&P := \{S \to aSa, S \to bSb, S \to \varepsilon\}\end{aligned} \tag{13.68}$$

Behauptung:

$$L(G) = \left\{ ww^R \middle| w \in \{a,b\}^* \right\} \tag{13.69}$$

„*R*“ bedeutet in diesem Zusammenhang das „umgedrehte Wort“. Die Grammatik beschreibt also *Palyndrome*.

Beweis:

Wir führen den Beweis mittels vollständiger Induktion über die Länge der Produktionen durch.

Induktionsbehauptung:

$$\forall n \geq 1, x \in \Sigma^*: \quad S \stackrel{n}{\Rightarrow} x \quad \Leftrightarrow \quad \left(x = wSw^R \wedge |w| = n\right) \vee \left(x = ww^R \wedge |w| = n-1\right) \tag{13.70}$$

Induktionsverankerung (n=1):

„Hinrichtung“:

$$\begin{aligned}
&S \overset{1}{\Rightarrow} x \\
&\Downarrow \\
&(x = aSa) \vee (x = bSb) \vee (x = \varepsilon) && (13.71) \\
&\Downarrow \\
&\left(x = wSw^R \,\|w| = 1\right) \vee \left(x = ww^R \,\|w| = 0\right)
\end{aligned}$$

Rückrichtung:

Fallunterscheidung, Fall 1:

$$\begin{aligned}
&\text{Sei } \left(x = wSw^R\right) \wedge \left(|w| = 1\right) \\
&\Downarrow \\
&(x = aSa) \vee (x = bSb) && (13.72) \\
&\Downarrow \\
&\left(S \overset{1}{\Rightarrow} aSa = x\right) \vee \left(S \overset{1}{\Rightarrow} bSb = x\right)
\end{aligned}$$

Fallunterscheidung, Fall 2:

$$\begin{aligned}
&\text{Sei } \left(x = ww^R\right) \wedge \left(|w| = 0\right) \\
&\Downarrow \\
&x = \varepsilon && (13.73) \\
&\Downarrow \\
&S \overset{1}{\Rightarrow} \varepsilon = x
\end{aligned}$$

Induktionsschritt (n→n+1):

„Hinrichtung“:

$$\begin{aligned}
&S \overset{n+1}{\Rightarrow} x \\
&\Downarrow \\
&S \overset{n}{\Rightarrow} x' \overset{1}{\Rightarrow} x \\
&\Downarrow && (13.74) \\
&\left(\left(x' = wSw^R\right) \wedge \left(|w| = n\right)\right) \vee \cancel{\left(\left(x' = ww^R\right) \wedge \left(|w| = n-1\right)\right)} \\
&\Downarrow \\
&\vdots
\end{aligned}$$

Die „Kürzung" in dem obigen Ausdruck ist erlaubt, bzw. muss sogar sein, weil in diesem Fall kein Nicht-Terminalsymbol (d.h. *S*) mehr in dem Wort enthalten wäre. Es gäbe keine Produktion, die es erlauben würde, das Wort noch weiter zu verändern.

$$\vdots$$
$$\Downarrow$$
$$\left(x = waSaw^R\right) \vee \left(x = wbSbw^R\right) \vee \left(x = ww^R\right) \tag{13.75}$$
$$\Downarrow$$
$$\left(\left(x = w'Sw'^R\right) \wedge \left(|w| = n+1\right)\right) \vee \left(\left(x = ww^R\right) \wedge \left(|w| = n\right)\right)$$

Rückrichtung:

Fallunterscheidung, Fall 1:

Sei $\left(x = wSw^R\right) \wedge \left(|w| = n+1\right)$
$$\Downarrow$$
$$\left(\left(x = w'aSaw'^R\right) \vee \left(x = w'bSbw'^R\right)\right) \wedge \left(|w'| = n\right)$$
$$\Downarrow$$
$$S \overset{n}{\Rightarrow} w'Sw'^R \wedge \left(\left(w'Sw'^R \overset{1}{\Rightarrow} w'aSaw'^R\right) \vee \left(w'Sw'^R \overset{1}{\Rightarrow} w'bSbw'^R\right)\right) \tag{13.76}$$
$$\Downarrow$$
$$\left(S \overset{n+1}{\Rightarrow} w'aSaw'^R\right) \vee \left(S \overset{n+1}{\Rightarrow} w'bSbw'^R\right)$$
$$\Downarrow$$
$$\left(S \overset{n+1}{\Rightarrow} wSw^R\right)$$

Fallunterscheidung, Fall 2:

Sei $\left(x = ww^R\right) \wedge \left(|w| = n\right)$
$$\Downarrow$$
$$\left(S \overset{n}{\Rightarrow} wSw^R\right) \wedge \left(wSw^R \overset{1}{\Rightarrow} ww^R\right) \tag{13.77}$$
$$\Downarrow$$
$$\left(S \overset{n+1}{\Rightarrow} ww^R\right)$$

14 Vorlesung vom 25. Mai 2000

14.1 Ableitungen von Wörtern in kontextfreien Grammatiken

14.1.1 Ableitungen

Durch Ableitungen auf einer kontextfreien Grammatik formt man Nichtterminalsymbole in einem oder mehreren Ableitungsschritten auf Konkatenationen von Terminal- und Nichtterminalsymbolen um. Diesen Vorgang wollen wir näher betrachten.

Sei $G=(V,\Sigma,P,S)$ eine Grammtik, dann wir eine Sprache L(G), die von dieser Grammatik generiert wird. Hier und im Folgenden sei V die Menge der Nichtterminalsymbole und Σ die Menge der Terminalsymbole. Die Produktionen P dieser Grammatik wandeln (auch resubstitiuieren, ersetzen) Nichtterminalsymbole A in Folgen von Nichtterminal- und Terminalsymbolen $(V\cup\Sigma)^*$ um.

$$A \rightarrow X^* \qquad \text{mit } A \in V, X \in (V\cup\Sigma) \tag{14.1}$$

Eine Produktion aus P: $A \rightarrow \beta$ angewandt auf $\alpha A \gamma$ mit $\alpha,\gamma \in (V\cup\Sigma)^*$ liefert

$$\alpha A \gamma \Rightarrow \alpha\beta\gamma \tag{14.2}$$

Damit ist $\Rightarrow$ eine Relation auf der Menge der vereingten Terminal- und Nichtterminalsymbole $(V\cup\Sigma)^*$ und bezeichnet die Anwendung einer einzelnen Produktion. Die Anzahl der angewandten Produktionen, d. h. die Ableitungsschritte, kann über den Doppelpfeil geschrieben werden. Erfolgt die Ableitung der linken auf die rechte Seite in n Schritten, schreibt man entsprechend ein n über den Doppelpfeil.

Bildet man den transitiven Abschluß (transitive Hülle) der Relation, so bedeutet das die Ableitung über alle möglichen Schritte bis keine weiteren Ableitungen mehr möglich sind.

$$\begin{gathered} \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_m \\ \alpha_1 \overset{*}{\Rightarrow} \alpha_m \end{gathered} \tag{14.3}$$

Reflexivität gilt, da dies dem Anwenden keiner Produktion auf ein Nichtterminalsymbol entspricht. Es gilt somit immer

$$\alpha \overset{*}{\Rightarrow} \alpha \tag{14.4}$$

Mit dieser Relation als Hilfsmittel zur Beschreibung von Ableitungen läßt sich die von der Grammatik G generierte Sprache L(G) beschreiben

$$L(G) = \left\{ x \middle| x \in \Sigma^* \wedge S \overset{*}{\Rightarrow} x \right\} \tag{14.5}$$

Die Sprache L(G) besteht aus Zeichenketten, die nur Terminalsymbole ($x \in \Sigma^*$) enthalten und durch beliebig viele Ableitungsschritte ausgehend vom Startsymbol S erzeugt ($S \overset{*}{\Rightarrow} x$) werden. Hierzu drei Anmerkungen

1. Eine Sprache ist *kontextfrei*, wenn sie von einer *kontextfreien* Grammatik erzeugt wird.
2. Während Zeichenketten bestehend aus Terminalsymbolen (Alphabet) Wörter der Sprache genannt werden, heißen die Zwischenschritte der Ableitung bestehend aus Nichtterminal- und - wenn nötig - Terminalsymbolen *Satzform*.
3. Zwei Grammatiken G_1, G_2 sind äquivalent, wenn sie dieselbe Sprache generieren ($L(G_1) = L(G_2)$).

14.1.2 Ableitungsbäume

Ableitungen lassen sich mit Ableitungsbäumen (auch Parse-Bäume genannt) visualisieren. Jede kontextfreie Grammatik kann mit Ableitungsbäumen beschrieben werden. Die Knoten solcher Bäume sind Terminal- und Nichtterminalsymbole. Die Wurzel ist immer ein Nichtterminalsymbol, während Terminalsymbole nur in Blättern auftreten. Betrachtet man den gesamten Ableitungsbaum, ist die Wurzel das Startsymbol der Grammatik.

Teilbäume eines Ableitungsbaumes sind Bäume mit einem Nichtterminalsymbol A als Wurzel und allen sich daraus ergebenden Kinderknoten. Die Anzahl der Knoten ist kleiner als die des gesamten zugrundliegenden Baumes, erseidenn der Teilbaum ist gleich dem Ableitungsbaum. Solche Teilbäume werden nach ihrer Wurzel genannt. Heißt die Wurzel also A spricht man von einem A-Baum.

Damit läßt sich jede Produktion einer Grammatik als Teilbaum beginnend mit der linken Seite als Wurzel und den 1 bis n Symbolen der rechten Seite als Kinderknoten darstellen. Die Symbole der rechten Seite seien abgekürzt mit $X_1, X_2, \dots, X_m \quad mit \quad X_i \in (V \cup \Sigma)$

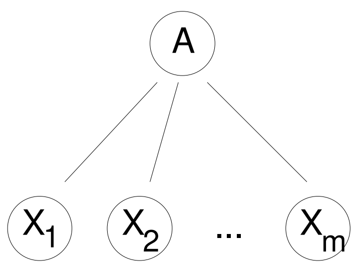


Abbildung 27 - Ableitungsbaum einer Produktion

Für einen Ableitungsbaum einer Grammatik $G = (V, \Sigma, P, S)$ gelten folgende Regeln

1. Jeder Knoten repräsentiert ein Symbol $X \in V \cup \Sigma \cup \{\varepsilon\}$.
2. Die Wurzel stellt das Startsymbol der Grammatik dar.
3. Alle inneren Knoten sind Nichtterminalsymbole, da kontexfreie Grammatiken nur solche auf der linken Seite der Produktionen stehen haben. Entsprechend sind alle Terminalsymbole Blätter, aber nicht notwendigerweise alle Blätter Terminalsymbole.
4. Wenn ein Knoten ein Nichtterminalsymbol A repräsentiert und die Kinderknoten X_1, X_2,...X_m hat, dann gibt es eine Produktion aus der Menge der Produktionen von G mit $A \to X_1 X_2 \dots X_m$ (siehe Grafik vorangegangener Absatz).
5. Knoten, die das leere Wort ε repräsentieren, sind immer Blatt, da sie das einzige Kind eines Knoten mit Nichtterminalsymbol darstellen (sogenannte ε-Produktionen, wie wir später sehen werde). Aus dem leeren Wort können keine Abbildungen erfolgen und es kann ausschließlich auf das leere Wort und nicht auf Konkatenationen mit dem leeren Wort abgebildet werden, da sonst die restlichen Symbole der rechten Seite und nicht das leere Wort die Kinderknoten darstellen würden.

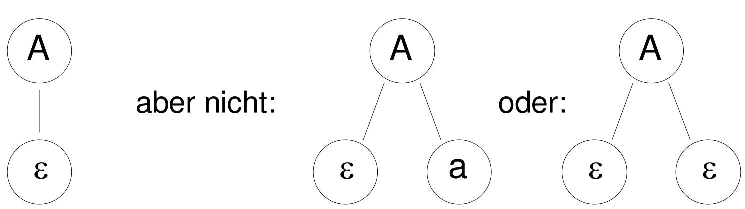


Abbildung 28 - richtige und falsche Ableitungsbäume von e-Produktionen

14.1.3 Beispiel für einen Ableitungsbaum

Gegeben sei eine Grammtik

$$
\begin{aligned}
G &= (\{S, A\}, \{a, b\}, P, S) \\
P &= \{(S \to aAS \mid a), (A \to SbA \mid ba \mid b)\}
\end{aligned}
\tag{14.6}
$$

Das erste und letzte Zeichen der Wörter der zugehörigen Sprache sind also immer a. In der Mitte können b und a in mehrfacher Ausführung auftreten. Für das Wort

$$aabbaa \in L(G) \tag{14.7}$$

kann man folgende Ableitung mit dem entsprechenden Ableitungsbaum angeben.

$$S \to aAS \to aSbAS \to aabbaa \tag{14.8}$$

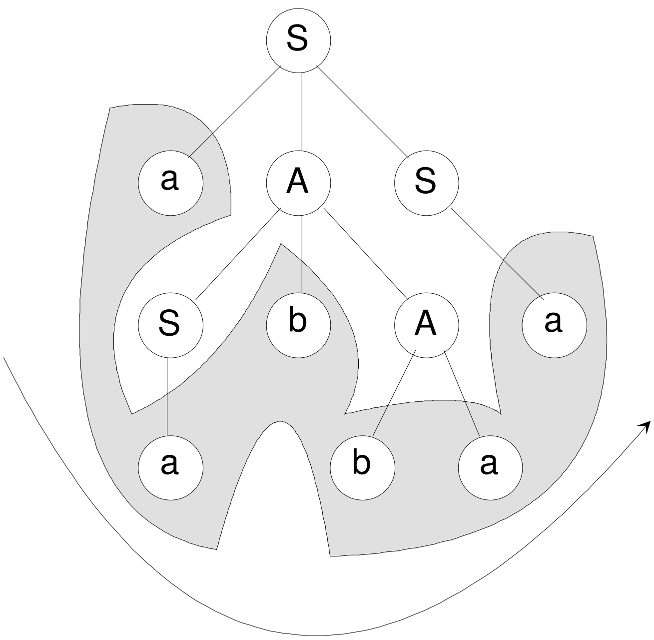


Abbildung 29 - Beispiel Ableitungsbaum für Wort aabbaa

Als „*yield*“ oder „*Front*“ bezeichnet man die Zeichenkette, die sich ergibt, wenn man die Blätter des Baumes perorder von links nach rechts abliest. In diesem Beispiel ist die Front das Wort aabbaa, es muß sich aber nicht ausschließlich um Terminalsymbole handeln, auch Nichtterminalsymbole sind je nach Grammatik möglich (bei Grammatiken, bei denen Blätter auch Nichtterminalsymbole sein können).

14.1.4 Die Beziehung zwischen Ableitungsbäumen und Grammatiken

Die Zusammenhänge zwischen Ableitungsbäumen und Ableitungen wurden im vorangegangenen Abschnitt nur exemplarisch und beschreibend dargelegt. Dies kann auch formal geschehen, wie im Folgenden gezeigt wird. Dazu wird wieder auf die Pfeilrelation zur Darstellung der Abbildungen und insbesondere auf die transitive Hülle dieser Relation zurückgegriffen.

Satz:

Für eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ gilt $S \overset{*}{\Rightarrow} \alpha$ genau dann, wenn ein Ableitungsbaum für G mit der Front (yield) α existiert.

$$\left(S \overset{*}{\Rightarrow} \alpha\right) \Leftrightarrow \exists \text{ Ableitungsbaum für } G \text{ mit Front } \alpha \tag{14.9}$$

Beweis:

Da eine ⇔ Beziehung vorliegt, muß in beide Richtungen bewiesen werden, was sich hier aber sehr ähnlich gestaltet. Zunächst die

„Hinrichtung“

Geht man von einem Baum aus und will damit die Ableitungsschritte zeigen, so beweist man per Induktion über die Anzahl der inneren Knoten. Anstatt des gesamten Ableitungsbaums vom Startsymbol S aus, wählen wir eine Ableitung der Form

$$A \overset{*}{\Rightarrow} \alpha \tag{14.10}$$

von einem beliebigen Nichtterminalsymbol A zur Front α aus.

Induktionsverankerung (Anzahl innerer Knoten = 1)

Es handelt sich um eine einzelne Produktion aus G. Aus dem inneren Knoten A folgen n verschiedene Terminal- und Nichtterminalsymbole $X_1,X_2,...,X_n$ als Front des Ableitungsbaumes. Die bereits weiter oben verwendete und hier nochmals abgebildete Grafik verdeutlich dies

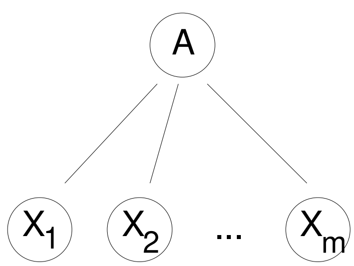


Abbildung 30 - Ableitung bei einem einzelnen inneren Knoten

Formal ausgedrückt ist der Satz für die Hinrichtung bei einem inneren Knoten bewiesen

$$\begin{aligned} &A \rightarrow X_1 X_2 \ldots X_n = \alpha \\ &A \overset{1}{\rightarrow} \alpha = A \overset{*}{\Rightarrow} \alpha \end{aligned} \tag{14.11}$$

Induktionsschritt:

Gelte die Behauptung nun für Teilbäume mit k - 1 inneren Knoten und sei α die Front eines A-Baums mit k inneren Knoten (k > 1). Die Kinderknoten eines solchen Wurzelementes A können nicht alle Blätter sein, da wegen k größer eins mindestens ein Kinderknoten innerer Knoten (bzw. Nichtterminalsymbol) ist.

Die Kinderknoten seien von 1 bis n durchnummeriert und mit $X_1,X_2,...,X_m$ bezeichnet. Dies entspricht wieder einer Produktion der Grammatik

$$A \rightarrow X_1 X_2 \ldots X_m \tag{14.12}$$

Wegen k größer eins ist eines der X_i Nichtterminalsymbol. Von diesen nichtterminalen X_i verzweigen die verbleibenden k - 1 inneren Knoten.

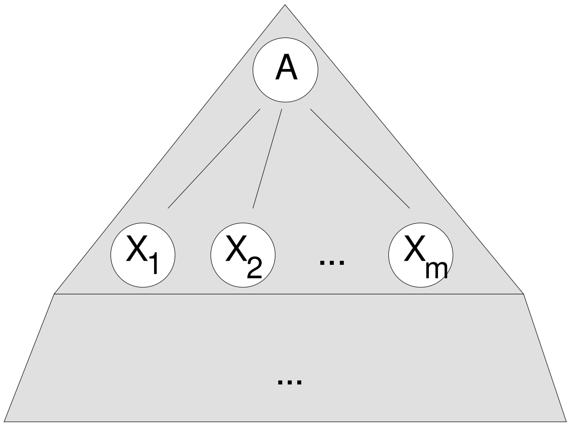


Abbildung 31 - Illustration eines A-Baum

Wenn X_i kein Blatt ist, dann ist X_i also Nichtterminalsymbol und Wurzel eines Unterbaums. Dieser X_i-Baum habe die Front α_i. Ist ein X_i allerdings Blatt und somit Nichtterminalsymbol, dann sei $X_i = \alpha_i$. Für j < i sind ein X_j sowie etwaige Kinderknoten in Front des A-Baums weiter links von einem X_i mit seinen Kinderknoten. Dann setzt sich die Front α des A-Baums folgendermaßen zusammen

$$\alpha = \alpha_2\alpha_1 \ldots \alpha_m \tag{14.13}$$

Die Grafik illustriert das mit der Darstellung von X_j- und X_i-Teilbäume nochmal

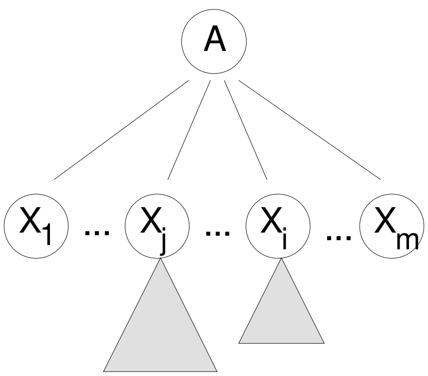


Abbildung 32 - A-Baum mit Kinderknoten und Xj-/Xi-Teilbäumen

Da A-Bäume k innere Knoten enthalten, ist die maximale Anzahl von inneren Knoten bei X_i-Bäumen entsprechend k - 1. Ausnahme ist natürlich der Fall, dass ein X_i-Baum genau der A-Baum ist, was aber hier unbeachtet bleiben soll. Gemäß Induktionsannahme, dass der Satz in Hinrichtung für Bäume mit inneren Knoten k < 1 bereits bewiesen sei, folgt

$$X_i \overset{*}{\Rightarrow} \alpha_i \qquad \forall i \mid X_i \neq Blatt \tag{14.14}$$

Faßt man diese Teilableitungen für jedes X_i zusammen, ergibt sich für die Front des A-Baums

$$A \overset{1}{\Rightarrow} X_1 X_2 \ldots X_m \overset{*}{\Rightarrow} \alpha_1 X_2 \ldots X_m \overset{*}{\Rightarrow} \alpha_1\alpha_2 X_3 \ldots X_m \overset{*}{\Rightarrow} \ldots \overset{*}{\Rightarrow} \alpha_1\alpha_2 \ldots \alpha_m \tag{14.15}$$

Damit gilt $A \overset{*}{\Rightarrow} \alpha$. Wenn dies für alle Nichtterminalsymbole A gilt, dann auch für das Startsymbol S. Allerdings handelt es sich bei der durchgeführten Ab $G = (V, \Sigma, P, S)$ leitung nur um eine mögliche Ableitung des Ableitungsbaumes.

Die Rückrichtung des Beweises erfolgt analog. Hier wird von einem gegebenen $A \overset{*}{\Rightarrow} \alpha$ auf einen Ableitungsbaum geschlossen. Da sowohl Hin- als auch Rückrichtung in der Vorlesung ausgespart wurden, wird an dieser Stelle auf eine weitere Darlegung verzichtet, zumal die Ähnlichkeit von Rückrichtung und Hinrichtung sehr hoch ist. Stattdessen gehen wir lieber zum nächsten Thema.

14.1.5 Linksableitung

Werden in einer Ableitung die Produktionen immer zuerst auf die am weitesten links stehenden Nichtterminalsymbole angewandt, dann spricht man von einer Links- oder leftmost-Ableitung. Wendet man die Produktionen hingegen immer auf die rechte Seite an, so spricht man von Rechts- bzw. rightmost-Ableitungen.

Definition:

Für eine kontextfreie Grammatik ist eine Linksableitung (leftmost)

$$x \underset{lm}{\overset{1}{\Rightarrow}} y \quad \Leftrightarrow \quad x = uA\alpha \quad \wedge \quad y = u\beta\alpha \qquad u \in \Sigma^*, a \in (V-\Sigma)^*, A \in V, (A \to \beta) \in P \tag{14.16}$$

Die formale Definition sagt aus, dass in einem Ableitungsschritt (mit der Anwendung einer Produktion) immer das am weitesten links stehendene Terminalsymbol (das ist hier das A, gefolgt von einer belieben Kombination α aus Nichtterminal- und Terminalsymbolen) zuerst abgeleitet wird. Daraus folgen Ableitungen in n und beliebig vielen Schritten (transitive und reflexive Hülle).

$$\begin{aligned} &x \underset{lm}{\overset{n}{\Rightarrow}} y \quad \text{Ableitung in n Schritten} \\ &x \underset{lm}{\overset{*}{\Rightarrow}} y \quad \text{transitive Hülle} \\ &x \underset{lm}{\overset{+}{\Rightarrow}} y \quad \text{reflexive Hülle} \end{aligned} \tag{14.17}$$

14.1.6 Eindeutigkeit

Wenn es eine Ableitung

$$S \overset{1}{\Rightarrow} x_1 \overset{1}{\Rightarrow} x_2 \overset{1}{\Rightarrow} x_3 \overset{1}{\Rightarrow} \ldots \overset{1}{\Rightarrow} x_n \tag{14.18}$$

gibt, dann existiert genau ein entsprechender Ableitungsbaum. Existiert umgekehrt ein Ableitungsbaum B mit der Front α (wie in den vorangegangenen Ausführungen mehrfach aufgetreten), dann gibt es im allgemeinen mehrere Ableitungen, die diesem Baum entsprechen. Allerdings gibt es nur genau eine Linksableitung für diesen Ableitungsbaum B.

Damit lassen sich *Eindeutigkeit* und *Mehrdeutigkeit* von Grammatiken definieren.

Definition:

Für eine kontextfreie Grammatik G gilt

$$G \text{ eindeutig} \Leftrightarrow \forall x \in L(G): x \text{ hat nur einen Ableitungsbaum} \tag{14.19}$$

und analog

$$G \text{ eindeutig} \Leftrightarrow \forall x \in L(G): x \text{ hat nur eine Linksableitung} \tag{14.20}$$

Weiter ist eine kontextfreie Sprache L *eindeutig*, wenn es eine eindeutige kontextfreie Grammatik G gibt, für die L = L(G) gilt.

Umgekehrt heißt eine kontextfreie Sprache L inhärent mehrdeutig, wenn keine eindeutige kontextfreie Grammatik G mit L = L(G) existiert.

14.2 Vereinfachung kontextfreier Grammatiken

Grammatiken für kontextfreie Sprachen können überflüssige Produktionen und Nichtterminalsymbole enthalten. Mit verschiedenen Verfahren u. a. auch dem Umstellen auf Normalformen können Grammatiken auf einen einheitlichen und leichter zu untersuchenden Stand gebracht werden. In diesem Protokoll soll es um Entfernung von ε-Produktionen gehen. Das sind Produktionen, bei denen auf der rechten Seite das leere Wort steht.

14.2.1 ε-Produktionen

Solche Löschmöglichkeiten (auch Deletion genannt), sind für die Definition einer Sprache durch eine Grammatik nicht nötig. Eine Ausnahme muß gegeben sein, wenn die Sprache das leere Wort enthält. Dann kann das Startsymbol ins leere Wort übergehen. Enthält die Sprache das leere Wort nicht, so ist auch diese Produktion nicht nötig. Wir werden mit dieser Vorstellung zeigen, dass es zu beliebigen kontextfreien Sprachen L $L \subseteq \Sigma^*$ eine kontextfreie Grammatik gibt, die nur eine oder keine ε-Produktionen enthält.

Definition:

Zu einer kontextfreien Sprache $L \subseteq \Sigma^*$ existiert eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit den Eigenschaften

1. $L = L(G)$ (14.21)

2. $\forall A \in V - \{S\} : (A \to \varepsilon) \notin P$ (14.22)

3. $S \to \varepsilon \Leftrightarrow \varepsilon \in L$ (14.23)

4. S erscheint auf keiner rechten Seite einer Produktion (14.24)

Die Sprache L wird also von der Grammatik G erzeugt und verzichtet dabei auf ε-Produktionen. Lediglich der Übergang $S \to \varepsilon$ wird zugelassen, wenn das leere Wort Bestandteil von L ist.

Beweis:

Für jede kontextfreie Sprache L existiert eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit L = L(G').

Um auf die Sprache G zu gelangen, sind zwei Schritte zu gehen. Zunächst werden alle Nichtterminalsymbole aussortiert, die direkt oder über mehrere Produktionen zum leeren Wort ε führen.

$$\begin{aligned} &V_1 = \{A | (A \to \varepsilon) \in P'\} \\ &V_2 = \{A | (A \to \alpha) \in P' \wedge \alpha \in V_1^*\} \\ &\dots \\ &V_n = \{A | (A \to \alpha) \in P' \wedge \alpha \in V_{n-1}^*\} \end{aligned} \tag{14.25}$$

Damit werden schrittweise alle Nichtterminalsymbole in V_i übernommen, die direkt oder über andere V_j mit j < i zum leeren Wort führen. Da die Anzahl der Symbole in V' nicht unendlich ist, sind nach N Läufen alle Nichtterminalsymbole gefunden, die direkt oder direkt aber nach maximal N Ableitungen zum leeren Wort kommen.

$$\begin{aligned} V_N &= V_N - 1 \\ V_{N+k} &= V_N \quad \forall k \end{aligned} \tag{14.26}$$

V_n sind dann die löschbaren Nichtterminalsymbole, d. h. Symbole, die mit endlich vielen Ableitungsschritten ins leere Wort ε überführt werden können. Das wird in bekannter Notation festgehalten:

$$A \overset{*}{\Rightarrow} \varepsilon \Leftrightarrow A \in V_N \tag{14.27}$$

Daraus folgt

$$\begin{aligned} & S \overset{*}{\Rightarrow} \varepsilon \Leftrightarrow S \in V_N \\ & \varepsilon \in L \Leftrightarrow S' \in V_N \end{aligned} \tag{14.28}$$

Nachdem die löschbaren Nichtterminalsymbole gefunden wurden, werden im zweiten Schritt die Produktionen auf Basis der verbliebenen Symbole umgebaut.

$$P_1 = P' \cup \left\{ \begin{aligned} & A \to x_1 Y_1 x_2 Y_2 \ldots x_n Y_n x_{n+1} \mid x_i \in ((V' - V_n) \cup \Sigma)^*, \\ & A \to x_1 A_1 x_2 A_2 \ldots x_n A_{n+1} \in P', Y_i \in \{A_i, \varepsilon\}, A_i \in V_n \end{aligned} \right\} \tag{14.29}$$

$$P_2 = P_1 - \{A \to \varepsilon \mid (A \to \varepsilon) \in P_1\} \tag{14.30}$$

$$P = P_2 \cup \{S \to S'\} \cup \{S' \to \varepsilon \mid \varepsilon \in L\} \tag{14.31}$$

Daraus wird anschaulich klar, dass die Sprache L mit der so entworfenen Grammatik $G = (V' \cup \{S\}, \Sigma, P, S)$ generiert werden kann und die Grammatik über die in der Definition vorgegebenen Eingenschaften verfügt.

15 Vorlesung vom 30. Mai 2000

15.1 Reduktion von Grammatiken

15.1.1 Definition Brauchbarkeit von Nichtterminalsymbolen

Sei $G = (V, \Sigma, Y, S)$ eine kontextfreie Grammatik. Dann ist ein Nichtterminalsymbol $A \in V - \Sigma$ definitionsgemäß dann nicht-überflüssig (brauchbar) wenn damit ein Wort generiert werden kann, und es vom Startsymbol aus erreichbar ist.

Formal bedeutet das:

$$\exists w \in \Sigma^* : A \overset{*}{\Rightarrow} w \tag{15.1}$$

und auch:

$$\exists \mathrm{x,y} \in \mathrm{V}^* : S \overset{*}{\Rightarrow} xAy \tag{15.2}$$

Die Umkehrung gilt dann natürlich auch. Ein Nichtterminalsymbol A heißt dann überflüssig, wenn A nicht (nicht-überflüssig ist). Also genau dann wenn die oben erwähnten zwei Bedingungen nicht zutreffen.

Satz:

Für jede kontextfreie Sprache L (welche von einer kontextfreien Grammatik erzeugt wurde) gibt es eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit:

1. $L = L(G)$ und
2. G hat keine überflüssigen Symbole

Diese Grammatik kann man auch eine „*reduzierte Grammatik*" nennen.

Beweis:

Sei nun die Sprache L kontextfrei. Dann existierte eine kontextfreie Grammatik $G' = (V', \Sigma, P', S)$ mit $L = L(G')$. Der Beweis erfolgt in zwei Schritten. Dabei ist zu beachten, dass es wichtig ist, die Reihenfolge der beiden Schritte genau einzuhalten. Aber dazu später:

- Schritt1: Man bilde die Menge V_1 indem man all die Nichtterminalsymbole auswählt, welche direkt ein Terminalsymbol generieren können. Rekursiv fährt man dann für V_n fort all die Nichtterminalsymbole in die Menge aufzunehmen, die Nichtterminalsymbole enthalten, die Terminalsymbole erzeugen. Da die Zeichenmenge endlich ist muss es ein N geben, für das die zwei so entstandenen Mengen $V_N = V_{N-1}$ gleich sind. Alle weiteren Mengen V_{K+N} sind dann auch gleich. Formal ausgedrückt bedeutet dies:

$$V_1 =_{Df} \left\{ A \middle| A \to x \in P', x \in \Sigma^* \right\} \tag{15.3}$$

$$V_n =_{Df} V_{n-1} \cup \left\{ A \middle| A \to \alpha \in P', \alpha \in \left(V_{n-1} \cup \Sigma\right)^* \right\} \tag{15.4}$$

$$\exists N : V_N = V_{N-1} \text{ und es gilt } V_{N+K} = V_N, \forall K \tag{15.5}$$

Nun betrachtet man alle Symbole in V_N. Man sieht, dass sie alle Terminalsymbole generieren. Die Umkehrung gilt auch, wird ein Terminalsymbol generiert, so ist das dafür verantwortliche Nichtterminalsymbol in V_N. Die Sprache ist genau dann leer, wenn dass Startsymbol nicht in V_N ist. Man definiert nun die Grammatik $\overline{G} =_{Df} \left(\overline{V}, \Sigma, \overline{P}, S\right)$. Hierbei ist $\overline{V}$ beschränkt auf

Nichtterminalsymbole, welche in Terminalsymbole überführt werden können und $\overline{P}$ ist die Menge der Produktionen, welche Terminalsymbole produzieren.

Formal:

$$\begin{aligned}
&\exists w \in \Sigma^* \text{ mit } A \overset{*}{\Rightarrow} w \Leftrightarrow A \in V_N \\
&L = \emptyset \Leftrightarrow S \notin V_N \\
&\overline{G} =_{Df} \left(\overline{V}, \Sigma, \overline{P}, S\right) \text{ mit} \\
&\overline{V} =_{Df} V' \cap \left(V_N \cup \Sigma\right) = V_N \cup \Sigma \\
&\overline{P} =_{Df} P' \cap \left(\overline{V} \times \overline{V}^*\right)
\end{aligned} \tag{15.6}$$

Somit haben wir nun nur noch Nichtterminalsymbole, welche in Terminalsymbole überführt werden können.

- Schritt 2: Nun gilt zu überprüfen welche Symbole eigentlich erreichbar sind. Dazu betrachtet man alle Nichtterminalsymbole A, welche auf der Rechten Seite einer Produktion auftauchen. Man definiert rekursiv die Übernahme aller Nichtterminalsymbole A, welche über ein weiteres Nichtterminalsymbol B erreichbar sind, das wiederum vom Startsymbol erreichbar ist. Auf diese Weise bekommt man alle Symbole, welche in Terminalsymbole überführt werden können und die vom Startzustand aus erreichbar sind. Ist dann einmal Gleichheit von zwei aufainander folgenden Mengen H_i und H_{i-1} erreicht, so bleibt diese Gleichheit bestehen. Dieser Fall muss auftreten, da der Symbolvorrat endlich ist.

  Formal drückt man das so aus:

$$H_0 =_{Df} \left\{S \middle| S \in \overline{V}\right\} \tag{15.7}$$

$$H_1 =_{Df} \left\{A \middle| S \to \alpha A \beta \in \overline{P}, \alpha, \beta \in \overline{V}^*\right\} \cup H_0 \tag{15.8}$$

$$H_n =_{Df} \left\{A \middle| B \to \alpha A \beta \in \overline{P}, B \in H_{n-1}, \alpha, \beta \in \overline{V}^*\right\} \cup H_{n-1} \tag{15.9}$$

$$\exists N : H_N = H_{N-1} \quad \Rightarrow \quad H_{N+K} = H_N \quad \forall K \tag{15.10}$$

  Ein Nichtterminalsymbol A ist also dann enthalten, wenn es erreichbar ist. Wieder gilt auch die Umkehrung. Ist ein Element erreichbar, so ist es auch in der Menge enthalten. Schließlich ist V um all die Nichtterminalsymbole reduziert, die nicht erreichbar sind. Ebenso werden die Produktionen aus P genommen, welche nicht verwendet werden können.

  Formal:

$$\begin{aligned}
&A \in H_N \Leftrightarrow S \overset{*}{\Rightarrow} \alpha A \beta, \alpha, \beta \in \overline{V}^* \\
&V =_{Df} \overline{V} \cap \left(H_N \cup \Sigma\right) = H_N \cup \Sigma \\
&P =_{Df} \overline{P} \cap \left(V \times V^*\right)
\end{aligned} \tag{15.11}$$

15.1.2 Das Problem der Ausführung der beiden Schritte

Kann nun Schritt 2 das Ergebnis von Schritt 1 kaputtmachen? Angenommen es gäbe kein Wort w, welches von A erzeugt wird. Dann muss aufgrund von Schritt 1 ein Symbol existieren, das Abgeleitet werden könnte, jedoch aufgrund von Schritt 2 entfernt worden ist. Die Produktion war also nicht erreichbar. Formal:

angenommen $A \overset{*}{\nRightarrow} w, \forall w$

dann gilt nach Schritt 1 : $A \overset{*}{\Rightarrow} \alpha B \beta \Rightarrow \alpha \gamma \beta \overset{*}{\Rightarrow} w$

aufgrund von Schritt 2 : $\{B, \gamma\} \not\subseteq (H_N \cup \Sigma)^*$

Da aber A erreichbar ist, ist auch B und alle Nichtterminalsymbole in γ erreichbar.

Die Antwort darauf ist also *Nein*, jedoch ergibt sich ein Problem wenn man Schritt 2 vor Schritt 1 ausführt. Dies wird durch das folgende Beispiel deutlich:

Beispiel:

$S \to AD$
$S \to D$
$A \to BC$
$B \to bB$
$C \to cC$
$C \to c$
$D \to d$
$F \to f$

Für nebenstehende Grammatik ergibt sich durch den ersten Schritt zunächst:

$V_1 = \{C, D, F\}$

Hier wurden also die Nichtterminalsymbole genommen, die direkt in Terminalsymbolen enden.

$V_2 = \{C, D, F\} \cup \{S\} = \{S, C, D, F\}$

Nun wurde das Startsymbol aufgrund der Produktion $S \to D$ übernommen.

$V_3 = \{S, C, D, F\} = V_2 = V_N$

Die weitere Betrachtung zeigt, dass keine weiteren Nichtterminalsymbole übernommen werden müssen also folgt:

$\overline{V} = \{S, C, D, F, b, c, d, f\}$

$\overline{P}$: $S \to D$
$C \to cC$
$C \to c$
$D \to d$
$F \to f$

Mit $\overline{P}$ wie nebenstehend definiert, kann man die Grammatik $\overline{G} = (\overline{V}, \Sigma, \overline{P}, S)$ aufstellen.

Für den zweiten Schritt ergibt sich dann:

$$H_0 = \{S\}$$
$$H_1 = \{S\} \cup \{D\}$$
$$H_2 = \{S, D\} = H_1 = H_N$$

P: $S \to D$
$D \to d$

Dabei wurde jeweils nach der Erreichbarkeit der Symbole gesehen. Zuerst wird das Startsymbol eingeschlossen, das einzige von dort aus zu erreichende Symbol ist dann D gewesen.

Was passiert nun, wenn man diese Reihenfolge vertauscht?

$S \to AD$
$S \to D$
$A \to BC$
$B \to bB$
$C \to cC$
$C \to c$
$D \to d$

Durch die Anwendung des zweiten Schrittes zuerst ergibt sich:

$$H_0 = \{S\}$$
$$H_1 = \{S\} \cup \{A, D\}$$
$$H_2 = \{S, A, D\} \cup \{B, C\}$$
$$H_3 = \{S, A, D, B, C\} = H_2 = H_N$$

Dabei wurde jeweils nach der Erreichbarkeit der Symbole gesehen. Zuerst wird das Startsymbol eingeschlossen, dann übernimmt man die Symbole A, D und schließlich noch B, C.

Wendet man nun den ersten Schritt an, so übernimmt man zuerst C, D und kommt zu:

$\bar{P}$: $S \to D$
$C \to cC$
$C \to c$
$D \to d$

$$V_1 = \{C, D\}$$

nun nimmt man noch das Startsymbol hinzu, da gilt $S \to D$:

$$V_2 = \{C, D\} \cup \{S\} = \{S, C, D\}$$

Für alle folgenden Iterationen gilt:

$$V_3 = \{S, C, D\} = V_2 = V_N$$

Man bekommt also ein anderes Ergebnis. Die überflüssigen (nicht erreichbaren) Produktionen $C \to cC$ und $C \to c$ wurden mit übernommen. Der Grund dafür liegt hier bei der Produktion $A \to BC$ hier kann B nicht in ein Terminalsymbol überführt werden. C hingegen kann in ein Terminalsymbol überführt werden. Es ist also wichtig die Reihenfolge einzuhalten.

Wie kann man die Grammatik nun noch weiter minimieren?

15.1.3 Zyklische Produktionen

Sei nun G eine Grammatik. Dann heißt $A \to B \in P$ eine zyklische Produktion. Da wir eine kontextfreie Grammatik gegeben haben, kann man zyklische Produktionen herausschneiden. Die Ableitungsschritte der zyklischen Produktionen müssen dann jedoch simuliert werden.

Satz:

Sei $L \subseteq \Sigma^*$ eine kontextfreie Sprache. Dann gibt es eine kontextfreie Grammatik G mit:

1. $L = L(G)$
2. $\forall A, B \in V - \Sigma : A \to B \notin P$

Beweis:

Sei $G' = (V', \Sigma, P', S)$ eine kontextfreie Grammatik mit $L(G') = L$. Dann gilt für alle $A \in V - \Sigma$:

$$\begin{aligned} &V_A^{(1)} = \{B \mid A \to B \in P'\} \\ &V_A^{(n)} = V_A^{(n-1)} \cup \left\{C \mid B \to C \in P' \wedge B \in V_A^{(n-1)}\right\} \\ &\exists N : V_A^{(N)} = V_A^{(N+1)} \wedge k \geq 0 : V_A^{(N)} = V_A^{(N+k)} \end{aligned} \tag{15.12}$$

Als neue Produktionen gelten also die „alten“ Produktionsregeln, ohne zyklische Produktionen, jedoch erweitert um die Produktionen, die durch den Zyklus erreicht werden könnten.

$$P =_{Df} \left[P' - \left\{ A \to B \middle| A \to B \in P' \right\} \right] \cup \left\{ A \to \alpha \middle| \alpha \notin V - \Sigma \wedge B \in V_A^{(N)} \right\} \qquad (15.13)$$

Offensichtlich ist nun $L = L(G') = L(G)$ für $G = \{V', \Sigma, P, S\}$.

Beispiel:

$S \to AB$
$A \to C$
$B \to D$
$C \to c$
$D \to DD$
$D \to d$
$D \to E$
$E \to f$

$V_S^{(N)} = \varnothing$
$V_E^{(N)} = \varnothing$
$V_{\hat{A}}^{(1)} = \{C\}$
$V_A^{(2)} = \{C\} = V_A^{(N)}$
$V_B^{(1)} = \{D\}$
$V_B^{(2)} = \{D, E\} = V_B^{(N)}$
$V_D^{(1)} = \{E\} = V_D^{(N)}$

$P:$
$S \to AB$
$D \to DD$
$C \to c$
$D \to d$
$E \to f$
$A \to c$
$B \to d$
$B \to DD$
$B \to f$
$D \to f$

15.1.4 Zusammenfassung

Eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ heißt reduziert, genau dann wenn:

1. P hat keine Produktionen vom Typ $A \to B$ mit $A, B \in V - \Sigma$. Es gibt also keine zyklischen Produktionen.
2. Es gibt keine ε-Produktionen. Das Startsymbol kann nur dann gleich ε sein, wenn es in der Sprache ist.
   - S taucht auf keiner rechten Seite auf,
   - Alle Symbole $A \in (V - \Sigma) - \{S\}$ können keine Produktionen $A \to \varepsilon$ haben.
   - $S \to \varepsilon \in P \Leftrightarrow \varepsilon \in L(G)$
3. Jedes Nichtterminalsymbol ist brauchbar. G hat keine überflüssigen Symbole und Produktionen d.h.:
   - $\forall A \in V - \Sigma \exists w \in \Sigma^* : A \overset{*}{\Rightarrow} w$
   - $\forall A \exists \alpha : \alpha, \beta \in V^* : S \overset{*}{\Rightarrow} \alpha A \beta$

Satz:

Sei L kontextfrei. Dann existiert eine reduzierte kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit $L = L(G)$.

Beweis:

Sei L kontextfrei. Dann existiert eine kontextfreie Grammatik G' mit $L = L(G')$:

a) Man modifiziere G', um Eigenschaft 2) zu erhalten, und erhält G_1,

b) Man modifiziere G_1 zu G_2 mit Eigenschaft 1)

c) Man modifiziert schließlich G_2 zu G mit Eigenschaft 3)

G ist dann reduziert und es gilt weiterhin $L = L(G)$.

Bemerkung:

Es ist jedoch darauf zu achten die Reihenfolge zu beachten. Wie wir schon an dem Beispiel gesehen haben, kann die Ausführung in anderer Reihenfolge das Ergebnis verändern. Insbesondere bedeutet das für den letzten Beweis:

a) man muss A) vor B) ausführen, da A) Produktionen vom Typ $A \to B$ mit $A, B \in V - \Sigma$ einführen kann.

b) B) zerstört nicht die Eigenschaft 2)

c) C) zerstört nicht die Eigenschaften 1) und 2)

16 Vorlesung vom 6. Juni 2000

16.1 Chomsky-Normalform

Satz (Chomsky-Normalform):

Sei $G=(V,\Sigma,P,S)$ eine reduzierte kontextfreie Grammatik. Dann existiert eine reduzierte kontextfreie Grammatik $G'=(V',\Sigma,P',S)$ mit

1. $L(G)=L(G')$ (16.1)

2. P‘ enthält nur Produktionen vom Typ

$$\begin{array}{lll} A \to BC & \text{mit} & A,B,C \in V-\Sigma \\ A \to a & \text{mit} & A \in V-\Sigma, a \in \Sigma \\ S \to \varepsilon & \Leftrightarrow & \varepsilon \in L(G') \end{array} \tag{16.2}$$

Diese reduzierte Form, die nur noch einen binären Ableitungsbaum erlaubt, heißt auch „*Chomsky-Normalform*“.

Beweis

Sei $G=(V,\Sigma,P,S)$ reduziert. Man konstruiert *G‘* in mehreren Schritten. Ziel ist es, alle Produktionen, die auf der rechten Seite mehr als zwei Nichtterminale oder mehr als ein Terminal bzw. das leere Wort stehen haben, umzubauen. Hierzu bauen wir neue Produktionsregeln aus den alten (*P*) auf:

$$P_1 := \{A \to a \mid A \to a \in P\} \tag{16.3}$$

diese Menge nimmt also alle diejenigen Regeln aus *P* auf, die gemäß unserer Bedingung 2) ohnehin in *P‘* enthalten sein dürften. Wir definieren nun einen Homomorphismus *h()*, einerseits die Terminale auf neue Nichtterminale (in eckigen Klammern geschrieben) abbildet und andererseits die Nichtterminale ohne Änderung überträgt:

$$\begin{array}{ll} h(a) := [a] & \forall a \in \Sigma \\ h(A) := A & \forall A \in V-\Sigma \end{array} \tag{16.4}$$

Damit bauen wir eine neue Produktionsmenge auf, die alle Produktionen aufnimmt, die in *P* auf der rechten Seite mehr als zwei symbole (Terminale bzw. Nichtterminale) enthielten. Damit haben wir sämtliche Regeln aus *P* abgehandelt.

$$P_2 := \{A \to h(\alpha) \mid A \to \alpha \wedge |\alpha| \geq 2\} \tag{16.5}$$

Wir müssen jetzt P_1 und P_2 noch zusammenfassen und die neuen Nichtterminale in eckigen Klammern zurück auf die ursprünglichen Terminale abbilden:

$$P_3 := P_1 \cup P_2 \cup \{[a] \to a \mid a \in \Sigma\} \tag{16.6}$$

Wir erhalten nun also offensichtlich $G_3=(V_3,\Sigma,P_3,S)$ mit $L(G)=L(G_3)$. Dabei gilt:

$$V_3 = V \cup \{[a] \mid \forall a \in \Sigma\}. \tag{16.7}$$

Nun haben wir in P_2 noch Regeln enthalten, die mehrere Nichtterminale auf der rechten Seite stehen haben. Diese reduzieren wir nun durch Aufblähen unserer Produktionsregelmenge, indem wir jede Regel in mehrere aufsplitten. Dies wird durch das spätere Beispiel noch deutlicher.

Wir ersetzen also jede Produktion aus P_3 mit der Form $A \rightarrow B_1 B_2 \dots B_n$ durch

$$\begin{aligned}&A \rightarrow B_1 \langle B_2 B_3 \dots B_n \rangle \\ &\langle B_i B_{i+1} \dots B_n \rangle \rightarrow B_i \langle B_{i+1} \dots B_n \rangle \quad \forall 2 \le i \le n-2 \\ &\langle B_{n-1} B_n \rangle \rightarrow B_{n-1} B_n\end{aligned} \tag{16.8}$$

Dabei sind die $\langle B_i \dots B_n \rangle$ neue Nichtterminale, die *V'* hinzuzufügen sind. Somit erhält man $G' = (V', \Sigma, P', S)$ mit $L(G) = L(G')$.

Satz:

Sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik in Chomsky-Normalform. Dann gilt:

$$\begin{aligned}&\forall n \ge 1 : (|x| = n) \wedge (x \in L(G)) \\ &\Downarrow \\ &S \overset{2n-1}{\underset{G}{\Rightarrow}} x\end{aligned} \tag{16.9}$$

Die Anzahl der Schritte (*2n-1*) läßt sich aus dem folgenden Schaubild ersehen:

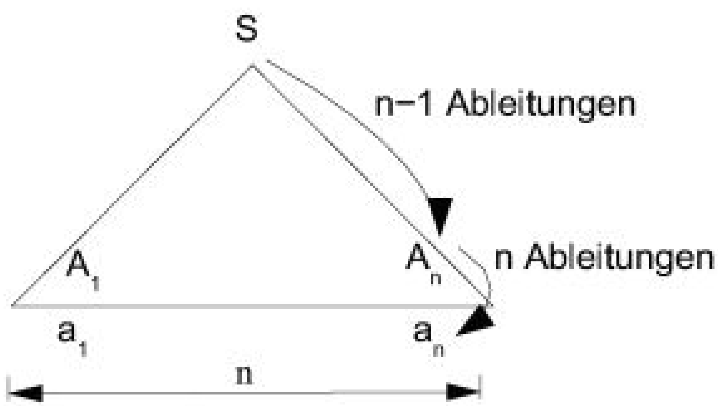


Abbildung 33

Beispiel

Wir wollen die Grammatik $G = (\{S, A\}, \{a, b, c\}, P, S)$ mit $P = \{S \rightarrow aAbb, S \rightarrow aAcb, A \rightarrow ab\}$ auf Chomsky-Normalform bringen und wenden hierzu die aus dem obigen Satz bekannten zwei Schritte an:

P	Schritt 1	Schritt 2	
$S \rightarrow aAbb$	$S \rightarrow [a]A[b][b]$	S	$\rightarrow [a]\langle A[b][b]\rangle$
		$\langle A[b][b]\rangle$	$\rightarrow A\langle [b][b]\rangle$
		$\langle [b][b]\rangle \rightarrow [b][b]$	
$S \rightarrow aAcb$	$S \rightarrow [a]A[c][b]$	S	$\rightarrow [a]\langle A[c][b]\rangle$
		$\langle A[c][b]\rangle$	$\rightarrow A\langle [c][b]\rangle$
		$\langle [c][b]\rangle \rightarrow [c][b]$	

P	Schritt 1	Schritt 2
$A \to ab$	$A \to [a][b]$	$A \to [a][b]$
	$[a] \to a$	$[a] \to a$
	$[b] \to b$	$[b] \to b$
	$[c] \to c$	$[c] \to c$

Damit haben wir eine Grammatik in Chomsky-Normalform mit den in Schritt 2 bezeichneten Produktionen und den Nichtterminalen:

$$V = \left\{ S, A, [a], [b], [c], \left\langle A[b][c] \right\rangle, \left\langle [b][b] \right\rangle, \left\langle A[c][b] \right\rangle, \left\langle [c][b] \right\rangle \right\} \tag{16.10}$$

17 Pushdown-Automaten

17.1 Einführung

Wir kommen nun zu dem Thema der „*Pushdown-Automaten*" („*Kellerautomaten*"). Wie wir später sehen werden, beschreiben kontextfreie Grammatiken und Pushdown-Automaten (*PDA*) dieselben Sprachen.

Im Wesentlichen ist ein PDA ein endlicher Automat, der sowohl über ein Band als auch über einen Stack (Keller) gesteuert wird. Weiter besitzt er eine endliche Kontrolle. Der Stack ist eine Symbolfolge über einem Alphabet. Wir stellen uns den Keller um 90° gekippt vor, so daß das erste Kellersymbol („Top") immer das am weitesten links stehende Symbol ist.

Der Automat kann auf zwei Arten arbeiten: Entweder liest er ein Symbol der Eingabe und kann in Abhängigkeit des aktuellen Zustandes und des obersten Elementes des Stacks den zustand wechseln. Das oberste Element des Stacks wird dann entfernt und eventuell durch ein neues Alphabetsymbol, einer Symbolfolge oder durch das leere Wort ersetzt. Sodann geht der Lesekopf eine Position auf dem Band nach rechts.

In der zweiten Arbeitsweise liest der Automat kein Symbol (eine sogenannte „ε-Bewegung ") vom Eingabeband (der Lesekopf bewegt sich also nicht). Genau wie oben können Zustand und/oder Stack verändert werden.

Bildlich kann man sich das ganze wie in Abbildung 34 vorstellen:

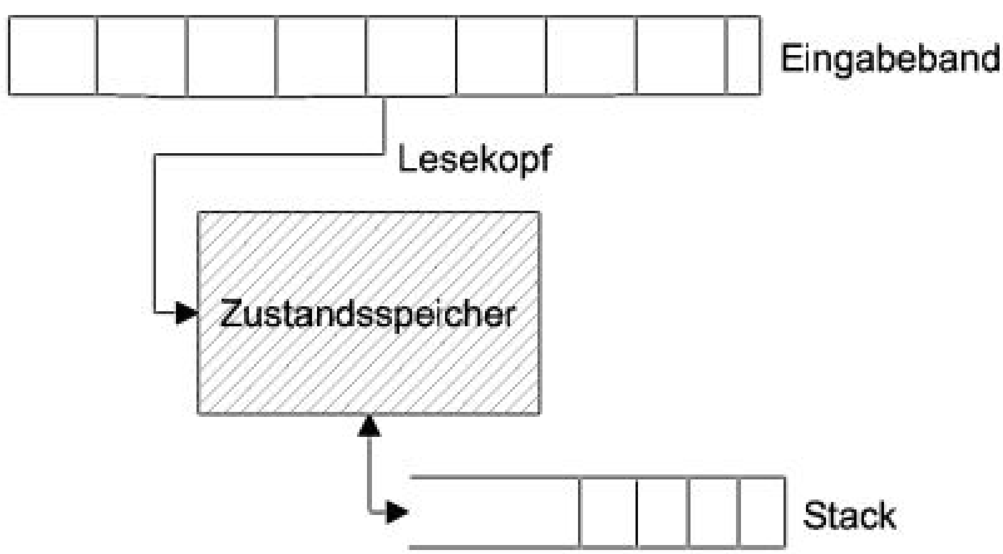


Abbildung 34

17.2 Formale Definition, Konfiguration, Akzeptierung

Definition (Pushdown-Automat)

Ein *Pushdown-Automat* (*PDA*, *Kellerautomat*) ist ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ mit:

1. Q ist eine endliche Menge von Zuständen
2. Σ ist eine endliche Menge von Eingabesymbolen
3. Γ ist eine endliche Menge von Kellersymbolen
4. $q_0 \in Q$ ist der Anfangszustand

5. $Z_0 \in \Gamma$ ist das Anfangskellersymbol (also das Symbol, mit dem der Keller initialisiert wird)
6. $F \subseteq Q$ ist die Menge der akzeptierenden Zustände
7. $\delta : Q \times \left(\Sigma \cup \{\varepsilon\}\right) \times \Gamma \to \left\{ A \middle| A \subseteq Q \times \Gamma^* \wedge A \text{ endlich} \right\}$

In Punkt 7 ist die Bedingung „*A* endlich" eingefügt, da $Q \times \Gamma^*$ unendlich sein könnte (wegen der Kleeneschen Hülle).

Wir können zwei Feststellungen machen:

1. Ein PDA ist in der Regel nicht deterministisch. Dies ergibt sich aus der Tatsache, daß δ eine Funktion ist, die auf eine Menge abbildet.
2. Die Stillstandsbewegung des „Lesekopfs" ergibt sich aus der Funktionsverwendung $\delta\left(q, \varepsilon, A\right)$. Dies bedeutet, dass das Eingabesymbol nicht gelesen wird und nur eine Zustandsveränderung und/oder Stackveränderung vorgenommen wird.

Definition (Konfiguration)

Sei $M = \left(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\right)$ ein PDA. Dann ist $c \in Q \times \Sigma^* \times \Gamma^*$ eine *Konfiguration* von *M*. Die Bedeutung einer solchen Konfiguration $c = \left(q, ax, A\alpha\right)$ ist:

- q augenblicklicher Zustand
- ax noch verbleibende Eingabe mit *a* als nächstem Symbol, $a \in \Sigma, x \in \Sigma^*$
- $A\alpha$ Kellerinhalt, mit *A* als oberstem Symbol, $A \in \Gamma, \alpha \in \Gamma^*$

Anmerkung: Eine Konfiguration kann man also als den Gesamtzustand eines PDAs ansehen. Sie beschreibt zu jedem Zeitpunkt genau den Zustand eines PDAs.

Definition

Sei $M = \left(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\right)$ ein PDA. Dann definieren wir die folgenden Operationen:

$$\left(q, ax, A\alpha\right) \overset{1}{\mapsto} \left(q', x, u\alpha\right) \Leftrightarrow \left(q', u\right) \in \delta\left(q, a, A\right) \tag{17.1}$$

$$\left(q, ax, A\alpha\right) \overset{1}{\mapsto} \left(q', ax, u\alpha\right) \Leftrightarrow \left(q', u\right) \in \delta\left(q, \varepsilon, A\right) \tag{17.2}$$

Seien *c* und *c'* zwei beliebige Konfigurationen. Dann gilt:

$$\begin{gathered} c \overset{n}{\mapsto} c' \quad \middle| n \geq 1 \\ \Updownarrow \\ \exists c_0 = c, c_1, c_2 \ldots, c_{n-1}, c_n = c' : \quad c_i \overset{1}{\mapsto} c_{i+1} \forall 0 \leq i \leq n-1 \end{gathered} \tag{17.3}$$

Für alle Konfigurationen *c* gilt:

$$c \overset{0}{\mapsto} c \tag{17.4}$$

$$c \overset{*}{\mapsto} c' \Leftrightarrow \exists n \geq 0 : \quad c \overset{n}{\mapsto} c' \tag{17.5}$$

Diese Definition bezieht sich also auf die Übergänge innerhalb des Automaten, die durch die Funktion δ realisiert werden und gibt uns eine einfachere Notation.

Die folgende Definition zeigt uns, dass wir drei Möglichkeiten haben, wie wir die Akzeptierung eines Wortes definieren. Es ist daher immer wichtig, zu wissen, von welcher Akzeptierungsart gesprochen wird.

Definition (Akzeptierungsarten)

Sei $M = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ ein PDA. Dann existieren drei Möglichkeiten der Akzeptierung:

1. Akzeptierung durch akzeptierende Zustände und leeren Stack:

$$T(M) := \left\{ x \middle| x \in \Sigma^* \quad \wedge \quad (q_0,x,Z_0) \overset{*}{\mapsto} (f,\varepsilon,\varepsilon) \quad \wedge \quad f \in F \right\} \tag{17.6}$$

2. Akzeptierung nur durch akzeptierende Zustände:

$$N(M) := \left\{ x \middle| x \in \Sigma^* \quad \wedge \quad (q_0,x,Z_0) \overset{*}{\mapsto} (f,\varepsilon,\alpha) \quad \wedge \quad f \in F, \alpha \in \Gamma^* \right\} \tag{17.7}$$

3. Akzeptierung nur durch leeren Stack:

$$L(M) := \left\{ x \middle| x \in \Sigma^* \quad \wedge \quad (q_0,x,Z_0) \overset{*}{\mapsto} (q,\varepsilon,\varepsilon) \quad \wedge \quad q \in Q \right\} \tag{17.8}$$

Beispiel eines PDA

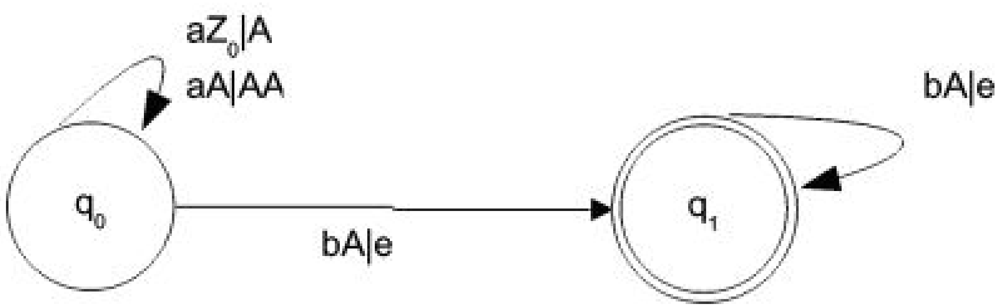


Abbildung 35

Dieser PDA stellt die Sprache $L(M) = \{a^n b^n | n \geq 1\} = T(M)$ dar. Dabei bedeutet $aZ_0 | A$, dass bei Lesen des Zeichens *a* und wenn Z_0 das oberste Element des Stacks ist, Z_0 vom Stack entfernt wird und durch *A* ersetzt wird.

Würde man die Akzeptierung vom Typ 2) annehmen, so gälte $N(M) = \{a^n b^m | n \geq 1, m \leq n\}$, da bei dieser Variante bereits akzeptiert wird, wenn nur ein akzeptierender Zustand vorliegt.

In diesem Beispiel realisiert der Stack also eine Art Zähler.

17.3 Äquivalenz von PDA und kontextfreier Grammatik

Satz (Äquivalenz Akzeptierungsarten):

Sei $L \subseteq \Sigma^*$ eine Sprache. Dann sind die folgenden Aussagen gleichwertig:

1. $L = T(M_1)$ für einen PDA M_1
2. $L = T(M_2)$ für einen PDA M_2
3. $L = T(M_3)$ für einen PDA M_3

Beweis

1)⇒2):

Sei $L = T(M_1)$ für $M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_0^1, Z_0^1, F_1)$. Wir definieren einen zweiten PDA:

$$M_2 := (Q_1 \cup \{p\}, \Sigma, \Gamma_1 \cup \{Y\}, \delta_2, q_0^1, Y, \{p\}) \tag{17.9}$$

mit $p \notin Q_1, Y \notin \Gamma_1$. Die Übergangsfunktion definieren wir so:

$$\begin{aligned} \delta_2(q_0^1, \varepsilon, Y) &:= \{(q_0^1, Z_0^1, Y)\} \\ \delta_2(f, \varepsilon, Y) &:= \{(p, \varepsilon)\} \ \forall f \in F_1 \\ \delta_2(q, a, A) &:= \delta_1(q, a, A) \qquad q \notin F_1, a \in (\Sigma_1 \cup \varepsilon) \end{aligned} \tag{17.10}$$

Man kann verifizieren, dass gilt:

$$(q_0^1, x, Z_0^1) \overset{*}{\underset{M_1}{\mapsto}} (f, \varepsilon, \varepsilon) \quad \Leftrightarrow \quad (q_0^1, x, Y) \overset{*}{\underset{M_2}{\mapsto}} (p, \varepsilon, \varepsilon) \tag{17.11}$$

und damit $T(M_1) = N(M_2)$.

Bildlich kann man sich die Konstruktion so vorstellen, dass zu Anfang ein „*Stoppersymbol*" *Y* auf den Stack gelegt wird, dann der alte Automat M_1 normal laufen gelassen wird. Wenn dieser ein Wort akzeptiert, dann ist der Stack normalerweise leer. In unserem Falle wird also noch *Y* auf dem Stack liegen. Wenn das der Fall ist, dann wird in den Zustand *p* verzweigt, der der einzige akzeptierende Zustand ist. Aufgrund dieses Zustandes akzeptiert nun der Automat M_2 aufgrund von $N(M_2)$ alleinig wegen des Zustandes.

2)⇒3):

Sei $L = N(M_2)$ für $M_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_0^2, Z_0^2, F_2)$. Wir definieren einen neuen PDA M_3 wie folgt:

$$M_3 := (Q_2 \cup \{p\}, \Sigma, \Gamma_3, \delta_3, q_0^2, Y, \varnothing) \tag{17.12}$$

mit $p \notin Q_2, Y \notin \Gamma_2$. Wir haben dann $\Gamma_3 := \Gamma_2 \cup \{Y\}$ und definieren die Übergangsfunktion so:

$$\begin{aligned} \delta_3(q_0^2, \varepsilon, Y) &:= \{(q_0^2, Z_0^2 Y)\} \\ \delta_3(f, \varepsilon, A) &:= \{(p, \varepsilon)\} \cup \delta_2(f, \varepsilon, A) \quad \forall A \in \Gamma_3, f \in F_2 \\ \delta_3(p, \varepsilon, A) &:= \{(p, \varepsilon)\} \qquad \forall A \in \Gamma_3 \\ \delta_3(q, a, A) &:= \delta_2(q, a, A) \qquad \text{sonst}, \forall a \in \Sigma \cup \{\varepsilon\} \end{aligned} \tag{17.13}$$

Bei der zweiten Zeile von (17.13) zeigt sich der Nichtdeterminismus. Hier steht $\{(p, \varepsilon)\}$ für den Start des Löschens des Stacks und $\delta_2(f, \varepsilon, A)$ für das Weiterarbeiten in der alten Maschine.

Die Funktionsweise ist die folgende: Solange wir in einem Zustand aus dem alten Automaten M_2 sind, arbeiten wir im alten Automaten (letzte Regel). Sind wir in einem akzeptierenden Zustand des alten Automaten, so löschen wir den Stack (zweite Regel erster Teil bzw. dritte Regel) oder arbeiten im alten Automaten weiter (zweite Regel zweiter Teil), da man auch in einem akzeptierenden Zustand im alten Automaten weiterarbeiten könnte ohne direkt abzubrechen.

Dies alles zeigt uns (was man auch formal beweisen könnte), dass gilt

$$(q_0^2, x, Z_0^2) \overset{*}{\underset{M_2}{\mapsto}} (f, \varepsilon, \alpha) \quad f \in F, \alpha \in \Gamma^* \quad \Leftrightarrow \quad (q_0^2, x, Y) \overset{*}{\underset{M_3}{\mapsto}} (p, \varepsilon, \varepsilon) \tag{17.14}$$

und damit $N(M_2) = L(M_3)$.

3)$\Rightarrow$1):

Sei $L = L(M_3)$ für $M_3 = (Q_3, \Sigma, \Gamma_3, \delta_3, q_0^3, Z_0^3, F_3)$. Dann definieren wir einfach einen Automaten, der die gesamte Zustandsmenge akzeptiert:

$$M_1 := (Q_3, \Sigma, \Gamma_3, \delta_3, q_0^3, Z_0^3, Q_3) \tag{17.15}$$

Damit gilt offensichtlich $L(M_3) = T(M_1)$, da der Automat M_1 mit der Definition von T nur dann ein Wort akzeptiert, wenn der Zustand akzeptiert wird und der Stack leer ist. Da M_3 aber nur dann akzeptiert, wenn der Stack leer ist und in M_1 alle Zustände akzeptieren, gilt dies bereits.

17.3.1 Äquivalenz PDA und kontextfreie Grammatiken

Satz (Äquivalenz PDA/ kontextfreie Grammatik):

Sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik in Chomsky-Normalform. Dann existiert ein PDA M, so dass $L(G) = L(M)$.

Beweis:

Wir definieren zunächst einen PDA und zeigen dann, dass das Gesagte gilt. Sei also

$$M := (\{q\}, \Sigma, V, \delta, q, S, \varnothing) \tag{17.16}$$

der PDA. Wir betrachten nun die Definition von $L(M) = \left\{ x \,\middle|\, x \in \Sigma^*, (q, x, S) \overset{*}{\mapsto} (q, \varepsilon, \varepsilon) \right\}$. Hieraus folgt die Definition der Übergangsfunktion:

$$\begin{aligned} \delta(q, \varepsilon, A) &:= \{(q, \alpha) \mid A \to \alpha \in P\} \quad & A \in (V - \Sigma) \\ \delta(q, a, a) &:= \{(q, \varepsilon)\} & a \in \Sigma \end{aligned} \tag{17.17}$$

Mit der ersten Regel können wir das jeweils oberste (linkeste) Stacksymbol durch Produktionen ersetzen und damit den Stack „aufblähen“ (nichtdeterministisch, wie man natürlich an der Mengenklammer sieht). Dabei handelt es sich um reine ε-Bewegungen, d.h. es wird kein Zeichen der Eingabe gelesen, sondern es werden lediglich sämtliche möglichen Produktionen des obersten Stacksymbols, das ein Nichtterminal ist, „durchprobiert“.

Die zweite Regel dient dazu, Terminale wieder vom Stack löschen zu können. Diese Löschen kann man sich wie ein „Matching“ vorstellen, d.h. ein Terminal wurde als „möglich“ erkannt und vom Stack gelöscht. Dabei wird der Lesekopf auf das nächste Eingabesymbol weitergeschoben.

Wir vollziehen den Beweis nun in zwei Schritten. Die beiden Schritte zusammen ergeben den Gesamtbeweis:

$$\begin{array}{llcl} 1. & \forall n \geq 1:\ A \overset{n}{\Rightarrow} w \quad (w \in \Sigma^*) & \Rightarrow & (q, w, A) \overset{*}{\mapsto} (q, \varepsilon, \varepsilon) \\ 2. & \forall n \geq 1:\ (q, w, A) \overset{n}{\mapsto} (q, \varepsilon, \varepsilon) & \Rightarrow & A \overset{*}{\Rightarrow} w \end{array} \tag{17.18}$$

1. und 2. zusammen ergeben die Aussage, wie wohl offensichtlich ist (1 ist die eine Richtung und 2 die andere).

Beweis von 1:

Wir beweisen per Induktion:

Induktionsverankerung (n=1):

Es sei $A\overset{1}{\Rightarrow}w$. Dann folgt daraus $A\rightarrow w\in P$ und $w=a\in\Sigma$ (also dass w ein einzelnes Eingabesymbol ist). Hieraus folgt mit der Definition von δ (Punkt 1)(siehe (17.17)): $(q,a)\in\delta(q,\varepsilon,a)$. Daraus wiederum folgt:

$$(q,a,A)\overset{1}{\mapsto}(q,a,a)\overset{1}{\mapsto}(q,\varepsilon,\varepsilon) \tag{17.19}$$

(der letzte Ableitungsschritt gilt wegen Punkt 2 der Definition von δ (siehe (17.17)).

Induktionsannahme n:

Für alle $m\leq n$ gelte die folgende Implikation:

$$A\overset{m}{\Rightarrow}w \quad w\in\Sigma^* \;\Rightarrow\qquad (q,w,A)\overset{*}{\mapsto}(q,\varepsilon,\varepsilon) \tag{17.20}$$

Induktionsschritt ($n\rightarrow n+1$):

Wir haben $A\overset{n+1}{\Rightarrow}w$. Dies ist äquivalent zu $A\overset{1}{\Rightarrow}BC\overset{n}{\Rightarrow}w \quad (n\geq 1)$. Die Einzelableitung gilt, weil wir angenommen haben, dass die kontextfreie Grammatik G in Chomsky-Normalform vorliegt. Mit Regel 1 der Definition von δ (siehe (17.17)) folgt:

$$(q,BC)\in\delta(q,\varepsilon,A) \tag{17.21}$$

also $(q,w,A)\overset{1}{\mapsto}(q,w,BC)$. Es gilt weiter $w=w_1w_2$ mit $B\overset{m_1}{\Rightarrow}w_1$ und $C\overset{m_2}{\Rightarrow}w_2$ mit $m_1,m_2\leq n$. Damit gilt nach zweimaliger Anwendung der Induktionsannahme

$$\begin{aligned}(q,w,BC) \quad &= (q,w_1w_2BC)\\ &\overset{*}{\mapsto}(q,w_2,C)\\ &\overset{*}{\mapsto}(q,\varepsilon,\varepsilon)\end{aligned} \tag{17.22}$$

Damit gilt insgesamt

$$\begin{aligned}(q,w,A) \quad &\overset{*}{\mapsto}(q,w,BC)\\ &\overset{*}{\mapsto}(q,\varepsilon,\varepsilon)\end{aligned} \tag{17.23}$$

und damit die Behauptung.

18 Vorlesung vom 8. Juni 2000

18.1 Für jede CFL existiert ein PDA

18.1.1 Fortsetzung des Beweises vom 06. Juni 2000

Der Beweis zur Äquivalenz von PDA und kontextfreier Grammatik wurde in der Vorlesung vom 06. Juni 2000 begonnen, und wird hier fortgeführt.

Es ist zu zeigen:

1: „Hin-Richtung":

$$\forall n \geq 1 \quad : \quad \left(A \overset{n}{\Rightarrow} w \right) \quad \Rightarrow \quad \left((q, w, A) \overset{*}{\mapsto} q, \varepsilon, \varepsilon \right) \tag{18.1}$$

2: „Rückrichtung":

$$\forall n \geq 1 \quad : \quad \left((q, w, A) \overset{n}{\mapsto} q, \varepsilon, \varepsilon \right) \quad \Rightarrow \quad \left(A \overset{*}{\Rightarrow} w \right) \tag{18.2}$$

Das Symbol „*A*" sei hierbei ein Nichtterminal der kontextfreien Grammatik.

Die „Hin-Richtung" wurde bereits gezeigt. Nun folgt der Beweis der „Rückrichtung", d.h. wir müssen zeigen, dass jedes Wort, welches unser Automat akzeptiert, auch durch die Grammatik erzeugt werden kann. Diesen Beweis erbringen wir durch vollständige Induktion über die Länge der Produktion *n*.

Wir erinnern uns: Der PDA wurde so konstruiert, dass die Terminalsymbole der kontextfreien Grammatik als Bandalphabet dienen, und sowohl die Terminalsymbole als auch die Nichtterminalsymbole der kontextfreien Grammatik als Stacksymbole dienen.

Induktionsverankerung (n=1):

Für $n=1$ gilt:

$$\begin{aligned} &(q, w, A) \overset{1}{\mapsto} (q, \varepsilon, \varepsilon) \\ &\Downarrow \\ &\delta(q, w, A) \ni (q, \varepsilon) \quad \wedge \quad |A| = 1 \end{aligned} \tag{18.3}$$

Bei der Definition unseres Kellerautomaten, dem ja eine kontextfreie Grammatik zugrunde liegt, haben wir einen Zustandsübergang genau dann hinzu genommen, wenn für diese kontextfreie Grammatik mindestens eine von zwei Bedingungen erfüllt ist:

$$\begin{aligned} &\vdots \\ &\Downarrow \\ &\left((w = \varepsilon) \wedge (A \rightarrow \varepsilon \in P) \right) \vee (w = A) \end{aligned} \tag{18.4}$$

Das bedeutet in geschriebener Sprache, dass entweder a) kein Symbol vom Band gelesen wird, und ein auf dem Stack liegendes „Nichtterminal"[13] durch das „Terminal" „leeres Wort" ersetzt wird, oder b) das vom Band gelesene „Terminal" dem (einzigen) Zeichen auf dem Stack entspricht, und der Stack daher leergeräumt wird.

Fall a) tritt definitionsgemäß (siehe Definition unseres PDA) genau dann ein, wenn in der kontextfreien Grammatik ein Übergang des Nichtterminales in das Terminal „leeres Wort" existiert. Da die kontextfreie Grammatik in Chromsky-Normalform ist, kann es sich nur um den Übergang des Startsymboles *S* in das leere Wort handeln. Unser Nichtterminalsymbol *A* muss dann also („von vorne

[13] Man kann bei einem PDA natürlich nicht von „Terminalen" und „Nichtterminalen" sprechen. Diese Bezeichnungen beruhen auf der Bedeutung der jeweiligen Symbole in Bezug auf die kontextfreie Grammatik, die dem PDA zugrunde liegt.

herein“) das Startsymbol S gewesen sein, und es muss eine Produktionsregel geben, die das Startsymbol S in das leere Wort übergehen lässt.

In diesem Fall handelt es sich um einen ε-Übergang:

$$\begin{aligned}&\vdots\\&\Downarrow\\&(A \to \varepsilon \in P) \wedge (w = \varepsilon)\\&\Downarrow \quad \text{Chromsky-Normalform}\\&(S \to \varepsilon \in P) \wedge (w = \varepsilon)\end{aligned} \tag{18.5}$$

Da es also diese Produktionsregel gibt, gilt:

$$S \overset{1}{\Rightarrow} \varepsilon \quad \text{bzw.} \quad S \overset{*}{\Rightarrow} \varepsilon \tag{18.6}$$

Damit ist für die Produktionslänge $n=1$ und den Fall a) gezeigt, dass jedes Wort, dass unser PDA akzeptiert, von der kontextfreien Grammatik erzeugt wird.

Fall b) ist noch einfacher. Fall b) kann nämlich gar nicht eintreten, da das Symbol auf dem Stack (A) ein „Nichtterminalsymbol“ ist. Demzufolge wird es auf dem Band nie das Zeichen A geben, und Fall b) tritt nie ein. Damit ist für die Produktionslänge $n=1$ (für Fall a) und Fall b)) gezeigt, dass jedes Wort, dass unser PDA akzeptiert, von der kontextfreien Grammatik erzeugt wird.

Induktionsannahme:

$$\forall m \leq n \quad : \quad \left((q,w,A) \overset{m}{\mapsto} q,\varepsilon,\varepsilon \right) \quad \Rightarrow \quad \left(A \overset{*}{\Rightarrow} w \right) \tag{18.7}$$

Induktionsschritt ($n \to n+1$):

$$(q,w,A) \overset{n+1}{\mapsto} (q,\varepsilon,\varepsilon) \tag{18.8}$$

Wir betrachten nun ausdrücklich nur Ableitungen, die eine Länge größer als zwei haben. Dies geschieht aber in Übereinstimmung mit dem Skript zur Vorlesung[14]. Der Fall, dass das „Nichtterminal“ A, das auf dem Stack liegt, im ersten Zustandsübergang durch ein Terminal ersetzt wird, kann nicht eintreten. Könnte er eintreten, wäre der Stack nach dem zweiten Zustandsübergang bereits leer – und damit die Verarbeitung des PDA beendet; das ist aber ein Widerspruch zu der Annahme, dass mindestens drei Zustandsübergänge erfolgen.

Daher gilt:

$$\begin{aligned}&(q,w,A) \overset{1}{\mapsto} (q,w,BC) \overset{n}{\mapsto} (q,\varepsilon,\varepsilon)\\&\Downarrow\\&\delta(q,\varepsilon,A) \ni (q,BC)\end{aligned} \tag{18.9}$$

Dieser Übergang des Stacksymboles A in die Stacksymbole B und C kann nur definiert worden sein, wenn die kontextfreie Grammatik die entsprechende Produktion enthält:

$$\begin{aligned}&\vdots\\&\Downarrow\\&A \to BC \in P\end{aligned} \tag{18.10}$$

Der Ausdruck (18.9) bedeutet nichts anderes, als dass das vorhandene „Nichtterminal“ auf dem Stack durch zwei andere „Nichtterminale“ ersetzt wird. Je nach Länge des Eingabewortes geschieht dies einige Male. Letzten Endes werden aber alle so auf dem Stack abgelegten „Nichtterminale“ wieder „abgebaut“, das bedeutet Stück für Stück durch Vergleichen mit dem Eingabewort durch das leere

[14] Dieses Vorgehen ist an sich beweistechnisch nicht korrekt.

Wort ersetzt – bis der Stack irgendwann leer ist. Insbesondere bedeutet das, dass „*C*" nicht vom Stack gelöscht werden kann, bevor nicht auch „*B*" vom Stack gelöscht wurde. Und wenn wir einen Schritt weiter denken, und davon ausgehen, dass sowohl *C* als auch *B* auf dem Stack durch mehrere andere „Nichtterminale" ersetzt werden, bedeutet das trotzdem, dass die „Nichtterminale", die aus *B* hervorgegangen sind, nicht gelöscht werden können, bevor nicht die „Nichtterminale" gelöscht wurden, die aus *C* hervorgegangen sind.

Daher lässt sich die Verarbeitung des Eingabewortes *w* wie folgt zerlegen:

$$\begin{aligned} \exists w_1, w_2 \in \Sigma^* \quad : \quad \left(w_1 w_2 = w\right) \qquad & \wedge \left(\left(q, w_1, B\right) \overset{m_1}{\mapsto} \left(q, \varepsilon, \varepsilon\right) \right) \wedge \\ & \wedge \left(\left(q, w_2, C\right) \overset{m_2}{\mapsto} \left(q, \varepsilon, \varepsilon\right) \right) \end{aligned} \tag{18.11}$$

mit $m_1, m_2 \leq n$ und $m_1 + m_2 = n$. Unter Zuhilfenahme der Induktionsannahme folgt daraus:

$$B \overset{*}{\Rightarrow} w_1 \quad \wedge \quad C \overset{*}{\Rightarrow} w_2 \tag{18.12}$$

Den Übergang von *A* nach *BC* haben wir selber in (18.9) definiert, daher gilt:

$$A \overset{*}{\Rightarrow} BC \overset{*}{\Rightarrow} w_1 C \overset{*}{\Rightarrow} w_1 w_2 \overset{*}{\Rightarrow} w \tag{18.13}$$

Damit ist der Induktionsbeweis abgeschlossen:

$$\left(\left(q, w, A\right) \overset{n+1}{\mapsto} \left(q, \varepsilon, \varepsilon\right) \right) \quad \Rightarrow \quad A \overset{*}{\Rightarrow} w \tag{18.14}$$

Gesamtbeweis:

Wenn wir nun in den (bewiesenen) Behauptungen (18.1) und (18.2) das allgemeine „Nichtterminal" *A* durch das spezielle Nichtterminal, das Startsymbol *S* ersetzen, erhalten wir:

$$\begin{aligned} & \forall n \geq 1 \quad : \quad \left(S \overset{n}{\Rightarrow} w \right) \quad \Rightarrow \quad \left(\left(q, w, S\right) \overset{*}{\mapsto} q, \varepsilon, \varepsilon \right), \\ & \forall n \geq 1 \quad : \quad \left(\left(q, w, S\right) \overset{n}{\mapsto} q, \varepsilon, \varepsilon \right) \quad \Rightarrow \quad \left(S \overset{*}{\Rightarrow} w \right) \\ & \Downarrow \\ & w \in L\left(M\right) = w \in L\left(G\right) \\ & \Updownarrow \\ & L\left(M\right) = L\left(G\right) \end{aligned} \tag{18.15}$$

19 Vorlesung vom 15. Juni 2000

19.1 Greibach Normalform (GNF)

19.1.1 Vorbemerkung

Neben der Chomsky Normalform ist auch die Greibach Normalform als weitere standardisierte Darstellungsform für kontextfreie Grammatiken bekannt. Hier sind alle Produktionen derart gestaltet, dass die rechte Seite immer mit einem Terminalsymbol beginnt und keines oder beliebig viele Nichtterminalsymbolen folgen.

$$A \to aB_1B_2...B_k \qquad k \geq 0 \tag{19.1}$$

Um zu zeigen, dass sich jede kontextfreie Grammatik für Sprachen ohne leeres Wort ε in Greibach Normalform bringen lässt, wie wir in den folgenden Abschnitten zeigen werden, muss es möglich sein, äquivalente Grammatiken aufzustellen, bei denen mehrere Ableitungsschritte in einzelnen Produktionen zusammengefasst werden. Dass solche Zusammenfassungen möglich sind, wird im Folgenden gezeigt.

Eine A-Produktion ist eine Produktion der Form

$$A \to \alpha \tag{19.2}$$

mit einem Nichtterminalsymbol A auf der linken Seite. Bei einer kontextfreien Grammatik $G = (V, \Sigma, P, S)$ gebe es eine A-Produktion $A \to \alpha_1 B a_2$ mit B-Produktionen

$$B \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_s \tag{19.3}$$

Dann gibt es eine aus G hervorgehende kontextfreie Grammatik $G_1 = (V, \Sigma, P_1, S)$ mit neuen Produktionen

$$A \to \alpha_1\beta_1\alpha_2 \mid \alpha_1\beta_2\alpha_2 \mid \ldots \mid \alpha_1\beta_s\alpha_2 \tag{19.4}$$

die durch Ableiten von $A \to \alpha_1 B a_2$ aus G mit $B \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_s$ entstehen. Die Produktion $A \to \alpha_1 B a_2$ ist dann überflüssig und fehlt in G_1. Dann gilt $L(G) = L(G_1)$.

Beweis:

Der Beweis ist einfach, da sich $L(G_1) \subseteq L(G)$ ergibt, wenn man für jede Ableitung

$$A \underset{G}{\Rightarrow} \alpha_1 B\alpha_2 \underset{G}{\Rightarrow} \alpha_1\beta\alpha_2 \tag{19.5}$$

in G_1 das entsprechende

$$A \underset{G_1}{\Rightarrow} \alpha_1\beta\alpha_2 \tag{19.6}$$

verwendet. Umgekehrt ist $L(G) \subseteq L(G_1)$, da $A \to \alpha_1 B a_2$ die einzige Produktion ist, die in G und nicht in G_1 enthalten ist. Immer, wenn $A \to \alpha_1 B a_2$ benötigt wird, folgt in einem späteren Ableitungsschritt $B \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_s$, was man in G1 in einem einzigen Ableitungsschritt mit $A \to \alpha_1\beta_1\alpha_2 \mid \alpha_1\beta_2\alpha_2 \mid \ldots \mid \alpha_1\beta_s\alpha_2$ realisiert wird. Folglich ist $L(G) = L(G_1)$.

19.1.2 Umwandlung von linksrekursiven in rechtsrekursive Produktionen

Mit der im vorangegangen Abschnitt vorgestellten Möglichkeit, Zusammenfassungen von Ableitungen vorzunehmen, kann man bereits viele Produktionen in Greibach Normalform überführen. Problematisch wird es allerdings bei linksrekursiven Produktionen, die Nichtterminalsymbole auf sich selbst und weitere Symbole abbilden, da diese nicht ohne Weiteres in die gewünschte Form zu bringen sind.

Solche Produktionen der Gestalt

$$A \to A\alpha \qquad A \in V, \alpha \in (V \cup \Sigma)^* \tag{19.7}$$

heißen linksrekursiv. Um sie kompatibel für die Greibach Normalform zu machen, kann man äquivalente rechtsrekursive Produktionen einführen. Gegeben sei eine Menge von A-Produktionen

$$\begin{aligned} &A \to A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_r \quad \alpha_i \in (V \cup \Sigma)^* \\ &A \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_s \qquad\qquad \beta_j \in ((V - A) \cup \Sigma) \cdot (V \cup \Sigma)^* \end{aligned} \tag{19.8}$$

wobei die oberen Produktionen linksrekursiv und die unteren nicht linkrekursiv sind. Damit können die Wörter abgeleitet werden, die durch den regulären Ausdruck

$$\underbrace{(\beta_1 \mid \beta_2 \mid \ldots \mid \beta_s)}_{\text{1 mal}}\underbrace{(\alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_r)}_{\text{n mal}}{}^* \tag{19.9}$$

beschrieben werden. Dabei handelt es sich um Wörter, die aus einem β_j an erster Stelle und beliebig vielen folgenden α_i bestehen. Diese Wörter lassen sich genauso durch Hinzufügen eines neuen Nichtterminalsymbols B mit Produktionen der folgenden Form erzeugen

$$\begin{aligned} &A \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_s \\ &A \to \beta_1 B \mid \beta_2 B \mid \ldots \mid \beta_s B \\ &B \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_r \\ &B \to \alpha_1 B \mid \alpha_2 B \mid \ldots \mid \alpha_r B \end{aligned} \tag{19.10}$$

Damit liegen dann keine linksrekursiven, sondern rechtsrekursive Abbildungen vor, die die gleichen Wörter generieren und die zugehörigen Grammatiken somit die gleiche Sprache beschreiben.

Formaler gesprochen, gibt es zu einer kontextfreien Grammatik $G = (V, \Sigma, P, S)$ mit linksrekursiven Produktionen eine entsprechende Grammatik $G_1 = ((V \cup B_k), \Sigma, P_1, S)$ mit rechtsrekursiven Produktionen. Dabei sind alle Produktionen identisch, außer den linksrekursiven, die durch rechtsrekursive ersetzt werden.

G habe nun Produktionen der Form

$$\begin{aligned} &A \to A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_r \quad \alpha_i \in (V \cup \Sigma)^* \\ &A \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_s \qquad\qquad \beta_j \in ((V - A) \cup \Sigma) \cdot (V \cup \Sigma)^* \end{aligned} \tag{19.11}$$

und G_1 entsprechend zusätzliche

$$\begin{aligned} &A \to \beta_1 B \mid \beta_2 B \mid \ldots \mid \beta_s B \\ &B \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_r \\ &B \to \alpha_1 B \mid \alpha_2 B \mid \ldots \mid \alpha_r B \end{aligned} \tag{19.12}$$

wobei die $A \to A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_r$ in G_1 nicht mehr auftauchen. Dann gilt $L(G) = L(G_1)$.

Die folgende Illustration verdeutlicht die Ableitungsschritte.

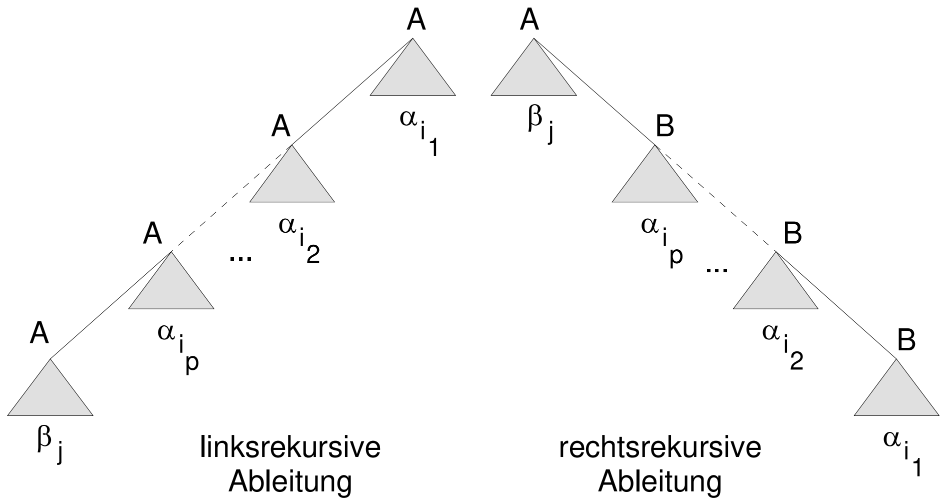


Abbildung 36 - Links- und rechtrekursive Ableitungen

Beweis:

In G endet jede Ableitung von $A \rightarrow A\alpha_i$ mit $A \rightarrow \beta_i$. Damit erhält man

$$A \underset{G}{\Rightarrow} A\alpha_{i_1} \underset{G}{\Rightarrow} A\alpha_{i_2}\alpha_{i_1} \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} A\alpha_{i_p}\alpha_{i_{p-1}} \ldots \alpha_{i_1} \underset{G}{\Rightarrow} \beta_j\alpha_{i_p}\alpha_{i_{p-1}} \ldots \alpha_{i_1} \tag{19.13}$$

was in G_1 folgendermaßen erzielt wird

$$A \underset{G_1}{\Rightarrow} \beta_j B \underset{G_1}{\Rightarrow} \beta_j\alpha_{i_p} B \underset{G_1}{\Rightarrow} \beta_j\alpha_{i_p}\alpha_{i_{p-1}} B \underset{G_1}{\Rightarrow} \ldots \underset{G_1}{\Rightarrow} \beta_j\alpha_{i_p}\alpha_{i_{p-1}} \ldots \alpha_{i_1} \tag{19.14}$$

Da beide Ableitungen zu den gleichen Wörtern führen, die dem oben genannten regulären Ausdruck entsprechen, gilt $L(G) = L(G_1)$. Somit wurde gezeigt, dass sich linksrekursiven Produktionen unter Zuhilfenahme neuer Nichtterminalsymbols B_k sowie weiterer Produktionen in rechtsrekursive umwandeln lassen und dabei die mit der Grammatik erzeugte Sprache $L(G)$ erhalten bleibt.

19.1.3 Kontextfreie Grammatiken in Greibach Normalform (GNF)

Satz:

Jede kontextfreie Sprache L ohne leeres Wort ε kann durch eine kontextfreie Grammatik G generiert werden, deren Produktionen die Form

$$A \rightarrow aB_1B_2...B_k \qquad k \geq 0 \tag{19.15}$$

haben. A und B_i sind Nichtterminalsymbole, a ist ein Terminalsymbol. Sind alle Produktionen von dieser Form, heißt die Grammatik in Greibach Normalform.

Beweis:

Der Nachweis über die Richtigkeit dieser Aussage wird mit Hilfe eines Algorithmus erbracht, der eine beliebige Grammatik $G_1 = (V_1, \Sigma, P_1, S_1)$, die sich bereits in Chomsky Normalform (CNF) befindet, in eine Grammatik $G = (V, \Sigma, P, S)$ in Greibach Normalform (GNF) umwandelt.

Der Umwandlungsprozess wird in drei Schritten vorgenommen. Zunächst nummerieren wir die Nichtterminalsymbole der Ausgangsgrammatik G_1 mit i = 1...m

$$V_1 = \{A_1, A_2, \ldots, A_m\} \tag{19.16}$$

mit dem Ziel, alle Produktionen in die Form

$$A_i \rightarrow A_j\gamma \qquad i < j \tag{19.17}$$

zu überführen. Dazu durchlaufen wir alle A_i-Produktionen mit $A_i \rightarrow A_j\gamma$ und i = 1...m und ersetzen jeweils - wie in der Vorbemerkung gezeigt - die A_j mit allen A_j-Produktionen. Für den Fall, dass eine linksrekursive Produktion $A_i \rightarrow A_i\gamma$ auftritt, werden wie im vorherigen Abschnitt beschrieben, ein neues Nichtterminalsymbol und neue Produktionen eingeführt und die linksrekursive in eine rechtsrekursive Produktion umgewandelt.

Am Ende dieses ersten Schrittes haben wir Produktionen erhalten, für die mit $A_i \rightarrow A_j\gamma$ i < j gilt. Die Produktionen für A_m sind dann in Greibach Normalform $A_m \rightarrow aB_1B_2\ldots B_k$. Algorithmisch betrachtet sieht die Sortierung nach i < j so aus

```
for i := 1 to m do
  for j := 1 to i - 1 do
    foreach Aᵢ -> Aⱼα ∈ P₁ do
      Füge hinzu Aᵢ -> β₁α|β₂α|...|βₛα für alle
         Produktionen Aⱼ -> β₁α|β₂α|...|βₛα ;
      Entferne Aᵢ -> Aⱼα ;
    next
  next j

  if Aᵢ -> Aᵢα ∈ P₁ then
    Füge Nichtterminalsymbol Bᵢ hinzu ;
    Füge hinzu Aᵢ -> β₁Bᵢ|β₂Bᵢ|...|βₛBᵢ für alle
       Produktionen Aⱼ -> β₁|β₂|...|βₛ mit Aᵢ nicht erstem Zeichen von ß₁ ;
    Füge hinzu Bᵢ -> α₁|α₂|...|αᵣ ;
    Füge hinzu Bᵢ -> α₁Bᵢ|α₂Bᵢ|...|αᵣBᵢ ;
    Entferne Aᵢ -> Aᵢα ;
  endif
next i
```

Im zweiten Schritt können wir die so geordneten Produktionen in Greibach Normalform überführen, indem wir sukzessive alle A_i-Produktionen beginnend bei A_m (das ja bereits ein Terminalsymbol als erstes Zeichen auf der rechten Seite und dann wegen der ursprünglichen Chomsky Normalform von G1 folgende Nichtterminalsymbole hat, also in Greibach Normalform ist) in die rechte Seite der A_{i-1}-Produktion einsetzen. Da $A_i \rightarrow A_j\gamma$ i < j für alle Produktionen gilt, sind so im Anschluss an diesen Schritt alle Produktionen in Greibach Normalform. Algorithmisch kann das folgendermaßen beschrieben werden

```
for i := m - 1 downto 1 do
  foreach Aᵢ -> Aⱼα ∈ P₁ mit i < j do
    Füge hinzu Aᵢ -> β₁α|β₂α|...|βₛα für alle
       Produktionen Aⱼ -> β₁α|β₂α|...|βₛα ;
    Entferne Aᵢ -> Aⱼα ;
  next
next I
```

Im Anschluss daran befinden sich alle A_i-Produktionen in Greibach Normalform. Bleiben nur noch die möglicherweise hinzugekommenen B_i-Produktionen, deren rechte Seite im dritten und letzten Schritt analog umgeformt werden.

Da die ursprüngliche Grammatik G_1 in Chomsky Normalform ist, besteht die rechte Seite jeder Produktion entweder aus einem Terminalsymbol oder aus zwei Nichtterminalsymbolen. Durch die in den beiden ersten Schritten vorgenommen Umformungen sind deswegen keine Terminalsymbole weiter nach rechts gewandert, sondern können nach wie vor nur an erster Stelle stehen, wie es die Greibach Normalform fordert.

Bei den B_i-Produktionen besteht die rechten Seite ausschließlich aus Nichtterminalsymbolen. Da nach dem zweiten Schritt alle A_i-Produktionen in Greibach Normalform sind, muss im dritten Schritt lediglich das erste Nichtterminalsymbol der rechten Seite einer B_i-Produktion durch die rechte Seite der jeweiligen A_i-Produktion ersetzt werden (siehe Vorbemerkung) und folglich sind dann auch die B_i-Produktionen in der gewünschten Greibach Normalform.

19.1.4 Beispiel 1

Gegeben sei eine Grammatik $G = (\{A_1, A_2, A_3\}, \{a, b\}, P, A_1)$ mit den zugehörigen Produktionen

$$\begin{aligned} A_1 &\to A_2 A_3 \\ A_2 &\to A_3 A_1 \mid b \\ A_3 &\to A_1 A_2 \mid a \end{aligned} \tag{19.18}$$

Diese wird im Folgenden in Greibach Normalform überführt.

1. Schritt

Das Umwandlungsverfahren in Greibach Normalform beruht auf Ausgangsgrammatiken in Chomsky Normalform. Die hier gegebene Grammatik liegt in CNF vor, da auf der rechten Seite entweder zwei Nichtterminalsymbole oder ein Terminalsymbol stehen. Wir können daher direkt mit der Umwandlung in GNF beginnen.

Die Produktionen müssen alle in die Form $A_i \to A_j \gamma$ mit i < j gebracht und mögliche linksrekursive Produktionen in rechtsrekursive umgewandelt werden. Für die A_1- und A_2-Produktionen ist die Bedingung bereits gegeben, in die A_3-Produktion wird die rechte Seite von A_1 eingesetzt.

Dann erhält man

$$\begin{aligned} A_1 &\to A_2 A_3 \\ A_2 &\to A_3 A_1 \mid b \\ A_3 &\to A_2 A_3 A_2 \mid a \end{aligned} \tag{19.19}$$

und in einem weiteren Schritt wird A_2 in der rechten Seite von A_3 durch A_3A_1 und b ersetzt, so das man folgende Menge von Produktionen erhält

$$\begin{aligned} A_1 &\to A_2 A_3 \\ A_2 &\to A_3 A_1 \mid b \\ A_3 &\to A_3 A_1 A_3 A_2 \mid b A_3 A_2 \mid a \end{aligned} \tag{19.20}$$

Die A_3-Produktion entpuppt sich jetzt als eine linksrekursive Produktion, die durch Hinzufügen eines Nichtterminalsymbols B_3 und weiterer Produktionen entfernt wird.

$$\begin{aligned} A_1 &\to A_2 A_3 \\ A_2 &\to A_3 A_1 \mid b \\ A_3 &\to b A_3 A_2 B_3 \mid a B_3 \mid b A_3 A_2 \mid a \\ B_3 &\to A_1 A_3 A_2 B_3 \mid A_1 A_3 A_2 \end{aligned} \tag{19.21}$$

2. Schritt

Da sich jetzt alle Produktionen der Bedingung $A_i \to A_j \gamma$ mit i < j genügen und $A_m = A_3$ bereits in Greibach Normalform ist, können wir die anderen A_i-Produktionen auch beginnend bei $A_{m-1} = A_2$ in die gewünschte Form umgestalten.

Für A_2 erhalten wir dann

$$\begin{aligned}
&A_1 \to A_2A_3\\
&A_2 \to bA_3A_2B_3A_1 \mid aB_3A_1 \mid bA_3A_2A_1 \mid aA_1 \mid b\\
&A_3 \to bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a\\
&B_3 \to A_1A_3A_2B_3 \mid A_1A_3A_2
\end{aligned} \tag{19.22}$$

und entsprechend für A_1

$$\begin{aligned}
&A_1 \to bA_3A_2B_3A_1A_3 \mid aB_3A_1A_3 \mid bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3\\
&A_2 \to bA_3A_2B_3A_1 \mid aB_3A_1 \mid bA_3A_2A_1 \mid aA_1 \mid b\\
&A_3 \to bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a\\
&B_3 \to A_1A_3A_2B_3 \mid A_1A_3A_2
\end{aligned} \tag{19.23}$$

Damit sind alle A_i-Produktionen in Greibach Normalform. Fehlen noch die B_i-Produktionen.

3. Schritt

In das Nichtterminalsymbol der rechten Seite der beiden B_3-Produktionen werden alle fünf A_1-Produktionen eingesetzt, so dass 10 neue Produktionen entstehen.

$$\begin{aligned}
&A_1 \to bA_3A_2B_3A_1A_3 \mid aB_3A_1A_3 \mid bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3\\
&A_2 \to bA_3A_2B_3A_1 \mid aB_3A_1 \mid bA_3A_2A_1 \mid aA_1 \mid b\\
&A_3 \to bA_3A_2B_3 \mid aB_3 \mid bA_3A_2 \mid a\\
&B_3 \to bA_3A_2B_3A_1A_3A_3A_2B_3 \mid aB_3A_1A_3A_3A_2B_3 \mid bA_3A_2A_1A_3A_3A_2B_3\\
&B_3 \to aA_1A_3A_3A_2B_3 \mid bA_3A_3A_2B_3\\
&B_3 \to bA_3A_2B_3A_1A_3A_3A_2 \mid aB_3A_1A_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2\\
&B_3 \to aA_1A_3A_3A_2 \mid bA_3A_3A_2
\end{aligned} \tag{19.24}$$

Somit erfüllen alle Produktionen die Bedingung der Greibach Normalform, dass die rechte Seite immer aus einem Terminalsymbol gefolgt von keinem oder beliebig vielen Nichtterminalsymbolen besteht. Zu bemerken ist, dass aus den 5 Produktionen der Grammatik in Chomsky Normalform immerhin 24 Produktionen sowie ein weiteres Nichtterminalsymbol in Greibach Normalform geworden sind.

19.1.5 Beispiel 2

Gegeben sei eine Grammatik $G = (\{A,S\},\{a,b\},P,S)$ mit den zugehörigen Produktionen

$$\begin{aligned}
&S \to AA \mid a\\
&A \to SS \mid b
\end{aligned} \tag{19.25}$$

1. Schritt

Das Umwandlungsverfahren in Greibach Normalform beruht auf Ausgangsgrammatiken in Chomsky Normalform. Die hier gegebene Grammatik liegt in CNF vor, wir können daher direkt beginnen.

Zunächst werden die Nichtterminalsymbole durchnummeriert. Sagen wir, S sei das erste und A das zweite Nichtterminalsymbol, also $A_1 = S$ und $A_2 = A$, wobei wir aufgrund der Einfachheit auf diese Umbenennung verzichten können und die Reihenfolge im Hintergrund behalten. Dann ist die Zuordnung $S \to AA$ in Ordnung, $A \to SS$ muss jedoch umgewandelt werden, damit $A_i \to A_j\gamma$ mit $i < j$ für alle Produktionen gilt.

Dadurch erhält man

$$\begin{aligned}
&S \to AA \mid a\\
&A \to AAS \mid aS \mid b
\end{aligned} \tag{19.26}$$

was eine linksrekursive Ableitung erzeugt. Daher führen wir ein neues Nichtterminalsymbol B_2 ein und wandeln in rechtsrekursive Produktion um

$$\begin{aligned} S &\to AA \mid a \\ A &\to aS \mid b \mid aSB_2 \mid bB_2 \\ B_2 &\to AS \mid ASB_2 \end{aligned} \tag{19.27}$$

Dann liegen die A-Produktionen schon in Greibach Normalform vor und wir können zum zweiten Schritt übergehen.

2. Schritt

Jetzt muss das erste A der rechten Seite von S durch die A-Produktionen ersetzt werden und man erhält

$$\begin{aligned} S &\to aSA \mid bA \mid aSB_2A \mid bB_2A \mid a \\ A &\to aS \mid b \mid aSB_2 \mid bB_2 \\ B_2 &\to AS \mid ASB_2 \end{aligned} \tag{19.28}$$

Schön! Jetzt liegen alle Produktion in der gewünschten Form vor, bis auf die B_2-Produktionen, was jedoch auch leicht bewerkstelligt ist.

3. Schritt

Das erste Nichtterminalsymbol der rechten Seite der B_2-Produktionen ist A und wird jeweils durch die beiden A-Produktionen ersetzt, so dass vier neue Produktionen entstehen.

$$\begin{aligned} S &\to aSA \mid bA \mid aSB_2A \mid bB_2A \mid a \\ A &\to aS \mid b \mid aSB_2 \mid bB_2 \\ B_2 &\to aSS \mid bS \mid aSB_2S \mid bB_2S \\ B_2 &\to aSSB_2 \mid bSB_2 \mid aSB_2SB_2 \mid bB_2SB_2 \end{aligned} \tag{19.29}$$

Somit liegen alle Produktionen die Greibach Normalform und die Umwandlung ist abgeschlossen.

20 Vorlesung vom 20. Juni 2000

20.1 Odgens Lemma

Satz:

Sei $G=(V,\Sigma,P,S)$ eine kontextfreie Grammatik. Dann existiert eine Konstante $k\geq 1$ (welche im Allgemeinen recht groß ist), so dass für alle Wörter z der Sprache mit $z\in L(G)$ und $|z|\geq k$, in denen mindestens k Positionen markiert sind, gilt:

Das Wort z kann geschrieben werden, als die Zerlegung $z=uvwxy$ mit den folgenden 4 Bedingungen:

1. w enthält mindestens eine markierte Position,
2. u und v enthalten beide markierte Positionen, oder aber x und y enthalten beide markierte Positionen,
3. vwx enthält höchstens k markiere Positionen
4. Es existiert ein Nichtterminal A mit:

$$S\overset{+}{\underset{G}{\Rightarrow}}uAy\overset{+}{\underset{G}{\Rightarrow}}uvAxy\overset{+}{\underset{G}{\Rightarrow}}\ldots\overset{+}{\underset{G}{\Rightarrow}}uv^iAx^iy\overset{+}{\underset{G}{\Rightarrow}}uv^iwx^iy\quad\forall i\geq 0 \tag{20.1}$$

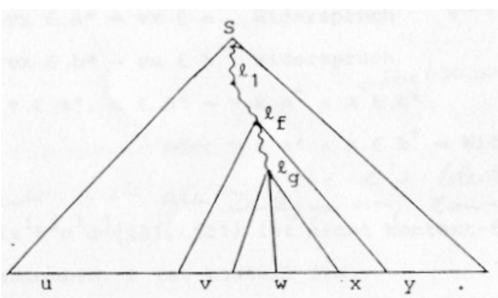


Abbildung 37

Das Bild zeigt, wie man sich dieses am besten vorstellen kann. Es gilt einen Weg vom Startsymbol S aus hin zu den markierten Blättern zu finden. Da nun der Weg sehr lang ist, existiert eine Schliefe. Man betrachtet dann den Teilbaum der letzten Wiederholung. Man sieht nun, dass w zumindest eine markierte Position enthalten muss.

Beweis des Satzes:

Es gelte:

$$\begin{aligned} m &=_{Df} \#(V-\Sigma) \\ l &=_{Df} \max.\left(n \middle| A\rightarrow\alpha\in P,\ |\alpha|=n\right) \\ k &=_{Df} l^{2\cdot m+3} \end{aligned} \tag{20.2}$$

Es sei dann $z\in L(G)$ mit $|z|\geq k$ und weiterhin mindestens k Positionen in z markiert. Schließlich sei T der Ableitungsbaum für z mit der Länge $\geq 2\cdot m+3$. Ein Knoten n heißt Verzweigungsknoten, wenn n mindestens zwei direkte Nachfolger hat. Ein Beispiel wäre, wenn der Knoten n zwei Nachfolger n_1 und n_2 hat, welche beide markierte Blätter unter ihren Nachfolgern haben.

Nun lässt sich ein Weg $n_1,\ldots,n_p$ durch den Ableitungsbaum T wie folgt konstruieren:

1. n_1 ist die Wurzel von T,

2. betrachtet man die Nachfolger eines Knoten n_i. Sollte nur einer von n_i's direkten Nachfolgern markierte Knoten haben, so wird dieser der Nachfolger n_{i+1}.
3. Wenn n_i ein Verzweigungsknoten ist, dann soll n_{i+1} derjenige direkte Nachfolger sein, der die größte Zahl von markierten Blättern als Nachfolger hat. Bei Gleichheit kann n_{i+1} beliebig gewählt werden.
4. Wenn n_i ein Blatt ist, endet die Konstruktion.

Nun ist $n_1, n_2, \ldots, n_p$ der Weg.

Behauptung:

Wenn der Weg $n_1, n_2, \ldots, n_i$ r Verzweigungsknoten enthält, dann hat $n_{i+1} \geq l^{2\cdot m+3-r}$ markierte Blätter. Der Beweis folgt durch Induktion. Für $i = 0, r = 0$ gilt:

$$n_1 \text{ hat } \geq \mathrm{l}^{2\mathrm{m+r-0}} = k \tag{20.3}$$

markierte Blätter. Angenommen, die Aussage ist richtig für $(i-1)$. Wenn n_i kein Verzweigungsknoten ist, dann haben n_i und n_{i+1} dieselbe Anzahl von markierten Blättern. Wenn n_i ein Verzweigungsknoten ist, dann hat n_{i+1} mindestens $\frac{1}{l} \cdot l^{2m+3-r} = l^{2m+3-(r+1)}$ viele markierte Blätter bei $(r+1)$ Verzweigungsknoten.

In $n_1, n_2, \ldots, n_p$ muss es mindestens $2 \cdot m + 3$ viele Verzweigungsknoten geben. Denn anderenfalls hätte $n_p \geq l^{2m+3-r} > 1 \cdot (r < 2m+3)$ viele markierte Blätter. Das wäre jedoch ein Widerspruch, denn n_p ist ein Blatt, und somit kein Verzweigungsknoten. Es gilt also $p > 2 \cdot m + 3$.

Seien nun $b_1, \ldots, b_{2m+3}$ die letzten $(2m+3)$ Verzweigungsknoten. b_i heißt „*linker Verzweigungsknoten*“, wenn ein linker direkter Nachfolger von b_i, der nicht auf dem Weg liegt, ein markiertes Blatt links von n_p enthält. Analog dazu ist b_i der „*rechte Verzweigungsknoten*“, wenn ein rechter direkter Nachfolger von b_i, der nicht auf dem Weg liegt, ein markiertes Blatt rechts von n_p enthält.

Ohne Beschränkung der Allgemeinheit gilt: Es gibt mindestens $(m+2)$ viele linke Verzweigungsknoten. Seien nun $l_1, \ldots, l_{m+2}$ die letzten linken Verzweigungsknoten in $b_1, \ldots, b_{2m+3}$. Unter $l_2, \ldots, l_{m+2}$ gibt es dann zwei Knoten l_f und l_g mit der selben Bezeichnung, da $\#(V - \Sigma) = m$. Für den Fall, dass f echt kleiner als g ist, sieht dies wie auf dem folgenden Bild aus:

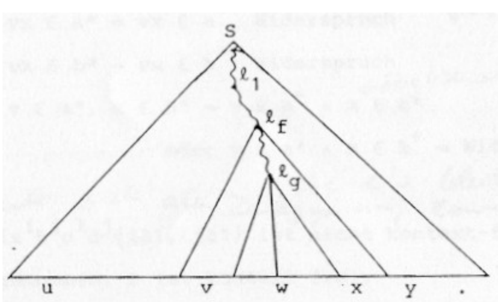


Abbildung 38

Man kann nun $l_f = l_g = A$ nennen. Es ergibt sich dann:

$$S \overset{+}{\Rightarrow} uAy \tag{20.4}$$

$$A \overset{+}{\Rightarrow} vAx \tag{20.5}$$

$$A \overset{+}{\Rightarrow} w \tag{20.6}$$

Und damit gilt dann:

$$S \overset{+}{\Rightarrow} uAy \overset{+}{\Rightarrow} uvAxy \overset{+}{\Rightarrow} uv^i Ax^i y \overset{+}{\Rightarrow} uv^i wx^i y \quad \forall i \geq 0 \tag{20.7}$$

Da nun l_1 ein linker Verzweigungsknoten ist, hat u mindestens ein markiertes Blatt. Des weiteren ist l_f ein linker Verzweigungsknoten und damit hat v mindestens ein markiertes Blatt. w enthält als markiertes Blatt n_p.

Es gilt auch, dass b_1 der $(2m+3)$-te Verzweigungsknoten vom ende ist. Somit hat b_1 höchstens $l^{2m+3} = k$ viele markierte Blätter. Da l_f ein Nachfolger von b_1 ist, hat vwx höchstens k markierte Blätter.

Analog zu dem eben vorgeführten Beweis für linke Verzweigungsknoten, läuft auch der Fall für $m+2$ rechte Verzweigungsknoten. Hier enthalten dann x und y jeweils mindestens ein markiertes Blatt.

Korollar: (Pumping Lemma nach Bar-Hillel, Perles, Shamir)

Sei L eine kontextfreie Sprache. Dann existiert eine Konstante k in der Art, dass, wenn $|z| \geq k$ und $z \in L$ ist, dann gilt:

Es existiert für z eine Zerlegung in $z = uvwxy$ mit $u, v, w, x, y \in \Sigma^*$ und:

$$\begin{aligned} & vx \neq \varepsilon \\ & |vwx| \leq k \\ & \forall i \geq 0: \; uv^i wx^i y \in L \end{aligned} \tag{20.8}$$

Zum Beweis dieses Korollars, wählt man eine beliebige kontextfreie Grammatik G für L und markiert alle Positionen in z

20.1.1 Beispiele für Kontextfreiheit und Nichtkontextfreiheit

Behauptung:

Die Sprache $L = \{a^n b^n c^n \,|\, n \geq 1\}$ ist nicht kontextfrei.

Beweis:

Angenommen die Sprache L wäre kontextfrei. Es sei dann k die Konstante des Pumping Lemmas. Und es gilt:

$$\begin{aligned} & z =_{Df} a^k b^k c^k \\ & z \in L \end{aligned} \tag{20.9}$$

Es existiert dann eine Zerlegung für z mit:

$$\begin{array}{rcl} \exists u,v,w,x,y: z & = & uvwxy \\ |vwx| & \leq & k \\ \Rightarrow vwx & \in & a^*b^* \\ \vee vwx & \in & b^*c^* \end{array} \tag{20.10}$$

Es soll hier nur der Fall $vwx \in a^*b^*$ betrachtet werden. Dafür gilt:

$$\begin{array}{l} v \in a^* \cup b^* \\ x \in a^* \cup b^* \end{array} \tag{20.11}$$

Nun gibt es drei Fälle zu beachten:

1. $vx \in a^* \Rightarrow vx \in a^+$ dies stellt aber einen Widerspruch dar, denn die Anzahl der a's würde schneller wachsen.
2. $vx \in b^* \Rightarrow vx \in b^+$ und wieder haben wir einen Widerspruch, da hie die b's zu schnell wachsen, schließlich bleibt noch
3. $v \in a^*, x \in b^* \Rightarrow \begin{array}{rl} & v \in a^+ \wedge x \in b^* \\ oder & v \in a^* \wedge x \in b^+ \end{array}$ auch in diesem Fall bekommt man einen Widerspruch, denn die c's bleiben auf der Strecke.

Behauptung:

Die Sprache $L = \left\{ a^i b^j c^i d^j \,\middle|\, i \geq 1, j \geq 1 \right\}$ ist nicht kontextfrei.

Beweis:

Angenommen, die Sprache L wäre kontextfrei. Es sei nun n die Konstante des Pumping Lemmas. Dann gibt es eine Zerlegung von $z = a^n b^n c^n d^n$ in $z = uvwxy$ mit $u,v,w,x,y \in \Sigma^*$ und $|vwx| \leq n$.

Wie man sieht, gilt:

1. vx enthält höchstens zwei verschiedene Symbole,
2. Falls vx zwei verschiedene Symbole enthält, müssen diese „benachbart" sein. Dann erhält man durch „pumpen":

$$\left.\begin{array}{rcl} \#a's & > & \#c's \\ \#b's & > & \#d's \\ \#c's & > & \#a's \\ \#d's & > & \#b's \end{array}\right\} \text{und dies stellt einen Widerspruch dar}$$

Man erhält also jedes Mal einen Widerspruch, da es nicht zu schaffen ist eine gleiche Anzahl von a's und c's oder b's und d's zu erreichen.

Behauptung:

Die Sprache $L = \left\{ a^i b^j c^k \,\middle|\, i \neq j, j \neq k, k \neq i \right\}$ ist nicht kontextfrei.

Beweis:

Angenommen die Sprache L wäre kontextfrei. Sei nun n die Odgens Konstante. Dann gibt es für ein Wort $z = a^n b^{n+n!} c^{nü2 \cdot n!}$ der Sprache L eine Zerlegung $z = uvwxy$. Es seien nun alle a's markiert.

1. u und v enthalten a's.

   Dann gilt: $v \in a^*$ und damit gilt $v = a^i$ für $i \le n$. Es gibt dann die folgenden 3 Fälle:

   $x \in a^* \vee x \in b^* \vee x \in c^*$

   - $x \in a^*$

     Dann gilt: $x = a^j$ und somit $vx = a^{i+j}$ mit $(i+j) \le n$. Dann sollte das Wort: $uv^{1+\frac{n!}{i+j}} wx^{1+\frac{n!}{i+j}} y = a^{n+(i+j)\cdot\frac{n!}{(i+j)}} n^{n+n!} c^{n+2n!}$ in der Sprache sein, dies kann jedoch nicht sein, da die Anzahl der a's gleich der Anzahl der b's.

   - $x \in b^*$

     Dann ist $x = b^j$ mit $j \ge 0$ und somit müsste das Wort: $uv^{1+2\cdot\frac{n!}{i}} wx^{1+2\cdot\frac{n!}{i}} = a^{n+i\cdot 2\frac{n!}{i}} b^r c^{n+2n!}$ in der Sprache sein. Jedoch führt auch dies zu einem Widerspruch, da die Anzahl der a's gleich der Anzahl der c's ist. Bleibt noch ein Fall zu behandeln.

   - $x \in c^*$

     hier ist $x = c^j$ für $j \ge 0$. Dann müsste das Wort: $uv^{1+\frac{n!}{i}} wx^{1+\frac{n!}{i}} y = a^{n+i\cdot\frac{n!}{i}} b^{n+n!} c^r$ in der Sprache sein. Jedoch ist hier die Anzahl der a's so groß wie die Anzahl der b's, was einen Widerspruch darstellt.

2. Der Fall, dass x und y beide a's enthalten, funktioniert analog zum 1. Fall. Somit soll es hier nicht wiederholt werden.

Behauptung:

Die Sprache $L = \{a^i b^j c^k \mid i = j \vee j = k\}$ ist eine kontextfreie Sprache, die inhärent mehrdeutig ist.

Beweis:

Zuerst soll noch mal wiederholt werden, was inhärent mehrdeutig bedeutet. Dies bedeutet, dass egal welche Ableitung gewählt wird, immer 2 mögliche Wege existieren.

Um dies zu zeigen, genügt es eine kontextfreie Grammatik aufzustellen, welche mehrdeutig ist. Eine solche kontextfreie Grammatik ist:

$$\begin{aligned} &S \to AB \mid DC \\ &A \to aA \mid \varepsilon \\ &B \to bBc \mid \varepsilon \\ &C \to cC \mid \varepsilon \\ &D \to aDb \mid \varepsilon \end{aligned} \tag{20.12}$$

Diese Grammatik ist mehrdeutig, wie man schnell an folgendem Beispiel erkennt:

$$S \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aabBc \Rightarrow aabbBcc \Rightarrow a^2b^2c^2 \tag{20.13}$$

eine andere Ableitung für dieses Wort ist:

$$S \Rightarrow DC \Rightarrow aDbC \Rightarrow aaDbbC \Rightarrow a^2b^2C \Rightarrow a^2b^2cC \Rightarrow a^2b^2c^2C \Rightarrow a^2b^2c^2 \tag{20.14}$$

wir behaupten, dass jede kontextfreie Grammatik für L mehrdeutig sein muss. Sei G eine beliebige Grammatik, die L erzeugt. Sei weiterhin k die Konstante des Odgens Lemma. Es sei nun ohne Einschränkung $k \ge 3$. Sei $z = a^k b^k c^{k+k!}$. Nun seien alle a's markiert und $z \in L$. Nach Odgens Lemma gibt es dann für z eine Zerlegung $z = uvwxy$ für $u, v, w, x, y \in \Sigma^*$. Dann gilt:

1. w enthält mindestens ein a.

2. $uv \in a^*$
3. $x \in a^* \cup b^* \cup c^*$. Hierbei gilt, dass x mit Sicherheit nicht zwei verschiedene Symbole enthalten kann, denn sonst würde durch das Pumpen „gemischte“ Folgen erzeugt werden.
4. vwx enthält höchstens k viele a’s.

Nun sind die folgenden Fälle zu betrachten: $x \in a^* \vee x \in b^* \vee x \in c^*$:

1. $x \in a^*$

   Dann gilt: $vx \in a^*$ und damit $v = a^i$ für $0 < i \le k$ und man erhält Wörter: $uv^2wx^2y = a^{k+i}b^kc^{k+k!}$ die auch aus der Sprache stammen müssten. Jedoch stellt dieses einen Widerspruch dar, da weder die Anzahl von a’s und b’s, noch die Anzahl der b’s und c’s gleich ist.

2. $x \in c^*$

   Es ist dann $v = a^i$ mit $0 < i \le k$ und es gilt $x = c^j$ für $j \ge 0$. Damit bekommt man Wörter: $uv^2wx^2y = a^{k+i}b^kc^{k+k!+j}$, die auch in der Sprache sein müssen. Doch wie schon bei a) bekommt man einen Widerspruch.

3. $x \in b^*$

   Es gilt: $v = a^i$ für $0 < i \le k$ und weiterhin $x = b^j$ für $j \ge 0$. Damit bekommt man die folgenden Wörter: $uv^2wx^2y = a^{k+i}b^{k+j}c^{k+k!}$, die Element der Sprache sein müssen. Es gilt also zwei weitere Fälle zu unterscheiden zum einen den Fall $i = j$ und zum anderen $(j = k! \wedge j \ne i)$:

   - $(j = k! \wedge j \ne i)$

     Damit würden Wörter: $uv^3wx^3y = a^{k+2i}b^{k+2k!}c^{k+k!}$ diese müssten in der Sprache sein, jedoch ist dies ein Widerspruch.

   - $i = j$

     Es gilt damit: $S \overset{+}{\Rightarrow} uAy \overset{+}{\Rightarrow} uv^mAx^my \overset{+}{\Rightarrow} uv^mwx^my$. Für $m = k!/i + 1$ gilt dann:

     $$uv^mwx^my = a^{k+k!}b^{k+k!}c^{k+k!} \tag{20.15}$$

Eine ähnliche Argumentation liefert dann für den umgekehrten Fall: $a^{k+k!}b^kc^k$. Für $u', v', w', x', y' \in \Sigma^*$ mit $a^{k+k!}b^kc^k = u'v'w'x'y'$ und mit $v' \in b^*$. Schließlich kommt man zu der Ableitung:

$$S \overset{+}{\Rightarrow} u'By' \overset{+}{\Rightarrow} u'(v')^{m'}B(x')^{m'}y' \overset{+}{\Rightarrow} u'(v')^{m'}w'(x')^{m'}y' = a^{k+k!}b^{k+k!}c^{k+k!} \tag{20.16}$$

Behauptung:

Diese beiden Ableitungen repräsentieren verschiedene Ableitungsbäume.

Beweis:

Angenommen, diese beiden Ableitungen beschreiben denselben Ableitungsbaum.

A erzeuge a’s und b’s während, B b’s und c’s erzeugt. Somit ist A kein Nachfolger von B und B ist kein Nachfolger von A. Betrachtet man nun die Ableitung:

$$S \overset{+}{\Rightarrow} t_1At_2Bt_3 \text{ mit } t_1, t_2, t_3 \in \Sigma^*$$

Nun muss für alle i, j gelten:

$$t_1v^iwx^it_2(v')^jw'(x')^jt_3 \in L$$

Nun ist $i = j$ hinreichend groß zu wählen.

Es gilt $|v| = |x|$ und $|v'| = |x'|$. Damit ist also $v \in a^+, x \in b^+ \wedge v' \in b^+, x' \in c^+$. Somit erhält man also $z' \in L$ mit $\#_b(z') > \#_a(z') \wedge \#_b(z') > \#_c(z')$. Dies ist jedoch ein Widerspruch, da wir entweder eine gleiche Anzahl von a's und b's wollten, oder aber eine gleiche Anzahl von b's und c's. Somit können die beiden Ableitungen nicht den selben Ableitungsbaum beschrieben haben.

21 Vorlesung vom 27. Juni 2000

21.1 Deterministische PDA

Wir betrachten nun eine Sprachklasse, die echt zwischen den regulären Mengen und den kontextfreien Sprachen liegen: die *deterministischen kontextfreien Sprachen* (DCFL). Diese sind interessant, da es sich herausstellt, dass die Syntax vieler Programmiersprachen mittels DCFLs beschrieben werden kann.

Definition:

Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ ein PDA. M heißt dann deterministisch (DPDA): $\Leftrightarrow$

$$\begin{aligned} &1.\ \forall q \in Q, a \in \Sigma \cup \{\varepsilon\}, A \in \Gamma : \left|\delta(q,a,A)\right| \leq 1 \\ &2.\ \delta(q,\varepsilon,A) \neq \varnothing \Rightarrow \forall a \in \Sigma : \delta(q,a,A) = \varnothing \end{aligned} \tag{21.1}$$

$$T(M) := \left(x \middle| x \in \Sigma^* \wedge (q_0, x, Z_0) \overset{*}{\mapsto} (f, \varepsilon, \alpha), f \in F, \alpha \in \Gamma^* \right) \tag{21.2}$$

Regel 1 verhindert Wahlmöglichkeiten für dieselbe Eingabe. Regel 2 verhindert die Wahl zwischen der Benutzung eines Eingabesymbols und einer ε-Bewegung.

Definition:

Eine Sprache heißt deterministisch kontextfrei (dkfS): $\Leftrightarrow$ Es existiert ein DPDA mit L = T(M).

21.1.1 Beispiele

- Die Sprache $\{a^n b^n \mid n \geq 1\}$ ist deterministisch.
- Die Sprache $\{wcw^R \mid w \in \{a,b\}^*\}$ ist deterministisch.
- Die Sprache $\{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$ ist nicht deterministisch.

Beim letzten Beispiel müsste der Automat raten, ob für ein a nun ein oder zwei b vorliegen müssen.

Dagegen gilt:

- $d_1\{a^n b^n \mid n \geq 1\} \cup d_2\{a^n b^{2n} \mid n \geq 1\}$ ist deterministisch, falls gilt $d_1 \neq d_2$.
- $\{a^n d_1 b^n \mid n \geq 1\} \cup \{a^n d_2 b^{2n} \mid n \geq 1\}$ ist deterministisch, falls gilt $d_1 \neq d_2$. Dabei wird so vorgegangen: zunächst werden für jedes a zwei Symbole auf den Keller gelegt. Wenn d_1 gelesen wird, so wird je ein Symbol wieder vom Keller gelöscht, ansonsten wird mit den je zweiten weitergearbeitet.
- $\{a^n b^n \mid n \geq 1\} d_1 \cup \{a^n b^{2n} \mid n \geq 1\} d_2$ ist nicht deterministisch.

21.1.2 kfS und dkfS

Satz:

Sei L eine kontextfreie Sprache (kfS). Dann gibt es eine deterministische kontextfreie Sprache L‘ (dkfS) und einen Homomorphismus h mit h(L‘) = L.

Beweis:

Sei L kontextfrei. Daraus folgt, dass es einen PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ geben muss mit L = T(M).

Es sei

$$m := \max\left\{n \mid (q',\alpha) \in \delta(q,a,A), |\alpha| = n\right\} \tag{21.3}$$

Also ist m die maximale Länge von Symbolen, die auf den Keller gestellt werden. Es sei weiter

$$\Sigma' := \Sigma \times Q \times \Gamma^{\leq m} \tag{21.4}$$

mit $\Gamma^{\leq m} := \bigcup_{i=0}^{m} \Gamma^{i}$.

Wir definieren nun den deterministischen PDA M‘ wie folgt:

$$M' = \left(Q, \Sigma', \Gamma, \delta', q_0, Z_0, F\right) \tag{21.5}$$

mit

$$\delta'\left(q, [a, q', \alpha], A\right) := \left\{(q',\alpha) \mid (q',\alpha) \in \delta(q,a,A)\right\} \tag{21.6}$$

Wir müssen durch Induktion den Determinismus zeigen:

$$\begin{gathered} \left(q_0, [a_1 \ldots a_n, Z_0]\right) \underset{M}{\mapsto} \left(q_{i_1}, a_2 \ldots a_n, \alpha_1\right) \\ \ldots \mapsto \left(f, \varepsilon, \gamma\right) \\ \Leftrightarrow \\ \left(q_0, \left[a_1, q_{i_1}, \alpha_1\right]\left[a_2, q_{i_2}, \square\right] \ldots \left[a_n, f, \square\right], Z_0\right) \underset{M'}{\mapsto} \left(q_{i_1}, \left[a_2, q_{i_2}, \square\right] \ldots \left[a_n, f, \square\right]\right) \\ \ldots \mapsto \left(f, \varepsilon, \gamma\right) \end{gathered} \tag{21.7}$$

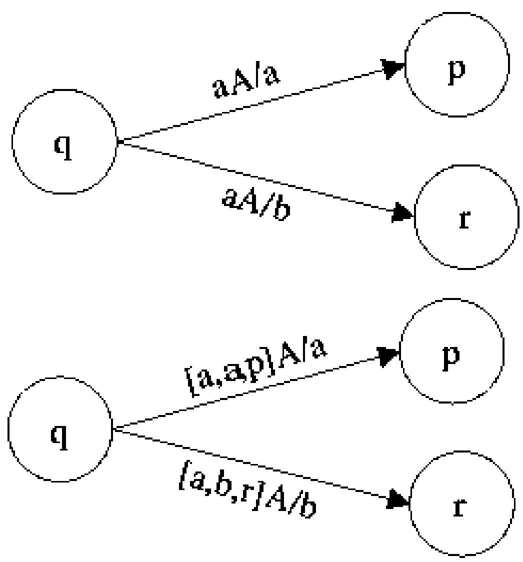


Abbildung 39 - Beispiel zum Beweis

In Abbildung 39 sieht man das Prinzip des Beweises. Oben sieht man einen nichtdeterministischen Kellerautomaten (da bei der gleichen Eingabe in zwei verschiedene Zustände p, r gewechselt werden kann).

Im unteren Teil der Abbildung sieht man nun den deterministischen Automaten mit den neuen Eingabesymbolen [a,α,p] und [a,β,r]. Es wird also eine Alphabetvergrößerung vorgenommen. Um diese Zeichen wieder auf die alten Zeichen zurückzumappen wird der folgende Homomorphismus definiert:

$$h : \Sigma' \to \Sigma \text{ mit } h([a, q', \alpha]) := a \tag{21.8}$$

Somit gilt $h(T(M')) = T(M) = L$.

21.2 Turing-Maschinen und Entscheidbarkeit

21.2.1 Problem, Algorithmus, Entscheidbarkeit

Definition (Problem):

Ein *Problem* P ist ein Paar $P = \langle V, \chi \rangle$ mit

$$\begin{aligned} &1.\ \text{V ist die Menge der Objekte} \\ &2.\ \chi \text{ ist ein Prädikat } \chi : V \to \{0,1\} \end{aligned} \tag{21.9}$$

χ ist also letztendlich die Beschreibung des Problems, das wir lösen wollen.

Definition (Algorithmus):

Ein *Algorithmus A* für ein Problem $P = \langle V, \chi \rangle$ ist eine endliche Menge von Instruktionen, die für alle $x \in V$ hält mit

$$A(x) = \chi(x)\ \forall x \in V \tag{21.10}$$

Definition (Entscheidbarkeit):

Ein Problem $P = \langle V, \chi \rangle$ heißt entscheidbar $:\Leftrightarrow \exists$ ein Algorithmus A für $\langle V, \rho \rangle$

Beispiel 1

Das Problem $P = \langle V, \rho \rangle$ mit

$$\begin{aligned} &V = \{(G,x) \mid \text{G eine reguläre Grammatik, x ein Wort}\} \\ &\rho : v \to \{0,1\} \\ &\rho(G,x) = 1 \Leftrightarrow x \in L(G) \\ &\rho(G,x) = 0 \Leftrightarrow x \notin L(G) \end{aligned} \tag{21.11}$$

Beispiel 2

Das Problem (Leerheitsproblem) $P = \langle V, \rho \rangle$ mit

$$\begin{aligned} &V = \{G \mid \text{G eine reguläre Grammatik}\} \\ &\rho : v \to \{0,1\} \\ &\rho(G,x) = 1 \Leftrightarrow L(G) = \emptyset \\ &\rho(G,x) = 0 \Leftrightarrow L(G) \neq \emptyset \end{aligned} \tag{21.12}$$

21.2.2 Modelle für Algorithmen

Es gibt unter anderem die folgenden beiden Modelle für Algorithmen:

- Turing Maschinen (Alan Turing)
- Rekursive Funktionen (Kurt Gödel)

Diese beiden Ansätze sind jedoch äquivalent.

21.2.3 Churchsche These

Ende endliche Menge A von Instruktionen ist ein Algorithmus genau dann, wenn eine Turing-Maschine m existiert mit:

$$\begin{aligned}&1.\ m(x) \text{ hält für alle x}\\&2.\ m(x) \text{ erreicht einen Endzustand} \Leftrightarrow A(x)=1.\end{aligned} \tag{21.13}$$

21.2.4 Turing-Maschine

Definition (Turing-Maschine):

Eine Turing-Maschine M ist ein 7-Tupel $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ mit

$$\begin{aligned}&1.\ \text{Q ist eine endliche Menge von Zuständen}\\&2.\ \Gamma \text{ ist eine endliche Menge von Symbolen (Bandalphabet)}\\&3.\ \mathrm{B} \in \Gamma \text{ ist das "Blank"-Symbol}\\&4.\ \Sigma \text{ ist das Eingabealphabet mit } \Sigma \subset \Gamma, \mathrm{B} \notin \Sigma\\&5.\ q_0 \text{ ist der Anfangszustand}\\&6.\ \text{F ist die Menge der akzeptierenden Zustände}\\&7.\ \delta{:}\mathrm{Q} \times \Gamma \to \mathrm{Q} \times (\Gamma - \{B\}) \times \{L, R\}\end{aligned} \tag{21.14}$$

Dabei ist zu beachten, dass {L, R} die Bewegungen des Schreib-/Lesekopfes darstellen sollen. δ ist in der Regel partiell, d. h. nicht überall definiert.

Eine Turingmaschine kann man sich also vorstellen als eine Maschine, die auf einem (Eingabe)band operiert. Sie kann dabei ein Zeichen lesen und abhängig von diesem und dem momentanen Zustand, in dem sie sich befindet, ein neues Zeichen an die Stelle des Gelesenen schreiben, in einen neuen Zustand wechseln und den Schreib-/Lesekopf nach recht oder links um eine Stelle bewegen.

Definition (Konfiguration):

Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ eine Turing-Maschine. Eine *Konfiguration* von M wird wie folgt bezeichnet:

$$\alpha_1 q \alpha_2 \tag{21.15}$$

Dabei gilt $\alpha_1\alpha_2 \in \Gamma^*$, so dass α_1 das am weitesten links stehende Nicht-Blank-Symbol und α_2 das am weitesten rechts stehende Nicht-Blank-Symbol enthält. q ist der gegenwärtige Zustand der Maschine und kennzeichnet auch die Position des Schreib-/Lesekopfes.

Wir können die Bewegung einer Turing-Maschine wie folgt formal beschreiben. Zunächst die Linksbewegung:

$$\begin{aligned}X_1X_2\ldots X_{i-1}qX_i\ldots X_n \mapsto X_1X_2\ldots X_{i-2}pX_{i-1}Y\ldots X_n\\ \Leftrightarrow \delta(q, X_i) = (p, Y, L)\end{aligned} \tag{21.16}$$

Hierbei wird vom Zustand q aus das aktuelle Zeichen (X_i) ersetzt durch Y, in den Zustand p verzweigt und der Kopf eine Stelle nach links bewegt.

Analog die Rechtsbewegung:

$$\begin{aligned}X_1X_2\ldots X_{i-1}qX_i\ldots X_n \mapsto X_1X_2\ldots X_{i-1}YpX_{i+1}\ldots X_n\\ \Leftrightarrow \delta(q, X_i) = (p, Y, R)\end{aligned} \tag{21.17}$$

Wir bezeichnen mit $\overset{+}{\mapsto}$ und $\overset{*}{\mapsto}$ die üblichen Erweiterungen.

Es ist

$$T(M)=\left\{w \mid w\in\Sigma^* \wedge (q_0w)\overset{*}{\mapsto}(\alpha_1 f\alpha_2), f\in F\right\} \tag{21.18}$$

die Menge der von M akzeptierten Eingabewörter.

Wir vereinbaren, dass M in einem akzeptierenden Zustand f ∈ F stets anhält.

Beispiel

Wir betrachten eine Turing-Maschine für die Sprache $L=\left\{0^n1^n \mid n\geq 1\right\}$.

Es sind $M=(Q,\Sigma,\Gamma,\delta,q_0,B,F), Q=\{q_0,q_1,\ldots,q_5\}, \Sigma=\{0,1\}, \Gamma=\{0,1,B,X,Y\}, F=\{q_5\}$.

Die Übergangsfunktion δ geben wir in folgender Tabelle an:

$q_0 0$	$q_1 XR$	erste 0 von links durch X ersetzen
$q_1 0$	$q_1 0R$	zur ersten 1 nach rechts gehen
$q_1 Y$	$q_1 YR$	durch Y ersetzen
$q_1 1$	$q_2 YL$	
$q_2 Y$	$q_2 YL$	nach links die erste 0 suchen
$q_2 X$	$q_3 XR$	falls keine mehr da: in q_3 gehen
$q_2 0$	$q_4 0L$	
$q_4 0$	$q_4 0L$	nach links das erste X suchen
$q_4 X$	$q_0 XR$	
$q_3 Y$	$q_3 YR$	keine Nullen mehr da ⇒
$q_3 B$	$q_5 YR$	prüfen, ob auch keine Einsen mehr da sind.
		Falls ja: in akzeptierenden Zustand gehen!

21.2.5 Universelle Turingmaschine

Wir wollen nun eine Turing-Maschine bauen, die andere Turingmaschinen simulieren kann – und zwar auf jedes Eingabewort hin. Wir wollen also eine Turing-Maschine U mit folgender Eigenschaft:

- M sei eine beliebige Turingmaschine
- x sei ein beliebiges Eingabewort
- Dann führt U bei Eingabe von (M,x) die Simulation von M auf x durch.

Jetzt ist aber die Frage, wie die Eingabe (M,x) aussehen soll. Hierfür müssen wir eine Kodierung auf dem Band vornehmen.

Wir stellen zunächst folgendes fest:

Zu jedem Alphabet Γ - {B} gibt es eine Kodierung in $\{0,1\}^*$, so dass gilt

$$Kod(M) \text{ akzeptiert } Kod(x) \Leftrightarrow \text{M akzeptiert x} \tag{21.19}$$

Wir nehmen also im weiteren an, dass alle Turing-Maschinen auf einem Alphabet {0,1,B} mit Σ = {0,1} arbeiten. Wir wollen nun die zu simulierende Turing-Maschine selbst kodieren. Wir wählen hierfür das Alphabet

$$\{c,0,1,L,R\} \tag{21.20}$$

Dabei ist

- c ein Trennzeichen für verschiedene Blöcke unserer Codierung
- 1 verwenden wir zur Darstellung der Zustände
- 0 verwenden wir, wenn δ an einer Stelle nicht definiert ist
- L, R sind die Kopfbewegungen

Wir verwenden folgendes Schema:

ccc	.	c	.	c	.	cc	.	c	.	c	.	cc	.	ccc
1)	Block für Eingabe „B“	2)	„0“	2)	„0“	3)	„B“	1)	„0“	1)	„1“	3)		1)

Dabei gilt folgendes:

1. ccc bezeichnet die Randmarkierung
2. c bezeichnet die Eingabesymbolblockmarkierung
3. cc bezeichnet die Zustandsblockmarkierung

Jeder Zustand erhält einen Zustandsblock. In jedem Zustandsblock gibt es je einen Eingabesymbolblock für die drei möglichen Eingabesymbole B, 0 , 1. Ein Eingabesymbolblock sieht beispielhaft so aus:

c111L1c

Dies bedeutet, dass in den Zustand 3 (111) gegangen, der Kopf nach links bewegt und an die aktuelle Position eine 1 geschrieben werden soll.

Beispiel:

Wir wollen die Kodierung folgender Übergangstabelle haben:

$q_1 1$	$q_2 0R$
$q_2 B$	$q_3 1L$
$q_2 0$	$q_3 1L$
$q_2 1$	$q_2 1R$
$q_3 B$	$q_4 0R$
$q_3 0$	$q_4 0R$
$q_3 1$	$q_3 1L$

Nach dem obigen Schema lautet die Kodierung:

ccc	0c0c11R0	cc	111L1c111L1c11R1	cc	1111R0c1111R0c111L1	cc	0c0c0	ccc
	q_1		q_2		q_3		q_4	

21.2.6 Arbeitsweise der universellen Turing-Maschine

Bei der Arbeit benutzt die virtuelle Turing-Maschine zwei zusätzliche Symbole m_1, m_2 zur Markierung. Dabei markiert m_1 den aktuellen Zustandsblock auf dem Eingabeband (auf dem „cc“) und m_2 das aktuelle Eingabezeichen.

Das Band sieht zu Beginn wie folgt aus (mit Markierung von m_1, m_2):

	m_1		m_2
ccc	...	ccc	Kod(x)

Die Vorgehensweise ist nun die folgende:

1. U sucht m2 und merkt sich das dort gefundene Symbol, z. B. A $\in$ {B,0,1}
2. U sucht im aktuellen Zustandsblock (Merker: m1) die zugehörige Anweisung für A
3. U merkt sich, wodurch A zu ersetzen ist und ob nach rechts oder links gegangen wird
4. U versetzt m1 an den Anfang des Zustandsblocks, der als Nachfolgezustand angegeben wird (Subroutine)
5. U geht zu m2, ändert A und versetzt m2 nach links oder nach rechts
6. Gehe nach 1

Diese Vorgehensweise kann natürlich auch formalisiert werden.

Insgesamt folgt aber

$$\begin{aligned} &U \text{ hält auf } Kod(M,x) \Leftrightarrow M \text{ hält auf } x \\ &U \text{ akzeptiert } (M,x) \Leftrightarrow M \text{ akzeptiert } x \end{aligned} \tag{21.21}$$

In diesem Ablauf haben wir die akzeptierenden Zustände von M nicht betrachtet. Man kann sich aber leicht vorstellen, dass jede Turingmaschine mit mehreren akzeptierenden Zuständen in eine äquivalente Turingmaschine mit nur einem akzeptierenden Zustand umgewandelt werden kann (indem man einfach einen neuen Zustand einführt und alle akzeptierenden Zustände auf diesen verweist). Dann kann die universelle Turingmaschine durch eine geeignete Erweiterung der Kodierung von M leicht feststellen, ob diese in einem akzeptierenden Zustand ist.

21.2.7 Halteproblem für Turing-Maschinen

Wir formulieren das Halteproblem:

Gibt es einen Algorithmus (Turing-Maschine), der für beliebige Paare (M, x), wobei M eine Turing-Maschine und x ein Wort sei, entscheidet, ob M auf x hält?

Wir kodieren wieder, diesmal jedoch *alle* Turing-Maschinen in einem endlichen Alphabet und alle Eingabewerte in $\{0,1\}^*$. In beiden Kodierungen existiert eine lineare (lexikographische Ordnung).

Wir beweisen zunächst einen Hilfssatz:

Lemma:

Sei $L_1 := \left\{ x_i \mid x_i \notin T\left(M_i\right) \right\}$ die Sprache der Wörter mit der Nummer i, die nicht von der i-ten Turing-Maschine akzeptiert werden. Zu dieser Sprache existiert keine Turingmaschine mit T(M) = L_1.

Beweis:

Angenommen, es gäbe eine solche Turingmaschine M mit L_1 = T(M). Daraus folgt automatisch

$$\exists j : M = M_j \wedge L_1 = T(M_j) \tag{21.22}$$

(weil M ja selbst eine Turingmaschine ist und sie daher eine Nummer j besitzen muss). Dann aber gilt

$$\begin{aligned} &x_j \in L_1 \Rightarrow x_j \in T\left(M_j\right) \Rightarrow x_j \notin L_1 \\ &x_j \notin L_1 \Rightarrow x_j \notin T\left(M_j\right) \Rightarrow x_j \in L_1 \end{aligned} \tag{21.23}$$

Beide Fälle sind Widersprüche, so dass die Annahme falsch gewesen sein muss.

Satz (Halteproblem):

Es gibt keinen Algorithmus (Turing-Maschine), der für ein beliebiges Paar (M,x), wobei M eine Turing-Maschine und x ein Wort sei, entscheidet, ob M auf x hält.

Beweis:

Wir nehmen an, dass ein solcher Algorithmus A existiert:

$$A(M,x) = \begin{cases} 0 \Leftrightarrow \text{M hält nicht auf x} \\ 1 \Leftrightarrow \text{M hält auf x} \end{cases} \tag{21.24}$$

Wir behaupten, dass $L_1 := \{x_i \mid x_i \notin T(M_i)\}$ von einer Turing-Maschine akzeptierbar ist. U sei die universelle Turing-Maschine.

Wir konstruieren jetzt eine Turing-Maschine M:

1. Eingabe ist x
2. Durch sukzessives Iterieren der Worte $x_1, x_2, ..., x_n$ bestimmen wir das i mit $x_i = x$. (Wir bestimmen also die Nummer des Wortes x)
3. Durch sukzessives Iterieren erzeugen wir über dem Kodierungsalphabet die Turing-Maschine M_i.
4. Jetzt wenden wir A (M_i, x_i) an.
5. Wenn A $(M_i, x_i) = 0 \Rightarrow$ M akzeptiert x_i.
6. Wenn A $(M_i, x_i) = 1 \Rightarrow$ M akzeptiert nicht x_i
7. Mi akzeptiert nicht $x_i \Rightarrow$ M akzeptiert x_i

Daraus folgt: $T(M) = \{x_i \mid x_i \notin T(M_i)\}$. Das ist aber ein Widerspruch zum obigen Lemma. Damit folgt ein Widerspruch zur Annahme und die gewünschte Turing-Maschine kann nicht existieren.

21.2.8 Rekursive Mengen

Definition (rekursive Menge):

Eine Menge S heißt „*rekursiv*" $\Leftrightarrow$ Es existiert eine Turing-Maschine M, die auf *allen Eingaben* aus Σ^* hält, mit S = T(M). Wichtig ist hierbei die Aussage „auf allen Eingaben". Bildlich kann man sich die Definition wie in Abbildung 40 vorstellen.

Defintion (rekursiv aufzählbar):

Eine Menge heißt „*rekursiv aufzählbar*" :$\Leftrightarrow$ Es existiert eine Turing-Maschine M mit S = T(M).

Diese Situation ist in Abbildung 41 dargestellt.

Satz:

Eine Menge $L \subseteq \Sigma^*$ ist rekursiv $\Leftrightarrow \overline{L} = \Sigma^* - L$ ist rekursiv.

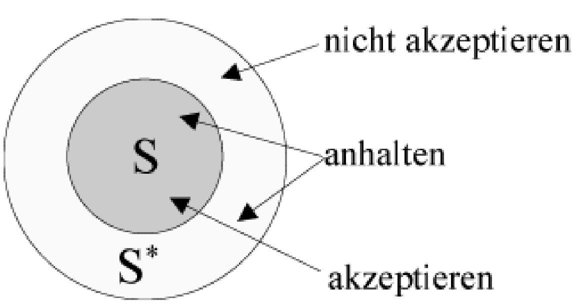


Abbildung 40 - Rekursive Mengen

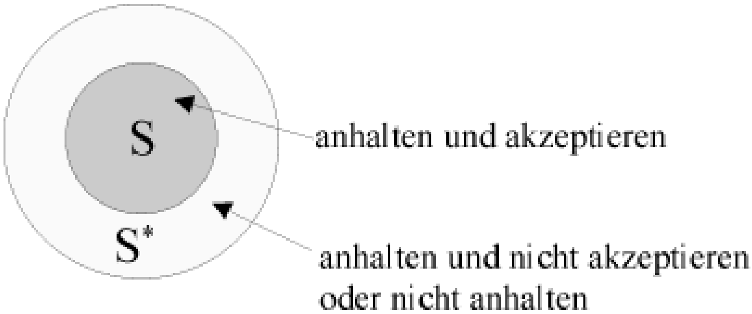


Abbildung 41 - Rekursiv aufzählbare Mengen

Beweis:

L ist rekursiv. Daraus folgt, dass es eine Turing-Maschine $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ gibt mit

- $\forall x \in \Sigma^* : M(x)$ hält
- L = T(M)

Sei $M = (Q_1, \Sigma, \Gamma, \delta_1, q_0, B, F)$ eine neue Turingmaschine, die M simuliere. Wir haben $Q_1 = Q \cup \{q'\}$ mit $q' \notin Q$.

Es gelte nun

$$\begin{gathered}\forall q \in Q, a \in \Gamma: \text{ Wenn } \delta(q,a) \text{ für } q \in Q - F \text{ nicht definiert } : \delta_1(q,a) = q' \\ f_1 := \{q'\}\end{gathered} \quad (21.25)$$

Die Maschine M_1 hält immer!

Weiter gilt $x \in T(M) \Leftrightarrow x \notin T(M_1) \Rightarrow \overline{L} = T(M_1)$.

Damit ist $\overline{L}$ auch rekursiv. Die umgekehrte Richtung zeigt man analog.

Satz:

Es gibt eine rekursiv aufzählbare Menge, deren Komplement nicht rekursiv aufzählbar ist.

Beweis:

Es sei $L = \{x_i \mid x_i \in T(M)\}$. Weiter sei U die modifizierte universelle Turingmaschine[15]. Dann gilt:

$$\begin{gathered}L = T(U) \Rightarrow \text{L ist rekursiv aufzählbar} \\ L = \{x_i \mid x_i \notin T(M_i)\} \text{ist nicht rekursiv aufzählbar!}\end{gathered} \quad (21.26)$$

[15] Modifiziert heißt hier: die i-te Turingmaschine muss erst wie in Abschnitt 21.2.7 beschrieben konstruiert werden.

22 Literaturverzeichnis

1. Uwe Schöning: "Theoretische Informatik - kurzgefaßt", 3. Auflage, 1997
2. Spektrum Akademischer Verlag GmbH Heidelberg, Berlin
3. Skript „Theoretische Informatik II", Prof. Dr. Wotschke Universität Frankfurt
4. Vorlesung „Theoretische Informatik II", Prof. Dr. Wotschke Universität Frankfurt
5. Skript „Theoretische Informatik II", Prof. Dr. Schnittger Universität Frankfurt
6. J. E. Hopcroft und J. D. Ullman: "Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie", Addison-Wesley, 3. Auflage, 1996
7. "Schülerduden: Informatik", Dudenverlag, 3. Auflage, 1997
8. "Schülerduden: Die Mathematik II", Dudenverlag, 3. Auflage, 1991

23 Abbildungsverzeichnis

24 Index