Inhaltsverzeichnis

Kapitel 2

Formale Sprachen und Automaten

2.1 Die Sprachfamilien der Chomsky-Hierarchie

Im Kapitel 1 haben wir mehrere gleichwertige Definitionen für Algorithmen behandelt. Als Grundlage dienten dabei einmal eine spezielle einfache Programmiersprache, die **LOOP**/ **WHILE**-Programme erzeugt, und ein anderes Mal ein spezieller Typ von Maschinen, die TURING-Maschinen. In diesem Kapitel werden wir uns direkt dem Studium von formalen Sprachen bzw. Automaten als Abstraktionen von Programmier- und natürlichen Sprachen bzw. von Computern und Rechenmaschinen zuwenden.
Wir beginnen dabei mit der Definition eines allgemeinen Typs von formalen Grammatiken und Sprachen und geben dann einige wichtige und interessante Spezialfälle an.

2.1.1 Definition der Sprachfamilien

Jede natürliche Sprache basiert auf einer Grammatik, in der die Regeln zusammengestellt sind, nach denen sich syntaktisch richtige Sätze der Sprache bilden lassen. Eine ähnliche Rolle spielen die Handbücher für Programmiersprachen; auch sie enthalten verschiedene Anweisungen und Kommandos, durch deren Anwendung korrekte Programme erzeugt werden.

Die Syntax einer natürlichen Sprachen gibt an, wie ein Satz bzw. Teile eines Satzes aus grammatischen Einheiten aufgebaut werden kann. Wir erwähnen hier beispielhaft die folgenden Konstruktionen.

(Satz) → (Substantivphrase)(Verbphrase)
(Satz) → (Substantivphrase)(Verbphrase)(Objektphrase)
(Substantivphrase) → (Artikel)(Substantiv)
(Verbphrase) → (Verb)(Adverb)

Das erste Konstrukt besagt, dass ein Satz aus einem Substantiv und einem Verb bestehen kann, das zweite entspricht dem vom Englischunterricht her bekannten Aufbau eines Satzes aus Subjekt, Prädikat und Objekt (man sieht, dass für einen Satz verschiedene Zerlegungen in grammatikalische Teile möglich sind). Die beiden letzten Vorschriften sagen, wie eine Substantivphrase bzw. eine Verbphrase weiter zergliedert bzw. wie diese aufgebaut werden können. Weiterhin gibt es eine Zuordnung der Wörter der deutschen Sprache zu Wortarten. Dies kann durch die folgenden Konstruktionen beschrieben werden.

(Substantiv) → Hund
(Substantiv) → Banane
(Artikel) → der
(Artikel) → ein
(Verb) → geht
(Verb) → singt
(Adverb) → langsam

Durch Nacheinanderanwendung der obigen Vorschriften können u. a.

$$\begin{aligned}(\text{Satz}) &\Longrightarrow (\text{Substantivphrase})(\text{Verbphrase})\\ &\Longrightarrow (\text{Substantivphrase})(\text{Verb})(\text{Adverb})\\ &\Longrightarrow (\text{Substantivphrase})\ \text{geht}\ (\text{Adverb})\\ &\Longrightarrow (\text{Substantivphrase})\ \text{geht langsam}\\ &\Longrightarrow (\text{Artikel})(\text{Substantiv})\ \text{geht langsam}\\ &\Longrightarrow \text{der}\ (\text{Substantiv})\ \text{geht langsam}\\ &\Longrightarrow \text{der Hund geht langsam}\end{aligned}$$

und in analoger Weise kann auch

$$(\text{Satz}) \Longrightarrow \ldots \Longrightarrow \text{ein Banane singt langsam}$$

hergeleitet werden. Wir machen darauf aufmerksam, dass der letzte Satz zwar inhaltlich falsch, aber syntaktisch korrekt ist.
Kommen wir nun zu den Programmiersprachen. Hier legt das Programmierhandbuch fest, in welcher Weise das Programm selbst bzw. seine Teilstücke aufgebaut sein können. Als Beispiel geben wir nachfolgend einige Regeln, die sagen, wie Zahlen in einem PASCAL-Programm aussehen können.

(unsigned integer) → (digit) | (digit){digit}
(unsigned real) → (unsigned integer).(digit){digit} | (unsigned integer)E(scale factor)
(scale factor) → (unsigned integer) | (sign) (unsigned integer)
(digit) → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
(sign) → + | –

Hieraus erhalten wir die folgende Sequenz

$$\begin{aligned}(\text{unsigned real}) &\Longrightarrow (\text{unsigned integer})\text{E}(\text{scale factor})\\ &\Longrightarrow (\text{digit})\{\text{digit}\}\text{E}(\text{scale factor})\\ &\Longrightarrow 3\{\text{digit}\}\text{E}(\text{scale factor})\\ &\Longrightarrow 314\text{E}(\text{scale factor})\\ &\Longrightarrow 314\text{E}(\text{sign})(\text{unsigned integer})\\ &\Longrightarrow 314\text{E}–(\text{unsigned integer})\\ &\Longrightarrow 314\text{E}–(\text{digit})\\ &\Longrightarrow 314\text{E}–2\end{aligned}$$

aus der hervorgeht, dass (die Näherung) 3,14 (für π) eine reelle Zahl ist.
Wir stellen folgende Gemeinsamkeiten fest:

- Eigentlich handelt es sich bei den Vorschriften um Ersetzungsregeln. Gewisse Objekte werden durch andere ersetzt.
- Es gibt Objekte, die ersetzt werden (z. B. (Substantivphrase), (unsigned real)), und andere Objekte, die durch die Ersetzungen nicht verändert werden, sondern endgültigen Charakter haben (wie die Wörter der Sprache selbst oder die Ziffern 0,1,2,...,9 und die Zeichen + und –.
- Die Erzeugungen beginnen mit festgelegten Objekten (wie (Satz) oder (program)) und enden, wenn nur noch unveränderliche Objekte vorhanden sind.

Wir werden auf dieser Basis im Folgenden formale Grammatiken und Sprachen definieren. Dabei wollen wir Objekte als Buchstaben eines Alphabets auffassen, und die erzeugten Sätze bzw. Programme bzw. Programmstücke sind dann Wörter über dem Alphabet, das z. B. als Buchstaben alle deutschen Wörter bzw. die Elemente `if`, `while`, Ziffern usw. enthält.
Um die Möglichkeiten zur Wahl von Alphabeten nicht ausufern zu lassen, wollen wir im Folgenden immer annehmen, dass die betrachteten Alphabete (endliche) Teilmengen einer festen abzählbar-unendlichen Menge sind.
Unter einer Sprache über dem Alphabet V verstehen wir im Folgenden stets eine beliebige Teilmenge von V^*. In den folgenden Abschnitte werden verschiedene Möglichkeiten der Beschreibung von (unendlichen) Sprachen durch endliche Objekte untersucht.
Wir beginnen mit der definition einer Grammatik.

Definition 2.1 *Eine Regelgrammatik (oder kurz Grammatik) ist ein Quadrupel*

$$G = (N, T, P, S),$$

wobei

- N *und* T *endliche, disjunkte Alphabete sind, deren Vereinigung wir mit* V *bezeichnen,*
- P *eine endliche Teilmenge von* $(V^* \setminus T^*) \times V^*$ *ist, und*
- $S \in N$ *gilt.*

N ist das Alphabet der Nichtterminale oder Hilfssymbole (wie (Substantivphrase) oder (unsigned real)) und T das der Terminale. Im Folgenden werden wir meist große lateinische Buchstaben zur Bezeichnung der Nichtterminale und kleine lateinische Buchstaben für die Terminale verwenden. Die Elemente aus P heißen Regeln. Meistens werden wir das Paar (α, β) aus P in der Form $\alpha \longrightarrow \beta$ schreiben, da diese Notation der Anwendung von Regeln (in der nächsten Definition) als Ersetzung entspricht. S heißt Axiom oder Startwort (und entspricht (Satz) bzw. (program)).

Definition 2.2 *Sei* $G = (N, T, P, S)$ *eine Regelgrammatik wie in Definition 2.1 beschrieben. Wir sagen, dass aus dem Wort* $\gamma \in V^+$ *das Wort* $\gamma' \in V^*$ *erzeugt wird, wenn*

$$\gamma = \gamma_1 \alpha \gamma_2,\ \gamma' = \gamma_1 \beta \gamma_2,\ \alpha \longrightarrow \beta \in P$$

für gewisse $\gamma_1, \gamma_2 \in V^*$ *gelten. Wir schreiben dann*

$$\gamma \Longrightarrow \gamma'.$$

Entsprechend Definition 2.2 entsteht γ' aus γ, indem ein Teilwort α in γ durch β ersetzt wird, wenn eine Regel $\alpha \longrightarrow \beta$ in P existiert. Die Regeln geben also an, welche lokalen Ersetzungen ausgeführt werden können, um aus einem Wort ein neues zu erzeugen.
Die Anwendung einer Regel nennen wir auch einen Ableitungsschritt oder sagen, dass γ' aus γ direkt abgeleitet oder generiert wird. Falls die bei der Erzeugung verwendete Regel $p = \alpha \longrightarrow \beta$ betont werden soll, so schreiben wir $\gamma \Longrightarrow_p \gamma'$. Durch $\Longrightarrow$ wird offenbar eine Relation, d.h. eine Teilmenge von $V^+ \times V^*$, definiert. Wie üblich kann hiervon der reflexive und transitive Abschluss $\Longrightarrow^*$ gebildet werden, d.h. es gilt

$$\gamma \Longrightarrow^* \gamma'$$

genau dann, wenn es eine natürliche Zahl $n \geq 0$ und Wörter $\delta_0, \delta_1, \delta_2, \ldots, \delta_{n-1}, \delta_n$ mit

$$\gamma = \delta_0 \Longrightarrow \delta_1 \Longrightarrow \delta_2 \Longrightarrow \ldots \Longrightarrow \delta_{n-1} \Longrightarrow \delta_n = \gamma'$$

gibt (im Fall $n = 0$ gilt $\gamma = \gamma'$, und im Fall $n = 1$ haben wir $\gamma \Longrightarrow \gamma'$). Somit gilt $\gamma \Longrightarrow^* \gamma'$ genau dann, wenn γ' durch iterierte Anwendung von (nicht notwendigerweise gleichen) Regeln aus γ entsteht. Gilt $\gamma \Longrightarrow^* \gamma'$, so sagen wir auch, dass γ' aus γ (in mehreren Schritten) ableitbar oder erzeugbar ist.
Ein Wort $w \in V^*$ heißt *Satzform* von G, wenn $S \Longrightarrow^* w$ gilt, d.h. wenn w aus S erzeugt werden kann.

Definition 2.3 *Für eine Grammatik $G = (N, T, P, S)$ aus Definition 2.1 ist die von G erzeugte Sprache $L(G)$ durch*

$$L(G) = \{w \mid w \in T^* \text{ und } S \Longrightarrow^* w\}$$

definiert.

Entsprechend dieser Definition besteht die von G erzeugte Sprache also aus allen Satzformen von G, die nur Terminale enthalten. Ferner zeigt diese Definition die Notwendigkeit der Angabe von S in der Definition 2.1, da nur die aus S in mehreren Schritten ableitbaren Wörter über T die Sprache bilden.
Diese Definition macht auch deutlich, warum die Elemente aus N bzw. T Nichtterminale oder Hilfssymbole bzw. Terminale heißen. Die Elemente aus N werden für die Sprache selbst nicht benötigt, sie erscheinen nur in Zwischenschritten der Ableitung, haben daher Hilfscharakter. Die Terminale dagegen bilden das Alphabet, über dem die Endwörter definiert werden, wobei Endwort so zu verstehen ist, dass aus diesen Wörtern keine weiteren mehr abgeleitet werden können.
Wir betrachten nun einige Beispiele.

Beispiel 2.1 Wir betrachten die Regelgrammatik

$$G_1 = (\{S, A, B\}, \{a, b\}, \{p_1, p_2, p_3, p_4, p_5\}, S)$$

mit

$$p_1 = S \longrightarrow AB,\ p_2 = A \longrightarrow aA,\ p_3 = A \longrightarrow \lambda,\ p_4 = B \longrightarrow Bb,\ p_5 = B \longrightarrow \lambda.$$

Wir zeigen zuerst, dass jede Satzform von G_1 eine der folgenden Formen hat, wobei n und m beliebige natürliche Zahlen sind:

$$S,\ a^nABb^m,\ a^nAb^m,\ a^nBb^m,\ a^nb^m. \qquad (*)$$

Dies gilt offensichtlich für das Startwort S und das einzige daraus in einem Schritt ableitbare Wort AB ($n = m = 0$). Wir betrachten nun ein Wort der Form a^nABb^m. Hierfür ergeben sich nur die folgenden direkten Ableitungen

$$a^nABb^m \Longrightarrow_{p_2} a^naABb^m, \quad a^nABb^m \Longrightarrow_{p_3} a^n\lambda Bb^m,$$
$$a^nABb^m \Longrightarrow_{p_4} a^nABbb^m, \quad a^nABb^m \Longrightarrow_{p_5} a^nA\lambda b^m.$$

Folglich sind aus a^nABb^m nur die Wörter

$$a^{n+1}ABb^m,\ a^nBb^m,\ a^nABb^{m+1},\ a^nAb^m$$

in einem Schritt ableitbar, die alle von der gewünschten Form sind. Analog kann man leicht nachweisen, dass auch alle in einem Schritt aus a^nAb^m bzw. a^nBb^m ableitbaren Wörter von einer der Formen aus $(*)$ sind. Da aus a^nb^m keine Wörter ableitbar sind, ist damit die obige Aussage bewiesen.
Wir beweisen nun, dass sogar jedes Wort der in $(*)$ genannten Form eine Satzform von G_1 ist. Mit Ausnahme von a^nAb^m folgt dies aus der folgenden Ableitung:

$$\begin{aligned} S \ &\Longrightarrow_{p_1} \ \underbrace{AB \Longrightarrow aAB \Longrightarrow aaAB \Longrightarrow \ldots \Longrightarrow a^{n-1}AB}_{(n-1)-malige\ Anwendung\ von\ p_2} \\ &\Longrightarrow_{p_2} \ \underbrace{a^nAB \Longrightarrow a^nABb \Longrightarrow a^nABb^2 \Longrightarrow \ldots \Longrightarrow a^nABb^m}_{m-malige\ Anwendung\ von\ p_4} \\ &\Longrightarrow_{p_3} \ a^nBb^m \Longrightarrow_{p_5} a^nb^m. \end{aligned}$$

Da die von G_1 erzeugte Sprache nur Wörter über $\{a, b\}$ enthält, besteht $L(G_1)$ aus allen Wörtern der Form $(*)$ in $\{a, b\}^*$. Somit gilt

$$L(G_1) = \{a^nb^m \mid n \geq 0, m \geq 0\}.$$

Beispiel 2.2 Es sei

$$G_2 = (\{S\}, \{a, b\}, \{S \longrightarrow aSb, S \longrightarrow ab\}, S).$$

Mittels vollständiger Induktion zeigen wir nun, dass durch $n \geq 1$ Ableitungsschritten genau die Wörter a^nSb^n und a^nb^n aus S erzeugt werden können.
Dies gilt offenbar für $n = 1$, denn aus dem Axiom S werden durch Anwendung der beiden Regel $S \longrightarrow aSb$ bzw. $S \longrightarrow ab$ die Wörter aSb bzw. ab abgeleitet.
Sei nun w ein Wort, das durch n Ableitungsschritte aus S erzeugt wird. Nach Definition muss w dann durch Anwendung einer Regel auf ein Wort v entstehen, wobei sich v in $n - 1$ Schritten erzeugen lässt. Nach Induktionsannahme muss also $v = a^{n-1}Sb^{n-1}$ oder $v = a^{n-1}b^{n-1}$ gelten. Im ersten Fall sind durch Ersetzung von S entsprechend den beiden Regeln die Wörter a^nSb^n und a^nb^n ableitbar; im zweiten Fall enthält v nur Terminale, womit aus v kein Wort mehr abgeleitet werden kann. Somit sind in n Schritten nur a^nSb^n

und $a^n b^n$ erzeugbar. Dies beweist aber gerade die Induktionsbehauptung.
Da die Wörter aus $L(G_2)$ in einer endlichen Anzahl von Schritten abgeleitet werden müssen und nur Terminale enthalten dürfen, folgt

$$L(G_2) = \{a^n b^n \mid n \geq 1\}.$$

Beispiel 2.3 Wir betrachten die Regelgrammatik

$$G_3 = (\{S, A\}, \{a, b\}, \{S \longrightarrow \lambda, S \longrightarrow aS, S \longrightarrow Sb\}, S).$$

Wie in Beispiel 2.1 können wir zeigen, dass die Menge der Satzformen aus allen Wörtern der Form $a^n S b^m$ oder $a^n b^m$ mit $n \geq 0$ und $m \geq 0$ besteht, oder wir beweisen in Analogie zu Beispiel 2.2, dass in $k \geq 1$ Schritten genau die Wörter $a^n S b^m, a^{n-1} b^m, a^n b^{m-1}$ mit $n + m = k$ erzeugt werden können. Daraus ergibt sich

$$L(G_3) = \{a^n b^m \mid n \geq 0, m \geq 0\}.$$

Beispiel 2.4 Es sei

$$G_4 = (\{S, A\}, \{a, b\}, \{S \longrightarrow \lambda, S \longrightarrow aS, S \longrightarrow a, S \longrightarrow A, A \longrightarrow bA, A \longrightarrow b\}, S).$$

In Abbildung 2.1 sind die Anfänge aller möglichen Ableitungen dargestellt, wobei die nach rechts gerichteten Pfeile der Anwendung von $S \longrightarrow aS$ bzw. $A \longrightarrow bA$, die nach oben der von $S \longrightarrow a$, die nach unten der von $S \longrightarrow A$ bzw. $A \longrightarrow b$ und die nach links oben der von $S \to \lambda$ entsprechen. Daraus ist leicht zu ersehen, dass sich erneut

$$L(G_4) = \{a^n b^m \mid n \geq 0, m \geq 0\}$$

ergibt. Ein formaler Beweis wie in den vorangegangenen Beispielen bleibt dem Leser überlassen.

Beispiel 2.5 Es sei die Regelgrammatik

$$G_5 = (\{S, A, B, B', B''\}, \{a, b, c\}, \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}, S)$$

mit

$$p_1 = S \longrightarrow ABA,\ p_2 = AB \longrightarrow aAbB',\ p_3 = AB \longrightarrow abB'',\ p_4 = B'b \longrightarrow bB',$$
$$p_5 = B''b \longrightarrow bB'',\ p_6 = B'A \longrightarrow BAc,\ p_7 = B''A \longrightarrow c,\ p_8 = bB \longrightarrow Bb$$

gegeben. Durch eine Analyse aller möglichen Ableitungen wollen wir $L(G_5)$ bestimmen.
Für $n \geq 0$ sei $w_n = a^n ABb^n Ac^n$.
Wir betrachten zuerst den Fall $n \geq 2$. Die einzigen auf w_n anwendbaren Regeln sind p_2 und p_3.
Fall 1: Anwendung von p_2. Wir erhalten das Wort $a^{n+1} AbB' b^n Ac^n$. Nun ist nur p_4 anwendbar, und die Anwendung dieser Regel liefert $a^{n+1} AbbB' b^{n-1} Ac^n$, d.h. wir haben B' um eine Position nach rechts verschoben. Erneut ist nur p_4 anwendbar, und wir können B' um eine Position weiter nach rechts verschieben. Diese Situation hält an, bis wir das Wort $a^{n+1} Ab^{n+1} B' Ac^n$ erzeugt haben. Nun ist nur p_6 anwendbar, durch deren Anwendung

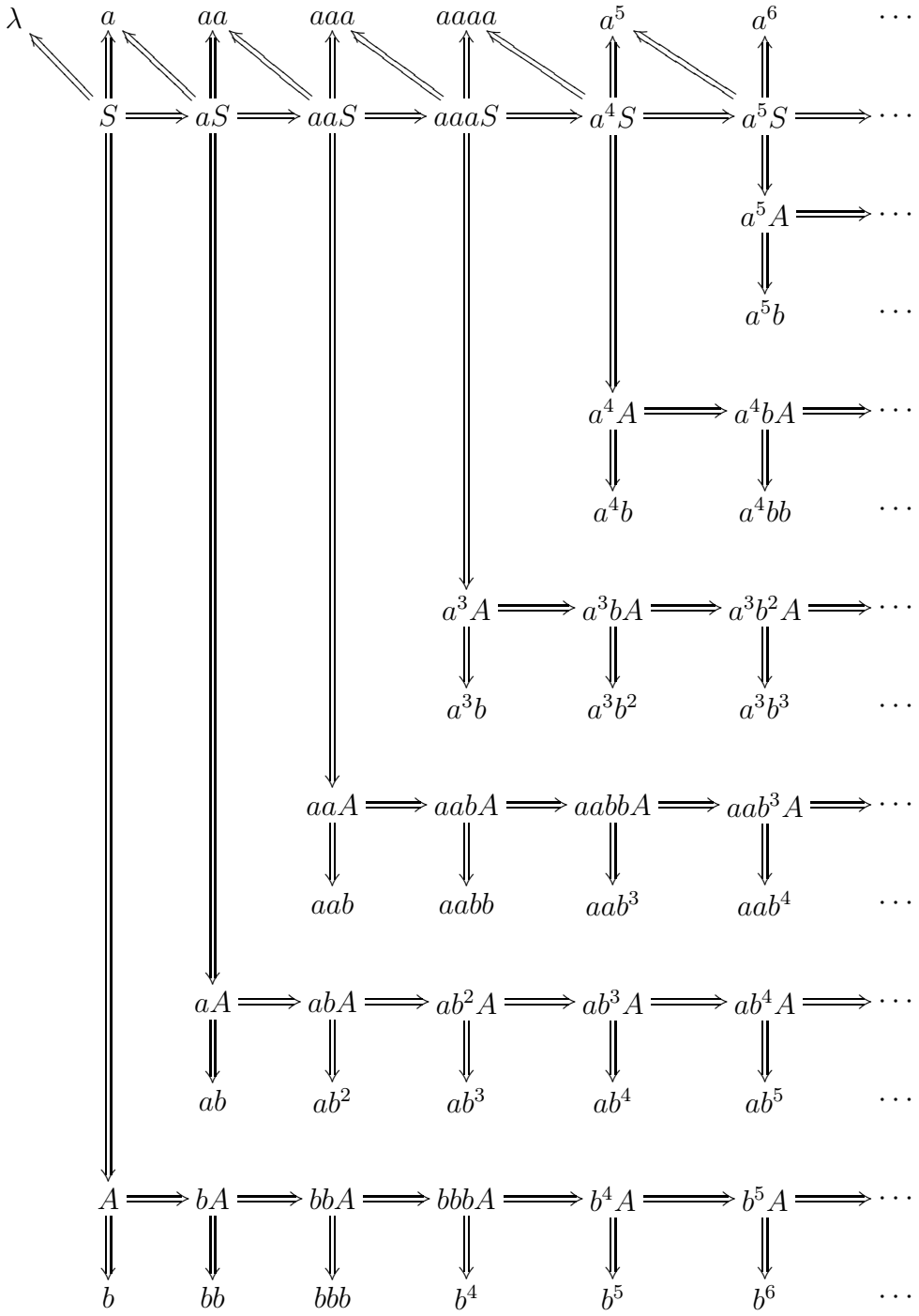


Abbildung 2.1: Ableitungen bez. der Grammatik aus Beispiel 2.4

$a^{n+1}Ab^{n+1}BAc^{n+1}$ entsteht. Jetzt kann nur p_8 angewendet werden, wodurch eine Verschiebung von B um eine Position nach links bewirkt wird. Erneut ist nur diese Verschiebung möglich, bis wir $w_{n+1} = a^{n+1}ABb^{n+1}Ac^{n+1}$ erhalten.
Fall 2: Anwendung von p_3. Wir erhalten das Wort $a^{n+1}bB''b^nAc^n$. Nun ist nur p_5 anwendbar, d.h. B'' wird um eine Position nach rechts verschoben. Diese Situation bleibt erhalten,

bis wir das Wort $a^{n+1}b^{n+1}B''Ac^n$ erzeugt haben. Nun ist nur p_7 anwendbar, durch deren Anwendung $a^{n+1}b^{n+1}c^{n+1}$ entsteht.
Somit wird aus w_n entweder w_{n+1}, womit der eben beschriebene Prozess erneut gestartet werden kann, oder $a^{n+1}b^{n+1}c^{n+1}$ abgeleitet.
Analog kann man sich überlegen, dass w_0 und w_1 nur die Ableitungen

$$w_0 \Longrightarrow^* w_1, w_0 \Longrightarrow^* abc, w_1 \Longrightarrow^* w_2, w_1 \Longrightarrow^* a^2b^2c^2$$

gestatten. Wegen $S \Longrightarrow w_0$ gilt folglich

$$L(G_5) = \{a^nb^nc^n \mid n \geq 1\}.$$

Beispiel 2.6 Wir betrachten die Regelgrammatik

$$G_6 = (\{S, A, B, B', B''\}, \{a, b, c\}, \{p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}, S)$$

mit

$$\begin{aligned} &p_0 = S \to abc,\ p_1 = S \longrightarrow aABbA,\ p_2 = AB \longrightarrow aAbB',\\ &p_3 = AB \longrightarrow abB'',\ p_4 = B'b \longrightarrow bB',\ p_5 = B''b \longrightarrow bB'',\\ &p_6 = B'A \longrightarrow BAc,\ p_7 = B''A \longrightarrow cc,\ p_8 = bB \longrightarrow Bb. \end{aligned}$$

Wie im vorhergehenden Beispiel können wir

$$L(G_6) = \{a^nb^nc^n \mid n \geq 1\}$$

zeigen.

Beispiel 2.7 Wir betrachten die Regelgrammatik $G_7 = (N, T, P, S)$ mit

$$\begin{aligned} N &= \{S\},\\ T &= \{x, y, z, +, -, \cdot, :, (,)\},\\ P &= \{S \longrightarrow (S + S), S \longrightarrow (S - S), S \longrightarrow (S \cdot S), S \longrightarrow (S : S),\\ &\qquad S \longrightarrow x, S \longrightarrow y, S \longrightarrow z\}. \end{aligned}$$

Wir wollen beweisen, dass $L(G_7)$ aus allen exakt geklammerten arithmetischen Ausdrücken mit den Variablen x, y, z (wobei keine Vorrangregeln für die Operationen beachtet werden und auch äußere Klammern mitgeführt werden) besteht.
Hierfür zeigen wir erst, dass jede Satzform, die aus S erzeugt werden kann, ein exakt geklammerter Ausdruck in den Variablen S, x, y, z ist. Dies folgt aber sofort daraus, dass das Axiom ein solcher Ausdruck ist und aus exakt geklammerten Ausdrücken wieder nur solche entstehen, denn die Ersetzung von S durch x, y, z oder $(S \circ S)$ mit $\circ \in \{+, -, \cdot, :\}$ bewahrt exakte Klammerungen.
Wir zeigen nun mittels Induktion über die Anzahl der Variablen, dass *alle* exakt geklammerten Ausdrücke in $L(G_7)$ sind. Für $n = 1$ sind die Variablen x, y, z aus S mittels der Anwendung der Regeln $S \longrightarrow x, S \longrightarrow y, S \longrightarrow z$ direkt erzeugbar. Seien nun $n \geq 2$ und w ein exakt geklammerter Ausdruck mit n Variablen. Dann gilt $w = (w_1 \circ w_2)$ für eine

Operation $\circ \in \{+, -, \cdot, :\}$ und exakt geklammerte Ausdrücke w_1 und w_2, von denen jeder höchstens $n-1$ Variablen enthält. Nach Induktionsannahme gelten damit

$$S \Longrightarrow^* w_1 \quad \text{und} \quad S \Longrightarrow^* w_2.$$

Somit gibt es auch die Ableitung

$$S \Longrightarrow (S \circ S) \Longrightarrow^* (w_1 \circ S) \Longrightarrow^* (w_1 \circ w_2) = w.$$

Damit ist $w \in L(G_7)$ gezeigt.

Beispiel 2.8 In diesem Beispiel wollen eine Regelgrammatik angeben, die alle **LOOP**/-**WHILE**-Programme aus Abschnitt 1.1 erzeugt.
Entsprechend den Definitionen müssen sich alle **LOOP**/**WHILE**-Programme aus dem Startsymbol herleiten lassen. Die Regeln, mittels derer **LOOP**/**WHILE**-Programme erzeugt werden können, sind im Wesentlichen bei der Definition von **LOOP**/**WHILE**-Programmen angegeben worden; es handelt sich um die Grundanweisungen, das Hintereinanderausführen und den **LOOP**- bzw. **WHILE**-Befehl. Wir müssen diesen Prozess nur formal als Grammatik aufschreiben. Dafür verwenden wir das Nichtterminal A als Bezeichnung für ein beliebiges Programm und ersetzen es jeweils durch die zugelassen Befehle; wir haben also A für die Bezeichnungen Π, Π_1 und Π_2 von Programmen zu ersetzen. A ist dann natürlich auch das Axiom, da wir Programme erzeugen wollen. (Wir wählen die Bezeichnung A, da S bereits für die Nachfolgerfunktion vergeben ist.)
Ein Problem bereiten noch die Variablen, da wir davon unendlich viele benötigen, unsere Alphabete der Terminale und Nichtterminale aber endlich sein müssen. Deshalb gehen wir wie folgt vor. Anstelle von x_i verwenden wir die Notation $x[i]$ (wie in Programmiersprachen üblich). Nun muss i eine natürliche Zahl sein, und kann daher durch eine Folge von Ziffern repräsentiert werden. Wir gehen daher von $x[I]$ aus, wobei I ein zusätzliches Nichtterminal ist, aus dem wir alle Ziffernfolgen (ohne führende Nullen) ableiten.
Aus diesen Bemerkungen ergibt sich formal die Regelgrammatik

$$G_8 = (\{A, I, J\}, T, P, A)$$

mit dem Terminalalphabet

$$\begin{aligned} T \;=\; \{&S, P, \mathbf{LOOP}, \mathbf{WHILE}, \mathbf{BEGIN}, \mathbf{END}, :=, \neq, ;, (,) \\ &0, 1, 2, 3, 4, 5, 6, 7, 8, 9, x, [,] \;\} \end{aligned}$$

(man beachte, dass das Semikolon ein Element von T ist, während die Kommata beim Aufschreiben von T als Trennzeichen zwischen den Elementen aus T fungieren) und der Regelmenge

$$\begin{aligned} P \;=\; \{&A \to x[I] := 0,\ A \to x[I] := x[I],\ A \to x[I] := S(x[I]),\ A \to x[I] := P(x[I]), \\ &A \to A; A,\ A \to \mathbf{LOOP}\ x[I]\ \mathbf{BEGIN}\ A\ \mathbf{END}, \\ &A \to \mathbf{WHILE}\ x[I] \neq 0\ \mathbf{BEGIN}\ A\ \mathbf{END}\} \\ &\cup \{I \to z,\ I \to Jz,\ J \to Jz \mid z \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}\} \\ &\cup \{J \to z \mid z \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}\} \end{aligned}$$

(zuerst erzeugen wir aus I die letzte Ziffer mittels $I \to z$ oder $I \to Jz$, wobei z eine beliebige Ziffer ist; nun werden aus J analog die davor stehenden Ziffern erzeugt; bei der abschließenden Terminierung durch $J \to z$ darf dann z nicht 0 sein, da sonst eine führende Null entstehen würde).

Wir führen nun einige spezielle Typen von Regelgrammatiken ein.

Definition 2.4 *Es sei $G = (N, T, P, S)$ eine Regelgrammatik wie in Definition 2.1. Wir sagen,*

- *G ist monoton, wenn für alle Regeln $\alpha \longrightarrow \beta \in P$ die Bedingung $|\alpha| \leq |\beta|$ erfüllt ist, wobei als Ausnahme $S \longrightarrow \lambda$ zugelassen ist, wenn $|\beta'|_S = 0$ für alle Regeln $\alpha' \longrightarrow \beta' \in P$ gilt,*
- *G ist kontextabhängig, wenn alle Regeln in P von der Form $uAv \longrightarrow uwv$ mit $u, v \in V^*$, $A \in N$ und $w \in V^+$ sind, wobei als Ausnahme $S \longrightarrow \lambda$ zugelassen ist, wenn $|\beta'|_S = 0$ für alle Regeln $\alpha' \longrightarrow \beta' \in P$ gilt,*
- *G ist kontextfrei, wenn alle Regeln in P von der Form $A \longrightarrow w$ mit $A \in N$ und $w \in V^*$ sind,*
- *G ist regulär, wenn alle Regeln in P von der Form $A \longrightarrow wB$ oder $A \longrightarrow w$ mit $A, B \in N$ und $w \in T^*$ sind.*

Die monotonen Grammatiken haben – abgesehen von der Ausnahmeregelung – die Eigenschaft, dass bei Anwendung einer Regel die Länge des abgeleiteten Wortes nicht kleiner ist als die des Ausgangswortes, d.h. $\longrightarrow$ ist bezüglich der Wortlänge eine monotone Relation. Bei kontextabhängigen Grammatiken wird bei Anwendung einer Regel $uAv \longrightarrow uwv$ eigentlich nur das Nichtterminal A durch das Wort w ersetzt; aber diese Ersetzung ist nur erlaubt, wenn links bzw. rechts von A das Wort u bzw. v stehen, d.h. es wird die Existenz eines lokalen Kontextes von A für die Ersetzung gefordert. Genau dieser Kontext wird bei kontextfreien Grammatiken nicht gefordert (daher wäre der Begriff „kontextunabhängig“ eigentlich besser, denn A steht in einem Kontext, der aber für die Ersetzung unerheblich ist; es hat sich aber „kontextfrei“ eingebürgert und durchgesetzt).
Reguläre Grammatiken sind entsprechend der Definition 2.4 ein Spezialfall kontextfreier Grammatiken, die durch zusätzliche strukturelle Forderungen an die rechten Seiten der Regeln gekennzeichnet sind.
Da das Leerwort als rechte Seite bei Regeln von Regelgrammatiken, kontextfreien und regulären Grammatiken zugelassen ist, ist klar, dass das Leerwort auch in der erzeugten Sprache liegen kann. Die Ausnahmeregelungen in der Definition monotoner und kontextabhängiger Grammatiken dienen dazu, diese Eigenschaft auch für diese Typen von Grammatiken abzusichern.
Außer den in Definition 2.4 eingeführten Bezeichnungen wird vielfach auch Typ 0 für beliebige Regelgrammatiken, Typ 1 für kontextabhängige, Typ 2 für kontextfreie und Typ 3 für reguläre Grammatiken benutzt.

Wir klassifizieren nun die Grammatiken aus den obigen Beispielen hinsichtlich der Eigenschaften von Definition 2.4.

G_1 ist wegen der Regel $p_3 = A \longrightarrow \lambda$ nicht monoton und nicht kontextabhängig. G_1 ist auch nicht regulär, da die Regel $p_4 = B \longrightarrow Bb$ in der Regelmenge von G_1 existiert. G_1 ist aber offensichtlich kontextfrei.
G_2 ist monoton, kontextabhängig (für alle Regeln gilt $u = v = \lambda$) und kontextfrei, aber nicht regulär.
G_3 ist nicht monoton und nicht kontextabhängig (wegen der gleichzeitigen Existenz der regeln $S \longrightarrow \lambda$ und $S \longrightarrow aS$) und nicht regulär, aber kontextfrei.
G_4 ist regulär und damit auch kontextfrei, aber nicht monoton und nicht kontextabhängig.
G_5 hat keine der in Definition 2.4 gegebenen Eigenschaften.
G_6 ist monoton, aber weder kontextabhängig noch kontextfrei noch regulär. G_7 und G_8 sind monoton, kontextabhängig und kontextfrei, jedoch nicht regulär.

Definition 2.5 *Eine Sprache L heißt monoton (bzw. kontextabhängig, kontextfrei oder regulär), wenn es eine monotone (bzw. kontextabhängige, kontextfreie oder reguläre) Grammatik G mit $L = L(G)$ gibt.*

Nach dieser Definition ist $L = \{a^n b^m \mid n \geq 0, m \geq 0\}$ eine kontextfreie Sprache, denn es gilt $L = L(G_3)$ und G_3 ist eine kontextfreie Grammatik. Jedoch lässt sich aus der Tatsache, dass G_3 keine reguläre Grammatik ist, nicht schließen, dass L keine reguläre Sprache ist. Da nämlich G_4 ebenfalls die Sprache L erzeugt und G_4 eine reguläre Grammatik ist, ist L auch regulär.

Mit $\mathcal{L}(REG)$, $\mathcal{L}(CF)$, $\mathcal{L}(CS)$, $\mathcal{L}(MON)$ und $\mathcal{L}(RE)$ bezeichnen wir die Menge aller Sprachen, die von regulären, kontextfreien, kontextabhängigen, monotonen und beliebigen Regelgrammatiken erzeugt werden.[1]
Wir bemerken zuerst, dass für zwei Typen X und Y von Grammatiken aus dem Fakt, dass jede Grammatik vom Typ X auch eine vom Typ Y ist, die Aussage $\mathcal{L}(X) \subseteq \mathcal{L}(Y)$. Hieraus folgt sofort das folgende Lemma.

Lemma 2.1 $\mathcal{L}(CS) \subseteq \mathcal{L}(MON) \subseteq \mathcal{L}(RE)$ *und* $\mathcal{L}(REG) \subseteq \mathcal{L}(CF) \subseteq \mathcal{L}(RE)$. □

Im nächsten Abschnitt werden weitere Beziehungen zwischen den eingeführten Mengen hergeleitet und festgestellt, ob die Inklusionen in Lemma 2.1 echt oder Gleichheiten sind.

2.1.2 Normalformen und Schleifensätze

Wir werden in diesem Abschnitt zuerst zeigen, dass für die im vorangegangenen Abschnitt eingeführten Typen von Grammatiken jeweils Normalformen existieren, d.h. Grammatiken dieses Typs mit weiteren Einschränkungen an die Regeln, die es aber trotzdem gestatten, jede Sprache dieses Typs von einer Grammatik in Normalform zu erzeugen. Wir benutzen diese Normalformen vor allem als beweistechnische Hilfsmittel und zur Herleitung von Eigenschaften, die uns den Nachweis gestatten, dass gewisse Sprachen nicht durch Grammatiken eines gegebenen Typs erzeugt werden können.
Wir beweisen jeweils nicht nur die Existenz der Normalform, sondern zeigen auch, dass eine Grammatik in Normalform konstruktiv gewonnen werden kann.
Wir beginnen mit Normalformen für monotone Grammatiken.

[1]Die hierbei verwendeten Bezeichnungen *REG, CF, CS, MON, RE* sind Abkürzungen der entsprechenden englischen Wörter regular, context-free, context-sensitive, monotone, recursively enumerable.

Lemma 2.2 *Zu jeder Regelgrammatik $G = (N, T, P, S)$ kann eine Regelgrammatik $G' = (N', T, P', S)$ konstruiert werden, bei der alle Regeln aus P' von der Form $\alpha \longrightarrow \beta$ mit $\alpha, \beta \in (N')^*$ oder $A \longrightarrow a$ mit $A \in N', a \in T$ sind und für die $L(G) = L(G')$ gilt. Ist außerdem G eine monotone, kontextabhängige bzw. kontextfreie Grammatik, so ist auch G' monoton, kontextabhängig bzw. kontextfrei.*

Beweis. Für jedes Terminal a sei a' ein neues Symbol (das also weder in N noch in T liegt). Ferner sei für $a \neq b, a, b \in T$ auch $a' \neq b'$. Wir setzen

$$N' = N \cup \{a' : a \in T\}.$$

Ist $w = x_1 x_2 \dots x_n$ ein Wort aus V^*, so sei $w' = y_1 y_2 \dots y_n$ das Wort mit

$$y_i = \begin{cases} x_i & \text{für } x_i \in N \\ x_i' & \text{für } x_i \in T \end{cases}$$

für $1 \leq i \leq n$. Wir definieren nun die Regelmenge von G' durch

$$P' = \{\alpha' \longrightarrow \beta' : \alpha \longrightarrow \beta \in P\} \cup \{a' \longrightarrow a : a \in T\}.$$

Wir beweisen nun $L(G') = L(G)$.
Sei dazu zuerst $w \in L(G)$. Dann gibt es in G eine Ableitung

$$S = w_0 \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \dots \Longrightarrow w_n = w.$$

Entsprechend der Konstruktion von P gibt es dann in G' die Ableitung

$$S = w_0' \Longrightarrow w_1' \Longrightarrow w_2' \Longrightarrow \dots w_n' = w' = v_0 \Longrightarrow v_1 \Longrightarrow v_2 \Longrightarrow \dots \Longrightarrow v_m = w,$$

bei der wir für den Übergang von w_i' zu w_{i+1}' stets die Regel $\alpha' \longrightarrow \beta' \in P'$ anwenden, wenn w_{i+1} aus w_i durch Anwendung der Regel $\alpha \longrightarrow \beta \in P$ entstanden ist und die direkten Ableitungen $v_j \Longrightarrow v_{j+1}$ durch Anwendung einer Regel der Form $a' \longrightarrow a$ geschehen. Daher gilt auch $w \in L(G')$, womit $L(G) \subseteq L(G')$ gezeigt ist.
Sei nun $x \in L(G')$. Dann gibt es für x eine Ableitung der Form

$$S = x_0' \Longrightarrow x_1' \Longrightarrow x_2' \Longrightarrow \dots \Longrightarrow x_n' = x' = y_0 \Longrightarrow y_1 \Longrightarrow y_2 \Longrightarrow \dots \Longrightarrow y_m = x$$

(eine Ableitung dieser Form entsteht aus einer beliebigen Ableitung von w, indem man die Reihenfolge der angewendeten Regeln so vertauscht, dass im ersten Teil nur Regeln der Form $\alpha' \longrightarrow \beta'$ und im zweiten Teil nur Regeln der Form $a' \longrightarrow a$ angewendet werden, wodurch auch abgesichert ist, dass die im ersten Teil der Ableitung entstehenden Satzformen sämtlich nur Symbole aus N' enthalten). Wenn wir nun die Reihenfolge der Regelanwendung nicht ändern, aber stets statt $\alpha' \longrightarrow \beta' \in P'$ die Regel $\alpha \longrightarrow \beta \in P$ benutzen, so erhalten wir die Ableitung

$$S = x_0 \Longrightarrow x_1 \Longrightarrow x_2 \Longrightarrow \dots \Longrightarrow x_n = x$$

in G. Dies beweist $x \in L(G)$ und damit $L(G') \subseteq L(G)$.
Aus den beiden nachgewiesenen Inklusionen folgt $L(G) = L(G')$.

Bei der Konstruktion von P' wird eine Regel $\alpha \longrightarrow \beta$ mit $|\alpha| \leq |\beta|$ in eine Regel $\alpha' \longrightarrow \beta'$ mit $|\alpha'| \leq |\beta'|$ überführt, da $|\alpha| = |\alpha'|$ und $|\beta| = |\beta'|$ gelten. Damit ist G' monoton, wenn G monoton ist. Analog ist sofort zu sehen, dass Regeln der Form $uAv \longrightarrow uwv$ bzw. $A \longrightarrow w$ wieder in Regeln dieser Form übergehen. Hieraus folgt sofort die Aussage über die Kontextabhängigkeit und Kontextfreiheit. □

Satz 2.3 *Zu jeder monotonen Grammatik $G = (N, T, P, S)$ kann eine monotone Grammatik $G' = (N', T, P', S)$ konstruiert werden, bei der jede Regel aus P' von einer der Formen*

$$A \longrightarrow BC,\ A \longrightarrow B,\ AB \longrightarrow CB,\ AB \longrightarrow AC \text{ oder } A \longrightarrow a$$

mit $A, B, C \in N', a \in T$ oder $S \longrightarrow \lambda$ ist und für die $L(G) = L(G')$ gilt.

Beweis. Wegen Lemma 2.2 können wir annehmen, dass alle Regeln von P von der Form $\alpha \longrightarrow \beta$ oder $A \longrightarrow a$ mit $\alpha, \beta \in N^+, A \in N, a \in T$ (oder $S \longrightarrow \lambda$) sind.
Jeder Regel aus P werden wir nun eine Menge von Regeln und Nichtterminalen so zuordnen, dass die Mengen P' und N' mit den gewünschten Eigenschaften als Vereinigung aller dieser Mengen von Regeln bzw. aller dieser Mengen von Nichtterminalen und N entstehen. Die dabei neu eingeführten Symbole sollen stets paarweise verschieden sein und nicht in $N \cup T$ liegen.
Sei $p = X_1X_2 \ldots X_n \longrightarrow Y_1Y_2 \ldots Y_m$ eine Regel aus P.
Fall 1. $n = 1$ *und* $m \leq 2$. Dann setzen wir

$$P_p = \{p\} \quad \text{und} \quad N_p = \emptyset,$$

d.h. wir übernehmen die Regel p in P' und führen keine neue Hilfssymbole ein.
Fall 2. $n = 1$ *und* $m \geq 3$. Dann setzen wir

$$N_p = \{C_{p,1}, C_{p,2}, \ldots, C_{p,m-2}\}$$

und

$$P_p = \{X_1 \longrightarrow Y_1C_{p,1}, C_{p,1} \longrightarrow Y_2C_{p,2}, \ldots, C_{p,m-3} \longrightarrow Y_{m-2}C_{p,m-2}, C_{p,m-2} \longrightarrow Y_{m-1}Y_m\}.$$

Fall 3. $n \geq 2$. Dann gilt auch $m \geq 2$. Wir setzen nun

$$N'_p = \{C_{p,1}, C_{p,2}, \ldots, C_{p,n}, D_p\}$$

und

$$\begin{aligned} P'_p \;=\; \{&X_1X_2 \longrightarrow C_{p,1}X_2, C_{p,1}X_2 \longrightarrow C_{p,1}C_{p,2}, C_{p,2}X_3 \longrightarrow C_{p,2}C_{p,3},\\ &\ldots, C_{p,n-2}X_{n-1} \longrightarrow C_{p,n-2}C_{p,n-1}, C_{p,n-1}X_n \longrightarrow C_{p,n-1}C_{p,n},\\ &C_{p,1}C_{p,2} \longrightarrow Y_1C_{p,2}, C_{p,2}C_{p,3} \longrightarrow Y_2C_{p,3},\\ &\ldots, C_{p,n-2}C_{p,n-1} \longrightarrow Y_{n-2}C_{p,n-1}, C_{p,n-1}C_{p,n} \longrightarrow Y_{n-1}C_{p,n},\\ &Y_{n-1}C_{p,n} \longrightarrow Y_{n-1}D_p, D_p \longrightarrow Y_nY_{n+1} \ldots Y_m\}\,. \end{aligned}$$

Die Mengen N_p und P_p entstehen nun aus N'_p und P'_p indem wir $D_p \in N'_p$ und $D_p \longrightarrow Y_nY_{n+1} \ldots Y_m \in P'_p$ entsprechend Fall 2 durch Nichtterminale und Regeln mit einer rechten Seite der Länge ≤ 2 ersetzen.

Wir konstruieren $G' = (N', T, P', S)$ durch

$$N' = N \cup \bigcup_{p \in P} N_p \qquad \text{und} \qquad P' = \bigcup_{p \in P} P_p.$$

Aus der Konstruktion ist sofort zu sehen, dass alle Regeln von P' von der geforderten Form sind.

Sei nun $v = w_1 X_1 X_2 \dots X_n w_2$ mit $w_1, w_2 \in V^*$ und $n \geq 2$ eine Satzform von G. Durch Anwendung von p entsteht $v' = w_1 Y_1 Y_2 \dots Y_m w_2$. In G' haben wir dann die folgende Ableitung

$$\begin{aligned}
v &\Longrightarrow w_1 C_{p,1} X_2 X_3 \dots X_n w_2 \Longrightarrow w_1 C_{p,1} C_{p,2} X_3 \dots X_n w_2 \\
&\Longrightarrow \dots \Longrightarrow w_1 C_{p,1} C_{p,2} \dots C_{p,n-1} X_n w_2 \Longrightarrow w_1 C_{p,1} C_{p,2} \dots C_{p,n-1} C_{p,n} w_2 \\
&\Longrightarrow w_1 Y_1 C_{p,2} \dots C_{p,n-1} C_{p,n} w_2 \Longrightarrow w_1 Y_1 Y_2 \dots C_{p,n-1} C_{p,n} w_2 \\
&\Longrightarrow \dots \Longrightarrow w_1 Y_1 Y_2 \dots Y_{n-1} C_{p,n} w_2 \\
&\Longrightarrow w_1 Y_1 Y_2 \dots Y_{n-1} D_{p,n} w_2 \Longrightarrow w_1 Y_1 Y_2 \dots Y_{n-1} Y_n Y_n D_{p,n+1} w_2 \\
&\Longrightarrow w_1 Y_1 Y_2 \dots Y_{n-1} Y_n Y_{n+1} D_{p,n+2} w_2 \Longrightarrow \dots \\
&\Longrightarrow w_1 Y_1 Y_2 \dots Y_{n-1} Y_n Y_{n+1} \dots Y_{m-1} D_{p,m} w_2 \\
&\Longrightarrow w_1 Y_1 Y_2 \dots Y_{n-1} Y_n Y_{n+1} \dots Y_{m-1} Y_m w_2 = v',
\end{aligned}$$

wobei wir die Regeln aus P_p genau in der in Fall 3 angegebenen Reihenfolge anwenden. Damit ist gezeigt, dass wir die Anwendung von p in G durch Anwendung der Regeln aus P_p in G' simulieren können. Analoges gilt auch in den Fällen 1 und 2. Damit kann jede Ableitung in G in G' simuliert werden.

Wir zeigen nun, dass bis auf die Reihenfolge in der Anwendung von Regeln in G' nur derartige Simulationen möglich sind. Dies sieht man wie folgt ein: Wenden wir auf v die Regel $X_1 X_2 \longrightarrow C_{p,1} X_2$ an, so können wir auf die entstehende Satzform $v_1 = w_1 C_{p,1} X_2 \dots X_n w_2$ nur die Regel $C_{p,1} X_2 \longrightarrow C_{p,1} C_{p,2}$ aus P_p anwenden. Wir setzen dann die Ableitung mittels Regeln aus P_p wie oben fort oder durch Anwendung von $C_{p,1} C_{p,2} \longrightarrow Y_1 C_{p,2}$ fort, wodurch $w_1 Y_1 C_{p,2} X_3 \dots X_n w_2$ entsteht. Auf letztere Satzform ist nur $C_{p,2} X_3 \longrightarrow C_{p,2} C_{p,3}$ anwendbar, wodurch $w_1 Y_1 C_{p,2} C_{p,3} X_4 \dots X_n w_2$ generiert wird. Auch nun gibt es die Möglichkeit durch Regeln aus P_p das Symbol $C_{p,2}$ durch Y_2 oder X_4 durch $C_{p,4}$ zu ersetzen. Man erkennt also, dass bis auf die Reihenfolge der Regeln schließlich $w_1 Y_1 \dots Y_{n-1} D_p w_2$ erzeugt wird. Nun sind die folgenden anwendbaren Regeln stets eindeutig bestimmt, und wie oben wird v' abgeleitet.

Wir haben noch zu diskutieren, was passiert, wenn auf eine Satzform, die während dieser Simulation entsteht, eine Regel angewendet wird, die nicht zu P_p gehört und mindestens eines der Symbole $X_1, X_2, X_3, \dots, X_n$ verändert. Wir diskutieren dies nur für v_1; die Überlegungen bei den anderen Satzformen sind ähnlich. Es ist leicht zu sehen, dass die Regeln zur Änderung von Symbolen aus $N_p \setminus \{D_p\}$ (und mindestens das in v_1 vorkommende $C_{p,1} \in N_p$ ist zu ändern, damit die Ableitung auf ein Wort über T führt) ein weiteres Symbol aus N_p einführt. Damit kann v_1 nur dann in ein terminales Wort überführt werden, wenn nach einigen Schritten nur noch $D_{p,m}$ in der Satzform ist und $D_p \longrightarrow Y_n Y_{n+1} \dots Y_m$ angewendet wird. Dies erfordert aber, dass alle Regeln aus P_p angewendet wurden und damit die Anwendung von p in G simuliert wurde.

Fall 2. $n = 1$ *und* $m \geq 3$. Dann setzen wir

$$N_p = \{C_{p,1}, C_{p,2}, \ldots, C_{p,m-2}\}$$

und

$$P_p = \{X_1 \longrightarrow Y_1 C_{p,1}, C_{p,1} \longrightarrow Y_2 C_{p,2}, \ldots, C_{p,m-3} \longrightarrow Y_{m-2} C_{p,m-2}, C_{p,m-2} \longrightarrow Y_{m-1} Y_m\}.$$

Da somit in G' alle direkten Ableitungen in G simuliert werden können und nur Simulationen von Ableitungen in G möglich sind, gilt für Wörter w, w' über $N \cup T$, dass $w \Longrightarrow_G^* w'$ genau dann gilt, wenn auch $w \Longrightarrow_{G'}^* w'$ gültig ist. Hieraus folgt $S \Longrightarrow_G^* w$ mit $w \in T^*$ gilt genau dann, wenn $S \Longrightarrow_{G'}^* w$ gültig ist. Dies impliziert $L(G) = L(G')$. □

Folgerung 2.4 $\mathcal{L}(MON) = \mathcal{L}(CS)$.

Beweis. Am Ende von Abschnitt 2.1.1 wurde bereits bemerkt, dass $\mathcal{L}(CS) \subseteq \mathcal{L}(MON)$ gilt.
Wir haben also nur $\mathcal{L}(MON) \subseteq \mathcal{L}(CS)$ zu zeigen, d.h. wir müssen nachweisen, dass jede monotone Sprache auch kontextabhängig ist.
Sei L eine monotone Sprache. Dann gibt es eine monotone Grammatik G mit $L = L(G)$. Nach Satz 2.3. gibt es dann eine monotone Grammatik G', deren Regeln alle von kontextabhängiger Form sind, d.h. G' ist kontextabhängig. Wegen $L = L(G) = L(G')$ ist L folglich eine kontextabhängige Sprache. □

Entsprechend Satz 2.3 wird jede kontextfreie Sprache durch eine Grammatik erzeugt, die nur Regeln der Form

$$A \to BC, A \to B, A \to \lambda \text{ und } A \to a \text{ mit } A, B, C \in N, a \in T$$

hat. Wir zeigen nun, dass auch die Regeln der Form $A \to \lambda$ eliminiert werden können, wobei wir dann natürlich die gleiche Ausnahmeregelung zulassen müssen, die uns schon von den monotonen oder kontextabhängigen Grammatiken geläufig ist.

Lemma 2.5 *Zu jeder kontextfreien Grammatik* $G = (N, T, P, S)$ *existiert eine kontextfreie Grammatik* $G' = (N', T, P', S')$ *derart, dass*
i) P' *keine Regel der Form* $A \longrightarrow \lambda$ *mit* $A \neq S'$ *enthält,*
ii) $|w|_{S'} = 0$ *für alle Regeln* $A \longrightarrow w \in P'$ *gilt, und*
iii) $L(G) = L(G')$ *ist.*

Beweis. Wir konstruieren als erstes zu der Grammatik $G = (N, T, P, S)$ eine kontextfreie Grammatik $G'' = (N'', T, P'', S')$, die die Bedingung ii) und $L(G) = L(G'')$ erfüllt. Dazu fügen wir zu N ein neues Nichtterminal S' hinzu, d.h. wir setzen $N'' = N \cup \{S'\}$. Weiterhin erweitern wir die Regelmenge durch $P'' = P \cup \{S' \to S\}$. Bedingung ii) gilt dann nach Definition. Da alle Ableitungen in G'' von der Form $S'' \Longrightarrow S \Longrightarrow^* w$ sind, haben wir auch $L(G'') = L(G)$.
Es sei

$$M = \{A : A \in N'', A \Longrightarrow^* \lambda\}.$$

Mit jeder Regel

$$q'' = A \longrightarrow v_1 A_1 v_2 A_2 \dots v_m A_m v_{m+1}$$

mit

$$m \geq 0, A_1, A_2, \dots, A_m \in N'', v_1, v_2, \dots, v_{m+1} \in T^*$$

assoziieren wir die Menge $P_{q''}$ aller Regeln der Form

$$A \longrightarrow v_1 X_1 v_2 X_2 \dots v_m X_m v_{m+1} \neq \lambda,$$

für die

$$X_i = A_i \text{ für } A_i \notin M \quad \text{und} \quad X_i \in \{A_i, \lambda\} \text{ für } A_i \in M$$

für $1 \leq i \leq m$ gilt. Aufgrund dieser Definition kann keine Menge $P_{q''}$ eine Regel der Form $Y \longrightarrow \lambda$ enthalten. Damit ist es nicht möglich das Leerwort unter Verwendung von Regeln aus $P_{q''}$ zu erzeugen. Deshalb setzen wir

$$\overline{P} = \begin{cases} \{S' \longrightarrow \lambda\} & \text{falls } S' \in M \\ \emptyset & \text{sonst} \end{cases}.$$

Weiterhin definieren wir $G' = (N', T, P', S')$ durch

$$N' = N'' \quad \text{und} \quad P' = \overline{P} \cup \bigcup_{q'' \in P''} P_{p''}.$$

Wir bemerken, dass bei der Konstruktion von P' aus P'' die Eigenschaft ii) erhalten geblieben ist, und dass P' nach Konstruktion die Eigenschaft i) hat.
Wir zeigen jetzt, dass auch die Bedingung iii) erfüllt ist. Dafür reicht es $L(G'') = L(G')$ zu zeigen.

Zuerst beweisen wir mittels vollständiger Induktion über die Anzahl der Ableitungsschritte, dass für jedes Nichtterminal A und jedes Wort $x \in T^+$ mit $A \Longrightarrow_{G''}^* x$ auch $A \Longrightarrow_{G'}^* x$ gilt.
Sei $n = 1$. Jede direkte Ableitung ist in beiden Grammatiken von der Form $A \Longrightarrow v$, bei der in beiden Fällen die Regel $A \longrightarrow v$ angewendet wird. Somit ist der Induktionsanfang gezeigt.
Sei nun x ein in $n \geq 2$ Schritten aus A ableitbares terminales Wort. Dann gilt

$$A \Longrightarrow_{G''} v_1 A_1 v_2 A_2 \dots v_m A_m v_{m+1} \Longrightarrow_{G''}^* v_1 x_1 v_2 x_2 \dots v_m x_m v_{m+1} = x,$$

wobei die Ableitungen $A_i \Longrightarrow_{G''}^* x_i$ für $1 \leq i \leq m$ sämtlich aus weniger als n Schritten bestehen. Wir unterscheiden nun zwei Fälle:
Fall 1. $x_i \neq \lambda$. Dann setzen wir $X_i = A_i$ und haben nach Induktionsannahme $X_i = A_i \Longrightarrow_{G'}^* x_i$.
Fall 2. $x_i = \lambda$. Dann gilt $A_i \in M$ und wir setzen $X_i = \lambda$.
Nach Konstruktion gibt es in P' die Regel $A \longrightarrow v_1 X_1 v_2 X_2 \dots v_m X_m v_{m+1}$ und wir erhalten in G' die Ableitung

$$A \Longrightarrow_{G'} v_1 X_1 v_2 X_2 \dots v_m X_m v_{m+1} \Longrightarrow_{G'}^* v_1 x_1 v_2 x_2 \dots v_m x_m v_{m+1},$$

wobei wir für $x_i = \lambda$ einfach $X_i = x_i = \lambda$ und für $x_i \neq \lambda$ die Ableitungen $X_i \Longrightarrow^*_{G'} x_i$ benutzen.
Betrachten wir die gerade bewiesene Aussage für $A = S$, so ist jedes vom Leerwort verschiedene Wort aus $L(G'')$ auch in G' ableitbar. Damit gilt $L(G'') \setminus \{\lambda\} \subseteq L(G') \setminus \{\lambda\}$. Da durch $\overline{P}$ gesichert ist, dass $\lambda \in L(G'')$ genau dann gilt, wenn $\lambda \in L(G')$ ist, ist sogar $L(G'') \subseteq L(G')$ gültig.
Wir zeigen nun wiederum mittels vollständiger Induktion die Umkehrung, d.h., dass jede Ableitung $A \Longrightarrow^*_{G'} y$ eines terminalen Wortes y auch eine Entsprechung $A \Longrightarrow^*_{G''} y$ findet. Der Induktionsanfang ergibt sich wie oben.
Sei daher $A \Longrightarrow^*_{G'} y$ eine Ableitung aus $n \geq 2$ Schritten. Dann gilt

$$A \Longrightarrow v_1X_1v_2X_2 \dots v_mX_mv_{m+1} \Longrightarrow^*_{G'} v_1x_1v_2x_2 \dots v_mx_mv_{m+1},$$

wobei für $X_i = \lambda$ auch $x_i = \lambda$ ist, und für $X_i \neq \lambda$ ist $X_i \Longrightarrow^*_{G'} x_i$ eine Ableitung mit weniger als n Schritten. Nach Konstruktion der Regel $A \longrightarrow v_1X_1v_2X_2 \dots v_mX_mv_{m+1}$ aus P' gibt es dann eine Ableitung $A_i \Longrightarrow^* \lambda = x_i$, falls $X_i = \lambda$ ist, und nach Induktionsvoraussetzung gilt auch $A_i \Longrightarrow^*_{G''} x_i$ für $X_i \neq \lambda$. Deshalb existiert in G'' die Ableitung

$$A \Longrightarrow_{G''} v_1A_1v_2A_2 \dots v_mA_mv_{m+1} \Longrightarrow^*_{G'} v_1x_2v_2x_2 \dots v_mx_mv_{m+1}.$$

Hiervon ausgehend zeigt man wie oben $L(G') \subseteq L(G'')$. □

Um den Beweis vollständig konstruktiv zu haben, müssen wir eine Konstruktion für die Menge M der Nichtterminale, aus denen das Leerwort abgeleitet werden kann, angeben. Wir setzen

$$\begin{aligned}
M_0 &= \emptyset, \\
P_0 &= P, \\
M_i &= M_{i-1} \cup \{A : A \in N'', A \to \lambda \in P_{i-1}\}, \\
P_i &= \{A \to w_1w_2 \dots w_{n+1} : A \to w_1A_1w_2A_2 \dots w_nA_nw_{n+1} \in P_{i-1} \\
&\quad n \geq 0, w_j \in (N'' \setminus M_i)^* \text{ für } 1 \leq j \leq n+1, A_j \in M_i \text{ für } 1 \leq j \leq n\} \cup P_{i-1}
\end{aligned}$$

für $i \geq 1$.
Wir zeigen zuerst mittels Induktion $M_i \subseteq M$ für $i \geq 0$.
Für $i = 0$ und $i = 1$ ist dies nach Konstruktion klar.
Es sei $i \geq 2$ und $A \in M_i$. Falls $A \in M_{i-1}$, so gilt schon nach Induktionsvoraussetzung $A \in M$.
Für $A \in M_i \setminus M_{i-1}$, $i \geq 2$, gibt es eine Regel $A \to A_1A_2 \dots A_n$ in P mit $A_j \in M_{i-1}$ für $1 \leq j \leq n$. Dies ist wie folgt zu sehen. Nach Definition gibt es $A \to \lambda$ in P_{i-1}. Folglich gibt es in P_{i-2} eine Regel $A \to B_1B_2 \dots B_r$ mit $B_k \in M_{i-1}$ für $1 \leq k \leq r$. In P_{i-3} gibt es dann eine Regel $A \to u_1B_1u_2B_2 \dots u_rB_ru_{r-1}$, wobei die u_k, $1 \leq k \leq r+1$, Wörter über M_{i-2} sind. So fortfahrend gewinnen wir eine Regel $A \to A_1A_2 \dots A_n$, wobei jedes A_j, $1 \leq j \leq n$ in einer Menge M_{j_t} mit $1 \leq j_t \leq i-1$ liegt. Wegen $M_{j_t} \subseteq M_{i-1}$ folgt obige Aussage. Da nach Induktionsvoraussetzung $A_j \in M$ für $1 \leq j \leq n$ gilt, gibt es die Ableitung

$$A \Longrightarrow A_1A_2 \dots A_n \Longrightarrow^* \lambda A_2A_3 \dots A_n \Longrightarrow^* \lambda\lambda A_3 \dots A_n \Longrightarrow^* \lambda^n = \lambda,$$

woraus $A \in M$ folgt.

Sei nun $A \in M$. Wir betrachten eine Ableitung $A \Longrightarrow^* \lambda$. In keiner Satzform dieser Ableitung kann ein Terminal vorkommen, die Satzformen sind also alle Wörter über N''. Durch Umordnen der Ableitungsschritte können wir eine Ableitung

$$A = w_0 \Longrightarrow^* w_1 \Longrightarrow^* w_2 \Longrightarrow^* \ldots \Longrightarrow^* w_m = \lambda$$

erreichen, bei der $w_{i-1} \Longrightarrow^* w_i$ dadurch entsteht, dass alle Nichtterminale aus w_{i-1} entsprechend einer Regel ersetzt werden. Offenbar gilt dann $w_{m-1} \in M_1^*$, da die darin enthaltenen Nichtterminale in einem Ableitungsschritt durch das Leerwort ersetzt werden. Für ein Nichtterminal B aus w_{m-2} gilt daher $B \to \lambda$ oder $B \to w \in M_1^+$, womit sich $B \in M_1$ oder $B \in M_2$ und damit sicher $B \in M_2$ ergibt. So fortfahrend erhalten wir $w_{m-3} \in M_3^*$, $w_{m-4} \in M_4^*$ und schließlich $A = w_0 = w_{m-m} \in M_m$.
Aus dem bisher Bewiesenem folgt

$$M = \bigcup_{i \geq 0} M_i.$$

Entsprechend den obigen Definitionen impliziert $M_i = M_{i+1}$ sofort $P_i = P_{i+1}$ und dann

$$M_i = M_{i+1} = M_{i+2} = \ldots \quad \text{und} \quad P_i = P_{i+1} = P_{i+2} = \ldots$$

Da außerdem stets $M_i \subseteq M_{i+1}$ gilt, tritt die Gleichheit spätestens bei M_t ein. Somit ergibt sich

$$M_t = \bigcup_{i \geq 0} M_i = M.$$

Beispiel 2.9 Wir illustrieren die eben beschriebene Konstruktion anhand der Grammatik

$$G = (\{S, A, B\}, \{a, b\}, \{S \to SA, S \to \lambda, A \to aAb, A \to B, B \to \lambda\}, S)\,.$$

Wir bemerken, dass

$$L(G) = \{a^{n_1} b^{n_1} a^{n_2} b^{n_2} \ldots a^{n_k} b^{n_k} : k \geq 0, n_i \geq 0, 1 \leq i \leq k\}$$

gilt, da durch die ersten beiden Regeln eine beliebige Anzahl von A's erzeugt wird, von denen jedes eine Sprache der Form $\{a^n b^n : n \geq 0\}$ erzeugt.
Es ergeben sich dann

$$\begin{aligned}
& N'' = N \cup \{S'\} = \{S, A, B, S'\}, \\
& P'' = \{S' \to S, S \to SA, S \to \lambda, A \to aAb, A \to B, B \to \lambda\} \\
& M_0 = \emptyset \text{ und } P_0 = P''\,, \\
& M_1 = \{S, B\} \text{ und } P_1 = \{S' \to \lambda, S \to A, S \to \lambda, A \to aAb, A \to \lambda, B \to \lambda\} \cup P''\,, \\
& M_2 = \{S, B, S', A\} = N'' \\
& N' = N'' = \{S', S, A, B\}, \\
& \overline{P} = \{S' \longrightarrow \lambda\}, \\
P' \;=\; & \overline{P} \cup \{S' \to S, S \to SA, S \to A, S \to S, A \to aAb, A \to ab, A \to B\}\,.
\end{aligned}$$

Man sieht sofort, dass P' offenbar überflüssige Regeln enthält. Dies trifft auf $S \to S$ zu, da diese Regel keine Änderung bei ihrer Anwendung bewirkt, und auf $A \to B$ zu, da P' keine Regeln enthält, die B auf der rechten Seite haben. Wir werden diese Regeln aber hier nicht streichen, da dies der Algorithmus im Beweis von Lemma 2.5 nicht vorsieht.

Es ist offenbar, dass – mit Ausnahme der eventuell existierenden Regel $S \longrightarrow \lambda$ – für alle anderen Regel $A \longrightarrow w \in P'$ bei der in Lemma 2.5 konstruierten Grammatik G' die Beziehung $w \in (N' \cup T)^+$ und damit $|w| \geq 1 = |A|$ gilt. Dies bedeutet, dass G' eine monotone Grammatik ist. Somit erhalten wir das folgende Resultat.

Folgerung 2.6 $\mathcal{L}(CF) \subseteq \mathcal{L}(MON)$. □

Wir zeigen nun, dass die in Satz 2.3 zugelassenen Regeln der Form $A \longrightarrow B$ mit $A, B \in N$ ebenfalls eliminiert werden können.

Lemma 2.7 *Zu jeder kontextfreien Grammatik $G = (N, T, P, S)$ kann eine kontextfreie Grammatik $G' = (N, T, P', S)$ so konstruiert werden, dass P' keine Regel der Form $A \longrightarrow B$ mit $A, B \in N$ enthält und $L(G) = L(G')$ gilt.*

Beweis. Für ein Nichtterminal A definieren wir

$$M_A = \{B : B \Longrightarrow_G^* A, B \in N\}$$

(man beachte, dass nach Definition stets $A \in M_A$ gilt). Für eine Regel $p = A \to w$ mit $w \notin N$ setzen wir

$$P_p = \{B \to w : B \in M_A\}$$

(d.h. wir ersetzen eine Ableitung

$$B \Longrightarrow B_1 \Longrightarrow B_2 \Longrightarrow \ldots \Longrightarrow B_k = A \Longrightarrow w$$

durch eine Regel $B \to w$). Wir setzen nun

$$P' = \bigcup_{p \in P} P_p.$$

Offensichtlich erfüllt P' nach Konstruktion die geforderte Bedingung. Die Gültigkeit von $L(G) = L(G')$ lässt sich nun in Analogie zum Beweis von Lemma 2.5. zeigen. □

Wenn wir die löschenden Regeln $A \to \lambda$ schon beseitigt haben, so kann $A \Longrightarrow^* B$ nur durch Regel der Form $C \to D$ gewonnen werden. Da Ableitungen $X \Longrightarrow^* X$ gestrichen werden können, sind damit nur endlich viele Ableitungen $A \Longrightarrow^* B$ möglich, womit die Mengen M_A konstruktiv ermittelt werden können.

Beispiel 2.10 Wenden wir die im Beweis von Lemma 2.7 gegebene Konstruktion auf Beispiel 2.9 an, so erhalten wir

$$M_B = \{B, A, S, S'\},\ M_A = \{A, S, S'\},\ M_S = \{S, S'\} \text{ und } M_{S'} = \{S'\}$$

und daher

$$\begin{aligned} P_{S' \to \lambda} &= \{S' \to \lambda\}, \\ P_{S \to SA} &= \{S \to SA, S' \to SA\}, \\ P_{A \to aAb} &= \{A \to aAb, S \to aAb, S' \to aAb\}, \\ P_{A \to ab} &= \{A \to ab, S \to ab, S' \to ab\} \end{aligned}$$

und die gesamte Regelmenge ergibt sich als Vereinigung der vier vorstehenden Mengen.

Wir geben nun die Normalform an, die auf N. CHOMSKY zurückgeht und durch Kombination der vorstehenden Normalform gewonnen werden kann.

Satz 2.8 *Zu jeder kontextfreien Grammatik $G = (N, T, P, S)$ kann eine kontextfreie Grammatik $G' = (N', T, P', S')$ so konstruiert werden, dass P' nur Regeln der Form*

$$A \longrightarrow BC \quad \textit{und} \quad A \longrightarrow a \quad \textit{mit} \quad A, B, C \in N',\ a \in T$$

enthält, wobei $S' \longrightarrow \lambda$ als Ausnahme zugelassen ist, falls S' in keiner rechten Seite einer Regel aus P' vorkommt, und $L(G) = L(G')$ gilt.

Beweis. Durch Nacheinanderausführung der Konstruktionen in den Beweisen von Lemma 2.2, Satz 2.3, Lemma 2.5 und Lemma 2.7 erreichen wir eine Grammatik, die keine Regeln der Form $A \longrightarrow w$ mit $|w| > 2$ oder $w = \lambda$ bei $A \neq S$ und keine der Form $A \longrightarrow B$ mit Nichtterminalen A und B enthält. □

Wir bemerken, dass bei Grammatiken in CHOMSKY-Normalform eine nichtleeres Wort $w \in L(G)$ der Länge n durch genau $(2n-1)$ Ableitungsschritte gewonnen wird, und zwar $n-1$ Anwendungen von Regeln der Form $A \to BC$ zur Erzeugung eines Wortes der Länge n aus n Nichtterminalen, auf die dann n Regeln der Form $A \to a$ angewendet werden.

Wir geben nun noch eine Normalform für reguläre Grammatiken.

Satz 2.9 *Zu jeder regulären Grammatik $G = (N, T, P, S)$ kann eine reguläre Grammatik $G' = (N', T, P', S')$ so konstruiert werden, dass P' nur Regeln der Form*

$$A \longrightarrow aB \quad \textit{und} \quad A \longrightarrow a \quad \textit{mit} \quad A, B \in N',\ a \in T$$

enthält, wobei $S' \to \lambda$ als Ausnahme zugelassen ist, falls P' keine Regel der Form $A \to aS'$ enthält, und $L(G) = L(G')$ gilt.

Beweis. Entsprechend Lemma 2.5 und 2.7 können wir ohne Beschränkung der Allgemeinheit annehmen, dass die Regelmenge P der gegebenen Grammatik $G = (N, T, P, S)$ unter Beachtung der Ausnahmeregel $S \longrightarrow \lambda$ und den damit verbundenen Bedingungen nur Regeln der Form $A \longrightarrow wB$ und $A \longrightarrow w$ mit $A, B \in N$, $w \in T^+$ enthält.
Mit der Regel

$$p = A \longrightarrow a_1 a_2 \ldots a_n B \text{ mit } a_1, a_2, \ldots, a_n \in T$$

assozieren wir nun die Menge

$$N_p = \{B_{p,1}, B_{p,2}, \ldots, B_{p,n-1}\}$$

zusätzlicher Nichtterminale und die Menge

$$\begin{aligned} P_p \;=\; \{&A \longrightarrow a_1 B_{p,1}, B_{p,1} \longrightarrow a_2 B_{p,2}, B_{p,2} \longrightarrow a_3 B_{p,3}, \ldots \\ &\ldots, B_{p,n-2} \longrightarrow a_{n-1} B_{p,n-1}, B_{p,n-1} \longrightarrow a_n B\} \end{aligned}$$

von Regeln. Für die Regel

$$q = A \longrightarrow a_1 a_2 \dots a_n \text{ mit } a_1, a_2, \dots, a_n \in T$$

setzen wir ebenfalls

$$N_q = \{B_{q,1}, B_{q,2}, \dots, B_{q,n-1}\}$$

und

$$\begin{aligned} P_q \;=\; \{&A \longrightarrow a_1 B_{q,1}, B_{q,1} \longrightarrow a_2 B_{q,2}, B_{q,2} \longrightarrow a_3 B_{q,3}, \dots \\ &\dots, B_{q,n-2} \longrightarrow a_{n-1} B_{q,n-1}, B_{q,n-1} \longrightarrow a_n\}. \end{aligned}$$

Hierbei seien alle neu eingeführten Symbole wieder paarweise voneinander verschieden. Wir definieren dann $G' = (N', T, P', S)$ durch

$$N' = N \cup \bigcup_{r \in P} N_r \quad \text{und} \quad P' = \bigcup_{r \in P} P_r \cup \overline{P},$$

wobei $\overline{P}$ erneut genau dann die leere Menge ist, wenn $S \longrightarrow \lambda$ nicht in P liegt und sonst nur aus dieser Regel besteht. Es ist leicht zu sehen, dass durch die Anwendung der Regeln aus P_r in der in der Definition angegebenen Reihenfolge zu einer Simulation der Anwendung von r führt, und umgekehrt jede Anwendung einer Regel $A \longrightarrow a_1 B_{r,1}$ die Simulation von r zur Folge hat. Daher gilt $L(G) = L(G')$. □

Wir geben nun zwei Folgerungen aus den in Satz 2.8 und Satz 2.9 gegebenen Normalformen an, die es uns dann gestatten, zu beweisen, dass gewisse Sprachen nicht kontextfrei bzw. nicht regulär sind.
Für reguläre Sprachen leistet der folgende Satz das Gewünschte.

Satz 2.10 *Es sei L eine reguläre Sprache. Dann gibt es eine (von L abhängige) Konstante k derart, dass es zu jedem Wort $z \in L$ mit $|z| \geq k$ Wörter u, v, w gibt, die den folgenden Eigenschaften genügen:*
i) $z = uvw$,
ii) $|uv| \leq k$, $|v| > 0$, *und*
iii) $uv^i w \in L$ *für alle* $i \geq 0$.

Beweis. Wegen Satz 2.9 können wir annehmen, dass $L = L(G)$ für eine reguläre Grammatik $G = (N, T, P, S)$ gibt, deren Regelmenge P (mit Ausnahme von vielleicht $S \longrightarrow \lambda$) nur Regeln der Form $A \longrightarrow aB$ und $A \longrightarrow a$ mit $A, B \in N$ und $a \in T$ enthält. Wir setzen dann $k = |N| + 1$.
Aufgrund der Form der Regeln aus P gibt es für ein Wort

$$z = a_1 a_2 \dots a_n \text{ mit } a_i \in T \text{ für } 1 \leq i \leq n \text{ und } n \geq k$$

eine Ableitung

$$\begin{aligned} S = A_0 \;\Longrightarrow\; & a_1 A_1 \Longrightarrow a_1 a_2 A_2 \Longrightarrow a_1 a_2 a_3 A_3 \Longrightarrow \dots \\ & \Longrightarrow a_1 a_2 \dots a_{n-1} A_{n-1} \Longrightarrow a_1 a_2 \dots a_{n-1} a_n = z. \end{aligned}$$

Dann enthält die Menge $\{A_0, A_1, A_2, \dots, A_{k-1}\}$ wegen der Wahl von k ein Nichtterminal A doppelt. Es sei $A = A_i = A_j$ mit $0 \leq i < j \leq k - 1$. Wir setzen

$$u = a_1 a_2 \dots a_i,\ v = a_{i+1} a_{i+2} \dots a_j \text{ und } w = a_{j+1} a_{j+2} \dots a_n.$$

Man sieht sofort, dass die Bedingungen i) und ii) erfüllt sind.
Mit den eingeführten Bezeichnungen erhält die obige Ableitung von z die folgende Form

$$S = A_0 \Longrightarrow^* uA \Longrightarrow^* uvA \Longrightarrow^* uvw = z,$$

und wir haben überdies für $i \geq 2$ die Ableitungen

$$S = A_0 \Longrightarrow^* uA \Longrightarrow^* uvA \Longrightarrow^* uvvA \Longrightarrow^* uvvvA \Longrightarrow^* \ldots \Longrightarrow^* uv^iA \Longrightarrow^* uv^iw \in T^*$$

und für $i = 0$ die Ableitung $S \Longrightarrow^* uA \Longrightarrow^* uw \in T^*$. Hieraus folgt $uv^iw \in L(G) = L$ für $i \geq 0$, womit auch Bedingung iii) nachgewiesen ist. □

Wir benutzen die Aussage von Satz 2.10, um zu zeigen, dass die kontextfreie Sprache

$$L = \{a^nb^n : n \geq 1\}$$

aus Beispiel 2.2 nicht regulär ist.
Wir zeigen dies indirekt. Wir nehmen also an, dass L regulär ist. Dann sei k die Konstante aus Satz 2.10 und $z = a^kb^k$. Dann gibt es eine Zerlegung $z = uvw$ von z mit

$$|uv| \leq k,\ |v| > 0, \text{ und } uv^iw \in L \text{ für alle } i \geq 1. \qquad (*)$$

Aus den beiden erstgenannten Bedingungen und $z = uvw$ folgen

$$u = a^r,\ v = a^s \text{ und } w = a^{k-r-s}b^k \text{ mit } r \geq 0 \text{ und } s \geq 1.$$

Damit folgt

$$uv^iw = a^ra^{is}a^{k-r-s}b^k = a^{k+(i-1)s}b^k.$$

Wegen der Form der Wörter in L ist daher $uv^iw \notin L$ für $i \geq 2$. Dies widerspricht aber der oben abgeleiteten Aussage in $(*)$.
Somit haben wir das folgende Lemma bewiesen.

Lemma 2.11 $L = \{a^nb^n : n \geq 1\} \in \mathcal{L}(CF) \setminus \mathcal{L}(REG)$. □

Um ein analoges Vorgehen für kontextfreie Sprachen zu ermöglichen, benötigen wir den Begriff des Ableitungsbaumes, der aber auch zur Veranschaulichung von ableitungen sehr geeignet ist. Es seien eine kontextfreie Grammatik $G = (N, T, P, S)$ in der Normalform aus Lemma 2.5 (P enthält also keine Regeln $A \longrightarrow \lambda$), eine Satzform $w \neq \lambda$ von G und eine Ableitung von w gegeben. Der zugehörige Ableitungsbaum t wird mittels vollständiger Induktion über die Anzahl der Ableitungsschritte wie folgt definiert.
Wir werden t als Paar (K, E) beschreiben, wobei K die Menge der Knoten und E die der Kanten bezeichnet. Wir werden die Konstruktion so gestalten, dass S die Wurzel des Baumes sein wird und die Blätter beim Lesen von links nach rechts die Satzform w ergeben.
Sei $n = 0$. Dann handelt es sich um die „Ableitung“ $S \Longrightarrow^* S$ von $w = S$. Der Ableitungsbaum ist für $n = 0$ der Baum, der keine Kanten enthält und dessen einziger Knoten S ist. S ist dann sowohl Wurzel als auch Blatt.
Sei $n = 1$. Dann hat die Ableitung die Form $S \Longrightarrow w$, wobei die Regel $S \longrightarrow w$ angewendet wird. Sei $w = x_1x_2 \ldots x_m$ mit $x_i \in N \cup T$. Dann wird die Menge K der Knoten von t

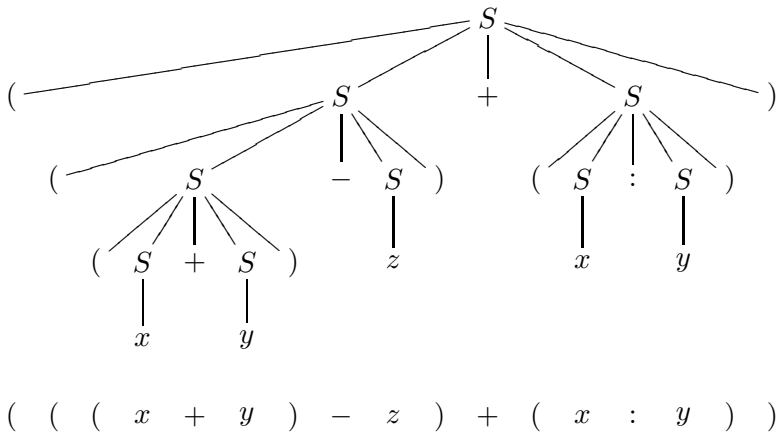
durch die Symbole $S, x_1, x_2, \ldots, x_m$ gebildet, und die Menge E besteht aus allen Kanten (S, x_i), $1 \leq i \leq m$. S ist dabei die Wurzel des Baumes, und $x_1, x_2, \ldots, x_m$ sind die Blätter von t. Dabei ordnen wir die Kanten so an, dass wir $w = x_1 x_2 \ldots x_m$ erhalten, wenn wir die Blätter von links nach rechts lesen.
Sei $n \geq 2$. Dann gibt es eine Ableitung

$$S \Longrightarrow^* u = y_1 y_2 \ldots y_s A z_1 z_2 \ldots z_r \Longrightarrow y_1 y_2 \ldots y_s x_1 x_2 \ldots x_m z_1 z_2 \ldots z_r = w,$$

wobei $x_i, y_j, z_k \in N \cup T$ für $1 \leq i \leq m, 0 \leq j \leq s, 0 \leq k \leq r$ gilt. Die Ableitung $S \Longrightarrow^* u$ besteht dabei aus $n-1$ Schritten, und der zu ihr gehörende Ableitungsbaum $t' = (K', E')$ hat daher die Wurzel S und die Blätter ergeben von links nach rechts gelesen u. Wir konstruieren nun $t = (K, E)$ durch die Setzungen

$$K = K' \cup \{x_1, x_2, \ldots, x_m\} \quad \text{und} \quad E = E' \cup \{(A, x_i) : 1 \leq i \leq m\},$$

wobei wir die neuen Kanten wieder so anordnen, dass die Blätter von links nach rechts gelesen gerade w ergeben.
Zur Illustration geben wir den Ableitungsbaum für das Wort $(((x+y)-z)+(x:y))$, das von der in Beispiel 2.6 gegebenen Grammatik G_6 erzeugt wird. Dabei schreiben wir unter den Baum noch einmal die Blätter, um zu dokumentieren, dass sie von links nach rechts gelesen die zur Diskussion stehende Satzform ergeben.

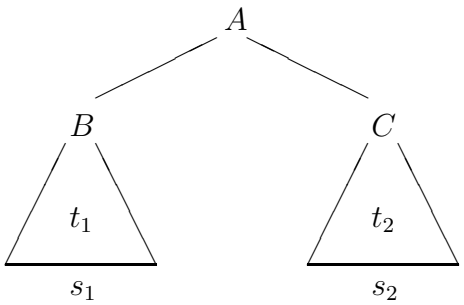


Wir geben jetzt ein Analogon zu Satz 2.10 für kontextfreie Sprachen.

Satz 2.12 *Es sei L eine kontextfreie Sprache. Dann gibt es eine (von L abhängige) Konstante k derart, dass es zu jedem Wort $z \in L$ mit $|z| \geq k$ Wörter u, v, w, x, y gibt, die folgenden Eigenschaften genügen:*
i) $z = uvwxy$,
ii) $|vwx| \leq k$, $|v| + |x| > 0$, *und*
iii) $uv^i wx^i y \in L$ *für alle* $i \geq 0$.

Beweis. Wegen Satz 2.8 können wir annehmen, dass $L = L(G)$ für eine kontextfreie Grammatik $G = (N, T, P, S)$ in CHOMSKY-Normalform gilt. Es sei $n = |N|$. Dann setzen wir $k = 2^n$.

Es sei $A \Longrightarrow^* s \in T^*$ eine Ableitung einer nichtleeren[2] Satzform s, deren zugehöriger Ableitungsbaum die Tiefe m hat. Wir zeigen zuerst mittels vollständiger Induktion über die Tiefe m des Ableitungsbaumes, dass dann $|s| < 2^m$ gilt.
$m = 1$. Die Ableitung kann nur aus einem Schritt bestehen und ist folglich aufgrund der CHOMSKY-Normalform $A \Longrightarrow a$ für ein $a \in T$. Dies bedeutet aber $s = \lambda$ oder $s = a$, woraus sofort $|s| \leq 1 < 2 = 2^1$ folgt. Damit ist der Induktionsanfang gezeigt.
Sei nun $A \Longrightarrow w$ eine Ableitung mit einem Ableitungsbaum t der Tiefe $m \geq 2$. Dann hat t wegen der CHOMSKY-Normalform die Form



wobei t_1 und t_2 Ableitungsbäume mit einer maximalen Tiefe $m-1$ sind und $s_1 s_2 = s$ gilt. Nach Induktionsannahme gilt dann

$$|s| = |s_1| + |s_2| < 2^{m-1} + 2^{m-1} = 2^m,$$

womit auch die Induktionsbehauptung gezeigt ist.
Wir benutzen die gerade bewiesene Aussage für Wörter $z \in L$ mit $|z| \geq k$. Sie liefert, dass der zu z gehörige Ableitungsbaum t' entsprechend der obigen Wahl von k eine Tiefe $m \geq n + 1$ hat. Damit hat t' die Form gemäß Abbildung 2.2.

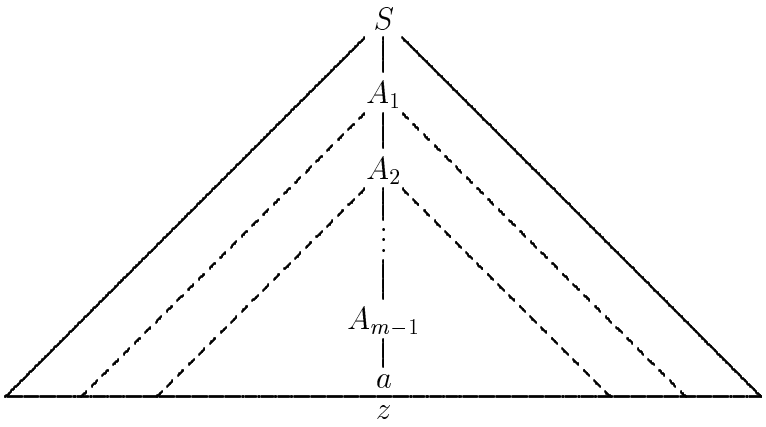


Abbildung 2.2:

Nun müssen wegen $m - 1 \geq n$ mindestens zwei Elemente aus $\{S, A_1, A_2, \ldots, A_{m-1}\}$ identisch sein. Sei A dieses Nichtterminal. Damit ergibt sich für t' dann die Form gemäß Abbildung 2.3.

[2]Da nur Wörter $z \in L(G)$ betrachtet werden, deren Länge größer als k ist, kann auf die Betrachtung des Leerwortes der Länge 0 verzichtet werden.

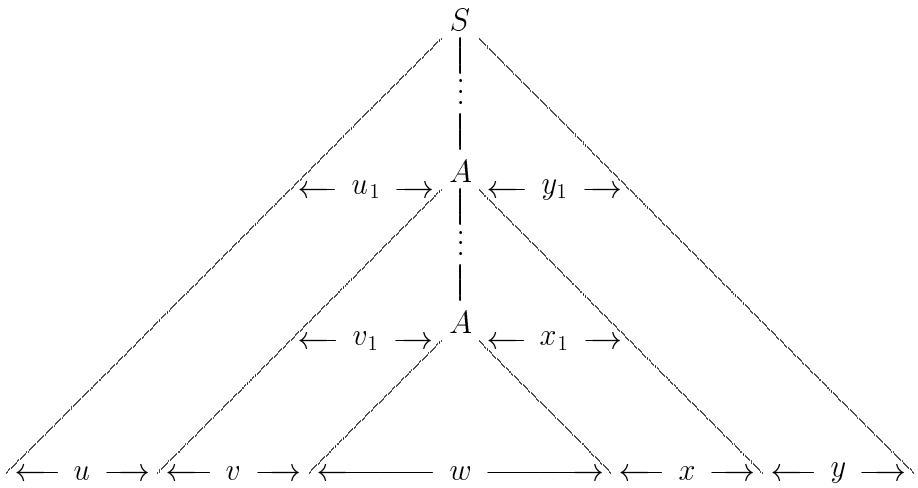


Abbildung 2.3:

Dabei gilt $vx \neq \lambda$, da G eine Grammatik in CHOMSKY-Normalform ist. Weiterhin können wir ohne Beschränkung der Allgemeinheit annehmen, dass $|vwx| \leq k$ ist, da sonst im Ableitungsbaum zu $A \Longrightarrow^* vwx$ ein Weg existiert, auf dem ein Nichtterminal A' doppelt auftritt, und wir könnten dann mit A' anstelle von A argumentieren. Damit sind die Bedingungen i) und ii) nachgewiesen.
Ferner entnehmen wir dem Ableitungsbaum t' auch die Existenz der folgenden Ableitungen:

$$S \Longrightarrow^* uAy, \qquad A \Longrightarrow^* vAx, \qquad A \Longrightarrow^* w.$$

Damit gibt es auch die Ableitung

$$\begin{aligned} S \Longrightarrow^* uAy \Longrightarrow^* uvAxy \Longrightarrow^* uvvAxxy \Longrightarrow^* uvvvAxxxy \Longrightarrow^* \ldots \\ \ldots \Longrightarrow^* uv^iAx^iy \Longrightarrow^* uv^iwx^iy \end{aligned}$$

für $i \geq 0$ (für $i = 1$ entsteht gerade z). Da diese Ableitung zu einem Wort aus T^* führt, gilt $uv^iwx^iy \in L(G) = L$ für $i \geq 0$, womit auch Bedingung iii) nachgewiesen ist. □

Die in Satz 2.10 und 2.12 gegebenen Aussagen heißen *Schleifensätze* (oder auch *Pumping-Sätze*), da sie im Wesentlichen besagen, dass gewisse Ableitungen wie eine Schleife beliebig oft hintereinander ausgeführt werden können (oder einige Teilwörter „aufgepumpt“ werden können, z.B. v zu v^i).

Wir benutzen nun Satz 2.12, um ein zu Lemma 2.11 analoges Resultat herzuleiten.

Lemma 2.13 $L = \{a^nb^nc^n : n \geq 1\} \in \mathcal{L}(MON) \setminus \mathcal{L}(CF)$.

Beweis. $L \in \mathcal{L}(MON)$ folgt aus Beispiel 2.6.
Wir nehmen nun an, dass L kontextfrei ist. Nach Satz 2.12 gibt es dann eine Konstante k und für $z = a^kb^kc^k$ eine Zerlegung $z = uvwxy$ mit den in Satz 2.12 genannten Eigenschaften. Wir betrachten im Folgenden nur den Fall $v \neq \lambda$; die Überlegungen für $v = \lambda, x \neq \lambda$ verlaufen analog.
Wir unterscheiden die folgenden Fälle (wegen $|vwx| \leq k$ ist diese Fallunterscheidung vollständig):

Fall 1. $v = a^r b^s$ mit $r \geq 1, s \geq 0$. Wegen $|vwx| \leq k$ enthält vwx kein Vorkommen von c. Damit enthält uv^2wx^2y mindestens $k + r > k$ Vorkommen des Buchstaben a, aber nur k Vorkommen von c. Aufgrund der Form der Wörter in L, ergibt sich daraus $uv^2wx^2y \notin L$ im Widerspruch zur Eigenschaft iii) aus Satz 2.12.
Fall 2. $v = b^s c^t$ mit $s \geq 1, t \geq 0$. Dann enthält vwx kein Vorkommen von a, und daher gelten $|uv^2wx^2y|_a = k$ und $|uv^2wx^2y|_b \geq k + s > k$, womit sich in Analogie zum Fall 1 ein Widerspruch ergibt.
Fall 3. $v = c^t$ mit $t \geq 1$. Erneut enthält vwx kein Vorkommen von a, und daher gelten $|uv^2wx^2y|_a = k$ und $|uv^2wx^2y|_c \geq k + t > k$, womit sich in Analogie zum Fall 1 ein Widerspruch ergibt. □

Wir kombinieren nun die Aussagen der Lemmata 2.1, 2.11 und 2.13 und der Folgerungen 2.4 und 2.6 und erhalten den folgenden Satz. Man entnimmt ihm, dass die in Definition 2.4 eingeführten Sprachmengen eine Hierarchie bilden, die nach N. CHOMSKY benannt wird.

Satz 2.14 $\mathcal{L}(REG) \subset \mathcal{L}(CF) \subset \mathcal{L}(CS) = \mathcal{L}(MON) \subseteq \mathcal{L}(RE)$. □

Entsprechend Satz 2.14 ist also nur noch die Bestimmung der genauen Relation zwischen $\mathcal{L}(RE)$ und $\mathcal{L}(MON)$ offen. Wir werden die Klärung dieses Problems erst im Kapitel 2.4 herbeiführen.

2.2 Sprachen als akzeptierte Wortmengen

Zur Beschreibung von Sprachen haben wir im vorangehenden Abschnitt Grammatiken benutzt; bei diesen werden die Worte der Sprache mittels eines Ableitungsprozesses aus einem Startwort generiert. Ein grundsätzlich anderes Vorgehen liegt der Beschreibung von Sprachen durch Automaten zugrunde. Hier wird ein Wort als Eingabe verwendet und der Automat sagt „ja“, falls das Wort zu der Sprache gehört, und „nein“, falls das Wort nicht zu der Sprache gehört. Dies erinnert an die Funktionen, die mit Entscheidungsproblemen verbunden sind und im ersten Kapitel (insbesondere in Abschnitt 1.2) untersucht wurden. Wir werden hier eine Modifikation des dortigen Vorgehens betrachten, die sich von der in Kapitel 1 dadurch unterscheidet, dass die Antwort „ja“ oder „nein“ nicht der Ausgabe des Automaten entnommen wird, sondern mittels der Zustände gegeben wird, da die Ausgabe in diesem Zusammenhang nicht von Bedeutung ist.

2.2.1 TURING-Maschinen als Akzeptoren

Wir formalisieren den oben beschriebenen Ansatz.

Definition 2.6 *Eine akzeptierende* TURING-*Maschine M ist ein Sechstupel*

$$M = (X, Z, z_0, Q, \delta, F)\,,$$

wobei X, Z, z_0, Q und δ wie in Definition 1.10 gegeben sind und $F \subseteq Q$ gilt. Die von M akzeptierte Sprache $T(M)$ wird durch

$$T(M) = \{w : w \in X^*,\ (\lambda, z_0, w) \models^* (v_1, q, v_2) \text{ für ein } q \in F\}$$

definiert.

Literaturverzeichnis

[1] J.Albert, Th.Ottmann: Automaten, Sprachen und Maschinen für Anwender. B.-I.-Wissenschaftsverlag, 1983.

[2] A.Aho, J.E.Hopcroft, J.D.Ullman: The Design and Analysis of Algorithms. Reading, Mass., 1974.

[3] A.Aho, R.Sethi, J.D.Ullman: Compilerbau. Band 1 und 2, Addison-Wesley, 1990.

[4] A.Asteroth, Ch.Baier: Theoretische Informatik. Pearson Studium, 2002.

[5] L.Balke, K.H.Böhling: Einführung in die Automatentheorie und Theorie formaler Sprachen. B.-I.-Wissenschaftsverlag, 1993.

[6] W.Bucher, H.Maurer: Theoretische Grundlagen der Programmiersprachen. B.-I.-Wissenschaftsverlag, 1983.

[7] J.Carrol, D.Long: Theory of Finite Automata (with an Introduction to Formal Languages). Prentice Hall, London, 1983.

[8] E.Engeler, P.Läuchli: Berechnungstheorie für Informatiker. Teubner-Verlag, 1988.

[9] M.R.Garey, D.S.Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, 1979.

[10] J.Hopcroft, J.Ullman: Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie. 2. Aufl., Addison-Wesley, 1990.

[11] E.Horowitz, S.Sahni: Fundamentals of Computer Algorithms. Computer Science Press, 1978.

[12] D.E.Knuth: The Art of Computer Programming. Volumes 1-3, Addison-Wesley, 1968-1975.

[13] U.Manber, Introduction to Algorithms. Addison-Wesley, 1990.

[14] K.Mehlhorn: Effiziente Algorithmen. Teubner-Verlag, 1977.

[15] Ch.Meinel: Effiziente Algorithmen. Fachbuchverlag Leipzig, 1991.

[16] W.Paul: Komplexitätstheorie. Teubner-Verlag, 1978.

[17] Ch.Posthoff, K.Schulz: Grundkurs Theoretische Informatik. Teubner-Verlag, 1992.

[18] U.Schöning: Theoretische Informatik kurz gefaßt. B.I.Wissenschaftsverlag, 1992.

[19] R.Sedgewick: Algorithmen. Addison-Wesley, 1990.

[20] B.A.Trachtenbrot: Algorithmen und Rechenautomaten. Berlin, 1977.

[21] G. Vossen, K.-U. Witt: Grundlagen der Theoretischen Informatik mit Anwendungen. Vieweg-Verlag, Braunschweig, 2000.

[22] K.Wagner: Einführung in die Theoretische Informatik. Springer-Verlag, 1994.

[23] D.Wätjen: Theoretische Informatik. Oldenbourg-Verlag, 1994.

[24] I.Wegener: Theoretische Informatik. Teubner-Verlag, 1993.

[25] D.Wood: Theory of Computation. Harper & Row Publ., 1987.