

**Endlichkeitsproblem für CFG.** Das Endlichkeitsproblem für CFG hat eine kontextfreie Grammatik  $G$  als Eingabe und fragt, ob die erzeugte Sprache endlich ist. Dies lässt sich algorithmisch wie folgt lösen. Zunächst wenden wir die bereits besprochenen Verfahren an, um  $G$  zu bereinigen (siehe Seite 77) und in CNF zu bringen (siehe Beweis von Satz 3.4). Falls  $G$  das leere Wort akzeptiert, so erzeugt die konstruierte CNF-Grammatik  $G'$  die Sprache  $\mathcal{L}(G) \setminus \{\varepsilon\}$ . Diese ist genau dann endlich, wenn  $\mathcal{L}(G)$  endlich ist. Wir nehmen nun an, dass eine bereinigte CNF-Grammatik  $G$  vorliegt. Für diese führen wir einen Zyklentest im Variablengraphen  $\mathcal{V}_G$  durch. (Siehe Seite 76.) Hierfür gibt es Linearzeitalgorithmen, die hier nicht besprochen werden. Ist  $\mathcal{V}_G$  zyklisch, so ist  $\mathcal{L}(G)$  unendlich. Andernfalls ist  $\mathcal{L}(G)$  endlich.

Die Korrektheit des Verfahrens kann wie folgt bewiesen werden. Zunächst nehmen wir an, dass  $\mathcal{V}_G$  zyklisch ist und zeigen, dass aus  $G$  unendlich viele Wörter über dem Terminalalphabet  $\Sigma$  herleitbar sind. Sei etwa  $A_0 A_1 \dots A_m$  ein Zyklus in  $\mathcal{V}_G$  und  $A = A_0 = A_m$ . Da  $G$  bereinigt ist, gibt es Wörter  $u, v, w \in \Sigma^*$  mit  $S \Rightarrow^* uAy$  (da  $A$  erzeugbar ist und keine Variable nutzlos ist) und  $A \Rightarrow^* w$  (da  $A$  nicht nutzlos ist). Da  $A$  auf einem Zyklus liegt, hat  $G$  keine  $\varepsilon$ -Regeln und keine Variable von  $G$  nutzlos ist, gibt es Wörter  $v, x \in \Sigma^*$  mit  $A \Rightarrow^* vAx$  und  $|vx| \geq 1$ . Ab hier können wir wie im Pumping-Lemma argumentieren und erhalten, dass für jedes  $k \geq 0$  das Wort  $uv^kwx^ky$  aus  $G$  herleitbar ist:

$$S \Rightarrow^* uAy \Rightarrow^* uv^kAx^ky \Rightarrow^* uv^kwx^ky.$$

Da wenigstens eines der Wörter  $v$  oder  $x$  nicht-leer ist, sind die Wörter  $uv^kwx^ky$  paarweise verschieden. Wir nehmen nun umgekehrt an, dass  $\mathcal{L}(G)$  unendlich ist und zeigen, dass  $\mathcal{V}_G$  einen Zyklus enthält. Sei  $n$  die Anzahl an Variablen in  $G$ . Da es nur endlich viele Wörter der Länge  $\leq 2^n$  gibt, muss es ein Wort  $w \in \mathcal{L}(G)$  mit  $|w| > 2^n$  geben. Die weitere Argumentation ist nun wie im Beweis des Pumping-Lemmas. Wir betrachten einen Ableitungsbaum  $T$  für  $w$  und einen längsten Pfad in  $T$ . Auf diesem Pfad muss wenigstens eine Variable mehrfach vorkommen, also auf einem Zyklus in  $\mathcal{V}_G$  liegen.

### 3.4.2 Abschlusseigenschaften kontextfreier Sprachen

Wir betrachten hier zunächst nur die drei Operatoren Vereinigung, Konkatenation und Kleeneabschluss, die sich recht einfach mit kontextfreien Grammatiken realisieren lassen.

**Vereinigung.** Seien  $G_1 = (V_1, \Sigma, \mathcal{P}_1, S_1)$  und  $G_2 = (V_2, \Sigma, \mathcal{P}_2, S_2)$  kontextfreie Grammatiken mit demselben Terminalalphabet  $\Sigma$ . Wir können o.E. annehmen, dass  $V_1 \cap V_2 = \emptyset$ . Andernfalls können die Nichtterminale entsprechend umbenannt werden.  $G_1 \uplus G_2$  bezeichnet diejenige Grammatik, die man erhält, wenn man alle Regeln in  $G_1$  und  $G_2$  zulässt und ein neues Startsymbol  $S$  hinzufügt, aus dem die Startsymbole von  $G_1$  und  $G_2$  in einem Schritt herleitbar sind. Formal ist  $G_1 \uplus G_2 \stackrel{\text{def}}{=} (V, \Sigma, \mathcal{P}, S)$  wie folgt definiert. Die Variablenmenge ist  $V = V_1 \cup V_2 \cup \{S\}$  mit einem neuen Startsymbol  $S \notin V_1 \cup V_2$ . Die Regeln von  $G_1 \uplus G_2$  sind die Regeln aus  $G_1$  und  $G_2$  sowie die beiden neuen Startregeln  $S \rightarrow S_1$  und  $S \rightarrow S_2$ . Offenbar ist  $G_1 \uplus G_2$  kontextfrei und  $\mathcal{L}(G_1 \uplus G_2) = \mathcal{L}(G_1) \cup \mathcal{L}(G_2)$ .

Die angegebene Konstruktion einer Grammatik für die Vereinigungssprache ist auch für beliebige Grammatiken (Typ 0 der Chomsky Hierarchie) und kontextsensitive Grammatiken geeignet. Genauer gilt:

**Lemma 3.12 (Vereinigungsoperator für Grammatiken (Typ 0,1,2)).** Sind  $G_1$  und  $G_2$  vom Typ  $i \in \{0, 1, 2\}$ , dann ist auch  $G_1 \uplus G_2$  vom Typ  $i$  und es gilt  $\mathcal{L}(G_1 \uplus G_2) = \mathcal{L}(G_1) \cup \mathcal{L}(G_2)$ .

**Konkatenation.** Wie zuvor seien  $G_1, G_2$  kontextfreie Grammatiken mit demselben Terminalalphabet  $\Sigma$  und disjunkten Variablenmengen. Das Ziel ist die Definition einer kontextfreien Grammatik  $G_1 \circ G_2$ , deren erzeugte Sprache gleich  $\mathcal{L}(G_1) \circ \mathcal{L}(G_2)$  ist. Diese Grammatik  $G_1 \circ G_2$  erhält man, indem man alle Regeln von  $G_1$  und  $G_2$  zulässt und ein neues Startsymbol  $S$  einfügt, aus dem sich das Wort  $S_1 S_2$  ableiten lässt. Formal ist die Grammatik

$$G_1 \circ G_2 \stackrel{\text{def}}{=} (V, \Sigma, \mathcal{P}, S)$$

wie folgt definiert. Die Variablenmenge ist  $V = V_1 \cup V_2 \cup \{S\}$  mit einem neuen Startsymbol  $S \notin V_1 \cup V_2$ . Das Produktionssystem setzt sich aus den Regeln in  $G_1$  und in  $G_2$  und der neuen Startregel  $S \rightarrow S_1 S_2$  zusammen. Offenbar ist mit  $G_1, G_2$  auch die Grammatik  $G_1 \circ G_2$  kontextfrei. Die Korrektheit der angegebenen Definition von  $G_1 \circ G_2$  im Sinne von " $\mathcal{L}(G_1 \circ G_2) = \mathcal{L}(G_1) \circ \mathcal{L}(G_2)$ " ist wie folgt einsichtig.

" $\supseteq$ ": Sei  $w = w_1 w_2$ , wobei  $w_1 \in \mathcal{L}(G_1)$  und  $w_2 \in \mathcal{L}(G_2)$ . Dann gibt es Herleitungen  $S_1 \Rightarrow^* w_1$  in  $G_1$  und  $S_2 \Rightarrow^* w_2$  in  $G_2$ . Diese können zu einer Herleitung von  $w$  in  $G_1 \circ G_2$  kombiniert werden:

$$S \Rightarrow S_1 S_2 \Rightarrow^* w_1 S_2 \Rightarrow^* w_1 w_2 = w$$

Also ist  $w \in \mathcal{L}(G_1 \circ G_2)$ .

" $\subseteq$ ": Durch Induktion nach der Länge  $n$  einer Herleitung  $S \Rightarrow y_1 \Rightarrow y_2 \Rightarrow \dots \Rightarrow y_n$  in  $G_1 \circ G_2$  kann gezeigt werden, dass für  $n \geq 1$  das hergeleitete Wort  $y_n \in (V_1 \cup V_2 \cup \Sigma)^*$  die Gestalt  $y_n = x_1 x_2$  hat, wobei  $w_1$  und  $w_2$  Wörter mit folgenden Eigenschaften sind:

- $w_1 \in (V_1 \cup \Sigma)^*$  und  $S_1 \Rightarrow^* w_1$  in  $G_1$
- $w_2 \in (V_2 \cup \Sigma)^*$  und  $S_2 \Rightarrow^* w_2$  in  $G_2$

Ist nun  $y_n \in \Sigma^*$ , so folgt  $w_1 \in \mathcal{L}(G_1)$  und  $w_2 \in \mathcal{L}(G_2)$  und somit  $y_n \in \mathcal{L}(G_1) \circ \mathcal{L}(G_2)$ . ■

Die angegebene Konstruktion  $G_1 \circ G_2$  liefert zwar auch für Grammatiken vom Typ 0 oder 1 eine Grammatik des entsprechenden Typs, jedoch ist nicht garantiert, dass die akzeptierte Sprache mit  $\mathcal{L}(G_1) \circ \mathcal{L}(G_2)$  übereinstimmt. Lediglich die Inklusion  $\mathcal{L}(G_1) \circ \mathcal{L}(G_2) \subseteq \mathcal{L}(G_1 \circ G_2)$  kann für beliebige Grammatiken  $G_1$  und  $G_2$  garantiert werden, nicht aber die umgekehrte Inklusion. Besteht z.B. Grammatik  $G_1$  aus der Regel  $S_1 \rightarrow a$  und  $G_2$  aus der Regel  $aS_2 \rightarrow aa$ , dann ist die erzeugte Sprache von  $G_2$  leer und somit  $\mathcal{L}(G_1) \circ \mathcal{L}(G_2) = \emptyset$ . Andererseits ist

$$S \Rightarrow S_1 S_2 \Rightarrow a S_2 \Rightarrow aa$$

eine Herleitung in  $G_1 \circ G_2$ . Also ist  $aa \in \mathcal{L}(G_1 \circ G_2) \setminus (\mathcal{L}(G_1) \circ \mathcal{L}(G_2))$ .

**Kleeneabschluss.** Sei  $G = (V, \Sigma, \mathcal{P}, S)$  eine kontextfreie Grammatik. Eine kontextfreie Grammatik  $G^*$  für die Sprache  $\mathcal{L}(G)^*$  erhält man, indem man  $G$  um ein neues Startsymbol  $S'$  erweitert, aus dem sich  $\varepsilon$  und  $SS'$  herleiten lassen. Formal ist

$$G^* \stackrel{\text{def}}{=} (V', \Sigma, \mathcal{P}_*, S_*),$$

wobei  $V' = V \cup \{S_*\}$  mit  $S_* \notin V$  und das Produktionssystem  $\mathcal{P}_*$  aus den Regeln in  $\mathcal{P}$  und den beiden neuen Startregeln  $S_* \rightarrow \varepsilon$  und  $S_* \rightarrow SS_*$  besteht. Es ist klar, dass mit  $G$  auch  $G^*$  kontextfrei ist. Ähnlich wie für den Konkatenationsoperator kann gezeigt werden, dass  $\mathcal{L}(G^*) = \mathcal{L}(G)^*$ , sofern  $G$  kontextfrei ist. Für Grammatiken vom Typ 0 oder Typ 1 ist der angegebene Operator für den Kleeneabschluss jedoch falsch.

**Corollar 3.13.** *Die Klasse der kontextfreien Sprachen ist unter Vereinigung, Konkatenation und Kleeneabschluss abgeschlossen.*

Die Klasse der kontextfreien Sprachen ist jedoch *nicht* abgeschlossen unter der Durchschnitts- und Komplementbildung. Dies lässt sich wie folgt begründen. Die Sprachen

$$L_1 = \{a^n b^n c^m : n, m \geq 1\}, \quad L_2 = \{a^n b^m c^m : n, m \geq 1\}$$

sind kontextfrei. Z.B. wird  $L_1$  durch die CFG mit den Regeln

$$S \rightarrow DC \quad C \rightarrow c \mid Cc \quad D \rightarrow aDb \mid ab$$

erzeugt. Eine ähnliche Grammatik kann für  $L_2$  angegeben werden. Die Durchschnittssprache

$$L_1 \cap L_2 = \{a^n b^n c^n : n \geq 1\}$$

ist jedoch *nicht* kontextfrei (wie zuvor gezeigt). Wäre die Klasse der kontextfreien Sprachen abgeschlossen unter der Komplementbildung, dann wäre sie auch unter der Durchschnittsbildung abgeschlossen. Dies folgt aus dem de Morganschen Gesetz

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

und der Abgeschlossenheit kontextfreier Sprachen unter der Vereinigung.

**Satz 3.14.** *Die Klasse der kontextfreien Sprachen ist nicht abgeschlossen unter Durchschnitt und Komplement.*

### 3.5 Kellerautomaten

Das Automatenmodell für reguläre Sprachen sind endliche Automaten. Diese haben keinen unbegrenzten Speicher und können höchstens Daten eines festen endlichen Bereichs in ihren Zuständen speichern. Für Sprachen wie  $L = \{a^n b^n : n \geq 1\}$  ist dies jedoch nicht ausreichend, da ein Automat für  $L$  die Information über die genaue Anzahl an gelesenen  $a$ 's speichern muss. Kellerautomaten erweitern endliche Automaten um ein unbegrenztes Speichermedium, nämlich einen Keller. Wir machen uns zunächst klar, warum für kontextfreie Sprachen genau Keller als Speichermedium geeignet sind und betrachten dazu eine weitere Normalform kontextfreier Grammatiken.

### 3.5.1 Die Greibach Normalform

Neben der Chomsky Normalform gibt es eine Reihe weiterer Normalformen für kontextfreie Grammatiken. Wir erwähnen hier kurz die Greibach Normalform, die wir dann als Ausgangspunkt für die Diskussion von Kellerautomaten und deren Äquivalenz zu CFG nutzen werden.

**Definition 3.15 (Greibach Normalform).** Sei  $G$  eine CFG.  $G$  ist in Greibach Normalform, falls alle Produktionen in  $G$  von der Form

$$A \rightarrow aB_1B_2\dots B_k$$

sind, wobei  $a \in \Sigma$ ,  $B_1, B_2, \dots, B_k \in V$  und  $k \in \mathbb{N}$ . Der Fall  $k = 0$ , also Regeln der Form  $A \rightarrow a$ , ist zugelassen. ■

Z.B. ist durch  $S \rightarrow aB \mid aSB$ ,  $B \rightarrow b$  eine CFG in Greibach Normalform für die Sprache  $\{a^n b^n : n \geq 1\}$  gegeben. Weiter ist jede reguläre Grammatik, die keine  $\varepsilon$ -Regeln enthält, in Greibach Normalform. Wie die Chomsky Normalform ist auch die Greibach Normalform universell für kontextfreie Sprachen, die das leere Wort nicht enthalten. Wir zitieren dieses Ergebnis ohne Beweis:

**Satz 3.16.** *Zu jeder CFG mit  $\varepsilon \notin \mathcal{L}(G)$  gibt es eine äquivalente CFG in Greibach Normalform. (ohne Beweis)*

Linksableitungen für Grammatiken in Greibach Normalform haben in  $i$  Schritten ein Wort der Form  $x_i = a_1 \dots a_i A_1 A_2 \dots A_k$  erzeugt. Dabei sind  $a_1, \dots, a_i$  Terminalzeichen. Für den jeweils nächsten Ableitungsschritt ist nur die Kenntnis der ersten Variablen  $A_1$  entscheidend. Diese wird durch ein Wort  $a_{i+1} B_1 B_2 \dots B_\ell$  ersetzt. Das hergeleitete Wort hat nun die Form

$$x_{i+1} = a_1 \dots a_i a_{i+1} B_1 B_2 \dots B_\ell A_2 \dots A_k.$$

Der nächste Linksableitungsschritt muss nun eine Regel für Variable  $B_1$  einsetzen, etwa  $B_1 \rightarrow a_{i+2} C_1 C_2 \dots C_m$ . Man erhält das Wort

$$x_{i+2} = a_1 \dots a_i a_{i+1} a_{i+2} C_1 C_2 \dots C_m B_2 \dots B_\ell A_2 \dots A_k,$$

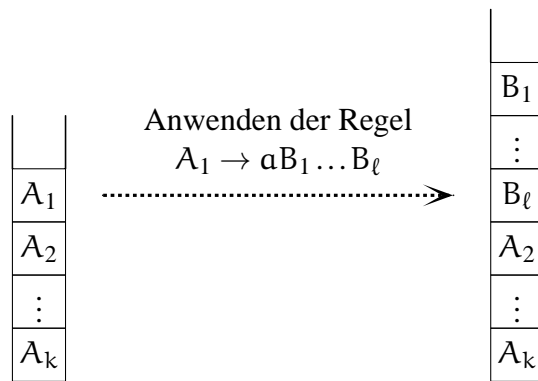
für das nun  $C_1$  durch Anwenden einer Regel zu ersetzen ist. Das Suffix bestehend aus den noch zu ersetzenden Variablen kann durch Anwenden von Terminalregeln, also Regeln des Typs  $A \rightarrow a$ , verkürzt werden. Wird etwa für  $x_{i+2}$  die Terminalregel  $C_1 \rightarrow a_{i+3}$  eingesetzt, so hat das nun hergeleitete Wort die Form

$$x_{i+3} = a_1 \dots a_i a_{i+1} a_{i+2} a_{i+3} C_2 \dots C_m B_2 \dots B_\ell A_2 \dots A_k,$$

so dass als nächstes eine Regel für Variable  $C_2$  eingesetzt werden muss. Linksableitungen behandeln also die Variablen im LIFO-Prinzip.<sup>9</sup>

---

<sup>9</sup>Die Abkürzung LIFO steht für "last-in, first-out".



Zur Verwaltung der jeweils hergeleiteten Teilwörter, bestehend aus den noch zu ersetzenden Variablen, scheint daher ein Keller geeignet zu sein. Für die Auswahl, welche der Regeln  $A_i \rightarrow a_{i+1}B_1 \dots B_\ell$  angewandt werden, setzen wir das Konzept von Nichtdeterminismus ein. Diese Überlegungen führen zu den nichtdeterministischen Kellerautomaten.