Inhaltsverzeichnis

Kapitel 7

Ergänzungen IV: Erweiterungen von kontextfreien Sprachen

Bei der Untersuchung von Programmiersprachen und natürlichen Sprachen wird häufig ausgenutzt, dass sie sich zu einem großen Teil durch kontextfreie Sprachen beschreiben lassen. Jedoch ist festzustellen, dass nicht alle Aspekte natürlicher Sprachen bzw. von Programmiersprachen durch kontextfreie Grammatiken beschreiben lassen. Wir geben dafür jetzt drei Beispiele.

Wir beginnen mit der englischen Sprache. Dazu betrachten wir die Sätze.

Mary		and John		are		
a woman		and a man,		respectively.		
Mary,	John	and William		are		
a woman,	a man	and a man,		respectively.		
Mary,	John,	William	and Jenny		are	
a woman,	a man,	a man	and a woman,		respectively.	

Sie sind offensichtlich von der gleichen Bauart: Zuerst wird eine Folge von Frauen- und Männernamen gegeben und dann wird hinzugesetzt, ob es sich bei der Person mit dem Namen, um eine Frau oder einen Mann handelt. Derartige Sätze beliebiger Länge sind im Englischen zugelassen (obwohl man sie praktisch natürlich höchstens mit fünf oder sechs Namen gebrauchen wird). Wir betrachten nun die Menge

$$R = \{x, \mid x \in X\}\{x, \mid x \in X\}^+\{and\}X\{are\}YY^+\{and\}Y\{respectively.\} \ ,$$

wobei X die Menge der Vornamen der englischen Sprache und $Y = \{a\ woman,\ ,\ a\ man,\}$ sind. Die Menge R beschreibt die Sätze obiger Form, ohne allerdings darauf zu achten, ob Name und Geschlecht zueinander passen und ob die Folgen der Namen und die Folge aus *a woman* und *a man* die gleiche Länge haben. Diese beiden zusätzlichen Eigenschaften sind aber bei der englischen Sprache gefordert. Folglich beschreibt die Menge $Englisch \cap R$ genau die obigen Sätze. Wir betrachten nun einen Homomorphismus h, der durch

$h(and) = h(are) = h(a) = h(respectively) = h(,) = h(.) = \lambda$
$h(woman) = a$, $h(x) = a$ für jeden Frauennamen x,
$h(man) = b$, $h(y) = b$ für jeden Männernamen y

gegeben ist. Dann erhalten wir

$$R' = h(Englisch \cap R) = \{ww \mid w \in \{a,b\}^+, |w| \geq 3\}.$$

Wir wissen, dass R' keine kontextfreie Sprache ist. Nehmen wir nun an, dass *Englisch* eine kontextfreie Sprache ist. Nach den Abschlusseigenschaften von $\mathcal{L}(CF)$ ist dann auch R' kontextfrei, was falsch ist. Somit ist unsere Annahme falsch, also ist *Englisch* keine kontextfreie Sprache.

Dieses Beispiel ist nicht ganz befriedigend, weil die Beziehung zwischen den Namen und dem Geschlecht eine semantische Relation ist, die Theorie formaler Sprachen aber nur die Syntax beschreiben kann. Wir betrachten daher die Sätze

> *Jan säit das mer em Hans hälfed.*
> (Jan sagt, dass wir Hans helfen.)
>
> *Jan säit das mer em Hans es Huus hälfed aastriche.*
> (Jan sagt, dass wir Hans helfen, das Haus zu streichen.)
>
> *Jan säit das mer d'chind em Hans es Huus lönd hälfed aastriche.*
> (Jan sagt, dass wir den Kindern erlauben, Hans zu helfen, das Haus zu streichen.)

aus einem Dialekt des Schweizerdeutschen. Auch hier liegt die gleiche Struktur vor, weil die Tätigkeiten und die zugehörigen Objekte in der gleichen Reihenfolge stehen (man beachte, dass dies bei der Übersetzung ins Deutsche nicht mehr der Fall ist). Daher ergibt sich hier wie oben

$$h'(Schweizerdeutsch \cap Q) = \{ww \mid w \in \{a,b\}^+\}$$

für einen passenden Homomorphismus h' und eine passende reguläre Sprache Q. Folglich ist Schweizerdeutsch keine kontextfreie Sprache (was sich ebenfalls analog ergibt). Hierbei haben wir die syntaktische Beziehungen Verb – Objekt ausgenutzt.

Wir kommen nun Programmiersprache ALGOL60, einer der klassische deklarativen Programmiersprachen (ähnliche Konstruktionen lassen sich aber auch für andere Programmiersprachen durchführen). Bei ALGOL60 ist es erforderlich, jede auftretende Variable zu deklarieren. Wir betrachten das Programm

```
begin integer x;
      y := 1
end
```

wobei die beiden auftretenden Variablen x und y jeweils durch ein Wort über $\{a,b\}$ gegeben seien. Damit das Programm richtig ist, muss an beiden Stellen des Auftretens einer Variablen jeweils das gleiche Wort stehen. Daher ergibt sich erneut

$$h''(ALGOL \cap S) = \{xx \mid x \in \{a,b\}^*\}$$

für eine entsprechend gewählte reguläre Menge S und einen passenden Homomorphismus h''. Damit ist auch ALGOL60 keine kontextfreie Sprache.

Zur Beschreibung von natürlichen Sprachen und Programmiersprachen bedarf es also Mechanismen, die auch gewisse nicht-kontextfreie Sprachen erzeugen können. Dies ist sicher für die kontextsensitiven Grammatiken der Fall, die aber zwei negative Aspekte haben: Einige Entscheidbarkeitsprobleme sind für kontextsensitive Grammatiken unentscheidbar bzw. sehr schwer; so sind z.B. nach Satz 5.20 das Endlichkeits- und das Leerheitsproblem unentscheidbar. Zum anderen gibt es für kontextsensitive Grammatiken keine vernünftigen Ableitungsbäume, da mehrere Nichtterminale aus verschiedenen Schichten des Baumes von einer regel betroffen sein können. Ableitungsbäume haben sich aber als sehr gutes Instrument für die Analyse von Sätzen bzw. Programmen erwiesen.
Daher ist man an Erweiterungen der kontextfreien Grammatiken interessiert, die zum einen Ableitungsbäume ermöglichen (d.h. jede angewendete Regel muss kontextfrei sein) und zum anderen möglichst einfach zu entscheidende Probleme haben. Wir präsentieren in diesem Abschnitt zwei derartigen Grammatiken, wobei dr Fokus stärker auf dem erstgenannten Aspekt der Existenz von Ableitungsbäumen liegt.

Definition 7.1 *i) Eine Grammatik mit Auswahlkontext ist ein Quadrupel $G = (N, T, P, S)$, wobei*

- *N, T, S wie bei einer Regelgrammatik spezifiziert sind,*
- *P eine endliche Menge von Tripeln $p = (r_p, E_p, F_p)$ ist, wobei jeweils $r_p = A_p \to w_p$ eine kontextfreie Regel mit $w_p \neq \lambda$ ist, und E_p und F_p Teilmengen von N mit $E_p \cap F_p = \emptyset$ sind.*

ii) Für zwei nichtleere Wörter x und y über $N \cup T$ sagen wir, dass y aus x durch Anwendung von $(A \to w, E, F)$ erzeugt wird (geschrieben als $X \Longrightarrow_p y$), wenn folgende Bedingungen erfüllt sind:
— $x = uAv$, $y = uwv$ (kontextfreie Ersetzung)
— jedes Symbol aus E kommt in uv vor,
— kein Symbol aus F kommt in uv vor.
Wir sagen, dass y aus x in G durch einen Ableitungsschritt entsteht (geschrieben als $x \Longrightarrow_G y$), wenn es eine Regel p mit $x \Longrightarrow_p y$ gibt.
iii) Die von einer Grammatik $G = (N, T, P, S)$ mit Auswahlkontext erzeugte Sprache $L(G)$ ist

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

wobei $\Longrightarrow_G^*$ der reflexive und transitive Abschluss von $\Longrightarrow_G$ ist.

Bei einer Regel $p = (A \to w, E, F)$ heißen E und F der geforderte bzw. verbotene Kontext der Regel $A \to w$
Die Definition der Grammatiken mit Auswahlkontext ist speziell darauf abgestellt, einen Zusammenhang zwischen Deklaration und Benutzung von Variablen in Programmiersprachen zu ermöglichen, d.h. den Grund dafür, dass oben im dritten Beispiel eine nicht kontextfreie Sprache auftritt, zu eliminieren. Dies geschieht dadurch, das eine Benutzung

einer Variablen entsprechend einer Regel nur möglich ist, wenn diese schon in der Satzform vorkommt, was dadurch erreicht wird, dass die Variable in der zur Regel gehörenden Menge E liegt.
Wir bemerken weiterhin, dass Grammatiken mit Auswahlkontext eine Erweiterung der kontextfreien Grammatiken sind, denn bei der Wahl von $F = \emptyset$ für jede Regel, ergibt sich eine kontextfreie Grammatik, da nun keine Bedingungen mehr für die Anwendung von Regeln vorliegen (also jede Regel jederzeit anwendbar ist); und umgekehrt kann jede kontextfreie Grammatik als eine Grammatik mit Auswahlkontext aufgefasst werden, bei der alle geforderten und verbotenen Kontexte leer sind.
Wir zeigen nun, dass Grammatiken mit Auswahlkontext Sprachen erzeugen können, die nicht kontextfrei sind.

Beispiel 7.2 Wir betrachten die Grammatik mit Auswahlkontext

$$G_1 = (\{S, A, A', A_a, A_b, B, B'\}, \{a, b, c\}, \{p_0, p_1, \ldots, p_{10}\}, S)$$

mit den Regeln

$$\begin{array}{ll}
p_0 = (S \to AB, \emptyset, \emptyset), & p_1 = (A \to aA_a, \{B\}, \emptyset), \\
p_2 = (A \to bA_b, \{B\}, \emptyset), & p_3 = (B \to aB', \{A_a\}, \emptyset), \\
p_4 = (B \to bB', \{A_b\}, \emptyset), & p_5 = (A_a \to A, \{B'\}, \emptyset), \\
p_6 = (A_b \to A, \{B'\}, \emptyset), & p_7 = (B' \to B, \{A\}, \emptyset) \\
p_8 = (A \to A', \{B\}, \emptyset), & p_9 = (B \to c, \{A'\}, \emptyset), \\
p_{10} = (A' \to c, \emptyset, \emptyset). &
\end{array}$$

Offensichtlich beginnt jede Ableitung mit der einzigen Regel für S, d.h. wir erhalten $S \Longrightarrow AB$. Wir wollen etwas allgemeiner von einer Satzform $wAwB$ ausgehen (das in einem Schritt erhaltene Wort AB wird gerade bei $w = \lambda$ erhalten). Wir können keine der Regeln mit linker Seite B anwenden, da der jeweilige geforderte Kontext A_a bzw. A_b bzw.A' in der Satzform nicht vorhanden ist. Folglich muss eine Regel mit linker Seite A verwendet werden. Wir unterscheiden drei Fälle.

Fall 1. Es wird die Regel p_1 angewendet. Dadurch erhalten wir waA_awB. Da die einzige Regel für A_a den geforderten Kontext B', ist eine Regel für B anzuwenden. Es kommt nur Regel p_3 in Frage, da bei den anderen Regeln wieder der geforderte Kontext fehlt. Daher ergibt sich waA_awaB'. Jetzt ist nur Regel p_5 anwendbar, wodurch $waAwaB'$ entsteht. Nun ist Regel p_7 anzuwenden, und wir erhalten $waAwaB$. Damit haben wir wieder die Ausgangssituation erhalten, nur dass das Wort an beiden Stellen um den Buchstaben a verlängert wurde.

Fall 2. Es wird die Regel p_2 angewendet. Dann ergibt sich analog zu Fall 1 als einzig mögliche Ableitung

$$wAwB \Longrightarrow_{p_2} wbA_bwB \Longrightarrow_{p_4} wbA_bwbB' \Longrightarrow_{p_6} wbAwbB' \Longrightarrow_{p_7} wbAwbB,$$

d.h. wir haben das Wort an beiden Stellen um den Buchstaben b verlängert.

Fall 3. Es wird die Regel p_8 angewendet. Dann ergibt sich die eindeutige Ableitung

$$wAwB \Longrightarrow_{p_8} wA'wB \Longrightarrow_{p_9} wA'wc \Longrightarrow_{p_{10}} wcwc$$

(man beachte, dass die Anwendung von p_{10} auf $wA'wB$ zwar möglich ist, aber $wcwB$ liefert, worauf keine Regel mehr anwendbar ist und somit kein terminales Wort erreicht werden kann).
Aufgrund der drei Fälle ist sofort zu sehen, dass

$$L(G_1) = \{wcwc \mid w \in \{a,b\}^*\}$$

gilt.
Wir machen darauf aufmerksam, dass in allen Regeln von G_1 kein verbotener Kontext vorkommt.

Beispiel 7.3 Es sei die Grammatik mit Auswahlkontext

$$G_2 = (\{S, A, A', B, C, D\}, \{a\}, \{p_0, p_1, \ldots, p_8\}, S)$$

mit den Regeln

$$\begin{array}{lll} p_0 = (S \to CA, \emptyset, \emptyset), & p_1 = (S \to BA, \emptyset, \emptyset), & p_2 = (A \to a, \{C\}, \emptyset), \\ p_3 = (C \to a, \emptyset, \{A, A'\}), & p_4 = (A \to A'A', \{B\}, \emptyset), & p_5 = (B \to D, \emptyset\{A\}), \\ p_6 = (A' \to A, \{D\}, \emptyset), & p_7 = (D \to B, \emptyset, \{A'\}), & p_8 = (D \to C, \emptyset, \{A'\}) \end{array}$$

gegeben. Zu Beginn ist eine der Regeln p_0 oder p_1 anzuwenden. Hierdurch entstehen CA und BA. Wir betrachten erneut allgemeiner die Situation, dass die Satzform CA^{2^n} oder BA^{2^n} für eine nicht-negative ganze Zahl n vorliegt (CA und BA entsprechen $n = 0$). Sei zuerst CA^{2^n} gegeben. Die einzige Regel für C ist nicht anwendbar, da A zum verbotenen Kontext gehört. Die Regel p_4 ist ebenfalls nicht anwendbar, da bei ihr der Kontext B gefordert wird. Folglich ist nur p_2 anwendbar, wodurch CA^raA^s mit $r + s = 2^n - 1$ entsteht. Solange noch ein A und kein B vorhanden sind, sind p_3 und p_4 nicht anwendbar. Damit erhalten wir die Ableitung

$$CA^{2^n} \Longrightarrow_{p_2} CA^raA^s \Longrightarrow_{p_2} CA^naA^maA^k \Longrightarrow_{p_2} \ldots \Longrightarrow_{p_2} Ca^{2^n}.$$

Nun ist nur p_3 anwendbar und wir terminieren mit $aa^{2^n} = a^{2^n+1}$.
Starten wir mit BA^{2^n}, so erhalten wir durch analoge Überlegungen die eindeutige Ableitung

$$\begin{array}{lll} BA^{2^n} & \Longrightarrow_{p_4} & BA^raA'A'A^s \Longrightarrow_{p_4} BA^nA'A'A^mA'A'A^k \Longrightarrow_{p_4} \ldots \Longrightarrow_{p_4} B(A'A')^{2^n} = B(A')^{2^{n+1}} \\ & \Longrightarrow_{p_5} & D(A')^{2^{n+1}} \Longrightarrow_{p_6} D^(A')^pA(A')^q \Longrightarrow_{p_6} \ldots \Longrightarrow_{p_6} DA^{2^{n+1}} \end{array}$$

Jetzt gibt es zwei Fortsetzung mittels der regeln p_7 oder p_8, wodurch $CA^{2^{n+1}}$ und $BA^{2^{n+1}}$ entstehen. Damit erhalten wir wieder unsere Ausgangssituation nur mit einer um Eins erhöhten Zweierpotenz. Hieraus folgt nun sofort

$$L(G_2) = \{a^{2^n+1} \mid n \geq 0\}.$$

Wir merken ohne Beweis an, dass die Erzeugung von $L(G_2)$ verbotene Kontexte erfordert.

Wir definieren nun die zweite Art von Grammatiken, die zu einer Erweiterung der kontextfreien Sprachen führt.

Definition 7.4 *i) Eine programmierte Grammatik ist eine Quintupel* $G = (N, T, Lab, P, S)$, *wobei*

- N, T, S *wie bei einer Regelgrammatik spezifiziert sind,*
- Lab *eine endliche Menge von Labels ist,*
- P *eine endliche Menge von Regeln* $p = (l_p, A_p \to w_p, \sigma_p, \varphi_p)$ *ist, wobei jeweils* $l_p \in Lab$, $A_p \in N$, $w_p \in (N \cup T)^+$, $\sigma_p \subseteq Lab$ *und* $\varphi_p \subseteq Lab$ *gelten, und*
- *für zwei verschiedene Regeln* p *und* q *die zugehörigen Labels* l_p *und* l_q *auch verschieden sind.*

ii) Die von einer programmierten Grammatik $G = (N, T, Lab, P, S)$ *erzeugte Sprache besteht aus allen Wörtern* $w \in T^*$, *für die es eine Ableitung*

$$S = w_0 \Longrightarrow_{p_1} w_1 \Longrightarrow_{p_2} w_2 \Longrightarrow_{p_3} \ldots \Longrightarrow_{p_k} w_k = w,$$

mit $k \geq 1$ *so gibt, dass für* $1 \leq i \leq k$ *mit* $p_i = (l_i, A_i \to v_i, \sigma_i, \varphi_i)$, *entweder*

$$w_{i-1} = w'_{i-1} A_i w''_{i-1},\ w_i = w'_{i-1} v_i w''_{i-1} \text{ für gewisse } w'_{i-1}, w''_{i-1} \in V_G^*,\ l_{i+1} \in \sigma_i$$

oder

$$A_i \text{ kommt in } w_{i-1} \text{ nicht vor, } w_{i-1} = w_i,\ l_{i+1} \in \varphi_i.$$

gilt.

Die Mengen σ_p und φ_p einer Regel p heißen Erfolgs- bzw. Fehlerfeld. Ist die Regel p anwendbar (hatte ihre Anwendung Erfolg), so wird mit einer Regel fortgesetzt, deren Label im Erfolgsfeld σ_p liegt; ist die Regel dagegen nicht anwendbar, so wird ohne Änderung der Satzform mit einer Regel fortgesetzt, deren Label im Fehlerfeld φ_p liegt. Mit einer Regel ist also immer eine Angabe der möglichen nächsten Regeln verbunden.
Erneut ist einfach zu sehen, dass programmierte Grammatiken eine Erweiterung der kontextfreien Grammatiken darstellen. Die kontextfreien Grammatiken ergeben sich gerade durch die Festsetzung $\sigma_p = \varphi_p = Lab$, da dann keine Steuerung der Reihenfolge der Regeln mehr gegeben ist, sondern jede Regel nach jeder Regel angewendet werden kann, wie es bei den kontextfreien Grammatiken der Fall ist.

Beispiel 7.5 Wir betrachten die programmierte Grammatik

$$G'_1 = (\{S, A, B\}, \{a, b\}, \{q_0, q_1, \ldots, q_8\}, \{r_0, r_1, r_2, \ldots, r_8\}, S)$$

mit

$$\begin{array}{ll}
r_0 = (q_0, S \to AB, \{q_1, q_3, q_5, q_7\}, \emptyset), & \\
r_1 = (q_1, A \to aA, \{q_2\}, \emptyset), & r_2 = (q_2, B \to aB, \{q_1, q_3, q_5, q_7\}, \emptyset), \\
r_3 = (q_3, A \to bA, \{q_4\}, \emptyset), & r_4 = (q_4, B \to bB, \{q_1, q_3, q_5, q_7\}, \emptyset), \\
r_5 = (q_5, A \to a, \{q_6\}, \emptyset), & r_6 = (q_6, B \to a, \emptyset, \emptyset), \\
r_7 = (q_7, A \to b, \{q_8\}, \emptyset), & r_8 = (q_8, B \to b, \emptyset, \emptyset).
\end{array}$$

Jede Ableitung in G beginnt mit einer Anwendung von r_0, wodurch die Satzform AB entsteht. Da r_0 anwendbar war, sind im nächsten Schritt die Regeln r_1, r_3, r_5 und r_7 anwendbar.
Wir diskutieren nun die möglichen Ableitungen einer Satzform $wAwB$, wobei wir davon ausgehen, dass darauf die Regeln r_1, r_3, r_5 und r_7 im nächsten Schritt anwendbar sind. Dann ergeben sich die folgenden Ableitungen

$$\begin{aligned}
wAwB &\Longrightarrow_{r_1} waAwB \Longrightarrow_{r_2} waAwaB,\\
wAwB &\Longrightarrow_{r_3} wbAwB \Longrightarrow_{r_4} wbAwbB,\\
wAwB &\Longrightarrow_{r_5} wawB \Longrightarrow_{r_6} wawa,\\
wAwB &\Longrightarrow_{r_7} wbwB \Longrightarrow_{r_8} wbwb.
\end{aligned}$$

In allen Fällen wird das vorhandene Wort w an beiden Stellen seines Auftretens um jeweils den gleichen Buchstaben verlängert. In den ersten beiden Fällen kann die Ableitung erneut mit den Regeln r_1, r_3, r_5 und r_7 fortgesetzt werden; in den beiden letzten Fällen ist ein terminales Wort erreicht und die Ableitung beendet. Damit ergibt sich

$$L(G_1') = \{ww \mid w \in \{a,b\}^+\}.$$

Beispiel 7.6 Es sei die programmierte Grammatik

$$G_2' = (\{S,A\}, \{a\}, \{q_1,q_2,q_3\}, \{r_1,r_2,r_3\}, S)$$

mit den Regeln

$$r_1 = (q_1, S \to AA, \{q_1\}, \{q_2\}),\ r_2 = (q_2, A \to S, \{q_2\}, \{q_1,q_3\}), r_3 = (q_3, S \to a, \{q_3\}, \emptyset)$$

gegeben. Auf das Startsymbol S sind die Regeln r_1 und r_3 anwendbar. Wir betrachten die Situation, dass auf eine Satzform S^n die Regeln r_1 und r_3 anwendbar sind. Wenn wir r_3 anwenden, so haben wir das immer wieder zu tun, da das Erfolgsfeld von r_3 nur den Label q_3 enthält, d.h. wir haben der Reihe nach alle vorkommenden S jeweils durch a ersetzt. Dies liefert

$$S^n \Longrightarrow_{r_3} S^p a S^q \Longrightarrow_{r_3} \ldots \Longrightarrow_{r_3} a^n.$$

Damit hat die Ableitung terminiert. Wenden wir dagegen r_1 an, so ist dies weiterhin zu tun, bis alle S durch AA ersetzt sind. Die erneute Anwendung von r_3 schlägt jetzt fehl, so dass mit r_2 fortzusetzen ist. Diese Regel ist wiederum solange anzuwenden bis alle A wieder durch S ersetzt sind. Formal ergibt sich also

$$S^n \Longrightarrow_{r_1} S^p AAS^q \Longrightarrow_{r_1} \ldots \Longrightarrow_{r_1} (AA)^n = A^{2n} \Longrightarrow_{r_2} A^s S A^t \Longrightarrow_{r_2} \ldots \Longrightarrow_{r_2} S^{2n}.$$

Eine Fortsetzung muss nun durch r_1 oder r_3 erfolgen, d.h. wir haben die gleiche Situation erreicht, aber die Anzahl der Ss verdoppelt. Daher ergibt sich

$$L(G_2') = \{a^{2^n} \mid n \geq 0\}.$$

Mit $\mathcal{L}(P)$ und $\mathcal{L}(RC)$ bezeichnen wir die Mengen der von programmierten Grammatiken bzw. von Grammatiken mit Auswahlkontext erzeugten Sprachen. Wir vergleichen zuerst die beiden Sprachmengen.

Lemma 7.7 $\mathcal{L}(RC) \subseteq \mathcal{L}(P)$.

Beweis. Es sei $L \in \mathcal{L}(RC)$. Dann gibt es eine Grammatik $G = (N, T, P, S)$ mit Auswahlkontext so, dass $L = L(G)$ gilt. Wir setzen zuerst

$$N' = N \cup \{A' \mid A \in N\}.$$

Es sei $p = (A \to w, \{A_1, A_2, \ldots, A_r\}, \{B_1, B_2, \ldots, B_s\})$ eine Regel aus P. Diese ist auf eine Satzform xAy nur anwendbar, wenn xy alle Symbole $A_1, A_2 \ldots, A_r$ und keinen der Buchstaben $B_1, B_2, \ldots, B_s$ enthält; die Anwendung liefert dann xwy. Für p führen wir die Labels (p, i) mit $1 \leq i \leq r + s + 2$ und die zugehörigen Regeln

$$\begin{aligned}
r_{p,1} &= ((p,1), A \to A', \{(p,2)\}, \emptyset),\\
r_{p,2} &= ((p,2), A_1 \to A_1, \{(p,3)\}, \emptyset),\\
r_{p,3} &= ((p,2), A_2 \to A_2, \{(p,4)\}, \emptyset),\\
\ldots & \qquad\qquad \ldots\\
r_{p,r} &= ((p,r-1), A_{r-1} \to A_{r-1}, \{(p,r+1)\}, \emptyset),\\
r_{p,r+1} &= ((p,r+1), A_r \to A_r, \{(p,r+2)\}, \emptyset),\\
r_{p,r+2} &= ((p,r+2), B_1 \to B_1, \emptyset, \{(p,r+3)\}),\\
r_{p,r+3} &= ((p,r+3), B_2 \to B_2, \emptyset, \{(p,r+4)\}),\\
\ldots & \qquad\qquad \ldots\\
r_{p,r+s} &= ((p,r+s), B_{s-1} \to B_{s-1}, \emptyset, \{(p,r+s+1)\}),\\
r_{p,r+s+1} &= ((p,r+s+1), B_s \to B_s, \emptyset, \{(p,r+s+2)\}),\\
r_{p,r+s+2} &= ((p,r+s+2), A' \to w, \{(p,1) \mid p \in P\}, \emptyset)
\end{aligned}$$

ein. Durch die Regel $r_{p,i+1}$ testen wir, ob A_i in der Satzform vorhanden ist, denn $A_i \to A_i$ ist nur dann anwendbar, ändert aber die Satzform nicht. Ist A_i nicht vorhanden, kann die Ableitung nicht fortgesetzt werden, da das Fehlerfeld von $r_{p,i+1}$ leer ist. Analog testen wir durch die Regeln $r_{p,r+s+j+1}$ ob B_j in der Satzform vorkommt, jedoch ist jetzt nur eine Fortsetzung möglich, wenn B_j nicht vorhanden ist, da das Erfolgsfeld leer ist. Weiterhin sind die Regeln genau in der gegebenen Reihenfolge anzuwenden, da die Erfolgs- bzw. Fehlerfelder genau eine Nachfolgeregel spezifizieren. Da die Regeln $r_{p,2}$ – $r_{p,r+s+1}$ keine Veränderung der Satzform bewirken, wird nur entsprechend $r_{p,1}$ und $r_{p,r+s+2}$ eine Veränderung erreicht, die in $xAy \Longrightarrow xA'y \Longrightarrow xwy$ resultieren. Damit wird die Satzform erreicht, die auch bei Anwendung von p entsteht und die Anwendbarkeitsbedingungen für p wurden auch getestet. Das Nichtterminal A' wurde eingeführt, damit es beim Testen der Vorkommen von $A_1, A_2, \ldots, A_r, B_1, B_2, \ldots, B_s$ nicht berücksichtigt wird.
Wir konstruieren die programmierte Grammatik $G' = (N', T, Lab, P', S)$, wobei Lab und P' aus allen Labels und Regeln bestehen, die sich entsprechend obigem Verfahren ergeben. Es ist nun leicht einzusehen, dass die Ableitung $xAy \Longrightarrow_p xwy$ in G genau dann existiert, wenn wir in G' die Ableitung $xAy \Longrightarrow_{r_{p,1}} xA'y \Longrightarrow_{r_{p,2}} \ldots \Longrightarrow_{r_{p,r+s+2}} xwy$ gibt. Daher erzeugen beide Grammatiken die gleiche Sprache. Somit haben wir $L = L(G) = L(G') \in \mathcal{L}(P)$. □

Für den Beweis der Umkehrung benötigen wir das folgende Lemma.

Lemma 7.8 *Es seien $L \in \mathcal{L}(P)$ eine Sprache mit $L \subseteq T^*$ und $a \in T$ gegeben. Wenn die Menge*

$$L_a = \{w \mid aw \in L\} \in \mathcal{L}(P)$$

nicht nur aus dem Leerwort besteht, so liegt sie in $\mathcal{L}(P)$.

Beweis. Es sei L eine Sprache aus $\mathcal{L}(P)$. Dann gibt es eine programmierte Grammatik $G = (N, T, Lab, P, S)$, die L erzeugt. Ferner sei

$$q = \max\{|w| \mid (l, A \to w, \sigma, \varphi) \in P\}.$$

Wenn $q = 1$ gilt, so werden nur Wörter der Länge 1 erzeugt. Daher ist L_a entweder leer (wenn a nicht erzeugt wird), oder es ist $L_a = \{\lambda\}$. Im ersten Fall wird L_a von der programmierten Grammatik $(\{S\}, T, \{l\}, \{(l, S \to S, \{l\}, \emptyset)\}, S)$ erzeugt. Der zweite Fall braucht wegen der Voraussetzungen des Lemmas nicht betrachtet zu werden. Wir können daher im Folgenden annehmen, dass $q \geq 2$ gilt.
Sei nun w eine Satzform der Länge k mit $q + 1 \leq k \leq 2q$. Dann gibt es eine Ableitung

$$D : S \Longrightarrow_{p_1} w_1 \Longrightarrow_{p_2} w_2 \Longrightarrow_{p_3} w_3 \Longrightarrow_{p_4} \ldots \Longrightarrow_{p_n} w_n = w.$$

Wir definieren dann $s(w, D)$ und $v(w, D)$ als das Erfolgs- bzw. Fehlerfeld von p_n. Offensichtlich gibt es nur endliche viele w und für jedes w nur endlich viele verschiedene Ableitungen.
Wir setzen zuerst

$$N' = N \cup \{(x, y) \mid x, y \in N \cup T\} \cup \{S'\}$$

(die neuen Buchstaben (x, y) stehen für ein Wort der Länge 2 und werden im Folgenden anstelle des Anfangsstückes der Länge 2 verwendet). Für eine Regel $(l, A \to x_1 x_2 \ldots x_n, \sigma, \varphi)$ aus P konstruieren wir die Regeln

$$(l, A \to x_1 x_2 \ldots x_n, \bigcup_{k \in \sigma} \{k, k', k''\} \cup \{u_x \mid x \in T\}, \bigcup_{t \in \varphi} \{t, t', t''\} \cup \{u_x \mid x \in T\}),$$

$$(l', (y, A) \to (y, x_1) x_2 x_3 \ldots x_n, \bigcup_{k \in \sigma} \{k, k', k''\} \cup \{u_x \mid x \in T\}, \bigcup_{t \in \varphi} \{t, t', t''\} \cup \{u_x \mid x \in T\}),$$

$$(l'', (A, y) \to (x_1, y), \bigcup_{k \in \sigma} \{k, k', k''\} \cup \{u_x \mid x \in T\}, \bigcup_{t \in \varphi} \{t, t', t''\} \cup \{u_x \mid x \in T\})$$

für $n = 1$,

$$(l'', (A, y) \to (x_1, x_2) x_3 \ldots x_n y, \bigcup_{k \in \sigma} \{k, k', k''\} \cup \{u_x \mid x \in T\}, \bigcup_{t \in \varphi} \{t, t', t''\} \cup \{u_x \mid x \in T\})$$

für $n \geq 2$.

Durch Regeln mit den Labels l' und l'' wird abgesichert, dass auch eine Anwendung auf den beiden ersten gekoppelten Buchstaben möglich wird. Außerdem benutzen wir Regeln der Form

$((w), S' \to w, \emptyset, \emptyset)$ für $w \in L, |w| \leq q$,

$((w, D), S' \to w, \bigcup_{k \in s(w,D)} \{k, k', k''\} \cup \{u_x \mid x \in T\}, \bigcup_{t \in v(w,D)} \{t, t', t''\} \cup \{u_x \mid x \in T\})$

für eine Satzform w von G mit $q + 1 \leq |w| \leq 2q$ und eine Ableitung D von w,

$(u_x, (a, x) \to x, \emptyset, \emptyset)$ für $x \in T$.

Durch die Regeln der beiden erstgenannten Formen werden die erforderlichen kurzen Wörter der Sprache bzw. Satzformen der Grammatik G direkt erzeugt. Durch die Anwendung der letztgenannten Regeln wird der erste Buchstabe a gestrichen, wodurch ein Wort aus L_a erzeugt wird; ist der erste Buchstabe nicht a, so kann der gekoppelte Buchstabe nicht eliminiert werden, d.h., die Ableitung kann nicht terminieren; falls eine solche Regel angewendet wird, so stoppt die Ableitung, da Erfolgs- und Fehlerfeld leer sind, d.h. eine solche Regel kann nur als letzte der Ableitung angewendet werden.
Nach diesen Ausführungen ist leicht zu sehen, dass für die programmierte Grammatik

$$G' = (N', T, \bigcup_{l \in Lab} \{l, l', l''\}, P', S'),$$

bei der P' aus allen oben konstruierten Regeln besteht, $L(G') = L_a$ gilt. □

Lemma 7.9 $\mathcal{L}(P) \subseteq \mathcal{L}(RC)$.

Beweis. Es sei L eine Sprache aus $\mathcal{L}(P)$ über dem Alphabet V. Für $a \in V$ setzen wir $L_a = \{w \mid aw \in L\}$. Dann gilt

$$L = \bigcup_{a \in V} aL_a.$$

Wir werden nun zeigen, dass für jedes $a \in V$ die Sprache aL_a in $\mathcal{L}(RC)$ liegt. DA es leicht zu zeigen ist, dass $\mathcal{L}(RC)$ unter Vereinigung abgeschlossen ist (der Standardbeweis aus Abschnitt 5 ist nur leicht zu modifizieren), ist dann auch L in $\mathcal{L}(RC)$.
Nach Lemma 7.8 wird L_a von einer programmierten Grammatik $G = (N, T, Lab, P, S)$ erzeugt. Dabei sei p_l stets die Regel mit Label l. Wir setzen

$$G' = (N \cup \{A_l \mid A \in N, l \in Lab\} \cup Lab \cup \{S'\}, T, P', S'),$$

wobei P' aus allen Regeln besteht, die wie folgt aussehen: Für das neue Startsymbol S' gibt es die Regeln

$(S' \to lS, \emptyset, \emptyset)$, wobei l Label einer Regel mit linker Seite S ist

(mit einer solchen Regel beginnt die Ableitung und es soll anschließend die Anwendung der Regel $(l, S \to v_l, \sigma(l), \varphi(l))$ in G simuliert werden). Für jede Regel $(l, A \to w, \sigma(l), \varphi(l)) \in P$ führen wir die Regeln

$$\begin{aligned}
&(A \to A_l, \{l\}\{B_k \mid B \in N, k \in Lab\}),\\
&(l \in l', \{A_l\}, \emptyset,) \text{ für alle } l' \in \sigma(l),\\
&(A_l \to w, \emptyset, \{l\}),\\
&(l \to l', \emptyset, \{A\}) \text{ für alle } l' \in \varphi(l),\\
&(l \to a, \emptyset, N \cup \{B_k \mid B \in N, k \in Lab\})
\end{aligned}$$

ein (liegt die Satzform $luAv$ vor, so ist nur die Ableitung $luAv \Longrightarrow luA_lv \Longrightarrow l'uA_lv \Longrightarrow l'uwv$ möglich, d.h. ein Ableitungsschritt aus G wurde simuliert und danach ist sowohl in G als auch in G' mit einer Regel $l' \in \sigma(l)$ fortzusetzen, kommt A in der Satzform lw nicht vor so ist nur $(l \to l', \emptyset, \{A\})$ anwendbar, wodurch $l'w$ entsteht, d.h. wir haben erneut

einen Ableitungschritt aus G simuliert; die Regel $l \to a$ ist nur als letzte anwendbar, da sonst kein anderes Nichtterminal mehr vorhanden sein darf).
Aus diesen Erklärungen folgt, dass eine Ableitung

$$S \Longrightarrow_{p_{l_1}} w_1 \Longrightarrow_{p_{l_2}} w_2 \Longrightarrow_{p_{l_3}} w_3 \Longrightarrow_{p_{l_4}} \ldots \Longrightarrow_{p_{l_n}} w_n = z \in L(G) = L_a$$

in G genau dann existiert, wenn in G' eine Ableitung der Form

$$S' \Longrightarrow l_1 S \Longrightarrow l_2 w_1 \Longrightarrow l_3 w_2 \Longrightarrow l_4 w_3 \Longrightarrow \ldots \Longrightarrow l_n w_{n-1} \Longrightarrow l_{n+1} w_n \Longrightarrow a w_n = az$$

existiert. Damit folgt $L(G') = aL_a$. □

Satz 7.10 $\mathcal{L}(CF) \subset \mathcal{L}(RC) = \mathcal{L}(P) \subset \mathcal{L}(CS)$.

Beweis. $\mathcal{L}(CF) \subseteq \mathcal{L}(RC)$ ergibt sich einfach daraus, dass jede kontextfreie Grammatik als eine Grammatik mit Auswahlkontext aufgefasst werden kann, wir wie oben festgestellt haben. Die Echtheit der Inklusion folgt aus den Beispielen 7.2 und 7.3. Die Gleichheit $\mathcal{L}(RC) = \mathcal{L}(P)$ folgt aus den Lemmata 7.7 und 7.8.
Der Beweis der Inklusion $\mathcal{L}(RC) \subseteq \mathcal{L}(CS)$ ist leicht dadurch zu erbringen, dass eine kontextsensitive Grammatik erzeugt wird, bei der stets die Anwendbarkeit in Analogie zum Beweis von Lemma 7.7 getestet wird (durch Regeln $XA \to AX$ bewegt sich ein Symbol von links nach rechts über die Satzform und merkt sich dabei die vorkommenden Nichtterminale). Den Beweis der Echtheit der Inklusion geben wir hier nicht. □

Damit haben wir eines der eingangs formulierten Ziele erreicht; wir haben eine Sprachfamilie, die echt zwischen denen der kontextfreien und kontextsensitiven Sprachen liegt. Wir merken noch an, dass auch hinsichtlich der Entscheidbarkeits- bzw. Komplexitätsfragen eine verbesserte Situation vorliegt, denn es gelten die folgenden Aussagen, auf deren Beweis wir hier verzichten.

Satz 7.11 *i)* $\mathcal{L}(P)$ *ist eine AFL.*
ii) Das Mitgliedsproblem für programmierte Grammatiken is **NP***-vollständig.*
iii) Das Leerheitsproblem für programmierte Grammatiken, bei denen jedes Fehlerfeld leer ist, ist entscheidbar. □

Kapitel 8

Ergänzungen V : Eindeutigkeit kontextfreier Grammatiken

Ein Problem, das bei natürlicher Sprachen, Programmiersprachen und formalen Sprachen gleichermaßen störend ist, ist die Mehrdeutigkeit von Sätzen. Als ein Beispiel aus der deutschen Sprachen nehmen wir den Satz

Er öffnete die Kiste mit dem Hammer.

Hier ist nicht klar, ob der Hammer zum Öffnen benutzt wird oder ob der Hammer zur Kiste gehört. Ein bekanntes Beispiel der englischen Sprache ist der Satz

They are flying machines.

Welche Bedeutung hier gemeint ist kann beispielsweise von der Ableitung dieses Satzes gewonnen werden. Wir haben die folgenden beiden Ableitungen:

$$\begin{aligned}
\text{Satz} &\Longrightarrow \text{Subjekt Prädikat Objekt .} \\
&\Longrightarrow \text{They Prädikat Objekt .} \\
&\Longrightarrow \text{They (Verlaufsform eines Verbs) Objekt .} \\
&\Longrightarrow \text{They are flying Objekt .} \\
&\Longrightarrow \text{They are flying machines .}
\end{aligned}$$

und

$$\begin{aligned}
\text{Satz} &\Longrightarrow \text{Subjekt Prädikat Objekt .} \\
&\Longrightarrow \text{They Prädikat Objekt .} \\
&\Longrightarrow \text{They are Objekt .} \\
&\Longrightarrow \text{They are Adjektiv Substantiv .} \\
&\Longrightarrow \text{They are flying Substantiv .} \\
&\Longrightarrow \text{They are flying machines .}
\end{aligned}$$

Um Eindeutigkeit zu erreichen scheint es daher sinnvoll zu sein, dass die Ableitung eindeutig ist. In dieser Allgemeinheit ist die Forderung aber unbrauchbar, da in einer kontextfreien Grammatik keine Beschränkung hinsichtlich des Buchstaben, auf den die nächste Regel angewendet wird, gegeben ist. Dies ändert sich aber, wenn wir uns auf Linksableitungen beschränken, bei denen immer das am weitesten links stehende Nichtterminal in der Satzform zu ersetzen ist. Die beiden oben angegebenen Ableitungen des englischen Satzes sind beide Linksableitungen, aber sie sind verschieden.

Definition 8.1 *Eine kontextfreie Grammatik G heißt eindeutig, wenn für jedes Wort aus $w \in L(G)$ genau eine Linksableitung bez. G existiert. Anderenfalls heißt G mehrdeutig.*

Es ist offensichtlich, dass dies äquivalent zu der Forderung ist, dass es für jedes Wort w aus $L(G)$ genau einen Ableitungsbaum gibt, da ein jeder Ableitungsbaum auch Ableitungsbaum einer Linksableitung ist.
Wir betrachten zwei Beispiele.

Beispiel 8.2 Die Grammatik $GH_1 = (\{S\}, \{a, b\}, \{S \to aSb, S \to ab\}, S)$ (die $\{a^n b^n \mid n \geq 1\}$ erzeugt), ist offensichtlich eindeutig, denn zuerst ist die erste Regel eine beliebige Anzahl von Malen anzuwenden und zum Schluss die zweite Regel genau einmal.

Beispiel 8.3 Wir betrachten die lineare Grammatik

$$H_2 = (\{S\}, \{a, b\}, \{S \to aSa, S \to aS, S \to b\}, S),$$

die die Sprache

$$L(H_2) = \{a^n b a^m \mid n \geq m \geq 0\}$$

erzeugt. Die Grammatik H_2 ist mehrdeutig, denn für das Wort a^2ba haben wir die Linksableitungen

$$S \Longrightarrow aSa \Longrightarrow aaSa \Longrightarrow aaca \text{ und } S \Longrightarrow aS \Longrightarrow aaSa \Longrightarrow aaca,$$

die voneinander verschieden sind.

Der folgende Satz, dessen Beweis (durch Reduktion auf das Postsche Korrespondenzproblem) wir dem Leser als Übungsaufgabe überlassen, zeigt, dass Eindeutigkeit kein einfaches Konzept für Grammatiken ist.

Satz 8.4 *Es ist algorithmisch unentscheidbar, ob eine gegebene kontextfreie Grammatik eindeutig oder mehrdeutig ist.* □

Wir übertragen nun den Begriff der Eindeutigkeit auf Sprachen.

Definition 8.5 *Eine (kontextfreie Sprache Sprache L heißt eindeutig, wenn es eine eindeutige kontextfreie Grammatik G mit $L = L(G)$ gibt. Ansonsten heißt eine kontextfreie Sprache mehrdeutig.*

Eindeutige Sprachen sind also diejenigen, die von eindeutigen Grammatiken erzeugt werden. Es ist aber nicht ausgeschlossen, dass es auch eine mehrdeutige Grammatik gibt, die die eindeutige Sprache erzeugt (siehe das folgende Beispiel). Für mehrdeutige Sprachen gibt es keine sie erzeugende eindeutige Grammatiken, d.h. alle die Sprache erzeugenden Grammatiken sind mehrdeutig.

Beispiel 8.6 Die Sprache $L(H_2)$ aus Beispiel 8.3 ist eindeutig. Wir müssen also zeigen, dass es außer der mehrdeutigen Grammatik H_2 noch eine eindeutige Grammatik H_3 gibt, die auch L erzeugt. Dies leistet z.B. die Grammatik

$$H_3 = (\{S, A\}, \{a, b\}, \{S \to aSa, S \to A, A \to aA, A \to b\}, S).$$

Die Grammatik ist eindeutig, da bei jeder Ableitung in H_3 zuerst die Regel $S \to aSA$ eine gewisse Anzahl von Malen angewendet wird (sagen wir m mal), dann eine Anwendung von $S \to A$ folgt, worauf $A \to aA$ wieder eine beliebige Anzahl von Malen verwendet wird (sagen wir n mal), um dann mit $A \to b$ zu terminieren. Offensichtlich wird so $a^n a^m b a^m = a^{n+m} b a^m$ erzeugt. Hiermit ist dann auch $L(H_3) = L(H_2)$ gezeigt.

Eigentlich ist es aufgrund der Definition 8.5 nicht klar, dass es eine mehrdeutige Sprache gibt (denn es könnte sein, dass jede kontextfreie Sprache durch eine eindeutige Grammatik erzeugbar ist. Wir wollen nun zeigen, dass es mehrdeutige kontextfreie Sprachen gibt. Für den Beweis brauchen wir die folgende Normalform für eindeutige Grammatiken.

Definition 8.7 *i) Für eine kontextfreie Grammatik $G = (N, T, P, S)$ definieren wir die Menge $U(G)$ durch*

$$U(G) = \{A \mid A \in N,\ A \Longrightarrow^* xAy \text{ für gewisse } x, y \in T^* \text{ mit } xy \neq \lambda\}.$$

ii) Eine kontextfreie Grammatik $G = (N, T, P, S)$ heißt selbstzyklisch, wenn sie folgende Bedingungen erfüllt:

1. *Für jedes Nichtterminal $A \in N$gibt es eine terminierende Ableitung, d.h. eine Ableitung $A \Longrightarrow^* w$ mit $w \in T^*$.*
2. *Für jedes Nichtterminal $A \in N$ gibt es eine Ableitung $S \Longrightarrow^* w_1 A w_2$ mit gewissen $w_1, w_2 \in T^*$.*
3. *Entweder ist S in $U(G)$ enthalten oder S kommt in jeder Ableitung für jedes Wort $w \in L(G)$ genau einmal vor.*
4. *Jedes Nichtterminal $A \in N \setminus \{S\}$ ist in $U(G)$ enthalten.*

Lemma 8.8 *Für jede eindeutige Grammatik G mit $L(G) \neq \emptyset$ gibt es eine selbstzyklische eindeutige Grammatik G' mit $L(G) = L(G')$.*

Beweis. Die Bedingungen 1. und 2. lassen sich einfach dadurch erfüllen, dass wir alle Nichtterminale, die 1. oder 2. (oder beide Bedingungen) nicht erfüllen und alle Regeln, in denen diese auf der linken oder rechten Seite vorkommen, streichen. Es ist einfach zu sehen, dass hierdurch die erzeugte Sprache Sprache nicht verändert wird, wenn $L(G)$ nicht leer ist (ein formaler Beweis ist im Wesentlichen schon im Beweis von Satz 5.21 enthalten). Durch diesen Schritt werden existierende terminierende Ableitungen nicht verändert, womit Eindeutigkeit erhalten bleibt.
Wir zeigen nun, dass Bedingung 3. auch erfüllt ist. Dazu nehmen wir an, dass es eine Ableitung gibt, in der S mindestens zweimal vorkommt. Dann gilt $S \Longrightarrow^* w_1 S w_2 \Longrightarrow^* w_1 w w_2 \in T^*$. Falls $w_1 w_2 \neq lambda$ gilt, so ist $S \in U(G)$. Daher nehmen wir nun an,

dass $w_1 = w_2 = \lambda$ ist. Dann gibt es Linksableitungen $S \Longrightarrow^* SU$ und $U \Longrightarrow^* \lambda$, wobei $U \in (N \cup T)^*$ ist. Dann haben wir für ein Wort z aus $L(G)$ aber die zwei verschiedenen Linksableitungen

$$S \Longrightarrow^* z \quad \text{und} \quad S \Longrightarrow^* SU \Longrightarrow^* zU \Longrightarrow z.$$

Folglich ist G nicht eindeutig im Widerspruch zur Voraussetzung.
Es sei $N = \{S, A_1, A_2, \ldots A_k\}$. Wir konstruieren nun eine Folge von eindeutigen Grammatiken $G_0, G_1, \ldots, G_k$ derart, dass für $1 \leq i \leq k$ für die Grammatik $G_j = (N_j, T, P_j, S)$ gilt, dass

$$A_j \notin N_i \text{ oder } A_j \in U(G_i) \text{ für } 1 \leq j \leq i \tag{8.1}$$

gilt. Die Konstruktion der Grammatiken erfolgt induktiv.
Offenbar erfüllt $G_0 = G$ die Forderung (8.1).
Sei nun $G_{i-1} = (N_{i-1}, T, P_{i-1}, S)$ bereits konstruiert. Jedes A_j mit $1 \leq j \leq i-1$ kommt also entweder in N_{j-1} nicht vor oder liegt in $U(G_{-1})$. Falls $A_i \in U(G_{i-1})$ liegt, so setzen wir $G_i = G_{i-1}$ und folglich erfolgt G_i die Forderung (8.1). Falls $A_i \notin U(G_{i-1})$ gilt, so seien $A_i \to p_1,\ A_i \to p_2, \ldots,\ A \to p_r$ die Regeln mit rechter Seite A_i in G_{i-1}. Dann kann A_i in keinem der Wörter p_s, $1 \leq s \leq r$, vorkommen. Denn wenn $A_i \to p_s = q_1 A_i q_2$ für ein s, $1 \leq s \leq r$, gilt so gibt es auch eine Ableitung $A_i \Longrightarrow^* q_1' A_i q_2'$ mit $q_1', q_2' \in T^*$, $q_1 \Longrightarrow^* q_1'$ und $q_2 \Longrightarrow^* q_2'$. Bei $q_1' q_2' \neq \lambda$ ist A_i in $U(G_{i-1})$ im Widerspruch zu unserer Voraussetzung, und bei $q_1' q_2' = \lambda$ existiert eine Ableitung $A_i \Longrightarrow^* A_i$, die analog zu Obigem zu einem Widerspruch zur Eindeutigkeit führt. Nun konstruieren wir G_i, indem wir die Regel mit rechter Seite A_i alle streichen und jedes Vorkommen von A_i in einer linken Seite durch alle p_s, $1 \leq s \leq r$, ersetzen. Die so konstruierte erzeugt die gleiche Sprache wie $L(G_{i-1})$, da nur die einmal vorzunehmenden Ersetzungen $A_i \to p_s$ schon in der Regel realisiert sind. Außerdem ist klar, dass die Konstruktion die Eindeutigkeit erhält. Weiterhin folgt aus der Konstruktion sofort, dass G_i Forderung (8.1) erfüllt. [1] □

Satz 8.9 *Die Sprache*

$$L = \{a^n b^n c^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m \mid n \geq 1, m \geq 1\}$$

ist mehrdeutig.

Beweis. Angenommen, L ist nicht mehrdeutig. Dann gibt es eine eindeutige Grammatik G mit $L = L(G)$. Wegen Lemma 8.8 können wir ohne Beschränkung der Allgemeinheit voraussetzen, dass G selbstzyklisch ist.
Wir zeigen nun zuerst, dass jedes Nichtterminal $A \neq S$ von genau einem der folgenden Typen ist:

Typ (a) Es gibt eine natürliche Zahl $m \geq 1$ und Wörter u und v so, dass $A \Longrightarrow uAv$ und entweder $uv \in \{a\}^+$ oder $uv \in \{c\}^+$ gelten.

Typ (b) Es gibt eine natürliche Zahl $m \geq 1$ derart, dass $A \Longrightarrow^* a^m A b^m$ gilt.

[1] Der Leser mache sich klar, dass die Aussage von Lemma 8.8 auch ohne die Eindeutigkeit gilt. Denn die Eindeutigkeit wird nur benutzt, um Ableitungen der Form $A \Longrightarrow^* A$ auszuschließen. Dies kann aber auch dadurch geschehen, dass man zuerst Regel der Form $A \to \lambda$ und $A \to B$ eliminiert, was nach den Normalformen aus Abschnitt 2.2 möglich ist.

Typ (c) Es gibt eine natürliche Zahl $m \geq 1$ derart, dass $A \Longrightarrow^* b^m A c^m$ gilt.

Wegen der Selbstzyklizität von G gibt es für A eine Ableitung $A \Longrightarrow^* uAv$ mit $uv \in T^*$ und $uv \neq \lambda$. Falls in u zwei Buchstaben aus $\{a, b, c\}$ vorkommen, so hat u die Form $u = u_1 x u_2 y u_3$ mit $x, y \in \{a, b, c\}$ und $x \neq y$. Wegen der Existenz einer Ableitung $S \Longrightarrow^* z_1 A z_2 \Longrightarrow^* z_1 z z_2$ mit $z_1, z_2, z \in T^*$ (wegen der Forderungen 1. und 2. bei der Selbstzyklizität) haben wir auch eine Ableitung der Form

$$\begin{aligned} S &\Longrightarrow^* z_1 A z_2 \Longrightarrow^* z_1 u_1 x u_2 y u_3 A v z_2 \Longrightarrow z_1 u_1 x u_2 y u_3 u_1 x u_2 y u_3 A v v z_2 \\ &\Longrightarrow^* z_1 u_1 x u_2 y u_3 u_1 x u_2 y u_3 z v v z_2 \in T^*. \end{aligned}$$

Damit liegt $z_1 u_1 x u_2 y u_3 u_1 x u_2 y u_3 z v v z_2$ in $L(G)$. Dies widerspricht aber $L(G) = L$, da in L der Buchstabe x nicht gleichzeitig vor und nach dem Buchstaben y mit $x \neq y$ auftreten kann. Analog kann man zeigen, dass auch q keine Vorkommen von zwei verschiedenen Buchstaben enthalten kann. Wir diskutieren nun die möglichen Fälle für u und v.
Es sei $u = a^m$ für eine Zahl $m \geq 1$. Ist auch $v \in \{a\}^*$, so ist A vom Typ (a). Es sei nun $q = b^n$ für ein $n \geq 1$. Wegen den Forderungen 1. und 2. der Selbstzyklizität gibt es eine Ableitung

$$S \Longrightarrow^* z_1 A z_2 \Longrightarrow^* a^r b^s c^t \tag{8.2}$$

mit $r = s$ oder $s = t$. Wenden wir auf A α-mal hintereineinander $A \Longrightarrow^* a^m A b^n$ an, so ergibt sich das Wort $a^{r+\alpha m} b^{s+\alpha n} c^t$. Bei der Wahl $\alpha = t$ gilt sicher $s + tn \neq t$. Folglich gilt $r + \alpha m = s + \alpha n$. Dann ergibt sich bei $(\alpha + 1)$-maliger Anwendung noch $r + (\alpha + 1)m = s + (\alpha + 1)n$, woraus dann $n = m$ folgt. Damit ist A vom Typ (b). Es sei nun $v = c^n$ für ein $n \geq 1$. Dann ergibt wieder aus (8.2) die Ableitbarkeit von $a^{r+\alpha m} b^s c^{t+\alpha n} \in L(G)$. Durch die Wahl von α kann erreicht werden, dass $r + \alpha m$ und $t + \alpha n$ von s verschieden sind. Dies widerspricht $L = L(G)$.
Es sei $u = b^m$ für ein $m \geq 1$. Dann gilt wegen der Reihenfolge der Buchstaben in den Wörtern von L entweder $v = b^n$ oder $v = c^n$ für ein $n \geq 0$. Im ersten Fall erhalten wir aus (8.2), dass $a^r b^{s+\alpha m+\alpha n} c^t$ in $L(G)$ liegt, und erhalten einen Widerspruch, da bei passender Wahl von α das Wort $a^r b^{s+\alpha m+\alpha n} c^t$ nicht in L liegt. Im zweiten Fall können wir wie oben $m = n$ zeigen, womit A vom Typ (c) ist.
Es sei $u = c^m$ für ein $m \geq 1$. Dann gilt auch $v \in \{c\}^+$ und damit liegt Typ (a) vor.
Falls $u = \lambda$ ist, so ist $v \neq \lambda$. Bei $v \in \{a\}^+$ und $v \in \{c\}^+$ ist A vom Typ (a). Bei $v = b^n$ für ein $n \geq 0$ können wir alle Wörter $a^r b^{t+\alpha n} c^t$ in G erzeugen, die aber bei passender Wahl von α nicht in L liegen.
Damit ist gezeigt, dass jedes Nichtterminal $A \neq S$ mindestens einem Typ angehört. Wir zeigen jetzt, dass es nicht zwei verschiedenen Typen angehören kann. Angenommen, A ist vom Typ (a) und vom Typ (b). Dann gibt es Ableitungen $A \Longrightarrow^* x^f A x^g$ mit $x \in \{a, c\}$, $f + g \geq 1$ und $A \Longrightarrow^* a^m A b^m$ für ein $m \geq 1$. Dann gibt es aber auch eine Ableitung $A \Longrightarrow^* a^m A b^m \Longrightarrow^* a^m x^f A x^g b^m$. Wegen der Reihenfolge der Buchstaben ist $x = c$ nicht möglich. Bei $x = a$ und $g \neq 0$ haben wir eine Ableitung $A \Longrightarrow^* uAv$, bei der v zwei verschiedene Buchstaben enthält. Oben haben wir gezeigt, dass dies unmöglich ist. Falls $g = 0$ ist, so muss nach Obigem $m + f = m$ gelten. Dies impliziert aber $f = 0$ im Widerspruch zu $f + g \geq 1$.
Analog erhalten wir, dass es nicht möglich ist, dass A sowohl vom Typ (a) als auch vom Typ (b) ist.

Sei daher A vom Typ (b) und vom Typ (c). Dann haben wir Ableitungen $A \Longrightarrow^* a^m A b^m$ und $A \Longrightarrow^* b^{m'} A c^{m'}$, aus denen sich die Ableitung $A \Longrightarrow^* a^m A b^m \Longrightarrow^* a^m b^{m'} A c^{m'} b^m$ ergibt, die nach Obigem unmöglich ist.
Folglich ist jedes Nichtterminal A von genau einem der Typen (a), (b) oder (c).
Wegen der Selbstzyklizität der Grammatik G kommt S in jeder genau einmal vor, oder $S \in U(G)$. Wir zeigen nun, dass die zweite Möglichkeit nicht eintreten. Wir nehmen das Gegenteil an und werden einen Widerspruch herleiten.
Falls $S \in U(G)$ gilt, so gibt es terminale Wörter x und y mit $S \Longrightarrow^* xSy$ und $xy \neq \lambda$. Da abc^2, a^2bc und abc in L liegen, gibt es Ableitungen $S \Longrightarrow^* abc^2$ und $S \Longrightarrow^* a^2bc$. Damit gibt es auch die Ableitungen

$$S \Longrightarrow xSy \Longrightarrow xabc^2y \text{ und } S \Longrightarrow xSy \Longrightarrow xa^2bcy.$$

Somit liegen $xabc^2y$ und xa^2bcy auch in L. Da außerdem x und y Potenzen von einem Buchstaben a oder b oder c sein müssen, gibt es für x und y nur die Möglichkeiten

$$x = a^m \text{ und } y = \lambda \quad \text{oder} \quad x = \lambda \text{ und } y = c^n \quad \text{oder} \quad x = a^m \text{ und } y = c^n$$

mit gewissen positiven ganzen Zahlen m und n. Dann gilt im ersten Fall $xabc^2y = a^{m+1}bc^2$, im zweiten Fall $xa^2bcy = a^2bc^{n+1}$ und im dritten Fall $xa^2bcy = a^{m+2}bc^{n+1}$ mit gewissen $m, n \geq 1$. Dies widerspricht aber in allen Fällender Form der Wörter in L.
Es sei

$$\begin{aligned} S &\Longrightarrow^* u_1A_1v_1 \Longrightarrow^* u_1u_2A_2v_2 \Longrightarrow^* \ldots \Longrightarrow^* u_1u_2\ldots u_rA_rv_rv_{r-1}\ldots v_1 \\ &\Longrightarrow^* u_1u_2\ldots u_rwv_rv_{r-1}\ldots v_1 \end{aligned}$$

eine Ableitung, in der keine Nichtterminale vom Typ (b) oder (c) vorkommen. Ferner beginne jeder Teilableitung

$$u_1u_2\ldots u_{i-1}A_{i-1}v_{i-1}\ldots v_1 \Longrightarrow^* u_1u_2\ldots u_{i-1}u_iA_iv_iv_{i-1}\ldots v_1$$

mit der Anwendung einer Regel, deren rechte Seite mindestens einmal den Buchstaben b enthält. Dann sind die Buchstaben $A_1, A_2, \ldots, A_r$ alle paarweise verschieden. Wäre dies nicht der Fall, so gibt es eine Ableitung $A_i \Longrightarrow^* x_1bx_2A_ix_3$ oder $A_i \Longrightarrow^* x_1Ax_2bx_3$, womit A_i vom Typ (b) oder (c) wäre. Damit werden Regeln mit Vorkommen von b auf der rechten Seite in einer Ableitung ohne Nichtterminale vom Typ (b) oder (c) höchstens μ mal für ein passendes μ angewendet. Daher enthält jedes Wort, das durch eine Ableitung, in der keine Nichtterminale vom Typ (b) oder (c) vorkommen, höchsten $s \cdot \mu$ Vorkommen von b (jede Regel mit einem Vorkommen von b produziert höchstens s Vorkommen von b.
Für jedes Nichtterminal A vom Typ (b) oder (c) wählen wir nun eine Zahl $m(A)$ derart, dass $A \Longrightarrow^* a^{m(A)}Ab^{m(A)}$ bzw. $A \Longrightarrow^* b^{m(A)}Ac^{m(A)}$ gilt. Wir wählen nun p so, dass
— $p > s \cdot \mu$ ist,
— für jedes X vom Typ (b) oder (c) $m(X)$ ein Teiler von p ist. Wir betrachten nun eine Ableitung von $w = a^pb^pc^{2p}$. Da mehr als $s \cdot \mu$ Vorkommen von b in w sind, muss die Ableitung mindestens ein Nichtterminal A vom Typ (b) oder (c) enthalten. Wir nehmen nun an, dass ein Nichtterminal vom Typ (c) vorkommt. Dann hat die Ableitung die Form $S \Longrightarrow^* u_1Au_2 \Longrightarrow^* w$ mit A vom Typ (c). Es sei $m(A) \cdot r = p$. Durch r-malige

Anwendung der Ableitung $A \Longrightarrow^* b^{m(A)}Ac^{m(A)}$ erzeugen wir p zusätzliche Buchstaben b und p zusätzliche Buchstaben c, d.h. wir erhalten $S \Longrightarrow^* a^p b^{2p} c^{3p}$ und damit ein Wort, dass nicht in L liegt. Daher kommen in der Ableitung von w nur Nichtterminale der Typen (a) und (b) und mindestens ein Nichtterminal A' vom Typ (b) vor. Dann gibt es ein s mit $m(A') \cdot s = p$. Wir wenden $A' \Longrightarrow^* a^{m(A')}A'b^{m(A')}$ zusätzlich s-mal an, wodurch $a^{2p}b^{2p}c^{2p}$ entsteht. Daher gibt es eine Ableitung von $a^{2p}b^{2p}c^{2p}$, in der nur Nichtterminale der Typen (a) und (b) und mindestens ein Nichtterminal vom Typ (b) vorkommen. Mit gleicher Argumentation kann man (ausgehend von $a^{2p}b^pc^p$) zeigen, dass es eine Ableitung von $a^{2p}b^{2p}c^{2p}$ gibt, in der nur Nichtterminale der Typen (a) und (c) und mindestens ein Nichtterminal vom Typ (c) vorkommen. Damit gibt es für $a^{2p}b^{2p}c^{2p}$ zwei verschiedene Linksableitungen, was der Eindeutigkeit widerspricht. □