11 Kellerautomaten

Die Erkennung von Sprachen durch endliche Automaten ist angemessen schnell, aber in ihrem Anwendungsspektrum ziemlich eingeschränkt, weil ein endlicher Automat während des Abarbeitens eines Eingabewortes nur eine beschränkte Zahl von Informationen speichern kann, die durch die Zustände vorgegeben ist. Insbesondere kann nur bis zu einer Schranke gezählt werden, und nur beschränkte Abschnitte des Eingabewortes können für spätere Vergleiche aufgehoben werden. Um dagegen eine Sprache wie

$$L_{balance} = \{a^n b^n \mid n \in \mathbb{N}\}$$

zu erkennen, muss man unbeschränkt zählen können. Oder um das Wortproblem von

$$L_{palindrom} = \{w \in T^* \mid w = trans(w)\}$$

zu lösen, ist es nötig, die erste Hälfte des Wortes zwischenzuspeichern, damit sie mit der zweiten Hälfte des Wortes verglichen werden kann.[1]

Um die Technik des Erkennens endlicher Automaten beibehalten zu können, aber gleichzeitig auch in der Lage zu sein, mitzuzählen und sich beliebige Teile des gelesenen Wortes zu merken, werden die endlichen Automaten um einen Keller (Stapel, stack) als zweites Speichermedium erweitert. Ein Zustandsübergang wird dann auch vom obersten Kellersymbol abhängig gemacht, das dabei durch eine Sequenz von Kellersymbolen ersetzbar ist. Auf dem Keller werden in jedem Schritt also eine POP-Operation sowie eine Folge von PUSH-Operationen ausgeführt, die auch leer sein kann. Außerdem werden noch – anders als bei endlichen Automaten gemäß Abschnitt 3.1 – Zustandsübergänge erlaubt, bei denen kein Eingabesymbol gelesen wird.

11.1 Konzept des Kellerautomaten

Das Modell des Kellerautomaten formalisiert diese Beschreibung. Die Arbeitsweise des Kellerautomaten wird durch fortgesetzte Übergänge zwischen Konfigurationen beschrieben, wobei eine Konfiguration den aktuellen Zustand, das noch zu lesende Eingabewort und ein aktuelles Kellerwort umfasst. Einzelne Übergänge sind durch die Zustandsüberführung festgelegt. Ziel ist es, eine Anfangskonfiguration mit Anfangszustand, dem vollständigen Eingabewort und dem initialen Kellersymbol als Startkeller in eine Endkonfiguration zu überführen, bei der der aktuelle Zustand ein Endzustand und die Eingabe vollständig gelesen ist. Wenn das gelingt, ist das Eingabewort vom Kellerautomaten erkannt.

1. Ein *Kellerautomat* ist ein System $K = (Z, I, C, d, s_0, F, c_0)$ mit einer endlichen Menge Z von *Zuständen*, einem endlichen Alphabet I von *Eingaben*, einem endlichen

[1] Dabei wird vorausgesetzt, dass das Eingabewort nur einmal von links nach rechts gelesen werden kann.

Alphabet C von *Kellersymbolen*, einer *Zustandsüberführung* d, die jedem Zustand s, jeder Eingabe x und jedem Kellersymbol c eine Menge von Paaren aus Zuständen und Kellerwörtern $d(s, x, c) \subseteq Z \times C^*$ sowie jedem Zustand s und jedem Kellersymbol c eine entsprechende Menge $d(s, -, c) \subseteq Z \times C^*$ zuordnet, einem *Anfangszustand* $s_0 \in Z$, einer Menge von *Endzuständen* $F \subseteq Z$ und einem *initialen Kellersymbol* c_0.

2. Ein Kellerautomat lässt sich ähnlich einem endlichen Automaten graphisch darstellen. Unterschiedlich ist lediglich die Beschriftung der Kanten, wie Abbildung 7 illustriert.

für $(s', \alpha) \in d(s, x, c)$ für $(s', \alpha) \in d(s, -, c)$

Abbildung 7: Kantenbeschriftungen bei Kellerautomaten

3. Eine *Konfiguration* (s, v, γ) besteht aus einem Zustand $s \in Z$, einem Eingabewort $v \in I^*$ und einem Kellerwort $\gamma \in C^*$. Für $w \in I^*$ ist (s_0, w, c_0) die *Anfangskonfiguration* von w. Und (s'', λ, γ) wird *Endkonfiguration* genannt, falls $s'' \in F$ (bei beliebigem Kellerwort γ).
Falls $(s', \alpha) \in d(s, x, c)$, so hat die Konfiguration $(s, xv, c\gamma)$ die *Folgekonfiguration* $(s', v, \alpha\gamma)$; falls $(s', \alpha) \in d(s, -, c)$, so hat die Konfiguration $(s, v, c\gamma)$ die *Folgekonfiguration* $(s', v, \alpha\gamma)$.
Ist con' eine Folgekonfiguration von con, so schreibt man dafür auch $con \longmapsto con'$.
Eine Folge von n solchen direkten Übergängen
$$con = con_0 \longmapsto con_1 \longmapsto \cdots \longmapsto con_n = con'$$
(für $n \in \mathbb{N}$) kann durch $con \overset{n}{\longmapsto} con'$ oder $con \overset{*}{\longmapsto} con'$ abgekürzt werden.
4. Ein Wort $w \in I^*$ wird von K *erkannt*, falls die Anfangskonfiguration von w in eine Endkonfiguration überführbar ist, d.h. es existieren $s'' \in F$ und $\gamma \in C^*$ mit $(s_0, w, c_0) \overset{*}{\longmapsto} (s'', \lambda, \gamma)$.
Die Menge aller von K erkannten Wörter bildet die *erkannte Sprache* $L(K)$.

11.2 Deterministische Kellerautomaten

Bei einem Kellerautomaten kann eine Konfiguration mehrere Folgekonfigurationen haben, so dass das Erkennungsverfahren nichtdeterministisch ist. Es wird deterministisch, wenn es jeweils höchstens eine Folgekonfiguration gibt, was offenbar für folgende Kellerautomaten gilt:

Ein Kellerautomat $K = (Z, I, C, d, s_0, F, c_0)$ ist *deterministisch*, wenn für jedes $s \in Z$ und $c \in C$ die Mengen $d(s, x, c)$ für alle $x \in I$ und $d(s, -, c)$ leer oder einelementig sind und $d(s, -, c)$ höchstens dann nicht leer ist, wenn alle $d(s, x, c)$ leer sind.

Ohne Beweis sei angemerkt, dass es nicht möglich ist, zu jedem Kellerautomaten einen deterministischen Kellerautomaten zu konstruieren, der dieselbe Sprache erkennt. So gibt es z.B. einen Kellerautomaten, der die Sprache $L_{mirror} = \{w\, trans(w) \mid w \in \{a, b\}^*\}$ erkennt, aber keinen deterministischen Kellerautomaten.

Ebenfalls ohne Beweis sei festgehalten, dass deterministische Kellerautomaten das Wortproblem ihrer erkannten Sprachen wie endliche Automaten in linearer Zeit lösen, d.h. die Zahl der Berechnungsschritte ist proportional zur Länge der Eingabewörter. Deshalb wird in der Praxis des Compilerbaus in der Regel versucht, die Syntaxanalyse von Programmiersprachen mit Hilfe von deterministischen Kellerautomaten oder ähnlich funktionierenden Verfahren zu bewerkstelligen.

11.3 Beispiel: Reguläre Ausdrücke

Reguläre Ausdrücke über I (vgl. Abschnitt 6.3) werden von dem in Abbildung 8 dargestellten deterministischen Automaten erkannt (wobei $y \in I \cup \{empty, lambda\}$, $c \in \{op, br, c_0\}$ und $\oplus \in \{+, \circ\}$).

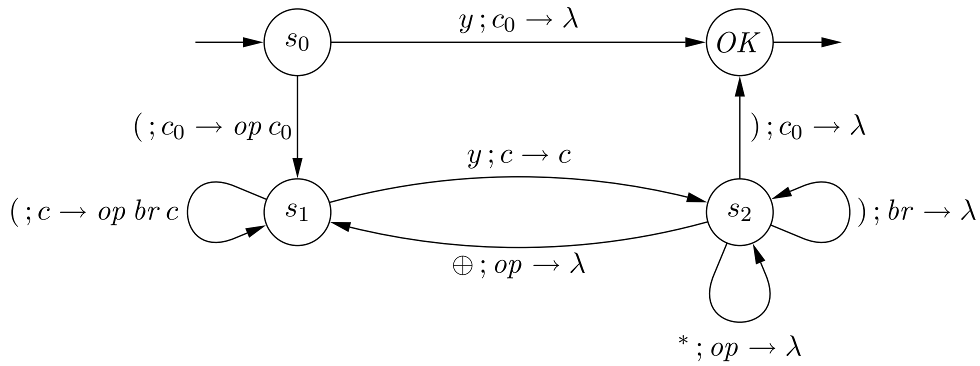

Abbildung 8: Ein deterministischer Kellerautomat, der $REX(I)$ erkennt

Durch folgende Konfigurationsfolge wird der Ausdruck $(((x+empty)^*)\circ lambda)$ als regulär erkannt:

$$
\begin{array}{lllr}
 & (s_0, & (((x+empty)^*)\circ lambda), & c_0) \\
\vdash & (s_1, & ((x+empty)^*)\circ lambda), & op\, c_0) \\
\vdash & (s_1, & (x+empty)^*)\circ lambda), & op\, br\, op\, c_0) \\
\vdash & (s_1, & x+empty)^*)\circ lambda), & op\, br\, op\, br\, op\, c_0) \\
\vdash & (s_2, & +\,empty)^*)\circ lambda), & op\, br\, op\, br\, op\, c_0) \\
\vdash & (s_1, & empty)^*)\circ lambda), & br\, op\, br\, op\, c_0) \\
\vdash & (s_2, &)^*)\circ lambda), & br\, op\, br\, op\, c_0) \\
\vdash & (s_2, & ^*)\circ lambda), & op\, br\, op\, c_0) \\
\vdash & (s_2, &)\circ lambda), & br\, op\, c_0) \\
\vdash & (s_2, & \circ\, lambda), & op\, c_0) \\
\vdash & (s_1, & lambda), & c_0) \\
\vdash & (s_2, &), & c_0) \\
\vdash & (OK, & \lambda, &)
\end{array}
$$

Die folgende Konfigurationsfolge zeigt, dass der Ausdruck $((x+empty)^*)\circ lambda)$ nicht regulär ist:

$$
\begin{array}{lllr}
 & (s_0, & ((x+empty)^*)\circ lambda), & c_0) \\
\vdash & (s_1, & (x+empty)^*)\circ lambda), & op\, c_0) \\
\vdash & (s_1, & x+empty)^*)\circ lambda), & op\, br\, op\, c_0) \\
\vdash & (s_2, & +\,empty)^*)\circ lambda), & op\, br\, op\, c_0) \\
\vdash & (s_1, & empty)^*)\circ lambda), & br\, op\, c_0) \\
\vdash & (s_2, &)^*)\circ lambda), & br\, op\, c_0) \\
\vdash & (s_2, & ^*)\circ lambda), & op\, c_0) \\
\vdash & (s_2, &)\circ lambda), & c_0) \\
\vdash & (OK, & \circ\, lambda), &)
\end{array}
$$

Denn die letzte Konfiguration besitzt keine Folgekonfiguration, ist aber auch keine Endkonfiguration.

12 Von kontextfreien Grammatiken zu Kellerautomaten

Kontextfreie Grammatiken haben nur Regeln, deren linke Seiten einzelne nichtterminale Zeichen sind. Da die Syntax von Programmiersprachen üblicherweise mit Hilfe solcher Regeln definiert wird, sind kontextfreie Grammatiken und die Lösung ihres Wortproblems besonders interessant. Es trifft sich deshalb gut, dass kontextfreie Grammatiken korrekt in Kellerautomaten übersetzt werden können. Dabei bedeutet Korrektheit, dass jede eingegebene Grammatik dieselbe Sprache erzeugt wie der aus der Eingabe konstruierte Automat erkennt. Der konstruierte Automat löst also das Wortproblem der Eingabegrammatik. Da Kellerautomaten im allgemeinen nichtdeterministisch arbeiten, ist diese Lösung jedoch nicht polynomiell (wenn man den Nichtdeterminismus z.B. durch Breitensuche vermeidet). Leider lässt sich im Gegensatz zu endlichen Automaten nicht jeder Kellerautomat in einen deterministischen umbauen, ohne dass sich die erkannte Sprache ändert.

12.1 Der Übersetzer

Zu jeder kontextfreien Grammatik wird ein Kellerautomat konstruiert, in dessen Keller praktisch der Ableitungsprozess der Eingabegrammatik abläuft. Sonstige Zustandsübergänge dienen lediglich dem richtigen Starten und Halten.

Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik und $c_0, c_{OK} \notin N \cup T$. Dann ist der zugehörige Kellerautomat $PDA(G)$ [2] gegeben durch:

$$PDA(G) = (\{s_0, OK\}, T, N \cup T \cup \{c_0, c_{OK}\}, d_G, s_0, \{OK\}, c_0)$$

mit

(i) $d_G(s_0, -, c_0) = \{(s_0, S\, c_{OK})\}$,
(ii) $d_G(s_0, -, A) = \{(s_0, u) \mid A ::= u \in P\}$,
(iii) $d_G(s_0, x, x) = \{(s_0, \lambda)\}$ für alle $x \in T$ und
(iv) $d_G(s_0, -, c_{OK}) = \{(OK, \lambda)\}$.

Theorem 6 (Korrektheit der Übersetzung)
Sei G eine kontextfreie Grammatik und $PDA(G)$ der zugehörige Kellerautomat. Dann gilt: $L(G) = L(PDA(G))$.

Die Situation der Übersetzung von kontextfreien Grammatiken in Kellerautomaten und ihre Korrektheit lassen sich durch das Diagramm in Abbildung 9 veranschaulichen.

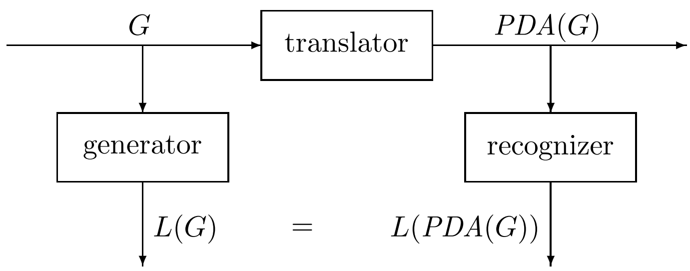

Abbildung 9: Korrekte Übersetzung kontextfreier Grammatiken in Kellerautomaten

Der Beweis von Theorem 6 wird auf ein späteres Kapitel verschoben, da dafür einige Eigenschaften von kontextfreien Grammatiken benötigt werden, die erst in den folgenden Kapiteln erarbeitet werden.

Zunächst soll die Übersetzung mit einem Beispiel veranschaulicht werden.

[2] *PDA* steht für die englische Bezeichnung *pushdown automaton* für Kellerautomat.

12.2 Beispiel: Klammergebirge

Die kontextfreie Grammatik $G = (\{A\}, \{\,[\,,\,]\,\}, \{A ::= AA \mid [A] \mid \lambda\}, A)$ erzeugt alle korrekten Klammerungen über dem Klammerpaar [und] . Der zugehörige Kellerautomat $PDA(G)$ ist in Abbildung 10 dargestellt.

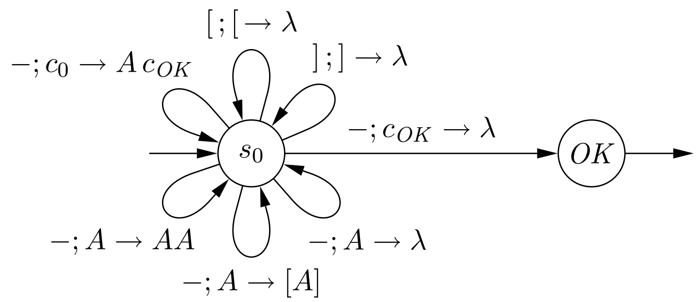

Abbildung 10: Ein zu einer kontextfreien Grammatik gehörender Kellerautomat

Wie die in der linken Hälfte von Abbildung 11 angegebene Berechnung zeigt, wird die Klammerung [ ][ ][[ ]] von $PDA(G)$ erkannt. Dieser Berechnung entspricht die rechts daneben angegebene Ableitung in G, wobei das jeweils im nächsten Schritt ersetzte nichtterminale Zeichen fett gedruckt ist.

Offenbar arbeitet die Zustandsüberführung von $PDA(G)$ wie folgt: Zur Anfangskonfiguration passt nur der Zustandsübergang nach (i), so dass in der Folgekonfiguration das Startsymbol zuoberst auf dem Keller steht. Ein Zustandsübergang nach (ii) kann ausgeführt werden, falls oben auf dem Keller ein Nichtterminal der Grammatik steht. Wenn es mehrere Übergänge für dieses Zeichen gibt, "rät" der Kellerautomat nichtdeterministisch, welche Produktion in der Ableitung angewendet wird, um das nächste Teilstück des Eingabewortes zu erzeugen. Ein Zustandsübergang nach (iii) wird ausgeführt, wenn das oberste Kellersymbol ein Terminalzeichen ist und dieses auch gerade dem nächsten gelesenen Zeichen gleicht. Damit das Eingabewort erkannt wird, muss zuletzt der Zustandsübergang nach (iv) ausgeführt werden.

Insgesamt wird beim Vergleich der Konfigurationsfolge und der Ableitung deutlich, dass der Kellerautomat alle in dem entsprechenden abgeleiteten Wort links von der Lücke stehenden Zeichen bereits gelesen und alle rechts stehenden Zeichen gerade auf dem Keller hat. Weiter fällt auf, dass in jedem Schritt der Ableitung das am weitesten links stehende Nichtterminal ersetzt wird, d.h. die Ableitung ist eine sogenannte *Linksableitung*. Dies ist kein Zufall, sondern liegt daran, dass der Kellerautomat immer nur das oberste Zeichen aus seinem Keller entfernen kann.

Damit liegt die Vermutung nahe, dass der zu einer kontextfreien Grammatik gehörige Kellerautomat genau dann ein Wort erkennt, wenn dieses Wort mit einer Linksableitung der Grammatik erzeugt werden kann. In Theorem 6 wird aber behauptet, dass der Kellerautomat jedes von der Grammatik erzeugte Wort erkennt, unabhängig davon, wie dieses Wort

$$
\begin{array}{llrrcll}
 & (s_0, & [\,][\,][[\,]], & c_0) & & & \\
\vdash & (s_0, & [\,][\,][[\,]], & A\,c_{OK}) & & & \boldsymbol{A} \\
\vdash & (s_0, & [\,][\,][[\,]], & AA\,c_{OK}) & \longrightarrow & & \boldsymbol{A}A \\
\vdash & (s_0, & [\,][\,][[\,]], & AAA\,c_{OK}) & \longrightarrow & & \boldsymbol{A}AA \\
\vdash & (s_0, & [\,][\,][[\,]], & [A]AA\,c_{OK}) & \longrightarrow & & [A]AA \\
\vdash & (s_0, &][\,][[\,]], & A]AA\,c_{OK}) & = & [& \boldsymbol{A}]AA \\
\vdash & (s_0, &][\,][[\,]], &]AA\,c_{OK}) & \longrightarrow & [&]AA \\
\vdash & (s_0, & [\,][[\,]], & AA\,c_{OK}) & = & [\,] & \boldsymbol{A}A \\
\vdash & (s_0, & [\,][[\,]], & [A]A\,c_{OK}) & \longrightarrow & [\,] & [A]A \\
\vdash & (s_0, &][[\,]], & A]A\,c_{OK}) & = & [\,][& \boldsymbol{A}]A \\
\vdash & (s_0, &][[\,]], &]A\,c_{OK}) & \longrightarrow & [\,][&]A \\
\vdash & (s_0, & [[\,]], & A\,c_{OK}) & = & [\,][\,] & \boldsymbol{A} \\
\vdash & (s_0, & [[\,]], & [A]\,c_{OK}) & \longrightarrow & [\,][\,] & [A] \\
\vdash & (s_0, & [\,]], & A]\,c_{OK}) & = & [\,][\,][& \boldsymbol{A}] \\
\vdash & (s_0, & [\,]], & [A]]\,c_{OK}) & \longrightarrow & [\,][\,][& [A]] \\
\vdash & (s_0, &]], & A]]\,c_{OK}) & = & [\,][\,][[& \boldsymbol{A}]] \\
\vdash & (s_0, &]], &]]\,c_{OK}) & \longrightarrow & [\,][\,][[&]] \\
\vdash & (s_0, &], &]\,c_{OK}) & = & [\,][\,][[\,] &] \\
\vdash & (s_0, & \lambda, & c_{OK}) & = & [\,][\,][[\,]] & \\
\vdash & (OK, & \lambda, & \lambda) & & &
\end{array}
$$

Abbildung 11: Eine Berechnung des Kellerautomaten und die zugehörige Ableitung in der Grammatik

abgeleitet wurde. Um dieses Problem zu lösen, wird später gezeigt, dass jede terminierende Ableitung einer kontextfreien Grammatik in eine Linksableitung umgebaut werden kann, ohne das erzeugte Wort zu verändern. Für den Beweis, dass diese Umbautechnik das Gewünschte leistet, wird eine wesentliche Eigenschaft von kontextfreien Grammatiken benötigt, die im nächsten Kapitel als *Kontextfreiheitslemma* formuliert ist.

Ohne das näher auszuführen, sei angemerkt, dass sich auch umgekehrt jeder Kellerautomat K korrekt in eine kontextfreie Grammatik $G(K)$ übersetzen lässt, d.h. $L(K) = L(G(K))$.

13 Kontextfreiheitslemma

Da die linke Seite einer kontextfreien Regel nur aus einem Zeichen besteht, ist die Stelle eines Wortes, auf die diese Regel angewendet wird, in einem sehr strengen Sinn lokalisierbar: Wenn das Wort zerteilt wird, ist das ersetzte Zeichen in genau einem der Teile. Auf Ableitungen übertragen, bedeutet das, dass die einzelnen Ableitungsschritte auf Teile des Ausgangswortes verteilbar sind, wobei eine horizontale Zerlegung von Ableitungen entsteht. Diese als Kontextfreiheitslemma bezeichnete Eigenschaft ist recht offensichtlich, hat aber viele interessante Konsequenzen.

Theorem 7 (Kontextfreiheitslemma)
Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik, $u \xrightarrow[P]{n} v$ eine Ableitung der Länge n und $u = u_1 u_2 \cdots u_k$ eine Zerlegung von u in k Teilwörter.

Dann gibt es k Ableitungen $u_i \xrightarrow[P]{n_i} v_i$, so dass $v = v_1 \cdots v_k$ und $n = \sum_{i=1}^{k} n_i$.

Beweis (mit Induktion über n).

IA: $n = 0$. Dann wähle $v_i = u_i$ und $u_i \xrightarrow[P]{0} u_i$.

IV: Die Behauptung gelte für n.

IS: Betrachte $u \xrightarrow[P]{n+1} v$. Der erste Schritt hat dann die Form $u = u'Au'' \xrightarrow[A::=r]{} u'ru'' = \bar{u}$. Da A ein einzelnes Zeichen ist, existiert ein i_0, so dass $u_{i_0} = u'_{i_0}Au''_{i_0}$ und $u' = u_1 \cdots u_{i_0-1}u'_{i_0}$ sowie $u'' = u''_{i_0}u_{i_0+1} \cdots u_k$. Wähle nun $\bar{u}_i = u_i$ für $i \neq i_0$ und $\bar{u}_{i_0} = u'_{i_0}ru''_{i_0}$, so dass insbesondere $u_{i_0} \xrightarrow[A::=r]{} \bar{u}_{i_0}$ gilt. Außerdem gilt nach Konstruktion:

$$\bar{u} = u'ru'' = u_1 \cdots u_{i_0-1}u'_{i_0}ru''_{i_0}u_{i_0+1} \cdots u_k = \bar{u}_1 \cdots \bar{u}_k.$$

Auf die restlichen n Ableitungsschritte $\bar{u} \xrightarrow[P]{n} v$ ist nun die Induktionsvoraussetzung anwendbar, was Ableitungen $\bar{u}_i \xrightarrow[P]{n_i} v_i$ mit $v = v_1 \cdots v_k$ und $\sum_{i=1}^{k} n_i = n$ ergibt. Für i_0 kann das zu $u_{i_0} \xrightarrow[A::=r]{} \bar{u}_{i_0} \xrightarrow[P]{n_{i_0}} v_{i_0}$, einer Ableitung der Länge $n_{i_0}+1$, zusammengesetzt werden. Für $i \neq i_0$ können die Ableitungen wegen $u_i = \bar{u}_i$ bleiben, wie sie sind. Die Zerlegung von v bleibt unverändert, die Summe der Ableitungslängen ergibt $n + 1$. Damit ist alles gezeigt. □

Es sei noch angemerkt, dass die Umkehrung des Kontextfreiheitslemmas ebenfalls gilt. Ableitungen $u_i \xrightarrow[P]{n_i} v_i$ für $i = 1, \ldots, k$ können immer zu einer Ableitung

$$u = u_1 \cdots u_k \xrightarrow[P]{n} v_1 \cdots v_k = v$$

mit $n = \sum_{i=1}^{k} n_i$ zusammengesetzt werden.

14 Linksableitungen

Wenn man in einer kontextfreien Grammatik beim Ableiten in jedem Schritt das ganz links stehende nichtterminale Zeichen ersetzt, spricht man von Linksableitungen.[3] Trennt

[3] In der Literatur ist es auch gebräuchlich, eine Ableitung Linksableitung zu nennen, wenn alle nichtterminalen Zeichen links vom gerade ersetzten im Rest der Ableitung nicht mehr ersetzt werden. Bei Ableitungen von terminalen Wörtern macht das keinen Unterschied, aber für den Nachweis der Korrektheit von Theorem 6 ist diese Definition weniger geeignet.

man dabei die links stehenden terminalen Zeichen vorher ab, so handelt es sich gerade um Schritte, die der aus einer kontextfreien Grammatik konstruierte Kellerautomat auf dem Keller vollführt. In diesem Kapitel wird gezeigt, dass jede Ableitung einer kontextfreien Grammatik in eine Linksableitung umgebaut werden kann. Es stellt sich also heraus, dass das Ableiten in der Grammatik dieselbe Leistungsfähigkeit hat wie das Bilden von Folgekonfigurationen im zugehörigen Kellerautomaten.

Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik. Eine *Linksableitung* ist eine Ableitung $u_1 \xrightarrow[P]{} u_2 \xrightarrow[P]{} \cdots \xrightarrow[P]{} u_n$, bei der in jedem Schritt $u_i \xrightarrow[A_i ::= v_i]{} u_{i+1}$ $(1 \leq i < n)$ das am weitesten links stehende Nichtterminal ersetzt wird, d.h. $u_i = x_i A_i y_i$ und $u_{i+1} = x_i v_i y_i$ mit $x_i \in T^*$. Um anzudeuten, dass eine Ableitung eine Linksableitung ist, wird der Pfeil $\overset{\ell}{\longrightarrow}$ statt $\longrightarrow$ verwendet.

Das folgende Lemma impliziert, dass man sich bei kontextfreien Grammatiken auf die Betrachtung von Linksableitungen beschränken kann, ohne dass dies eine Auswirkung auf die erzeugte Sprache hat.

Lemma 8
Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik. Dann lässt sich jede Ableitung $A \xrightarrow[P]{*} v$ mit $A \in N$, $v \in T^*$ in eine Linksableitung $A \overset{*}{\underset{P}{-\ell\!\!\to}} v$ umformen.

Beweis (Per Induktion über die Länge von Ableitungen).

IA: Eine Ableitung $A \xrightarrow[P]{0} v$ mit $v \in T^*$ existiert nicht, also muss nichts gezeigt werden.

IV: Gelte die Behauptung für alle Ableitungen der Länge $\leq n$.

IS: Betrachte eine Ableitung $A \xrightarrow[P]{} u \xrightarrow[P]{n} v$. Dann lässt u sich in $u = u_0 A_1 u_1 \cdots A_m u_m$ mit $u_0, \ldots, u_m \in T^*$ und $A_1, \ldots, A_m \in N$ zerlegen. Für $i = 0, \ldots, m$ gilt $u_i \xrightarrow[P]{0} u_i$ und aufgrund des Kontextfreiheitslemmas auch $A_i \xrightarrow[P]{n_i} v_i$ für $i = 1, \ldots, m$, wobei $v = u_0 v_1 u_1 \cdots v_m u_m$ und $n_i \leq \sum_{i=1}^{m} n_i = n$. Also kann nach Induktionsvoraussetzung jede der Ableitungen $A_i \xrightarrow[P]{n_i} v_i$ in eine Linksableitung $A_i \overset{*}{\underset{P}{-\ell\!\!\to}} v_i$ umgeformt werden. In der richtigen Reihenfolge zusammengesetzt, erhält man

$$\begin{aligned}
A &\underset{P}{-\ell\!\!\to} u_0 A_1 u_1 \cdots A_m u_m \\
&\overset{*}{\underset{P}{-\ell\!\!\to}} u_0 v_1 u_1 A_2 u_2 \cdots A_m u_m \\
&\overset{*}{\underset{P}{-\ell\!\!\to}} u_0 v_1 u_1 v_2 u_2 A_3 u_3 \cdots A_m u_m \\
&\ \vdots \\
&\overset{*}{\underset{P}{-\ell\!\!\to}} u_0 v_1 u_1 \cdots v_m u_m\,,
\end{aligned}$$

also insgesamt eine Linksableitung $A \overset{*}{\underset{P}{-\ell\!\!\to}} v$. □

15 Korrektheit der Übersetzung von kontextfreien Grammatiken in Kellerautomaten

In diesem Kapitel wird der Beweis für Theorem 6 nachgeliefert. Dazu wird zunächst der enge Zusammenhang zwischen dem Ableiten in einer kontextfreien Grammatik und dem Bilden von Folgekonfigurationen in dem zugehörigen Kellerautomat festgehalten.

Sei in diesem Kapitel $G = (N, T, P, S)$ eine gegebene kontextfreie Grammatik und $PDA(G)$, der zu ihr gehörige Kellerautomat, wie in Abschnitt 12.1 definiert.

Lemma 9
Sei $w \in (N \cup T)^*$. Wenn es eine Linksableitung $S \xrightarrow[\ell]{*} w = u\bar{u}$ gibt, wobei $u \in T^*$ das längste terminale Anfangsstück von w ist, dann gibt es für beliebiges $v \in T^*$ eine Konfigurationsfolge $(s_0, uv, S\, c_{OK}) \vdash^{*} (s_0, v, \bar{u}\, c_{OK})$.

Beweis (mittels Induktion über die Ableitungslänge)**.**

IA: Für $S \xrightarrow[\ell]{0} w$ gilt $w = S$, d.h. $u = \lambda$ und $\bar{u} = S$, und daher

$$(s_0, uv, S\, c_{OK}) = (s_0, v, \bar{u}\, c_{OK}) \vdash^{0} (s_0, v, \bar{u}\, c_{OK}).$$

IV: Die Behauptung gelte für alle Linksableitungen der Länge n.

IS: Betrachte nun eine Linksableitung $S \xrightarrow[\ell]{n} u_1 A \bar{u}_1 \xrightarrow[\ell]{} u_1 r \bar{u}_1 = w$ der Länge $n + 1$ und eine Zerlegung $w = u\bar{u}$, wobei u das längste terminale Anfangsstück von w ist. Da die Ableitung eine Linksableitung ist, gilt $u_1 \in T^*$. Also ist u_1 ein terminales Anfangsstück von w und damit auch Anfangsstück von u, d.h. es gibt ein $t \in T^*$ mit $u = u_1 t$ und $t\overline{u} = r\overline{u}_1$. Somit kann man die Konfigurationsfolge

$$\begin{array}{rcl} (s_0, uv, S\, c_{OK}) & = & (s_0, u_1 tv, S\, c_{OK}) \\ & \vdash^{*} & (s_0, tv, A\overline{u}_1\, c_{OK}) \\ & \vdash & (s_0, tv, r\overline{u}_1\, c_{OK}) \\ & = & (s_0, tv, t\overline{u}\, c_{OK}) \\ & \vdash^{*} & (s_0, v, \overline{u}\, c_{OK}) \end{array}$$

konstruieren, wobei sich die zweite Zeile aus der Anwendung der Induktionsvoraussetzung für die Linksableitung $S \xrightarrow[\ell]{*} u_1 A \bar{u}_1$, die dritte Zeile aus Zeile (ii) der Definition von d_G und die letzte Zeile aus Zeile (iii) der Definition von d_G ergibt.

□

Lemma 10
Seien $u, v \in T^*$. Wenn es eine Konfigurationsfolge $(s_0, uv, S\, c_{OK}) \vdash^{*} (s_0, v, \bar{u}\, c_{OK})$ gibt, dann auch eine Linksableitung $S \xrightarrow[\ell]{*} u\bar{u}$.

Beweis (mittels Induktion über die Länge der Konfigurationsfolge)**.**

IA: Für $(s_0, uv, S\,c_{OK}) \overset{0}{\longmapsto} (s_0, v, \bar{u}\,c_{OK})$ gilt $u = \lambda$ und $\bar{u} = S$, woraus $S \overset{0}{-\ell\!\!\longrightarrow} S = u\bar{u}$ folgt.

IV: Gelte die Behauptung für Konfigurationsfolgen der Länge n.

IS: Betrachte $(s_0, uv, S\,c_{OK}) \overset{n+1}{\longmapsto} (s_0, v, \bar{u}\,c_{OK})$. Da die Zustandsüberführung nie c_0 auf den Keller schreibt, kann keine der Folgekonfigurationen nach Zeile (i) der Definition von d_G gebildet worden sein. Damit scheidet auch Zeile (iv) aus, denn sonst würde c_{OK} nicht mehr auf den Keller stehen. Also ist die letzte Folgekonfiguration nach Zeile (ii) oder (iii) der Definition von d_G gebildet worden.

1. Entsprechend Zeile (ii) hat die Konfigurationsfolge die Form

$$(s_0, uv, S\,c_{OK}) \overset{n}{\longmapsto} (s_0, v, A\bar{u}_1\,c_{OK}) \longmapsto (s_0, v, \bar{u}_0\bar{u}_1\,c_{OK}) = (s_0, v, \bar{u}\,c_{OK})$$

   und $A ::= \bar{u}_0$ ist eine Regel in P. Zusammen mit der Induktionsvoraussetzung für die Konfigurationsfolge $(s_0, uv, S\,c_{OK}) \overset{n}{\longmapsto} (s_0, v, A\bar{u}_1\,c_{OK})$ ergibt sich dann die Linksableitung

$$S \overset{*}{-\ell\!\!\longrightarrow} uA\bar{u}_1 -\ell\!\!\longrightarrow u\bar{u}_0\bar{u}_1 = u\bar{u}.$$

2. Entsprechend Zeile (iii) hat die Konfigurationsfolge die Form

$$(s_0, uv, S\,c_{OK}) \overset{n}{\longmapsto} (s_0, xv, x\bar{u}\,c_{OK}) \longmapsto (s_0, v, \bar{u}\,c_{OK}).$$

   Dann hat also u die Form $u = u_0x$, und aus der Induktionsvoraussetzung für die Konfigurationsfolge $(s_0, u_0xv, S\,c_{OK}) \overset{n}{\longmapsto} (s_0, xv, x\bar{u}\,c_{OK})$ ergibt sich die Linksableitung $S \overset{*}{-\ell\!\!\longrightarrow} u_0x\bar{u} = u\bar{u}$. □

Mit Hilfe der beiden Lemmata kann nun folgendermaßen geschlossen werden.

Beweis von Theorem 6**.**
Sei $w \in L(G)$, d.h. $S \overset{*}{\longrightarrow} w$ und $w \in T^*$. Damit existiert nach Lemma 8 eine Linksableitung $S \overset{*}{-\ell\!\!\longrightarrow} w$. Da w schon das längste terminale Anfangsstück von w ist, folgt mit Lemma 9, dass es eine Konfigurationsfolge $(s_0, w, S\,c_{OK}) \overset{*}{\longmapsto} (s_0, \lambda, \lambda\,c_{OK})$ gibt, wobei $\bar{u} = \lambda$ sein muss und $v = \lambda$ gewählt wurde. Mit den Zeilen (i) und (iv) der Definition von d_G lässt sich diese Folgekonfigurationsbildung vorn und hinten verlängern zu:

$$(s_0, w, c_0) \longmapsto (s_0, w, S\,c_{OK}) \overset{*}{\longmapsto} (s_0, \lambda, c_{OK}) \longmapsto (OK, \lambda, \lambda). \tag{$*$}$$

Das bedeutet aber gerade $w \in L(PDA(G))$.

Sei umgekehrt $w \in L(PDA(G))$, d.h. $(s_0, w, c_0) \overset{*}{\longmapsto} (OK, \lambda, \gamma)$ für irgendein γ. Das lässt sich immer in die Form $(*)$ zerlegen, denn der erste und letzte Schritt können nicht anders aussehen. Der mittlere Teil impliziert nach Lemma 10 $S \overset{*}{\longrightarrow} w\lambda = w$. Somit gilt $w \in L(G)$.

Insgesamt sind die beiden Sprachen also gleich. □