7. Kellerautomaten

7.1 Kellerautomaten und akzeptierte Sprachen

Endliche Automaten haben, abgesehen von den Zuständen, kein "Gedächtnis" und können sich nicht "merken", wie sie in einen bestimmten Zustand gekommen sind.

Kellerautomaten (PDA: Push down automaton) verfügen zusätzlich über einen Speicher, der in Form eines Kellers organisiert ist: gelesen wird jeweils nur das oberste Zeichen, und bei einem Zustandsübergang wird dieses ersetzt und möglicherweise mehrere neue Zeichen im Keller gespeichert.

Ein Hinweis zu verschiedenen Definitionen in der Literatur:
man kann die von einem PDA akzeptierten Wörter wie bisher durch akzeptierende Zustände, aber auch durch leeren Keller definieren. Im ersteren Fall (etwa Hopcroft, Motwani, Ullman) sind PDAs 7-Tupel, im zweiten Fall (Schöning) 6-Tupel. Die Ansätze sind äquivalent, hier wollen wir (zunächst) den zweiten verwenden.

Def.: Ein (nichtdeterministischer) Kellerautomat ist ein Tupel

$$M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$$

hierbei ist:

- *Z endliche Zustandsmenge*
- *Σ Eingabealphabet*
- *Γ Kelleralphabet*
- *δ: $Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Pot_f(Z \times \Gamma^*)$ die Überführungsfunktion ($Pot_f(X)$ ist die Menge aller endlichen Teilmengen von X)*
- *z_0 Startzustand*
- *$\# \in \Gamma$ unterstes Kellerzeichen*

Intuitiv heißt $(z', B_1...B_k) \in \delta(z,a,A)$:

wenn M in Zustand z Eingabe a liest und A oberstes Kellerzeichen ist, so kann M in z' übergehen, wobei A überschrieben wird mit $B_1...B_k$ (B_1 oberstes neues Kellerzeichen).

Die üblichen PUSH und POP-Operationen bei Stacks lassen sich simulieren
(PUSH(B) hier durch Wahl von $B_1...B_k = BA$, POP durch $B_1...B_k = \varepsilon$)

Es kann auch spontane ε-Übergänge geben.

Es gibt keine Endzustände! w wird akzeptiert, wenn nach Abarbeitung von w Keller leer ist!

Notation:
Wir schreiben $zaA \rightarrow z'B_1..B_k$, falls $(z', B_1..B_k) \in \delta(z,a,A)$ ($a \in \Sigma$ oder $a = \varepsilon$).
Umgekehrt können wir δ vollständig durch Angabe einer Menge von Regeln R spezifizieren, indem wir festlegen: $\delta(z,a,A) = \{(z', B_1..B_k) \mid zaA \rightarrow z'B_1..B_k \in R\}$.

Def.: Eine Konfiguration eines Kellerautomaten ist ein Element (z,w_1,w_2) aus $Z \times \Sigma^* \times \Gamma^*$. Hierbei ist z der aktuelle Zustand, w_1 das noch zu verarbeitende Restwort, w_2 der Kellerinhalt.

Die Konfigurationsübergangsrelation ist die kleinste Relation $\vdash$ so dass:

$(z, a_1...a_n, A_1...A_m) \vdash (z', a_2...a_n, B_1..B_kA_2...A_m)$ falls $za_1A_1 \to z'B_1..B_k$
$(z, a_1...a_n, A_1...A_m) \vdash (z', a_1...a_n, B_1..B_kA_2...A_m)$ falls $z\varepsilon A_1 \to z'B_1..B_k$

Sei $\vdash^*$ die reflexive und transitive Hülle von $\vdash$. Die durch einen Kellerautomaten

$$M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$$

(durch leeren Speicher) akzeptierte Sprache ist:

$$L(M) = \{x \in \Sigma^* \mid (z_0, x, \#) \vdash^* (z, \varepsilon, \varepsilon)\}$$

Beispiel: $M = (\{z_0,z_1\}, \{a,b,\$\}, \{A,B,\#\}, \delta, z_0, \#)$ mit:

$z_0a\# \to z_0A\#$	$z_0aA \to z_0AA$	$z_0aB \to z_0AB$
$z_0b\# \to z_0B\#$	$z_0bA \to z_0BA$	$z_0bB \to z_0BB$
$z_0\$\# \to z_1\#$	$z_0\$A \to z_1A$	$z_0\$B \to z_1B$
$z_1aA \to z_1\varepsilon$	$z_1bB \to z_1\varepsilon$	$z_1\varepsilon\# \to z_1\varepsilon$

M akzeptiert z.B. ab$ba, denn:

$(z_0, ab\$ba, \#) \vdash (z_0, b\$ba, A\#) \vdash (z_0, \$ba, BA\#) \vdash (z_1, ba, BA\#)$
$\vdash (z_1, a, A\#) \vdash (z_1, \varepsilon, \#) \vdash (z_1, \varepsilon, \varepsilon)$

M akzeptiert nicht ab$ab, denn:

$(z_0, ab\$ab, \#) \vdash (z_0, b\$ab, A\#) \vdash (z_0, \$ab, BA\#) \vdash (z_1, ab, BA\#)$ hier geht es nicht weiter!

akzeptierte Sprache: $\{c_1...c_n\$c_n...c_1 \mid c_i \in \{a,b\}\}$

M ist deterministisch: es gibt jeweils höchstens 1 Nachfolgekonfiguration. Im allgemeinen Fall kommt man mit deterministischen nicht hin, z.B. bei der Sprache

$$\{c_1...c_nc_n...c_1 \mid c_i \in \{a,b\}\}$$

Da das $ in der Mitte fehlt, muss man jetzt "raten", ob man in der Mitte ist. Das lässt sich nur durch Nichtdeterminismus erreichen, hier etwa durch Ersetzen von

	$z_0\$\# \to z_1\#$	$z_0\$A \to z_1A$	$z_0\$B \to z_1B$
durch:	$z_0\varepsilon\# \to z_1\varepsilon$	$z_0aA \to z_1\varepsilon$	$z_0bB \to z_1\varepsilon$

Herleitung von baab:

$(z_0, baab, \#) \vdash (z_0,aab, B\#) \vdash (z_0, ab, AB\#) \vdash (z_1, b, B\#)$
$\vdash (z_1, \varepsilon, \#) \vdash (z_1, \varepsilon, \varepsilon)$

Hier wurde die richtige Auswahl getroffen (d.h. die Mitte des Wortes richtig geraten). Man hätte als zweiten Übergang auch wählen können: $(z_0, bbb, B\#) \vdash (z_1, bb, \#)$. Man wäre dann aber nicht zur Zielkonfiguration gekommen.

Im Allgemeinen bilden die möglichen von der Startkonfiguration erreichbaren Konfigurationen einen Baum.

7.2 Akzeptanzbedingungen

Wir haben bisher Wörter durch leeren Keller akzeptiert. Es ist auch möglich, durch akzeptierende Zustände zu akzeptieren.

Ein Kellerautomat mit akzeptierenden Zuständen ist ein Tupel $M = (Z, \Sigma, \Gamma, \delta, z_0, \#, F)$, wobei $F \subseteq Z$ die akzeptierenden Zustände sind, alle anderen Komponenten wie vorher.

Die von M durch Endzustände akzeptierte Sprache ist

$$L_F(M) = \{w \mid (z_0, w, \#) \vdash^* (z, \varepsilon, W), z \in F, W \in \Gamma^*\}$$

Im allgemeinen Fall macht es keinen Unterschied, ob man die von einem PDA akzeptierte Sprache über den leeren Keller oder über Endzustände definiert. Es gelten folgende Sätze:

Satz: Zu jedem Kellerautomaten mit akzeptierenden Zuständen $K = (Z, \Sigma, \Gamma, \delta, z_0, \#, F)$ gibt es einen Kellerautomaten K' so dass $L_F(K) = L(K')$.

Beweisskizze: Sei K per Endzustand akzeptierender PDA. Wir erzeugen aus K einen per leeren Keller akzeptierenden Automaten. Um zu verhindern, dass im neuen Automaten Eingaben akzeptiert werden, für die K ohne zu akzeptieren in eine Konfiguration mit leerem Keller kommt, muss zuerst ein neues unterstes Stacksymbol $\perp$ eingeführt werden: ein neuer Anfangszustand z_0' wird nur dazu verwendet, das neue Symbol unter das alte Symbol # zu schreiben und in den ursprünglichen Anfangszustand z_0 zu gehen: $z_0'\varepsilon\# \rightarrow z_0\#\perp$. Des Weiteren wird ein neuer Zustand q_f benötigt, in den man aus jedem Endzustand per ε-Übergang kommen kann. In diesem Zustand wird nur noch der Keller geleert, was durch Hinzufügen von Regeln der Form $q_f\varepsilon B \rightarrow q_f\varepsilon$ für alle Kellersymbole B erreicht wird.

Formal: $K' = (Z \cup \{z_0', q_f\}, \Sigma, \Gamma \cup \{\perp\}, \delta', z_0', \#)$

mit

$\delta'(z_0', \varepsilon, \#) = \{(z_0, \#\perp)\}$		
$\delta'(q_f, \varepsilon, A) = \{(q_f, \varepsilon)\}$	für alle A in $\Gamma \cup \{\perp\}$
$\delta'(q, a, A) = \delta(q, a, A)$	für alle Zustände q in $Z \setminus F$, a in $\Sigma \cup \{\varepsilon\}$, A in Γ
$\delta'(p, a, A) = \delta(p, a, A)$	für alle Zustände p in F, a in Σ, A in Γ
$\delta'(p, \varepsilon, A) = \delta(p, \varepsilon, A) \cup \{(q_f, \varepsilon)\}$	für alle Zustände p in F, A in Γ

und $\delta'(x, y, z) = \emptyset$ in allen anderen Fällen.

Dieser Automat modelliert den ursprünglichen wie oben beschrieben: erst wird ein neues unterstes Stacksymbol eingefügt, dann die Berechnung von K modelliert. In jedem Endzustand kann nichtdeterministisch in einen Zustand gesprungen werden, von dem aus nur noch der Keller enleert wird.

Satz: Zu jedem Kellerautomaten $K = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ gibt es einen Kellerautomaten K' mit akzeptierenden Zuständen, so dass $L(K) = L_F(K')$.

Beweisskizze:

Auch hier muss man zunächst wie oben beschrieben ein neues unterstes Stacksymbol $\perp$ unter # "schieben". Dieses Symbol dient dazu, Situationen zu erkennen, in denen im ursprünglichen Automaten K der Keller leer ist.

Durch Hinzufügen eines neuen, akzeptierenden Zustandes q_f und Einfügen von Regeln der Form $z\varepsilon\perp \rightarrow q_f\varepsilon$ (für alle $z \in Z$) entsteht der äquivalente per Endzustand akzeptierende Automat:

$$K' = (Z \cup \{ z_0', q_f\}, \Sigma, \Gamma \cup \{\perp\}, \delta', z_0', \#, \{q_f\})$$

mit $\delta'(z_0', \varepsilon, \#) = \{(z_0, \#\perp)\}$

$\delta'(q, a, A) = \delta(q, a, A)$ für alle Zustände q in Z, a in $\Sigma \cup \{\varepsilon\}$, A in Γ

$\delta'(q, \varepsilon, \perp) = \{(q_f, \varepsilon)\}$ für alle Zustände q in Z

und $\delta'(x, y, z) = \varnothing$ in allen anderen Fällen.

7.3 Kellerautomaten und kontextfreie Sprachen

Satz: Eine Sprache L ist kontextfrei genau dann, wenn L von einem PDA erkannt wird.

Beweis: (kontextfrei => wird von PDA erkannt)

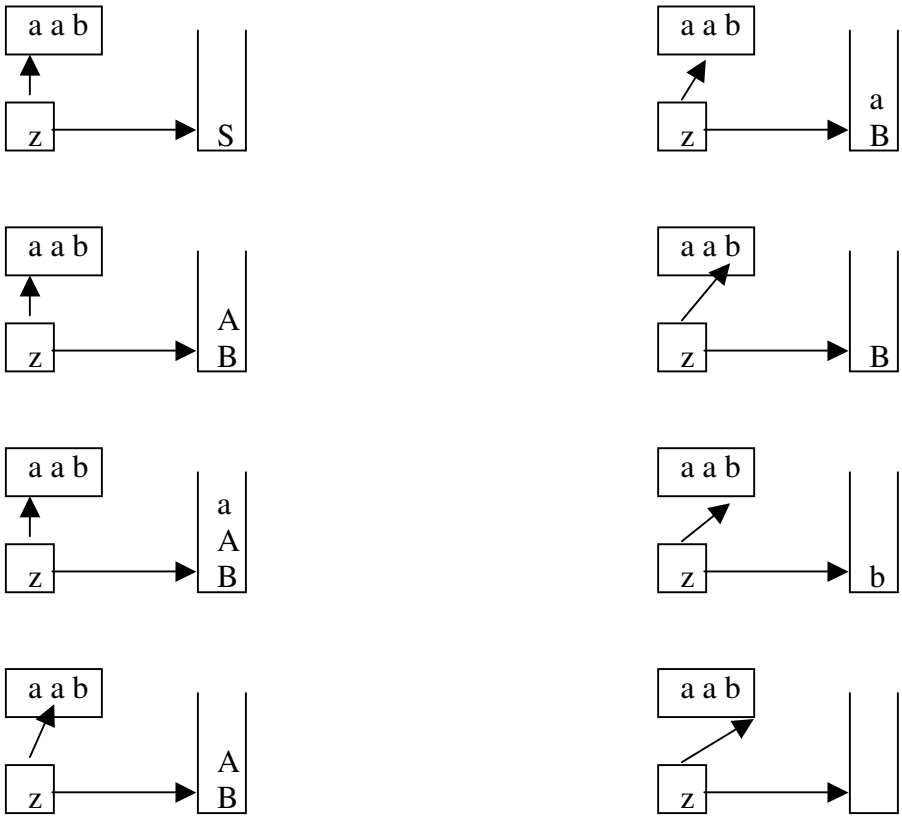
Zur Motivation: Betrachte folgende Grammatik:

$S \rightarrow AB$
$A \rightarrow a \mid aA$
$B \rightarrow b \mid bB$

Linksableitung von aab:

S => AB => aAB => aaB => aab

Modellierung mit Kellerautomat:



Es entspricht jeweils der bereits abgearbeitete Teil der Eingabe gefolgt vom Kellerinhalt dem in der Linksableitung aktuell abgeleiteten Wort.

Um das hinzukriegen brauchen wir einen ε-Übergang für jede Regel der Grammatik:

$z\varepsilon S \rightarrow zAB$
$z\varepsilon A \rightarrow zaA$
$z\varepsilon A \rightarrow za$
$z\varepsilon B \rightarrow zbB$
$z\varepsilon B \rightarrow zb$

Außerdem brauchen wir Übergänge, die uns erlauben, in der Eingabe vorzurücken, wenn das passende Terminalsymbol oben im Stack steht:

$zaa \rightarrow z\varepsilon$
$zbb \rightarrow z\varepsilon$

Hier der eigentliche Beweis (Sketch):

Sei G = (V, Σ, P, S) kontextfreie Grammatik für L. Wir definieren einen Kellerautomaten M, so dass durch M Ableitungen von G simuliert werden:

$$M = (\{z\}, \Sigma, V \cup \Sigma, \delta, z, S)$$

wobei δ durch die folgende Regelmenge spezifiziert wird:

für jede Regel $A \to \alpha \in P$, $\alpha \in (V \cup \Sigma)^*$:	$z\varepsilon A \to z\alpha$
für alle $a \in \Sigma$:	$zaa \to z\varepsilon$

Es gilt nun für alle $x \in \Sigma^*$:

$x \in L(G)$
es gibt Linksableitung in G der Form S => ... => x
es gibt Konfigurationsübergänge (z,x,S) |-- ... |-- (z,ε,ε)
$x \in L(M)$

(informelle) Begründung: betrachte Linksableitung in G. Für jeden Ableitungsschritt gibt es passende Konfigurationsübergänge in M, so dass das in der Ableitung erzeugte Wort jeweils dem bereits abgearbeiteten Teil der Eingabe x zusammen mit dem Stackinhalt von M entspricht. Wird ein Wort abgeleitet, das nur noch aus Terminalsymbolen besteht, so führen weitere Konfigurationsübergänge zu leerem Keller und das Wort wird vom Automaten akzeptiert.
Umgekehrt können vom Automaten nur solche Konfigurationsübergänge vorgenommen werden, die entweder einem weiteren Schritt der Linksableitung entsprechen, oder dem Vorrücken im bisher erzeugten Anfangsstück des Wortes, das aus Terminalsymbolen besteht. In jedem Fall werden nur Wörter akzeptiert, die auch eine Linksableitung besitzen. qed.

Weiteres Beispiel:

G: $S \to aSb$
$S \to aCb$
$S \to ab$
$C \to cC$
$C \to c$

M: $zaa \to z\varepsilon$
$zbb \to z\varepsilon$
$zcc \to z\varepsilon$
$z\varepsilon S \to zaSb$
$z\varepsilon S \to zaCb$
$z\varepsilon S \to zab$
$z\varepsilon C \to zcC$
$z\varepsilon C \to zc$

(Links-)Ableitung von aacbb: S => aSb => aaCbb => aacbb

entsprechende Konfigurationsübergänge:

(z, aacbb, S) |-- (z, aacbb, aSb) |-- (z, acbb, Sb) |-- (z, acbb, aCbb) |-- (z, cbb, Cbb)
|-- (z, cbb, cbb) |-- (z, bb, bb) |-- (z, b, b) |-- (z,ε,ε)

Länge der Ableitung in M = Länge der G-Ableitung plus Länge von x.

Bevor wir den Beweis fortsetzen, noch folgende Vorbemerkung:

Variablen einer kontextfreien Grammatik heißen nutzlos, wenn aus ihnen entweder kein Terminalwort hergeleitet werden kann, oder wenn sie selbst vom Startsymbol S aus nicht hergeleitet werden können. Nutzlose Variablen können mitsamt allen Regeln, in denen sie auf der linken oder rechten Seite vorkommen, aus einer Grammatik eliminiert werden. Die von der Grammatik erzeugte Sprache bleibt unverändert.

Beispiel: $S \rightarrow C \mid SB \mid BE$
$B \rightarrow CbD$
$D \rightarrow dD$
$C \rightarrow cC \mid c$
$E \rightarrow e$

Wir betrachten zunächst die Menge der Variablen, aus denen Terminalwörter hergeleitet werden können. Sie lässt sich induktiv definieren als kleinste Menge T, für die gilt: falls es eine Regel $V \rightarrow \beta$ gibt, so dass β nur aus Terminalsymbolen und Symbolen aus T besteht, so gehört V zu T. In unserem Beispiel ist das die Menge {E, C, S}. B und D sind nutzlos, wir können die Grammatik also reduzieren zu

$S \rightarrow C$
$C \rightarrow cC \mid c$
$E \rightarrow e$

Offensichtlich ist jetzt E nicht mehr aus S erzeugbar, und wir können auch $E \rightarrow e$ streichen.

Beweis (kontextfrei <= von PDA erkannt):
Sei L die von

$$M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$$

akzeptierte Sprache. Wir nehmen an, dass für jedes $zaA \rightarrow z'B_1...B_k$ gilt: $k \leq 2$. Diese Annahme ist keine Einschränkung, denn falls $k > 2$ können wir k-2 neue Zustände $z_1,...,z_{k-2}$ einführen und $zaA \rightarrow z'B_1...B_k$ ersetzen durch :

$zaA \rightarrow z_1B_{k-1}B_k$
$z_1\varepsilon B_{k-1} \rightarrow z_2B_{k-2}B_{k-1}$
...
$z_{k-2}\varepsilon B_2 \rightarrow z'B_1B_2$

Wir konstruieren Grammatik G, die Rechenschritte von M durch Linksableitungen simuliert:

$G = (V, \Sigma, P, S)$ mit
$V = \{S\} \cup Z \times \Gamma \times Z$

Grundidee: Regeln für Variable (z, A, z') so definiert, dass man alle Wörter herleiten kann, deren Eingabe im Automaten im Zustand z bei oberstem Kellerelement A dazu führen, dass das von A belegte Feld im Stack wieder freigemacht wird, wobei nach dem Freimachen in den Zustand z' übergegangen wird.

$P = \{S \to (z_0, \#, z) \mid z \in Z\} \cup$
$\{(z,A,z') \to a \mid (z', \varepsilon) \in \delta(z, a, A)\} \cup$
$\{(z,A,z') \to a(z_1, B, z') \mid (z_1, B) \in \delta(z, a, A), z' \in Z\} \cup$
$\{(z,A,z') \to a(z_1, B, z_2)\ (z_2, C, z') \mid (z_1, BC) \in \delta(z, a, A), z', z_2 \in Z\}$

Erläuterung zu den Regeln in P:

1. Teilmenge: aus S muss alles herleitbar sein, was man in M vom Startzustand aus durch Entfernen des untersten Stackelementes # herleiten kann, wobei in beliebigen Zustand übergegangen werden darf.
2. Teilmenge: wenn M bei Lesen von a in z' übergeht und dabei A aus dem Stack löscht, so gehört a zu den Wörtern, deren Eingabe im Zustand z mit oberstem Stacksymbol A zu Freigeben der Position von A und Übergang in z' führt.
3. Teilmenge: wenn M bei Lesen von a in z_1 übergeht und dabei A im Stack durch B ersetzt, so gehört a gefolgt von jedem aus (z_1, B, z') herleitbaren Wort (d.h. jedem Wort, dessen Eingabe in z_1 zu Leeren der Position von B und Übergang in z' führt), zu den Wörtern, deren Eingabe im Zustand z mit oberstem Stacksymbol A zu Freigeben der Position von A und Übergang in z' führt.
4. Teilmenge: wenn M bei Lesen von a in z_1 übergeht und dabei A im Stack durch BC ersetzt, so gehört a gefolgt von jedem aus (z_1, B, z_2) herleitbaren Wort gefolgt von jedem aus (z_2, C, z') herleitbaren Wort (hier ist z_2 ein beliebiger Zustand) zu den Wörtern, deren Eingabe im Zustand z mit oberstem Stacksymbol A zu Freigeben der Position von A und Übergang in z' führt.

Es lässt sich nun zeigen:

$$(z, A, z') \Rightarrow^* x \text{ genau dann wenn } (z, x, A) \vdash^* (z', \varepsilon, \varepsilon)$$

Beweis durch Induktion über Anzahl der Rechenschritte von M (<=) bzw. die Länge der Linksableitung von x (=>).
(hier nicht: Interessenten seien auf Schöning verwiesen)
Mit dieser Aussage ergibt sich:

$x \in L(M) \quad <=> \quad (z_0, x, \#) \vdash^* (z, \varepsilon, \varepsilon)$ für ein $z \in Z$
$<=> \quad S \Rightarrow (z_0, \#, z) \Rightarrow^* x$ für ein $z \in Z$
$<=> \quad x \in L(G)$

Beispiel: Automat für $L = a^n\$a^n$

$M = (\{z_0,z_1\}, \{a, \$\}, \{A, \#\}, \delta, z_0, \#)$ mit:

$z_0a\# \to z_0A\#$	$z_0aA \to z_0AA$
$z_0\$\# \to z_1\#$	$z_0\$A \to z_1A$
$z_1aA \to z_1\varepsilon$	$z_1\varepsilon\# \to z_1\varepsilon$

Wir konstruieren Grammatik G mit

$V = \{S,(z_0, A, z_0), (z_0, A, z_1), (z_1, A, z_0), (z_1, A, z_1), (z_0, \#, z_0), (z_0, \#, z_1), (z_1, \#, z_0), (z_1, \#, z_1)\}$

P besteht aus folgenden Regeln:

$S \rightarrow (z_0, \#, z_0)$
$S \rightarrow (z_0, \#, z_1)$

$(z_1, A, z_1) \rightarrow a$	wegen $z_1aA \rightarrow z_1\varepsilon$
$(z_1, \#, z_1) \rightarrow \varepsilon$	wegen $z_1\varepsilon\# \rightarrow z_1\varepsilon$
$(z_0, \#, z_0) \rightarrow \$(z_1, \#, z_0)$	wegen $z_0\$\# \rightarrow z_1\#$
$(z_0, \#, z_1) \rightarrow \$(z_1, \#, z_1)$	
$(z_0, A, z_0) \rightarrow \$(z_1, A, z_0)$	wegen $z_0\$A \rightarrow z_1A$
$(z_0, A, z_1) \rightarrow \$(z_1, A, z_1)$	
$(z_0, \#, z_0) \rightarrow a(z_0, A, z_0)\ (z_0, \#, z_0)$	wegen $z_0a\# \rightarrow z_0A\#$
$(z_0, \#, z_0) \rightarrow a(z_0, A, z_1)\ (z_1, \#, z_0)$	
$(z_0, \#, z_1) \rightarrow a(z_0, A, z_0)\ (z_0, \#, z_1)$	
$(z_0, \#, z_1) \rightarrow a(z_0, A, z_1)\ (z_1, \#, z_1)$	
$(z_0, A, z_0) \rightarrow a(z_0, A, z_0)\ (z_0, A, z_0)$	wegen $z_0aA \rightarrow z_0AA$
$(z_0, A, z_0) \rightarrow a(z_0, A, z_1)\ (z_1, A, z_0)$	
$(z_0, A, z_1) \rightarrow a(z_0, A, z_0)\ (z_0, A, z_1)$	
$(z_0, A, z_1) \rightarrow a(z_0, A, z_1)\ (z_1, A, z_1)$	

Wir eliminieren zunächst nutzlose Variablen und ihre Regeln. Nicht nutzlos sind:
$(z_1, A, z_1), (z_1, \#, z_1), (z_0, \#, z_1), (z_0, A, z_1)$, S. Wir können deshalb P wie folgt reduzieren:

$S \rightarrow (z_0, \#, z_1)$
$(z_1, A, z_1) \rightarrow a$
$(z_1, \#, z_1) \rightarrow \varepsilon$
$(z_0, \#, z_1) \rightarrow \$(z_1, \#, z_1)$
$(z_0, A, z_1) \rightarrow \$(z_1, A, z_1)$
$(z_0, \#, z_1) \rightarrow a(z_0, A, z_1)\ (z_1, \#, z_1)$
$(z_0, A, z_1) \rightarrow a(z_0, A, z_1)\ (z_1, A, z_1)$

Für die Eingabe aa$aa ergeben sich in M folgende Konfigurationsübergänge:

$(z_0, aa\$aa, \#)$ |-- $(z_0, a\$aa, A\#)$ |-- $(z_0, \$aa, AA\#)$ |-- $(z_1, aa, AA\#)$ |-- $(z_1, a, A\#)$
|-- $(z_1, \varepsilon, \#)$ |-- $(z_1, \varepsilon, \varepsilon)$

Entsprechende Linksableitung in G:

S => $(z_0, \#, z_1)$
=> $a(z_0, A, z_1)\ (z_1, \#, z_1)$
=> $aa(z_0, A, z_1)\ (z_1, A, z_1)\ (z_1, \#, z_1)$
=> $aa\$(z_1, A, z_1)\ (z_1, A, z_1)\ (z_1, \#, z_1)$
=> $aa\$a\ (z_1, A, z_1)\ (z_1, \#, z_1)$
=> $aa\$aa\ (z_1, \#, z_1)$
=> aa$aa

In der Linksableitung erscheint am Anfang jeweils das in den Konfigurationsübergängen bereits abgearbeitete Stück des Eingabeworts, der mittlere Teil der folgenden Variablen ergibt jeweils den Stackinhalt. Die in den Variablen gespeicherten Zustände entsprechen jeweils denen, die vor und nach Abarbeitung des jeweiligen Stacksymbols in M gelten.

Korollare aus dem Beweis dieser Proposition:

1. Jeder Kellerautomat kann in einen äquivalenten Kellerautomaten mit genau 1 Zustand überführt werden (konstruiere Grammatik für N(M), erzeuge wie in Beweis Kellerautomat für diese Grammatik).
2. Für jede kontextfreie Sprache gibt es eine Grammatik, in der alle Regeln die Form
   $A \to a \qquad A \to aB \qquad A \to aBC$
   haben (erzeuge PDA M für Grammatik G, erzeuge wie in Beweis Grammatik für M).

7.3 Deterministisch kontextfreie Sprachen

Def.: Ein Kellerautomat M heißt deterministisch, falls für alle Zustände z, Eingabesymbole a und Kellersymbole A gilt:

$$|\delta(z, a, A)| + |\delta(z, \varepsilon, A)| \leq 1$$

Akzeptieren per leerem Keller/ per Endzustand:

Ein per Endzustand akzeptierender PDA ist ein Tupel

$$M = (Z, \Sigma, \Gamma, \delta, z_0, \#, F)$$

wobei F Menge der Endzustände ist. Die von M per Endzustand akzeptierte Sprache ist

$$L(M) = \{w \mid (z_0, w, \#) \vdash^* (z, \varepsilon, W), z \in F, W \in \Gamma^*\}$$

Im allgemeinen Fall macht es, wie wir gesehen haben, keinen Unterschied, ob man die von einem PDA akzeptierte Sprache über den leeren Keller oder über Endzustände definiert: wenn es einen per leerem Keller akzeptierenden PDA für L gibt, dann gibt es auch einen per Endzustand akzeptierenden PDA und umgekehrt.

Für deterministische PDAs (DPDAs) gilt diese Äquivalenz nicht!

Durch leeren Keller akzeptierende DPDAs sind ziemlich uninteressant, denn es können nicht einmal alle regulären Sprachen von ihnen akzeptiert werden:

Def.: Ein Sprache L heißt präfixfrei, wenn $a_1...a_n \in L$ impliziert: für alle $k < n$, $a_1...a_k \notin L$.

Satz: Sei L = N(M) für einen per leerem Keller akzeptierenden DPDA. L ist präfixfrei.

Beweis: Sei L nicht präfixfrei. Es gibt ein Wort w mit Präfix $w' \in L$. Da der Keller von M nach Abarbeiten von w' leer ist, wird w nicht akzeptiert.

Viele sehr einfache reguläre Sprachen sind nicht präfixfrei, etwa a*. Alle diese Sprachen können nicht von einem DPDA durch leeren Keller erkannt werden.
Deshalb betrachtet man per Endzustand akzeptierende DPDAs.

Deterministisch kontextfreie Sprache: durch DPDA (per Endzustand) erkannt.
Standardbeispiel: $a_1...a_n\$a_n...a_1$ (deterministisch) versus $a_1...a_na_n...a_1$ (nichtdeterministisch)

Deterministisch kontextfreie Sprachen sind echte Obermenge der regulären Sprachen:

Satz: Jede reguläre Sprache kann durch einen DPDA (per Endzustand) erkannt werden.

Beweis: offensichtlich, da ein DPDA, der den Stack nicht berücksichtigt und unverändert lässt, sich genauso verhält wie ein DEA.

Wie das Beispiel $a_1...a_n\$a_n...a_1$ zeigt, lassen sich auch nicht reguläre Sprachen erkennen.

Bei deterministischen Kellerautomaten gibt es für jede Konfiguration höchstens 1 Nachfolgekonfiguration. Deshalb ist das Wortproblem für deterministisch kontextfreie Sprachen in linearer Zeit lösbar. Aus diesem Grunde spielen sie bei der Definition von Programmiersprachen und im Compilerbau eine erhebliche Rolle.

Satz: Die deterministisch kontextfreien Sprachen sind unter Komplementbildung abgeschlossen.
(ohne Beweis)

Die kontextfreien Sprachen sind nicht unter Schnitt abgeschlossen:
$L_1 = \{a^nb^nc^m \mid n, m > 0\}$ kontextfrei
$L_2 = \{a^nb^mc^m \mid n, m > 0\}$ kontextfrei
$L_1 \cap L_2 = \{a^nb^nc^n \mid n > 0\}$ nicht kontextfrei (Pumping Lemma)

Da L_1 und L_2 deterministisch kontextfrei sind (Übungsaufgabe), bilden sie auch ein Gegenbeispiel für Nichtabgeschlossenheit unter Schnitt für deterministisch kontextfreie Sprachen.
Damit gilt auch Nichtabgeschlossenheit unter Vereinigung: mit Komplement und Vereinigung könnte Schnitt gebildet werden:

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

Satz: Die deterministisch kontextfreien Sprachen sind nicht unter Schnitt und Vereinigung abgeschlossen.

Satz: Der Schnitt einer (deterministisch) kontextfreien Sprache mit einer regulären Sprache ist eine (deterministisch) kontextfreie Sprache.

Beweis: Sei $M_1 = (Z_1, \Sigma, \Gamma, \delta_1, z_{01}, \#, E_1)$ (deterministischer) Kellerautomat (mit Endzuständen E_1), der (deterministisch) kontextfreie Sprache L_1 akzeptiert. Sei $M_2 = (Z_2, \Sigma, \delta_2, z_{02}, E_2)$ DEA, der reguläre Sprache L_2 akzeptiert. Der (deterministische) Kellerautomat

$$M_3 = (Z_1 \times Z_2, \Sigma, \Gamma, \delta_3, (z_{01}, z_{02}), \#, E_1 \times E_2)$$

mit $((z_1', z_2'), B_1\ldots B_k) \in \delta_3((z_1,z_2),a,A)$ gdw. $(z_1', B_1\ldots B_k) \in \delta_1(z_1,a,A)$ und $\delta_2(z_2,a) = z_2'$ erkennt $L_1 \cap L_2$.