Hans U. Simon
Annette Ilgen

Bochum, den 7.10.2008

Beispiele zur Vorlesung

Theoretische Informatik
WS 08/09

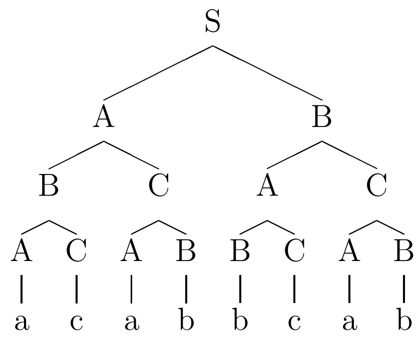
Vorbemerkung: Hier findet sich eine Sammlung von Beispielen und Motivationen zur Vorlesung Theoretische Informatik.

3 kontextfreie Sprachen

3.1 Chomsky Normalform

Sei eine Grammatik in Chomsky Normalform gegeben.
$V = \{A, B, C, S\}$
$\Sigma = \{a, b, c\}$
P in Regelnotation:
$S \to AB|BC$
$A \to BC|a$
$B \to AC|b$
$C \to AB|c$
$S =$ Startvariable

Dann ist der Syntaxbaum zu jedem Wort aus der zugehörigen Sprache binär, da sich jede Variable in genau zwei Variablen aufspaltet oder auf ein Terminalzeichen abbildet. Betrachten wir z.B. das Wort *acabbcab*. Der Syntaxbaum zu diesem Wort ist binär und sieht wie folgt aus.



3.2 Transformation in Chomsky Normalform

Sei eine kontextfreie Grammatik gegeben. $\Sigma = \{a, b\}$
$V = \{X, Y, S\}$

P in Regelnotation:
$S \rightarrow bXYXb|XY$
$X \rightarrow Xa|a|\epsilon$
$Y \rightarrow bYb|\epsilon$
$S =$ Startvariable

Diese lässt sich in vier Schritten in Chomsky Normalform transformieren.

Schritt 1 Separierung der Grammatik
Für jedes Terminalzeichen a wird eine neue Variable X_a und eine neue Regel $X_a \rightarrow a$ aufgenommen. Es brauchen aber nur die Terminalzeichen ersetzt werden die nicht alleine stehen.
Wir erhalten
$V = \{X, Y, X_a, X_b, S\}$
P in Regelnotation:
$S \rightarrow X_bXYXX_b|XY$
$X \rightarrow XX_a|a|\epsilon$
$Y \rightarrow X_bYX_b|\epsilon$
$X_a \rightarrow a$
$X_b \rightarrow b$
$S =$ Startvariable

Schritt 2 Verkürzung der rechten Seite.
Alle Regeln die auf eine Satzform der Länge drei oder mehr verweisen wie z.B.

$$S \rightarrow XYX$$

Werden mit Hilfe neuer Variabeln X_1, X_2, X_3 wie folgt ersetzt

$$S \rightarrow X_bXYXX_b \text{ durch } S \rightarrow X_bX_1,\ X_1 \rightarrow XX_2, X_2 \rightarrow YX_3, X_3 \rightarrow XX_b$$

Desweiteren ersetzt wird

$$Y \rightarrow X_bYX_b \text{ durch } Y \rightarrow X_bY_1\ Y_1 \rightarrow YX_b$$

Wir erhalten
$V = \{X, Y, X_a, X_b, X_1, X_2, X_3, Y_1, S\}$
P in Regelnotation:
$S \rightarrow X_bX_1|XY$
$X_1 \rightarrow XX_2$
$X_2 \rightarrow YX_3$
$X_3 \rightarrow XX_b$
$X \rightarrow XX_a|a|\epsilon$
$Y \rightarrow X_bY_1|\epsilon$
$Y_1 \rightarrow YX_b$
$X_a \rightarrow a$
$X_b \rightarrow b$

$S =$ Startvariable

Schritt 3 Elimination von ϵ-Regeln
Zunächst wird die Menge V_ϵ bestimmt. V_ϵ ist die Menge aller Variablen aus denen ϵ ableitbar ist.

Wir nehmen in V_ϵ zunächst alle Variablen auf, die direkt auf ϵ ableitbar sind.

$$V_\epsilon \supseteq \{X, Y\}$$

Dann werden zusätzlich alle Variablen aufgenommen, die auf eine Satzform ableiten, die nur aus bereits gefundenen Variablen besteht.

$$V_\epsilon \supseteq \{X, Y, S\}$$

Können keine weiteren gefunden werden, erhalten wir

$$V_\epsilon = \{X, Y, S\}$$

Nun bilden wir zusätzliche Regeln für alle Regeln der Form $A \to BC$ mit $B \in V_\epsilon$ oder $C \in V_\epsilon$ und entfernen dafür alle ϵ-Regeln.

Wir erhalten
$V = \{X, Y, X_a, X_b, X_1, X_2, X_3, Y_1, S\}$
P in Regelnotation:
$S \to X_b X_1 | XY | X | Y$
$X_1 \to XX_2 | X_2$
$X_2 \to YX_3 | X_3$
$X_3 \to XX_b | X_b$
$X \to XX_a | a$
$Y \to X_b Y_1$
$Y_1 \to YX_b | X_b$
$X_a \to a$
$X_b \to b$
$S =$ Startvariable

Schritt 4 Elimination von Kettenregeln.
Zunächst bestimmen wir K^+ zu der Relation K welche alle Paare von Variablen enthält zu denen es eine Kettenregel gibt.

$$K = \{(S, X), (S, Y), (X_1, X_2), (X_2, X_3), (X_3, X_b), (Y_1, X_b)\}$$

$$K^+ = K \cup \{(X_1, X_3), (X_1, X_b), (X_2, X_b)\}$$

Wir eliminieren alle Kettenregeln und führen dafür, wie in der Vorlesung beschrieben neuen Regeln ein.

Wir erhalten
$V = \{X, Y, X_a, X_b, X_1, X_2, X_3, Y_1, S\}$
P in Regelnotation:
$S \to X_bX_1|XY|XX_a|a|X_bY_1$
$X_1 \to XX_2|YX_3|XX_b|b$
$X_2 \to YX_3|XX_b|b$
$X_3 \to XX_b|b$
$X \to XX_a|a$
$Y \to X_bY_1$
$Y_1 \to YX_b|b$
$X_a \to a$
$X_b \to b$
$S =$ Startvariable

Die Grammatik ist nun in Chomsky Normalform.

3.3 Greibach Normalform

Folgende kontextfreie Grammatik ist in Greibach Normalform, denn die rechten Seiten der Regeln beginnen immer mit einem Terminalzeichen gefolgt von beliebig vielen Variablen.

$V = \{S, A, B, C\}$
P in Regelnotation:
$S \to aAB|bBA|cC$
$A \to a$
$B \to aBCA|b$
$C \to cBB$
$S =$ Startvariable

3.4 Transformation in Greibach Normalform

Sei folgende Grammatik über dem Alphabet $\Sigma = \{a, b, c\}$ gegeben.
$V = \{S, X, Y\}$
P in Regelnotation:
$S \to XSY|YX|a$
$X \to XY|b$
$Y \to SX|c$
$S =$ Startvariable

Um die Grammatik in Greibach Normalform umformen zu können, muss sichergestellt sein, dass die Grammatik separiert und gesäubert ist. Sie darf auch keine ϵ-Regeln enthalten. Dies ist hier gegeben.

Wir wählen eine Ordnung auf den Variablen. Dies können wir möglichst geschickt machen, sodass wir am Ende weniger Rechnenschritte brauchen. Um die Monotonieeigenschaft möglichst gut zu erfüllen sollte $S < X, S < Y$ und $Y < S$ sein. Da sich die beiden letzteren widersprechen wählen wir die erste Variante.

$V = \{S = A_1, X = A_2, Y = A_3\}$
P in Regelnotation:
$A_1 \to A_2A_1A_3|A_3A_2|a$
$A_2 \to A_2A_3|b$
$A_3 \to A_1A_2|c$
$S = A_1$ Startvariable

Schritt 1: Erzwingen der Monotonieeigenschaft
Wir wenden den Algorithmus aus der Vorlesung an.

- $i = 2, j = 2$ Die Regeln

  $$A_2 \to A_2A_3|b$$

  enthalten eine Linksrekursion, da A_2 auf sich selber gefolgt von einer beliebigen Satzform abgebildet wird. Sie wird ersetzt durch.

  $$A_2 \to b|bB_1$$

  $$B_1 \to A_3|A_3B_1$$

  wobei B_1 eine neu hinzugefügte Variable ist.

  Wir erhalten
  $V = \{A_1, A_2, A_3, B_1\}$
  P in Regelnotation:
  $A_1 \to A_2A_1A_3|A_3A_2|a$
  $A_2 \to b|bB_1$
  $B_1 \to A_3|A_3B_1$
  $A_3 \to A_1A_2|c$
  $S = A$ Startvariable

- $i = 3, j = 1$

  Die Regel $A_3 \to A_1A_2$ erfüllt nicht die Monotonieeigenschaft. Wir ersetzen sie durch $A_3 \to A_2A_1A_3A_2|A_3A_2A_2|aA_2$ und erhalten

  P in Regelnotation
  $A_1 \to A_2A_1A_3|A_3A_2|a$
  $A_2 \to b|bB_1$
  $B_1 \to A_3|A_3B_1$
  $A_3 \to A_2A_1A_3A_2|A_3A_2A_2|aA_2|c$
  $S = A$ Startvariable

- $i = 3, j = 2$

  Die Regel $A_3 \to A_2A_1A_3A_2$ erfüllt nicht die Monotonieeigenschaft. Wir ersetzen sie durch $A_3 \to bA_1A_3A_2|bB_1A_1A_3A_2$ und erhalten

  P in Regelnotation
  $A_1 \to A_2A_1A_3|A_3A_2|a$
  $A_2 \to b|bB_1$
  $B_1 \to A_3|A_3B_1$
  $A_3 \to bA_1A_3A_2|bB_1A_1A_3A_2|A_3A_2A_2|aA_2|c$
  $S = A$ Startvariable

- $i = 3, j = 3$

  Es gibt die linksrekursive Regl $A_3 \to A_3A_2A_2$. Daher werden alle Regeln $A_3 \to bA_1A_3A_2|bB_1A_1A_3A_2|A_3A_2A_2|aA_2|c$ ersetzt durch

  $$A_3 \to bA_1A_3A_2|bB_1A_1A_3A_2|aA_2|c|bA_1A_3A_2B_2|bB_1A_1A_3A_2B_2|aA_2B_2|cB_2$$

  $$B_2 \to A_2A_2|A_2A_2B_2$$

  wobei B_2 eine weitere neue Variable ist. Wir erhalten

  P in Regelnotation
  $A_1 \to A_2A_1A_3|A_3A_2|a$
  $A_2 \to b|bB_1$
  $A_3 \to bA_1A_3A_2|bB_1A_1A_3A_2|aA_2|c|bA_1A_3A_2B_2|bB_1A_1A_3A_2B_2|aA_2B_2|cB_2$
  $B_1 \to A_3|A_3B_1$
  $B_2 \to A_2A_2|A_2A_2B_2$
  $S = A$ Startvariable

Schritt 2: Transformation in Greibach-Normalform abgesehen von B-Regeln.
Die rechten Seiten der A_3-Regeln beginnen nun mit einem Terminalzeichen. Wir wenden den Algorithmus aus der Vorlesung an um dies auch fr die A_1 und A_2-Regeln zu erreichen.

- $i = 1, j = 2$

  Die Regel $A_1 \to A_2A_1A_3$ beginnt nicht mit einem Terminalzeichen. Wir ersetzen sie durch $A_1 \to bA_1A_3|bB_1A_1A_3$ und erhalten.

  P in Regelnotation
  $A_1 \to bA_1A_3|bB_1A_1A_3|A_3A_2|a$
  $A_2 \to b|bB_1$
  $A_3 \to bA_1A_3A_2|bB_1A_1A_3A_2|aA_2|c|bA_1A_3A_2B_2|bB_1A_1A_3A_2B_2|aA_2B_2|cB_2$
  $B_1 \to A_3|A_3B_1$
  $B_2 \to A_2A_2|A_2A_2B_2$
  $S = A$ Startvariable

- $i=1, j=3$

  Die Regel $A_1 \to A_3A_2$ beginnt nicht mit einem Terminalzeichen. Wir ersetzen sie durch $A_1 \to bA_1A_3A_2A_2|bB_1A_1A_3A_2^2|aA_2^2|cA_2|bA_1A_3A_2B_2A_2|bB_1A_1A_3A_2B_2A_2|aA_2B_2A_2|cB_2A_2$ und erhalten.

  P in Regelnotation
  $A_1 \to bA_1A_3|bB_1A_1A_3|bA_1A_3A_2^2|bB_1A_1A_3A_2^2|aA_2^2|cA_2|bA_1A_3A_2B_2A_2$
  $|bB_1A_1A_3A_2B_2A_2|aA_2B_2A_2|cB_2A_2|a$
  $A_2 \to b|bB_1$
  $A_3 \to bA_1A_3A_2|bB_1A_1A_3A_2|aA_2|c|bA_1A_3A_2B_2|bB_1A_1A_3A_2B_2|aA_2B_2|cB_2$
  $B_1 \to A_3|A_3B_1$
  $B_2 \to A_2^2|A_2^2B_2$
  $S = A$ Startvariable

Schritt 3: Transformation der B-Regeln in Greibach Normalform für jede Hilfsvariable B
Zuletzt werden auch die B-Regeln in Greibach Normalform gebracht. Wir wenden wieder den Algorithmus aus der Vorlesung an.

- $i=1, j=3$ Die rechte Seite der Regel $B_1 \to A_3$ beginnt nicht mit einem Terminalzeichen. Wir ersetzen sie durch
  $B_1 \to bA_1A_3A_2|bB_1A_1A_3A_2|aA_2|c|bA_1A_3A_2B_2|bB_1A_1A_3A_2B_2|aA_2B_2|cB_2$

  Die rechte Seite der Regel $B_1 \to A_3B_1$ beginnt nicht mit einem Terminalzeichen. Wir ersetzen sie durch
  $B_1 \to bA_1A_3A_2B_1|bB_1A_1A_3A_2B_1|aA_2B_1|cB_1|bA_1A_3A_2B_2B_1|bB_1A_1A_3A_2B_2B_1$
  $|aA_2B_2B_1|cB_2B_1$ und erhalten

  P in Regelnotation
  $A_1 \to bA_1A_3|bB_1A_1A_3|bA_1A_3A_2^2|bB_1A_1A_3A_2^2|aA_2^2|cA_2|bA_1A_3A_2B_2A_2$
  $|bB_1A_1A_3A_2B_2A_2|aA_2B_2A_2|cB_2A_2|a$
  $A_2 \to b|bB_1$
  $A_3 \to bA_1A_3A_2|bB_1A_1A_3A_2|aA_2|c|bA_1A_3A_2B_2|bB_1A_1A_3A_2B_2|aA_2B_2|cB_2$
  $B_1 \to bA_1A_3A_2|bB_1A_1A_3A_2|aA_2|c|bA_1A_3A_2B_2|bB_1A_1A_3A_2B_2|aA_2B_2|cB_2$
  $B_1 \to bA_1A_3A_2B_1|bB_1A_1A_3A_2B_1|aA_2B_1|cB_1|bA_1A_3A_2B_2B_1|bB_1A_1A_3A_2B_2B_1$
  $|aA_2B_2B_1|cB_2B_1$
  $B_2 \to A_2^2|A_2^2B_2$
  $S = A$ Startvariable

- $i=2, j=2$ Die rechte Seite der Regel $B_2 \to A_2^2$ beginnt nicht mit einem Terminalzeichen. Wir ersetzen sie durch $B_2 \to bA_2|bB_1A_2$

  Die rechte Seite der Regel $B_2 \to A_2^2B_2$ beginnt nicht mit einem Terminalzeichen. Wir ersetzen sie durch $B_2 \to bA_2B_2|bB_1A_2B_2$ und erhalten.

  P in Regelnotation
  $A_1 \to bA_1A_3|bB_1A_1A_3|bA_1A_3A_2^2|bB_1A_1A_3A_2^2|aA_2^2|cA_2|bA_1A_3A_2B_2A_2$

$|bB_1A_1A_3A_2B_2A_2|aA_2B_2A_2|cB_2A_2|a$
$A_2 \to b|bB_1$
$A_3 \to bA_1A_3A_2|bB_1A_1A_3A_2|aA_2|c|bA_1A_3A_2B_2|bB_1A_1A_3A_2B_2|aA_2B_2|cB_2$
$B_1 \to bA_1A_3A_2|bB_1A_1A_3A_2|aA_2|c|bA_1A_3A_2B_2|bB_1A_1A_3A_2B_2|aA_2B_2|cB_2$
$B_1 \to bA_1A_3A_2B_1|bB_1A_1A_3A_2B_1|aA_2B_1|cB_1|bA_1A_3A_2B_2B_1$
$|bB_1A_1A_3A_2B_2B_1|aA_2B_2B_1|cB_2B_1$
$B_2 \to bA_2|bB_1A_2|bA_2B_2|bB_1A_2B_2$
$S = A$ Startvariable

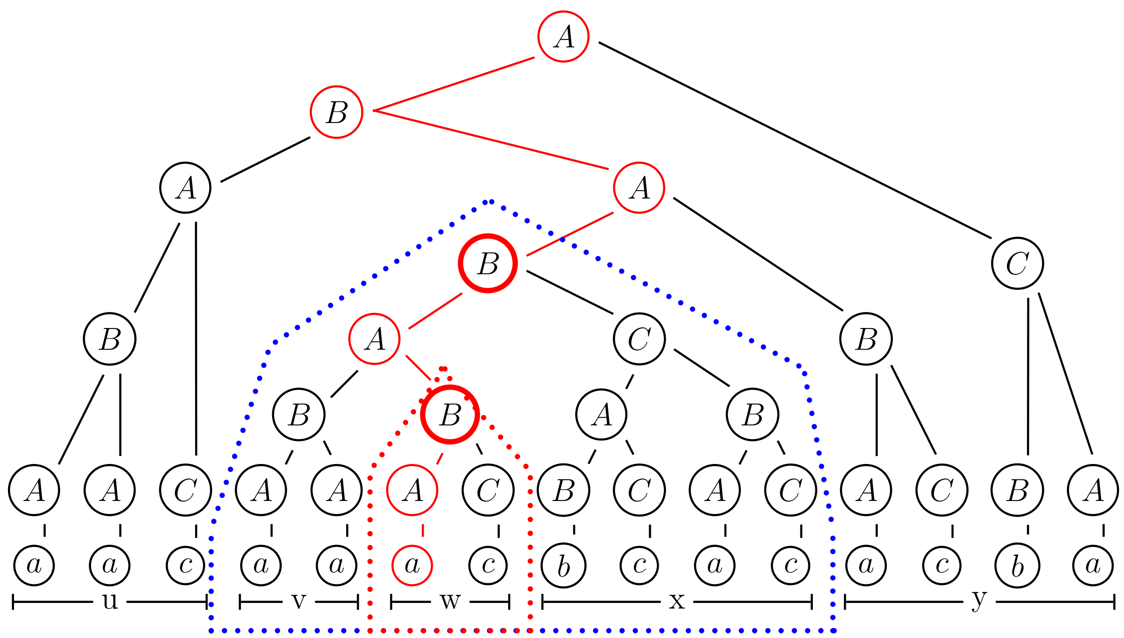
Damit befindet sich die Grammatik nun in Greibach Normalform ohne, dass die Sprache die sie beschreibt verändert wurde.

3.5 Pumping Lemma

Wir wollen ein Beispiel für eine konkrete Zerlegung eine Wortes nach dem Pumping Lemma finden. Betrachten wir ein Beispiel für eine kontextfreie Grammatik G über dem Alphabet $\Sigma = \{a, b, c\}$

$V = \{A, B, C\}$
P in Regelnotation
$A \to BB|BC|a$
$B \to AC|AA|b$
$C \to AB|BA|c$
$S = A$ Startvariable

Dann ist $k := |V| = 3$ und sei $n := 2^k = 8$.
Betrachten wir zum Beispiel das Wort $z = a^2ca^3cbcacacba \in L(G)$ mit $|z| = 15 \geq 2^k = 8$.
Nach dem Pumping-Lemma gibt es eine Zerlegung $z = uvwxy$ mit $1 \leq |vx| \leq |vwx| \leq n$, sodass $uv^iwx^iy \in L(G)$ für alle $i \geq 0$.
Wir suchen nun diese Zerlegung und betrachten dazu den Syntaxbaum des Wortes z.



Ein maximaler Pfad, welcher die Variable B doppelt enthält ist hier rot markiert. Durch die beiden B werden zwei Teilbäume aufgespannt. Durch sie zerfällt das Wort in die fünf Teile u, v, w, x, y mit $u = aac, v = aa, w = ac, x = bcac, y = acba$. Indem man nun den großen Teilbaum durch den kleinen ersetzt erhält man das Wort uv^0wx^0y. Und indem man den kleineren Teilbaum iterativ durch den größeren ersetzt erhält man die Wörter uv^iwx^iy mit $i \geq 2$. Insgesamt ergibt sich also, dass $aac(aa)^iac(bcac)^iacba \in L(G)$ für alle $i \geq 1$.

3.6 Der CYK-Algorithmus

Betrachten wir die Grammatik aus Beispiel 3.5. Wir wollen nun mittels des CYK-Algorithmus testen ob das Wort $z = bcacacba$ in $L(G)$ liegt.

Wir stellen die Tabelle auf und berechnen iterativ die $T[i, j]$. Die farbige Graphik veranschaulicht dabei das Vorgehen zur Berechnung von $T[1, 4]$. Es werden alle Variablen gesucht die in den Regeln der Grammatik auf B, AB, BC abgeleitet werden können. Dies sind genau A und C.

$i\downarrow \mid j \rightarrow$	1	2	3	4	5	6	7	8
b	B	A	B	A, C	B	A, B, C	A, B, C	A, B, C
c	C							-
a	A	B	C	A	C	A, B, C	-	-
c	C					-	-	-
a	A	B	A	B, A	-	-	-	-
c	C			-	-	-	-	-
b	B	C	-	-	-	-	-	-
a	A	-	-	-	-	-	-	-

Wir erhalten für $T[1, 8]$ schlielich A, B, C. Das sagt uns, dass das Wort $bcacacba$ aus allen drei Buchstaben A, B, C abgeleitet werden kann. Da $A = S$ die Startvariable ist, ist das

Wort also Element der Sprache $L(G)$.

3.7 Kellerautomat(PDA)

Wir geben ein Beispiel für einen Kellerautomaten an, der die Sprache $L = \{w \in \Sigma^* | \ |w|_a = |w|_b\}$ über dem Alphabet $\Sigma = \{a, b\}$ erkennt.

Wir benötigen nur einen Zustand, der zugleich Startzustand ist.
$Z = \{z\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \#\}$
Wobei $\#$ das unterste Kellerzeichen ist.

Wir definieren folgende Zustandswechsel.

$$z\# \xrightarrow{a} za\# \quad , \quad za \xrightarrow{a} zaa, \quad zb \xrightarrow{a} z\epsilon$$
$$z\# \xrightarrow{b} zb\# \quad , \quad za \xrightarrow{b} z\epsilon, \quad zb \rightarrow zbb$$

und

$$z\# \xrightarrow{\epsilon} z\epsilon$$

Sie sind so zu verstehen, dass wir bei einem, bis auf das unterste Kellerzeichen leerem Keller, den Buchstaben des Wortes einfügen auf den der Lesekopf zeigt. Im folgenden werden gleiche Buchstaben im Keller gespeichert, wobei ungleiche sich gegeneinander wegheben. Hatte unser Wort gleich viele a wie b, so ist der Keller nach Abarbeitung des Wortes, bis auf das unterste Kellerzeichen leer, da im Keller niemals zwei unterschiedliche Buchstaben gespeichert werden.

Auf Eingabe von $abba$ gibt es zum Beispiel die Rechnung:

$$(z, abba, \#) \vdash (z, bba, a\#) \vdash (z, ba, \#) \vdash (z, a, b\#) \vdash (z, \epsilon, \#) \vdash (z, \epsilon, \epsilon)$$

Auf Eingabe von $aabb$ gibt es die Rechnung

$$(z, , aabb\#) \vdash (z, abb, a\#) \vdash (z, bb, aa\#) \vdash (z, b, a\#) \vdash (z, \epsilon, \#) \vdash (z, \epsilon, \epsilon)$$

Beachte, dass der PDA nicht deterministisch arbeitet. So kann er auf dem Wort $abba$ auch folgende Rechnung ausführen.

$$(z, abba, \#) \vdash (z, bba, a\#) \vdash (z, ba, \#) \vdash (z, ba, \epsilon)$$

Das unterste Kellerzeichen wird somit frühzeitig gelöscht und der Automat stopp verwerfend, da das Wort noch nicht bis zum Ende abgearbeitet ist, jedoch keine Rechenschritte für einen komplett leeren Keller möglich sind ($\epsilon \notin \Gamma$). Dennoch liegt $abba \in N(M)$, denn es reicht aus, dass es eine Rechnung gibt(s.o.), die das Wort akzeptiert.

3.8 Von der kontextfreien Grammatik zum PDA

Gegeben sei die Grammatik
$G = (V, \Sigma, P, S)$
$\Sigma = \{a, b\}$
$V = \{S\}$
P in Regelnotation
$S \to aSbb|aaSb|\epsilon$
$S =$ Startvariable

Wir suchen nun einen Kellerautomaten PDA $M = (Z, \Sigma, \Gamma, \delta, z, S)$ der die selbe Sprache erkennt die die Grammatik erzeugt. Wir benötigen nur einen Zustand $Z = \{z\}$, der dann natürlich auch Startzustand ist.

$$\Gamma = V \cup \Sigma = \{S, a, b\}$$

wobei S das unterste Kellerzeichen und δ wie folgt gegeben ist.

$$zS \xrightarrow{\epsilon} zaaSb \quad , \quad zS \xrightarrow{\epsilon} zaSbb \quad , \quad zS \xrightarrow{\epsilon} z\epsilon$$

$$za \xrightarrow{a} z\epsilon \quad , \quad zb \xrightarrow{b} z\epsilon$$

Man beachte, dass der PDA nichtdeterministisch arbeitet. Betrachten wir zum Beispiel das Wort a^3b^3 und wenden zunächst die ϵ-Transition auf es an, welche einer Linksableitung entsprechen die das Wort erzeugen. Mit einer a-Transition wird der Keller dann soweit gelöscht, bis wieder eine Variable oben steht, welche mittels ϵ-Transition wieder ersetzt wird.

$$(z, a^3b^3, S) \vdash (z, a^3b^3, aSb^2) \vdash (z, a^2b^3, Sb^2) \vdash (z, a^2b^3, a^2Sb^3) \vdash (z, ab^3, aSb^3)$$

$$\vdash (z, b^3, Sb^3) \vdash (z, b^3, b^3) \vdash (z, b^2, b^2) \vdash (z, b, b) \vdash (z, \epsilon, \epsilon)$$

Wenn sich die Grammatik in Greibach Normalform befindet, können wir auf die ϵ-Transitionen sogar ganz verzichten.
$V = \{S\}$
P in Regelnotation
$S \to aSBB|aASB|aBB|aAB$
$A \to a$
$B \to b$
$S =$ Startvariable

Wir konstruieren den PDA wie oben, jedoch reicht es wenn wir für δ folgende Transitionen angeben.

$$zS \xrightarrow{a} zSBB \quad , \quad zS \xrightarrow{a} zASB \quad , \quad zS \xrightarrow{a} zBB, \quad zS \xrightarrow{a} zAB$$

$$zA \xrightarrow{a} z\epsilon \quad , \quad zB \xrightarrow{b} z\epsilon$$

Um das Wort a^3b^3 mit dem PDA zu verarbeiten wählen wir nun folgende Transitionen

$$(z, a^3b^3, S) \vdash (z, a^2b^3, SBB) \vdash (z, ab^3, ABBB) \vdash (z, b^3, BBB) \vdash (z, b^2, BB) \vdash (z, b, B) \vdash (z, \epsilon, \epsilon)$$

3.9 Deterministischer Kellerautomat

Betrachte die Sprache der korrekten Klammerung von Ausdrücken mit $\Sigma = \{[,]\}$, die erzeugt wird durch die Grammatik
$V = \{S\}$
$S =$ Startvariable
P in Regelnotation
$S \rightarrow [S]|SS|\epsilon$

Ein DPDA der diese Sprache erkennt hat folgende Komponenten.
$Z = \{z_0, z_1\}$
$\Sigma = \{[,]\}$
$\Gamma = \{\tilde{[}, [, \#\}$
$E = \{z_0\}$
Wobei $\#$ das unterste Kellerzeichen ist. Wir definieren folgende Zustandswechsel.

$$
\begin{array}{ll}
z_0\# \xrightarrow{[} z_1\tilde{[}\# & z_1[\xrightarrow{[} z_1[[\\
z_1\tilde{[} \xrightarrow{[} z_1[\tilde{[} & z_1[\xrightarrow{]} z_1\epsilon \\
z_1\tilde{[} \xrightarrow{]} z_0\epsilon &
\end{array}
$$

Beachte, dass der DPDA mit Endzuständen akzeptiert. Daher müssen wir die erste in den Keller eingefügte Klammer kenntlich machen, damit wir merken wenn der Keller durch schließende Klammern bis auf das unterste Kellerzeichen geleert wird. In diesem Fall gehen wir in einen Endzustand über.

Der Automat arbeitet deterministisch. Es gibt immer nur einen vorgegebenen Rechenschritt den er ausführen muss. Für das Wort "[][]" rechnet er zum Beispiel wie folgt.

$$(z_0, [][], \#) \vdash (z_1,][], \tilde{[}\#) \vdash (z_0, [], \#) \vdash (z_1,], \tilde{[}\#) \vdash (z_0, \epsilon, \#)$$

Beachte, dass der DPDA in Zwischenschritten bereits einen Endzustand erreicht. Er bricht jedoch nicht ab, da das Wort noch nicht bis zum Ende gelesen wurde.

3.10 Abschlusseigenschaften

Betrachte die Sprache $L_{ab} := \{w \in \{a, b, c\}^* |\ |w|_a = |w|_b\}$. Sie ist deterministisch Kontextfrei. Ein DPDA arbeitet analog wie in Beispiel 3.9. Das erste eingefüge Zeichen wird gesondert gespeichert. Wird dieses gesonderte Zeichen im Verlauf der Rechnung gelöscht so gehe in einen Endzustand über.

- deterministisch kontextfreie Sprachen sind Abgeschlossen unter Komplementbildung Daher ist die Sprache $\overline{L_{ab}} = \{w \in \{a, b, c\}^* |\ |w|_a \neq |w|_b\}$ ebenfalls deterministisch Kontextfrei.

- kontextfreie Sprachen sind abgeschlossen unter Vereinigung Daher sind die Sprachen $L_{ab} \cup L_{ac} \cup L_{bc} = \{w \in \{a,b,c\}^* |\ |w|_a = |w|_b$ oder $|w|_a = |w|_c$ oder $|w|_b = |w|_c\}$ und $\overline{L_{ab}} \cup \overline{L_{ac}} \cup \overline{L_{bc}} = \{w \in \{a,b,c\}^* |\ |w|_a \neq |w|_b$ oder $|w|_a \neq |w|_c$ oder $|w|_a \neq |w|_b\}$ kontextfrei.

- kontextfreie Sprachen sind nicht abgeschlossen unter Schnittmengenbildung Ein Gegenbeispiel liefert $L_{abc} = L_{ab} \cap L_{bc} = \{w \in \{a,b,c\}^* |\ |w|_a = |w|_b$ und $|w|_b = |w|_c\} = \{w \in \{a,b,c\}^* |\ |w|_a = |w|_b = |w|_c\}$. Denn L_{abc} ist nicht kontextfrei. Der Beweis wird mittels des Pumping Lemmas geführt und verläuft analog zum Beispiel für die Sprache $\{a^m b^m c^m \in \Sigma^* | m \geq 1\}$ aus dem Vorlesungsskript (Kapitel 1.3 S. 28)

3.11 Säubern einer Grammatik

Sei folgende Grammatik über dem Alphabet $\Sigma = \{a,b,c\}$ gegeben.

$V = \{S, A, B, C, X, Y, Z\}$
P in Regelnotation:
$S \to AY|YZ|BZ$
$X \to AbZ|ABc$
$Y \to aYC|AYc$
$Z \to aYC|bC$
$A \to bc$
$B \to Y|ac$
$C \to Z|ab$
$S =$ Startvariable

Wir suchen alle generierenden Variabeln V_g.

$$V_g \supseteq \{A, B, C\} \text{ wegen } A \to bc, B \to ac \text{ und } C \to ab$$

$$V_g \supseteq \{A, B, C, X, Z\} \text{ wegen } X \to ABc \text{ und } Z \to bC$$

$$V_g \supseteq \{A, B, C, X, Z, S\} \text{ wegen } S \to BZ$$

Y ist nicht generierend.

$$V_g = \{A, B, C, X, Z, S\}$$

Wir entfernen aus der Grammatik alle nicht generierenden Variabeln und erhalten.
$V = \{S, A, B, C, X, Z\}$
P in Regelnotation:
$S \to BZ$
$X \to AbZ|ABc$
$Z \to bC$
$A \to bc$
$B \to ac$
$C \to Z|ab$

$S =$ Startvariable

Wir suchen alle erreichbaren Variablen V_e.

$$V_e \supseteq \{S, B, Z\} \text{ wegen } S \to BZ$$

$$V_e \supseteq \{S, B, Z, C\} \text{ wegen } Z \to bC$$

A und X sind nicht erreichbar.

$$V_e \supseteq \{S, B, C, Z\}$$

Wir entfernen aus der Grammatik alle nicht erreichbaren Variabeln und erhalten.
$V = \{S, B, C, Z\}$
P in Regelnotation:
$S \to BZ$
$Z \to bC$
$B \to ac$
$C \to Z|ab$
$S =$ Startvariable
Die Grammatik ist nun gesäubert.