Algorithmen und Datenstrukturen (EI)

Zentralübung

Stefan Schmid

14. Januar 2009

TECHNISCHE UNIVERSITÄT MÜNCHEN

Fitnesstraining...

| aDSei | = 5 ***(„Länge des Strings“)***

| {a,d,s} | = 3 ***(„Kardinalität der Menge“)***

| {} | = 0 ***(Leere Menge enthält keine Elemente!)***

| ε | = 0 ***(Leerer String hat Länge 0)***

Potenzmenge von {a,d,s} = $2^{\{a,d,s\}}$

= {{},{a} ,{d} ,{s} ,{a,d} ,{a,s},{d,s} ,{a,d,s}}

| 2^{} | = |{{}}| = 1 ***(Menge bestehend aus leerer Menge)***

Ist {} ∈ {}? Nein, leere Menge nicht in leerer Menge!

Ist {} ⊆ {}? Ja.

| $\emptyset^*$ | = | {ε} | = 1 ***(Eine Menge, also Kardinalität...)***

Fitnesstraining...

$\bigcup_{i \in \{1,3,6,7\}} x_i = \{x_1, x_3, x_6, x_7\}$

Ist Schnitt von regulärer und kontextfreier Sprache auch kontextfrei?

Ja. Idee: Mache PDA der gleichzeitig die Uebergänge des endlichen Automaten „mitverfolgt“ (siehe Konstruktion der Schnittmenge von regulären Sprachen). Akzeptiere nur wenn beide akzeptieren.

Was ist mit der Vereinigung? Mit dem Linksquotienten? Mit ...??

f(n) = f(n-1) + n, f(1) = 1. Was ist f(10)?

f(10) = f(9) + 10 = f(8) + 9 + 10 = f(7) + 8 + 9 + 10
= ... = 1 + 2 + ... + 10 = (10*11)/2 = 55

Fitnesstraining...

$n^3 - n^2 - 1 \in O(n^2)$? **Nein**

$n^3 + n^2 - 1 \in O(n^3)$? **Ja** http://en.wikipedia.org/wiki/Big_O_notation

Für jede kontextfreie Sprache gibt‘s einen deterministischen PDA?

Nein, deterministische Sprachen sind Subklasse

Für jede LL(k) Grammatik gibt‘s äquivalente LR(k) Grammatik?

Ja

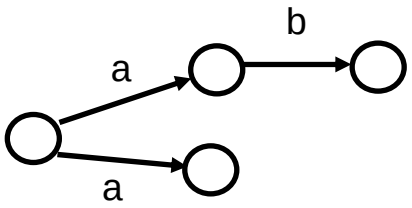
Etc. etc.

Gute Multiple-Choice Aufgaben! (vermutlich erste Prüfungsaufgabe...)

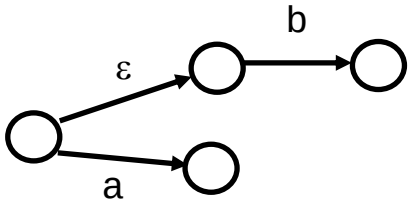
Deterministisch vs Nichtdeterministisch

Deterministisch: Aus gegebener Konfiguration folgt eindeutig nächste Konfiguration ($|\delta(\cdot)| \leq 1$)

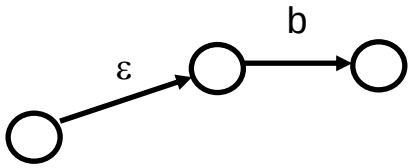
Nichtdeterministisch: Nicht deterministisch ☺ ($|\delta(\cdot)| > 1$ zum Teil)



Nichtdeterministischer endlicher Automat



Nichtdeterministischer endlicher Automat



Deterministischer endlicher Automat
(mit ε-Uebergang, aber immer weglassbar)

Pumping Lemmas (1)

Allgemein: Wir verwenden Pumping Lemmas häufig, um zu zeigen dass gewisse Sprachen *nicht* von einem gewissen Typ sind.
Wir machen dazu einen Widerspruchsbeweis!

Pumping Lemmas machen meistens eine Aussage der Art: „Wenn die Sprache von einem Typ ist, gibt es für genug lange Worte eine Zerlegung uvwxy, sodass das gepumpte Wort (gewisse Teile wiederholen) auch in der Sprache ist.

Pumping Lemmas (2)

Um zu zeigen, dass eine Sprache nicht von einem Typ ist, müssen wir also zeigen dass es keine solche Zerlegung gibt. Wir müssen also für alle möglichen Zerlegungen zeigen, dass es nicht geht.

ALSO: Wir müssen für ein gut gewähltes genug langes Wort ($|z| \geq n$), für alle möglichen Zerlegungen, zeigen, dass es für eine Art zu pumpen (für ein i) nicht geht.

Pumping Lemma: Reguläre Sprachen

Pumping Konstante (entspricht Anzahl Zuständen)

Satz 82 (Pumping Lemma für reguläre Sprachen)

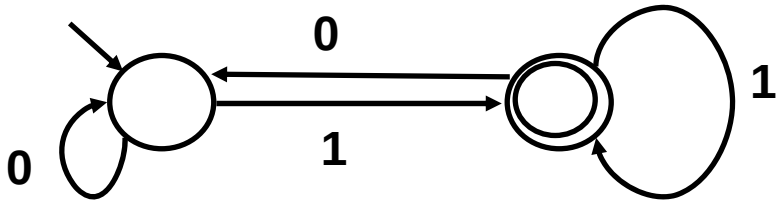
Sei $R \subseteq \Sigma^*$ regulär. Dann gibt es ein $n > 0$, so dass für jedes $z \in R$ mit $|z| \geq n$ es $u, v, w \in \Sigma^*$ gibt, so dass gilt:

1. $z = uvw$,
2. $|uv| \leq n$,
3. $|v| \geq 1$, *und*
4. $\forall i \geq 0 : \; uv^i w \in R$.

ALSO: Wir müssen für ein gut gewähltes genug langes Wort ($|z| \geq n$), für alle möglichen Zerlegungen, zeigen, dass es für eine Art zu pumpen (für ein i) nicht geht.

Beispiel: n=2

Ein endlicher Automat mit zwei Zuständen: Spätestens nach 2 Zeichen Ist er wieder im gleichen Zustand wie schon mal!



Jedes Wort aus mind. zwei Zeichen ist pumpbar gemäss Pumping Lemma.

Beispiel: 01 ist in der Sprache. Sei u=ε, v=0, w=1 (wähle v als Loop)

Dann sind auch 1, 001, 0001, 00001, etc. in der Sprache!

Pumping Lemma: Kontextfreie Sprachen

Pumping Konstante (hat mit Anzahl Variablen in CFG zu tun)

Satz 96 (Pumping-Lemma)

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ zerlegen lässt in

$$z = uvwxy,$$

mit

1. $|vx| \geq 1$,
2. $|vwx| \leq n$, *und*
3. $\forall i \in \mathbb{N}_0 : \; uv^i wx^i y \in L$.

ALSO: Wir müssen für ein gut gewähltes genug langes Wort ($|z| \geq n$), für alle möglichen Zerlegungen, zeigen, dass es für eine Art zu pumpen (für ein i) nicht geht.

Pumping Lemma: Ogden

Verallgemeinerung: mit Markierungen!

Satz 98 (Ogdens Lemma)

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass für jedes Wort $z \in L$ mit $|z| \geq n$ die folgende Aussage gilt: Werden in z mindestens n (beliebige) Buchstaben markiert, so lässt sich z zerlegen in

$$z = uvwxy,$$

so dass

1. *in vx mindestens ein Buchstabe und*
2. *in vwx höchstens n Buchstaben markiert sind und*
3. $(\forall i \in \mathbb{N}_0)[uv^iwx^iy \in L]$.

Bemerkung: Das Pumping-Lemma ist eine triviale Folgerung aus Ogdens Lemma (markiere alle Buchstaben in z).

Pumping Lemma: Zeigen dass etwas in der Sprache

Bemerkung:
Wie wir gerade gesehen haben, gilt die Umkehrung des Pumping-Lemmas nicht allgemein (d.h., aus dem Abschluss einer Sprache unter der Pumpoperation des Pumping-Lemmas folgt i.A. nicht, dass die Sprache kontext-frei ist).
Es gibt jedoch stärkere Versionen des Pumping-Lemmas, für die auch die Umkehrung gilt. Siehe dazu etwa

David S. Wise:
A strong pumping lemma for context-free languages.
Theoretical Computer Science **3**, pp. 359–369, 1976

Richard Johnsonbaugh, David P. Miller:
Converses of pumping lemmas.
ACM SIGCSE Bull. **22**(1), pp. 27–30, 1990

Tandem Pumping Lemma: Korrektur

Beispiel 97

Wir wollen sehen, dass die Sprache

$$\{a^i b^i c^i;\ i \in \mathbb{N}_0\}$$

nicht kontextfrei ist.

Satz 96 (Pumping-Lemma)

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ zerlegen lässt in

$$z = uvwxy,$$

mit

1. $|vx| \geq 1$,
2. $|vwx| \leq n$, *und*
3. $\forall i \in \mathbb{N}_0 :\ uv^i wx^i y \in L$.

Betrachte $a^n b^n c^n$

Wo kann vwx sein?

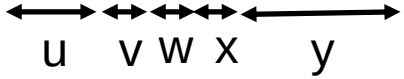
Kann nicht sowohl a‘s wie b‘s wie c‘s enthalten, da maximal n lange!
Durch Pumpen wird dadurch aber die Anzahl der Buchstaben unterschiedlich.
Also nicht mehr in Sprache. Widerspruch.

Beispielzerlegung

Sei n=3: **aaabbbccc**

Wo kann vwx sein? Höchstens 3 lange!

Beispiel hier: **aaabbbccc**



Gepumpt mit i=3?

aaaaaabbbbbccc nicht in L!

Zusammenfassung für Prüfung... (Nur Tipp!)

Selber machen & klein schreiben ☺

Was man nicht weiss, z.B. genaue Definitionen

- Definitionen der Grammatiken (Typ 0, Typ 1, etc.)
- Pumping Lemmas
- Algorithmen (ev. auch in eigenen Worten zusätzlich), z.B. State Minimization
- Normalformen der Logik
- Ev. kleine „Ideen-Sammlung“: Wie beweist man dass eine Sprache nicht deterministisch-kontextfrei ist? Etc.

Uebersichten / Mindmaps machen (ev. dann nicht mal nötig mitzunehmen)

- Welche Maschinen sind deterministisch und nichtdeterministisch gleich stark?
- „Alles was ich zu XYZ weiss“! Zum Beispiel reguläre Sprachen: NFA ⇔ DFA, reguläre Ausdrücke, reguläre Grammatiken, abgeschlossen unter Vereinigung und Komplement und Schnitt, Algorithmen zur Umwandlung von regulär zu Grammatik zu NFA zu DFA (und zurück)

Ziel: alles „vom Prinzip her“ verstanden haben, Faktenwissen zur Sicherheit auf Zusammenfassung

Prüfung!

Aufgabe 3 (8 Punkte)

Sei $\Sigma = \{b, c, d\}$. Sei L die Sprache

$$L = (b^*c \mid db^*)^* .$$

1. Geben Sie eine Grammatik G an, die L erzeugt.

1. Sieht man einfach, wie Sprache ist?

2. Kennen wir einen direkten Algorithmus?

3. Falls nicht grad klar, Idee schreiben was ginge (wenn auch zu aufwendig)!

   Z.B.: Mache NFA draus, wandle ihn in DFA, daraus lässt sich direkt Grammatik machen => 3 Algorithmen, aufwendig, aber Lösungsweg trotzdem hinschreiben!

Zusatzaufgaben

Mehr Aufgaben (auch einfachere und „Tutoraufgaben“) auch von Prof. Mayrs Vorlesung zur „Theoretischen Informatik“

- ev. stellen wir da noch was zur Verfügung!

Buchtipp Student?

Aufgabe 1

Zeigen Sie, dass die Sprache $L = \{a^{(k^3)} ; k \in \mathbb{N}\}$ nicht kontextfrei ist.

Widerspruchsbeweis mit Pumping Lemma!

Betrachte Wort $a^{n^3} \in L$, wobei n die Pumping Länge ist.

Erfüllt Pumping Bedingungen, also muss auch $z=uv^2wx^2y \in L$ sein, für eine Zerlegung

Aus wievielen a's besteht z? z ist von der Form $z = a^{n^3+|vx|}$, einmal gepumpt

Weil aber $0 < |vx| \leq n$ (siehe Lemma), ist

$n^3 < |z|=n^3+|vx| < n^3 + n+1 < (n+1)^3 = n^3 + 3n^2 + 3n + 1$

Also z nicht in L. Widerspruch.

Satz 96 (Pumping-Lemma)

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ zerlegen lässt in

$$z = uvwxy,$$

mit

1. $|vx| \geq 1$,
2. $|vwx| \leq n$, *und*
3. $\forall i \in \mathbb{N}_0 : uv^iwx^iy \in L$.

Aufgabe 2

Gegeben seien $\Sigma = \{d, e, f\}$ und $L = \{d^k e^m f^n \,;\, k, m, n \in \mathbb{N}, m \neq n\}$.
Zeigen Sie, dass die Sprache L kontextfrei ist.

Idee: Kontextfreie Grammatik angeben! Aufwendig!

Bessere Idee: Nutze Tatsache dass kontextfreie Sprachen abgeschlossen bezüglich Konkatenation und Vereinigung!

Wir zeigen: L kann konstruiert werden aus drei kontextfreien Sprachen L_1, L_2, L_3

$$L = L_1(L_2 \bigcup L_3)$$

$$\begin{aligned} L_1 &= \{d^k \,;\, k \in \mathbb{N}\}\,, \\ L_2 &= \{e^m f^n \,;\, m, n \in \mathbb{N}, m < n\}\,, \\ L_3 &= \{e^m f^n \,;\, m, n \in \mathbb{N}, m > n\}\,. \end{aligned}$$

Beliebige Anzahl d‘s

Weniger e‘s als f‘s

Mehr e‘s als f‘s

Aufgabe 2

Gegeben seien $\Sigma = \{d, e, f\}$ und $L = \{d^k e^m f^n \,;\, k, m, n \in \mathbb{N}, m \neq n\}$.
Zeigen Sie, dass die Sprache L kontextfrei ist.

Wir zeigen: L kann konstruiert werden aus drei kontextfreien Sprachen L_1, L_2, L_3

$$L = L_1(L_2 \bigcup L_3)$$

$$L_1 = \{d^k \,;\, k \in \mathbb{N}\}\,,$$ Beliebige Anzahl d‘s

$$L_2 = \{e^m f^n \,;\, m, n \in \mathbb{N}, m < n\}\,,$$ Weniger e‘s als f‘s

$$L_3 = \{e^m f^n \,;\, m, n \in \mathbb{N}, m > n\}\,.$$ Mehr e‘s als f‘s

Wieso sind diese 3 Sprachen kontextfrei?

L_1 ist wegen $L_1 = dd^*$ regulär, also auch kontextfrei.

L_2 wird erzeugt durch die kontextfreie Grammatik G_2 mit den Produktionen

$$S \rightarrow eSf \mid T,$$
$$T \rightarrow Tf \mid f.$$

Am Schluss erzeuge ein oder mehr f zusätzlich...

L_3 wie L_2...

Also auch L kontextfrei!

Aufgabe 3

Überführen Sie die folgende Grammatik in Greibach-Normalform:

$$G = (\{S, X\}, \{a, b\}, \{S \to XX, S \to a, X \to SS, X \to b\}, X)\,.$$

Form dieser Grammatik?

Gute Form für CYK Algorithmus

Definition 94

Eine kontextfreie Grammatik G ist in Chomsky-Normalform, falls alle Produktionen eine der Formen

$$\begin{array}{ll} A \to a & A \in V, a \in \Sigma, \\ A \to BC & A, B, C \in V, \text{ oder} \\ S \to \epsilon & \end{array}$$

haben.

In jedem Ableitungsschritt entsteht ein Terminalsymbol!

Gut zum Umformen von Grammatik in NPDAs!

Ziel:

Definition 108

Sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik. G ist in Greibach-Normalform (benannt nach Sheila Greibach (UCLA)), falls jede Produktion $\neq S \to \epsilon$ von der Form

$$A \to a\alpha \text{ mit } a \in \Sigma,\ \alpha \in V^*$$

ist.

Von Chomsky zu Greibach:

for $k = 1, \ldots, m$ **do**
  for $j = 1, \ldots, k-1$ **do**
    for all $(A_k \to A_j\alpha) \in P$ **do**
      ersetze die Produktion gemäß der Konstruktion in Lemma 109
    od
  od
  co die rechte Seite keiner A_k-Produktion beginnt nun noch mit einer Variablen A_j, $j < k$ **oc**

  ersetze alle linksrekursiven A_k-Produktionen gemäß der Konstruktion in Lemma 110
  co die rechte Seite keiner A_k-Produktion beginnt nun noch mit einer Variablen A_j, $j \leq k$ **oc**
od

Lemma 109
Sei $G = (V, \Sigma, P, S)$ kontextfrei, $(A \to \alpha_1 B \alpha_2) \in P$, und sei $B \to \beta_1 | \beta_2 | \ldots | \beta_r$ die Menge der B-Produktionen (also die Menge der Produktionen mit B auf der linken Seite). Ersetzt man $A \to \alpha_1 B \alpha_2$ durch $A \to \alpha_1\beta_1\alpha_2 | \alpha_1\beta_2\alpha_2 | \ldots | \alpha_1\beta_r\alpha_2$, so ändert sich die von der Grammatik erzeugte Sprache nicht.

Definition 101
$A \in V$ heißt nutzlos, falls es keine Ableitung

$$S \to^* w, \quad w \in \Sigma^*$$

gibt, in der A vorkommt.

Lemma 110
Sei $G = (V, \Sigma, P, S)$ kontextfrei, sei $A \to A\alpha_1 | A\alpha_2 | \ldots | A\alpha_r$ die Menge der linksrekursiven A-Produktionen (alle $\alpha_i \neq \epsilon$, die Produktion $A \to A$ kommt o.B.d.A. nicht vor), und seien $A \to \beta_1 | \beta_2 | \ldots | \beta_s$ die restlichen A-Produktionen (ebenfalls alle $\beta_i \neq \epsilon$).
Ersetzen wir alle A-Produktionen durch

$$A \to \beta_1 | \ldots | \beta_s | \beta_1 A' | \ldots | \beta_s A'$$
$$A' \to \alpha_1 | \ldots | \alpha_r | \alpha_1 A' | \ldots | \alpha_r A',$$

wobei A' ein neues Nichtterminal ist, so ändert sich die Sprache nicht, und die neue Grammatik enthält keine linksrekursive A-Produktion mehr.

Aufgabe 3

Überführen Sie die folgende Grammatik in Greibach-Normalform:

$$G = (\{S, X\}, \{a, b\}, \{S \to XX, S \to a, X \to SS, X \to b\}, X)\,.$$

for $k = 1, \ldots, m$ **do**
  for $j = 1, \ldots, k-1$ **do**
    for all $(A_k \to A_j\alpha) \in P$ **do**
      ersetze die Produktion gemäß der Konstruktion in Lemma 109
    od
  od
  co die rechte Seite keiner A_k-Produktion beginnt nun noch mit einer Variablen A_j, $j < k$ **oc**

  ersetze alle linksrekursiven A_k-Produktionen gemäß der Konstruktion in Lemma 110
  co die rechte Seite keiner A_k-Produktion beginnt nun noch mit einer Variablen A_j, $j \leq k$ **oc**
od

Plan:

Im ersten Schritt der Konstruktion wird eine Schleife von $k = 1, 2, \ldots, m$ durchlaufen, die sicherstellt, dass für alle Produktionen der Form $A_i \to A_j\alpha$ mit $1 \leq i, j \leq m$ stets $i < j$ gilt. Damit gilt dann bereits, dass jede A_m-Produktion mit einem Terminalzeichen beginnt. Im zweiten Schritt der Konstruktion wenden wir das Lemma der Vorlesung beginnend mit A_{m_0-1} genügend oft an, um zu erreichen, dass jede Produktion mit einem Terminalzeichen beginnt.

Aufgabe 3

Überführen Sie die folgende Grammatik in Greibach-Normalform:

$$G = (\{S, X\}, \{a, b\}, \{S \to XX, S \to a, X \to SS, X \to b\}, X)\,.$$

for $k = 1, \ldots, m$ **do**
  for $j = 1, \ldots, k-1$ **do**
    for all $(A_k \to A_j\alpha) \in P$ **do**
      ersetze die Produktion gemäß der Konstruktion in Lemma 109
    od
  od
  co die rechte Seite keiner A_k-Produktion beginnt nun noch mit einer Variablen A_j, $j < k$ **oc**

  ersetze alle linksrekursiven A_k-Produktionen gemäß der Konstruktion in Lemma 110
  co die rechte Seite keiner A_k-Produktion beginnt nun noch mit einer Variablen A_j, $j \leq k$ **oc**
od

Vorab nummerieren wir die Variablen, indem wir sie (willkürlich) als A_1, A_2 umbezeichnen. Die Variable S soll jetzt dem A_1 und X dem A_2 entsprechen. Eventuell notwendig werdende zusätzliche Variablen werden dann mit $A_3, \ldots$ bezeichnet. Die Anzahl der Variablen ist also zunächst $m = m_0 = 2$. Der Wert von m wird dann nach Bedarf erhöht.

A_1 => A_2 A_2

A_1 => a

A_2 => A_1 A_1

A_2 => b

Aufgabe 3

Überführen Sie die folgende Grammatik in Greibach-Normalform:

$$G = (\{S, X\}, \{a, b\}, \{S \to XX, S \to a, X \to SS, X \to b\}, X)\,.$$

A_1 => A_2 A_2 | a

A_2 => A_1 A_1 | b

for $k = 1, \ldots, m$ **do**
  for $j = 1, \ldots, k-1$ **do**
    for all $(A_k \to A_j\alpha) \in P$ **do**
      ersetze die Produktion gemäß der Konstruktion in Lemma 109
    od
  od
  co die rechte Seite keiner A_k-Produktion beginnt nun noch mit einer Variablen A_j, $j < k$ **oc**

  ersetze alle linksrekursiven A_k-Produktionen gemäß der Konstruktion in Lemma 110
  co die rechte Seite keiner A_k-Produktion beginnt nun noch mit einer Variablen A_j, $j \leq k$ **oc**
od

Die A_1-Produktionen sind alle in der gewünschten Form, d. h. für $k = 1$ ist nichts zu tun und wir gehen sofort zu $k = 2$ über.
Die A_2-Produktionen sind nicht alle in der gewünschten Form. Die erste der beiden Produktionen muss gemäss der Vorlesung ersetzt werden durch alle Produktionen, die man durch Anwendung der A_1-Produktionen auf die erste A_1 Variable in $A_2 \to A_1A_1$ erhält.

A_1 => A_2 A_2 | a

A_2 => A_2 A_2 A_1 | a A_1 | b

Aufgabe 3

Überführen Sie die folgende Grammatik in Greibach-Normalform:

$$G = (\{S, X\}, \{a, b\}, \{S \to XX, S \to a, X \to SS, X \to b\}, X)\,.$$

for $k = 1, \ldots, m$ **do**
  for $j = 1, \ldots, k-1$ **do**
    for all $(A_k \to A_j\alpha) \in P$ **do**
      ersetze die Produktion gemäß der Konstruktion in Lemma 109
    od
  od
  co die rechte Seite keiner A_k-Produktion beginnt nun noch mit einer Variablen A_j, $j < k$ **oc**

  ersetze alle linksrekursiven A_k-Produktionen gemäß der Konstruktion in Lemma 110
  co die rechte Seite keiner A_k-Produktion beginnt nun noch mit einer Variablen A_j, $j \leq k$ **oc**
od

A_1 => A_2 A_2 | a
A_2 => A_2 A_2 A_1 | a A_1 | b

Nun ist die linksrekursive A_2-Produktion mit dem Vorlesungslemma zu eliminieren. Um zu verdeutlichen, wie dies geschieht, führen wir die entsprechenden Bezeichnungen ein, also $\alpha_1 = A_2A_1$ und $\beta_1 = aA_1, \beta_2 = b$. Die laut Lemma neu einzuführende Variable B bezeichnen wir mit A_3, wobei wir m inkrementieren, also $m = 3$ setzen. Die Linksrekursion ist nun zu ersetzen durch

A_2 => β_1 A_3 | β_2 A_3
A_3 => α_1 | α_1 A_3

Als Ergebnis für $k = 2$ erhalten wir

A_1-Produktionen:	$A_1 \to$	$A_2A_2 \mid a$
A_2-Produktionen:	$A_2 \to$	$aA_1A_3 \mid bA_3 \mid aA_1 \mid b$
	$A_3 \to$	$A_2A_1 \mid A_2A_1A_3$

Aufgabe 3

Überführen Sie die folgende Grammatik in Greibach-Normalform:

$$G = (\{S, X\}, \{a, b\}, \{S \to XX, S \to a, X \to SS, X \to b\}, X)\,.$$

for $k = 1, \ldots, m$ **do**
  for $j = 1, \ldots, k-1$ **do**
    for all $(A_k \to A_j\alpha) \in P$ **do**
      ersetze die Produktion gemäß der Konstruktion in Lemma 109
    od
  od
  co die rechte Seite keiner A_k-Produktion beginnt nun noch mit einer Variablen A_j, $j < k$ **oc**

lle linksrekursiven A_k-Produktionen gemäß der tion in Lemma 110
chte Seite keiner A_k-Produktion beginnt nun nit einer Variablen A_j, $j \leq k$ **oc**

Als Ergebnis für $k = 2$ erhalten wir

$$\begin{array}{lrl}
A_1\text{-Produktionen:} & A_1 & \to \quad A_2A_2 \mid a \\
A_2\text{-Produktionen:} & A_2 & \to \quad aA_1A_3 \mid bA_3 \mid aA_1 \mid b \\
 & A_3 & \to \quad A_2A_1 \mid A_2A_1A_3
\end{array}$$

Nun setzen wir $k = 3$. Die A_3-Produktionen sind nicht in der gewünschten Form. Die zwei Produktionen müssen nach dem Lemma ersetzt werden durch die Menge der Produktionen, die man durch Anwendung der A_2-Produktionen auf die erste Variable in den A_3-Produktionen erhält. Dies ergibt für $k = 3$ die neue Produktionenmenge

$$\begin{array}{lrl}
A_1\text{-Produktionen:} & A_1 & \to \quad A_2A_2 \mid a \\
A_2\text{-Produktionen:} & A_2 & \to \quad aA_1A_3 \mid bA_3 \mid aA_1 \mid b \\
A_3\text{-Produktionen:} & A_3 & \to \quad aA_1A_3A_1 \mid bA_3A_1 \mid aA_1A_1 \mid bA_1 \mid \\
 & & \quad\; aA_1A_3A_1A_3 \mid bA_3A_1A_3 \mid aA_1A_1A_3 \mid bA_1A_3
\end{array}$$

Man beachte, dass für die neuen Variablen, wie z. B. A_3, keine linksrekursiven Produktionen vorhanden sein können. Deshalb sind wir bereits mit dem ersten Schritt fertig.

Aufgabe 3

Überführen Sie die folgende Grammatik in Greibach-Normalform:

$$G = (\{S, X\}, \{a, b\}, \{S \to XX, S \to a, X \to SS, X \to b\}, X)\,.$$

$$\begin{array}{lll}
A_1\text{-Produktionen:} & A_1 \to & A_2A_2 \mid a \\
A_2\text{-Produktionen:} & A_2 \to & aA_1A_3 \mid bA_3 \mid aA_1 \mid b \\
A_3\text{-Produktionen:} & A_3 \to & aA_1A_3A_1 \mid bA_3A_1 \mid aA_1A_1 \mid bA_1 \mid \\
 & & aA_1A_3A_1A_3 \mid bA_3A_1A_3 \mid aA_1A_1A_3 \mid bA_1A_3
\end{array}$$

2. Schritt:

Man beachte, dass alle A_k-Produktionen mit $k \geq m_0$ notwendigerweise bereits mit einem Terminalzeichen beginnen. Es bleibt in unserem Fall also lediglich die Anwendung vom Vorlesungslemma auf die A_1 Produktionen, hier also auf $A_1 \to A_2A_2$, wobei alle A_2-Produktionen auf das erste A_2-Vorkommen angewendet werden müssen. Wir erhalten als Menge der A_1-Produktionen

$$A_1\text{-Produktionen:} \quad A_1 \to aA_1A_3A_2 \mid bA_3A_2 \mid aA_1A_2 \mid bA_2 \mid a$$

Diese bilden zusammen mit den A_2- und A_3-Produktionen aus dem 1. Schritt das Endergebnis.

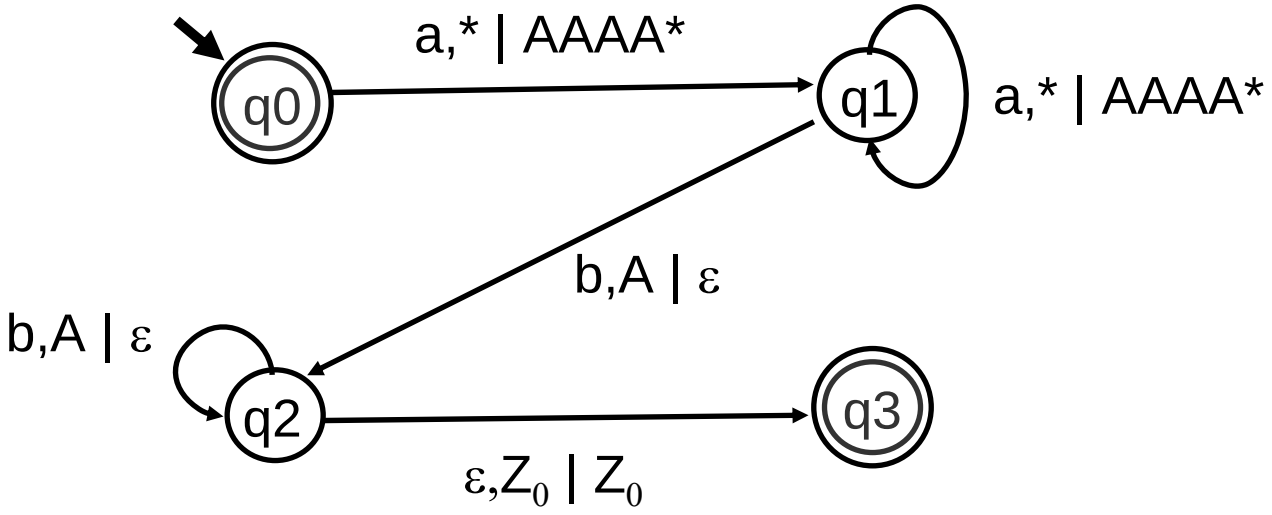
Aufgabe 4

Konstruieren Sie für die folgende Sprache L einen Kellerautomaten, der L durch Endzustände akzeptiert.

$$L = \{a^n b^{4n} \,;\, n \in \mathbb{N}_0\} \,.$$

Idee: für jedes a müssen später 4 b‘s kommen!

(* = egal was auf Stack, Z_0 = Stackboden)



Aufgabe 4

Konstruieren Sie für die folgende Sprache L einen Kellerautomaten, der L durch Endzustände akzeptiert.

$$L = \{a^n b^{4n} \,;\, n \in \mathbb{N}_0\} \,.$$

Beispiel aabbbbbbbb ∈ L :

Stack Z_0

Lese „a“: Stack AAAAZ_0

Lese „a“: Stack AAAAAAAAZ_0

Lese „b“: Stack AAAAAAAZ_0

Lese „b“: Stack AAAAAAZ_0

Lese „b“: Stack AAAAAZ_0

Lese „b“: Stack AAAAZ_0

Lese „b“: Stack AAAZ_0

Lese „b“: Stack AAZ_0

Lese „b“: Stack AZ_0

Lese „b“: Stack Z_0 => Fertig!

Aufgabe 4

Konstruieren Sie für die folgende Sprache L einen Kellerautomaten, der L durch Endzustände akzeptiert.

$$L = \{a^n b^{4n} \,;\, n \in \mathbb{N}_0\} \,.$$

Idee: für jedes a müssen später 4 b‘s kommen!

Der Kellerautomat $M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{A, B, Z_0\}, \delta, q_0, Z_0, \{q_0, q_3\})$, wobei

$$\begin{aligned}
\delta(q_0, a, *) &= \{(q_1, AAAA*)\} \,, \\
\delta(q_1, a, *) &= \{(q_1, AAAA*)\} \,, \\
\delta(q_1, b, A) &= \{(q_2, \epsilon)\} \,, \\
\delta(q_2, b, A) &= \{(q_2, \epsilon)\} \,, \\
\delta(q_2, \epsilon, Z_0) &= \{(q_3, Z_0)\} \,,
\end{aligned}$$

akzeptiert L durch Endzustände q_0 (leeres Wort) und q_3.