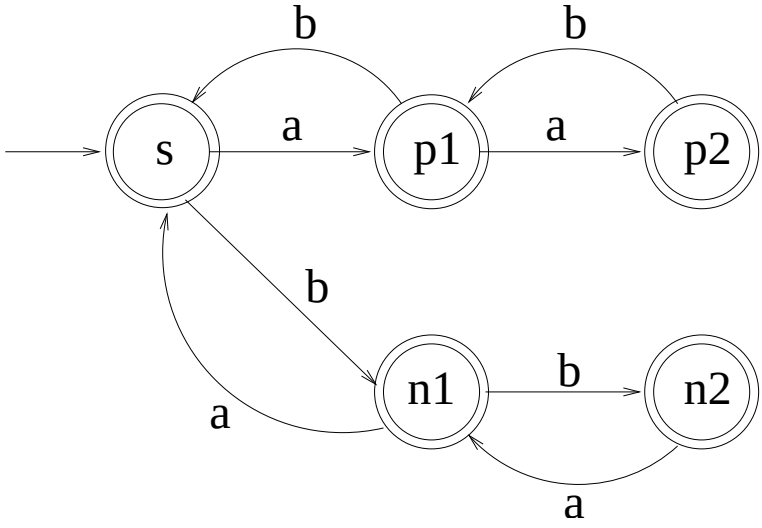
Theoretische Informatik (WS04)

Prof. Dr. Raimund Seidel
Wei Ding, Dierk Johannes
Lösungsvorschlag zu Aufgabenblatt 3

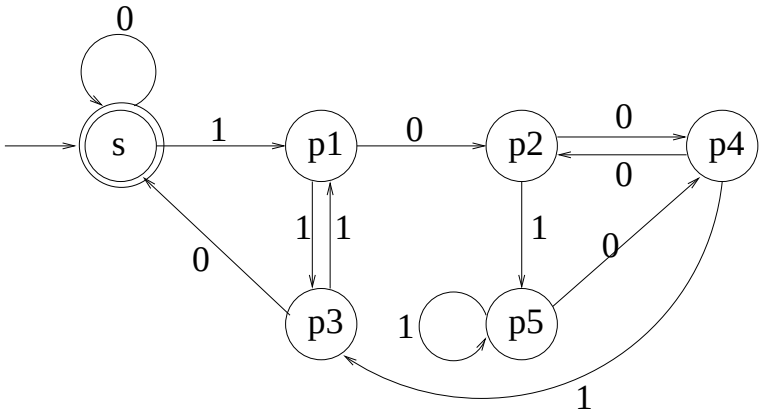
Aufgabe 1

- (a) nicht-reguär, also mit Pumping-Lemma wiederlegbar:
  Um zu zeigen ,daß L_a nicht reguär muss für jede Zahl $n \geq 1$ gezeigt werden können, daß für das Wort
  $x = a^n b^n \in L_a$ für jede Zerlegung $x = uvw$ mit
  $v = a^k$ wegen Bed. 2
  gilt, daß es eine Zahl i gibt, so daß
  $uv^i w = a^{n+k\times(i-1)}b^n \notin L_a$ Da wir mit $i = 0$ bereits eine solche Zahl gefunden haben, kann L_a nicht regulär sein.

- (b) reguär, also mit Automat zu zeigen:



- (c) nicht-regulär, also mit Pumping-Lemma wiederlegbar:
  Wieder wählen wir ein n frei $n \geq 1$ zeigen, daß für das Wort
  $x = a^n b^{2n} \in L_c$ jede Zerlegung $x = uvw$ mit
  $v = a^k$ wegen Bed. 2
  gilt, daß es eine Zahl i gibt, so daß
  $uv^i w = a^{n+k\times(i-1)}b^{2n} \notin L_c$ Auch hier ist i=0 wieder ein solcher Kandidat, also ist L_c nicht regulär. (d) regulär, also mit Automat zu zeigen:



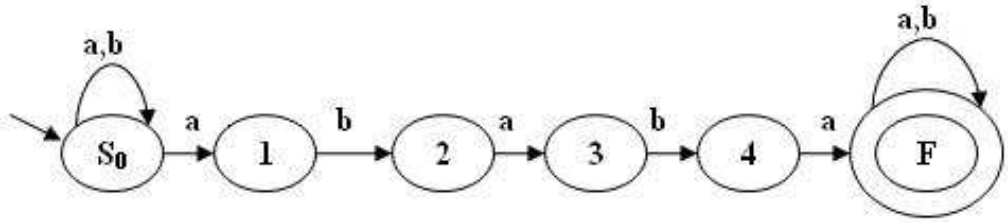
Aufgabe 2

a) Wir bauen einen NEA, der die Sprache L=$\{w \in \{a,b\}^* \mid$ w enthält ababa als Teilwort$\}$ akzeptiert, also L:=(a|b)* ababa (a|b)*

NEA: $M = (Q, \Sigma, \Delta, S_0, F)$

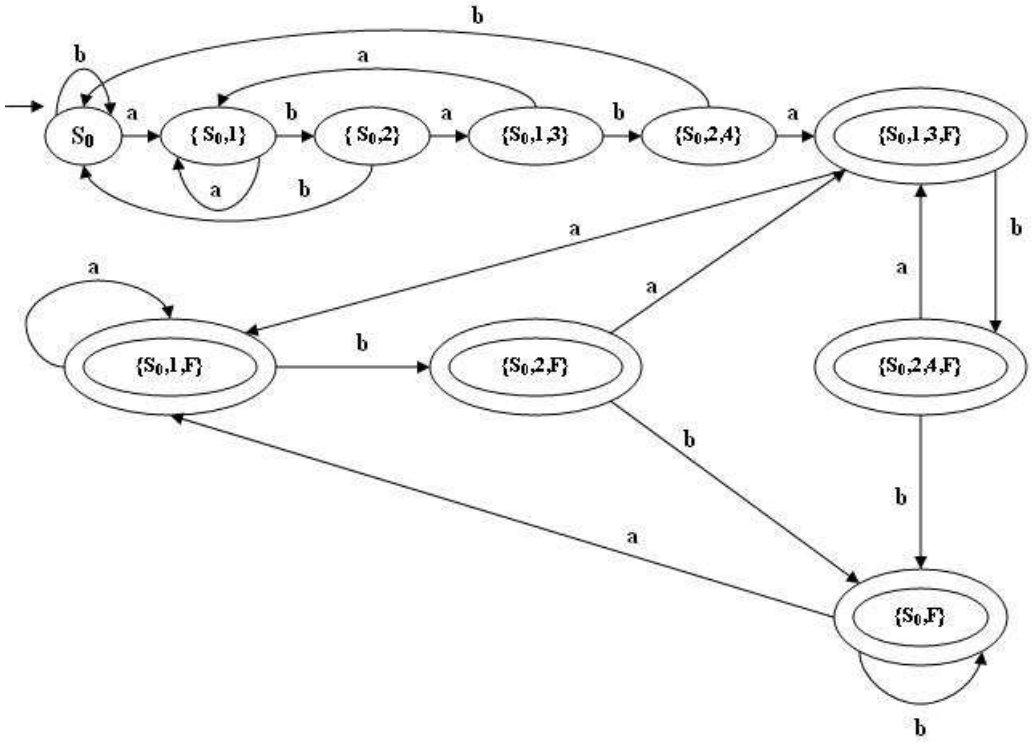
$\Sigma = \{a, b\}$
S_0 - Startzustand
F - Endzustand
$Q = F \cup S_0 \cup \{1, 2, 3, 4\}$

$\Delta = \{(S_0, a, S_0), (S_0, b, S_0), (S_0, a, 1),$
$(1, b, 2), (2, a, 3), (3, b, 4),$
$(4, a, F), (F, a, F), (F, b, F)\}$



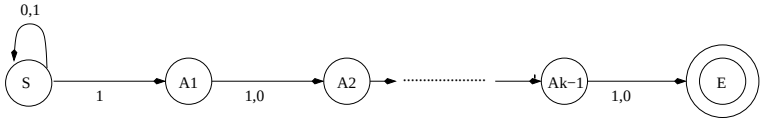
b) Sei M ein NEA aus (a). Wir können einen DEA: $M' = (Q', \Sigma, \Delta', S_0', F')$ nach der Methode,

die in der Vorlesung gezeigt wurde, bauen. DEA M' akzeptiert L(M), wobei:
$Q' \subseteq Q$,
$\Sigma = \{a, b\}$,
$S'_0 = S_0$,
$F' = \{\{S_0, 1, 3, F\}, \{S_0, 1, F\}, \{S_0, 2, F\}, \{S_0, 2, 4, F\}, \{S_0, F\}\}$
$Q' = S_0 \cup F' \cup \{\{S_0, 1\}, \{S_0, 2\}, \{S_0, 1, 3\}, \{S_0, 2, 4\}\}$
$\Delta' = \{q \in Q | \exists q' \in Q' : (q', a, q) \in \Delta\}$



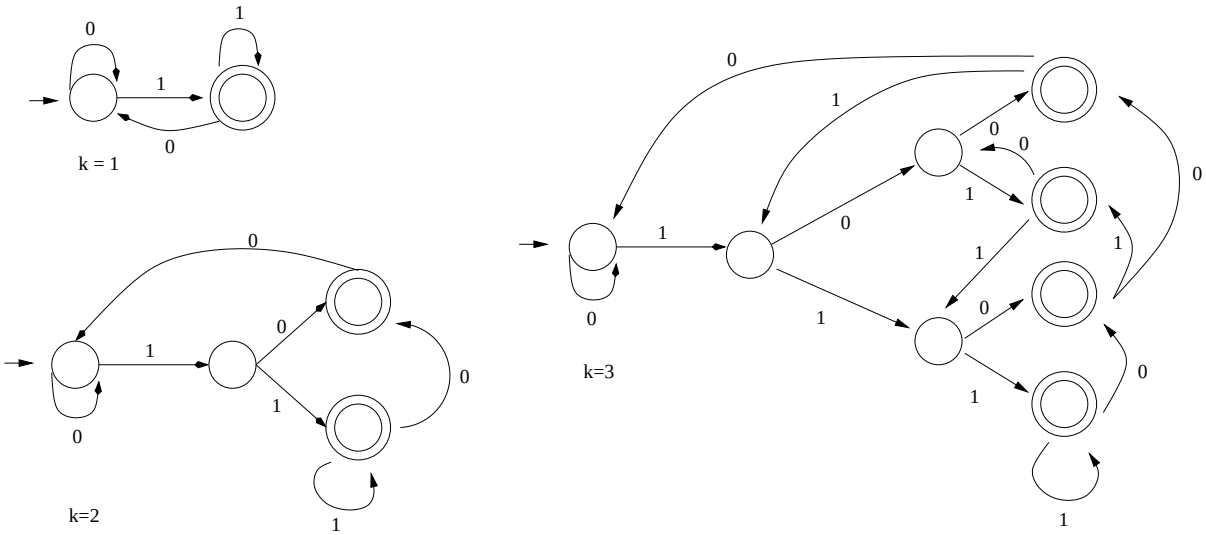
Aufgabe 3

1. Ein NEA braucht genau k+1 Zustände, um diese Sprache zu entscheiden. Er hat einen Startzustand, in dem er bleibt, bis er die k-t letzte Eins erreicht hat. Zusätzlich benötigt er noch k-1 viele Zustände, mit denen er verifiziert, dass dies auch die k-te Stelle war.



2. Ein DEA hingegen kann die k-t letzte Stelle nicht raten, sondern muss für jede Eins, die er auf dem Band liest, testen, ob es die richtige Stelle war. Er verzweigt also in eine

baumartige Struktur der Tiefe k und braucht deshalb 2^k Zustände. Man muss jetzt noch zeigen, dass dieser Automat minimal ist. Ein Anhaltspunkt dafür ist seine Baumstruktur. Zur Veranschaulichung betrachten wir hier den Fall $k = 3$. Die Resultate lassen sich dann leicht auf den allgemeinen Fall übertragen.



Zum formalen Beweis betrachten wir die Anzahl der Fortsetzungssprachen von L. Geben wir uns ein beliebiges Wort w vor, dann ist $L(w) = \{x \mid wx \in L\}$. Zunächst stellen wir fest, dass für jedes Wort x, das aus mindestens 3 Zeichen besteht, wx genau dann in L liegt, wenn schon x zu L gehört. Damit ist L in *jeder* Fortsetzungssprache enthalten. Wir müssen also lediglich noch untersuchen, welche Worte mit höchstens 2 Zeichen ein Wort aus Σ^* zu einem Wort aus L ergänzen. Dazu geben wir uns ein beliebiges Wort w und unterscheiden verschiedene Fälle in Abhängigkeit der letzten 3 Zeichen von w.

w	Fortsetzungssprache
$\|w\| = 0$	L
$\|w\| = 1$	
0	L
1	$\Sigma^2 \cup L$
$\|w\| = 2$	
00	L
01	$\Sigma^2 \cup L$
10	$\Sigma^1 \cup L$
11	$\Sigma^1 \cup \Sigma^2 \cup L$
$\|w\| \geq 3$	
$* 000$	L
$* 001$	$\Sigma^2 \cup L$
$* 010$	$\Sigma^1 \cup L$
$* 011$	$\Sigma^1 \cup \Sigma^2 \cup L$
$* 100$	$L \cup \{\varepsilon\}$
$* 101$	$\Sigma^2 \cup L \cup \{\varepsilon\}$
$* 110$	$\Sigma^1 \cup L \cup \{\varepsilon\}$
$* 111$	$\Sigma^1 \cup \Sigma^2 \cup L \cup \{\varepsilon\}$

Damit haben wir sämtliche Fälle betrachtet und können feststellen, dass es $8 = 2^3$ verschienden Fortsetzungssprachen gibt. Wenn wir das Ergebnis verallgemeinern, erhalten wir: Für $k \in \mathbb{N}$ kann es keinen DEA mit weniger als 2^k Zuständen geben, der die in der Aufgabe angegebene Sprache akzeptiert.

Aufgabe 4

Bei der in der Vorlesung vorgestellten Methode wird eine einzelne ϵ-Kante entfernt, ohne daß sich die Sprache, die der Automat entscheidet, dadurch verändert. Da es in einem Automaten nur endlich viele ϵ-Kanten geben kann (Maximal $|Q|^2$ viele), erscheint es plausibel anzunehmen, daß man durch iteratives anwenden dieser Methode alle ϵ-Kanten entfernen kann. Diese Annahme beruht insbesondere darauf, daß man in einem einzelnen Iterationsschritt die Zahl der Kanten von $0 < n \leq |Q|^2$ auf $n - 1$ Kanten reduziert. Genau dieser Teil ist jedoch falsch, da der Algorithmus auch neue ϵ-Kanten erzeugen kann: Die Regel besagt, falls $(B, \epsilon, C) \in \Delta$), definiere $\Delta' = \Delta \setminus (B, \epsilon, C) \cup \{(A, s, C) | \exists (A, s, B) \in \Delta\} \cup \{(B, s, D) | \exists (C, s, D) \in \Delta\}$ Insbesondere kann $s = \epsilon$ gelten!
Das Problem tritt immer dann auf, wenn man Pfade von ϵ-Kanten der Länge 2 oder länger hat. Eine Abwandlung des Algorithmus setzt an genau dieser Stelle an: Primärziel soll es nicht mehr sein, die Zahl der Kanten zu verringern (das ist auf Dauer ein Nebeneffekt), sondern die längsten ϵ-Pfade zu verkürzen. Beachte: Hat der längste Pfad Länge 1, entspricht es obiger Methode, d.h. insbesondere, daß die Kante entfernt wird und keine neuen entstehen.

Die Verbesserung besteht darin, auf einem längstem Pfad von ϵ-Kanten alle Kanten, die im Anfangsknoten enden, als Duplikate auf alle Knoten des Pfades weiterzuleiten, ebenso werden alle Kanten, die den Endknoten verlassen auf diese Weise für jeden Knoten des Pfades dupliziert. Beachte: Handelt es sich um einen längsten Pfad, können die Duplizierten Kanten keine ϵ-Kanten sein. Nun war es im Original-Graph aber auch möglich, innerhalb des Pfades ϵ-Kanten zu nutzen. Deshalb müssen auch alle anderen Kanten, die auf den Pfad führen, für alle nachfolgenden Knoten des Pfades dupliziert werden, analog müssen alle Kanten, die von einem inneren Knoten abzweigen für alle vorangehenden Knoten des Pfadesdupliziert werden. Dabei können nun wieder ϵ-Kanten entstehen, ein längster Pfad verschwindet jedoch, amn erzeugt höchstens kleinere Pfade.
Auch dieser Algorithmus hat wieder eine Schwachstelle: Er funktioniert nicht für unendlich lange Pfade, wie sie in Form eines ϵ-Zyklus' vorkommen können. Solche Zyklen sind jedoch leicht zu eliminieren: Alle Knoten auf dem Zyklus sind äquivalent, alle diese Knoten können zu einem einzigen zusammengefasst werden, wobei die beteiligten ϵ-Kanten entfernt werden, alle Kanten, die vorher in den Zyklus geführt haben nun zu dem neuen Knoten führen und alle Kanten, die vorher aus dem Zyklus heraus führten, nun in dem neuen Knoten starten. Non-ϵ-Kanten, die vorher 2 Knoten innerhalb des Zyklus verbanden, werden nun durch Kombination der 2 vorherigen Fälle zu Selbstkanten des neuen Knoten. Beachte, daß ϵ-Zyklen der Länge 1 keinen Sonderfall darstellen, sondern auch auf diesem Weg entfernt werden können.