VORLESUNGSSCRIPT THEORETISCHE INFORMATIK I
Vom 23.06.2000

Rückblick: Nerode-Äquivalenz

Die Nerode - Äquivalenz besagt: Zwei Wörter *u* und *v* sind äquivalent, wenn es einen Akzeptoren *A* gibt, der nach dem Lesen von *v* im gleichen Zustand ist wie nach dem Lesen *u*. Ist dieser erreichte Zustand dann auch noch ein Endzustand, dann ist sowohl $u \in L(A)$ als auch $v \in L(A)$.
Wenn man nun prüft, wie der Akzeptor auf die Fortsetzung von eingegebenen Wörtern reagiert, kann man feststellen, dass es Sequenzen in den Wörtern gibt, die die funktionelle Bedeutung des Gesamtwortes nicht beeinflussen, wenn man sie entfernt. Bei dieser Fortsetzung der Wörter stellt man dann fest, dass es Zustände im Akzeptor geben kann, die miteinander verschmolzen werden können. Dies führt auf den Begriff der Äquivalenzklassen, die man wie folgt definiert:

$$[u] = \{v \in X^* \mid u \equiv_L v\} \text{ heißt: } u \equiv_L v \Leftrightarrow \forall w \in X^* (uw \in L \Leftrightarrow vw \in L).$$

Zur Verdeutlichung seien nun Beispiele aufgeführt:

Beispiel 1

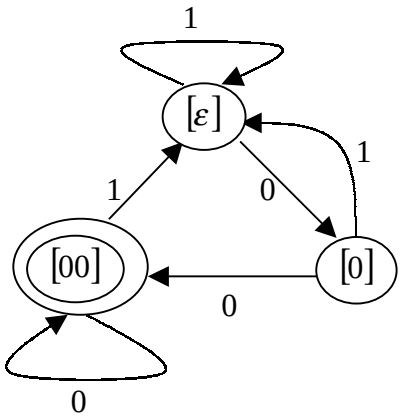
$L = \{x \in \{0,1\}^* \mid \text{x endet auf } 00\}$

offenbar gilt:

$$\left.\begin{array}{l} [\varepsilon] = \{w \in \{0,1\}^* \mid \text{w endet nicht auf } 0\} \\ [0] = \{w \in \{0,1\}^* \mid \text{w endet mit 0 aber nicht mit 00}\} \end{array}\right\} \notin L$$

$$[00] = \{w \in \{0,1\}^* \mid \text{w endet mit 00}\}$$

Dies sind die einzigen Äquivalenzklassen. Sie bilden die Zustände des minimalen Automaten.



Beispiel 2

$L = \{a^n b^n \mid n \geq 0\}$ und $X = \{a, b\}$ ist nicht regulär (*Klammersprache*)

Sprachen dieser Art sind nicht regulär (*wie man in einem späteren Beweis noch sehen wird*). Somit sind endliche Akzeptoren, wie wir sie bis jetzt benutzt haben, nicht in der Lage, solche Sprachen zu erkennen. Dies liegt in der Definition der Akzeptoren, da sie auf endlichen Zustandsmengen definiert sind. Der Exponent *n* der Sprache L durchläuft nun aber die Menge der natürlichen Zahlen $\mathbb{N}$, womit hier die Unendlichkeit ins Spiel kommt. Versucht man nun die Äquivanlenzklassen zu bestimmen, so erhält man folgendes:

$[\varepsilon] \neq [a]$, denn $\underbrace{\varepsilon}_{u}\underbrace{ab}_{w} \in L$, aber $\underbrace{a}_{v}\underbrace{ab}_{w} \notin L$

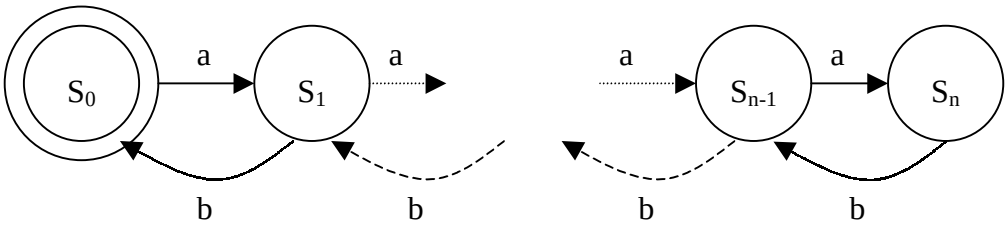
$[a] \neq [aa]$, denn $\underbrace{a}_{v}\underbrace{b}_{w} \in L$, aber $\underbrace{a}_{v}\underbrace{ab}_{w} \notin L$

$\vdots$

$[a^i] \neq [a^j]$, für $i \neq j$, denn $\underbrace{a^i}_{u}\underbrace{b^i}_{w} \in L$, aber $\underbrace{a^j}_{v}\underbrace{b^i}_{w} \notin L$

Ergebnis:
Wir erhalten unendlich viele verschiedene Äquivalenzklassen. Der Index von L ist somit nicht endlich – demnach ist diese Sprache nach Myhill - Nerode nicht regulär, da sie ja wie aus dem Versuch die Äquivalenzklassen zu bestimmen ersichtlich ist, nicht die Kriterien der Nerode – Äquivalenz erfüllt.

Auf einem endlichen Automaten (*PC*) sind diese Sprachen durch eine Beschränkung des *n* realisiert und werden somit zu Akzeptoren der Art:

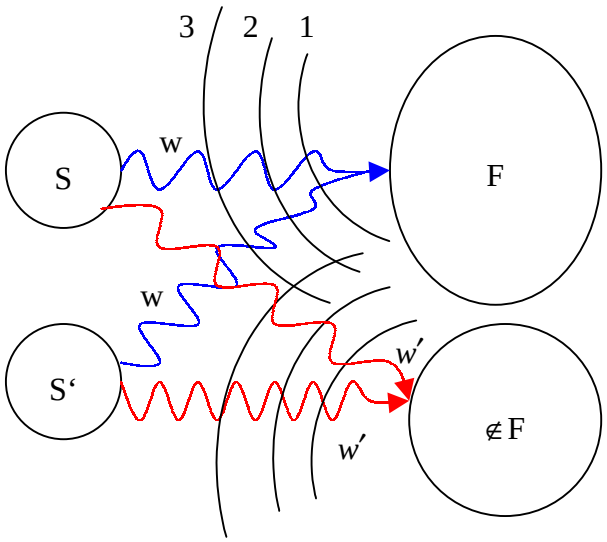


Für diese Sprachen benötigt man mächtigere Automaten, wenn man auf diese Beschränkung verzichten möchte. Man muß einen Speicher hinzufügen, z.B. einen Stack. Man bezeichnet diese Klasse der Automaten dann als Kellerautomaten.

4.5. Minimierung von Automaten

Myhill – Nerode: Ein Äquivalenzklassenautomat ist der Automat mit der kleinsten Menge von Zuständen (Minimalakzeptor).
Der Begriff des minimalen Automaten läßt sich durch folgende Vorstellung verdeutlichend darstellen. Für eine gegebene Sprache lassen sich mehrere, verschiedene (in Bezug auf die Anzahl der Zustände) Akzeptoren konstruieren. Hierbei kann man sich, zum Beispiel aus Effizienzgründen, fragen, ob es unter diesen Akzeptoren einen mit einer minimalen Anzahl an Zuständen gibt. Diese Frage führt auf den Akzeptoren mit der kleinst möglichen Zustandanzahl – den Minimalakzeptor. Dabei führt man folgende Überlegung an:
Für jeden beliebigen Automaten A', der die Sprache L erkennt gilt: $K_{L(A')} \subseteq K_L = K_{A_0}$, wobei A_0 ein Minimalakzeptor ist. Werden also zu einem Akzeptor die Äquivalenzklassen gebildet, so erhält man die Zustände des minimalen Automaten.
Ein Akzeptor A heißt also minimal, wenn es keinen weiteren Akzeptor mit weniger Zusänden gibt, der $L(A)$ akzeptiert.

Im folgenden soll nun Schritt für Schritt gezeigt werden, wie diese Minimierung von statten geht:



Fortsetzung der Wörter in die Endzustandsmengen

Man „hangelt“ sich in Schichten durch den Automaten und Prüft ob noch Äquivalenz besteht.

Sehr lange Wörter haben Zyklen. Diese können gefunden und weggelassen werden, so dass sich das Wort zwar verkürzt, jedoch keine Information verlorengeht.

Definition U

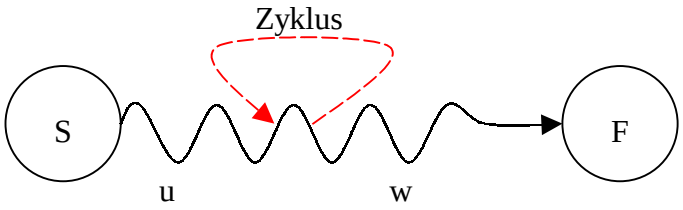
Sei $A=(X,S,\delta,s_0,F)$ ein endlicher Akzeptor. Zwei Zustände $s, s' \in S$ heißen äquivalent, wenn gilt: $\forall w \in X^* (\delta(s,w) \in F \Leftrightarrow \delta(s',w) \in F)$.

Bemerkung hierzu:

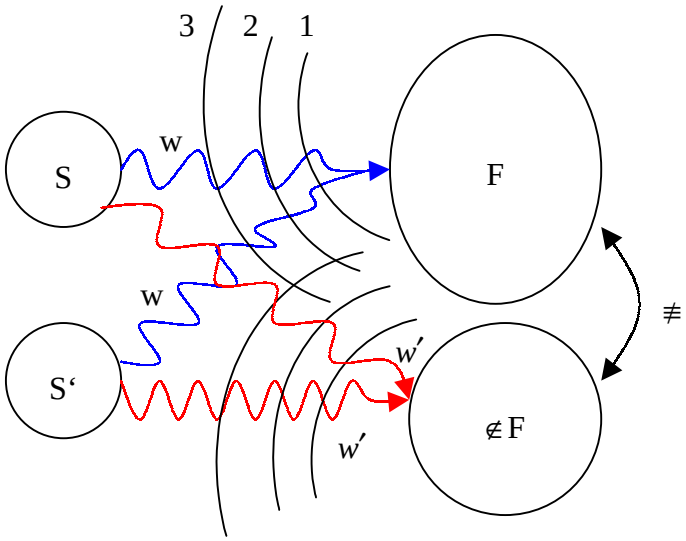
Wegen der Äquivalenz der Zustände s und s' kann man sie wegen des Satzes von Myhill – Nerode miteinander verschmelzen. Die beiden Zustände sind genau dann zueinander äquivalent, wenn s und s' akzeptieren oder wenn sie beide nicht akzeptieren – und wenn zwei Zustände äquivalent sind können sie verschmolzen werden. Der Akzeptor verkleinert sich also.

Offenbar kann man sich bei der Eingabe der Wörter w auf $|w| < |S|$ beschränken, da aus Wörtern der Länge $\leq |S|$ bereits alle Informationen entnommen werden können, da wie oben bereits besprochen durch das Entfernen der Zyklen kein Informationsverlust auftritt.

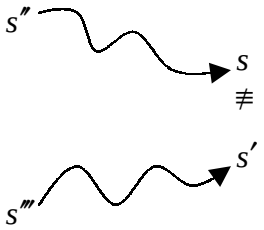
Beispiel



Es gilt offenbar: Wenn $s \in F \wedge s' \notin F$ dann ist s nicht äquivalent zu s‘. Die folgende Grafik soll dies noch einmal verdeutlichen:



Dann ist daraus folgendes ersichtlich:



Wenn also s und s' nicht äquivalent sind, dann können auch s'' und s''' nicht äquivalent gewesen sein.

Begründung: setze $w = \varepsilon$ in Definition U.

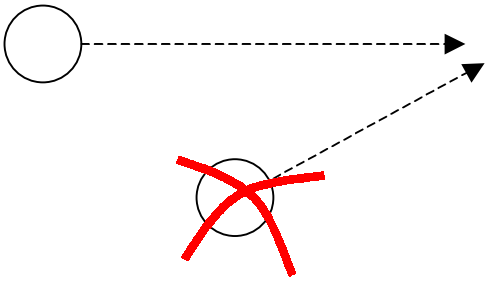
Nach dieser Einleitung sei nun das allgemeine Verfahren für so eine Minimierung angegeben:

Idee

Teste die Äquivalenz von Zuständen ausgehend von den Endzuständen mit fortlaufender inkrementeller Verlängerung der betrachteten Wörter.

Eingabe

deterministischer endlicher Akzeptor $A=(X,S,\delta,s_0,F)$ bei dem alle Zustände bereits entfernt sind, die vom Startzustand nicht erreichbar sind, also:



Ausgabe

Ausgabe, welche Zustände von A zu verschmelzen sind um den minimalen Automaten zu erhalten.

Es kann also nun ein Algorithmus zur Lösung dieses Problems angegeben werden.

i) *Bilden einer Tabelle aller Zustandspaare $\{s,s'\}$ mit $s \neq s'$ (evtl. {s,s'} zulassen).*

ii) *Markiere alle Paare mit $s \in F$ und $s' \notin F$ oder umgekehrt*

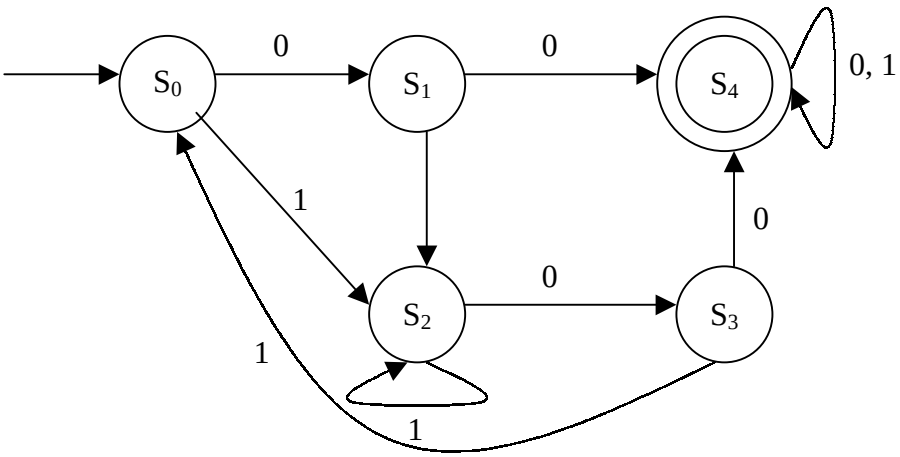
iii) *Für jedes noch unmarkierte Paar $\{s,s'\}$ und jedes $a \in X$ teste, ob die Paare $\{\delta(s,a),\delta(s',a)\}$ markiert sind, und wenn ja, dann markiere $\{s,s'\}$. Falls jedoch $\delta(s,a)=\delta(s',a)$ dann markiere in jedem Fall nicht!*

iv) *Wiederhole iii) bis die Tabelle keine Änderungen mehr ergibt.*

v) *Alle noch unmarkierten Paare können jeweils zu einem Zustand verschmolzen werden.*

Anmerkung zur Komplexität: Bei geeigneter Implementierung ist eine Komplexität von $O(|S|^2)$ erreichbar.

Beispiel



	S_0	S_1	S_2	S_3
S_0	*	X	X	X
S_1		*	X	X
S_2	*		*	X
S_3	*	*	*	*

Legende:
X nicht benutzt
* der eine ist aus F der andere nicht

Es ergeben sich nach Anwendung des oben aufgeführten Algorithmus folgende Markierungen:

$\{s_0, s_3\} \xrightarrow{0} \{s_1, s_4\}$ *
~~$\{s_0, s_3\} \xrightarrow{1} \{s_0, s_2\}$~~

$\{s_2, s_3\} \xrightarrow{0} \{s_3, s_4\}$ *
~~$\{s_2, s_3\} \xrightarrow{1} \{s_0, s_2\}$~~

$\{s_1, s_3\} \xrightarrow{0} \{s_4\}$
$\{s_1, s_3\} \xrightarrow{1} \{s_0, s_2\}$

$\{s_1, s_2\} \xrightarrow{0} \{s_3, s_4\}$ *
~~$\{s_1, s_2\} \xrightarrow{1} \{s_2\}$~~

$\{s_0, s_1\} \xrightarrow{0} \{s_1, s_4\}$ *
~~$\{s_0, s_1\} \xrightarrow{1} \{s_2\}$~~

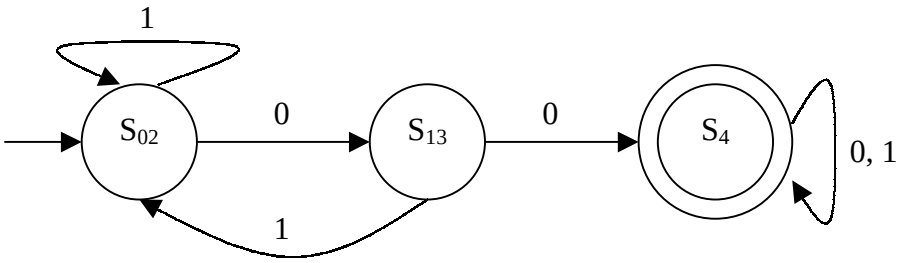
$\{s_0, s_2\} \xrightarrow{0} \{s_1, s_3\}$
$\{s_0, s_2\} \xrightarrow{1} \{s_2\}$

Betrachtet man die Markierungen weiter, so stellt man fest, dass die Paare:

$\{s_1, s_3\} \xrightarrow{0} \{s_4\}$
$\{s_1, s_3\} \xrightarrow{1} \{s_0, s_2\}$

$\{s_0, s_2\} \xrightarrow{0} \{s_1, s_3\}$
$\{s_0, s_2\} \xrightarrow{1} \{s_2\}$

noch nicht markiert sind. Somit können also $\{s_0, s_2\}$ und $\{s_1, s_3\}$ miteinander verschmolzen werden. Daraus erhält man direkt den minimalen Automaten:



Hierbei kann man dann auch gleich die Äquivalenzklassen feststellen, wobei sich ergibt:

$[\varepsilon] = s_{02}$
$[0] = s_{13}$
$[00] = s_4$

Womit der Index 3, also auch endlich ist und somit auch der Minimalautomat genau drei Zustände hat.

Satz V

Zu jedem endlichen deterministischen Akzeptoren A gibt es bis auf Isomorphie (im Sinne von identisch bis auf Benennung der Zustände) genau einen eindeutig bestimmbaren minimalen deterministischen Akzeptoren A' für den gilt $L(A) = L(A')$.

Beweis

An dieser Stelle sei nur ein Ansatz für den Beweisgedanken angegeben. Eine ausführliche Ausführung ist in *[Erk/ Priese: Theoretische Informatik, Satz 5.6.9, Seite 96]* zu finden.
Ansatz:

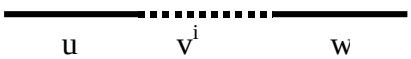
i) A ist minimal
ii) Sei A' ein minimaler Akzeptor, so ist A isomporph zu A'

Sei A minimal. Da A nach Voraussetzung erreichbar ist, gibt es einen sujektiven Automatenmorphismus $\varphi : A \to A'$.
Ferner: Angenommen, φ sei nicht injektiv, dann werden mindestens zwei Zustände von A auf den selben Zustand von A' abgebildet. Also ist $|K_A| > |K_{A'}|$. Nun ist aber A nach i) minimal, was einen Widerspruch ergibt. Somit ist φ bijektiv, also ist A isomorph zu A', in Zeichen: $A \cong A'$.

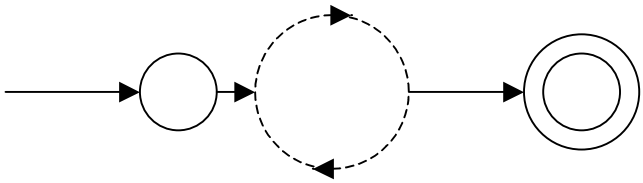
4.6. Das Pumping – Lemma

als eine typische Eigenschaft regulärer Sprachen

Reguläre Sprachen sind gerade diejenigen Sprachen, die man durch reguläre Ausdrücke beschreiben kann. Dadurch lassen sich Rückschlüsse auf die allgemeine Struktur dieser Sprachen ziehen, was folgende Grafik verdeutlichen soll:



Durch diese Teilung kann der Teil v^i des Wortes beliebig „aufgepumpt“ werden, so dass die Sprache trotzdem immer noch regulär ist. Dies zeigt sich dann in Form von Zyklen. Um solche Zyklen zu erzeugen, kann man in regulären Sprachen nur die *Kleene – Stern – Bildung* (*) benutzen. Es müssen sich also in unendlich langen Wörtern, die sich nur auf diese Weise erzeugen lassen, irgendwo bestimmte Zeichenketten wiederholen. Solche Zyklen stellt man sich einfach so vor:



Das Pumping – Lemma ist ein zentrales Beweismittel zum Nachweis, dass eine Sprache nicht regulär ist. Umgekehrt kann aber nicht gezeigt werden, dass eine Sprache regulär ist!

Satz W

Sei X ein Alphabet. Zu jeder regulären Sprache $R \subseteq X^*$ gibt es ein $n \in \mathbb{N}$, und für die Wörter $z \in R$ mit $|z| \geq n$ gilt: Es gibt eine Zerlegung $z = uvw$ mit $u, v, w \in X^*$ mit $v \neq \varepsilon$ und $|uv| \leq n$, so dass für alle $i \geq 0$ gilt: $uv^i w \in R$.

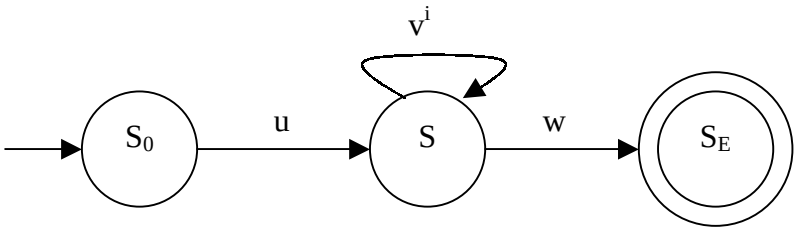
Rein formal ausgedrückt besitzt die Aussage diese Form:

$$\forall R \in \mathrm{Reg}\left(\exists n \in \mathbb{N}\left(\forall z \in R \text{ mit } |z| \geq n\left(\exists u, v, w \in X^*\left(z = uvw \wedge v \neq \varepsilon \wedge |uv| \leq n\left(\forall i \in \mathbb{N}_0\left(uvw \in R\right)\right)\right)\right)\right)\right)$$

Beweis

Ansatz: Es existiert in allen Wörtern $z \in R$ ab einer gewissen Länge eine Pumpstelle, das heißt nach einem beliebig langem Teilwort u folgt ein „Mittelwort“ v, welches beliebig oft durchlaufen werden oder auch entfallen kann.

i) Endlicher Akzeptor
Um unbeschränkte Wörter zu erkennen, muß ein endlicher Akzeptor Zyklen, auch mehrfach, durchlaufen!



ii) Reguläre Ausdrücke
Die einzige Möglichkeit in einem regulären Ausdruck Unendlichkeit zu erzeugen ist die Anwendung Kleene – Sternbildung (*). Folglich enthalten sehr lange Wörter viele Wiederholungen ein und des selben Teilwortes.

Für den Teil i) sei hier der Beweisansatz gegeben (*der komplette Beweis ist Thema der nächsten Vorlesung*).
Sei L eine reguläre Sprache, dann existiert ein Akzeptor $A = (X, S, \delta, s_0, F)$, der L akzeptiert. Man wähle für n die Anzahl der Zustände von A, also $n = |S|$. Sei nun

$x \in X^*$ ein beliebiges Wort mit einer Länge $\geq n$, das von A akzeptiert wird.
Beim Abarbeiten von x durchläuft A genau $|x|$, will man den Startzustand mitzählen, dann sind es $|x|+1$ Zustände.

Da nun aber gilt, das $|x| \geq |n|$ ist, muß A mindestens einen Zustand zweimal durchlaufen haben (Dirichlet'sches Schubfachprinzip). Mit anderen Worten: Der Akzeptor muß eine Schleife durchlaufen haben, es gibt also Zyklen.
Man wähle nun eine Zerlegung des Wortes derart: $x = uvw$, dass der Akzeptor nach Lesen des Teilwortes u und nach Lesen des Teilwortes uv im gleichen Zustand ist.
Hierbei muß dann aber Folgendes gewährleistet sein: $|v| \geq 1$ und $|uv| \leq n$ gemäß Definition.
Da nun die Zustände gleich sind, erreicht der Akzeptor nach dem Lesen von uw denselben Endzustand, wie nach dem Lesen des Wortes uvw. Dies heißt aber nichts anderes, als dass das Wort folgende Form haben kann: $uw = uv^0 w \in L$, oder $uvvw = uv^2 w \in L$, also letztendlich $uv^i w \in L$ für alle $i \in \mathbb{N}_0$. Ein endlicher Akzeptor durchläuft also zwangsläufig Zyklen, wenn er potentiell unendliche Wörter erkennt bzw. erkennen soll.

Anwendung des Pumping – Lemmas

Beispiel

Es sei die Sprache $L = \{a^k b^k | k \in \mathbb{N}\}$ gegeben. Es gibt das Wort $a^k b^k$ und es muß eine Zerlegung uvw wie folgt geben: $\underset{u\ v\ w}{a^k b^k}$ mit $v \neq \varepsilon$. Daraus ergibt sich die Behauptung:

L ist nicht regulär!
Beweis: Widerspruchsbeweis mit Hilfe des Pumping – Lemmas.
Annahme: L ist regulär!

Dann gibt es ein $n \in \mathbb{N}$, so dass wegen der Definition des Pumping – Lemmas gilt: $\forall z \in R \text{ mit } |z| \geq n(\exists u, v, w \in X^*(z = uvw \wedge v \neq \varepsilon \wedge |uv| \leq n(\forall i \in \mathbb{N}_0(uvw \in R))))$. Hier ist nun eine Fallunterscheidung in Bezug auf die möglichen Zerlegungen zu treffen:

Fall 1: es sei $u \in \{a\}^*$, $v \in \{a\}^*$ und $w \in \{b\}^*$
Beim Pumpen stellt ergibt sich offenbar, dass $uv^i w \notin L$ ist, da die Anzahl der a's (im Folgenden als $anz(a)$ bezeichnet) die der b's übersteigt, also $anz(a) > anz(b)$ ist. Dies entspricht jedoch nicht mehr der Sprachdefinition von L.

Fall 2: es sei nun $u \in \{a\}^*$, $v \in \{b\}^*$ und $w \in \{b\}^*$
Dann ergibt sich beim Pumpen analog zu Fall 1, dass die durch das Pumpen erzeugten Wörter nicht mehr in L sind, wegen $anz(a) < anz(b)$.

Fall 3: es sei jetzt $u \in \{a\}^*$, $w \in \{b\}^*$ und $v \in \{a, b\}^*$
Es ist offensichtlich, dass beim Pumpen eine Zerlegung entsteht, die nicht der Forderung des Pumping – Lemmas genügt, also für die gilt: $uv^i w \notin L$.
Daraus folgt also: $L \notin \text{Reg}$.