Universität Konstanz

Ergänzungen zur Vorlesung

Theoretische Grundlagen der Informatik

gelesen von: Prof. Dr. Dietmar Saupe
Textsatz: M. Hiller, O. Wiedemann

Die Vorlesungen "Theoretische Grundlagen der Informatik", gehalten von D. Saupe and der Universität Konstanz, beruhen auf dem Hochschultaschenbuch "Theoretische Informatik —- kurzgefasst" von Uwe Schöning (5. Auflage, 2008, Spektrum Akademischer Verlag). Diese Ergänzungen beinhalten eine kleine Sammlung von Beispielen, Vertiefungen oder weitere Erläuterungen, die in der Vorlesung behandelt werden und in dem Buch nicht oder nicht so ausführlich dargestellt sind. Sie bilden daher keinen zusammen hängenden Text und dienen auch keinesfalls als Ersatz des o.a. Manuskriptes.[1]

11. Mai 2016

[1] Mitteilungen von Fehlern im Text und Verbesserungsvorschläge gerne bitte an dietmar.saupe@uni-konstanz.de.

Inhaltsverzeichnis

I. Automatentheorie und Formale Sprachen **3**

1. Allgemeines **3**
1.1. Chomsky-Hierarchie 3

2. Reguläre Sprachen **7**
2.1. Nichtdeterministische Automaten 7
2.2. Reguläre Ausdrücke 9
2.3. Das Pumping Lemma 12
2.4. Äquivalenzrelationen und Minimalautomaten 12

3. Kontextfreie Sprachen **20**
3.1. Normalformen 20
3.2. Kellerautomaten 20

II. Berechenbarkeitstheorie **23**

III. Komplexitätstheorie **24**

4. Weitere NP-vollständige Probleme **26**

Teil I.
Automatentheorie und Formale Sprachen

1. Allgemeines

1.1. Chomsky-Hierarchie

ε-Sonderregelung: In einer Typ $1,2,3$-Grammatik G kann das leere Wort ε bedingt durch die Forderung $|w_1| \leq |w_2|$ für alle Produktionen $w_1 \to w_2$ in P nicht abgeleitet werden, d.h. es gilt immer $\varepsilon \notin L(G)$.
Ist $\varepsilon \in L(G)$ beabsichtigt, so sei für die Startvariable S die Ausnahmeproduktion $S \to \varepsilon$ unter der Bedingung gestattet, dass S auf keiner rechten Seite einer Produktion vorkommt.
In *kontextfreien* und *regulären* Grammatiken können Regeln der Form $A \to \varepsilon$ auch zugelassen werden, wenn A nicht die Startvariable ist. Jeder solchen Typ $2,3$-Grammatik G mit $\varepsilon \notin L(G)$ kann eine äquivalente, ε-Regel freie Grammatik G' zugeordnet werden:

Elimination von ε-Regeln

1. Zerlege die Menge V der Variablen in V_1 und V_2, so dass für genau die $A \in V_1$ gilt $A \Rightarrow^* \varepsilon$. Gehe dazu wie folgt vor:
   - Ist $A \to \varepsilon$ in P, so ist $A \in V_1$.
   - Weitere $B \in V_1$ findet man dadurch, dass es $B \to A_1 A_2 \dots A_k, k \geq 1$ gibt mit $A_i \in V_1, (i = 1, \dots, k)$
2. Entferne alle Regeln der Form $A \to \varepsilon$ aus P.
3. Füge für jede Regel der Form $B \to xAy$ mit $B \in V, A \in V_1$, $xy \in (V \cup \Sigma)^+$ eine weitere Regel der Form $B \to xy$ zu P hinzu.
   Dies nimmt die Möglichkeit vorweg, A auf das leere Wort abzuleiten.

Beispiel.
Betrachten wir eine Grammatik $G = (V, \Sigma, P, S)$ mit $V = \{A, B, C, S\}$, $\Sigma = \{0, 1\}$ und

$$
\begin{array}{lllll}
P = \{ & A & \to & \varepsilon, \\
 & A & \to & 0, \\
 & A & \to & 0A, \\
 & B & \to & \varepsilon, \\
 & C & \to & AB, \\
 & C & \to & 0C1, \\
 & S & \to & C \qquad \}
\end{array}
$$

Zunächst beobachten wir, dass für alle Regeln $w_1 \to w_2$ in P (mit Ausnahme der ε-Regeln) $|w_1| \leq |w_2|$ und $w_1 \in V$ gilt. Somit ist G vom Typ 2, also *kontextfrei*.

Behauptung: Für die von G erzeugte Sprache gilt:

$$L(G) = \{\varepsilon, 0^k 1^l | k \geq max(1, l)\} = \{0^k 1^l | k, l \in \mathbb{N}_0, k \geq l\}$$

Beweis:

- $\varepsilon \in L(G)$: Das leere Wort liegt in $L(G)$, da $S \to C \to AB \to \varepsilon B \to \varepsilon$ eine Linksableitung von ε in G ist.
- Die Startvariable S lässt sich nur mittels $S \to C$ ableiten.
- Durch Anwendung der Regeln $S \to C$ und (wiederholt) $C \to 0C1$ lassen sich Satzformen dergestalt $0^k C 1^k$ mit $k \geq 0$ ableiten.
- Die einzig alternative Ableitung von C ist $C \to AB$
- A lässt sich mittels $A \to 0A|0|\varepsilon$ ableiten, B hingegen nur zu ε. Insgesamt gilt also $C \Rightarrow^* 0^k, k \geq 0$.

Die durch G erzeugbaren Wörter sind also von der Form $0^k 1^l$ mit $k, l \in \mathbb{N}_0, k \geq l$ □

Elimination der ε-Regeln in G

1. *Zerlegung von V in V_1 und V_2:*
   - $A, B \in V_1$, da $A \to \varepsilon, B \to \varepsilon$ in P.
   - Somit $C \in V_1$, da $C \to AB$ in P mit $A, B \in V_1$.
   - Somit $S \in V_1$, da $S \to C$ in P, mit $C \in V_1$.

   Folglich $V_1 = \{S, A, B, C\}, V_2 = \emptyset$.
2. *Entferne alle Regeln der Form $A \to \varepsilon$ aus P:*
   $P' = P - \{A \to \varepsilon, B \to \varepsilon\}$
3. *Füge neue Ableitungsregeln hinzu:*

   $P'' = P' \cup \{C \to A, C \to B, C \to 01\}$
4. Somit ist $P'' = \{S \to C, C \to 0C1, A, B, AB, 01, A \to 0A|0\}$.

   B kann nicht weiter abgeleitet werden, entferne daher die Regeln $C \to AB$ und $C \to B$ aus P''.

   Zusätzlich wird die ε-Regel $S \to \varepsilon$ hinzugefügt, um $\varepsilon \in L(G')$ beizubehalten. Das Resultat lautet:

   $P'' = \{S \to C|\varepsilon, C \to 0C1|A|01, A \to 0A|0\}$

   mit $L((V, \Sigma, P, S)) = L((V, \Sigma, P'', S))$. □

Beispiel.

Betrachten wir erneut eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit $V = \{A, B, C, D, E, S\}$, $\Sigma = \{0, 1\}$ und

$$
\begin{array}{lllll}
P = \{ & A & \to & \varepsilon, & \\
& B & \to & \varepsilon, & \\
& C & \to & AB, & \\
& D & \to & AB, & \\
& D & \to & AB0E, & \\
& E & \to & 0CD1, & \\
& S & \to & E & \}
\end{array}
$$

Elimination der ε-Regeln in G

1. *Zerlegung von V in V_1 und V_2:*
   - $A, B \in V_1$, da $A \to \varepsilon, B \to \varepsilon$ in P.
   - $C \in V_1$, da $C \to AB$ in P mit $A, B \in V_1$.
   - $D \in V_1$, da $D \to AB$ in P mit $A, B \in V_1$.

   $V_2 = V - V_1 = \{E, S\}$

2. *Entferne alle Regeln der Form $A \to \varepsilon$ aus P:*
   $P' = P - \{A \to \varepsilon, B \to \varepsilon\}$

3. *Füge neue Ableitungsregeln hinzu:*
   $P'' = P' \cup \{C \to A|B, D \to A|B|0E|A0E|B0E, E \to 01|0C1|0D1\}$

4. Somit ist $P'' = \{S \to E, E \to 0CD1|0C1|0D1|01$
   $D \to A|B|0E|A0E|B0E|AB0E|, C \to A|B|AB\}$

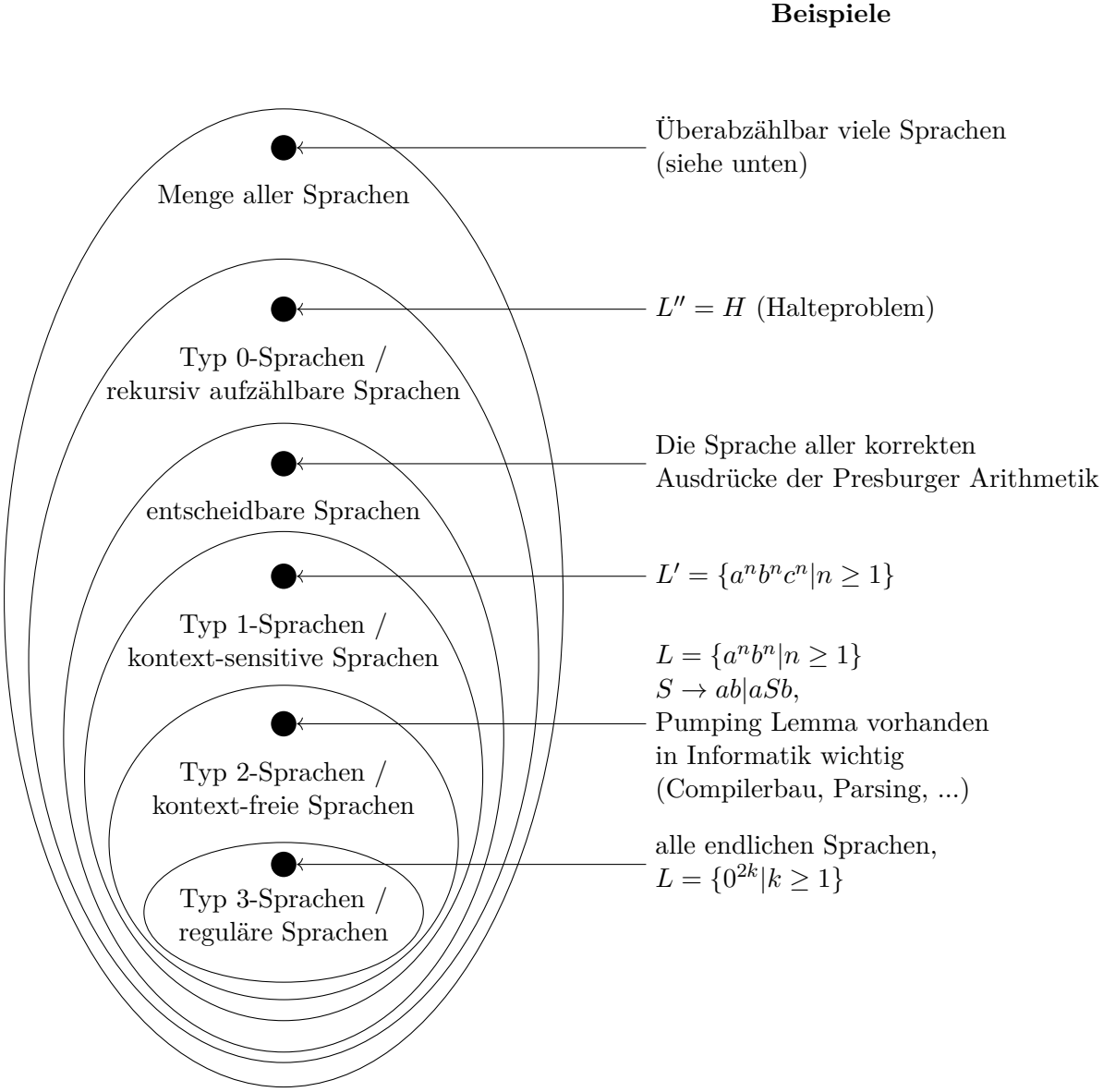
5. Die Variablen A, B und indirekt C sind nicht weiter ableitbar, somit kann P'' vereinfacht werden zum Resultat $P''' = \{S \to E, E \to 0D1|01, D \to 0E\}$

6. Die einzig mögliche Ableitung von D lautet $D \to 0E$, vereinfache P''' somit weiter zu $P^{\mathrm{IV}} = \{S \to E, E \to 00E1|01\}$

7. Als weitere Vereinfachung folgt aus der einzigen Ableitung $S \to E$ der Startvariable $P^{\mathrm{V}} = \{S \to 00S1|01\}$

Unmittelbar aus dem Ergebnis der Elminiation der ε-Regeln und einiger Vereinfachungen lässt sich der Sprachumfang der Grammatik ablesen: $L(G) = \{0^{2k+1}1^{k+1} | k \in \mathbb{N}_0\}$

Hierarchie von Chomsky:



Es gibt Sprachen, die nicht durch eine Grammatik beschreibbar sind. Aus der Abzählbarkeit der Menge aller Turingmaschinen folgt mittels einer Abbildung, die jeder *TM* die von ihr akzeptierte Sprache zuweist, dass auch die Menge aller Typ-0 Sprachen abzählbar ist.

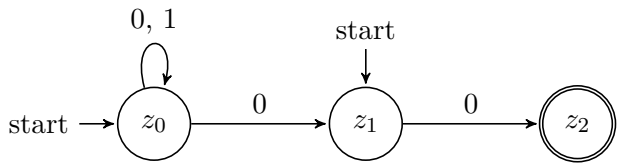
Die Menge *aller* Sprachen ist überabzählbar: Sprachen $L \subseteq \{a, b\}^*$ sind durch Funktionen $f : \mathbb{N} \to \{0, 1\}$ charakterisierbar:
Sei dazu $\gamma : \Sigma^* \to \mathbb{N}$ die lexikographische Nummerierung von Σ^* und setze $f(n) = 1$ genau dann, wenn $\gamma(n) \in L$. Die Menge der Zahlen $x_f = \Sigma_{n=0}^{\infty} f(n) \cdot 2^{-n-1}$ für alle $f : \mathbb{N} \to \{0, 1\}$ ist das Einheitsintervall mit $|[0, 1]| = |\mathbb{R}| > |\mathbb{N}|$,
aber $\{x_f | f$ charakterisiert Typ-0 Sprache$\}$ ist nur abzählbar. Somit existieren bereits über einem zweielementigen Alphabet mehr Sprachen als Typ-0 Grammatiken. □

2. Reguläre Sprachen

2.1. Nichtdeterministische Automaten

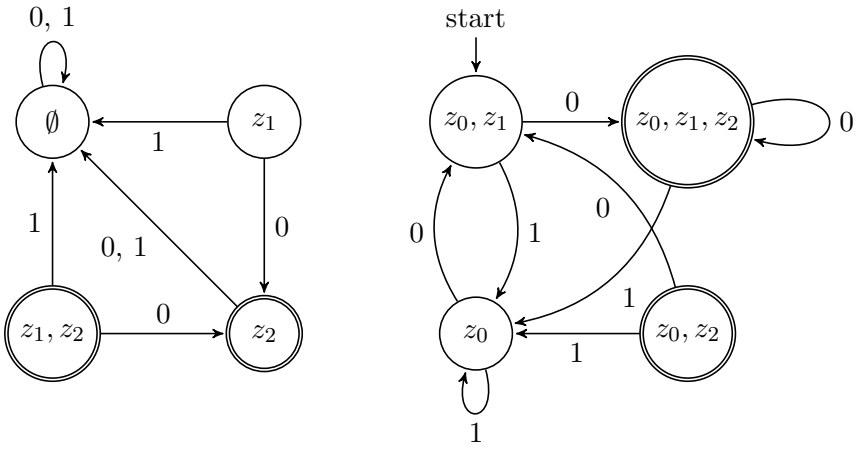
In der Vorlesung wurde bereits die Äquivalenz von deterministischen und nichtdeterministischen Automaten bewiesen. Im Folgenden behandeln wir ein Beispiel für die Umwandlung eines NFA M in einen DFA M'.

NFA-DFA-Umformung Gegeben sei ein NFA M:

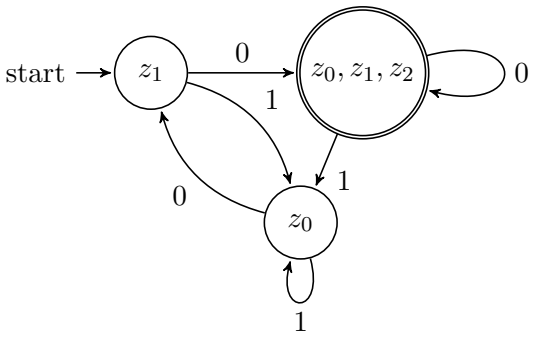


<table>
<tr><td>NFA M</td><td>DFA M'</td></tr>
<tr><td>$Z = \{z_0, z_1, z_2\}$</td><td>$Z' = \mathcal{P}(Z) = \{\emptyset, \{z_0\}, \{z_1\}, \{z_2\}, \{z_0, z_1\}, \{z_0, z_2\}, \{z_1, z_2\}, \{z_0, z_1, z_2\}\}$</td></tr>
<tr><td>$\Sigma = \{0, 1\}$</td><td>$\Sigma = \{0, 1\}$</td></tr>
<tr><td>$\delta : Z \times \Sigma \to P(Z)$</td><td>$\delta : Z' \times \Sigma \to Z'$
$\delta(Z', a) = \bigcup_{z \in Z'} \delta(z, a)$</td></tr>
<tr><td>
<table>
<tr><td>δ</td><td>0</td><td>1</td></tr>
<tr><td>$\{z_0\}$</td><td>$\{z_0, z_1\}$</td><td>$\{z_0\}$</td></tr>
<tr><td>$\{z_1\}$</td><td>$\{z_2\}$</td><td>$\emptyset$</td></tr>
<tr><td>$E \ni \{z_2\}$</td><td>$\emptyset$</td><td>$\emptyset$</td></tr>
</table>
</td><td>
<table>
<tr><td>δ'</td><td>0</td><td>1</td></tr>
<tr><td>$\{z_0\}$</td><td>$\{z_0, z_1\}$</td><td>$\{z_0\}$</td></tr>
<tr><td>$\{z_1\}$</td><td>$\{z_2\}$</td><td>$\emptyset$</td></tr>
<tr><td>$E' \ni \{z_2\}$</td><td>$\emptyset$</td><td>$\emptyset$</td></tr>
<tr><td>$\{z_0, z_1\}$</td><td>$\{z_0, z_1, z_2\}$</td><td>$\{z_0\}$</td></tr>
<tr><td>$E' \ni \{z_0, z_2\}$</td><td>$\{z_0, z_1\}$</td><td>$\{z_0\}$</td></tr>
<tr><td>$E' \ni \{z_1, z_2\}$</td><td>$\{z_2\}$</td><td>$\emptyset$</td></tr>
<tr><td>$E' \ni \{z_0, z_1, z_2\}$</td><td>$\{z_0, z_1, z_2\}$</td><td>$\{z_0\}$</td></tr>
</table>
</td></tr>
<tr><td>Startzustände $S \subseteq Z$, $S = \{z_0, z_1\}$</td><td>Startzustand $z_0' = S = \{z_0, z_1\}$</td></tr>
<tr><td>Endzustände $E \subseteq Z$, $E = \{z_2\}$</td><td>Endzustände $E' = \{Z' \subseteq P(Z) | Z' \cap E \neq \emptyset\}$
$E' = \{\{z_2\}, \{z_0, z_2\}, \{z_1, z_2\}, \{z_0, z_1, z_2\}\}$</td></tr>
</table>

Zum gegebenen NFA M ist der nichtminimale DFA M' äquivalent, d.h.:
$w \in T(M) \Leftrightarrow w \in T(M')$:



Bezüglich $T(M')$ gibt es überflüssige Zustände: Der gesamte linke Block ist von keinem Startzustand aus erreichbar, rechts existiert keine Überführung, mittels derer der Zustand z_0, z_2 betreten werden kann. Streiche diese irrelevanten Zustände aus dem Automaten:

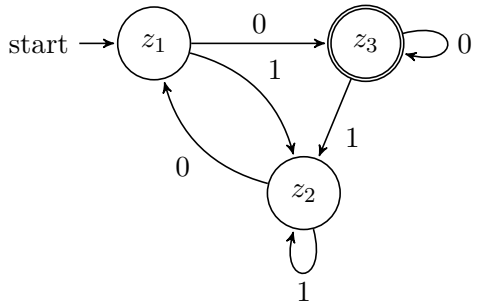


Man erkennt leicht, dass $T(M') = L(0|(0|1)^*00)$ gilt. Nicht immer gelingt eine Vereinfachung von M'. Im Worst Case ist für $|Z| = n$ und $|P| = 2^n$ keine Vereinfachung möglich, womit M' relativ zu M exponentiell viele Zustände besitzt.

2.2. Reguläre Ausdrücke

Beispiel zur Herleitung eines regulären Ausdrucks zu einer regulären Sprache, die durch einen gegebenen DFA M' definiert ist.

DFA M':



Zugehörige Grammatik:

$V = \{z_1, z_2, z_3\}$
$\Sigma = \{0, 1\}$
$S = z_1$
$P = \{$

$$\begin{array}{lcll} z_1 & \to & 0, \\ z_1 & \to & 0z_3, \\ z_1 & \to & 1z_2, \\ z_2 & \to & 0, \\ z_2 & \to & 0z_3, \\ z_2 & \to & 1z_2, \\ z_3 & \to & 0z_1, \\ z_3 & \to & 1z_2 & \} \end{array}$$

Die Vorgehensweise zur Konstruktion eines regulären Ausdrucks zu einem gegebenen DFA ergibt sich aus dem Beweis zum Satz von *Kleene*. Wir nehmen an, die Zustände des Automaten sind beginnend von 1 an aufsteigend nummeriert: $Z = \{z_1, \ldots, z_n\}$, wobei z_1 der Startzustand des Automaten sei.
Betrachte im konkreten Beispiel für $i, j = 1, 2, 3$, $k = 0, 1, 2, 3$, $n = |Z| = 3$ die Menge

$$R^k_{i,j} = \{x \in \Sigma^* | x \text{ überführt } z_i \text{ nach } z_j \text{ über Zwischenzustände mit Index } \leq k\}$$

welche induktiv gegeben ist durch

$$R^{k+1}_{i,j} = R^k_{i,j} \cup R^k_{i,k+1}(R^k_{k+1,k+1})^* R^k_{k+1,j}$$

Mit den folgenden Basisfällen:
Für $k = 0, i = j$ gilt: $R^0_{i,i} = \{\varepsilon\} \cup \{a \in \Sigma | \delta(z_i, a) = z_i\}$,
für $k = 0, i \neq j$ gilt: $R^0_{i,j} = \{a \in \Sigma | \delta(z_i, a) = z_j\}$.

Für die durch den Automaten akzeptierte Sprache gilt:

$$T(M) = \bigcup_{z_j \in E} R^n_{1,j} = R^3_{1,3}$$

In unserem Beispiel also:

$$T(M) = R^3_{1,3}$$

Bestimme den $T(M)$ zugehörigen regulären Ausdruck:

Rekursionsvorschrift:

$$R^3_{1,3} = R^2_{1,3} \cup R^2_{1,3}(R^2_{3,3})^* R^2_{3,3}$$
$$R^2_{1,3} = R^1_{1,3} \cup R^1_{1,2}(R^1_{2,2})^* R^1_{2,3}$$
$$R^1_{1,3} = R^0_{1,3} \cup R^0_{1,1}(R^0_{1,1})^* R^0_{1,3}$$

Basisfälle:

$$R^0_{1,1} = L(\varepsilon)$$
$$R^0_{1,2} = \{1\} = L(1)$$
$$R^0_{1,3} = \{0\} = L(0)$$
$$R^0_{2,3} = \emptyset = L(\emptyset)$$

Zwischenrechnungen:

$$\begin{aligned} R^1_{1,2} &= R^0_{1,2} \cup R^0_{1,1}(R^0_{1,1})^* R^0_{1,2} \\ &= L(1) \cup L(\varepsilon)(L(\varepsilon))^* L(1) \\ &= L(1) \end{aligned}$$

$$\begin{aligned} R^1_{1,3} &= R^0_{1,3} \cup R^0_{1,1}(R^0_{1,1})^* R^0_{1,3} \\ &= L(0) \end{aligned}$$

$$\begin{aligned} R^1_{2,2} &= R^0_{2,2} \cup R^0_{2,1}(R^0_{1,1})^* R^0_{1,2} \\ &= L(1) \cup L(0)L(\varepsilon)^* L(1) \\ &= L(1|01) \end{aligned}$$

$$\begin{aligned} R^1_{2,3} &= R^0_{2,3} \cup R^0_{2,1}(R^0_{1,1})^* R^0_{1,3} \\ &= L(\emptyset) \cup L(0)(L(\varepsilon))^* L(0) \\ &= L(00) \end{aligned}$$

$$\begin{aligned} R^1_{3,1} &= R^0_{3,1} \cup R^0_{3,1}(R^0_{1,1})^* R^0_{1,1} \\ &= L(\emptyset) \cup L(\emptyset)(L(\varepsilon))^* L(\varepsilon) \\ &= L(\emptyset) \end{aligned}$$

$$\begin{aligned} R^1_{3,2} &= R^0_{3,2} \cup R^0_{3,1}(R^0_{1,1})^* R_{1,2} \\ &= L(1) \cup Ł(\emptyset)L(\varepsilon)L(1) \\ &= L(1) \end{aligned}$$

$$\begin{aligned} R^1_{3,3} &= R^0_{3,3} \cup R^0_{3,1}(R^0_{1,1})^* R^0_{1,3} \\ &= L(\varepsilon|0) \cup L(\emptyset)(L(\varepsilon))^* L(0) \\ &= L(\varepsilon|0) \end{aligned}$$

$$
\begin{aligned}
R^2_{3,3} &= R^1_{3,3} \cup R^1_{3,2}(R^1_{2,2})^* R^1_{2,3} \\
&= L(\varepsilon|0) \cup L(1)(L(1|01))^* L(00) \\
&= L(\varepsilon|0|1(1|01)^*00)
\end{aligned}
$$

Folgerungen:

$$
\begin{aligned}
R^1_{1,3} &= R^0_{1,3} \cup R^0_{1,1}(R^0_{1,1})^* R^0_{1,3} \\
&= L(0) \cup L(\varepsilon)(L(\varepsilon))^* L(0) \\
&= L(0|\varepsilon(\varepsilon)^*0) \\
&= L(0)
\end{aligned}
$$

$$
\begin{aligned}
R^2_{1,3} &= R^1_{1,3} \cup R^1_{1,2}(R^1_{2,2})^* R^1_{2,3} \\
&= L(0) \cup L(1)L(1|01)^* L(00) \\
&= L(0|1(1|01)^*00)
\end{aligned}
$$

$$
\begin{aligned}
R^3_{1,3} &= R^2_{1,3} \cup R^2_{1,3}(R^2_{3,3})^* R^2_{3,3} \\
&= L(0|1(1|01)^*00) \cup L(0|1(1|01)^*00)(L(\varepsilon|0|1(1|01)^*00))^* L(\varepsilon|0|1(1|01)^*00) \\
&= L(0|1(1|01)^*00) \cup L(0|1(1|01)^*00)(L(\varepsilon|0|1(1|01)^*00))^* \\
&= L((0|1(1|01)^*00)|(0|1(1|01)^*00)(\varepsilon|0|1(1|01)^*00)^*) \\
&= L((0|1(1|01)^*00)(\varepsilon|0|1(1|01)^*00)^*) \\
&= L(0|1(1|01)^*00)^+
\end{aligned}
$$

Der so gefundene reguläre Ausdruck lässt sich weiter vereinfachen:

$$L(0|1(1|01)^*00)^+ = L(0|(1|0)^*00)$$

.

Dies resultiert in unserem Fall nicht aus einem Kalkül, sondern aus einer Beobachtung über den Aufbau des regulären Ausdrucks. Mittels des Teilausdrucks $(1|01)^*$ lassen sich beliebige Bitstrings erzeugen, die keine zwei aufeinander folgenden Nullen beinhalten. Werden schließlich zwei oder mehr Nullen benötigt, so wird die abschließende Sequenz des Teilausdrucks $1(1|01)^*00$ durchlaufen. Weitere 0en lassen sich mittels der ersten 0 im Regex einfügen. Folglich erzeugt L eine Sprache, die aus beliebigen Bitstrings gefolgt von 00 bzw. einer einfachen 0 besteht.

2.3. Das Pumping Lemma

Beispiel.

Behauptung: Die Sprache $L = \{a^i b^j c^k | i < j < k\}$ ist nicht regulär.
Beweis: Per Kontraposition mittels des Pumping-Lemmas.
Angenommen, L ist regulär mit zugehöriger Pumping-Lemma-Zahl $n \in \mathbb{N}$. Betrachte das Wort $x = a^n b^{n+1} c^{n+2}$, so gilt nach Definition der Sprache $x \in L$ und $|x| \geq n$. Sei nun $x = uvw$ eine Zerlegung des Wortes x. Aus der Forderung $|v| \geq 1$ folgt, dass $v \neq \varepsilon$. Durch $|uv| \leq n$ ergibt sich weiter, dass v nur aus a's besteht, sprich $v = a^m$ für $1 \leq m \leq n$.
Wähle nun $z \geq 1$ so, dass $m' := n + (z-1)m \geq n+1$. Dann ist $uv^z w = a^{m'} b^{n+1} c^{n+2}$, somit $uv^z w \notin L$, was im Widerspruch zum Pumping Lemma steht. L ist also nicht regulär. □

2.4. Äquivalenzrelationen und Minimalautomaten

Definition *Äquivalenzrelation*
Sei M eine Menge. $R \subseteq M \times M$ heißt Äquivalenzrelation, wenn:

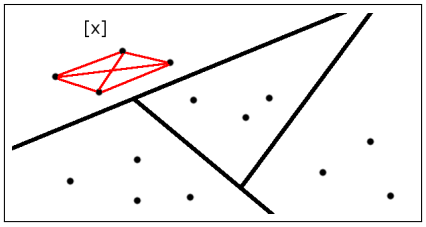
(1) $\forall x \in M$	$(x,x) \in R$	(Reflexivität)
(2) $\forall x, y \in M$	$(x,y) \in R \Leftrightarrow (y,x) \in R$	(Symmetrie)
(3) $\forall x, y, z \in M$	$(x,y) \in R \wedge (y,z) \in R \Rightarrow (x,z) \in R$	(Transitivität)

gilt. Schreibweisen: $(x,y) \in R$ oder xRy oder $x \sim y$

Definition *Äquivalenzklasse*
Sei R eine Äquivalenzrelation auf M. Definiere die *Äquivalenzklasse* von $x \in M$ bzgl. R als

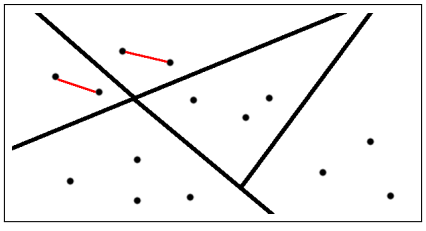
$$[x]_R := \{y \in M | (x,y) \in R\}$$

Dies ist wohldefiniert, denn es gilt $[y]_R = [x]_R$ für alle $y \in [x]_R$.



Eine Äquivalenzrelation R auf M induziert eine disjunkte Zerlegung von M durch $\bigcup_{x \in M} [x]_R$, da für $z \in [x]_R \cap [y]_R$ folgt, dass $(z,x) \in R \wedge (z,y) \in R$, ferner $(x,z) \in R$ durch Symmetrie und schlussendlich $(x,y) \in R$ bedingt durch die Transitivität.

Definition *Verfeinerung*
Eine Relation $S \subseteq M \times M$ heißt Verfeinerung einer Relation R, wenn $S \subseteq R$.
Dann gilt für alle $x \in M$: $[x]_S \subseteq [x]_R$.



Definition *Index*
Der Index $ind(R)$ einer Äquivalenzrelation R über M ist die Anzahl ihrer Äquivalenzklassen: $ind(R) = |\{[x]_R | x \in M\}|$. Beachte, dass $ind(R) = \infty$ möglich ist.
Wenn S eine Verfeinerung von R ist ($S \subseteq R$), dann gilt $ind(S) \geq ind(R)$.

Definition *Myhill-Nerode Äquivalenzrelation*
Sei $L \subseteq \Sigma^*$ eine Sprache. Durch xR_Ly wird genau dann eine Äquivalenzrelation definiert, wenn $\forall z \in \Sigma^* : (xz \in L \Leftrightarrow yz \in L)$. Wähle insbesondere $z = \varepsilon$, so gilt: $x \in L \Leftrightarrow y \in L$

Definition *DFA-Äquivalenzrelation*
Zum DFA $M = (Z, \Sigma, \delta, z_0, E)$ sei die Relation $R_M \subseteq \Sigma^* \times \Sigma^*$ definiert als

$$xR_My :\Leftrightarrow \widehat{\delta}(z_0, x) = \widehat{\delta}(z_0, y)$$

mit $x, y \in \Sigma^*$. R_M ist eine Äquivalenzrelation, es gilt $ind(R_M) = |Z| < \infty$.

Lemma
Zum DFA $M = (Z, \Sigma, \delta, z_0, E)$ sei $L = T(M)$, L folglich regulär.
Es gilt: R_M ist eine Verfeinerung von R_L, d.h. $R_M \subseteq R_L$ bzw. $xR_My \Rightarrow xR_Ly$.
Beweis:
Seien $x, y \in \Sigma^*$ mit xR_My, also $\widehat{\delta}(z_0, x) = \widehat{\delta}(z_0, y)$, sei $z \in \Sigma^*$. Dann gilt:

$$\begin{aligned} xz \in L &\Leftrightarrow \widehat{\delta}(z_0, xz) \in E \\ &\Leftrightarrow \widehat{\delta}(\widehat{\delta}(z_0, x), z) \in E \\ &\Leftrightarrow \widehat{\delta}(\widehat{\delta}(z_0, y), z) \in E \\ &\Leftrightarrow \widehat{\delta}(z_0, yz) \in E \\ &\Leftrightarrow yz \in L \end{aligned}$$

Damit gilt xR_Ly. □

Satz
Es gilt: Eine Sprache L ist genau dann regulär, wenn der Index von R_L endlich ist.
Beweis:
($\Rightarrow$): Sei L regulär und $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA mit $L = T(M)$, dann ist $R_M \subseteq R_L$, folglich gilt $ind(R_L) \leq ind(R_M) \leq |Z| < \infty$.

($\Leftarrow$): Sei $L \subseteq \Sigma^*$ und $ind(R_L) < \infty$. Somit gibt es $x_1, ..., x_n \in \Sigma^*$ mit $\Sigma^* = [x_1]_{R_L} \cup ... \cup [x_n]_{R_L}$.
Definiere DFA $M = (Z, \Sigma, \delta, z_0, E)$ mit $Z = \{[x_1]_{R_L}, ..., [x_n]_{R_L}\}$,
$\delta([x]_{R_L}, a) = [xa]_{R_L}$,
$z_0 = [\varepsilon]_{R_L}$
$E = \{[x]_{R_L} | x \in L\}$.

δ ist wohldefiniert, denn für $[x]_{R_L} = [y]_{R_L}$ sind alle $z \in \Sigma^*$ wegen xR_Ly für alle $z \in \Sigma^*$ $xaz \in L \Leftrightarrow yaz \in L$, also xaR_Lya und $[xa]_{R_L} = [ya]_{R_L}$
Per Induktion nach $|x|$ folgt $\hat{\delta}([\varepsilon]), x = [x]$ und $x \in T(M) \Leftrightarrow \hat{\delta}([\varepsilon], x) \in E \Leftrightarrow [x] \in E \Leftrightarrow x \in L$.
Der DFA akzeptiert also L, somit ist L regulär. M heißt *Äquivalenzklassenautomat.* □

Beispiel: Betrachte $L = \{a^n b^n | n \geq 1\}$ (nicht-regulär)

Es gibt unendlich viele von R_L induzierte Äquivalenzklassen:

$$\begin{aligned}[ab] &= L \\ [a^2 b] &= \{a^{k+1} b^k | k \geq 1\} \\ &... \\ [a^m b] &= \{a^{k+m} b^k | k \geq 1\} \text{ für alle } m = 1, 2, \ldots\end{aligned}$$

Dies folgt aus der Tatsache, dass für $i \neq j$ die Relation $a^i b R_L a^j b$ nicht gilt.
Damit gilt $ind(R_L) = \infty$ (es genügt, unendlich viele Äquivalenzklassen zu finden)

Beispiel:
$L = \{x \in \{0,1\}^* | x \text{ endet mit } 00\}$

$[\varepsilon] = \{x | x \text{ endet nicht mit } 00\}$
$[0] = \{x | x \text{ endet mit 0, aber nicht mit } 00\}$
$[00] = \{x | x \text{ endet mit } 00\}$

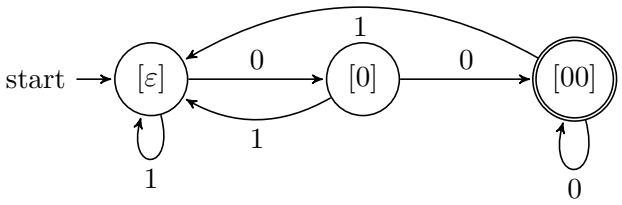
$\Sigma^* = [\varepsilon] \cup [0] \cup [00]$
$ind(R_L) = 3 < \infty \Rightarrow L$ regulär

Äquivalenzklassenautomat:

δ	0	1
$[\varepsilon]$	$[0]$	$[1] = [\varepsilon]$
$[0]$	$[00]$	$[01] = [\varepsilon]$
$[00]$	$[000] = [00]$	$[001] = [\varepsilon]$

$z_0 = [\varepsilon]$
$E = \{[00]\}$

Schaubild:



Lemma
Sei $M = (Z, \Sigma, \delta, z_0, E)$ der Äquivalenzklassenautomat zu $L \subseteq \Sigma^*$ mit $ind(R_L) < \infty$.
Es gilt: $R_M = R_L$.
Beweis:
R_M ist eine Verfeinerung $R_M \subseteq R_L$ (Lemma 1).
Zeige $R_L \subseteq R_M$:
Sei $x R_L y$, also $[x]_{R_L} = [y]_{R_L}$
Aus $\widehat{\delta}(z_0, x) = \widehat{\delta}([\varepsilon]_{R_L}, x) = [x]_{R_L} = [y]_{R_L} = \widehat{\delta}([\varepsilon]_{R_L}, y) = \widehat{\delta}(z_0, y)$ folgt $x R_M y$. □

Satz
Ein Äquivalenzklassenautomat ist ein Minimalautomat, d.h. er hat eine minimale Anzahl von Zuständen.
Beweis:
Sei M der Äquivalenzklassenautomat zu L, M' ein beliebiger DFA mit $T(M') = L$.
Wir zeigen, dass M' mindestens so viele Zustände hat wie M.
Aus den zuvor bewiesenen Lemmata folgt die Gültigkeit von $R_{M'} \subseteq R_L$ und $R_M = R_L$.
Somit folgt $R_{M'} \subseteq R_L = R_M$, und damit:

$$\begin{array}{ccccc} ind(R_{M'}) & \geq & ind(R_L) & = & ind(R_M) \\ \| & & & & \| \\ \#(\text{Zustände von } M') & & & & \#(\text{Zustände von } M) \end{array}$$

□

Proposition
Die Struktur eines Minimalautomaten ist eindeutig bis auf Isomorphie der Zustandsmenge.
Ausformuliert bedeutet die Behauptung das folgende:

- Sei M_0 der Äquivalenzklassenautomat mit $M_0 = (Z, \Sigma, \delta, z_1, E)$ und $T(M_0) = L$,
  $Z = \{z_1, ..., z_n\} = \{[x_1], [x_2], ..., [x_n]\}$, $x_1 = \varepsilon$
  $\delta_0([x_k], a) = [x_k a]$ für $[x_k] \in Z$, $a \in \Sigma$
  $E = \{[x] | x \in L\}$
  Es ist $R_{M_0} = R_L$ und M_0 ist Minimalautomat.

- Sei $M = (Z, \Sigma, \delta, z_M, E_M)$ ein beliebiger Minimalautomat mit $T(M) = L$. M und M_0 haben gleich viele Zustände.

- Dann gibt es eine Permutation $\pi : Z \to Z$ von Z mit

  1.) $\pi(z_1) = z_M$

  2.) für $a \in \Sigma$

$$\begin{array}{ccccc} Z & \xrightarrow{\delta_0(\cdot,a)} & Z & \text{in } M_0 \\ \pi\downarrow & // & \pi\downarrow & \\ Z & \xrightarrow{\delta(\cdot,a)} & Z & \text{in } M \end{array}$$

  als Formel: $\underset{z \in Z}{\forall} \delta(\pi(z), a) = \pi(\delta_0(z, a))$

  3.) $E_M = \{\pi(z) | z \in E\}$

Beweis.
Zur Motivation: Sei $x = a_1a_2a_3$. Wir haben:

$$\begin{array}{lcccccccc} \text{in } M_0 & Z & \xrightarrow{\delta_0(\cdot,a_1)} & Z & \xrightarrow{\delta_0(\cdot,a_2)} & Z & \xrightarrow{\delta_0(\cdot,a_3)} & Z \\ & \pi\downarrow & /\!/ & \pi\downarrow & /\!/ & \pi\downarrow & /\!/ & \pi\downarrow \\ \text{in } M & Z & \xrightarrow{\delta(\cdot,a_1)} & Z & \xrightarrow{\delta(\cdot,a_2)} & Z & \xrightarrow{\delta(\cdot,a_3)} & Z \end{array}$$

also zusammengefasst

$$\begin{array}{lccc} \text{in } M_0 & Z & \xrightarrow{\widehat{\delta_0}(\cdot,x)} & Z \\ & \pi\downarrow & /\!/ & \pi\downarrow \\ \text{in } M & Z & \xrightarrow{\widehat{\delta}(\cdot,x)} & Z \end{array}$$

Wir brauchen für alle $x \in \Sigma^*$

$$\begin{array}{ccc} Z & \xrightarrow{\widehat{\delta_0}(\cdot,x)} & Z \\ \pi\downarrow & /\!/ & \pi\downarrow \\ Z & \xrightarrow{\widehat{\delta}(\cdot,x)} & Z \end{array}$$

speziell für den Startzustand $z_1 \in Z$

$$\begin{array}{ccc} z_1 & \mapsto & \widehat{\delta_0}(z_1,x) \\ \downarrow & & \downarrow \\ z_M & \mapsto & \widehat{\delta}(z_M,x) \end{array}$$

also

$$\pi(\widehat{\delta_0}(z_1,x)) = \widehat{\delta}(z_M,x)$$

Da $\widehat{\delta_0}(z_1,x) = \widehat{\delta_0}([\varepsilon],x) = [x]$ ergibt sich $\pi([x]) = \widehat{\delta}(z_M,x)$. Das motiviert nun entsprechend der obigen Überlegungen den Ansatz

$$\pi(z_k) := \widehat{\delta}(z_M,x_k)$$

wobei $x_k \in z_k$, d.h. $z_k = [x_k]$ für $k = 1,\ldots,n$. Wir bemerken zunächst, dass $\pi(z_k) = \widehat{\delta}(z_M,x_k)$ wohldefiniert ist. Sei nun $x'_k \in z_k$, also $[x_k]_{R_L} = [x'_k]_{R_L}$. Dann ist $x_kR_Lx'_k = x_kR_Mx'_k$ und $\widehat{\delta}(z_M,x_k) = \widehat{\delta}(z_M,x'_k)$, d.h. $\pi(z_k)$ ist unabhängig von der Wahl von x_k.
Damit gilt:

1. $z_1 = [\varepsilon]$, $\pi(z_1) = \widehat{\delta}(z_M,\varepsilon) = z_M$
2. Sei $a \in Z$, $z \in Z$, $z = [x]$ für ein $x \in \Sigma^*$. Wir rechnen

$$\delta(\pi(z),a) = \delta(\widehat{\delta}(z_M,x),a) = \widehat{\delta}(z_M,xa)$$
$$\pi(\delta_0(z,a)) = \pi(\delta_0([x],a)) = \pi([xa]) = \widehat{\delta}(z_M,xa)$$

und erhalten

$$\delta(\pi(z),a) = \pi(\delta_0(z,a))$$

.

3. $E_M = \{\pi(z)|z \in E\}$ folgt schließlich aus:
   $\supseteq$: Sei $z \in E$, $z = [x]$. Dann ist $x \in L$ und es gilt $\pi(z) = \widehat{\delta}(z_M, x) \in E_M$.

   $\subseteq$: Sei $z' \in E_M$.
   Dann gibt es ein $x \in \Sigma^*$ mit $\widehat{\delta}(z_M, x) = z'$ (Sonst wäre z' nicht erreichbar in M, durch Löschen von z' erhielte man einen kleineren Automaten - Widerspruch zur Minimalität).
   Dann ist $x \in L$, $[x] \in E$, und
   $\pi([x]) = \widehat{\delta}(z_M, x) = z'$.
   Also gibt es $z = [x] \in E$ mit $\pi(z) = z'$.

Wir zeigen, dass mit der Definition $\pi(z_k) := \widehat{\delta}(z_M, x_k)$ mit $x_k \in z_k$ und für $k = 1, \ldots,$ die behaupteten Beziehungen 1. bis 3. gelten.
Um zunächst zu verifizieren, dass die Permutation π wohldefiniert, d.h. unabhängig von der Wahl von x_k ist, bemerken wir vorab, dass, da M minimal ist, $R_M = R_L$ gilt. M akzeptiert L, also gilt $R_M \subseteq R_L$ (s. Lemma oben). Damit gilt

$$n = ind(R_L) \leq ind(R_M) \leq |Z| = n$$

woraus $ind(R_L) = ind(R_M)$ und $R_M = R_L$ folgt.

Lemma
Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA. Wenn M nicht minimal ist, dann gilt:

$$\underset{z,z' \in Z}{\exists} z \neq z' \underset{x \in \Sigma^*}{\forall} \delta(z, x) \in E \Leftrightarrow \delta(z', x) \in E$$

Beweis. Sei M nicht minimal, d.h. es gilt $R_M \subseteq R_L$ und $ind(R_M) > ind(R_L)$, wobei $L = T(M)$. Somit muss es eine Klasse von R_L geben, die (mindestens) zwei Klassen von R_M enthält:

$$\exists u, u' \in \Sigma^* [u]_{R_M} \cup [u']_{R_M} \subseteq [u]_{R_L} \text{ und } [u]_{R_M} \cap [u'] = \emptyset$$

Also gilt nicht uR_Mu', d.h.

$$z = \widehat{\delta}(z_0, u) \neq z' = \widehat{\delta}(z_0, u')$$

aber uR_Lu' gilt, d.h.

$$\forall x \in \Sigma^* ux \in L \Leftrightarrow u'x \in L$$

Damit gilt für alle $x \in \Sigma^*$

$$ux \in L \Leftrightarrow \widehat{\delta}(z_0, ux) = \widehat{\delta}(\widehat{\delta}(z_0, u), x) = \widehat{\delta}(z, x) \in E$$

$$u'x \in L \Leftrightarrow \widehat{\delta}(z_0, u'x) = \widehat{\delta}(\widehat{\delta}(z_0, u'), x) = \widehat{\delta}(z', x) \in E$$

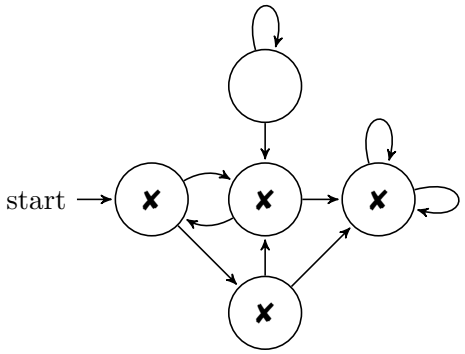
woraus die Behauptung folgt. □

Transformation eines DFA in einen Minimalautomaten: Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein gegebener DFA. Zunächst werden alle nicht erreichbaren Zustände mittels dem folgenden Algorithmus entfernt:

Algorithmus

while $(L \neq \{\})$ {
$L' := \{\delta(z_i, a) | z_i \in L, a \in \Sigma\} - M$
$M' := M \cup L$
$L := L'$
$M := M'$
}

Markierte zustände (✘)

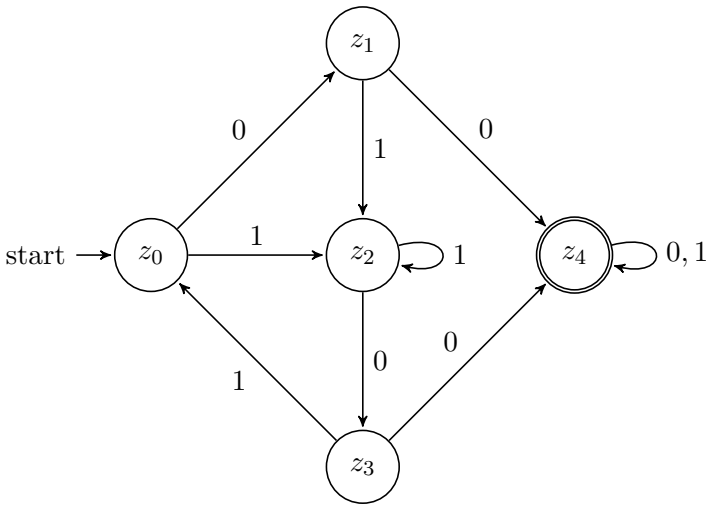


Algorithmus (Minimalautomat):
Eingabe: DFA M (nicht erreichbare Zustände seien bereits entfernt)
Ausgabe: Paare (z, z'), die für den Minimalautomaten „verschmolzen“ werden müssen.

1. Konstruiere eine Tabelle von Paaren $\{z, z'\}$ mit $z \neq z'$.
2. Markiere alle $\{z, z'\}$ mit $z \in E$, $z' \notin E$ (oder umgekehrt).
3. Für alle unmarkierten Paare $\{z, z'\}$, alle $a \in \Sigma$ teste, ob $\{\delta(z, a), \delta(z', a)\}$ markiert ist. Wenn ja, markiere $\{z, z'\}$.
4. Iteriere Schritt 3, bis keiner weitere Änderung mehr stattfindet.
5. Alle unmarkierten Zustandspaare können verschmolzen werden.

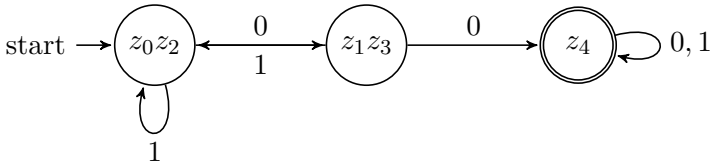
Algorithmus stellt iterativ sicher, dass die Bedingung im obigen Lemma nicht mehr gilt. Damit ist der resultierende Automat minimal.

Beispiel.



z_0					
z_1	✘				
z_2		✘			
z_3	✘		✘		
z_4	✘	✘	✘	✘	
	z_0	z_1	z_2	z_3	z_4

Ausgabe: $\{z_0, z_2\}, \{z_1, z_3\} \Rightarrow$ Verschmelze z_0 mit z_2 und z_1 mit z_3.
Damit erhält man folgenden Minimalautomaten:
$L = \{x \in \{0,1\}^* | \text{in } x \text{ kommt } 00 \text{ vor}\}$



Die zugehörige reguläre Sprache ist jetzt offensichtlich:
$L = \{x \in \{0,1\}^* | \text{in } x \text{ kommt } 00 \text{ vor}\}$

3. Kontextfreie Sprachen

3.1. Normalformen

Chomsky-Normalform-Umformung:

Beispiel.

$L = \{x^n y^n z^m | n, m \geq 1\}$ ist kontextfrei.

$$\begin{array}{lcll} S & \to & AB & \\ A & \to & xy & | \quad xAy \\ B & \to & z & | \quad zB \end{array}$$

Umformung:

$$\begin{array}{ll} A \to xy: & A \to XY \\ & X \to x \\ & Y \to y \\ A \to xAy: & A \to XC \\ & C \to AY \\ B \to zB: & B \to ZB \end{array}$$

3.2. Kellerautomaten

Der Versuch, einen (nicht-)deterministischen Automaten für die kontextfreie Sprache $L \subseteq \Sigma^*$ mit

$$\Sigma = \{\$, \alpha_0, ..., \alpha_{m-1}\} \quad \text{und} \quad L = \{a_1 ... a_n \$ a_n ... a_1 | n \geq 0, a_i \in \Sigma \backslash \{\$\}\}$$

zu konstruieren scheitert. Ein Automat dazu muss Präfixe „speichern“ können. Ein eingelesener Präfix $\alpha_{i_0} \alpha_{i_1} \dots \alpha_{i_k}$ wird als Integer

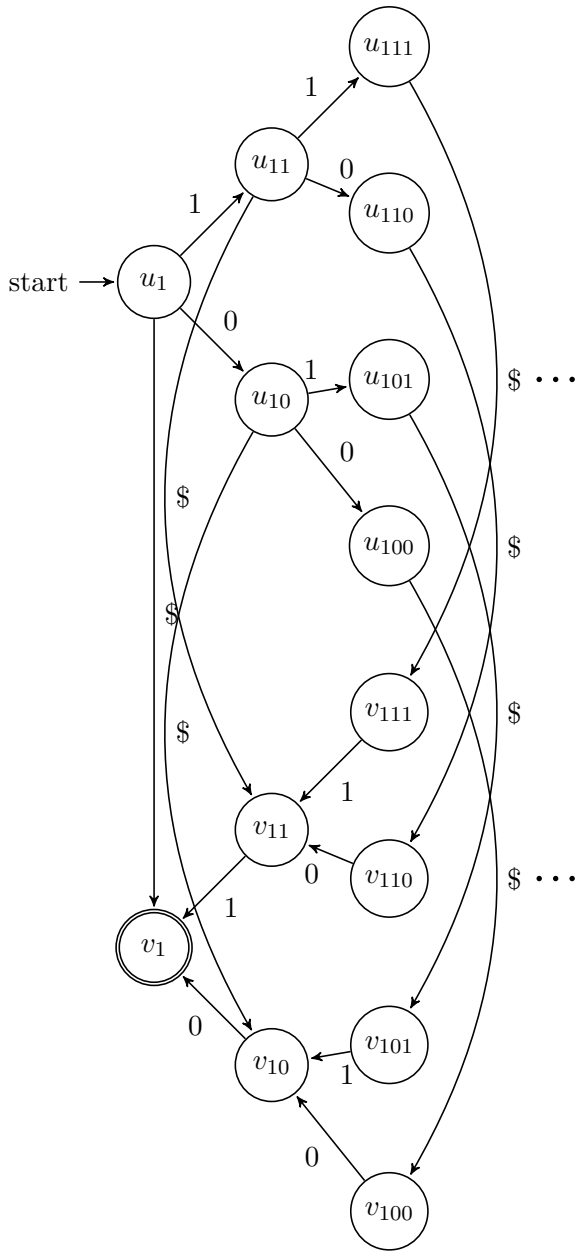
$$l = \sum_{j=0}^{k-1} i_j m^j + m^k$$

codiert und im Automaten durch den Zustand z_l repräsentiert:

$M = (Z, \Sigma, \delta, u_1, E)$
$Z = \{u_1, u_2, u_3, ..., v_1, v_2, v_3, ...\}$
$\delta(u_k, \alpha_i) = u_{m \cdot k + i}$ für alle i, k
$\delta(u_k, \$) = v_k$ für alle k
$\delta(v_k, \alpha_i) = v_{\lfloor \frac{k}{m} \rfloor}$ falls $k \equiv i \mod m$
$S = \{u_1\}$
$E = \{v_1\}$

Dieser Automat akzeptiert L, ist jedoch nicht endlich, somit kein NFA. Umseitig ist der Graph eines solchen Automaten exemplarisch gezeichnet.

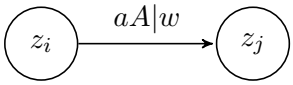
Für $m = 2$, $\alpha_0 = 0$, $\alpha_1 = 1$ und Indices k binär ergibt sich folgender nicht-endlicher Automat:



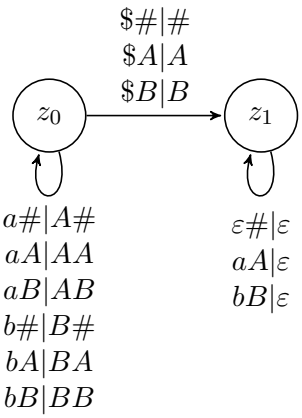
Nach dem Pumping-Lemma ist L nicht regulär!
Eine kontextfreie Grammatik hingegen lässt sich leicht dafür angeben:
$G : S \to \$,\ S \to \alpha_i S \alpha_i$ für $i = 0, ..., m-1$

Zustandsgraphen

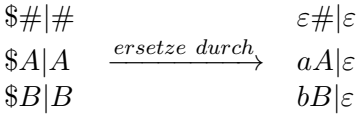
Für Kellerautomaten vereinbaren wir folgende Notation für Zustandsgraphen:
$\delta(z_i, w, A) \ni (z_j, w)$ wird gezeichnet als

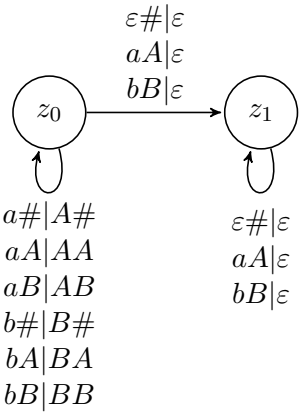


Beispiel.



Dieser (deterministische) Kellerautomat akzeptiert die Sprache $L = \{a_1, \ldots, a_n, \$a_n \ldots a_1\}$ mit $a_i \in \{a, b\}, n \geq 0$.
Modifikation:





Dieser (nicht-deterministische) Kellerautomat akzeptiert $L = \{a_1 ... a_n a_n ... a_1 | a_i \in \{a, b\}, n \geq 0\}$
Nichtdeterminismus wird an dieser Stelle benötigt, um die Mitte des Palindroms zu „raten“ .

Teil II.
Berechenbarkeitstheorie

Der Gödelsche Satz

Weltweit kürzester "Beweis" von Gödel's Satz: Eine informelle Begründung des Unvollständigkeitssatzes von Gödel wurde von Raymond Smallyan gegeben. Der Satz von Gödel sagt aus, dass es keine (Turing-) Maschine geben kann, die nacheinander alle wahren Aussagen ausgibt. Das lässt sich wie folgt begründen:
Sei M eine Maschine die Aussagen druckt.

z.B.:	X	(eine Aussage, z.B. die wahre Arithmetische Formel $(x+1)^2 = x^2 + 2x + 1$)
	$P * X$	(die Aussage, dass M X ausgeben wird)
	$NP * X$	(die Aussage, dass M X niemals ausgibt)
	$PR * X$	(die Aussage, dass M XX ausgeben wird)
	$NPR * X$	(die Aussage, dass M XX nicht ausgeben wird)

z.B. $NPR * FOO$ bedeutet: M gibt niemals $FOOFOO$ aus.

Betrachte Aussage $NPR * NPR*$, d.h. M druckt niemals $NPR * NPR*$ aus. Wenn M $NPR * NPR*$ ausgibt, dann hat sie eine *falsche* Aussage gemacht. Wenn M $NPR * NPR*$ nicht ausgibt, dann ist $NPR * NPR*$ eine *wahre* Aussage, die M niemals ausgibt. Also gimt M entweder ein falsches Statement aus, oder es bleiben wahre Aussagen, die M niemals ausgeben kann. Eine Maschine die nur wahre Aussagen ausgibt, kann folglich nicht alle wahren Aussagen ausgeben.

Teil III.
Komplexitätstheorie

NP-Vollständigkeit

Lemma
Die Relation $\leq_p$ ist transitiv.

Beweis. Zu zeigen: $L_1 \leq_p L_2 \wedge L_2 \leq_p L_3 \Rightarrow L_1 \leq_p L_3$
Sei also

$f_1 : \Sigma_1^* \to \Sigma_2^*$, total, berechenbar in polynomialer Zeit p_1
$f_2 : \Sigma_2^* \to \Sigma_3^*$, total, berechenbar in polynomialer Zeit p_2

mit $x \in L_1 \Leftrightarrow f_1(x) \in L_2$ und $y \in L_2 \Leftrightarrow f_2(x) \in L_3$

Setze $f_3 = f_2 \circ f_1 : \Sigma_1^* \to \Sigma_3^*$.

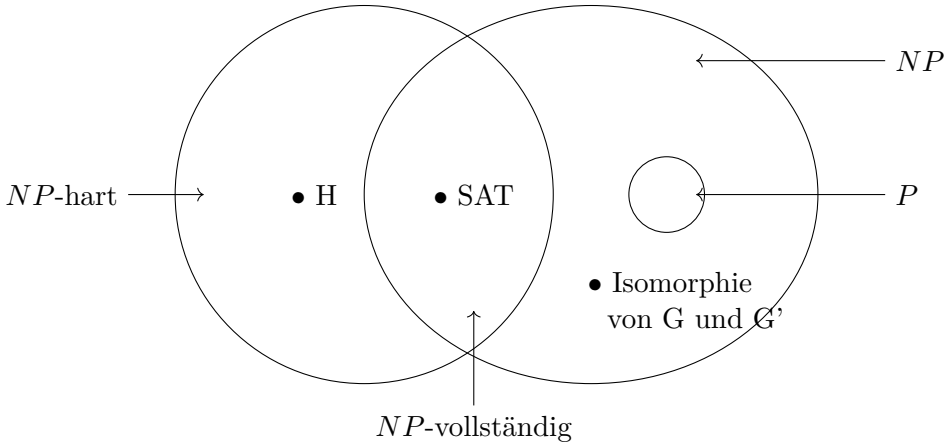
Dann ist die Ausgabelänge $|f_1(x)|$ höchstens so groß wie die Anzahl der Rechenschritte, $p_1(|x|)$.
Damit ist $|f_1(x)| \leq p_1(|x|)$

f_3 berechenbar in Zeit $p_1(|x|) + p_2(|f_1(x)|) \leq p_1(|x|) + p_2(p_1(|x|))$ und

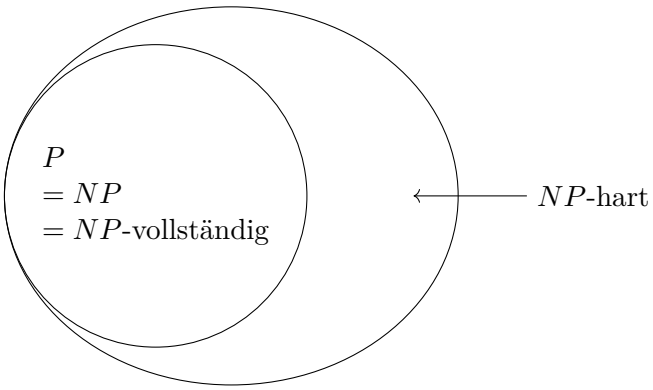
$$x \in L_1 \Leftrightarrow f_1(x) \in L_2 \Leftrightarrow f_2(f_1(x)) \in L$$

□

Komplexitätsklassen-Überblick



Fall $P \neq NP$



Fall $P = NP$

4. Weitere NP-vollständige Probleme

Mit dem „guess-and-check“ Argument weist man nach, dass ein Problem in NP ist.
Wir zeigen das nun ausführlich am Beispiel von `RUCKSACK`:

Gegeben: $a_1, ..., a_n, b \in \mathbb{N}$
Gesucht: Gibt es $J \subseteq \{1, ..., n\}$ mit $\sum_{k \notin J} a_k = b$?
Es gibt 2^n Teilmengen $J \subseteq \{1, ..., n\}$, somit 2^n Lösungskandidaten.
Eine Codierung ist gegeben durch: J_w, $w \in \{0,1\}^n$ mit $k \in J :\Leftrightarrow w_k = 1$

Eine deterministische Turingmaschine M prüft alle 2^n Teilmengen in Zeit $O(2^n)$. Die Rechenzeit ist exponentiell!
Eine nicht-deterministische Turingmaschine findet eine akzeptierende Rechnung in polynomialer Zeit, sofern das Problem überhaupt eine Lösung besitzt.
Codierung von J_w, $w \in \{0,1\}^n$ mit $K \in J :\Leftrightarrow w_k = 1$

1. Rate $w_k \in \{0,1\}$, $k = 1, ..., n$ in $O(n)$ Schritten (setze $w_k = 0$ oder $w_k = 1$).
2. Prüfe den Lösungskandidaten J_w in $O(n)$ Schritten:
   Falls $\sum_{k \in J} a_k = b$, stoppe, sonst gehe zurück zu Step 1.

Wenn das Problem eine Lösung hat, dann gibt es eine akzeptierende Rechnung mit $O(n)$ Schritten. Es gilt also `RUCKSACK` $\in NP$