Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Christian Ivicevic

Sommersemester 2015
Studentisches Tutorium
Übungsblatt 4
11. Mai 2015

Theoretische Informatik

– Pumping-Lemma für reguläre Sprachen, Quotientenautomat –

Im Laufe der ersten Wochen wurde vorgestellt, dass eine Sprache regulär ist, sofern ein zugehöriger DFA konstruiert werden kann. So kann man einen regulären Ausdruck angeben, der eine Sprache beschreibt und mithilfe der Thompson-Konstruktion, des Sättigungsalgorithmus und des Myhill-Verfahren einen passenden DFA herleiten. Solche Beweise können auch etwas komplexer sein. So kann zu zwei regulären Sprachen L_1, L_2 eine Kombination derer betrachtet werden, wie z.B. das Reißverschlussprodukt $L_1 \% L_2$ oder die Linksquotientensprache L_1 / L_2. Um nun zu beweisen, dass diese neu konstruierten Sprachen regulär sind reicht es, informell einen neuen Automaten zu beschreiben, der sich aus den Automaten zu L_1 und L_2 zusammensetzt. Meist ist es aber deutlich interessanter zu beweisen, dass eine Sprache nicht regulär ist.

Pumping-Lemma für reguläre Sprachen

Wir betrachten eine reguläre Sprache L mit dem zugehörigen DFA A, für den $L(A) = L$ gilt. Sofern nun ein Wort $z \in L$ vom Automaten eingelesen wird, so wird ein gewisser Kantenzug durchlaufen bis der Automat das Wort akzeptiert. Wählen wir nun ein Wort, dass lang genug ist, genauer gesagt länger als die Anzahl an Zuständen im Automaten $|Q| = n$, i.Z. $|z| \geq n$, so folgt aus dem Schubfachprinzip, dass der beschriebene Kantenzug durch den Automaten einen Kreis enthalten muss, da mehr Symbole eingelesen werden als Zustände vorhanden sind. Zerlegen wir nun das gegebene Wort in drei Teile, i.Z. $z = uvw$, so analysieren wir das nicht-leere Teilwort v, das durch den Kreis erzeugt werden soll. Nun folgt, dass der Kreis beliebig oft durchlaufen werden kann und für jedes $i \in \mathbb{N}_0$ das Wort uv^iw ebenso in der Sprache enthalten sein muss. Diese Eigenschaft wird von jeder (unendlichen) regulären Sprache erfüllt, was uns ermöglicht mit einer Umkehrung zu arbeiten und zu beweisen, dass eine gegebene Sprache nicht regulär ist.

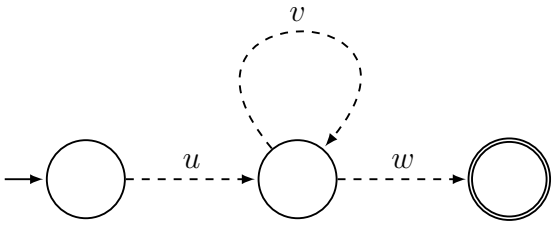


Abbildung 1: Graphische Interpretation des Pumping-Lemma

Definition (Pumping-Lemma)**.** *Sei L eine reguläre Sprache, dann existiert eine* Pumping-Lemma-Zahl $n \in \mathbb{N}_0$ *für alle ausreichend langen Wörter $z \in L$ mit $|z| \geq n$, die sich in Teilwörter u, v, w zerlegen lassen mit $z = uvw, v \neq \varepsilon, |uv| \leq n$, so dass für alle $i \in \mathbb{N}_0$ das Wort z aufpumpbar ist und $uv^i w \in L$ gilt, i.Z.*

$$\exists n \in \mathbb{N}_0 \, \forall z \in L, |z| \geq n \, \exists u, v, w : z = uvw, v \neq \varepsilon, |uv| \leq n \, \forall i \in \mathbb{N}_0 : uv^i w \in L.$$

Nun betrachten wir die Umkehrung des Pumping-Lemmas. Sei nun $n \in \mathbb{N}_0$ eine beliebige, aber feste Pumping-Lemma-Zahl zu einer Sprache L. Sofern ein ausreichend langes Wort $z \in L$ mit $|z| \geq n$ existiert, dessen Zerlegungen $z = uvw$ sich allesamt nicht aufpumpen lassen kann die Sprache L nicht regulär gewesen sein, i.Z.

$$\forall n \in \mathbb{N}_0 \, \exists z \in L, |z| \geq n \, \forall u, v, w : z = uvw, v \neq \varepsilon, |uv| \leq n \, \exists i \in \mathbb{N}_0 : uv^i w \notin L.$$

Zum Beweis, dass eine gegebene Sprache nicht regulär ist, nutzt man einen Widerspruchsbeweis. Hierzu ist der erste Teil immer identisch - die Schwierigkeit liegt in der Suche nach sämtlichen Zerlegungen. Hierzu ist es für uns ausschließlich von Interesse, aus welchen Symbolen sich das Teilwort v zusammensetzt, nicht jedoch die konkrete Anzahl. Die Teilwörter u und w sind implizit ableitbar. Mit diesem sehr wichtigen Hinweis werden wir nun zahlreiche Beispiele zur Anwendung des Pumping-Lemmas betrachten.

Beispiel 1. Einführend wollen wir beweisen, dass $L_1 = \{a^k b^k \mid k \in \mathbb{N}_0\}$ nicht regulär ist. Dies beweisen wir durch Widerspruch und nehmen an, dass L_1 doch regulär ist und $n \in \mathbb{N}_0$ eine beliebige, aber feste, Pumping-Lemma-Zahl (PLZ) für L_1 ist. Dann existiert ein Wort $z = a^n b^n \in L_1$ mit $|z| \geq n$. Nun ist zu zeigen, dass sich **keine** Zerlegung aufpumpen lässt. Schematisch sieht das betrachtete Wort wie folgt aus:

$$\underbrace{\overbrace{aaa \ldots aaa}^{n \times a}}_{1 \leq |uv| \leq n} b^n$$

Mit der untenstehenden Klammer sieht man, dass ausschließlich a's im Teilwort v enthalten sein können und unter keinen Umständen ein b enthalten sein kann. Somit ist nur eine Zerlegung vorhanden, mit der wir weiter arbeiten.

- Zerlegung 1: $u = a^{n-\ell}, v = a^\ell, w = b^n$. Wie oben festgestellt, kann v ausschließlich aus a's bestehen, muss aber per Definition ($v \neq \varepsilon$) mindestens eines beinhalten, somit gilt $\ell \geq 1$. Auch wenn es eine weitere Zerlegung gibt, in der w einige a's besitzt, so ist dies für uns nicht von Interesse, da wir uns ausschließlich auf die Zusammensetzung von u und v beziehen und es irrelevant ist, ob das Teilwort w noch a's besitzt, da v stets nur aus a's besteht. Nun müssen wir ein $i \in \mathbb{N}_0$ finden, so dass $uv^i w \notin L_1$ gilt. Wählen wir $i = 0$ so gilt

  $$uv^i w = uw = a^{n-\ell} b^n \notin L_1,$$

  da $\ell \geq 1$ und mindestens ein Zeichen gelöscht wird, weshalb $n - \ell \neq n$ gilt. Dies steht im Widerspruch zur Annahme, dass L_1 regulär sei!

Somit haben wir für jede mögliche Zerlegung (auch wenn es nur eine ist) gezeigt, dass diese nicht aufpumpbar ist und L_1 im Widerspruch zur Annahme nicht regulär sein kann.

Beispiel 2. Nun betrachten wir $L_2 = \{ab^{2k}cd^ke \mid k \in \mathbb{N}_0\}$ und wollen beweisen, dass diese Sprache nicht regulär ist. Hierzu nehmen wir erneut an, dass L_2 regulär sei und führen dies zum Widerspruch. Weiterhin sei $n \in \mathbb{N}_0$ eine beliebige, aber feste, PLZ für L_2. Dann existiert ein nicht aufpumpbares $z = ab^{2n}cd^ne \in L_2$ mit $|z| \geq n$. Nun ist zu zeigen, dass sich **keine** Zerlegung aufpumpen lässt. Schematisch sieht das betrachtete Wort wie folgt aus:

$$\underbrace{a}_{1 \leq |uv| \leq n} \overbrace{bbb \ldots \ldots bbb}^{2n \times b} cd^ne$$

Mit der untenstehenden Klammer sieht man, dass es zunächst drei Zerlegungen durch drei verschiedene Belegungen von v gibt. Das Teilwort v kann entweder nur aus a, aus a und b's oder nur aus b's bestehen.

- Zerlegung 1: $u = \varepsilon, v = a, w = b^{2n}cd^ne$. Nun müssen wir ein $i \in \mathbb{N}_0$ finden, so dass $uv^iw \notin L_2$ gilt. Wählen wir $i = 0$ so gilt

  $$uv^iw = uw = b^{2n}cd^ne \notin L_2,$$

  da das Wort nicht mit a beginnt. Dies steht im Widerspruch zur Annahme, dass L_2 regulär sei!

- Zerlegung 2: $u = \varepsilon, v = ab^\ell, w = b^{2n-\ell}cd^ne$. Da $v \neq \varepsilon$ gelten muss, ist $\ell \geq 0$. Aus $|uv| \leq n$ folgt $\ell \leq n - 1$. Nun müssen wir ein $i \in \mathbb{N}_0$ finden, so dass $uv^iw \notin L_2$ gilt. Wählen wir $i = 0$ so gilt

  $$uv^iw = uw = b^{2n-\ell}cd^ne \notin L_2,$$

  da das Wort erneut nicht mit a a beginnt. Dies steht im Widerspruch zur Annahme, dass L_2 regulär sei!

  An dieser Stelle sei zu erwähnen, dass geschickterweise Zerlegung 1 durch Zerlegung 2 mit $\ell = 0$ hätte abgedeckt werden können, es aber nicht falsch ist dennoch jede redundante Zerlegung explizit zu betrachten. Sofern man solche Analogien erkennt, kann man sich deutlich Arbeit ersparen.

- Zerlegung 3: $u = a, v = b^\ell, w = b^{2n-\ell}cd^ne$. Aus $v \neq \varepsilon$ folgt $\ell \geq 1$ und mit $|uv| \leq n$ folgt $\ell \leq n - 1$. Auch wenn u theoretisch die Form ab^k haben könnte so ändert dies nichts an der Struktur von v - ausschließlich an der Anzahl an b's, was irrelevant ist, mithin sind beide Fälle identisch. Nun müssen wir ein $i \in \mathbb{N}_0$ finden, so dass $uv^iw \notin L_2$ gilt. Wählen wir $i = 0$ so gilt

  $$uv^iw = uw = ab^{2n-\ell}cd^ne \notin L_2,$$

  da $2n - \ell \neq 2n$ gilt aufgrund von $\ell \geq 1$, somit können nicht doppelt so viele b's wie d's vorhanden sein Dies steht im Widerspruch zur Annahme, dass L_2 regulär sei!

Somit haben wir für jede mögliche Zerlegung gezeigt, dass diese nicht aufpumpbar ist und L_2 im Widerspruch zur Annahme nicht regulär sein kann.

Beispiel 3. Wir werden beweisen, dass $P = \{1^p \mid p \text{ prim}\}$ nicht regulär ist. Sei P regulär und n eine PLZ. Wir betrachten $z = 1^p$ mit p prim und $p \geq n$ - es gibt lediglich die Zerlegung $v = 1^\ell$ mit $\ell \geq 1$, da $v \neq \varepsilon$. Um zu zeigen, dass z nicht aufpumpbar ist wählen wir geschickt $i = |u| + |v| + |w| + 1$ und beweisen dies, indem wir zeigen, dass die Länge $|uv^iw|$ nicht prim ist. Es gilt

$$|uv^iw| = |u| + (|u| + |v| + |w| + 1)\,|v| + |w| = (|v| + 1)\,(|u| + |v| + |w|)$$

und somit ist die Länge keine Primzahl und das aufgepumpte Wort nicht in der Sprache enthalten. Somit haben wir für jede mögliche Zerlegung gezeigt, dass diese nicht aufpumpbar ist und P im Widerspruch zur Annahme nicht regulär sein kann.

Beispiel 4. Nun betrachten wir die Sprache $L_3 = \{a^\ell \mid \ell = 2^k, k \in \mathbb{N}_0\}$ und werden beweisen, dass diese nicht regulär ist, indem wir erneut mit der Wortlänge argumentieren. Wir nehmen nun an, dass L_3 regulär sei und n eine beliebige, aber feste PLZ. Wir wählen $z = a^{2^n}$ mit $v = a^\ell$. Aus $v \neq \varepsilon$ und $|uv| \leq n$ folgt erneut $1 \leq \ell \leq n$. Mit $i = 0$ ist zu zeigen, dass

$$2^{n-1} \overset{(2)}{<} |uv^iw| \overset{(1)}{<} 2^n \implies a^{2^n-\ell} \notin L_3.$$

Wir zeigen also, dass das aufgepumpte Wort stets zwischen zwei Zweierpotenzen liegt.

(1) $|uv^iw| = |uv^0w| = 2^n - \ell < 2^n$ folgt aus $\ell \geq 1$.

(2) $2^{n-1} < |uv^iw| = |uv^0w| = 2^n - \ell$ lösen wir nach ℓ auf und kombinieren mit $\ell \leq n$. Aus $\ell < 2^n - 2^{n-1} = 2^{n-1}$ folgt $\ell \leq n < 2^{n-1}$ was für alle $n \geq 3$ gilt. Wir haben nun gezeigt, dass für fast alle n und genügend lange Wörter diese aufgepumpt nicht die Länge einer Zweierpotenz haben.

Somit haben wir für jede mögliche Zerlegung gezeigt, dass diese nicht aufpumpbar ist und L_3 im Widerspruch zur Annahme nicht regulär sein kann.

Beispiel 5. Nun betrachten wir die Sprache $L_4 = \{a^\ell \mid \ell = k^2, k \in \mathbb{N}_0\}$ und werden beweisen, dass diese nicht regulär ist, indem wir erneut mit der Wortlänge argumentieren. Wir nehmen nun an, dass L_4 regulär sei und n eine beliebige, aber feste PLZ. Wir wählen $z = a^{n^2}$ mit $v = a^\ell$. Aus $v \neq \varepsilon$ und $|uv| \leq n$ folgt erneut $1 \leq \ell \leq n$. Mit $i = 0$ ist zu zeigen, dass

$$(n-1)^2 \overset{(2)}{<} |uv^iw| \overset{(1)}{<} n^2 \implies a^{n^2-\ell} \notin L_4.$$

Wir zeigen also, dass das aufgepumpte Wort stets zwischen zwei Quadratzahlen liegt.

(1) $|uv^iw| = |uv^0w| = n^2 - \ell < n^2$ folgt aus $\ell \geq 1$.

(2) $(n-1)^2 < |uv^iw| = |uv^0w| = n^2 - \ell$ lösen wir nach ℓ auf und kombinieren mit $\ell \leq n$. Aus $\ell < 2n - 1$ folgt $\ell \leq n < 2n - 1$ was für alle $n \geq 2$ gilt. Wir haben nun gezeigt, dass für fast alle n und genügend lange Wörter diese aufgepumpt nicht die Länge einer Quadratzahl haben.

Somit haben wir für jede mögliche Zerlegung gezeigt, dass diese nicht aufpumpbar ist und L_4 im Widerspruch zur Annahme nicht regulär sein kann.

Minimierung von Automaten

In diesem Abschnitt werden wir uns mit der Frage beschäftigen, wie ein Automat minimiert werden kann, indem man äquivalente Zustände findet und diese vereint. Hierzu beziehen wir uns auf ein Konzept welches in den ersten Semestern schon vorgestellt wurde, den *Äquivalenzrelationen*. Dabei handelt es sich um Relationen, die reflexiv, transitiv und symmetrisch sind. Mit einer solchen Äquivalenzrelation kann eine gegebene Menge partitioniert und Gemeinsamkeiten zwischen Elementen beschrieben werden.

Definition (Äquivalenzklasse)**.** *Sei* $\equiv \subset \Omega \times \Omega$ *eine* Äquivalenzrelation *über der Menge* Ω*, dann schreiben wir* $x \equiv y$ *für zwei Elemente* $x, y \in \Omega$*, falls diese* äquivalent *sind. Zu jedem* $x \in \Omega$ *ist* $[x] := \{y \in \Omega \mid x \equiv y\}$ *die zugehörige* Äquivalenzklasse *und wir bezeichnen ein* $y \in [x]$ *als* Repräsentant *dieser Klasse.*

Beispiel 6. Sei $\Omega = \{1, 2, 3, a, b, c, +, -, \cdot\}$, dann lässt sich Ω in 3 Äquivalenzklassen partitionieren je nach Typ des Elements (Ziffer, Buchstabe, Operation):

- $[1] = [2] = [3] = \{1, 2, 3\}$
- $[a] = [b] = [c] = \{a, b, c\}$
- $[+] = [-] = [\cdot] = \{+, -, \cdot\}$

Definition (Äquivalenz von Zuständen)**.** *Zu einem gegebenen DFA* A *bezeichnen wir zwei Zustände* $p, q \in Q$ *als* äquivalent*, wenn von den Zuständen aus die selbe Sprache akzeptiert wird:*

$$p \equiv_A q \Leftrightarrow \left(\forall w \in \Sigma^* \,:\, \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F\right)$$

Beispiel 7. Zu beweisen, dass zwei Zustände äquivalent sind ist meist etwas involvierter, daher schauen wir uns die einfachere Variante an, wie man beweist, dass zwei Zustände nicht äquivalent sind. Hierzu reicht es aus, ein Wort $w \in \Sigma^*$ zu finden, so dass ein Zustand p in einen Endzustand führt und ein Zustand q eben nicht. In einem solchen Fall sagen wir, dass wir die Zustände p und q mit dem Wort w unterscheiden können und diese nicht äquivalent sein können. Der Einfachheit halber reduzieren wir dies meistens auf das Testen mit nur einem Buchstaben mit dem wir zwei Zustände unterscheiden können. Wir können im Automaten aus Abbildung 2 ablesen, dass $q_1 \not\equiv_A q_2$ gilt, da $\delta(q_1, b) = q_3 \in F$ und $\delta(q_2, b) = q_2 \notin F$ gilt. Die beiden Zustände werden hierbei durch b unterschieden.

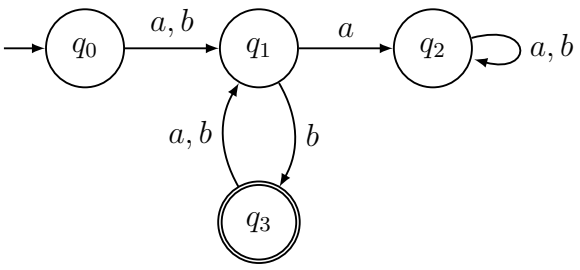
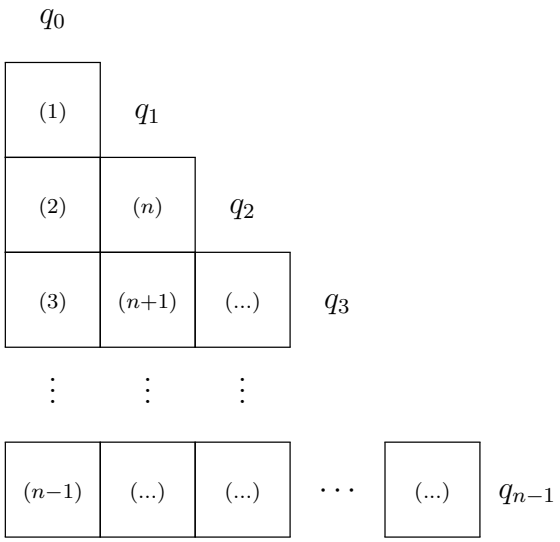


Abbildung 2: Beispielautomat zu Äquivalenz

Minimierungsalgorithmus

Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein zu minimierender DFA mit $|Q| = n$ Zuständen. Zur Minimierung nutzen wir eine Tabelle in Treppenform mit $n - 1$ Stufen. Der Algorithmus basiert auf dynamischer Programmierung und wird im Laufe mehrerer Iterationen durch die Tabelle berechnen welche Zustandspaare unterscheidbar sind. Die zugehörige Tabelle sieht wie folgt aus:



Schritt 1 Bei der Initialisierung werden sämtliche Zustandspaare $\{q_i, q_j\}$ markiert, die direkt unterscheidbar sind, d.h. für die $q_i \in F$ und $q_j \notin F$ gilt.

Schritt 2 Die Tabelle wird in diesem Schritt mehrfach von oben nach unten und links nach rechts durchlaufen, was durch die kleinen Zahlen in der Tabelle hervorgehoben ist. Ein Zustandspaar $\{q_i, q_j\}$ wird mit n_s markiert, falls in der n-ten Iteration abgelesen wird, dass $\{\delta(q_i, s), \delta(q_j, s)\}$ bereits markiert ist, d.h. q_i und q_j mit dem Symbol $s \in \Sigma$ unterscheidbar sind.

Schritt 3 Sofern sich die Tabelle in einer Iteration nicht mehr ändert, terminiert die Iteration. Für alle nun unmarkierten Zustandspaare $\{q_i, q_j\}$ gilt $q_i \equiv_A q_j$ und diese Zustände können zu einer Äquivalenzklasse kollabiert werden.

Schritt 4 Abschließend kann der *Quotientenautomat* $A / \equiv = (Q / \equiv, \Sigma, \delta', [q_0]_\equiv, F / \equiv)$ konstruiert werden. Die Zustandsmenge $Q / \equiv$ entspricht allen Äquivalenzklassen, die in Schritt 3 aus der Tabelle abgelesen wurden. Die Zustandsübergangsfunktion δ' ist durch $\delta'([q]_\equiv, s) = [\delta(q, s)]_\equiv$ festgelegt und kann somit direkt aus dem ursprünglichen Automaten A abgelesen werden. Weiterhin gilt $F / \equiv = \{[q]_\equiv \mid q \in F\}$, was analog aus dem alten Automaten abgelesen werden kann.

Beispiel 8. Wir werden den Minimierungsalgorithmus auf den Automaten aus Abbildung 3 anwenden. Im ersten Schritt werden die q_4 und q_6-Zeilen und Spalten markiert bis auf $\{q_4, q_6\}$. Diese Initialisierung ist in Abbildung 4 den mit $\times$ beschrifteten Zellen zu entnehmen. Beginnt man in $\{q_0, q_1\}$ und betrachtet mit dem Symbol a das Zustandspaar $\{\delta(q_0, a), \delta(q_1, a)\} = \{q_1, q_2\}$ so ist dieses nicht markiert und es lässt sich (noch) keine Aussage über die Äquivalenz von q_0 und q_1 machen. Mit dem Symbol b betrachtet man jedoch das schon markierte Paar $\{q_4, q_5\}$. Somit sind q_0 und q_1 mit b im ersten Iterationsschritt unterscheidbar und nicht äquivalent. Dies wird mit 1_b in der Tabelle notiert. Nachdem die Tabelle nach diesem Schema einmal komplett durchlaufen wurde erhalten wir die Tabelle aus Abbildung 5. Iteriert man ein zweites Mal über die Tabelle findet man heraus, dass das Paar $\{q_0, q_2\}$ auf Basis der vorherigen Ergebnisse nun im zweiten Schritt mit dem Symbol a unterscheidbar sind, was mit 2_a notiert wird. In einer dritten Iteration ändert sich nichts und wir erhalten das finale Ergebnis aus Abbildung 6.

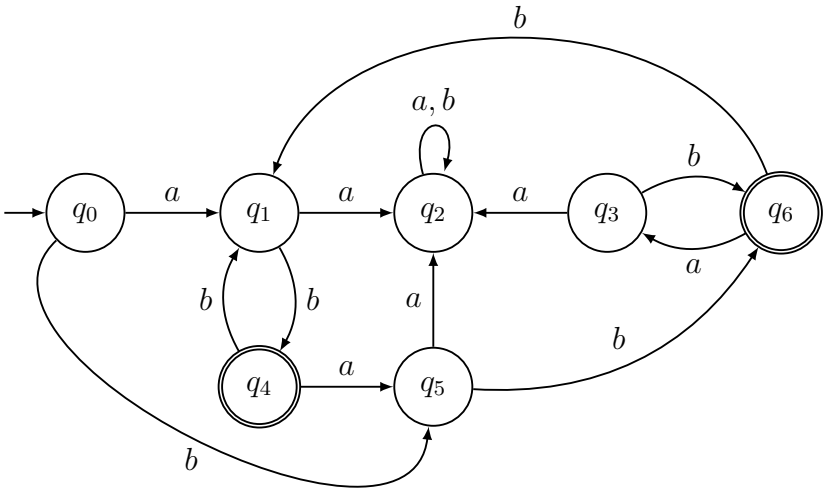


Abbildung 3: Beispielautomat zur Minimierung

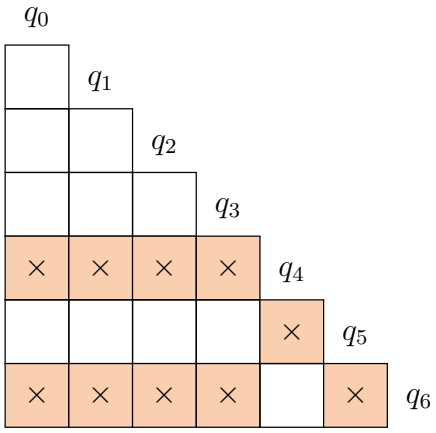


Abbildung 4: Minimierungs-Tabelle zum Beispielautomaten nach der Initialisierung

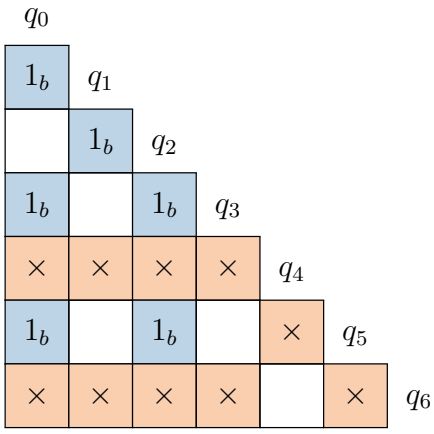


Abbildung 5: Minimierungs-Tabelle zum Beispielautomaten nach der ersten Iteration

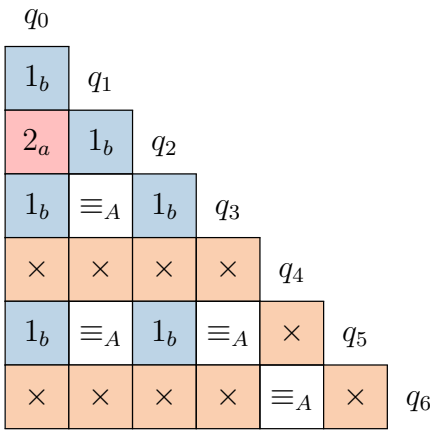


Abbildung 6: Minimierungs-Tabelle zum Beispielautomaten nach Ende der Iteration

Aus Abbildung 6 lässt sich ablesen, dass $q_1 \equiv_A q_3 \equiv_A q_5$ und $q_4 \equiv_A q_6$ gelten. Somit sind ebenfalls die vier Äquivalenzklassen konkret ablesbar und es gilt:

$$\begin{aligned} Q / \equiv = \{&[q_0]_{\equiv_A} = \{q_0\}, \\ &[q_1]_{\equiv_A} = \{q_1, q_3, q_5\}, \\ &[q_2]_{\equiv_A} = \{q_2\}, \\ &[q_4]_{\equiv_A} = \{q_4, q_6\}\} \end{aligned}$$

Der zugehörige Quotientenautomat ist schlussendlich in Abbildung 7 zu sehen.

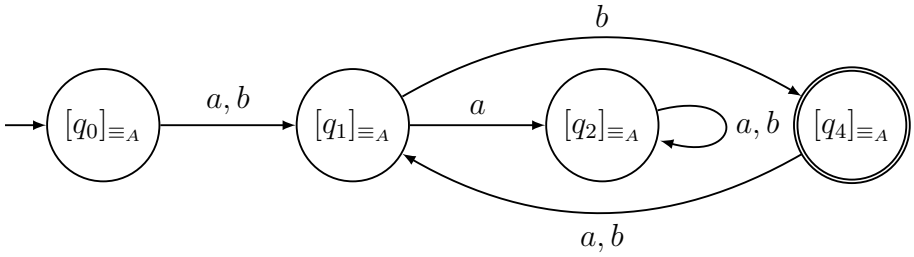


Abbildung 7: Finaler Quotientenautomat