Inhaltsverzeichnis

1 Zusatzmaterial: Automaten

Wir führen „Automaten mit Ausgaben“, die sogenannten Moore- und Mealy-Automaten, im Abschnitt 1.1 ein. Dieses Automatenmodell ist für den Entwurfsprozess von Schaltwerken von großer Wichtigkeit.

Um nicht-triviale Schaltwerke auch „bauen“ zu können, sollten wir kleine Automaten entwerfen, bzw. besser noch: Wir sollten einen vorgegebenen Automaten effizient minimieren können, also einen äquivalenten Automaten mit kleinstmöglicher Zustandszahl angeben können. Genau das machen wir im Abschnitt 1.2.

1.1 Automaten mit Ausgabe

Endliche Automaten spielen auch in der technischen Informatik eine wichtige Rolle und zwar im Entwurfsprozess von Schaltwerken. Bevor wir das Konzept der Schaltwerke einführen, beschreiben wir das einfachere Konzept eines Schaltnetzes.

Schaltnetz

Ein **Schaltnetz** ist ein „Netz“ aus elementaren logischen Gattern, aufgebaut aus Transistoren. Wir begnügen uns mit dieser sehr oberflächlichen Beschreibung und verweisen für eine ausführlichere Darstellung auf die Veranstaltung „Hardwarearchitekturen und Rechensysteme“ im zweiten Semester. Für uns genügt das Wissen, dass ein Schaltnetz ein Tupel

$$f : \{0,1\}^n \to \{0,1\}^m$$

boolescher Funktionen[1] berechnet. Schaltnetze können z.B. arithmetische Funktionen berechnen wie etwa die Addition oder Multiplikation von zwei Zahlen, wenn jede der beiden Eingabezahlen eine Binärdarstellung der jeweiligen Länge $n/2$ besitzt.

Betrachten wir zuerst eine einzige boolesche Funktion, wir nehmen also $m = 1$ an und das Schaltnetz berechnet die boolesche Funktion

$$f : \{0,1\}^n \to \{0,1\}.$$

Wir berechnen f mit einem endlichen Automaten

$$A_f = (\Sigma, Q, \delta, q_0, F).$$

Wir machen uns das Leben einfach und wählen das Eingabealphabet $\Sigma = \{0,1\}^n$. Der (partiell definierte) Automat A_f startet im Zustand ε, liest die gesamte Eingabe x in einem Schritt und wechselt dann entweder in den Zustand 0 (es ist $f(x) = 0$) oder in den Zustand 1 (es ist $f(x) = 1$). Wir beschreiben die einzelnen Komponenten von A_f:

1. A_f besitzt das Eingabealphabet $\Sigma = \{0,1\}^n$,
2. die Zustandsmenge $Q = \{\, \varepsilon \,, 0 \,, 1 \}$ mit Startzustand $q_0 = \varepsilon$,

[1] Wir erinnern uns, dass eine boolesche Funktion $g : \{0,1\}^m \to \{0,1\}$ jedem binären Wort der Länge m eine Null, bzw. eine Eins zuweist.

3. die (partiell definierte) Übergangsfunktion $\delta : Q \times \Sigma \to Q$ mit

$$\delta(\varepsilon, x) = f(x)$$

4. und die Menge $F = \{ 1 \}$ akzeptierender Zustände.

Endliche Automaten können „nur“ akzeptieren oder verwerfen. Um ein Tupel boolescher Funktionen zu berechnen, benötigen wir ein Automatenmodell mit Ausgabe.

1.1.1 Moore Automaten

Definition 1.1. Ein **Moore Automat** Moore Automat

$$(\Sigma, Q, \delta, q_0, \lambda, \Omega, F)$$

ist wie ein DFA aufgebaut, besitzt aber zusätzlich

- ein Ausgabealphabet Ω und
- eine Ausgabefunktion $\lambda : Q \to \Omega$.

Um ein Zustandsdiagramm eines Moore Automaten $M = (\Sigma, Q, \delta, q_0, \lambda, \Omega, F)$ zu erhalten, erstellen wir zuerst das Zustandsdiagramm des DFA $(\Sigma, Q, \delta, q_0, F)$ und beschriften zusätzlich einen jeden Zustand $q \in Q$ mit der Ausgabe $\lambda(q)$. Der Moore Automat M durchläuft für eine Eingabe $w = a_1 a_2 \cdots a_n$ einen Weg

$$(q_0, q_1, \ldots, q_n) \in Q^{n+1}$$

in seinem Zustandsdiagramm, wobei die Zustandsübergänge genau wie im Fall von DFAs durch die Übergangsfunktion δ definiert sind, d.h. es gilt $q_{i+1} = \delta(q_i, a_{i+1})$ für $i = 0, \ldots, n-1$. Im wesentlichen Unterschied zu DFAs kann der Moore Automat aber eine Folge von Ausgaben produzieren, nämlich für Eingabe w das Tupel

$$\mu = (\lambda(q_1), \ldots, \lambda(q_n)) \in \Omega^n,$$

und wir sagen, dass der Moore Automat die Ausgabe μ berechnet.

Wir nehmen jetzt an, dass ein Schaltnetz das Tupel

$$f : \{0,1\}^n \to \{0,1\}^m$$

boolescher Funktionen für eine beliebige natürliche Zahl $m \in \mathbb{N}$ berechnet und entwerfen einen Moore Automaten

$$A_f = (\Sigma, Q, \delta, q_0, \lambda, \Omega, F),$$

der die Ausgabe $f(x)$ für jede Eingabe $x \in \{0,1\}^n$ berechnet. Auch jetzt machen wir uns das Leben einfach und wählen $\{0,1\}^n$ als Ausgabealphabet. Unser Moore Automat A_f wechselt für Eingabe $x \in \Sigma$ vom Startzustand ε in den Zustand $f(x)$ und kann dann die Ausgabe $f(x)$ geben. Die einzelnen Komponenten von A_f haben die folgende Form:

1. $\Sigma = \{0,1\}^n$ ist das Eingabealphabet,
2. $Q = \{ \varepsilon \} \cup \{0,1\}^m$ ist die Zustandsmenge und $q_0 = \varepsilon$ der Startzustand,

3. $\delta : Q \times \Sigma \to Q$ mit
$$\delta(\varepsilon, x) = f(x)$$
ist die partielle Übergangsfunktion. Beachte, dass δ nur für von ε ausgehende Übergänge definiert ist.

4. $\Omega = \{0,1\}^m$ ist das Ausgabealphabet und

5. $\lambda : Q \to \Omega$ mit $\lambda(q) = q$ ist die Ausgabefunktion.

1.1.2 Mealy Automaten

Schaltwerk

Bisher haben wir nur eine einzige Ausgabe $f(x) \in \Omega$ gegeben. Wenn wir aber ein **Schaltwerk** durch einen Automaten modellieren möchten, müssen wir das ändern. Wie arbeitet ein Schaltwerk? Wie im Fall von Schaltnetzen nehmen wir an, dass $\Sigma = \{0,1\}^n$ das Eingabealphabet und $\Omega = \{0,1\}^m$ das Ausgabealphabet ist.

Wenn ein Schaltwerk ein Eingabetupel $(x_1, \ldots, x_n) \in \Sigma^n$ verarbeitet, dann wird mit Rückkopplung gerechnet. Was genau bedeutet das? Angenommen, r Bits werden rückgekoppelt. Um die Berechnung eines Schaltwerks formal zu definieren, benötigen wir zwei Funktionen, eine **Ausgabefunktion**

Ausgabefunktion

$$f_A : \Sigma \times \{0,1\}^r \to \Omega$$

Rückkopplungsfunktion

und eine **Rückkopplungsfunktion**

$$f_R : \Sigma \times \{0,1\}^r \to \{0,1\}^r.$$

Wenn das Schaltwerk die Eingaben $x_1, \ldots, x_k \in \{0,1\}^n$ verarbeitet, dann werden die

$$\text{Ausgaben } y_1, \ldots, y_k \in \Omega \text{ und Rückkopplungen } r_0 = 0^r, r_1, \ldots, r_{k-1}$$

berechnet, wobei
$$y_i = f_A(x_i, r_{i-1}) \text{ und } r_i = f_R(x_i, r_{i-1})$$
gilt. (Die anfängliche Rückkopplung ist also $r_0 = 0^r$.) Das Schaltwerk berechnet also im $i$ten Schritt die Ausgabe $(f_A(x_i, r_{i-1}), f_R(x_i, r_{i-1}))$ und benutzt den Teil $f_R(x_i, r_{i-1})$ der Ausgabe als Rückkopplung r_i für den $(i+1)$ten Schritt. Das Schaltwerk beschreiben wir durch das Tupel $(\Sigma, r, \Omega, f_A, f_R)$.

Steuerwerk

Bemerkung 1.2. Beispielsweise kann ein **Steuerwerk** durch ein Schaltwerk berechnet werden. Wie arbeitet ein Steuerwerk?

1. Ein Steuerwerk bestimmt den nächsten Befehl in Abhängigkeit vom gegenwärtigen Systemzustand r_i und interpretiert den Befehl. Insbesondere werden Operanden, auf die sich der Befehl bezieht, geladen und Steuersignale an andere Funktionseinheiten (wie etwa das Rechenwerk) erstellt.

2. Der Systemzustand wird aktualisiert, d.h. r_{i+1} wird berechnet.

Mehr über Schaltwerke erfahren Sie in der Vorlesung „Hardwarearchitekturen und Rechensysteme“.

Können wir ein Schaltwerk $(\Sigma, r, \Omega, f_A, f_R)$ mit einem Moore Automaten modellieren? Ja, aber die Modellierung wird einfacher, wenn wir mit Mealy Automaten arbeiten.

Definition 1.3. Ein **Mealy Automat** Mealy Automat

$$(\Sigma, Q, \delta, q_0, \lambda, \Omega, F)$$

ist wie ein DFA aufgebaut, besitzt aber zusätzlich

- ein Ausgabealphabet Ω und
- eine Ausgabefunktion $\lambda : Q \times \Sigma \to \Omega$.

Ein Mealy Automat bestimmt seine nächste Ausgabe also in Abhängigkeit vom aktuellen Zustand *und* dem aktuellen Eingabebuchstaben. Um ein Zustandsdiagramm eines Mealy Automaten $M = (\Sigma, Q, \delta, q_0, \lambda, \Omega, F)$ zu erhalten, erstellen wir wieder zuerst das Zustandsdiagramm des DFA $(\Sigma, Q, \delta, q_0, F)$ und beschriften zusätzlich die Übergange: Wenn $\delta(p, a) = q$, dann beschriften wir die Kante $p \to q$ wie üblich mit der Eingabe a und *zusätzlich* mit der Ausgabe $\lambda(p, a)$.

Der Mealy Automat M wird für eine Eingabe $w = a_1 a_2 \cdots a_n$ eine Zustandsfolge $(q_0, q_1, \ldots, q_n) \in Q^{n+1}$ durchlaufen. Wir sagen, dass M die Ausgabefolge $(\lambda(q_0, a_1), \lambda(q_1, a_2), \ldots, \lambda(q_{n-1}, a_n))$ berechnet.

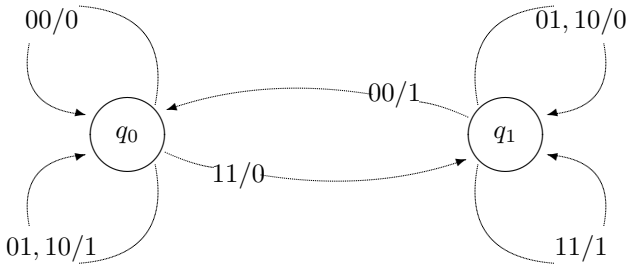
Beispiel 1.4. Wir können jeden Moore Automaten $(\Sigma, Q, \delta, q_0, \lambda_{\text{Moore}}, \Omega, F)$ auch als Mealy Automaten $(\Sigma, Q, \delta, q_0, \lambda_{\text{Mealy}}, \Omega, F)$ auffassen, wenn wir die Ausgabefunktion $\lambda_{\text{Mealy}} : Q \times \Sigma \to \Omega$ durch

$$\lambda_{\text{Mealy}}(p, a) := \lambda_{\text{Moore}}(p)$$

definieren: Im Mealy Automaten benutzen wir die Abhängigkeit vom Eingabesymbol nicht.

Beispiel 1.5. Wir bauen einen Mealy Automaten für die Addition zweier Binärzahlen $x = 0x_n \cdots x_1$ und $y = 0y_n \cdots y_1$. Dazu nehmen wir an, dass der Mealy Automat die Eingabe $x_i y_i$ im $i$ten Schritt erhält. (Beachte, dass 00 die Eingabe im letzten Schritt ist.) Wir wählen deshalb das Eingabealphabet $\Sigma := \{0, 1\}^2$ sowie das Ausgabealphabet $\Omega := \{0, 1\}$.

Das i-te Ausgabe-Bit hängt nur ab von den beiden Eingabebits x_{i-1}, y_{i-1} und dem im vorherigen Schritt evtl. erzeugten Übertrag. Wir „merken uns den Übertrag im Zustand“ (q_0 entspricht dem Übertrag 0, q_1 dem Übertrag 1) und können dann mit Hilfe der beiden neuen Eingabebits das entsprechende Ausgabebit der Summe berechnen. Hier ist das Zustandsdiagramm. (Lese ab/c als: Für Eingabebits ab wird das Bit c ausgegeben.)



Wir möchten ein Schaltwerk $S = (\Sigma, r, \Omega, f_A, f_R)$ durch einen Mealy Automaten simulieren. Die Idee ist sehr einfach, denn das Schaltwerk ist schon ein „verkappter“ Mealy Automat mit Zustandsmenge $Q = \{0,1\}^r$. Insbesondere sollten wir die Rückkopplungsfunktion f_R des Schaltwerks S als Übergangsfunktion unseres Mealy Automaten verwenden, denn der „Systemzustand“ von S wird durch f_R aktualisiert. Die Ausgabefunktion f_A von S berechnet die Ausgabe in Abhängigkeit vom Systemzustand und der aktuellen Eingabe. Wir können also f_A auch als Ausgabefunktion des Mealy Automaten verwenden. Unser Mealy Automat $(\Sigma, Q, \delta, q_0, \lambda, \Omega, F)$ hat die folgenden Komponenten:

1. Das Eingabealphabet Σ und das Ausgabealphabet Ω werden vom Schaltwerk S übernommen.

2. Die Zustandsmenge ist $Q := \{0,1\}^r$ und $q_0 = 0^r$ ist der Anfangszustand,

3. die Übergangsfunktion $\delta : Q \times \Sigma \to Q$ wird durch $\delta = f_R$ und

4. die Ausgabefunktion $\lambda : Q \times \Sigma \to \Omega$ durch $\lambda := f_A$ definiert.

Wir haben ausgenutzt, dass Mealy Automaten die Rückkopplung mit Hilfe ihrer Zustände auf natürliche Art und Weise simulieren können.

Bemerkung 1.6. Der Entwurfsprozess eines Schaltwerks wird grob gesprochen in den folgenden Schritten durchgeführt.

1. Ein Mealy Automat wird zur Lösung eines Teils der Problemstellung entwickelt.

2. Der Automat wird minimiert, d.h. ein äquivalenter Automat mit kleinster Zustandsmenge wird berechnet und seine Automatentabelle wird bestimmt.

3. Mit Verfahren der Logik-Optimierung, wie etwa mit dem Quine/McCluskey Verfahren bestimmt man die minimale Gatterzahl für den restlichen Teil des Schaltwerks. (Das Verfahren von Quine/McCluskey haben wir im Kapitel „Aussagenlogik“ erwähnt.)

Alle Details und viel, viel mehr in der Vorlesung

„Hardwarearchitekturen und Rechensysteme“.

Bemerkung 1.7. Können wir heutige Rechner durch Mealy Automaten modellieren?

„Im Prinzip“ ja: Heutige Rechner besitzen zwar einen modifizierbaren Speicher, aber dieser Speicher ist beschränkt. Was immer ein moderner Rechner berechnen kann, lässt sich auch durch Moore- oder Mealy-Automaten berechnen.

Aber, um k Flip-Flops zu simulieren, brauchen wir Mealy Automaten mit bis zu 2^k Zuständen. Eine Modellierung moderner Rechner durch Mealy Automaten ist völlig unsinning, weil der notwendige Speicher im schlimmsten Fall exponentiell ($\lightning$) anwächst.

Aber Mealy Automaten bieten sich zum Beispiel an, um Steuerwerke zu simulieren. Und dann sollten wir auch mit Ihnen arbeiten, denn wir können Mealy Automaten minimieren wie wir gleich sehen werden.

1.2 Minimierung

Definition 1.8. $A = (\Sigma, Q^A, \delta^A, q_0^A, F^A)$ und $B = (\Sigma, Q^B, \delta^B, q_0^B, F^B)$ seien DFAs.

(a) Wir nennen A und B **äquivalent**, wenn gilt äquivalent

$$\mathbf{L(A) = L(B)}.$$

(b) A heißt **minimal** wenn kein mit A äquivalenter DFA eine kleinere Zustandszahl besitzt. minimal

Äquivalente DFAs haben dasselbe Ausgabeverhalten, können aber völlig unterschiedliche Zustandszahlen besitzen. Für einen gegebenen DFA $A = (\Sigma, Q, \delta, q_0, F)$ suchen wir einen „kleinsten“ mit A äquivalenten DFA. Das Tolle ist die genial einfache Idee der Minimierung:

> Wir sollten doch zwei Zustände $p, q \in Q$ zu einem einzigen Zustand verschmelzen dürfen, wenn p und q „äquivalent“ sind,
>
> also dasselbe Ausgabeverhalten besitzen.

Aber was bedeutet das, dasselbe Ausgabeverhalten zu besitzen?

Definition 1.9. Der DFA $A = (\Sigma, Q, \delta, q_0, F)$ sei gegeben. Die **Verschmelzungsrelation** $\equiv_A$ ist eine 2-stellige Relation über der Zustandsmenge Q. Wir sagen, dass Zustände $p, q \in Q$ äquivalent bzgl. A sind (kurz $p \equiv_A q$), wenn Verschmelzungsrelation

$$\text{f.a. Worte } w \in \Sigma^* : \widehat{\delta}(p, w) \in F \;\Leftrightarrow\; \widehat{\delta}(q, w) \in F.$$

Wir nennen $\equiv_A$ Verschmelzungsrelation, da wir bzgl. $\equiv_A$ äquivalente Zustände in einen Zustand verschmelzen möchten. Die Verschmelzungsrelation hat eine sehr schöne Eigenschaft, sie ist nämlich eine Äquivalenzrelation. Was es damit auf sich hat, sehen wir im nächsten Abschnitt.

1.2.1 Äquivalenzrelationen

(Dieser Abschnitt entspricht dem Abschnitt 4.3.2 des Skripts.)

Jeder gerichtete Graph $G = (V, E)$ lässt sich als eine 2-stellige Relation über der Knotenmenge V auffassen, da die Kantenmenge E von G ja gerade eine Teilmenge von $V^2 = V \times V$ ist. Umgekehrt können wir natürlich auch jede 2-stellige Relation R über einer Menge V als gerichteten Graph mit Knotenmenge V und Kantenmenge R auffassen. Gerichtete Graphen mit Knotenmenge V sind also dasselbe wie 2-stellige Relationen über einer Menge V.

Von besonderem Interesse sind Äquivalenzrelationen:

Definition 1.10 (Äquivalenzrelation).
Sei E eine 2-stellige Relation über einer Menge V (d.h. $G = (V, E)$ ist ein gerichteter Graph).

(a) E heißt **reflexiv**, falls für alle $v \in V$ gilt: reflexiv

$$(v, v) \in E. \qquad \text{(Skizze: } v \text{ mit Schlinge)}$$

symmetrisch

(b) E heißt **symmetrisch**, falls f.a. $v, w \in V$ gilt:

$$\text{Wenn } (v,w) \in E, \text{ dann auch } (w,v) \in E.$$

(d.h. zu jeder Kante 

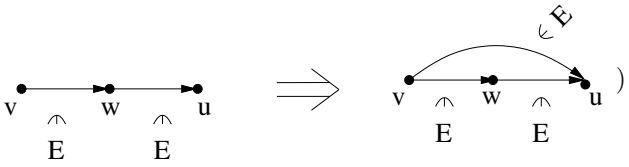
gibt es auch eine "Rückwärtskante" 

)

transitiv

(c) E heißt **transitiv**, falls f.a. $v, w, u \in V$ gilt:

$$\text{Ist } (v,w) \in E \text{ und } (w,u) \in E, \text{ so auch } (v,u) \in E.$$

(Skizze: 

)

Äquivalenzrelation

(d) Eine **Äquivalenzrelation** ist eine 2-stellige Relation, die **reflexiv**, **transitiv** und **symmetrisch** ist.

Beispiel 1.11. Beispiele für Äquivalenzrelationen:

(a) **Gleichheit:** Für jede Menge M ist

$$E := \{ (m,m) : m \in M \}$$

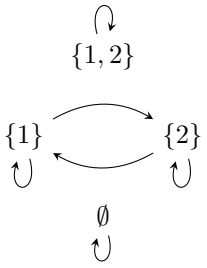
eine Äquivalenzrelation. Die Aussage "$(x,y) \in E$" entspricht gerade der Aussage "$x = y$".

(b) **Gleichmächtigkeit:** Für jede endliche Menge M ist

$$E := \{ (A,B) : A \subseteq M,\, B \subseteq M,\, |A| = |B| \}$$

eine Äquivalenzrelation über der Potenzmenge $\mathcal{P}(M)$.

Skizze für $M = \{1,2\}$:



(c) **Logische Äquivalenz:** Die Relation

$$E := \{ (\varphi,\psi) : \varphi, \psi \in \mathrm{AL},\, \varphi \equiv \psi \}$$

ist eine Äquivalenzrelation über der Menge AL aller aussagenlogischen Formeln.

Bemerkung 1.12 (Äquivalenzklassen). Sei E eine Äquivalenzrelation über einer Menge V. Für jedes $v \in V$ bezeichnet

$$[v]_E := \{ v' \in V : (v,v') \in E \}$$

die **Äquivalenzklasse** von v bezüglich E. D.h.: Die Äquivalenzklasse $[v]_E$ besteht aus allen Elementen von V, die laut E "äquivalent" zu v sind. Eine Menge $W \subseteq V$ heißt *Äquivalenzklasse* (bzgl. E), falls es ein $v \in V$ mit $W = [v]_E$ gibt. Das Element v wird dann ein *Vertreter*, bzw. *Repräsentant* seiner Äquivalenzklasse W genannt. Wir kommen zu der herausragenden Eigenschaft einer Äquivalenzrelation (bzgl. E):

$[v]_E$

Äquivalenzklasse

Satz 1.13. *E sei eine Äquivalenzrelation über der Menge V und $[v]_E, [w]_E$ seien zwei beliebige Äquivalenzklassen. Dann stimmen die beiden Äquivalenzklassen entweder überein (d.h. $[v]_E = [w]_E$) oder die Klassen sind disjunkt (d.h. $[v]_E \cap [w]_E = \emptyset$).*

Beweis: Wir nehmen an, dass die Äquivalenzklassen $[v]_E$ und $[w]_E$ ein gemeinsames Element u besitzen. Dann gilt

1. $(v, u) \in E$ und $(w, u) \in E$ nach Definition der Äquivalenzklassen,
2. Es folgt $(u, w) \in E$, denn E ist symmetrisch.
3. Wir benutzen, dass E transitiv ist und erhalten $(v, w) \in E$ aus $(v, u) \in E$ und $(u, w) \in E$.
4. Mit dem gleichen Argument folgt für einen beliebigen mit v äquivalenten Knoten x aus $(v, x) \in E$ (d.h. $(x, v) \in E$) und $(v, w) \in E$ auch $(x, w) \in E$, bzw. $(w, x) \in E$. Wir erhalten $[v]_E \subseteq [w]_E$.
5. Ebenso erhalten wir $[v]_E \subseteq [w]_E$ und wir haben die Gleichheit $[v]_E = [w]_E$ nachgewiesen.

□

Falls V endlich und nicht leer ist, folgt daraus, dass es eine Zahl $k \in \mathbb{N}_{>0}$ und Äquivalenzklassen $W_1, \ldots, W_k$ geben muss, so dass V eine disjunkte Vereinigung der verschiedenen Äquivalenzklassen ist, d.h. $V = W_1 \dot\cup \cdots \dot\cup W_k$ gilt.

> Die Zahl k wird auch **Index** von E genannt. Der Index der Äquivalenzrelation E gibt an, wie viele verschiedene Äquivalenzklassen E besitzt.

Index

Beispielsweise hat die Gleichmächtigkeits-Relation aus Beispiel 1.11 (b) den Index $|M| + 1$.

Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein DFA. Wir erinnern an die Definition der Verschmelzungsrelation: Zwei Zustände $p, q \in Q$ sind äquivalent, kurz $p \equiv_A q$, wenn

$$\text{f.a. Worte } w \in \Sigma^* : \widehat{\delta}(p, w) \in F \;\Leftrightarrow\; \widehat{\delta}(q, w) \in F. \tag{1.1}$$

Satz 1.14. Sei A ein DFA. Dann ist die Verschmelzungsrelation $\equiv_A$ eine Äquivalenzrelation.

Insbesondere ist die Zustandsmenge Q eine disjunkte Vereinigung von Äquivalenzklassen und unsere Idee, äquivalente Zustände zu einem einzigen Zustand zu verschmelzen, erscheint sinnvoll. Wenn alles klappt, wenn wir also die Zustandsübergänge nach Verschmelzung „vernünftig“ setzen können, dann ist der „Äquivalenzklassenautomat“, also der DFA nach Verschmelzung mit A äquivalent. Natürlich stimmt die Anzahl seiner Zustände stimmt mit dem Index der Verschmelzungsrelation überein. Ach wär das toll, wenn der Index die minimale Zustandszahl wäre.

Beweis: Wenn wir $q = p$ in (1.2) annehmen, erhalten wir $p \equiv_A p$ als Konsequenz und die Verschmelzungsrelation ist reflexiv. Die Symmetrie folgt auch sofort, denn aus $p \equiv_A q$, d.h

$$\text{f.a. Worte } w \in \Sigma^* : \widehat{\delta}(p, w) \in F \;\Leftrightarrow\; \widehat{\delta}(q, w) \in F.$$

folgt $q \equiv_A p$, d.h.

$$\text{f.a. Worte } w \in \Sigma^* : \widehat{\delta}(q, w) \in F \;\Leftrightarrow\; \widehat{\delta}(p, w) \in F.$$

Seien p, q, r beliebige Zustände. Um die Transitivität nachzuweisen, müssen wir $p \equiv_A q, q \equiv_A r$ annehmen und dann $p \equiv_A r$ zeigen. Nach Annahme gilt also

$$\text{f.a. Worte } w \in \Sigma^* : \big(\widehat{\delta}(p, w) \in F \;\Leftrightarrow\; \widehat{\delta}(q, w) \in F\big) \wedge \big(\widehat{\delta}(q, w) \in F \;\Leftrightarrow\; \widehat{\delta}(r, w) \in F\big).$$

und damit folgt

$$\text{f.a. Worte } w \in \Sigma^* : \widehat{\delta}(p, w) \in F \;\Leftrightarrow\; \widehat{\delta}(r, w) \in F.$$

Wir haben die Behauptung $p \equiv_A r$ gezeigt. □

1.2.2 Die Verschmelzungsrelation

Sei der DFA $A = (\Sigma, Q, \delta, q_0, F)$ gegeben. Um die Zustandszahl von A zu reduzieren, führen wir die folgenden Schritte durch:

1. Zuerst bestimmen wir die Äquivalenzklassen der Verschmelzungsrelation $\equiv_A$.
2. Für jede Äquivalenzklasse von $\equiv_A$ verschmelzen wir alle Zustände der Klasse zu einem einzigen Zustand und fügen „entsprechende“ Übergänge ein. Den neuen Automaten nennen wir $\mathbf{A'}$ und bezeichnen ihn als den **Äquivalenzklassenautomaten** von A.

Äquivalenzklassen-automaten

   - Wie sollen wir die Zustandsübergänge von A' definieren, so dass A und A' dieselbe Sprache berechnen?

Wenn der Index von $\equiv_A$, und damit die Zustandszahl von A', mit der minimalen Zustandszahl übereinstimmt, dann ist A' **minimal**! Natürlich müssen wir uns auch fragen, ob wir $\equiv_A$ und A' effizient berechnen können.

Wie können wir die Äquivalenzklassen der Verschmelzungsrelation berechnen?

Wir bestimmen alle Paare **nicht-äquivalenter** Zustände.

Definition 1.15. Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein DFA mit Zuständen $p, q \in Q$.

Zeuge

Wir sagen, dass das Wort $w \in \Sigma^*$ die Zustände p und q trennt, bzw. ein **Zeuge** für die Inäquivalenz von p und q ist, wenn

$$\Big(\widehat{\delta}(p, w) \in F \wedge \widehat{\delta}(q, w) \notin F\Big) \vee \Big(\widehat{\delta}(p, w) \notin F \wedge \widehat{\delta}(q, w) \in F\Big).$$

Um alle Paare nicht-äquivalenter Zustände zu bestimmen, wenden wir den **Verschmelzungs-algorithmus** an.

Verschmelzungs-algorithmus

1. **Markiere** alle Paarmengen $\{p, q\}$ mit $p \in F$ und $q \notin F$ (als nicht-äquivalent).

- Es ist $\delta(p, \varepsilon) \in F$ und $\delta(q, \varepsilon) \notin F$.
- $w = \varepsilon$ ist Zeuge für die Nicht-Äquivalenz von p und q.

2. Wenn die Paarmenge $\{p, q\}$ bereits markiert wurde und

$$\text{wenn } \delta(r, a) = p \text{ sowie } \delta(s, a) = q$$

für ein $a \in \Sigma$, dann **markiere** $\{r, s\}$.

- Da $p \not\equiv_A q$, gibt es einen **Zeugen** w mit

$$\big(\widehat{\delta}(p, w) \in F \text{ und } \widehat{\delta}(q, w) \notin F\big) \text{ oder } \big(\widehat{\delta}(p, w) \notin F \text{ und } \widehat{\delta}(q, w) \in F\big).$$

- Das Wort aw bezeugt, dass r und s nicht-äquivalent sind.

3. Halte, wenn keine neuen Paarmengen $\{r, s\}$ markiert werden können.
   - Unsere Hoffnung ist, dass $p \not\equiv_A q$ genau dann gilt, wenn die Paarmenge $\{p, q\}$ markiert wurde.

Der Verschmelzungsalgorithmus funktioniert!

Satz 1.16. Der Verschmelzungsalgorithmus findet alle Paare inäquivalenter Zustände.

Beweis: Sei P die Menge aller Paare $\{r, s\}$ nicht-äquivalenter Zustände, die aber von unserem Verfahren *nicht* gefunden werden. Wir müssen zeigen, dass P leer ist! Wir führen einen Beweis durch Widerspruch und nehmen an, dass P nicht-leer ist. $\{p, q\} \in P$ habe unter allen Paaren in P einen *kürzesten* Zeugen w.

Fall 1: Wenn $w = \varepsilon$, dann ist

- $\big(\delta(p, \varepsilon) \in F \text{ und } \delta(q, \varepsilon) \notin F\big)$ oder $\big(\delta(p, \varepsilon) \notin F \text{ und } \delta(q, \varepsilon) \in F\big)$,
- bzw. $\big(p \in F \text{ und } q \notin F\big)$ oder $\big(p \notin F \text{ und } q \in F\big)$.

Aber dann haben wir $\{p, q\}$ im Schritt 1. markiert. $\lightning$

Fall 2: Wenn $w = au$ für den Buchstaben $a \in \Sigma$, dann ist

- $\big(\widehat{\delta}(p, au) \in F \text{ und } \widehat{\delta}(q, au) \notin F\big)$ oder $\big(\widehat{\delta}(p, au) \notin F \text{ und } \widehat{\delta}(q, au) \in F\big)$,
- bzw. $\big(\widehat{\delta}(\delta(p, a), u) \in F \text{ und } \widehat{\delta}(\delta(q, a), u) \notin F\big)$ oder $\big(\widehat{\delta}(\delta(p, a), u) \notin F \text{ und } \widehat{\delta}(\delta(q, a), u) \in F\big)$.

Dann gilt $\delta(p, a) \not\equiv_A \delta(q, a)$ mit dem kürzeren Zeugen u: Aber $\{\delta(p, a), \delta(q, a)\}$ muss nach Annahme bereits markiert sein, und wir werden darauffolgend $\{p, q\}$ markieren. $\lightning$ □

Ist unser Algorithmus effizient? Die Anzahl aller Zustandspaare ist durch $|Q|^2$ nach oben beschränkt. Wir werden deshalb höchstens $|Q|^2$ Markierungsschritten benötigen.

Wir haben die Verschmelzungsrelation erfolgreich berechnet und bestimmen jetzt den Äquivalenzklassenautomaten.

1.2.3 Der Aquivalenzklassenautomat

Für Zustand $p \in Q$ bezeichnet

$$[p]_A := \big\{q \in Q \mid p \equiv_A q\big\}$$

die Äquivalenzklasse von p. Der **Äquivalenzklassenautomat**

$$A' = (\Sigma, Q', \delta', q_0', F')$$

für den DFA $A = (\Sigma, Q, \delta, q_0, F)$ besitzt

- die Zustandsmenge
$$Q' := \big\{\, [p]_A \mid p \in Q \,\big\},$$
- den Anfangszustand $q_0' := [q_0]_A$,
- die Menge $F' := \big\{\, [p]_A \mid p \in F \,\big\}$ der akzeptierenden Zustände und
- das Programm δ' mit
$$\delta'([p]_A, a) := [\ \delta(p,a)\ ]_A$$
für alle $q \in Q$ und $a \in \Sigma$.

Wir haben alle Zustände einer Äquivalenzklasse $[p]_A$ zu einem einzigen Zustand zusammengefasst und haben diesen Zustand $[p]_A$ genannt. Folgerichtig arbeiten wir deshalb mit der Menge Q' aller Äquivalenzklassen von $\equiv_A$.

Als Menge der akzeptierenden Zustände des Äquivalenzklassenautomaten haben wir alle mit einem akzpetierenden Zustand von A äquivalenten Zustände gewählt: Das scheint vernünftig, aber ist denn wirklich jeder mit einem akzeptierenden Zustand von A äquivalenter Zustand auch selbst akzeptierend?

Um die Übergangsfunktion δ' für eine Äquivalenzklasse $[p]_A$ und einen Buchstaben $a \in \Sigma$ zu definieren, sind wir fast gezwungen, die Äquivalenzklasse des Zustands $\delta(p,a)$ zu wählen, und genau das haben wir getan. Aber vorsichtig, gilt denn $\delta(p,a) \equiv_A \delta(q,a)$ für je zwei äquivalente Zustände p, q? Wenn nicht, dann wäre unsere Definition von δ' abhängig vom Vertreter der Äquivalenzklasse und das würde nichts Gutes verheißen!

Wir zeigen zuerst, dass die Definition von δ' unabhängig vom Vertreter der Äquivalenzklasse ist. Danach können wir beweisen, dass A und A' äquivalente DFAs sind.

Satz 1.17. *$A = (\Sigma, Q, \delta, q_0, F)$ sei ein DFA. Für alle Zustände $p, q \in Q$ mit $[p]_A = [q]_A$ gilt*

$$\delta(p,a) \equiv_A \delta(q,a).$$

Beweis: Seien also $p, q \in Q$ äquivalente Zustände, d.h. es gilt $p \equiv_A q$. Dann folgt:

$$\begin{aligned} p \equiv_A q \quad &\Longrightarrow \quad \text{für alle } w' \in \Sigma^* \text{ gilt: } \widehat{\delta}(p,w') \in F \iff \widehat{\delta}(q,w') \in F \\ &\Longrightarrow \quad \text{für alle } w \in \Sigma^* \text{ gilt: } \widehat{\delta}(p,aw) \in F \iff \widehat{\delta}(q,aw) \in F. \end{aligned} \tag{1.2}$$

Für die letzte Implikation haben wir die Aussage der ersten Implikation auf alle w' der Form $w' = aw$ eingeschränkt. Jetzt beachte $\widehat{\delta}(p,aw) = \widehat{\delta}(\delta(p,a),w)$ und $\widehat{\delta}(q,aw) = \widehat{\delta}(\delta(q,a),w)$:

$$\begin{aligned} (1.2) \quad &\Longrightarrow \quad \text{für alle } w \in \Sigma^* \text{ gilt: } \widehat{\delta}\big(\delta(\mathtt{p},\mathtt{a}),\mathtt{w}\big) \in \mathtt{F} \iff \widehat{\delta}\big(\delta(\mathtt{q},\mathtt{a}),\mathtt{w}\big) \in \mathtt{F} \\ &\Longrightarrow \quad \delta(p,a) \equiv_A \delta(q,a). \end{aligned}$$

Somit hängt die Definition von δ' nicht ab von der speziellen Wahl eines Vertreters der Äquivalenzklasse. □

Wir können jetzt nachweisen, dass A und A' äquivalente DFAs sind!

Satz 1.18. $A = (\Sigma, Q, \delta, q_0, F)$ sei ein DFA. Dann sind A und A' äquivalente DFAs, d.h. es gilt

$$L(A) = L(A').$$

Beweis: Wir zeigen die Behauptung in drei Schritten.

1. Schritt 1: Zeige, dass
$$\widehat{\delta'}([q_0]_A, w) = [\widehat{\delta}(q_0, w)]_A$$
für alle $w \in \Sigma^*$ gilt. Wenn wir also wissen möchten, in welchem Zustand sich A' nach Lesen des Worts w befindet, dann bestimmen wir den Zustand $p = \delta(q_0, w)$ von A nach Lesen von w. A' befindet sich dann im Zustand $[p]_A$ und das ist alles andere als überraschend.

2. In den Schritten 2 und 3 zeigen wir die Inklusionen $L(A) \subseteq L(A')$ und $L(A') \subseteq L(A)$.

Beweis von Schritt 1: Durch vollständige Induktion über die Länge von w.
Induktionsanfang: Betrachte $w = \varepsilon$. Nach Definition von δ' gilt: $\widehat{\delta'}([q_0]_A, \varepsilon) = [q_0]_A = [\widehat{\delta}(q_0, \varepsilon)]_A$.

Induktionsschritt: Betrachte $w = ua$ mit $u \in \Sigma^*$ und $a \in \Sigma$. Nach Induktionsannahme gilt $\widehat{\delta'}([q_0]_A, u) = [\widehat{\delta}(q_0, u)]_A$. Wir müssen zeigen
Beh.: $\widehat{\delta'}([q_0]_A, ua) = [\widehat{\delta}(q_0, ua)]_A$.
Beweis:

$$\begin{aligned}
\widehat{\delta'}([q_0]_A, ua) &= \delta'\big(\widehat{\delta'}([q_0]_A, u), a\big) \\
&\overset{\text{Induktionsannahme}}{=} \delta'\big([\widehat{\delta}(q_0, u)]_A, a\big) \\
&= \delta'([p]_A, a), \quad \text{für } p := \widehat{\delta}(q_0, u) \\
&\overset{\text{Definition von } \delta'}{=} [\delta(p, a)]_A \;=\; [\delta\big(\widehat{\delta}(q_0, u), a\big)]_A \;=\; [\widehat{\delta}(q_0, ua)]_A.
\end{aligned}$$

$\square_{\text{Schritt 1}}$

Beweis von Schritt 2: Zeige, dass $L(A) \subseteq L(A')$ gilt. Sei also w ein beliebiges Wort in $L(A)$.

1. A akzeptiert die Eingabe w, d.h. es gilt: $p := \widehat{\delta}(q_0, w) \in F$.

2. Nach Schritt 1 gilt: $\widehat{\delta'}([q_0]_A, w) = [\widehat{\delta}(q_0, w)]_A = [p]_A$.

3. Nach Definition von F' gilt: $[p]_A \in F'$, denn $p \in F$.

Somit wird w von A' akzeptiert, d.h. es gilt $w \in L(A')$.

$\square_{\text{Schritt 2}}$

Beweis von Schritt 3: Zeige, dass $L(A') \subseteq L(A)$ gilt. Wir zeigen eine Zwischenbehauptung.

Beh: Wenn $p \equiv_A q$ und $p \in F$, dann auch $q \in F$.
Beweis: Wenn $p \equiv_A q$, dann folgt aus der Definition der Verschmelzungsrelation $\widehat{\delta}(p, w) \in F \Leftrightarrow \widehat{\delta}(q, w) \in F$ für alle Worte $w \in \Sigma^*$. Also gilt insbesondere: $\delta(p, \varepsilon) = p \in F \Leftrightarrow \delta(q, \varepsilon) = q \in F$.
Zurück zu Schritt 3. Wir nehmen an, dass A' die Eingabe w akzeptiert, d.h. es gilt: $\widehat{\delta'}([q_0]_A, w) \in F'$. Wir müssen $w \in L(A)$ zeigen, d.h., dass $\widehat{\delta}(q_0, w) \in F$ gilt.

Nach Schritt 1 gilt:

$$\widehat{\delta'}([q_0]_A, w) = [\widehat{\delta}(q_0, w)]_A.$$

Aber $\widehat{\delta'}([q_0]_A, w) \in F'$. Nach Definition von F' muss es einen Zustand $p \in F$ mit $p \equiv_A \widehat{\delta}(q_0, w)$ geben. Also folgt aus der Zwischenbehauptung $\widehat{\delta}(q_0, w) \in F$, d.h. $w \in L(A)$.

$\square_{\text{Schritt 3}}$

$L(A) \subseteq L(A')$ gilt nach Schritt 2 und $L(A') \subseteq L(A)$ nach Schritt 3. Es folgt $L(A) = L(A')$ und die Behauptung von Satz 1.18 ist gezeigt.

$\square$

Der Äquivalenzautomat tut genau das, was wir von ihm verlangen. Allerdings müssen wir später noch zeigen, dass A' der kleinste mit A äquivalente DFA ist.

Bemerkung 1.19. Wenn wir den Äquivalenzklassenautomat für einen Moore oder Mealy Automaten bestimmen möchten, dann müssen wir die Verschmelzungsrelation dem Ausgabeverhalten des Automaten anpassen: Für die Ausgabefunktion λ führen wir die Erweiterung $\widehat{\lambda}$ für Worte $w \in \Sigma^*$ ein und definieren dann die Äquivalenz von Zuständen p, q durch

$$p \equiv_A q \;\Leftrightarrow\; \text{f.a. Worte } w \in \Sigma^* : \widehat{\lambda}(p, w) = \widehat{\lambda}(r, w).$$

1.2.4 Der Minimierungsalgorithmus

Minimierungs-algorithmus

Um einen DFA A zu minimieren, entfernen wir zuerst alle vom Startzustand aus nicht erreichbaren Zustände, berechnen dann die Äquivalenzklassen von $\equiv_A$ und bestimmen den Äquivalenzklassenautomaten A' mit Hilfe der Klassen. Hier ist unser **Minimierungsalgorithmus**.

Eingabe: Ein DFA $A = (Q, \Sigma, \delta, q_0, F)$.

Schritt 1: Entferne aus A alle **überflüssigen** Zustände, d.h. alle Zustände, die nicht von q_0 aus erreichbar sind.

Schritt 2: Bestimme alle Paare $\{p, q\}$ mit $p, q \in Q$ und $p \not\equiv_A q$:

1. Markiere alle Paarmengen in $M_0 := \{\ \{p, q\}\ :\ p \in F,\ q \in Q \setminus F\ \}$; Setze $i := 0$
2. Wiederhole
3. Für alle $\{p, q\} \in M_i$ und für alle $x \in \Sigma$ tue folgendes:
4. Markiere $\{p', q'\}$ für alle $p' \neq q'$ mit $\delta(p', x) = p$ und $\delta(q', x) = q$.
5. Sei M_{i+1} die Menge aller hierbei **neu** markierten Paarmengen.
6. $i := i + 1$
7. bis $M_i = \emptyset$
8. **Ausgabe:** $M := M_0 \cup \cdots \cup M_{i-1}$.

Schritt 3: Konstruiere $A' := (Q', \Sigma, \delta', q_0', F')$:

$$Q' := \{\ [q]_A\ :\ q \in Q\},\ \text{ wobei }\ [q]_A = \{\ p \in Q\ :\ \{p, q\} \notin M\ \}$$

$$q_0' := [q_0]_A,\quad F' := \{\ [q]_A\ :\ q \in F\ \}$$

$$\delta' : Q' \times \Sigma \to Q' \quad \text{mit} \quad \delta'\big([q]_A, a\big) := [\delta(q,a)]_A \quad \text{für alle } q \in Q \text{ und } x \in \Sigma.$$

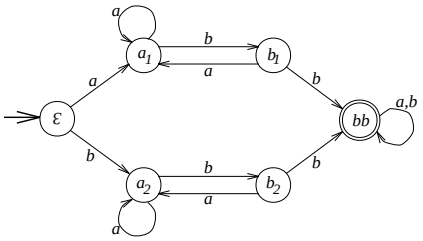
Ist der Algorithmus korrekt und effizient? In jeder Iteration in Schritt 2 wird mindestens eine neue Paarmenge markiert: Es gibt also höchstens $|Q|^2$ Iterationen und der Algorithmus ist effizient.

Die kritische Frage

„Ist A' minimal?“

muss noch geklärt werden. Aber zuerst rechnen wir einige Beispiele durch.

Beispiel 1.20. Wir möchten den folgenden DFA minimieren:



Die Menge $\mathbf{M_0}$: bb ist der einzige akzeptierende Zustand. Deshalb hat die Menge M_0 die Form

$$M_0 = \{\{\varepsilon, \mathtt{bb}\}, \{\mathtt{a}_1, \mathtt{bb}\}, \{\mathtt{a}_2, \mathtt{bb}\}, \{\mathtt{b}_1, \mathtt{bb}\}, \{\mathtt{b}_2, \mathtt{bb}\}\}.$$

Was haben wir gelernt? {bb} ist eine Klasse der Verschmelzungsrelation, die restlichen Klassen sind disjunkte Teilmengen von $\{\varepsilon, \mathtt{a}_1, \mathtt{a}_2, \mathtt{b}_1, \mathtt{b}_2\}$.

Wir veranschaulichen die Menge aller Paare von Zuständen durch eine zwei-dimensionale Tabelle und vermerken alle zu M_0 gehörenden Paare.

$\mathtt{a}_1$					
$\mathtt{a}_2$					
$\mathtt{b}_1$					
$\mathtt{b}_2$					
bb	M_0	M_0	M_0	M_0	M_0
	ε	$\mathtt{a}_1$	$\mathtt{a}_2$	$\mathtt{b}_1$	$\mathtt{b}_2$

Die Menge $\mathbf{M_1}$: Wir inspizieren alle nicht mit M_0 markierten Positionen der Tabelle und überprüfen, ob wir eine mit M_0 markierte Position erhalten, wenn der Buchstabe a (bzw. der Buchstabe b) gelesen wird. Dies ist der Fall zum Beispiel für die Position in Zeile $\mathtt{b}_1$ und Spalte ε, die zur Paarmenge $\{\mathtt{b}_1, \varepsilon\}$ gehört. Es ist $\delta(\mathtt{b}_1, \mathtt{b}) = \mathtt{bb}$ und $\delta(\varepsilon, \mathtt{b}) = \mathtt{a}_2$: $\{\mathtt{b}_1, \varepsilon\}$ gehört zur Menge M_1, weil $\{\mathtt{bb}, \mathtt{a}_2\}$ zur Menge M_0 gehört.

$\mathtt{a}_1$					
$\mathtt{a}_2$					
$\mathtt{b}_1$	M_1	M_1	M_1		
$\mathtt{b}_2$	M_1	M_1	M_1		
bb	M_0	M_0	M_0	M_0	M_0
	ε	$\mathtt{a}_1$	$\mathtt{a}_2$	$\mathtt{b}_1$	$\mathtt{b}_2$

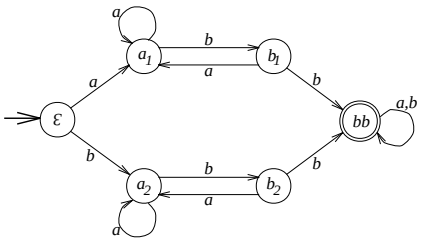
Die Menge M_2: Diesmal inspizieren alle nicht mit M_0 oder M_1 markierten Positionen der Tabelle und überprüfen, ob wir eine mit M_1 markierte Position erhalten, wenn der Buchstabe a (bzw. der Buchstabe b) gelesen wird. Dies ist der Fall zum Beispiel für die Position in Zeile $\mathtt{a}_1$ und Spalte ε, die zur Paarmenge $\{\mathtt{a}_1, \varepsilon\}$ gehört. Es ist $\delta(\mathtt{a}_1, \mathtt{b}) = \mathtt{b}_1$ und $\delta(\varepsilon, \mathtt{b}) = \mathtt{a}_2$: $\{\mathtt{a}_1, \varepsilon\}$ gehört zur Menge M_2, weil $\{\mathtt{b}_1, \mathtt{a}_2\}$ zur Menge M_1 gehört.

$\mathtt{a}_1$	M_2				
$\mathtt{a}_2$	M_2				
$\mathtt{b}_1$	M_1	M_1	M_1		
$\mathtt{b}_2$	M_1	M_1	M_1		
$\mathtt{bb}$	M_0	M_0	M_0	M_0	M_0
	ε	$\mathtt{a}_1$	$\mathtt{a}_2$	$\mathtt{b}_1$	$\mathtt{b}_2$

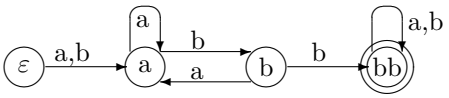
Die Menge M_3: Wir müssen nur die unmarkierten Positionen (mit den jeweiligen Paarmengen $\{\mathtt{a}_2, \mathtt{a}_1\}$ und $\{\mathtt{b}_2, \mathtt{b}_1\}$) inspizieren. Wir beginnen mit $\{\mathtt{a}_2, \mathtt{a}_1\}$ und erhalten die unmarkierte Paarmenge $\{\mathtt{a}_2, \mathtt{a}_1\}$ für den Buchstaben a, bzw. die unmarkierte Paarmenge $\{\mathtt{b}_2, \mathtt{b}_1\}$ für den Buchstaben b: Die Paarmenge $\{\mathtt{a}_2, \mathtt{a}_1\}$ bleibt unmarkiert.

Schließlich betrachten wir die Paarmenge $\{\mathtt{b}_2, \mathtt{b}_1\}$. Für den Buchstaben a erhalten wir die unmarkierte Paarmenge $\{\mathtt{a}_2, \mathtt{a}_1\}$ und für den Buchstaben b die Menge $\{\mathtt{bb}, \mathtt{bb}\}$, die natürlich nicht markiert werden kann.

Fazit: Es ist $M_3 = \emptyset$ und wir kennen die Verschmelzungsrelation: Die einzigen nicht-trivialen Äquivalenzbeziehungen sind $\mathtt{a}_1 \equiv_{\mathtt{A}} \mathtt{a}_2$ und $\mathtt{b}_1 \equiv_{\mathtt{A}} \mathtt{b}_2$, die weiteren Zustände ε und $\mathtt{bb}$ bilden jeweils ihre eigene Äquivalenzklasse. Wir erhalten nach Verschmelzung der Zustände $\mathtt{a}_1, \mathtt{a}_2$, bzw. $\mathtt{b}_1, \mathtt{b}_2$ im Ausgangsautomaten



den Äquivalenzklassenautomaten



Genügt es, wenn wir nur Zustände mit identischen Nachfolgezuständen verschmelzen? Nein, der Automat aus dem letzten Beispiel ist ein Gegenbeispiel.

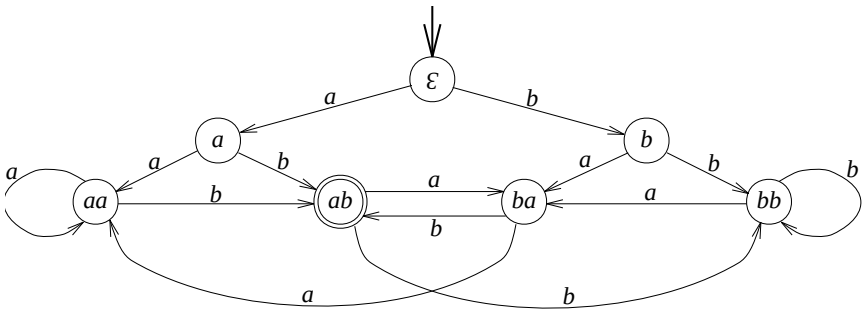
Beispiel 1.21. Wie stellt man fest, ob ein Wort das Suffix ab besitzt? Wir arbeiten mit dem Alphabet $\Sigma = \{a, b\}$ und speichern in unserem ersten Ansatz die beiden zuletzt gelesenen Buchstaben im aktuellen Zustand ab. Wir benutzen deshalb die Zustände

$$Q = \{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}\}$$

- mit Startzustand $q_0 = \varepsilon$ („wir haben noch nichts gelesen“ und

- dem akzeptierenden Zustand ab: Die beiden letzten Buchstaben sind ab.

Hier ist unser erster Automat.



Die Menge $\mathbf{M_0}$: ab ist der einzige akzeptierende Zustand. Deshalb hat die Menge M_0 die Form

$$M_0 = \{\{\varepsilon, \mathtt{ab}\}, \{\mathtt{a}, \mathtt{ab}\}, \{\mathtt{b}, \mathtt{ab}\}, \{\mathtt{aa}, \mathtt{ab}\}, \{\mathtt{ba}, \mathtt{ab}\}, \{\mathtt{bb}, \mathtt{ab}\}\}.$$

Was haben wir gelernt? {ab} ist eine Klasse der Verschmelzungsrelation, die restlichen Klassen sind disjunkte Teilmengen von $\{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ba}, \mathtt{bb}\}$.

Wir veranschaulichen die Menge aller Paare von Zuständen wieder durch eine zwei-dimensionale Tabelle und vermerken alle zu M_0 gehörenden Paare.

a						
b						
aa						
ab	M_0	M_0	M_0	M_0		
ba					M_0	
bb					M_0	
	ε	a	b	aa	ab	ba

Die Menge $\mathbf{M_1}$: Dazu inspizieren wir alle nicht mit M_0 markierten Positionen der Tabelle und überprüfen, ob wir eine mit M_0 markierte Position erhalten, wenn der Buchstabe a (bzw. der Buchstabe b) gelesen wird. Dies ist der Fall zum Beispiel für die Position oben links, die die Paarmenge $\{\mathtt{a}, \varepsilon\}$ repäsentiert. Es ist $\delta(\mathtt{a}, \mathtt{b}) = \mathtt{ab}$ und $\delta(\varepsilon, \mathtt{b}) = \mathtt{b}$: $\{\mathtt{a}, \varepsilon\}$ gehört zur Menge M_1, weil $\{\mathtt{ab}, \mathtt{b}\}$ zur Menge M_0 gehört.

Um nicht jede nicht markierte Position der Tabelle gesondert zu diskutieren, machen wir die folgende Beobachtung. Für $x = a$ oder $x = b$ gilt: Wenn $\delta(p', x) = p, \delta(q', x) = q$ und $\{p, q\} \in M_0$, dann ist $\{p', q'\} \in M_1$. Der Zustand ab wird aber genau dann erreicht, wenn der Buchstabe $x = b$ in den Zuständen a, aa oder ba gelesen wird. Demgemäß erhalten wir für M_1 die Tabelle

a	M_1					
b		M_1				
aa	M_1		M_1			
ab	M_0	M_0	M_0	M_0		
ba	M_1		M_1		M_0	
bb		M_1		M_1	M_0	M_1
	ε	a	b	aa	ab	ba

Was haben wir gelernt?

1. Kein Zustand in der Menge $\mathtt{a}, \mathtt{aa}, \mathtt{ba}$ kann mit einem Zustand in der Menge $\varepsilon, \mathtt{b}, \mathtt{bb}$ äquivalent sein.

2. Die weiteren Äquivalenzklassen der Verschmelungsrelation sind disjunkte Teilmengen entweder von $\{\mathtt{a}, \mathtt{aa}, \mathtt{ba}\}$ oder von $\{\varepsilon, \mathtt{b}, \mathtt{bb}\}$.

Lassen sich die Zustände in $\{\mathtt{a}, \mathtt{aa}, \mathtt{ba}\}$ (bzw. in $\{\varepsilon, \mathtt{b}, \mathtt{bb}\}$) voneinander trennen?

Die Menge $\mathbf{M_2}$:

1. Die Zustände $\mathtt{a}, \mathtt{aa}, \mathtt{ba}$ können nicht voneinander getrennt werden, da sie alle unter a auf den Zustand $\mathtt{aa}$ und unter b auf den Zustand $\mathtt{ab}$ führen.

2. $\mathtt{b}$ und $\mathtt{bb}$ lassen sich ebenfalls nicht mehr trennen. Beide führen unter a auf $\mathtt{ba}$ und unter b auf $\mathtt{bb}$.

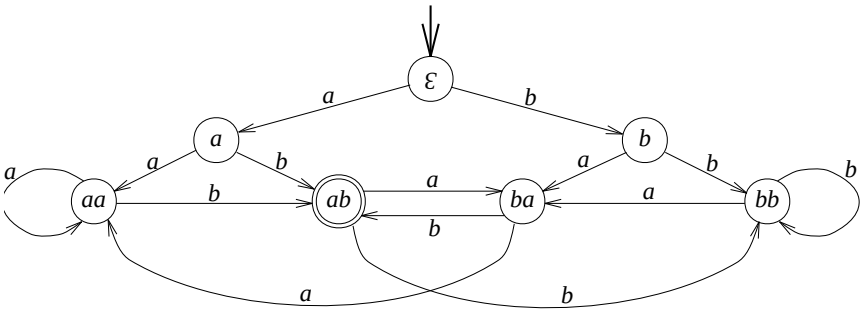
3. Aber auch ε führt unter a auf die Klasse $\{\mathtt{a}, \mathtt{aa}, \mathtt{ba}\}$ und unter b auf $\mathtt{b}$.

Es ist $M_2 = \emptyset$, denn ε lässt sich nicht von $\mathtt{b}, \mathtt{bb}$ trennen. Wie sehen die verschiedenen Äquivalenzklassen aus?

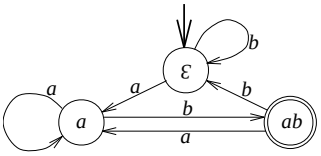
(a) $\{\mathtt{ab}\}$ ist die Klasse des einzigen akzeptierenden Zustands,

(b) $\{\mathtt{a}, \mathtt{aa}, \mathtt{ba}\}$ und $\{\varepsilon, \mathtt{b}, \mathtt{bb}\}$ sind die beiden verbleibenden Klassen.

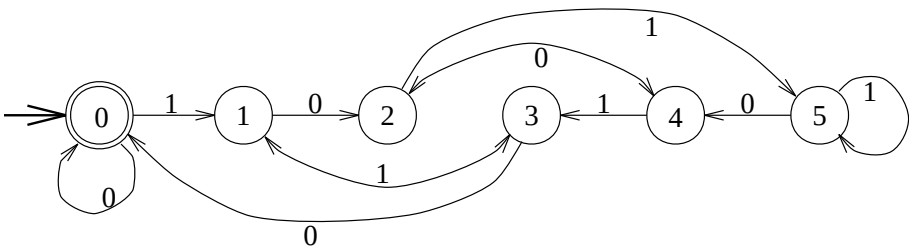
Und die Verschmelzung führt vom Ausgangsautomaten



auf den Äquivalenzklassenautomaten



Beispiel 1.22. Der DFA A mit Zustandsdiagramm



akzeptiert die Binärdarstellungen aller durch 6 teilbaren Zahle akzeptiert, also die Sprache

$$L(A) \quad = \quad \Big\{ w \in \{0,1\}^* \ : \ \sum_{i=1}^{|w|} w_i \, 2^{|w|-i} \equiv 0 \ (\text{mod } 6) \Big\}. \tag{1.3}$$

Wir möchten wieder den Äquivalenzklassenautomaten A' bestimmen und stellen zuerst die Tabelle aller Paarmengen auf:

1	M_0				
2	M_0	M_2			
3	M_0	M_1	M_1		
4	M_0		M_2	M_1	
5	M_0	M_2		M_1	M_2
	0	1	2	3	4

Wir können somit die Zustände 1 und 4, aber auch die Zustände 2 und 5 verschmelzen.

1.3 Die Nerode-Relation

Was haben wir bisher gelernt? Der ursprüngliche Automat $A = (Q, \Sigma, \delta, q_0, F)$ und sein Äquivalenzklassenautomat A' sind äquivalent. Wir können A' effizient berechnen. Aber hat A' unter allen mit A äquivalenten DFAs die kleinste Zustandszahl?

Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein DFA. Wenn zwei Worte $u, v \in \Sigma^*$ auf denselben Zustand von A führen (d.h. wenn $\widehat{\delta}(q_0, u) = \widehat{\delta}(q_0, v)$ gilt), dann folgt

$$\text{f.a. } w \in \Sigma^* : (uw \in L(A) \Leftrightarrow vw \in L(A)). \tag{1.4}$$

Wir interpretieren Eigenschaft (1.4) als Definition der Äquivalenz der Worte u und v und führen deshalb die Nerode-Relation ein.

Definition 1.23. Σ sei eine endliche Menge und L sei eine Sprache über Σ, d.h. es gilt $L \subseteq \Sigma^*$.

(a) Die **Nerode-Relation** $\equiv_L$ für L ist eine 2-stellige Relation über Σ^*. Für alle Worte $u, v \in \Sigma^*$ definiere Nerode-Relation

$$u \equiv_L v \ :\Leftrightarrow \ \text{f.a. } w \in \Sigma^* \text{ gilt: } \big(uw \in L \Leftrightarrow vw \in L\big).$$

(b) Wir sagen, dass das Wort $w \in \Sigma^*$ die Worte $u, v \in \Sigma^*$ trennt, bzw. dass w ein **Zeuge** für die Inäquivalenz von u und v ist, wenn Zeuge

$$\Big(uw \in L \wedge vw \notin L\Big) \vee \Big(uw \notin L \wedge vw \in L\Big)$$

(c) Index(L) ist die Anzahl der Äquivalenzklassen von $\equiv_L$.

Die Nerode-Relation erlaubt, dass wir die Perspektive der DFAs aufgeben können und uns nur noch auf die Struktur der Sprache L konzentrieren dürfen.

Satz 1.24. Sei L eine beliebige Sprache. Dann ist die Nerode-Relation für L eine Äquivalenzrelation.

Beweis: Die Nerode-Relation $\equiv_L$ ist

1. reflexiv, denn f.a. $z \in \Sigma^*$ gilt $\big(uz \in L \Leftrightarrow uz \in L\big)$ und deshalb gilt $u \equiv_L u$ für alle Worte $u \in \Sigma^*$.
2. symmetrisch, denn aus $u \equiv_L v$ folgt $\big(uz \in L \Leftrightarrow vz \in L\big)$ f.a. $z \in \Sigma^*$ und damit auch $\big(vz \in L \Leftrightarrow uz \in L\big)$ f.a. $z \in \Sigma^*$. Die letzte Aussage ist aber äquivalent zu $v \equiv_L u$.
3. transitiv, denn aus $u \equiv_L v$ und $v \equiv_L w$ folgt f.a. $z \in \Sigma^*$ $\big(uz \in L \Leftrightarrow vz \in L\big) \wedge \big(vz \in L \Leftrightarrow wz \in L\big)$. Also folgt $\big(uz \in L \Leftrightarrow wz \in L\big)$ f.a. $z \in \Sigma^*$ und damit $u \equiv_L w$.

□

Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein DFA für die Sprache $L = L(A)$. Wenn $k = \text{Index}(L)$, dann besitzen die k verschiedenen Äquivalenzklassen der Nerode-Relation Vertreter $u^1, \ldots, u^k$, und diese Vertreter sind natürlich paarweise inäquivalent.

Angenommen, zwei dieser Vertreter, z.B. u^i und u^j (mit $i \neq j$), führen auf denselben Zustand in Q. Dann gilt

$$\widehat{\delta}(q_0, u^i) = \widehat{\delta}(q_0, u^j)$$

und es ist $\widehat{\delta}(q_0, u^i w) = \widehat{\delta}(q_0, u^j w)$ für alle Worte $w \in \Sigma^*$. Aber A akzeptiert die Sprache L und es ist $u^i w \in L \Leftrightarrow u^j w \in L$: Wir erhalten $u^i \equiv_L u^j$ im Widerspruch zur Inäquivalenz von u^i und u^j. Wir haben gezeigt

Satz 1.25. Für jeden DFA $A = (\Sigma, Q, \delta, q_0, F)$ gilt

$$\text{Index}(L(A)) \leqslant |Q|.$$

Mit anderen Worten: Wenn $\text{Index}(L(A)) = |Q|$, dann ist A ein minimaler Automat!

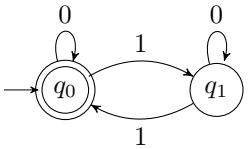
Beispiel 1.26. Wir betrachten die Sprache

$$\text{Parität } = \{w \in \{0,1\}^* : w \text{ hat eine gerade Anzahl von Einsen }\}.$$

Es ist $\varepsilon \not\equiv_{\text{Parität}} 1$ mit dem leeren Wort als Zeugen. Sei u ein beliebiges Wort mit gerade vielen Einsen und v ein beliebiges Wort mit ungerade vielen Einsen. Dann gilt

$$\Big(\varepsilon w \in \text{ Parität } \Leftrightarrow uw \in \text{ Parität } \Big) \text{ und } \Big(1w \in \text{ Parität } \Leftrightarrow vw \in \text{ Parität } \Big)$$

für alle $w \in \Sigma^*$. Mit anderen Worte, die Nerode-Relation für Parität hat genau zwei Äquivalenzklassen mit den jeweiligen Vertretern ε und 1. Der Index „zwei“ der Nerode-Relation stimmt überein mit der Zustandszahl des folgenden DFA für Parität:



Beispiel 1.27. In Beispiel 1.21 haben wir die Sprache

$$L = \{a, b\}^* \cdot \{ab\}$$

betrachtet, also die Menge aller Worte, die mit ab enden, und haben einen DFA für L mit drei Zuständen gefunden. Jetzt zeigen wir, dass der Index von L mindestens drei ist und konstruieren dazu die folgenden drei paarweise inäquivalenten Worte:

$$\varepsilon, a, ab$$

Das Wort ab trennen wir mit dem Zeugen ε von ε und a. Schließlich ist nur noch zu beobachten, dass $\varepsilon \not\equiv_L a$ mit dem Zeugen b gilt. Der in Beispiel 1.21 konstruierte DFA mit drei Zuständen ist minimal.

Beispiel 1.28. Sei Σ ein beliebiges Alphabet und u ein Wort über Σ Wir möchten das Teilwort-Problem studieren und betrachten deshalb die Sprache

$$L_u = \{\, w \in \Sigma^* \,:\, u \text{ ist ein Teilwort von } w \,\}.$$

Sei $u = u_1 \cdots u_k$ mit den Buchstaben $u_1, \ldots, u_k \in \Sigma$. Wir behaupten, dass der Index von L_u mindestens $k + 1$ ist. Wir betrachten dazu die $k + 1$ Präfixe

$$\varepsilon, u_1, u_1u_2, \ldots, u_1 \cdots u_i, \ldots, u_1 \cdots u_k.$$

Zwei verschiedene Präfixe $u_1 \cdots u_i$ und $u_1 \cdots u_j$ (mit $i < j$) lassen sich mit dem Zeugen $u_{j+1} \cdots u_k$ trennen, denn $u_1 \cdots u_j \cdot u_{j+1} \cdots u_k = u \in L_u$, aber $u_1 \cdots u_i u_{j+1} \cdots u_k$ kann als ein Wort mit weniger als k Buchstaben nicht in L_u liegen.

Editoren müssen das Teilwort-Problem blitzschnell lösen. Lässt sich ein kleiner DFA konstruieren, der genau alle Worte mit Teilwort u akzeptiert? Bei einer positiven Antwort können wir das Teilwort-Problem tatsächlich blitzschnell auf kleinem Speicherplatz lösen! Es gelingt sogar die Konstruktion eines DFA mit $k + 1$ Zuständen: Als Hinweis benutze, dass Zustand $i + 1$ vom Präfix $u_1 \cdots u_i$ erreicht wird. Beachte, dass dieser DFA minimal sein muss.

Beispiel 1.29. In Beispiel 1.22 haben wir einen DFA mit vier Zuständen konstruiert, der alle Binärdarstellungen mit durch sechs teilbarer Zahl akzeptiert. Sei $L = L(A)$ die in (1.3) definierte Sprache. Wir zeigen, dass Index von L mindestens vier ist.

Betrachte dazu die Worte $00, 01, 10, 11$. 00 lässt sich durch das leere Wort von $01, 10, 11$ trennen. Desweiteren können wir 11 von 01 und 10 trennen, wenn wir 0 als Zeugen verwenden. Schließlich folgt $01 \not\equiv_L 10$ mit dem Zeugen 10, denn 0110 stellt die Zahl sechs dar, während 1010 die nicht durch sechs teilbare Zahl zehn darstellt.

Der DFA nach Verschmelzen der Zustände 1 und 4, bzw. 2 und 5 ist somit minimal.

Der Index der Nerode-Relation $\equiv_{L(A)}$ stimmt mit der minimalen Zustandszahl eines DFA für die Sprache $L(A)$ überein. Und es kommt noch besser: Der Verschmelzungsautomat A' ist minimal!

Satz 1.30. Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein DFA.

(a) Der Verschmelzungsautomat A' ist minimal.

(b) Index($L(A)$) stimmt überein mit der minimalen Zustandszahl eines DFA für die Sprache $L(A)$.

Beweis: Angenommen, Worte $u, v \in \Sigma^*$ führen in A' zu verschiedenen Zuständen. Dann folgt $\widehat{\delta}'(q_0', u) \neq \widehat{\delta}'(q_0', v)$.

Was passiert im Ausgangsautomaten A mit den Worten u und v? Für A sei $p = \widehat{\delta}(q_0, u)$ und $q = \widehat{\delta}(q_0, v)$. Dann folgt $[p]_A = [\widehat{\delta}(q_0, u)]_A = \widehat{\delta}'(q_0', u) \neq \widehat{\delta}'(q_0', v) = [\widehat{\delta}(q_0, v)]_A = [q]_A$.
Es ist also $p \not\equiv_A q$ und es gibt einen Zeugen $w \in \Sigma^*$ für die Nicht-Äquivalenz. Also:

$$\Big(\widehat{\delta}(p, w) \in F \text{ und } \widehat{\delta}(q, w) \notin F\Big) \text{ oder } \Big(\widehat{\delta}(p, w) \notin F \text{ und } \widehat{\delta}(q, w) \in F\Big),$$

bzw.

$$\Big(\widehat{\delta}(q_0, uw) \in F \text{ und } \widehat{\delta}(q_0, vw) \notin F\Big) \text{ oder } \Big(\widehat{\delta}(q_0, uw) \notin F \text{ und } \widehat{\delta}(q_0, vw) \in F\Big)$$

bzw.

$$\Big(uw \in L(A) \text{ und } vw \notin L(A)\Big) \text{ oder } \Big(uw \notin L(A) \text{ und } vw \in L(A)\Big).$$

Wenn u und v in A' zu verschiedenen Zuständen führen, dann folgt $u \not\equiv_{L(A)} v$: Die Anzahl der Zustände von A' ist höchstens so groß wie der Index von $L(A)$. Aber nach Satz 1.25 ist die Anzahl der Zustände eines jeden DFA für $L(A)$ mindestens so groß wie der Index und A' ist minimal. Aber nicht nur das: Der Index stimmt mit der Zustandszahl des minimalen Automaten A' überein. □

1.4 Reguläre Sprachen

reguläre Sprache

Definition 1.31. Sei Σ eine endliche Menge. Eine Teilmenge $L \subseteq \Sigma^*$ heißt eine **reguläre Sprache**, wenn es einen DFA A gibt mit

$$L = L(A).$$

Gibt es Teilmengen von Σ^*, die keine regulären Sprachen sind? Zum Beispiel, ist

$$L = \{\, a^n b^n \;:\; n \in \mathbb{N} \,\}$$

eine reguläre Sprache? Mit Hilfe der Nerode-Relation können wir genau sagen, wann eine Sprache regulär ist.

Satz 1.32 (Der Satz von Nerode).
Eine Sprache $L \subseteq \Sigma^*$ ist regulär $\Leftrightarrow$ Index(L) ist endlich.

Beweis: $\Rightarrow$ Die Sprache $L \subseteq \Sigma^*$ sei regulär. Dann gibt es einen DFA A mit $L = L(A)$. A hat mindestens Index(L) viele Zustände und Index(L) ist endlich.

Index(L) sei endlich. Dann gibt es einen Automaten A mit $L = L(A)$, aber möglicherweise unendlich vielen Zuständen. Der Äquivalenzklassenautomat A' hat aber höchstens Index(L) Zustände: Es ist $L = L(A')$ für den DFA A' und L ist regulär. □

Satz 1.33. $L = \{a^n b^n \;:\; n \in \mathbb{N}\}$ ist nicht regulär: DFAs können nicht (unbeschränkt) zählen..

Beweis: Wir müssen unendlich viele Worte $u_k \in \{a,b\}^*$ bestimmen, so dass

$$u_k \not\equiv_L u_\ell$$

für alle $k \neq \ell$ gilt. Setze $u_i = a^i$. Für $k \neq \ell$ gilt $u_k b^k \in L$ und $u_\ell b^k \notin L$: $u_k \not\equiv_L u_\ell$ folgt. Wir erhalten $\text{Index}(L) = \infty$ und L ist nicht regulär. □

Satz 1.34. $L = \{ww \ : \ w \in \{a,b\}^*\}$ ist nicht regulär: DFAs können sich nur beschränkt viele Dinge merken.

Beweis: Wir bestimmen unendlich viele Worte $u_k \in \{a,b\}^*$, so dass

$$u_k \not\equiv_L u_\ell$$

für alle $k \neq \ell$ gilt. Setze $u_i = a^i b$. Für $k \neq \ell$ gilt $u_k a^k b \in L$ und $u_\ell a^k b \notin L$: $u_k \not\equiv_L u_\ell$ folgt. Wir erhalten $\text{Index}(L) = \infty$ und L ist nicht regulär. □

Weitere nicht-reguläre Sprachen L erhalten wir indem wir zeigen, dass $\text{Index}(L)$ unendlich groß ist. Das gelingt häufig sogar mit unendlich vielen Worten $u_1, u_2, \ldots$ und $v_1, v_2, \ldots$, die die Eigenschaften

- $u_i v_i \in L$ und
- $u_j v_i \notin L$ für $i \neq j$.

besitzen.

Satz 1.35. Keine der folgenden Sprachen ist regulär.

(a) $L_1 = \{\, a^n b^m \ : \ n,m \in \mathbb{N},\ n \leqslant m \,\}$: DFAs können nicht vergleichen,

(b) $L_2 = \{\, a^n b^m c^{n+m} \ : \ n,m \in \mathbb{N} \,\}$: DFAs können nicht addieren, weil sie nur ein beschränkt großes Gedächtnis haben,

(c) $L_3 = \{\, a^{n^2} \ : \ n \in \mathbb{N} \,\}$: DFAs können nicht quadrieren,

(d) $L_4 = \{\, w \in \{a,b\}^* \ : \ w \text{ ist ein Palindrom} \,\}$: Endliche Automaten haben nur ein beschränkt großes Gedächtnis.

1.5 Zusammenfassung

Wir haben Moore und Mealy Automaten als DFAs mit Ausgabe eingeführt und haben gesehen, dass diese Automatenmodelle für die (partielle) Modellierung von Schaltwerken geeignet sind.

Die Minimierung von DFAs (oder Moore, bzw. Mealy Automaten) ist wichtig, um effizient arbeiten zu können. Wir haben die Verschmelzungsrelation $\equiv_A$ für einen DFA A eingeführt und den Äquivalenklassenautomaten erhalten, indem wir äquivalente Zustände von A verschmolzen haben.

Mit Hilfe der Nerode-Relation haben wir gezeigt, dass der Äquivalenzklassenautomat minimal ist: Der Index einer Sprache stimmt überein mit der minimalen Zustandszahl eines DFA für die Sprache.

Wir haben benutzt, dass Verschmelzungsrelation und Nerode-Relation Äquivalenzrelationen sind: Die Äquivalenzklassen einer Äquivalenzrelation über V bilden eine disjunkte Vereinigung von V.