Vorlesung
Informatik III – Theoretische Informatik

Formale Sprachen, Berechenbarkeit, Komplexitätstheorie

Matthias Heizmann

Basierend auf einem Mitschrieb von Ralph Lesch*

der von Prof. Dr. Peter Thiemann im WS 2015/16 gehaltenen Vorlesung

WS 2017/18

Zuletzt aktualisiert: 2017-11-10

*ralph.lesch@neptun.uni-freiburg.de

Inhaltsverzeichnis

1 Vorspann: Sprachen

Vorlesung: 18.10.2017

Def. 1.1: Ein *Alphabet* ist eine endliche Menge von *Zeichen*. ◇

Zeichen sind hier beliebige abstrakte Symbole.

Bsp.: für Alphabete, die in dieser Vorlesung, im täglichem Umgang mit Computern oder in der Forschung an unserem Lehrstuhl eine Rolle spielen.

- $\{a, \ldots, z\}$
- $\{0, 1\}$
- $\{\mathit{rot}, \mathit{gelb}, \mathit{grün}\}$ (Ampelfarben)
- Die Menge aller ASCII Symbole
- Die Menge aller Statements eines Computerprogramms

Wir verwenden typischerweise den griechischen Buchstaben Σ als Namen für ein Alphabet und die lateinischen Buchstaben a, b, c als Namen für Zeichen. [1] Im Folgenden sei Σ immer ein beliebiges Alphabet. [2]

Def. 1.2: Wir nennen eine endliche Folge von Elementen aus Σ ein *Wort* und schreiben solch eine Folge immer ohne Trennsymbole wie z.B. Komma.[3] Die leere Folge nennen wir das *leere Wort* und verwenden immer den griechischen Buchstaben ε um das *leere Wort* zu bezeichnen.[4] Wir bezeichnen die Menge aller Wörter mit Σ^* und die Menge aller nicht leeren Wörter mit Σ^+. Die *Länge* eines Wortes, $|\cdot| : \Sigma^* \to \mathbb{N}$, ist die Anzahl der Elemente der Folge. ◇

Wir verwenden typischerweise u, v, w als Namen für Wörter.

Bsp.: für Wörter über $\Sigma = \{a, \ldots, z\}$

- `rambo` (Länge 5)
- `eis`, `ies` (beide Länge 3 aber ungleich)

[1] Dies ist eine Konventionen analog zu den folgenden die Sie möglicherweise in der Schule befolgten: Verwende n, m für natürliche Zahlen. Verwende α, β für Winkel in Dreiecken. Verwende A für Matrizen.

[2] Dieser Satz dient dazu, dass die Autoren dieses Skriptes nicht jede Definition mit "Sei Σ ein Alphabet..." beginnen müssen.

[3] Wir schreiben also z.B. `einhorn` statt `e,i,n,h,o,r,n`.

[4] Eine analoge Konvention die sie aus der Schule kennen: Verwende immer π für die Kreiszahl

- ε (Länge 0)

Wörter lassen sich „verketten"/„hintereinanderreihen". Die entsprechende Operation heißt *Konkatenation*, geschrieben „$\cdot$" (wie Multiplikation).

Def. 1.3 (Konkatenation von Wörtern)**:** Die *Konkatenation*, $\cdot : \Sigma^* \times \Sigma^* \to \Sigma^*$, ist für $u = u_1 \dots u_n \in \Sigma^*$ und $v = v_1 \dots v_m \in \Sigma^*$ definiert durch: $u \cdot v = u_1 \dots u_n v_1 \dots v_m$ ◇

Bsp.:

- $\texttt{eis} \cdot \texttt{rambo} = \texttt{eisrambo}$
- $\texttt{rambo} \cdot \varepsilon = \texttt{rambo} = \varepsilon \cdot \texttt{rambo}$

Eigenschaften von „$\cdot$":

- Assoziativität
- ε ist neutrales Element
- *nicht* kommutativ

Der Konkatenationsoperator „$\cdot$" wird oft weggelassen (ähnlich wie der Mulitplikationsoperator in der Arithmetik). Ebenso können durch die Assoziativität Klammern weggelassen werden:

$w_1 w_2 w_3$ steht also auch für $w_1 \cdot w_2 \cdot w_3$, für $(w_1 \cdot w_2) \cdot w_3$ und für $w_1 \cdot (w_2 \cdot w_3)$

Bemerkung: Die Zeichenfolge $\texttt{rambo}\varepsilon$ ist *kein* Wort. Diese Zeichenfolge ist lediglich eine Notation für eine Konkatenationsoperation die ein Wort der Länge 5 (nämlich `rambo` beschreibt.

Wörter lassen sich außerdem *potenzieren*:

Def. 1.4: Die *Potenzierung* von Wörtern, $\cdot^{\cdot} : \Sigma^* \times \mathbb{N} \to \Sigma^*$, ist induktiv definiert durch

1. $w^0 = \varepsilon$
2. $w^{n+1} = w \cdot w^n$

◇

Bsp.: $\texttt{eis}^3 \overset{(2.)}{=} \texttt{eis}\cdot\texttt{eis}^2 \overset{\text{zwei mal (2.)}}{=} \texttt{eis}\cdot\texttt{eis}\cdot\texttt{eis}\cdot\texttt{eis}^0 \overset{(1.)}{=} \texttt{eis}\cdot\texttt{eis}\cdot\texttt{eis}\cdot\varepsilon = \texttt{eiseiseis}$

Def. 1.5: Eine *Sprache* über Σ ist eine Menge $L \subseteq \Sigma^*$. ◇

Bsp.:

- $\{\texttt{eis}, \texttt{rambo}\}$
- $\{w \in \{0,1\}^* \mid w$ ist Binärcodierung einer Primzahl$\}$
- $\{\}$ (die „leere Sprache")
- $\{\varepsilon\}$ (ist verschieden von der leeren Sprache)
- Σ^*

Sämtliche Mengenoperationen sind auch Sprachoperationen, insbesondere Schnitt ($L_1 \cap L_2$), Vereinigung ($L_1 \cup L_2$), Differenz ($L_1 \setminus L_2$) und Komplement ($\overline{L_1} = \Sigma^* \setminus L_1$).

Weitere Operationen auf Sprachen sind Konkatenation und Potenzierung, sowie der *Kleene Abschluss*.

Def. 1.6 (Konkatenation und Potenzierung von Sprachen)**:** Sei $U, V \subseteq \Sigma^*$. Dann ist die *Konkatenation* U und V definiert durch

$$U \cdot V = \{uv \mid u \in U, v \in V\}$$

und die *Potenzierung* von U induktiv definiert durch

1. $U^0 = \{\varepsilon\}$
2. $U^{n+1} = U \cdot U^n$

◇

Bsp.:

- $\{\texttt{eis}, \varepsilon\} \cdot \{\texttt{rambo}\} = \{\texttt{eisrambo}, \texttt{rambo}\}$
- $\{\texttt{eis}, \varepsilon\} \cdot \{\} = \{\}$
- $\{\}^0 = \{\varepsilon\}$
- $\{\}^4 = \{\}$

Wie bei der Konkatenation von Wörtern dürfen wir den Konkatenationsoperator auch weglassen.

Def. 1.7 (Kleene-Abschluss, Kleene-Stern)**:** Sei $U \subseteq \Sigma^*$. Der *Kleene-Abschluss* ist definiert als

1. $U^* = \bigcup_{n \in \mathbb{N}} U^n \quad [\ni \varepsilon]$
2. $U^+ = \bigcup_{n \geq 1} U^n$

◇

2 Reguläre Sprachen und endliche Automaten

Vorlesung: 20.10.17

Wie können wir potentiell unendlich große Mengen von Wörtern darstellen? Eine Lösung für dieses Problem sahen wir bereits im vorherigen Kapitel, als wir die (unendlich große) Menge der binärcodierten Primzahlen mit Hilfe der folgenden Zeile darstellten.

$$L_{\mathsf{prim}} = \{w \in \{0,1\}^* \mid w \text{ ist Binärcodierung einer Primzahl}\}$$

Ein weiteres Beispiel ist die folgende Zeile.

$$L_{\mathsf{even}} = \{w \in \{0,1\}^* \mid \text{ die Anzahl der Einsen in } w \text{ ist gerade. }\}$$

Ein häufig interessante Fragestellung für ein gegebenes Wort w und eine Sprache L ist: "Ist w in L enthalten?" (Also gilt $w \in L$?) Wir nennen dieses Entscheidunsproblem das *Wortproblem*. Eine konkrete Instanz des Wortproblems wäre z.B. $1100101 \in L_1$? oder $1100101 \in L_2$?

Die obige Darstellung der unendlichen Mengen L_1 und L_2 ist zwar sehr kompakt, wir können daraus aber nicht direkt ein Vorgehen zur Lösung des Wortproblems ableiten. Wir müssen zunächst verstehen, was die Begriffe "Binärcodierung", "Primzahl" oder "gerade Anzahl" bedeuten und für L_1 und L_2 jeweils einen Algorithmus zur Entscheidung entwickeln.

In diesem Kapitel werden wir mit *endlichen Automaten* einen weiteren Formalismus kennenlernen um (potentiell unendlich große) Mengen von Wörtern darzustellen. Ein Vorteil dieser Darstellung ist, dass es einen einheitlichen und effizienten Algorithmus für das Wortproblem gibt. Wir werden aber auch sehen, dass sich nicht jede Sprache (z.B. L_1) mit Hilfe eines endlichen Automaten darstellen lässt.

2.1 Endliche Automaten

Wir beschreiben zunächst informell die Bestandteile eines endlichen Automaten:

Endliches Band (read-only, jede Zelle enthält ein $a_i \in \Sigma$, der Inhalt des Bandes ist das *Eingabewort*, bzw. die *Eingabe*)

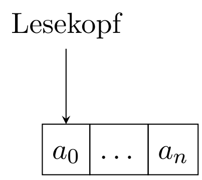


Abb. 1: Endliches Band

Lesekopf

- Der *Lesekopf* zeigt auf ein Feld des Bandes, oder hinter das letzte Feld.
- Er bewegt sich feldweise nach rechts; andere Bewegungen (Vor- bzw. Zurückspulen) sind nicht möglich.
- Wenn er hinter das letzte Zeichen zeigt, *stoppt* der Automat. Er muss sich nun „entscheiden” ob er das Wort *akzeptiert* oder nicht.

Zustände q aus *endlicher* Zustandsmenge Q

Startzustand $q^{\mathsf{init}} \in Q$

Akzeptierende Zustände $F \subseteq Q$

Transitionsfunktion Im Zustand q beim Lesen von a gehe nach Zustand $\delta(q) = q'$.

Der endliche Automat akzeptiert eine Eingabe, falls er in einem akzeptierenden Zustand stoppt.

Bsp. 2.1: Aufgabe:
„Erkenne alle Stapel von Macarons in denen höchstens ein grünes Macaron vorkommt.”


[5]

Ein passendes Alphabet wäre $\Sigma = \{\texttt{grün}, \texttt{nicht-grün}\}$. Wir definieren die folgenden Zustände. (die Metapher hier ist: „wenn ich mehr als einen grünen Macaron esse wird mir übel, und das wäre nicht akzeptabel”)

Zustand	Bedeutung
q_0	„alles gut”
q_1	„mir wird schon flau”
q_2	„mir ist übel”

Der Startzustand ist q_0. Akzeptierende Zustände sind q_0 und q_1. Die Transistionsfunktion δ ist

[5] Von links nach rechts:
By Mariajudit - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=48726001
By Michelle Naherny - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=44361114
By Keven Law - originally posted to Flickr as What's your Colour???, CC BY-SA 2.0, https://commons.wikimedia.org/w/index.php?curid=6851868

	`grün`	`nicht-grün`	
q_0	q_1	q_0	wechsle nach q_1 falls `grün`, ansonsten verweile
q_1	q_2	q_1	wechsle nach q_2 falls `grün`, ansonsten verweile
q_2	q_2	q_2	verweile, da es nichts mehr zu retten gibt

Def. 2.1 (DEA)**:** Ein *deterministischer endlicher Automat (DEA)*, (DFA $\hat{=}$ deterministic finite automaton) ist ein 5-Tupel

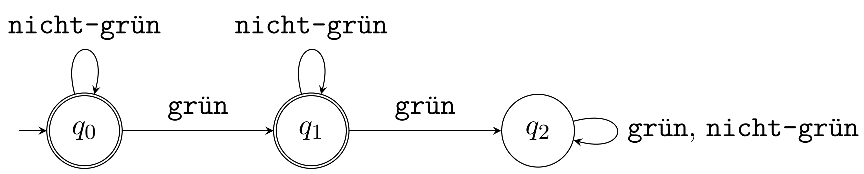
$$\mathcal{A} = (\Sigma, Q, \delta, q^{\mathsf{init}}, F)$$

dabei ist

- Σ ein Alphabet,
- Q eine *endliche* Menge deren Elemente wir *Zustände* nennen,
- $\delta : Q \times \Sigma \to Q$ eine Funktion die wir *Transitionsfunktion* nennen,
- $q^{\mathsf{init}} \in Q$ ein Zustand den wir *Startzustand* nennen und
- $F \subseteq Q$ eine Teilmenge der Zustände deren Elemente wir *akzeptierende* Zustände nennen.

$\diamond$

DEAs lassen sich auch graphisch darstellen. Dabei gibt man für den Automaten einen gerichteten Graphen an. Die Knoten des Graphen sind die Zustände und mit Zeichen beschriftete Kanten zeigen, welchen Zustandsübergang die Transitionsfunktion für das nächste Zeichen erlaubt. Der Startzustand ist mit einem unbeschrifteten Pfeil markiert und akzeptierende Zustände sind doppelt eingekreist. Hier ist die graphische Darstellung von $A_{\texttt{Macaron}}$ aus Beispiel 2.1:



Die folgenden beiden Definitionen erlauben uns mit Hilfe eines DEAs eine Sprache zu charakterisieren.

Def. 2.2: Die *induktive Erweiterung* von $\delta : Q \times \Sigma \to Q$ auf Worte $\tilde{\delta} : Q \times \Sigma^* \to Q$ ist (induktiv) definiert durch

1. $\tilde{\delta}(q, \varepsilon) = q$ (Wortende erreicht)
2. $\tilde{\delta}(q, aw) = \tilde{\delta}(\delta(q, a), w)$ (Rest im Folgezustand verarbeiten)

$\diamond$

Def. 2.3: Sei $\mathcal{A} = (\Sigma, Q, \delta, q^{\mathsf{init}}, F)$. Wir sagen ein Wort $w \in \Sigma^*$ wird von $\mathcal{A}$ *akzeptiert* falls $\tilde{\delta}(q^{\mathsf{init}}, w) \in F$. Die *von $\mathcal{A}$ akzeptierte Sprache*, geschrieben $L(\mathcal{A})$, ist die Menge aller Wörter die von $\mathcal{A}$ akzeptiert werden. D.h.,

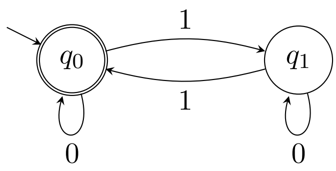
$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \tilde{\delta}(q^{\mathsf{init}}, w) \in F\}.$$

Eine durch einen DEA akzeptierte Sprache heißt *regulär*. $\diamond$

Bsp. 2.2: Der Automat für die Sprache

$$L_{\mathsf{even}} = \{w \in \{0,1\}^* \mid \text{ die Anzahl der Einsen in } w \text{ ist gerade. }\}$$

aus der Einleitung dieses Kapitels hat die folgende graphische Repräsentation.



Frage: Gegeben seien zwei reguläre Sprachen L_1, L_2 über einem gemeinsamen Alphabet Σ; ist dann auch die Vereinigung $L_1 \cap L_2$ eine reguläre Sprache? Wir beantworten diese Frage mit dem folgenden Satz.

Satz 2.1: Reguläre Sprachen sind abgeschlossen unter der Schnittoperation. (D.h. Gegeben zwei reguläre Sprachen L_1, L_2 über Σ dann ist auch der Schnitt $L_1 \cap L_2$ eine reguläre Sprache.)

Beweis: [6] Da L_1 und L_2 regulär sind, gibt es zwei DEAs $A_1 = (\Sigma, Q_1, \delta_1, {q^{\mathsf{init}}}_1, F_1)$ und $A_2 = (\Sigma, Q_2, \delta_2, {q^{\mathsf{init}}}_2, F_2)$ mit $L_1 = L(A_1)$ und $L_2 = L(A_2)$. Wir konstruieren nun zunächst den *Produktautomaten für Schnitt* $A_\cap = (\Sigma, Q_\cap, \delta_\cap, {q^{\mathsf{init}}}_\cap, F_\cap)$ wie folgt.

$$\begin{aligned}
Q_\cap &= Q_1 \times Q_2 \\
\delta_\cap((q_1, q_2), a) &= (\delta_1(q_1, a), \delta_2(q_2, a)) \ , \quad \text{für alle } a \in \Sigma \\
{q^{\mathsf{init}}}_\cap &= ({q^{\mathsf{init}}}_1, {q^{\mathsf{init}}}_2) \\
F_\cap &= F_1 \times F_2
\end{aligned}$$

[6] Dieser erste Beweis ist außergewöhnlich detailiert. Im den folgenden Beweisen werden wir einfache Umformungen zusammenfassen.

Anschließend zeigen wir, dass $L(A_\cap) = L(A_1) \cap L(A_2)$ gilt. Hierfür zeigen wir zunächst via Induktion über die Länge von w, dass für alle $w \in \Sigma^*$, für alle $q_1 \in Q_1$ und für alle $q_2 \in Q_2$ die folgende Gleichung gilt.

$$\tilde{\delta}_\cap((q_1, q_2), w) = (\tilde{\delta}_1(q_1, w), \tilde{\delta}_2(q_2, w))$$

Der Induktionsanfang für $n = 0$ folgt dabei direkt aus Def. 2.2, da ε das einzige Wort der Länge 0 ist.

$$\tilde{\delta}_\cap((q_1, q_2), \varepsilon) = (q_1, q_2)$$

Den Induktionsschritt $n \rightsquigarrow n+1$ zeigen wir mit Hilfe der folgenden Umformungen, wobei $a \in \Sigma$ ein beliebiges Zeichen und $w \in \Sigma^n$ ein beliebiges Wort der Länge n ist.

$$\begin{aligned}
\tilde{\delta}_\cap((q_1, q_2), aw) &\overset{\text{Def. 2.2}}{=} \tilde{\delta}_\cap(\delta_\cap((q_1, q_2), a), w) \\
&\overset{\text{def. } \delta_\cap}{=} \tilde{\delta}_\cap((\delta_1(q_1, a), \delta_2(q_2, a)), w) \\
&\overset{\text{I.V.}}{=} (\tilde{\delta}_1(\delta_1(q_1, a), w), \tilde{\delta}_2(\delta_2(q_2, a), w)) \\
&\overset{Def.\ 2.2}{=} (\tilde{\delta}_1(q_1, aw), \tilde{\delta}_2(q_2, aw))
\end{aligned}$$

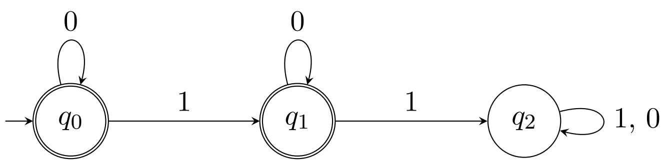
Schließlich zeigen wir $L(A_\cap) = L(A_1) \cap L(A_2)$ mit Hilfe der folgenden Umformungen für ein beliebiges $w \in \Sigma^*$.

$$\begin{aligned}
w \in L(A_\cap) &\overset{Def.\ 2.3}{\text{gdw}} \tilde{\delta}_\cap({q^{\mathsf{init}}}_\cap, w) \in F_\cap \\
&\overset{\text{def. } {q^{\mathsf{init}}}_\cap}{\text{gdw}} \tilde{\delta}_\cap(({q^{\mathsf{init}}}_1, {q^{\mathsf{init}}}_2), w) \in F_\cap \\
&\text{gdw} \quad (\tilde{\delta}_1({q^{\mathsf{init}}}_1, w), \tilde{\delta}_2({q^{\mathsf{init}}}_2, w)) \in F_\cap \\
&\overset{\text{def. } F_\cap}{\text{gdw}} \tilde{\delta}_1({q^{\mathsf{init}}}_1, w) \in F_1 \text{ und } \tilde{\delta}_2({q^{\mathsf{init}}}_2, w) \in F_2 \\
&\overset{Def.\ 2.3}{\text{gdw}} w \in L(A_1) \text{ und } w \in L(A_2)
\end{aligned}$$

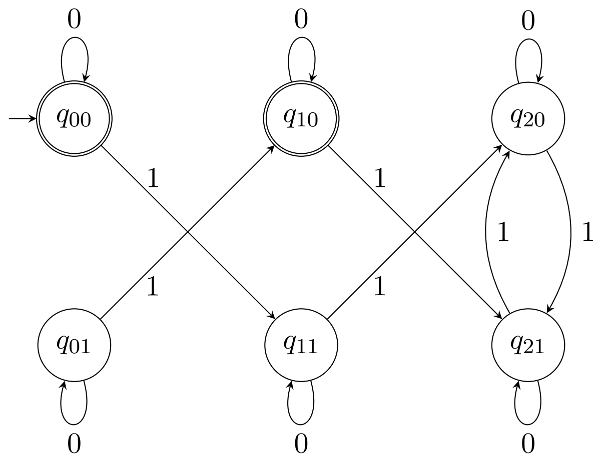
□

Vorlesung: 25.10.17

Im folgenden Beispiel sei A_1 der DEA über dem Alphabet $\Sigma = \{0, 1\}$ dessen graphische Repräsentation nahezu mit $A_{\texttt{Macaron}}$ identisch ist.



Bsp. 2.3: Der Produktautomat für Schnitt von A_1 und A_{even} hat die folgende graphische Repräsentation, wobei wir um Platz zu sparen "q_{ij}" statt "(q_i, q_j)" schreiben.



2.2 Minimierung endlicher Automaten

Beobachtung: Der Zustand q_{01} im Beispiel 2.3 scheint nutzlos. Wir charakterisieren diese "Nutzlosigkeit" formal wie folgt.

Def. 2.4: Ein Zustand $q \in Q$ heißt *erreichbar*, falls ein $w \in \Sigma^*$ existiert, so dass $\hat{\delta}(q^{\mathsf{init}}, w) = q$. ◇

Bemerkung: Die Menge der erreichbaren Zustände kann mit dem folgenden Verfahren in $O(|Q| * |\Sigma|)$ berechnet werden.

- Fasse A als Graph auf.
- Wende Tiefensuche an und markiere dabei alle besuchten Zustände.
- Die markierten Zustände bilden die Menge der erreichbaren Zustände.

Beobachtung: Auch nach dem Entfernen der nicht erreichbaren Zustände q_{01} und q_{10} scheint der DEA aus Bsp. 2.3 unnötig groß. Das Verhalten des DEA in den Zuständen q_{11}, q_{20} und q_{21} ist sehr ähnlich. Wir charakterisieren diese "Ähnlichkeit" formal wie folgt.

Def. 2.5: Wir nennen zwei Zustände $p, q \in Q$ eines DEA *äquivalent*, geschrieben $p \equiv q$, falls

$$\forall w \in \Sigma^*, \tilde{\delta}(p, w) \in F \text{ gdw } \tilde{\delta}(q, w) \in F$$

◇

Bsp. 2.4: Für Bsp. 2.3 gilt: Die Zustände q_{11}, q_{20} und q_{21} aus sind paarweise äquivalent, die Zustände q_{00} und q_{10} sind äquivalent, keine weiteren Zustandspaare sind äquivalent.

Geschrieben als Menge von Paaren sieht die Relation $\equiv \subset Q \times Q$ also wie folgt aus:

$$\{\, (q_{00}, q_{10}), (q_{10}, q_{01}),\ (q_{11}, q_{20}), (q_{20}, q_{11}),\ (q_{20}, q_{21}), (q_{21}, q_{20}),\ (q_{21}, q_{11}), (q_{11}, q_{21}) \,\}$$

2.2.1 Exkurs: Äquivalenzrelationen

Sie haben Äquivalenzrelationen bereits in "Mathematik II für Studierende der Informatik"[7] kennengelernt. Dieser kurze Exkurs wiederholt die für unsere Vorlesung relevanten Definitionen. Sie X eine beliebige Menge. Eine *Relation* R über X ist eine Teilmenge des Produktes $X \times X$ (d.h. $R \subseteq X \times X$).

Eine Relation $R \subseteq X \times X$ heißt

- *reflexiv*, wenn $\forall x \in X\ (x, x) \in R$,
- *symmetrisch*, wenn $\forall x, y \in X\ (x, y) \in R \quad \Rightarrow \quad (y, x) \in R$,
- *transitiv*, wenn $\forall x, y, z \in X\ (x, y) \in R \land (y, z) \in R \quad \Rightarrow \quad (x, z) \in R$.

Bsp. 2.5: Im folgenden interessieren wir uns nur Relationen die alle drei Eigenschaften erfüllen, aber die folgenden Beispiele sollen helfen sich mit diesen Eigenschaften vertraut zu machen.

	reflexiv	symmetrisch	transitiv
"gewinnt" bei Schere, Stein, Papier	nein	nein	nein
$(\mathbb{N}, <)$	nein	nein	ja
$(\mathbb{N}, \neq)$	nein	ja	nein
die leere Relation	nein	ja	ja
$\{(a, b) \in \mathbb{Z} \times \mathbb{Z} \mid a - b \leq 3\}$	ja	nein	nein
$(\mathbb{N}, \leq)$	ja	nein	ja
direkte genetische Verwandtschaft	ja	ja	nein
logische Äquivalenz von Formeln	ja	ja	ja

[7] `http://home.mathematik.uni-freiburg.de/junker/ss17/matheII.html`

Bemerkung: Wir können kein Beispiel für eine nicht leere, symmetrische, transitive Relation finden die nicht Reflexiv ist. Für nicht leere Relationen folgt Reflexivität bereits aus Symmetrie und Transitivität: $(a, b) \in R \overset{\text{sym}}{\Rightarrow} (b, a) \in R \overset{\text{trans}}{\Rightarrow} (a, a) \in R$.

Def. 2.6: Eine Äquivalenzrelation R ist eine Relation die reflexiv, symmetrisch und transitiv ist.

Für ein $x \in X$ nennen wir die Menge $\{y \in X \mid y \equiv x\}$ die *Äquivalenzklasse* von x und verwenden die Notation $[x]_R$ für diese Menge. Wenn aus dem kontext klar ist, welche Relation gemeint ist dürfen wir das Subskript $\cdot_R$ auch weglassen und schreiben nur $[x]$.

Wenn wir eine Äquivalenzklase mit Hilfe der Notation $[x]_R$ beschreiben, nennen wir x den *Repräsentanten* dieser Äquivalenzklasse.

Wir nennen die Anzahl der Äquivalenzklassen von R den *Index* von R. $\diamond$

Zwei Fakten (ohne Beweis).

Fakt 1 Die Äquivalenzklassen von R sind paarweise disjunkt.

Fakt 2 Die Vereinigung aller Äquivalenzklassen ist die Menge X.

Hiermit endet der Exkurs zu Äquivalenzrelationen und wir wollen mit diesem Wissen die oben definierte Relation $\equiv \subseteq Q \times Q$ genauer analysieren.

Lemma 2.2: Die Relation „$\equiv$" ist eine *Äquivalenzrelation.*

Beweis: Die Relation $\equiv$ ist offensichtlich reflexiv. Die Symmetrie und Transitivität von $\equiv$ folgt aus der Transitivität und Symmetrie der logischen Interpretation von „genau dann wenn" (gdw). $\square$

Bsp. 2.6: Für den DEA aus Bsp. 2.3 hat die Relation $\equiv$ drei Äquivalenzklassen. [8]

$$
\begin{aligned}
[q_{00}] &= \{q_{00}, q_{10}\}, \\
[q_{01}] &= \{q_{01}\}, \\
[q_{11}] &= \{q_{11}, q_{20}, q_{21}\}
\end{aligned}
$$

Idee: "Verschmelze" alle Zustände aus einer Äquivalenzklasse zu einem einzigen Zustand. Bedenken: Bei einem DEA hat jeder Zustand hat für jedes Zeichen einen Nachfolger.

[8] Den Zustand q_{11} als Repräsentanten für die dritte Äquivalenzklasse zu wählen ist eine völlig willkürliche Entschidung. Wir könnten genauso gut q_{20} oder q_{21} wählen.

Wenn wir Zustände verschmelzen könnte es mehrere Nachfolger geben und das Resultat wäre kein wohldefinierter DEA mehr.

Das folgende Lemma zeigt dass unsere Bedenken nicht gerechtfertigt sind. Sind zwei Zustände äquivalent so sind auch für jedes Zeichen ihre Nachfolger äquivalent.

Lemma 2.3: Für alle $p, q \in Q$ gilt:

$$p \equiv q \quad \Rightarrow \quad \forall a \in \Sigma \;\; \delta(p,a) \equiv \delta(q,a)$$

BEWEIS:

$$\begin{array}{lcl} p \equiv q & \text{gdw} & \forall w \in \Sigma^* : \tilde{\delta}(p,w) \in F \Leftrightarrow \tilde{\delta}(q,w) \in F \\ & \text{gdw} & (p \in F \Leftrightarrow q \in F) \wedge \forall a \in \Sigma : \forall w \in \Sigma^* : \tilde{\delta}(p,aw) \in F \Leftrightarrow \tilde{\delta}(q,aw) \in F \\ & \text{impliziert} & \forall a \in \Sigma : \forall w \in \Sigma^* : \tilde{\delta}(\delta(p,a),w) \in F \Leftrightarrow \tilde{\delta}(\delta(q,a),w) \in F \\ & \text{gdw} & \forall a \in \Sigma : \delta(p,a) \equiv \delta(q,a) \end{array}$$

□

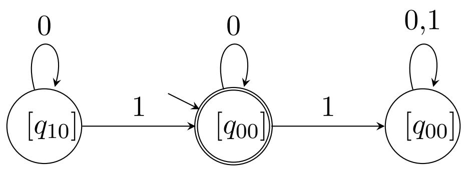
Wir formalisieren das “Verschmelzen” von Zuständen wie folgt.

Def. 2.7: Der Äquivalenzklassenautomat $\mathcal{A}_\equiv = (Q_\equiv, \Sigma, \delta_\equiv, {q^{\mathsf{init}}}_\equiv, F_\equiv)$ zu einem DEA $\mathcal{A} = (Q, \Sigma, \delta, q^{\mathsf{init}}, F)$ ist bestimmt durch:

$$\begin{array}{rlrl} Q_\equiv &= \{[q] \mid q \in Q\} & \delta_\equiv([q],a) &= [\delta(q,a)] \\ {q^{\mathsf{init}}}_\equiv &= [q^{\mathsf{init}}] & F_\equiv &= \{[q] \mid q \in F\} \end{array}$$

◇

Bsp. 2.7: Der Äquivalenzklassenautomat $\mathcal{A}_\equiv$ zum DEA aus Bsp. 2.3 hat das folgende Zustandsdiagramm.



Satz 2.4: Der Äquivalenzklassenautomat ist wohldefiniert und $L(\mathcal{A}_\equiv) = L(\mathcal{A})$.

Beweis:

1. Wohldefiniert: zu zeigen $\delta_\equiv([q], a) = [\delta(q, a)]$ ist nicht abhängig von der Wahl des Repräsentanten $q \in [q]$. Das folgt direkt aus Lemma 2.3.

2. $L(\mathcal{A}) = L(\mathcal{A}_\equiv)$: Zunächst zeigen wir via Induktion über die Länge von w dass für alle $w \in \Sigma^*$ und alle $q \in Q$ die folgende Äquivalenz. $\tilde{\delta}(q, w) \in F \Leftrightarrow \tilde{\delta}_\equiv([q], w) \in F_\equiv$

   I.A. ($n = 0$, also $w = \varepsilon$): $\tilde{\delta}(q, \varepsilon) = q \in F \Leftrightarrow \tilde{\delta}_\equiv([q], \varepsilon) = [q] \in F_\equiv$
   I.S.: ($n \rightsquigarrow n + 1$)

$$\begin{aligned}
\tilde{\delta}(q, aw_\equiv) \in F &\iff \tilde{\delta}(\delta(q, a), w_\equiv) \in F \\
&\overset{I.V.}{\iff} \tilde{\delta}_\equiv([\delta(q, a)], w_\equiv) \in F_\equiv \\
&\iff \tilde{\delta}_\equiv(\delta_\equiv([q], a), w_\equiv) \in F_\equiv \\
&\iff \tilde{\delta}_\equiv([q], aw_\equiv) \in F_\equiv
\end{aligned}$$

   Mir Hilfe dessen zeigen wir nun $L(\mathcal{A}) = L(\mathcal{A}_\equiv)$. Sei $w \in \Sigma^*$

$$\begin{aligned}
w \in L(\mathcal{A}) &\iff \tilde{\delta}(q^{\mathsf{init}}, w) \in F \\
&\iff \tilde{\delta}_\equiv([q^{\mathsf{init}}], w) \in F_\equiv \\
&\iff w \in L(\mathcal{A}_\equiv)
\end{aligned}$$

□

In den Übungen werden wir ein Verfahren mit $O(|Q|^4 \cdot |\Sigma| \log |Q|)$ Laufzeit zur Konstruktion des Äquivalenzklassenautomat kennenlernen. Es gibt aber auch schnellere Verfahren. Z.B. mit dem Algorithmus von Hopcroft kann $\mathcal{A}_\equiv$ in $O(|Q||\Sigma| \log |Q|)$ erzeugt werden.

Wir werden später ($\rightarrow$ Satz von Myhill-Nerode) sehen dass $\mathcal{A}_\equiv$ der kleinste DEA ist, der $L(\mathcal{A})$ akzeptiert.

Def. 2.8: Eine Äquivalenzrelation $R \subseteq \Sigma^* \times \Sigma^*$ heißt *rechtskongruent*, falls Vorlesung: 27.10.17

$$(u, v) \in R \quad \Rightarrow \quad \forall w \in \Sigma^* \ (u \cdot w, v \cdot w) \in R$$

◇

Bsp. 2.8: Für einen DEA $\mathcal{A}$ definiere

$$R_\mathcal{A} = \{(u, v) \mid \tilde{\delta}(q^{\mathsf{init}}, u) = \tilde{\delta}(q^{\mathsf{init}}, v)\}.$$

Beobachtung 1 $R_\mathcal{A}$ ist Äquivalenzrelation.

Folgt daraus dass “=” eine Äquivalenzrelation ist.

Beobachtung 2 $R_{\mathcal{A}}$ ist rechtskongruent.

Beweis: In den Übungen

Beobachtung 3 Wir haben pro Zustand der von q^{init} erreichbar ist genau eine Äquivalenzklasse. Index von $R_{\mathcal{A}}$ ist also die Anzahl der erreichbaren Zustände.

Für den DEA aus Bsp. 2.3 hat $R_{\mathcal{A}}$ die folgenden Äquivalenzklassen.

$$\begin{aligned}
[\varepsilon] &= \{0^n \mid n \in \mathbb{N}\}\\
[1] &= \{0^n 1 0^m \mid n, m \in \mathbb{N}\}\\
[11] &= \{w \mid \text{ Anzahl von Einsen in } w \text{ ist gerade und } \geq 2\}\\
[111] &= \{w \mid \text{ Anzahl von Einsen in } w \text{ ist ungerade und } \geq 2\}
\end{aligned}$$

Def. 2.9: Für eine Sprache $L \subseteq \Sigma^*$ ist die *Nerode Relation* wie folgt definiert.

$$R_L = \{(u, v) \mid \forall w \in \Sigma^* \; uw \in L \Leftrightarrow vw \in L\}$$

◇

Beobachtung 1 Die Nerode Relation ist eine Äquivalenzrelation.

Folgt daraus dass "⇔" (Biimplikation, "Genau dann wenn") eine Äquivalenzrelation ist.

Beobachtung 2 Die Nerode Relation ist rechtskongruent.

Beweis: Sei $(u, v) \in R_L$. Zeige $\forall w \in \Sigma^*$, dass $(uw, vw) \in R_L$. Wir führen diesen Beweis via Induktion über die Länge von w.
I.A. $(n = 0)$ Für $w = \varepsilon$ ist $(u\varepsilon, v\varepsilon) = (u, v) \in R_L$.
I.S. $(n \rightsquigarrow n + 1)$ Betrachte mit $w = w'a$ ein beliebiges Wort der Länge n. Nach Induktionsvoraussetzung ist dann auch $(uw', vw') \in R_L$.

$$\begin{aligned}
(uw', vw') \in R_L \quad & \overset{\text{def } R_L}{\text{gdw}} \quad \forall z \in \Sigma^*, \quad uw'z \in L \Leftrightarrow vw'z \in L\\
& \overset{\text{zerlege } z=az'}{\text{impliziert}} \quad \forall a \in \Sigma, z' \in \Sigma^* : uw'az' \in L \Leftrightarrow vw'az' \in L\\
& \overset{\text{def } R_L}{\text{gdw}} \quad (uw'a, vw'a) \in R_L
\end{aligned}$$

□

Bsp. 2.9: Sei $\Sigma = \{0, 1\}$. Die Sprache $L = \{w \in \Sigma^* \mid \text{vorleztes Zeichen ist } 1\}$ hat die folgenden Äquivalenzklassen bezüglich der Nerode Relation.

$$\begin{aligned}
[\varepsilon] &= \{w \mid w \text{ endet mit } 00\} \cup \{\varepsilon, 0\} \\
[1] &= \{w \mid w \text{ endet mit } 01\} \cup \{1\} \\
[10] &= \{w \mid w \text{ endet mit } 10\} \\
[11] &= \{w \mid w \text{ endet mit } 11\}
\end{aligned}$$

Bsp. 2.10: Für ein beliebiges Alphabet Σ gilt:

1. Die Sprache $L = \{\varepsilon\}$ hat genau zwei Äquivalenzklassen bezüglich der Nerode Relation. Eine Äquivalenzklasse ist $\{\varepsilon\}$ die andere ist Σ^+.
2. Die Sprache $L = \{\}$ hat genau eine Äquivalenzklassen (nämlich Σ^*) bezüglich der Nerode Relation.

Bsp. 2.11: Sei $\Sigma = \{0, 1\}$. Die Sprache $L_{\text{centered}} = \{0^n 1 0^n \mid n \in \mathbb{N}\}$ hat bezüglich der Nerode Relation die folgende Menge von Äquivalenzklassen:

$$\{[w'] \mid w' \text{ ist Prefix eines Wortes } w \in L_{\text{centered}}\} \cup \{[11]\}$$

Dabei gilt, dass für je zwei verschiedene $k \in \mathbb{N}$ auch die Äquivalenzklassen $[0^k 1]$ verschieden sind. Somit gibt es unendlich viele Äquivalenzklassen.

Bemerkung: Die Äquivalenzklasse [11] enthält alle Wörter die kein Präfix eines Wortes aus L_{centered} sind.

Satz 2.5 (Myhill und Nerode)**:** Die folgende Aussagen sind äquivalent:

1. $L \subseteq \Sigma^*$ wird von DEA akzeptiert.
2. L ist Vereinigung von Äquivalenzklassen einer rechtskongruenten Äquivalenzrelation mit *endlichem* Index.
3. Die Nerode Relation R_L hat *endlichen* Index

Beweis: Wir beweisen die paarweise Äquivalenz in drei Schritten:

$$(1) \Rightarrow (2), \quad (2) \Rightarrow (3) \quad \text{und} \quad (3) \Rightarrow (1)$$

(1) ⇒ (2) : Sei $\mathcal{A}$ ein DEA mit

$$L(\mathcal{A}) = \{w \mid \tilde{\delta}(q^{\mathsf{init}}, w) \in F\} = \bigcup_{q \in F} \{w \mid \tilde{\delta}(q^{\mathsf{init}}, w) = q\}$$

Nun sind $\{w \mid \tilde{\delta}(q^{\mathsf{init}}, w) = q\} = [q]_{\mathcal{A}}$ genau die Äquivalenzklassen der Relation $R_{\mathcal{A}}$ aus Bsp 2.8, einer rechtskongruenten Äquivalenzrelation. Der Index ist die Anzahl der erreichbaren Zustände und somit endlich: $\text{Index}(R_{\mathcal{A}}) \leq |Q| < \infty$.

(2) ⇒ (3) Sei R rechtskongruente Äquivalenzrelation mit endlichem Index, so dass L Vereinigung von R-Äquivalenzklassen

Es genügt zu zeigen, dass die Nerode Relation R_L eine Obermenge von R ist.[9]

$$\begin{aligned} (u,v) \in R \quad &\Rightarrow \quad u \in L \Leftrightarrow v \in L, \quad \text{da L Vereinigung von Äquivalenzklassen ist} \\ &\Rightarrow \quad \forall w \in \Sigma^* \; uw \in L \Leftrightarrow vw \in L, \quad \text{da } R \text{ rechtskongruent} \\ &\Rightarrow \quad (u,v) \in R_L, \quad \text{nach Definition der Nerode Relation} \end{aligned}$$

Es gilt also $R \subseteq R_L$ und somit $\text{Index}(R_L) \leq \text{Index}(R) < \infty$.

(3) ⇒ (1) Gegeben R_L, konstruiere $\mathcal{A} = (Q, \Sigma, \delta, q^{\mathsf{init}}, F)$

- $Q = \{[w]_{R_L} \mid w \in \Sigma^*\}$ endlich, weil $\mathit{index}(R_L)$ endlich
- $\delta([w], a) = [wa]$ wohldefiniert, da R_L rechtskongruent
- $q^{\mathsf{init}} = [\varepsilon]$
- $F = \{[w] \mid w \in L\}$

Wir wollen nun $L(\mathcal{A}) = L$ zeigen. Dafür beweisen wir zunächst via Induktion über die Länge von w die folgende Eigenschaft.

$$\forall w \in \Sigma^* : \forall v \in \Sigma^* : \tilde{\delta}([v], w) = [v \cdot w]$$

IA $(w = \varepsilon)$: $\tilde{\delta}([v], \varepsilon) = [v] = [v \cdot \varepsilon]$

IS Sei $w = aw'$ beliebiges Wort der Länge $n + 1$.

$$\begin{aligned} \tilde{\delta}([v], aw') &= \tilde{\delta}(\delta([v], a), w') \\ &= \tilde{\delta}([v \cdot a], w') \\ &\overset{\text{I.V.}}{=} [va \cdot w'] \\ &= [v \cdot \underbrace{aw'}_{=w}] \end{aligned}$$

[9] Zur Erklärung: Falls $R \subseteq R_L$, dann $\text{Index}(R) \geq \text{Index}(R_L)$. Intuitiv: Je mehr Elemente eine Äquivalenzrelation R enthält, desto mehr Elemente sind bzgl. dieser Relation äquivalent, d.h. desto weniger unterschiedliche Klassen gibt es.

Nun zeigen wir $L(\mathcal{A}) = L$ wie folgt:

$$\begin{aligned} w \in L(\mathcal{A}) \quad &\text{gdw} \quad \tilde{\delta}([\varepsilon], w) \in F \\ &\text{gdw} \quad [w] \in F, \quad \text{(via Induktion gezeigte Eigenschaft für } v = \varepsilon) \\ &\text{gdw} \quad w \in L \end{aligned}$$

□

Vorlesung: 3.11.16

Korollar 2.5: Der im Beweisschritt (3) $\Rightarrow$ (1) konstruierte Automat $\mathcal{A}$ ist minimaler Automat für eine reguläre Sprache L. ◇

Beweis: Sei $\mathcal{A}'$ *beliebiger* DEA mit $L(\mathcal{A}') = L$.

Aus "1 $\Rightarrow$ 2" wissen wir, dass $\text{index}(R_{\mathcal{A}'}) \leq |Q'|$ gilt.

Aus "2 $\Rightarrow$ 3" wissen wir, dass $R_{\mathcal{A}'} \subseteq R_L$ und somit $\text{index}(R_L) \leq \text{index}(R_{\mathcal{A}'})$ gilt.

In "3 $\Rightarrow$ 1" definieren wir A sodass $|Q| = \text{index}(R_L) \leq \text{index}(R_{\mathcal{A}}) \leq |Q'|$.

Für beliebigen DEA A' ist $|Q|$ also nie größer als $|Q'|$. □

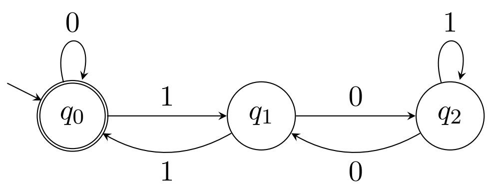
2.3 Pumping Lemma (PL) für reguläre Sprachen

Welche interessanten Eigenschaften haben reguläre Sprachen?

Notation: Sei $\text{bin} : \{0,1\}^* \to \mathbb{N}$ die Dekodierung von Bitstings in natürliche Zahlen.

Z.B. $\text{bin}(101) = 5$, $\text{bin}(\varepsilon) = 0$

Bsp.: Betrachete den folgenden DEA, der die Sprache der Binärcodierungen von durch drei Teilbaren Zahlen akzeptiert: $L = \{w \in \{0,1\}^* \mid \text{bin}(w) \equiv_3 0\}$



Beobachtungen:

- Es gilt offensichtlich, dass $11 \in L$
- Es gilt auch, dass $1\underline{00}1 \in L$.
- Der Automat hat eine Schleife bei $\tilde{\delta}(q_1, 00) = q_1$, die mehrfach „abgelaufen" werden kann ohne die Akzeptanz zu beinflussen.

- Also gilt auch $100001 \in L$,
- und im Allgemeinen $\forall i \in \mathbb{N} : 1(00)^i 1 \in L$

Verdacht: Alle "langen" Wörter lassen sich in der Mitte "aufpumpen". Wir formalisieren diesen Verdacht im folgenden Lemma.

Lemma 2.6 (Pumping Lemma)**:** Sei L eine reguläre Sprache. Dann gilt:

$$\begin{aligned} &\exists n \in \mathbb{N},\ n > 0 : \quad \forall z \in L,\ |z| \geq n : \\ &\exists u, v, w \in \Sigma^* : \\ &z = uvw,\ |uv| \leq n,\ |v| \geq 1 \\ \text{und } &\forall i \in \mathbb{N} :\ uv^i w \in L \end{aligned}$$

Beweis: Sei $\mathcal{A} = (\Sigma, Q, \delta, q^{\mathsf{init}}, F)$ ein beliebiger DEA für L. Wähle $n = |Q|$ und $z \in L$ beliebig mit $|z| \geq n$.

Beim Lesen von z durchläuft $\mathcal{A}$ genau $\overbrace{|z| + 1}^{\geq n+1}$ Zustände und somit gibt es mindestens einen Zustand $q \in Q$ der mehrmals besucht wird (Schubfachprinzip).

Wähle das q, dessen zweiter Besuch zuerst passiert.

$$\begin{array}{rrll} \text{Nun gilt:} & \exists u : & \tilde{\delta}(q^{\mathsf{init}}, u) = q & u \text{ Präfix von } z \\ & \exists v : & \tilde{\delta}(q, v) = q & uv \text{ Präfix von } z \\ & \exists w : & \tilde{\delta}(q, w) \in F & uvw = z \\ & & |v| \geq 1 & \\ & & |uv| \leq n & \text{da } q \text{ zwei mal besucht} \end{array}$$

$$\begin{aligned} \text{Es folgt für beliebiges } i \in \mathbb{N} : \quad \tilde{\delta}(q^{\mathsf{init}}, uv^i w) &= \tilde{\delta}(q, v^i w) \\ &= \tilde{\delta}(q, w) \qquad \text{denn } \forall i : \tilde{\delta}(q, v^i) = q \\ &\in F \end{aligned}$$ □

Bsp.: Die Sprache $L_{\text{centered}} = \{0^n 1 0^n \mid n \in \mathbb{N}\}$ ist nicht regulär.

Wir geben hierfür einen Widerspruchsbeweis mit Hilfe des Pumping Lemma PL.

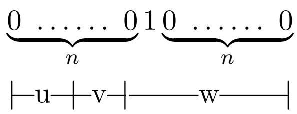
Sei n die Konstante aus dem PL. Wähle $z = 0^n 1 0^n$. (Gültige Wahl, da $|z| = 2n + 1 \geq n$) Laut PL existieren u, v, w, sodass $z = uvw$ mit $|v| \geq 1, |uv| \leq n$ und $\forall i \in \mathbb{N}\ uv^i w \in L$. Nach Wahl von z gilt nun

- $uv = 0^m$ mit $m \leq n$

- $v = 0^k$ mit $k \geq 1$
- $w = 0^{n-m}10^n$

Betrachte $uv^2w = 0^{m-k}0^{2k}0^{n-m}10^n = 0^{n+k}10^n \notin L$. Widerspruch! Somit ist L nicht regulär.

Zur Illustration:



2.4 Nichtdeterministischer endlicher Automat (NEA)

Aufgabe: Konstruiere für eine natürliche Zahl $n \in \mathbb{N}$ einen DEA für die folgende Sprache.

$$L_n = \{w \in \{0,1\}^* \mid \text{das } n\text{-letzte Symbol von } w \text{ ist } 1\}$$

Naiver Lösungsversuch:

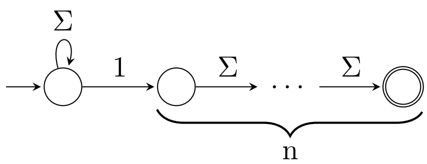


Abb. 2: Nichtdet. Automat für L_n

Problem: Diagramm beschreibt keinen DEA. Startzustand hat zwei ausgehende Kanten für 1.

Untersuche Sprache mit Hilfe von Nerode Relation. Beobachtung: Je zwei Wörter der Länge n sind in unterschiedlichen Äquivalenzklassen. Es gibt also mindestens 2^n Äquivalenzklassen und aus Korollar 2.5 wissen wir dass ein minimaler DEA, der L_n akzeptiert 2^n Zustände hat.

Idee: Definiere einen neue Art von Automaten bei dem ein Zustand pro Zeichen mehrere Nachfolger haben darf.

Def. 2.10 (NEA)**:** Ein *nichtdeterministischer endlicher Automat (NEA)*, (NFA $\hat{=}$ nichtdeterministischer endlicher Automat) ist ein 5-Tupel

$$\mathcal{N} = (\Sigma, Q, \delta, q^{\mathsf{init}}, F)$$

dabei ist

- Σ ein Alphabet,
- Q eine *endliche* Menge deren Elemente wir *Zustände* nennen,
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ eine Funktion die wir *Transitionsfunktion* nennen,
- $q^{\mathsf{init}} \in Q$ ein Zustand den wir *Startzustand* nennen und
- $F \subseteq Q$ eine Teilmenge der Zustände deren Elemente wir *akzeptierende* Zustände nennen.

$\diamond$

Bemerkung: Die Definition des NEA unterscheidet sich vom DEA also nur in der Transitionsfunktion. Beim DEA ist der Bildbereich der Transitionsfunktion die Menge der Zustände Q. Hier ist der Bildbereich die Potenzmenge der Zustandsmenge $\mathcal{P}(Q)$. Analog zu DEAs werden wir auch NEAs mit Hilfe eines Zustandsdiagramms beschrieben. Z.B. beschreibt Abb. 2 für jedes $n \in \mathbb{N}$ einen NEA für die Sprache L_n.

Im folgenden sei $\mathcal{N}$ immer ein NEA.

Def. 2.11: Wir nennen eine Folge von Zuständen $q_0 q_1 \ldots q_n$ einen *Lauf von $\mathcal{N}$ über $w = a_1 \ldots a_n$* wenn $q_i \in \delta(q_{i-1}, a_i)$ für alle i mit $1 \leq i \leq n$. Wir nennen einen Lauf *initial* falls $q_0 = q^{\mathsf{init}}$. Wir nennen einen Lauf *akzeptierend*, falls $q_n \in F$. $\diamond$

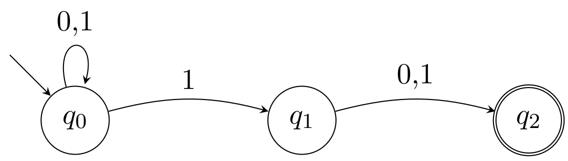
Vorlesung: 8.11.17

Def. 2.12: Ein Wort $w \in \Sigma^*$ wir von $\mathcal{N}$ *akzeptiert*, falls $\mathcal{N}$ einen initialen und akzeptierenden Lauf über w hat. Die von $\mathcal{N}$ akzeptierte Sprache ist die Menge der von $\mathcal{N}$ akzeptierten Wörter. D.h. $L(\mathcal{N}) = \{w \in \Sigma^* \mid \exists \text{ initialer, akzeptierender Lauf von } \mathcal{N} \text{ über } w\}$ $\diamond$

Bsp. 2.12: Der NEA für die Sprache

$$L_2 = \{w \in \{0,1\}^* \mid \text{ das zweitletzte Zeichen von } w \text{ ist } 1 \}$$

hat die folgende graphische Repräsentation.



Bemerkung: Die Frage ob ein gegebenes Wort w akzeptiert wird (das “Wortproblem”) lässt sich für NEAs nicht mehr so leicht beantworten wie wir es von DEAs gewohnt sind. Ein sinvolles Vorgehen scheint jeden initialen Lauf zu betrachten, doch z.B. für das Wort 111 hat obiger NEA bereits drei verschiedene initiale Läufe: $q_0q_0q_0$, $q_0q_0q_1$, $q_0q_0q_2$.

Bemerkung: Die Definitionen von NEA und DEA in der Literatur sind nicht einheitlich. Es gibt äquivalente NEA Definitionen die statt der Transitionsfunktion $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ eine Transitionsrelation $\delta \subseteq Q \times \Sigma \times Q$ verwenden. Es gibt alternative NEA Definitionen die eine Menge von Startzuständen erlauben. Alternativ könnte man auch zunächst den NEA einführen und den DEA als Spezialfall dessen definieren (Spezialfall: Bild von Transitionsfunktion ist einelementig für alle $q \in Q$ und $a \in \Sigma$)

Bemerkung: Zu jedem DEA $\mathcal{A} = (\Sigma, Q, \delta, q^{\mathsf{init}}, F)$ gibt es einen NEA der die gleiche Sprache akzeptiert. Z.B. der NEA $\mathcal{N} = (\Sigma, Q, \delta_{\mathsf{NEA}}, q^{\mathsf{init}}, F)$, mit $\delta_{\mathsf{NEA}}(q, a) = \{\delta(q, a)\}$ der sich von $\mathcal{A}$ nur in der Transitionsfunktion unterscheidet.

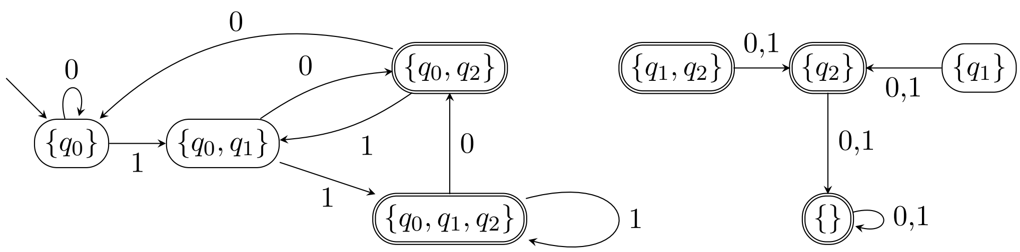
Satz 2.7 (Rabin und Scott)**:** Zu jedem NEA $\mathcal{N}$ mit n Zuständen gibt es einen DEA $\mathcal{A}_{\mathcal{P}}$ mit 2^n Zuständen, so dass $L(\mathcal{A}_{\mathcal{P}}) = L(\mathcal{N})$.

Zur Vorbereitung des Beweises machen wir zunächst die folgende Definition.

Def. 2.13 (Potenzmengenautomat)**:** Für gegebenen NEA $\mathcal{N} = (\Sigma, Q, \delta, q^{\mathsf{init}}, F)$ ist der Potenzmengenautomat $\mathcal{A}_{\mathcal{P}}$ wie folgt definiert

$$\begin{aligned} Q_{\mathcal{P}} &= \mathcal{P}(Q) \\ \delta_{\mathcal{P}}(p, a) &= \bigcup_{q \in p} \delta(q, a) \\ {q^{\mathsf{init}}}_{\mathcal{P}} &= \{q^{\mathsf{init}}\} \\ F_{\mathcal{P}} &= \{p \in Q_{\mathcal{P}} \mid p \cap F \neq \varnothing\} \end{aligned}$$ ◇

Bsp. 2.13: Der Potzenzmengenautomat für den NEA aus Bsp. 2.12 hat das folgende Zutandsdiagramm.



Die vier Zustände auf der rechten Seite sind nicht erreichbar.

Beweis (von Satz 2.7): Zeige $L(\mathcal{A}_\mathcal{P}) = L(\mathcal{N})$. Dafür beweisen wir zunächst via Induktion über die Länge von w, die folgende Eigenschaft:

$\forall w \in \Sigma^* \; \forall p \in Q_\mathcal{P} \backslash \{\{\}\} \; \forall q \in Q$

$$q \in \tilde{\delta}_\mathcal{P}(p, w) \Leftrightarrow \exists \underbrace{q_0, q_1, \ldots q_n}_{\text{Lauf}} \in Q, \text{ sodass } n = |w|, q_0 \in p \text{ und } q_n = q$$

I.A. ($n = 0$, also $w = \varepsilon$): Gilt trivialerweise da $p \neq \{\}$
I.S. ($n \rightsquigarrow n+1$): Seit $w = aw'$ beliebiges Wort der Länge $n+1$.

$$\begin{aligned}
q \in \tilde{\delta}_\mathcal{P}(p, w) \quad &\Leftrightarrow \quad q \in \tilde{\delta}_\mathcal{P}(\delta_\mathcal{P}(p, a), w') \\
&\overset{IV}{\Leftrightarrow} \quad \exists \underbrace{q_1 q_2 \ldots q_{n+1}}_{\text{Lauf}} \in Q, \text{ sodass } n = |w'|, q_1 \in \delta_\mathcal{P}(p, a) \text{ und } q_{n+1} = q \\
&\Leftrightarrow \quad \exists \underbrace{q_1 q_2 \ldots q_{n+1}}_{\text{Lauf}} \in Q, \text{ sodass } n = |w'|, \exists q_0 \in p, q_1 \in \delta(q_0, a) \text{ und } q_{n+1} = q \\
&\Leftrightarrow \quad \exists \underbrace{q_0 q_1 q_2 \ldots q_{n+1}}_{\text{Lauf}} \in Q, \text{ sodass } n+1 = |w|, q_0 \in p \text{ und } q_{n+1} = q
\end{aligned}$$

Mit Hilfe dieser Eigenschaft zeigen wir nun die Gleichheit $L(\mathcal{A}_\mathcal{P}) = L(\mathcal{N})$

$$\begin{aligned}
w \in L(\mathcal{A}_\mathcal{P}) &\Leftrightarrow \tilde{\delta}_\mathcal{P}({q^{\mathsf{init}}}_\mathcal{P}, w) \in F_\mathcal{P} \\
&\Leftrightarrow \exists q_f \in \tilde{\delta}_\mathcal{P}({q^{\mathsf{init}}}_\mathcal{P}, w) \cap F \\
&\Leftrightarrow \exists \underbrace{q_0, q_1, \ldots q_n}_{\text{Lauf}} \in Q, \text{ sodass } n = |w|, q_0 \in q^{\mathsf{init}} \text{ und } q_n \in F \\
&\Leftrightarrow \exists \text{ initialer, akzeptierender Lauf von } \mathcal{N} \text{ über } w \\
&\Leftrightarrow w \in L(\mathcal{N})
\end{aligned}$$

□

Bemerkung: Es gelten also die folgenden Äquivalenzen.

$$L \text{ regulär} \quad \overset{Def.\ 2.3}{\Longleftrightarrow} \quad L = L(\mathcal{A}) \text{ für einen DEA } \mathcal{A} \quad \Longleftrightarrow \quad L = L(\mathcal{N}) \text{ für einen NEA } \mathcal{N}$$

Bemerkung: NEAs sind eine exponentiell kompaktere Repräsentation von regulären Sprachen im folgenden Sinne:

1. Es gibt mit L_n (n-letztes Zeichen) eine Menge von Sprachen die sich mit Hilfe eines $n+1$ Zustands NEA darstellen lassen, aber bei denen ein minimaler DEA mindestens 2^n Zustände hat. (Siehe Übungsblatt 3, Aufgabe 2)

2. Zu jedem NEA mit n Zuständen gibt es einen DEA mit 2^n Zuständen der die gleiche Sprache akzeptiert. (Satz 2.7)
3. Zu jedem DEA mit n Zuständen gibt es einen NEA mit n Zuständen der die gleiche Sprache akzeptiert.

2.4.1 ε-Übergänge

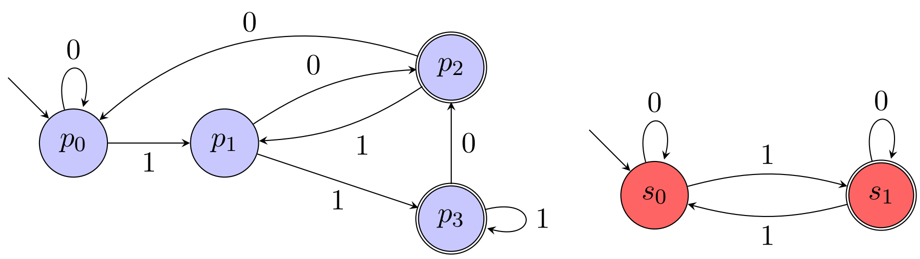
In diesem Unterkapitel führen wir mit dem ε-NEA ein weiteres Automatenmodell ein. Wir wollen ε-NEAs zunächst durch die folgende Fragestellung und anschließende Diskussion motivieren.

Frage: Gegeben zwei reguläre Sprachen L_1, L_2 ist dann auch die Konkatenation $L_1 \cdot L_2$ eine reguläre Sprache?

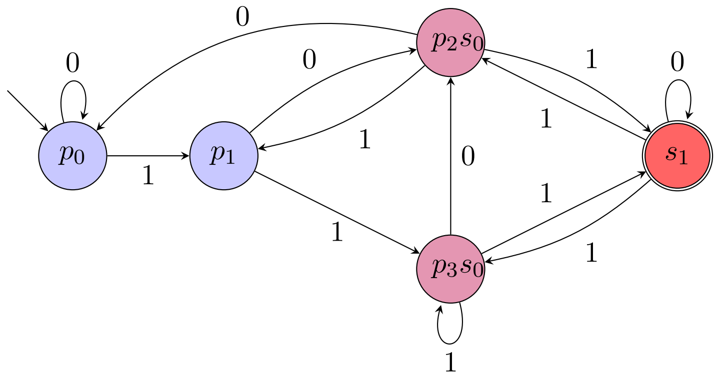
Idee: Geben DEA $\mathcal{A}_1$ mit $L(\mathcal{A}_1) = L_1$ und DEA $\mathcal{A}_2$ mit $L(\mathcal{A}_2) = L_2$, konstruiere NEA für $L_1 \cdot L_2$ durch "Hintereinanderschalten" von $\mathcal{A}_1$ und $\mathcal{A}_2$; immer wenn wir in einem akzeptierenden Zustand von $\mathcal{A}_1$ sind erlauben wir in $\mathcal{A}_2$ zu "wechseln".

Erste, naive (und inkorrekte) Umsetzung dieser Idee: Verschmelze akzeptierende Zustände von $\mathcal{A}_1$ mit dem Startzustand von $\mathcal{A}_2$ Wir betrachten die folgenden Automaten um zu sehen, dass diese Umsetzung nicht zielführend ist.

Bsp. 2.14: Links: DEA $\mathcal{A}_1$, der Automat aus Bsp. 2.12 eingeschränkt auf die erreichbaren Zustände. Rechts: DEA $\mathcal{A}_2$ dessen Sprache $\{w \in \{0,1\}^* \mid \text{Anzahl } 1 \text{ in } w \text{ ist ungerade}\}$ ist.



Unten: NEA $\mathcal{N}_{\mathsf{naiv}}$, Resultat einer einer naiven und inkorrekten Konstuktion für die Konkatenation.



Dieser NEA akzeptiert nun auch das Wort $w = 11011$. Allerdings ist w nicht in der Konkatenation $L(\mathcal{A}_1) \cdot L(\mathcal{A}_2)$ denn es gibt keine Zerlegung $w = w_1 \cdot w_2$ bei der sowohl der Prefix w_1 von $\mathcal{A}_1$ als auch der Suffix w_2 von $\mathcal{A}_2$ akzeptiert wird.

Das "Verschmelzen" von p_2 (bzw. p_3) mit s_0 war also keine gute Idee. Was uns aber helfen würde wäre ein Zustandsübergang der uns erlaubt von Zustand p_2 (bzw. p_3) in den Zustand s_0 zu gehen ohne dabei ein Zeichens zu Lesen.

Wir nennen solch einen Zustandsübergang ε-Übergang und definieren einen Automaten der solche Zustandsübergänge haben kann wie folgt.

Def. 2.14 (ε-NEA)**:** Ein *nichtdeterministischer endlicher Automat mit ε-Übergängen* ist ein 5-Tupel

$$\mathcal{B} = (\Sigma, Q, \delta, q^{\mathsf{init}}, F)$$

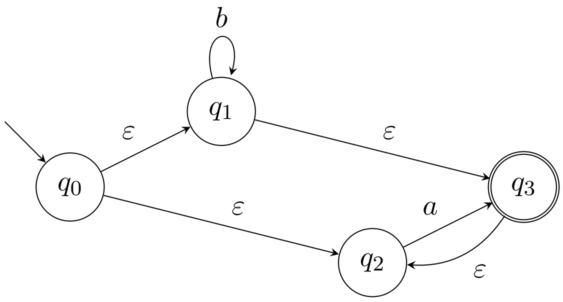
wobei Σ, Q, q^{init}, F wie bei NEAs (bzw. DEAs) definiert sind und die Transitionsfunkion den folgenden Typ hat.

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$$

◇

Bsp. 2.15: Zustandsdiagramm eines ε-NEA über dem Alphabet $\Sigma = \{a, b\}$.

Vorlesung: 10.11.17



Wie bei den bisher definierten Automaten wollen wir mit Hilfe eines ε-NEAs eine Sprache definieren. Wir benötigen dafür zunächst zwei weitere Definitionen.

Der ε-Abschluss ist eine Abbildung die jedem Zustand q die Menge der Zustände zuordnet, die von q über ε-Übergänge erreichbar sind. Wir definieren diese Abbildung formal wie folgt. Dabei verwenden wir den Abbildungsnamen ecl um an den englischen Begriff "ε closure" zu erinnern.

Def. 2.15: Der *ε-Abschluss* $\mathsf{ecl} : Q \to \mathcal{P}(Q)$ ist die kleinste Abbildung, die für alle $q, q', q'' \in Q$ die folgenden Eigenschaften erfüllt:

$$q \in \mathsf{ecl}(q)$$
$$q' \in \mathsf{ecl}(q) \text{ und } q'' \in \delta(q', \varepsilon) \quad \Rightarrow \quad q'' \in \mathsf{ecl}(q) \qquad \diamond$$

Offensichtlich kann immer eine endliche explizite Repräsentation von ecl berechnet werden: Starte in jedem Zustand einmal und folge mit Breitensuche allen ε-Kanten im Zustandsdiagramm.

Für den ε-NEA aus Bsp. 2.15 sieht ecl wie folgt aus:

q	q_0	q_1	q_2	q_3
$\mathsf{ecl}(q)$	$\{q_0, q_1, q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_2, q_3\}$	$\{q_2, q_3\}$

Als nächstes definieren wir eine dreistellige Relation die uns für je zwei Zustände sagt für welche Wörter den Automaten vom ersten Zustand in den zweiten Zustand überführen. Der Name der Relation "reach" soll dabei an des englische Wort "reachability" erinnern.

Def. 2.16: Die *Erreichbarkeitsrelation* $\mathsf{reach} \subseteq Q \times \Sigma^* \times Q$ ist die kleinste Relation, die für alle $q, q', q'', q''' \in Q$ und für alle $w \in \Sigma^*$ die folgenden Eigenschaften erfüllt:

$$q' \in \mathsf{ecl}(q) \quad \Rightarrow \quad (q, \varepsilon, q') \in \mathsf{reach}$$
$$q' \in \mathsf{ecl}(q), q'' \in \delta(q', a) \text{ und } (q'', w, q''') \in \mathsf{reach} \Rightarrow (q, aw, q''') \in \mathsf{reach} \qquad \diamond$$

Für den ε-NEA aus Bsp. 2.15 sieht reach wie folgt aus:

reach	q_0	q_1	q_2	q_3
q_0	$\{\varepsilon\}$	$\{b\}^*$	$\{b\}^* \cdot \{a\}^*$	$\{b\}^* \cdot \{a\}^*$
q_1	$\{\}$	$\{b\}^*$	$\{b\}^* \cdot \{a\}^*$	$\{b\}^* \cdot \{a\}^*$
q_2	$\{\}$	$\{\}$	$\{a\}^*$	$\{a\} \cdot \{a\}^*$
q_3	$\{\}$	$\{\}$	$\{a\}^*$	$\{a\} \cdot \{a\}^* \cup \{\varepsilon\}$

Def. 2.17: Ein Wort $w \in \Sigma^*$ wird von $\mathcal{B}$ *akzeptiert*, wenn $(q^{\mathsf{init}}, w, q_f) \in \mathsf{reach}$ für ein $q_f \in F$. Die von $\mathcal{B}$ akzeptierte Sprache $L(\mathcal{B})$ ist die Menge der von $\mathcal{B}$ akzeptierten Wörter. D.h. $L(\mathcal{B}) = \{w \in \Sigma^* \mid \exists q \in F : (q^{\mathsf{init}}, w, q) \in \mathsf{reach}\}$ ◇

Offensichtlich gibt es zu jedem NEA $\mathcal{N}$ einen ε-NEA $\mathcal{B}$ der die gleiche Sprache akzeptiert. Die Konstuktion ist dabei sehr einfach: Erweitere die Transitionsfunkion um $\delta(q, \varepsilon) = \{\}$ für alle $q \in Q$. Für die Sprachgleichheit zeigen wir via Induktion über die Länge von w, dass für alle Wörter und für alle Zustände

$$\exists \text{ Lauf } q_0 q_1 \ldots q_n \text{ von } \mathcal{N} \text{ über } w \quad \Leftrightarrow \quad (q_0, w, q_n) \in \mathsf{reach}$$

Der folgende Satz zeigt uns dass auch die umgekehrte Richtung gilt.

Satz 2.8: Zu jedem ε-NEA $\mathcal{B}$ gibt es einen NEA $\mathcal{N}$, so dass $L(\mathcal{N}) = L(\mathcal{B})$.

Zur Vorbereitung des Beweises machen wir zunächst die folgende Definition.

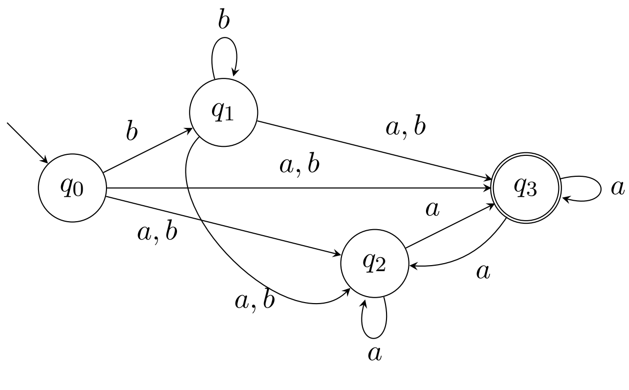
Def. 2.18 (ε-freier Automat)**:** Für gegebenen *Eps*-NEA $\mathcal{B} = (\Sigma, Q, \delta, q^{\mathsf{init}}, F)$ definieren wir den NEA $\mathcal{N} = (\Sigma, Q, \delta_{\mathcal{N}}, q^{\mathsf{init}}, F_{\mathcal{N}})$ mit

$$\delta_{\mathcal{N}}(q, a) = \bigcup_{q' \in \mathsf{ecl}} \{p' \mid \exists p : p' \in \mathsf{ecl}(p) \text{ und } p \in \delta(q', a)\}$$

$$F_{\mathcal{N}} = \bigcup_{q_f \in F} \mathsf{ecl}(q_f)$$

und nennen diesen den ε-freie Automat von $\mathcal{B}$. ◇

Bsp. 2.16: Der ε-freie Automat für den ε-NEA aus Bsp. 2.15 hat das folgende Zustandsdiagramm.



Beweis (ε-Eliminierung): Zeige $L(\mathcal{N}) = L(\mathcal{B})$. Dabei verwenden wir die folgende Eigenschaft, die wir via Induktion über die Länge von w in den Übungen zeigen werden.

$\forall w \in \Sigma^* \ \forall p \in Q_{\mathcal{P}} \backslash \{\} \ \forall q \in Q$

$$(q, w, q') \in \mathsf{reach} \Leftrightarrow \exists \underbrace{q_0, q_1, \ldots q_n}_{\text{Lauf}} \in Q, \text{ sodass } n = |w|, q_0 \in p \text{ und } q_n = q$$

$$\begin{aligned} w \in L(\mathcal{A}_{\mathcal{P}}) &\Leftrightarrow \tilde{\delta}_{\mathcal{P}}({q^{\mathsf{init}}}_{\mathcal{P}}, w) \in F_{\mathcal{P}} \\ &\Leftrightarrow \exists q_f \in \tilde{\delta}_{\mathcal{P}}({q^{\mathsf{init}}}_{\mathcal{P}}, w) \cap F \\ &\Leftrightarrow \exists \underbrace{q_0, q_1, \ldots q_n}_{\text{Lauf}} \in Q, \text{ sodass } n = |w|, q_0 \in q^{\mathsf{init}} \text{ und } q_n = q_f \\ &\Leftrightarrow \exists \text{ initialer, akzeptierender Lauf von } \mathcal{N} \text{ über } w \\ &\Leftrightarrow w \in L(\mathcal{N}) \end{aligned}$$

□

Mit Hilfe der ε-NEAs greifen wir nun die zu Beginn von Abschnitt 2.4.1 aufgeworfene Fragestellung wieder auf und zeigen dass für je zwei reguläre Sprachen auch die Konkatenation regulär ist.

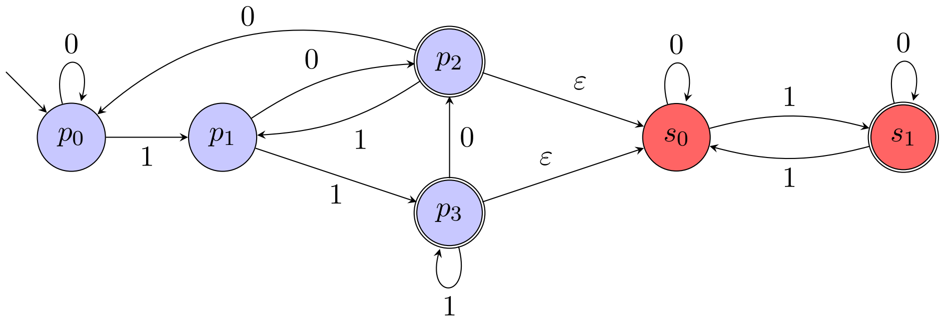
Wir geben hierfür zunächst eine Konstruktion an.

Def. 2.19: Gegeben zwei ε-NEA $\mathcal{B}_i = (\Sigma, Q_i, \delta_i, {q^{\mathsf{init}}}_i, F_i)$ definieren wir den *ε-NEA für Konkatenation* $\mathcal{B} = (\Sigma, Q, \delta, q^{\mathsf{init}}, F)$ wie folgt.

$$\begin{aligned} Q &= Q_1 \mathbin{\dot\cup} Q_2 \\ \delta(q, a) &= \begin{cases} \delta_1(q, a) & q \in Q_1 \wedge (q \notin F_1 \vee a \neq \varepsilon) \\ \delta_1(q, a) \cup \{{q^{\mathsf{init}}}_2\} & q \in F_1 \wedge a = \varepsilon \\ \delta_2(q, a) & q \in Q_2 \end{cases} \\ q^{\mathsf{init}} &= {q^{\mathsf{init}}}_1 \\ F &= F_2 \end{aligned}$$

◇

Bsp. 2.17: Der ε-NEA für Konkatenation für die beiden Automaten aus Bsp. 2.14 hat das folgende Zustandsdiagramm.



Lemma 2.9: Die vom ε-NEA für Konkatenation akzeptierte Sprache ist $L(\mathcal{B}_1) \cdot L(\mathcal{B}_2)$

BEWEIS: Zeige via Induktion über die Länge von w_1 dass $\forall w_1, w_2 \in \Sigma^* \forall q_1 \in Q_1, \forall q_1' \in F, \forall q_2' \in Q_2$ die folgende Eigenschaft gilt.

$$(q_1, w_1, q_1') \in \mathsf{reach} \text{ und } (q^{\mathsf{init}}, w_2, q_2') \in \mathsf{reach} \quad \Leftrightarrow \quad (q_1, w, q_2') \in \mathsf{reach}$$

□

2.5 Abschlusseigenschaften

Def. 2.20: Eine Menge X heißt *abgeschlossen* unter Operation $f : X^n \to X$ falls $\forall x_1, \ldots, x_n \in X : f(x_1, \ldots, x_n) \in X$. ◇

Z.B. sind die natürlichen Zahlen abgeschlossen unter Addition, aber nicht abgeschlossen unter Subtraktion.

Lemma 2.10: Die Menge *REG* der regulären Sprachen ist abgeschlossen unter Komplement.

BEWEIS: □

Lemma 2.11: Die Menge *REG* der regulären Sprache ist abgeschlossen unter dem Stern Operator

Beweis:

$$
\begin{aligned}
Q &= Q_1 \mathbin{\dot\cup} \{q_0\} \\
\delta(q,a) &= \begin{cases} \delta_1(q,a) & q \in Q_1 \setminus F_1 \\ \delta_1(q,a) \cup \delta_1(q_{01},a) & q \in F_1 \\ \delta_1(q_{01},a) & q = q_0 \end{cases} \\
F &= \{q_0\} \cup F_1 \\
\ldots\ L(\mathcal{A}) &= L(\mathcal{A}_1)^*
\end{aligned}
$$

□

Liste der Definitionen

Liste der Sätze

Abbildungsverzeichnis

Abkürzungsverzeichnis

AL Aussagenlogik

CFL Menge der kontextfreien Sprachen

CFG Menge der kontextfreien Grammatiken

CNF Chomsky Normalform

CP Korrespondenzproblem

CYK Cocke, Younger, Kasami

DAG gerichteter azyklischer Graph

DCFG deterministische CFG

DCFL deterministische CFL

DEA deterministischer endlicher Automat

DFA engl.: deterministic finite automaton

DPDA deterministischer Kellerautomat

DTM deterministische TM

EA endlicher Automat

LBA Linear Bounded Automaton

MPCP Das modifizierte PCP

ND Nicht-Determinismus

NEA nichtdeterministischer endlicher Automat

NFA engl.: nondeterministic finite automaton

NPDA nichtdeterministischer Kellerautomat

NT Nichtterminal

NTM Eine nichtdeterministische TM

PCP Das Postsche Korrespondenzproblem

PDA pushdown automaton (Kellerautomat)

PL Pumping Lemma

RE Menge der regulären Ausdrücke

REG Menge der regulären Sprachen

RM Registermaschine

TM Turing-Maschine

TT Turingtabelle

Anmerkungsverzeichnis