

2.3 Reguläre Ausdrücke

In Satz 2.23 auf Seite 44 haben wir gesehen, dass die Klasse der regulären Sprachen unter allen gängigen Verknüpfungsoperatoren (Vereinigung, Durchschnitt, Komplement, Konkatenation und Kleeneabschluss) abgeschlossen ist. Satz 2.23 kann dahingehend verschärft werden, dass die Klasse der regulären Sprachen über Σ die kleinste Klasse ist, die unter den Operationen Vereinigung, Konkatenation und Kleeneabschluss abgeschlossen ist und welche die Sprachen \emptyset , $\{\varepsilon\}$ und $\{a\}$, $a \in \Sigma$ enthält. Um diese Aussage zu belegen, betrachten wir einen weiteren Formalismus, der sehr intuitive Schreibweisen für reguläre Sprachen ermöglicht.

Definition 2.25 (Syntax regulärer Ausdrücke). Sei Σ ein Alphabet. Die Menge der regulären Ausdrücke über Σ ist durch folgende induktive Definition gegeben.

1. \emptyset und ε sind reguläre Ausdrücke.
2. Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck.
3. Mit α und β sind auch $(\alpha\beta)$, $(\alpha + \beta)$ und (α^*) reguläre Ausdrücke.
4. Nichts sonst ist ein regulärer Ausdruck.

Die Klammern werden oftmals weggelassen, wobei der Verknüpfungsoperator $+$ die schwächste Priorität hat und der Sternoperator am stärksten bindet. Z.B. steht $a\varepsilon + bc^*$ für $((a\varepsilon) + (b(c^*)))$.⁴ ■

Anstelle der oben angegebenen induktiven Definition verwendet man auch häufig eine saloppe BNF-ähnliche Kurzschreibweise (*abstrakte Syntax* genannt), die die Option für das Setzen von Klammern als Selbstverständlichkeit unterstellt und α , β als Metasymbole für reguläre Ausdrücke und a für die Elemente des zugrundeliegenden Alphabets verwendet.

$$\alpha ::= \emptyset \mid \varepsilon \mid a \mid \alpha\beta \mid \alpha + \beta \mid \alpha^*$$

Reguläre Ausdrücke stehen für Sprachen. Intuitiv sind ε und a Kurzschreibweisen für die jeweils einelementigen Sprachen $\{\varepsilon\}$ und $\{a\}$. Der Ausdruck $\alpha\beta$ steht für die Sprache, die sich durch Konkatenation der Sprachen für α und β ergibt. Manchmal verwenden wir hierfür auch das Konkatenationssymbol \circ und schreiben $\alpha \circ \beta$ statt $\alpha\beta$. (In der Literatur wird manchmal auch ein Strichpunkt “;” anstelle von \circ als Symbol für die Konkatenation verwendet.) Das Symbol $+$ steht für die Vereinigung, der Sternoperator $*$ für den Kleeneabschluss. Der Plusoperator ist durch $\alpha^+ = \alpha^*\alpha$ definiert. Anstelle von $\alpha + \beta$ wird in der Literatur auch oftmals $\alpha|\beta$ geschrieben.

Die *Länge* eines regulären Ausdrucks α wird mit $|\alpha|$ bezeichnet. Sie ist definiert als die Länge von α aufgefasst als Wort über dem Alphabet $\Sigma \cup \{+, *, (,), \varepsilon, \emptyset\}$. Für unsere Zwecke wird nur die asymptotische Länge eine Rolle spielen. Diese ist durch die Anzahl an Operatoren (Vereinigung $+$, Konkatenation \circ und Kleeneabschluss $*$) in α gegeben. So enthält z.B. der reguläre Ausdruck $a\varepsilon + bc^*$ insgesamt vier Operatoren (zwei Konkatenationen und je einmal “+” und “*”).

⁴Ähnlich wie in Beispiel 1.10 auf Seite 14 kann anstelle der induktiven Definition eine kontextfreie Grammatik für die Syntax vollständig oder sparsam geklammerter regulärer Ausdrücke angegeben werden.

Definition 2.26 (Semantik regulärer Ausdrücke). Die zu einem regulären Ausdruck α gehörende Sprache $\mathcal{L}(\alpha)$ ist wie folgt definiert.

$$\begin{array}{ll} \mathcal{L}(\emptyset) = \emptyset & \mathcal{L}(\varepsilon) = \{\varepsilon\} \\ \mathcal{L}(a) = \{a\} & \mathcal{L}(\alpha\beta) = \mathcal{L}(\alpha)\mathcal{L}(\beta) \\ \mathcal{L}(\alpha + \beta) = \mathcal{L}(\alpha) \cup \mathcal{L}(\beta) & \mathcal{L}(\alpha^*) = \mathcal{L}(\alpha)^* \end{array}$$

Zwei reguläre Ausdrücke α_1, α_2 heißen *äquivalent* (i. Z. $\alpha_1 \equiv \alpha_2$), wenn $\mathcal{L}(\alpha_1) = \mathcal{L}(\alpha_2)$. Wir nennen α nichtleer, wenn $\alpha \neq \emptyset$; also wenn $\mathcal{L}(\alpha) \neq \emptyset$. ■

Alle Rechengesetze für formale Sprachen und den Vereinigungs-, Konkatenations- und Sternoperator (siehe Seite 7) übertragen sich auf reguläre Ausdrücke. So gilt z.B. das Assoziativgesetz $(\alpha + \beta) + \gamma \equiv \alpha + (\beta + \gamma)$ und Kommutativgesetz $\alpha + \beta \equiv \beta + \alpha$ sowie die beiden Distributivgesetze:

$$\alpha(\beta + \gamma) \equiv \alpha\beta + \alpha\gamma \qquad (\beta + \gamma)\alpha \equiv \beta\alpha + \gamma\alpha$$

Im Allgemeinen gilt jedoch $(\alpha + \beta)^* \neq \alpha^* + \beta^*$, da z. B. das Wort ab in der zu $(a + b)^*$ gehörenden Sprache liegt, jedoch aber nicht in der zu $a^* + b^*$ gehörenden Sprache. Weitere Beispiele sind die Neutralität von ε für den Konkatenationsoperator $\varepsilon\alpha \equiv \alpha\varepsilon \equiv \alpha$ oder $\emptyset\alpha \equiv \alpha\emptyset \equiv \emptyset$.

Reguläre Ausdrücke bilden neben regulären Grammatiken und den bereits besprochenen Varianten endlicher Automaten einen weiteren Formalismus für reguläre Sprachen. Wir werden sehen, dass die Klasse der regulären Sprachen genau mit der Klasse der Sprachen $\mathcal{L}(\alpha)$ für einen regulären Ausdruck α übereinstimmt. Diese Aussage weisen wir nach, indem wir spracherhaltende Transformationen von regulären Ausdrücken zu endlichen Automaten und umgekehrt angeben.

Wir geben zwei Verfahren an, wie man zu gegebenem regulären Ausdruck α einen ε -NFA konstruieren kann, der genau die Sprache $\mathcal{L}(\alpha)$ akzeptiert. Diese Verfahren liefern den Beweis für folgendes Lemma.

Lemma 2.27 (Regulärer Ausdruck \rightsquigarrow ε -NFA). *Zu jedem regulären Ausdruck α gibt es einen ε -NFA \mathcal{M} mit $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{M})$.*

1. Verfahren “Regulärer Ausdruck \rightsquigarrow NFA”. Das erste Verfahren beruht auf einem kompositionellen Ansatz, der die Operatoren Vereinigung (+), Konkatenation und Kleeneabschluss regulärer Ausdrücke durch die entsprechenden Operatoren auf ε -NFA realisiert. Für $\alpha = \emptyset, \varepsilon$ oder $a \in \Sigma$ geben wir explizit einen NFA \mathcal{M}_α mit $\mathcal{L}(\mathcal{M}_\alpha) = \mathcal{L}(\alpha)$ an.

- Für $\alpha = \emptyset$ können wir einen beliebigen NFA mit leerer Endzustandsmenge verwenden, etwa bestehend aus einem Anfangszustand, der keine Transitionen hat und nicht final ist. (Alternativ könnte man sogar den NFA mit leerer Zustandsmenge verwenden.)
- Für $\alpha = \varepsilon$ verwenden wir einen NFA, der aus einem Zustand q_0 besteht. Dieser ist zugleich Anfangs- und Endzustand und hat keine ausgehenden Transitionen.

- Ist $\alpha = a \in \Sigma$, so verwenden wir einen NFA \mathcal{M}_a mit zwei Zuständen q_0, q_1 und der Transition

$$q_0 \xrightarrow{a} q_1.$$

Es gibt keine weiteren Transitionen in \mathcal{M}_a . Zustand q_0 wird als Anfangszustand, q_1 als Endzustand deklariert.

Ist α von der Form $\beta\gamma$ oder $\beta + \gamma$ oder β^* , dann konstruieren wir zuerst (rekursiv) ε -NFA \mathcal{M}_β bzw. \mathcal{M}_γ für die Teilausdrücke β und γ und wenden dann den entsprechenden Operator für ε -NFA an. Siehe Abschnitt 2.2 auf Seite 35 ff.

Die Größe des resultierenden ε -NFA \mathcal{M}_α gemessen an der Anzahl an Zuständen ist offenbar linear in der Länge des gegebenen regulären Ausdrucks α . Beachte, dass die NFA für atomare reguläre Ausdrücke \emptyset , ε oder $a \in \Sigma$ höchstens zwei Zustände haben und dass die Operatoren Konkatenation $\mathcal{M}_1 \circ \mathcal{M}_2$ und Vereinigung $\mathcal{M}_1 \uplus \mathcal{M}_2$ einen ε -NFA generieren, dessen Anzahl an Zuständen gleich $|\mathcal{M}_1| + |\mathcal{M}_2|$ ist. Für den Kleeneabschluss gilt $|\mathcal{M}^*| = |\mathcal{M}| + 1$. Durch strukturelle Induktion kann dann gezeigt werden, dass $|\mathcal{M}_\alpha| \leq |\alpha| + 1$.

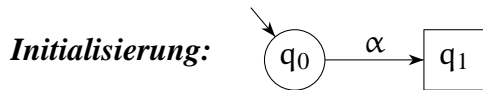
2. Verfahren “Regulärer Ausdruck \rightsquigarrow NFA”. Eine weitere Konstruktionsmöglichkeit für die Konstruktion eines ε -NFA besteht darin, den Automaten schrittweise durch sukzessives Einfügen von Zuständen und Kanten aufzubauen. Den Spezialfall $\alpha \equiv \emptyset$ können wir ausklammern, da die akzeptierte Sprache eines NFA ohne Endzustände (d.h. mit leerer Endzustandsmenge) offenbar leer ist. Im Folgenden setzen wir also $\alpha \neq \emptyset$ voraus. Beginnend mit einem gerichteten Graphen bestehend aus zwei Zuständen, die über eine mit α beschriftete Kante miteinander verbunden sind, fügen wir sukzessive neue Zustände ein. Die Transitionen sind zunächst mit regulären Ausdrücken beschriftet. Diese werden sukzessive durch echte Teilausdrücke ersetzt bis alle Kanten mit ε oder einem Terminalzeichen $a \in \Sigma$ beschriftet sind. Die präzise Vorgehensweise ist in Abbildung 19 angegeben. Der resultierende Graph ist ein ε -NFA, der genau die Sprache $\mathcal{L}(\alpha)$ akzeptiert. Wie für das erste Verfahren kann auch hier garantiert werden, dass die Größe des konstruierten NFA linear in der Länge des gegebenen regulären Ausdrucks ist.

Die bisherigen Ergebnisse belegen noch nicht, dass jede reguläre Sprache durch einen regulären Ausdruck beschrieben werden kann. Wir zeigen nun, dass zu jedem endlichen Automaten \mathcal{M} ein regulärer Ausdruck α konstruiert werden kann, so dass $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{M})$.

Lemma 2.28 (NFA \rightsquigarrow regulärer Ausdruck). *Zu jedem NFA \mathcal{M} gibt es einen regulären Ausdruck α mit $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{M})$.*

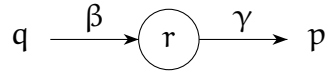
Auch hierzu diskutieren wir zwei Verfahren. Das erste Verfahren beruht auf einem Gleichungssystem für reguläre Ausdrücke, welche die Sprachen $L_q = \{x \in \Sigma^* : \delta(q, x) \cap F \neq \emptyset\}$ repräsentieren, und ist unter dem Stichwort “Ersetzungsmethode” bekannt. Das zweite Verfahren beruht auf der Methodik des dynamischen Programmierens zur Berechnung regulärer Ausdrücke für die Sprachen $L_{q,p}$ aller Wörter x , so dass $\delta(q, x) = p$.

Ungeachtet, welche Methode eingesetzt wird, können zunächst einige Vereinfachungen an dem gegebenen NFA vorgenommen werden, die die akzeptierte Sprache unverändert lassen, aber den Automaten verkleinern. Z.B. können alle Zustände q entfernt werden, von denen kein Endzustand erreichbar ist. Eine derartige Vereinfachung des NFA lässt sich mit einer Tiefen- oder Breiten-Rückwärtssuche, gestartet mit den F-Zuständen, realisieren. Mit Rückwärtssuche

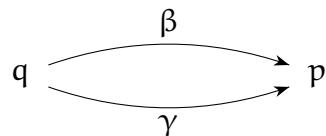


Iteration: solange möglich, wende eine der drei Regeln an:

Regel 1: generiere neuen Zustand r und ersetze $q \xrightarrow{\beta\gamma} p$ durch



Regel 2: ersetze $q \xrightarrow{\beta+\gamma} p$ durch



Regel 3: generiere neuen Zustand r und ersetze $q \xrightarrow{\beta^*} p$ durch

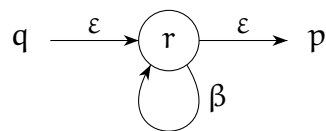


Figure 19: Zweites Verfahren “regulärer Ausdruck \rightsquigarrow ϵ -NFA”

ist hier gemeint, dass mit den Endzuständen beginnend nach einem Anfangszustand entlang der Kanten (p, q) , falls $p \in \bigcup_{a \in \Sigma} \delta(q, a)$, gesucht wird. Ebenso können alle unerreichbaren Zustände (also Zustände, die von keinem Anfangszustand erreichbar sind) eliminiert werden. Im Folgenden können wir also davon ausgehen, dass die Endzustandsmenge F von jedem Zustand erreichbar ist und dass alle Zustände in Q erreichbar sind.

1. Verfahren “NFA \rightsquigarrow regulärer Ausdruck”: Ersetzungsmethode. Im Folgenden sei $\mathcal{M} = (Q, \Sigma, \delta, Q_0, F)$ ein NFA, von dem vorausgesetzt wird, dass alle Zustände von wenigstens einem Anfangszustand $q_0 \in Q_0$ erreichbar sind und dass die Endzustandsmenge F von jedem Zustand erreichbar ist. Das Ziel ist die Konstruktion eines regulären Ausdrucks für die durch \mathcal{M} akzeptierte Sprache $\mathcal{L}(\mathcal{M}) \subseteq \Sigma^*$. Die Ersetzungsmethode beruht auf der Idee, jedem Zustand q einen regulären Ausdruck α_q zuzuordnen, welcher für die Sprache

$$L_q \stackrel{\text{def}}{=} \{w \in \Sigma^* : \delta(q, w) \cap F \neq \emptyset\}$$

aller Wörter $w \in \Sigma^*$ steht, die einen in Zustand q beginnenden, akzeptierenden Lauf im Automaten haben. Das Ziel ist also die Konstruktion regulärer Ausdrücke α_q für alle Zustände $q \in Q$, so dass $\mathcal{L}(\alpha_q) = L_q$. Wegen $\mathcal{L}(\mathcal{M}) = \bigcup_{q \in Q_0} L_q$ erhält man dann einen gesuchten regulären Ausdruck für \mathcal{M} , indem man die Summe der regulären Ausdrücke α_q für alle Anfangszustände betrachtet. Ist also etwa $Q_0 = \{q_1, \dots, q_k\}$, so ist

$$\alpha = \alpha_{q_1} + \dots + \alpha_{q_k}$$

ein regulärer Ausdruck mit $\mathcal{L}(\alpha) = \mathcal{L}(\mathcal{M})$.

Zunächst kann man ein Gleichungssystem für die α_q 's hinschreiben, welches sich aus der Übergangsrelation δ ergibt. Im Folgenden gehen wir von einer beliebigen, aber festen Nummerierung der Zustände aus, etwa $Q = \{q_1, \dots, q_n\}$, und schreiben kurz α_i für α_{q_i} .

Das Gleichungssystem für die regulären Ausdrücke ist nun wie folgt:

- Ist $q_i \notin F$, so ist

$$\alpha_i \equiv \sum_{a \in \Sigma} \sum_{r \in \delta(q_i, a)} a \alpha_r$$

- Ist q ein Endzustand, so ist der oben angegebene Ausdruck um “ $+\varepsilon$ ” zu erweitern. D.h., für $q_i \in F$ ist

$$\alpha_i \equiv \sum_{a \in \Sigma} \sum_{r \in \delta(q_i, a)} a \alpha_r + \varepsilon$$

Dabei ist $\sum_{a \in \Sigma} \sum_{r \in \delta(q_i, a)} a \alpha_r = \emptyset$, falls $\delta(q_i, a) = \emptyset$ für alle $a \in \Sigma$.

Das Gleichungssystem für die α_i 's kann nun gelöst werden, indem man sukzessive folgende *Ersetzungsregel* einsetzt:

“Aus $\alpha \equiv \beta\alpha + \gamma$ und $\varepsilon \notin \mathcal{L}(\beta)$ folgt $\alpha \equiv \beta^*\gamma$.”

Zusätzlich benötigt man einige Äquivalenzregeln für reguläre Ausdrücke, etwa die beiden Äquivalenzgesetze

$$\beta_1\gamma + \dots + \beta_k\gamma \equiv (\beta_1 + \dots + \beta_k)\gamma \quad \text{und} \quad \gamma\beta_1 + \dots + \gamma\beta_k \equiv \gamma(\beta_1 + \dots + \beta_k)$$

und einige Äquivalenzgesetze für ε und \emptyset (wie z.B. $\beta + \emptyset \equiv \beta$ oder $\beta\emptyset \equiv \emptyset$). Man beachte den Sonderfall $\gamma \equiv \emptyset$, für den sich die angegebene Regel durch

“Aus $\alpha \equiv \beta\alpha$ und $\varepsilon \notin \mathcal{L}(\beta)$ folgt $\alpha \equiv \emptyset$ ”

ersetzen lässt, da $\beta^*\emptyset \equiv \emptyset$. Für den Sonderfall $\gamma = \varepsilon$ ergibt sich

“Aus $\alpha \equiv \beta\alpha$ und $\varepsilon \notin \mathcal{L}(\beta)$ folgt $\alpha \equiv \beta^*$ ”

Zunächst zur Korrektheit der angegebenen Regel. Hierzu formulieren wir die Regel auf Ebene formaler Sprachen:

Lemma 2.29 (Korrektheit der Ersetzungsregel). *Seien $L, K, H \subseteq \Sigma^*$, so dass $L = KL \cup H$ und $\varepsilon \notin K$. Dann gilt $L = K^*H$.*

Proof. “ \subseteq ”: Durch Induktion nach n zeigen wir, dass

$$\{w \in L : |w| \leq n\} \subseteq K^*H.$$

Induktionsanfang $n = 0$: falls $\varepsilon \in L = KL \cup H$, so ist $\varepsilon \in H$ (da $\varepsilon \notin K$) und somit $\varepsilon \in K^*H$.

Induktionsschritt $n - 1 \implies n$ ($n \geq 1$): Sei $w \in L$ ein Wort der Länge n . Wegen $L = KL \cup H$ gilt $w \in KL$ oder $w \in H$. Falls $w \in H$, so ist $w = \varepsilon w \in K^*H$. Wir nehmen nun an, dass $w \in KL$, etwa

$$w = xy, \text{ wobei } x \in K \text{ und } y \in L.$$

Da $\varepsilon \notin K$, ist $x \neq \varepsilon$ und somit

$$|y| = |w| - |x| \leq n - 1.$$

Nach Induktionsvoraussetzung gilt $y \in K^*H$. Wegen $x \in K$ ist $w = xy \in K^*H$.

“ \supseteq ”: sei $w \in K^*H$, etwa $w = x_1 \dots x_m y$, wobei $m \geq 0$, $x_1, \dots, x_m \in K$ und $y \in H$. Wegen $L = KL \cup H$ gilt $H \subseteq L$ und $KL \subseteq L$. Wir erhalten:

$$\begin{aligned} & y \in L \quad (\text{da } y \in H) \\ \implies & x_m y \in L \quad (\text{da } x_m \in K \text{ und } y \in L) \\ \implies & x_{m-1} x_m y \in L \quad (\text{da } x_{m-1} \in K \text{ und } x_m y \in L) \\ & \vdots \\ \implies & x_1 \dots x_{m-1} x_m y \in L \quad (\text{da } x_1 \in K \text{ und } x_2 \dots x_{m-1} x_m y \in L) \end{aligned}$$

und somit $w = x_1 \dots x_{m-1} x_m y \in L$. □