

Reguläre und Kontextfreie Sprachen 1

Grundzüge digitaler Systeme

Vortrag von: Gernot Salzer

Reguläre und Kontextfreie Sprachen

- Reguläre Sprachen

- Operationen auf formalen Sprachen

- Definition regulärer Sprachen

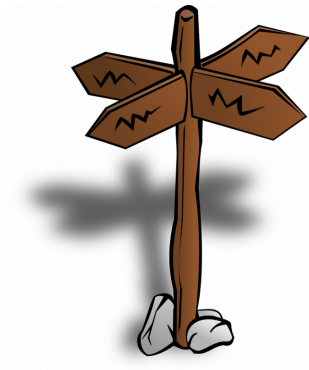
- Reguläre Ausdrücke

- Eigenschaften regulärer Sprachen

- Vom regulären Ausdruck zum Automaten

- Vom Automaten zum regulären Ausdruck

- Kontextfreie Grammatiken



Operationen auf formalen Sprachen

- Σ Alphabet, d.h., endliche, nicht-leere Menge atomarer Symbole
 w Wort über Σ , (endliche) Folge von Zeichen aus dem Alphabet Σ
 ε Leerwort
 Σ^* Menge aller endlichen Wörter über Σ (inklusive Leerwort)
 $w \cdot w' = ww'$ Verkettung der Wörter $w, w' \in \Sigma^*$

Seien $L, L' \subseteq \Sigma^*$ zwei Sprachen.

$$L \cup L' = \{ w \mid w \in L \text{ oder } w \in L' \} \quad \text{Vereinigung}$$

$$L \cdot L' = \{ w \cdot w' \mid w \in L, w' \in L' \} \quad \text{Verkettung}$$

$$\begin{aligned} L^0 &= \{\varepsilon\} \\ L^{n+1} &= L \cdot L^n \quad (n \geq 0) \end{aligned} \quad \text{Potenzen}$$

$$\begin{aligned} L^+ &= \bigcup_{n \geq 1} L^n \\ L^* &= \bigcup_{n \geq 0} L^n = L^0 \cup L^+ = \{\varepsilon\} \cup L^+ \quad \text{Kleene-Stern} \end{aligned}$$

Verkettung

$$\{a, b\} \cdot \{b, c, d\} = \{ab, ac, ad, bb, bc, bd\}$$

$$\{b, c, d\} \cdot \{a, b\} = \{ba, bb, ca, cb, da, db\}$$

$$\begin{aligned} (\{a, b\} \cdot \{1, 2\}) \cdot \{\#, \$\} &= \{a1\#, a1\$, a2\#, a2\$, b1\#, b1\$, b2\#, b2\$\} \\ &= \{a, b\} \cdot (\{1, 2\} \cdot \{\#, \$\}) \end{aligned}$$

$$\{a, b\} \cdot \{\varepsilon\} = \{a \cdot \varepsilon, b \cdot \varepsilon\} = \{a, b\} = \{\varepsilon \cdot a, \varepsilon \cdot b\} = \{\varepsilon\} \cdot \{a, b\}$$

$$\{a, b\} \cdot \{\} = \{\} \cdot \{a, b\} = \{\}$$

$$\{\varepsilon\} \cdot \{\varepsilon\} = \{\varepsilon\}$$

$$\{\} \cdot \{\} = \{\varepsilon\} \cdot \{\} = \{\} \cdot \{\varepsilon\} = \{\}$$

Beobachtungen:

- Sprachverkettung ist nicht kommutativ.
- Sprachverkettung ist assoziativ.
- $\{\varepsilon\}$ ist neutrales Element bzgl. Sprachverkettung.
- $\{\}$ ist Nullelement bzgl. Sprachverkettung.

Potenzen von $\{a, 42\}$

$$L = \{a, 42\}$$

$$L^0 = \{\varepsilon\}$$

$$L^1 = L \cdot L^0 = L \cdot \{\varepsilon\} = L = \{a, 42\}$$

$$L^2 = L \cdot L^1 = L \cdot L = \{aa, a42, 42a, 4242\}$$

$$L^3 = L \cdot L^2 = \{aaa, aa42, a42a, a4242, 42aa, 42a42, 4242a, 424242\}$$

\vdots

$$L^+ = \bigcup_{n \geq 1} L^n = L^1 \cup L^2 \cup L^3 \cup \dots$$

$$= \{a, 42, aa, a42, 42a, 4242, aaa, aa42, a42a, a4242, 42aa, \dots\}$$

$$L^* = \bigcup_{n \geq 0} L^n = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \dots$$

$$= \{\varepsilon, a, 42, aa, a42, 42a, 4242, aaa, aa42, a42a, a4242, 42aa, \dots\}$$

Potenzen eines Alphabets Σ

$$\Sigma^0 = \{\varepsilon\}$$

$$\Sigma^1 = \Sigma$$

$$\Sigma^n$$

alle Σ -Wörter der Länge n (d.h., mit n Symbolen)

$$\Sigma^+ = \bigcup_{n \geq 1} \Sigma^n$$

alle Σ -Wörter ohne Leerwort

$$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$$

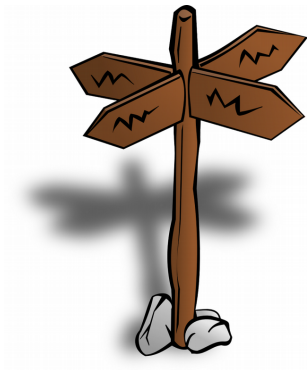
alle Σ -Wörter mit Leerwort

Reguläre und Kontextfreie Sprachen

■ Reguläre Sprachen

- Operationen auf formalen Sprachen
- Definition regulärer Sprachen
- Reguläre Ausdrücke
- Eigenschaften regulärer Sprachen
- Vom regulären Ausdruck zum Automaten
- Vom Automaten zum regulären Ausdruck

■ Kontextfreie Grammatiken



Reguläre Sprachen

Alle Sprachen, die aus einem Alphabet mit Hilfe von Vereinigung, Verkettung und Stern gebildet werden können.

Anwendungen:

- Betriebssystem-Shells: DOS („Wildcards“), UNIX-Shells (wie `sh`, `csch`, `ash`, `bash`, `zsh`), ...
- UNIX Kommandozeilenprogramme: `grep`, `awk`, `ed`, `sed`, ...
- Editoren: `vi`, `emacs`, ...
- Compilerbau: *Tokens* bilden reguläre Sprache, die durch sog. Scanner (Lexer) wie `lex` oder `flex` verarbeitet werden.
- Programmiersprachen: PERL, TCL, PHP, PYTHON, RUBY, R, JAVA, JAVASCRIPT, .NET-Sprachen, ...
- Websprachen: XML Schema, XQuery, XPath, DTDs, ...
- Datenbanken: MySQL, Oracle, PostgreSQL, ...
- ...

Regulären Sprachen über einem Alphabet

Die Menge der regulären Sprachen über Σ , $\mathcal{L}_{\text{reg}}(\Sigma)$, ist die kleinste Menge, sodass gilt:

- $\{\}$, $\{\epsilon\}$ und $\{s\}$ sind reguläre Sprachen (für alle $s \in \Sigma$).
- Wenn L und L' reguläre Sprachen sind, dann auch $L \cup L'$, $L \cdot L'$ und L^* .

Reellen Numerale: reguläre Sprache über $\Sigma = \{0, \dots, 9, ., E, +, -\}$

$real = digit \cdot digit^* \cdot \{.\} \cdot digit^* \cdot (\{\epsilon\} \cup scale)$

$scale = \{E\} \cdot \{+, -, \epsilon\} \cdot digit \cdot digit^*$

$digit = \{0, \dots, 9\} = \{0\} \cup \dots \cup \{9\}$

Wichtig: Unterscheide Symbole des Alphabets von Meta-Symbolen!

$0, \dots, 9, ., E, +, - \dots$ Symbole des Alphabets

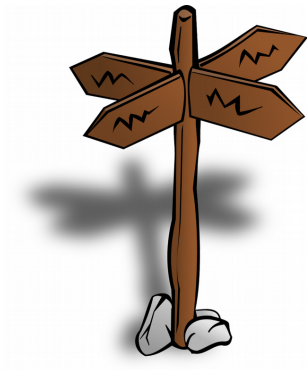
$\epsilon, real, scale, digit \dots$ Meta-Symbole, Abkürzungen

Reguläre und Kontextfreie Sprachen

■ Reguläre Sprachen

- Operationen auf formalen Sprachen
- Definition regulärer Sprachen
- Reguläre Ausdrücke
- Eigenschaften regulärer Sprachen
- Vom regulären Ausdruck zum Automaten
- Vom Automaten zum regulären Ausdruck

■ Kontextfreie Grammatiken



Reguläre Ausdrücke

Ausdrücke wie $digit \cdot digit^*$ und $digit^+$ sind ununterscheidbar: Beides sind semantische Beschreibungen der Menge aller Ziffernfolgen. Um Aussagen über die Form der Ausdrücke treffen zu können, benötigen wir eine formale Sprache.

Reguläre Ausdrücke (algebraische Notation)

Die regulären Ausdrücke über Σ sind die kleinste Menge, für die gilt:

- \emptyset , ε und s sind reguläre Ausdrücke (für alle Symbole $s \in \Sigma$).
- Sind X und Y reguläre Ausdrücke, dann auch $(X + Y)$, (XY) und X^* .

Vereinfachte Klammerung: $+$ bindet am schwächsten, $*$ am stärksten. Keine Klammern bei gleichartigen Operatoren (wegen Assoziativität).

Die Sprache $\mathcal{L}(X)$ zu einem regulären Ausdruck X ist definiert durch:

$$\mathcal{L}(\emptyset) = \{\}$$

$$\mathcal{L}(\varepsilon) = \{\varepsilon\}$$

$$\mathcal{L}(s) = \{s\} \quad \text{für } s \in \Sigma$$

$$\mathcal{L}(X + Y) = \mathcal{L}(X) \cup \mathcal{L}(Y)$$

$$\mathcal{L}(XY) = \mathcal{L}(X) \cdot \mathcal{L}(Y)$$

$$\mathcal{L}(X^*) = (\mathcal{L}(X))^*$$

Regulärer Ausdruck für die reellen Numerale

$$R = DD^* \cdot D^*(\varepsilon + S)$$

$$S = E(+ + - + \varepsilon)DD^*$$

$$D = 0 + 1 + \dots + 9$$

(R , S und D sind Abkürzungen für die jeweiligen regulären Ausdrücke.)

Die zugehörigen Sprachen:

$$\begin{aligned}\mathcal{L}(D) &= \mathcal{L}(0 + 1 + \dots + 9) \\ &= \mathcal{L}(0) \cup \mathcal{L}(1) \cup \dots \cup \mathcal{L}(9) \\ &= \{0\} \cup \{1\} \cup \dots \cup \{9\} \\ &= \textit{digit}\end{aligned}$$

$$\begin{aligned}\mathcal{L}(S) &= \mathcal{L}(E(+ + - + \varepsilon)DD^*) \\ &= \mathcal{L}(E) \cdot \mathcal{L}(+ + - + \varepsilon) \cdot \mathcal{L}(D) \cdot \mathcal{L}(D^*) \\ &= \{E\} \cdot (\mathcal{L}(+) \cup \mathcal{L}(-) \cup \mathcal{L}(\varepsilon)) \cdot \textit{digit} \cdot \mathcal{L}(D)^* \\ &= \{E\} \cdot (\{+\} \cup \{-\} \cup \{\varepsilon\}) \cdot \textit{digit} \cdot \textit{digit}^* \\ &= \textit{scale}\end{aligned}$$

$$\mathcal{L}(R) = \dots = \textit{real}$$

Zwei reguläre Ausdrücke X und Y heißen äquivalent, geschrieben $X = Y$, wenn $\mathcal{L}(X) = \mathcal{L}(Y)$ gilt.

$$((a + b)^* + \varepsilon)^* = (a + b)^*$$

$$\begin{aligned}\mathcal{L}(((a + b)^* + \varepsilon)^*) &= \dots \\ &= (\{a, b\}^* \cup \{\varepsilon\})^* \\ &= (\{a, b\}^*)^* \quad \text{da } \varepsilon \in L^* \text{ für alle } L \\ &= (\{a, b\}^*)^0 \cup (\{a, b\}^*)^1 \cup (\{a, b\}^*)^2 \cup \dots \\ &= \{a, b\}^* \cup (\{a, b\}^*)^0 \cup (\{a, b\}^*)^2 \cup \dots \\ &= \{a, b\}^* \quad \text{da } L^* \text{ alle Wörter über } L \text{ enthält} \\ &= \dots \\ &= \mathcal{L}((a + b)^*)\end{aligned}$$

Reguläre Ausdrücke in EBNF-Notation

EBNF ... Erweiterte Backus-Naur-Form (Formalismus zur Beschreibung der Syntax von Programmiersprachen, die reguläre Ausdrücke zulässt)



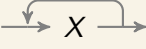
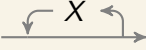

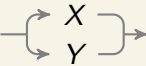
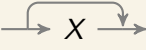
XY	$X \cdot Y$	Aufeinanderfolge
$X Y$	$X \cup Y$	Alternativen
$[X]$	$\{\epsilon\} \cup X$	Option
$\{X\}$	X^*	Wiederholung
(X)	(X)	Gruppierung
$"s"$	$\{s\}$	Symbol

Reelle Numerale in EBNF-Notation

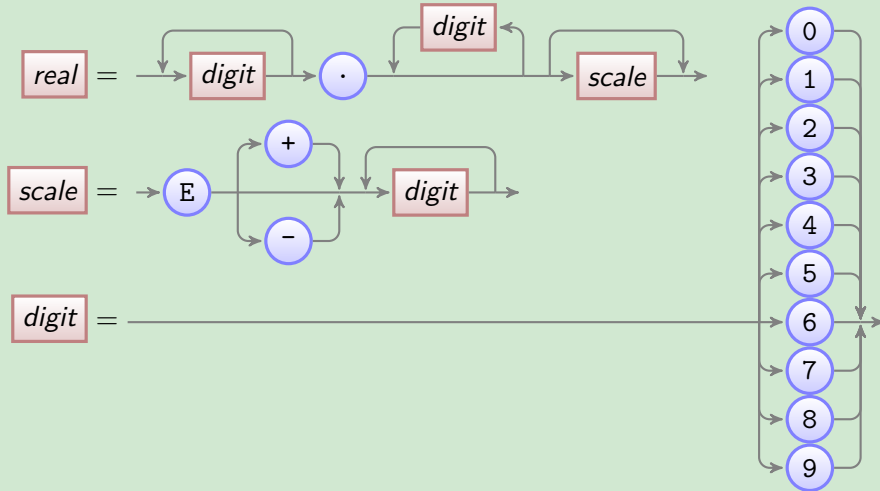
```
real = digit {digit} "." {digit} [scale]  
scale = "E" ["+" | "-"] digit {digit}  
digit = "0" | "1" | "2" | ... | "9"
```

Reguläre Ausdrücke als Syntaxdiagramme

Syntaxdiagramme ... graphische Form der EBNF

	A	Abkürzung
	$\{s\}$	Symbol
	X^+	Wiederholung ≥ 1
	X^*	Wiederholung ≥ 0
	$X \cdot Y$	Aufeinanderfolge
	$X \cup Y$	Alternativen
	$\{\varepsilon\} \cup X$	Option

Reelle Numerale als Syntaxdiagramm



Reguläre Ausdrücke im informatischen Alltag

UNIX := „30 definitions of regular expressions living under one roof“

(Donald E. Knuth)

```
grep -E -e "regex" file
```

liefert alle Zeilen der Datei *file*, die eine Zeichenkette enthalten, die dem regulären Ausdruck *regex* (in POSIX ERE Syntax) entspricht.

Verschiedene „Standards“:

- POSIX Basic Regular Expressions
- POSIX Extended Regular Expressions
- PERL Regular Expressions
- ...





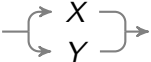
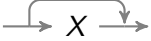
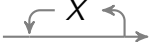
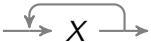
POSIX Extended Regular Expressions (ERE)

<i>regex</i>	trifft zu auf	<i>regex</i>	trifft zu auf
$\backslash s$	Zeichen s	XY	X gefolgt von Y
s	s , falls kein Sonderzeichen	$X Y$	X oder Y
$.$	alle Zeichen	X^*	≥ 0 Mal X
$^$	Zeilenanfang	X^+	≥ 1 Mal X
$\$$	Zeilenende	$X^?$	≤ 1 Mal X
$[s_1 \cdots s_n]$	ein Zeichen aus $\{s_1, \dots, s_n\}$	$X\{i\}$	i Mal X
$[^s_1 \cdots s_n]$	alle Zeichen außer s_1, \dots, s_n	$X\{i, \}$	$\geq i$ Mal X
(X)	X	$X\{i, j\}$	i bis j Mal X

Reelle Numerale als ERE

$digit = \{0, \dots, 9\}$ [0-9]
 $scale = \{E\} \cdot \{+, -, \varepsilon\} \cdot digit \cdot digit^*$ $E[+-]?[0-9]^+$
 $real = digit \cdot digit^* \cdot \{.\} \cdot digit^* \cdot (\{\varepsilon\} \cup scale)$ $\wedge [0-9]^+ \cdot \backslash . [0-9]^* (E[+-]?[0-9]^+)? \$$
 [0-9] ... Kurzform von [0123456789]; analog [a-zA-Z] für Buchstaben

Zusammenfassung der Notationen

Reg. Sprache	Algebra	EBNF	Syntaxdiagramm	POSIX ERE	
A	A	A			Abkürzung
$\{\}$	\emptyset				Leersprache
$\{\varepsilon\}$	ε				Leerwortsprache
$\{s\}$	s	"s"		$\backslash s$	Terminalsymbol
$X \cdot Y$	XY	XY		XY	Aufeinanderfolge
$X \cup Y$	$X + Y$	$X Y$		$X Y$	Alternativen
$\{\varepsilon\} \cup X$	$\varepsilon + X$	$[X]$		$X?$	Option
X^*	X^*	$\{X\}$		X^*	Wiederholung ≥ 0
X^+	XX^*, X^+	$X\{X\}$		X^+	Wiederholung ≥ 1
(X)	(X)	(X)		(X)	Gruppierung

Weitere POSIX-Notationen:

POSIX ERE	Bedeutung	Reguläre Sprache
s	s , falls kein Sonderzeichen	$\{s\}$
$.$	alle Zeichen	Σ
\wedge	Zeilenanfang	
$\$$	Zeilenende	
$[s_1 \cdots s_n]$	eines der Zeichen s_i	$\{s_1, \dots, s_n\}$
$[\wedge s_1 \cdots s_n]$	alle Zeichen außer s_1, \dots, s_n	$\Sigma - \{s_1, \dots, s_n\}$
$X\{i\}$	i Mal X	X^i
$X\{i, \}$	$\geq i$ Mal X	$X^i \cdot X^*$
$X\{i, j\}$	i bis j Mal X	$X^i \cup X^{i+1} \cup \dots \cup X^j$

■ Reguläre Sprachen

- Operationen auf formalen Sprachen
- Definition regulärer Sprachen
- Reguläre Ausdrücke
- **Eigenschaften regulärer Sprachen**
- Vom regulären Ausdruck zum Automaten
- Vom Automaten zum regulären Ausdruck

■ Kontextfreie Grammatiken



Eigenschaften regulärer Sprachen

Abschlusseigenschaften reguläre Sprachen

Reguläre Sprachen sind abgeschlossen gegenüber

- Vereinigung, Verkettung und Stern (warum?)
- Durchschnitt, Komplement, Differenz, Homomorphismen, Quotientenbildung, ...

Entscheidbarkeit eines Problems: Es gibt ein Verfahren (einen Algorithmus), das für jede Eingabe die richtige Antwort ja/nein liefert.

Nicht entscheidbar: Halteproblem Ihrer Lieblingsprogrammiersprache

- Gegeben ein Programm mit einer Eingabe, wird es anhalten?

Entscheidbare Probleme regulärer Sprachen

- Gegeben ein Wort w und einen regulären Ausdruck X , gilt $w \in \mathcal{L}(X)$? (Wortproblem)
- Gegeben zwei reguläre Ausdrücke X und Y , gilt $\mathcal{L}(X) = \mathcal{L}(Y)$? (Äquivalenzproblem)
- Gegeben einen regulären Ausdruck X , ist $\mathcal{L}(X)$ leer/endlich/unendlich?

Ausdrucks kraft regulärer Sprachen

Die regulären Sprachen sind genau jene, die von endlichen Automaten akzeptiert werden, d.h.:

- Zu jedem regulären Ausdruck X gibt es einen endlichen Automaten \mathcal{A} , sodass $\mathcal{L}(\mathcal{A}) = \mathcal{L}(X)$ gilt.
- Zu jedem endlichen Automaten \mathcal{A} gibt es einen regulären Ausdruck X , sodass $\mathcal{L}(X) = \mathcal{L}(\mathcal{A})$ gilt.

Nicht regulär sind Sprachen, deren Analyse ein unbegrenztes Gedächtnis erfordert:

- Klammerausdrücke: $\{(), (()), ()(), ((())), ((()))(), ()()(), \dots\}$
- $\{a^n b^n \mid n \geq 0\} = \{\varepsilon, ab, aabb, aaabbb, \dots\}$
- $\{a^n b^n c^n \mid n \geq 0\} = \{\varepsilon, abc, aabbcc, aaabbbccc, \dots\}$
- Palindrome: Wörter, die identisch mit ihrem Spiegelbild sind.
 $\{\text{otto}, \text{anna}, \text{reliefpfeiler}, \text{ogenie der herr ehre dein ego}, \dots\}$
- Doppelwörter: $\{ww \mid w \in \Sigma^*\}$ (falls $|\Sigma| > 1$)

■ Reguläre Sprachen

- Operationen auf formalen Sprachen
- Definition regulärer Sprachen
- Reguläre Ausdrücke
- Eigenschaften regulärer Sprachen
- Vom regulären Ausdruck zum Automaten
- Vom Automaten zum regulären Ausdruck

■ Kontextfreie Grammatiken

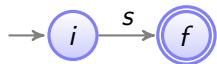


Vom regulären Ausdruck zum Automaten

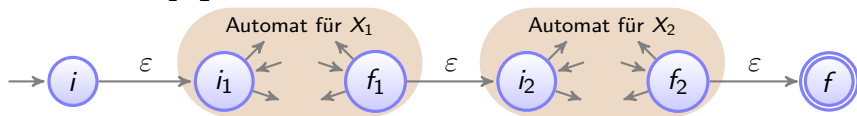
Automat für \emptyset :



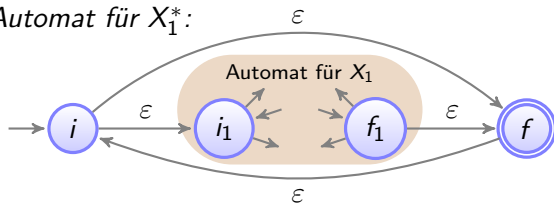
Automat für $s \in \Sigma \cup \{\varepsilon\}$:



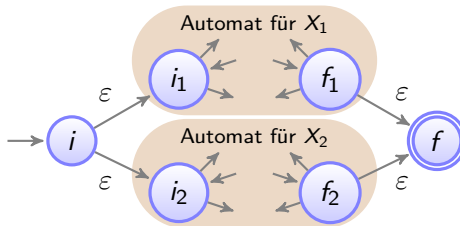
Automat für X_1X_2 :



Automat für X_1^* :

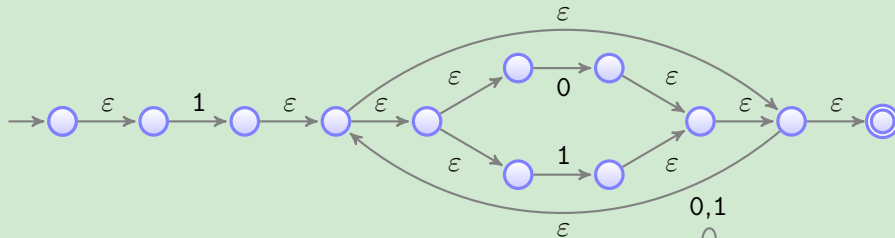


Automat für $X_1 + X_2$:

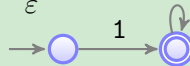


Automat für Binärnumerae ohne führende Null

Regulärer Ausdruck: $1(0 + 1)^*$



Minimaler deterministischer Automat:



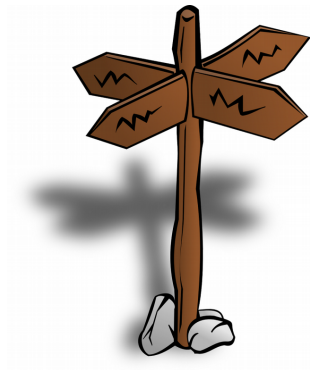
Manuelle Konstruktion von Automaten:

- 1 Konstruiere die Automaten zu einfachen Teilsprachen „durch Hinschauen“.
- 2 Verwende die allgemeine Konstruktion mit ϵ -Übergängen für undurchsichtige Situationen.

■ Reguläre Sprachen

- Operationen auf formalen Sprachen
- Definition regulärer Sprachen
- Reguläre Ausdrücke
- Eigenschaften regulärer Sprachen
- Vom regulären Ausdruck zum Automaten
- Vom Automaten zum regulären Ausdruck

■ Kontextfreie Grammatiken



Vom Automaten zum regulären Ausdruck

$R \dots$ Menge der regulären Ausdrücke über Σ

Verallgemeinerter endlicher Automat

\dots wird beschrieben durch ein 5-Tupel $\mathcal{A} = \langle Q, \Sigma, \delta, i, f \rangle$, wobei

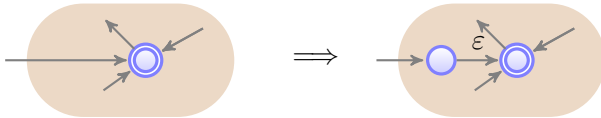
- $Q \dots$ endliche Zustandsmenge
- $\Sigma \dots$ Eingabealphabet
- $\delta: (Q - \{f\}) \times (\Sigma - \{\epsilon\}) \mapsto R \dots$ Übergangsfunktion
- $i \in Q \dots$ Anfangszustand
- $f \in Q, f \neq i \dots$ Endzustand

Unterschiede zu „normalen“ Automaten:

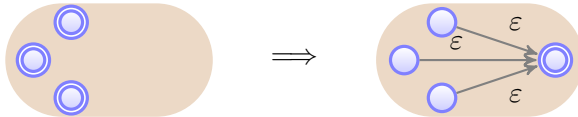
- keine Übergänge in den Anfangszustand;
- nur ein Endzustand, der nicht Anfangszustand ist;
- keine Übergänge weg vom Endzustand;
- nur ein Übergang zwischen je zwei Zuständen;
- Übergänge beschriftet mit regulären Ausdrücken.

Umwandlung eines endlichen Automaten in einen verallgemeinerten

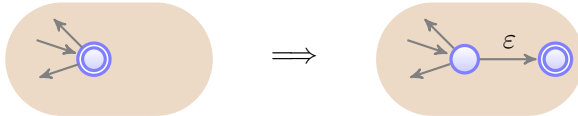
- Übergänge in den Anfangszustand oder Anfangszustand ist Endzustand:



- Mehrere Endzustände:



- Übergänge weg vom Endzustand:



- Mehrere Übergänge zwischen zwei Zuständen:

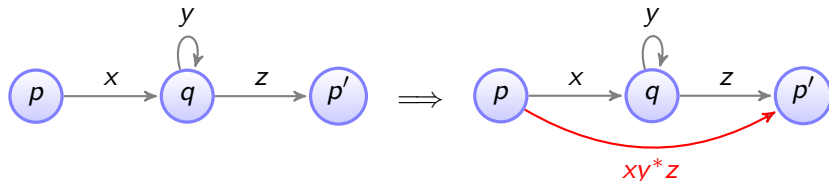


Vom verallgemeinerten Automaten zum regulären Ausdruck

Gegeben: Verallgemeinerter Automat $\mathcal{A} = \langle Q, \Sigma, \delta, i, f \rangle$

Für jeden Zustand $q \in Q - \{i, f\}$ führe folgende Schritte durch:

- 1 Füge zwischen allen Nachbarn p, p' von q neue Übergänge hinzu:



(x , y und z bezeichnen reguläre Ausdrücke.)

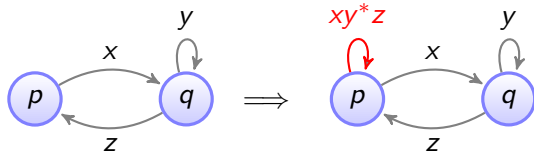
- 2 Entferne q und alle Kanten von und nach q aus dem Automaten.



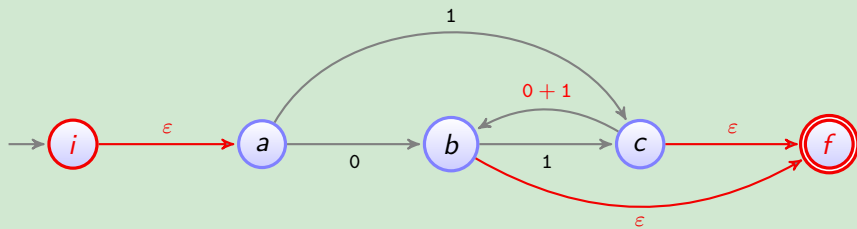
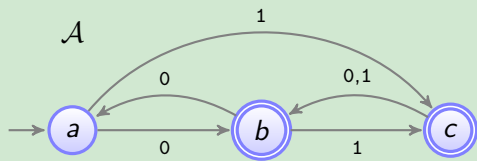
Ergebnis: r ist ein regulärer Ausdruck mit $\mathcal{L}(r) = \mathcal{L}(\mathcal{A})$.

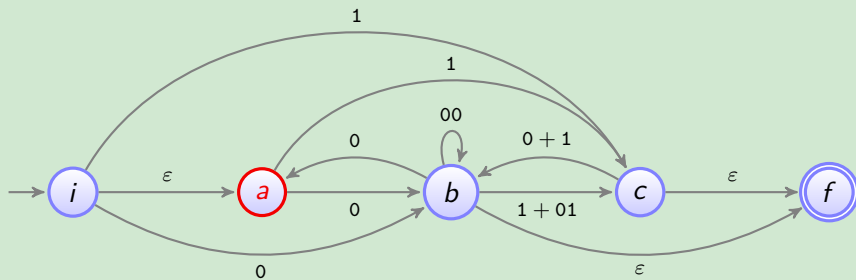
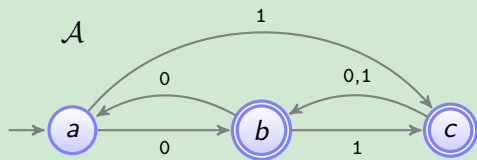
Anmerkungen:

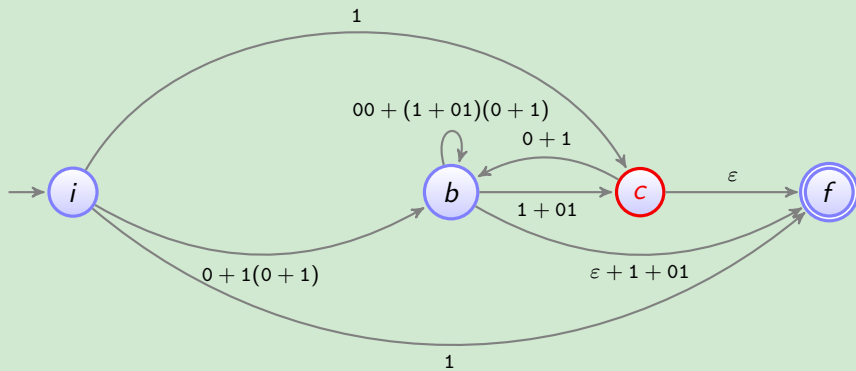
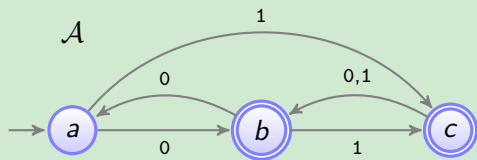
- Falls p und p' derselbe Knoten sind, erhält man:

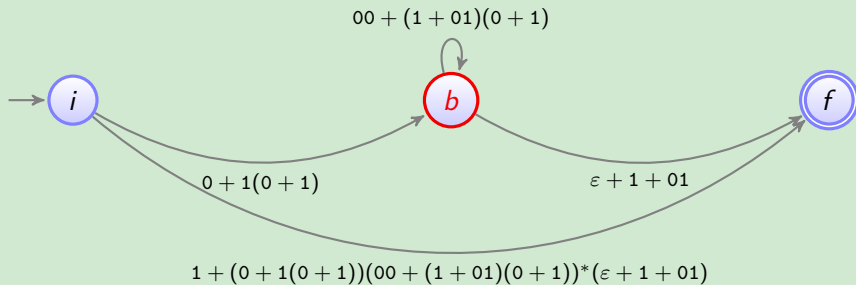
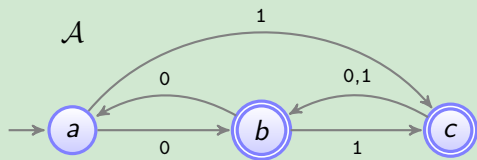


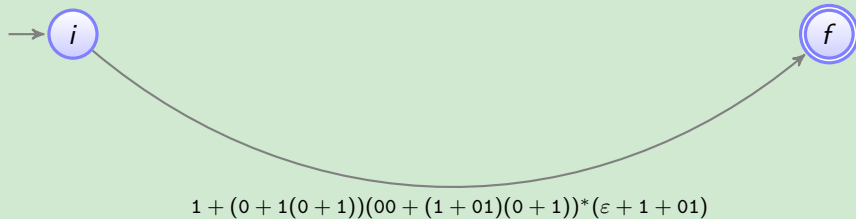
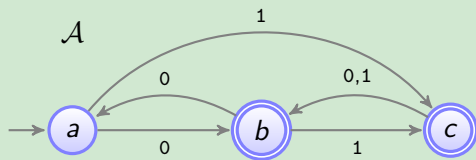
- Falls die Schleife mit dem Ausdruck y nicht existiert, entfällt y^* .
- Falls der Übergang $p-p'$ bereits existiert, wird xy^*z addiert.











$$r = 1 + (0 + 10 + 11)(00 + 10 + 11 + 010 + 011)^*(\varepsilon + 1 + 01)$$

Es gilt: $\mathcal{L}(r) = \mathcal{L}(\mathcal{A})$.