Theoretische Informatik

Manuel Strenge

Aplhabete

Mächtig was ist die Decke der Menge: Unendlich=sehr mächtig

Ein Alphabet ist eine endliche, nichtleere Menge von Symbolen

{:),2} -> alphabet

{1,2,3} -> alphabet

{1,2,3,. . . } -> kein da nicht endlich

{a,. . . ,b} -> alphabet

{a,a,a}-> ja

- $\Sigma = \{a, b, c\}$ ist die Menge der drei Symbole a, b und c.
- $\Sigma = \{-, +, \cdot, :\}$ ist die Menge der Symbole für die Grundrechenarten.
- $\Sigma_{\mathsf{Bool}} = \{0, 1\}$ ist das Boolesche Alphabet.
- $\Sigma_{\mathsf{lat}} = \{a, b, c, \ldots, z\}$ ist die Menge der lateinischen Kleinbuchstaben.
- $\mathbb{N}$ ist kein Alphabet (unendliche Mächtigkeit)

Wort

Ein Wort (Zeichenreihe, String) ist eine endliche Folge von Symbolen eines bestimmten Alphabets

- abc ist ein Wort über dem Alphabet Σ_{lat} (oder über $\Sigma = \{a, b, c\}$).
- 100111 ist ein Wort über dem Alphabet $\{0, 1\}$.

Leeres Wort

Das **leere** Wort ist ein Wort, das keine Symbole enthält. Es wird durch das Symbol ε dargestellt und ist ein Wort über jedem Alphabet.

Wörter

Die Länge eines Wortes w ist die Länge des Wortes als Folge, also die Anzahl der Symbole der Folge. Wir bezeichnen diese Länge mit $|w|$.

- $|abc| = 3$
- $|100111| = 6$
- $|\varepsilon| = 0$
- $|Informatik\ ist\ spannend| = 23$ *(Leerzeichen sind auch Symbole!)*

Definition (Häufigkeit eines Symbols in einem Wort)

$|w|_x$ bezeichnet die absolute Häufigkeit eines Symbols x in einem Wortes w.

- $|abc|_a = 1$
- $|100111|_1 = 4$
- $|\varepsilon|_0 = 0$
- $|Informatik\ ist\ spannend|_n = 4$

Definition (Spiegelung eines Wort)

Mit w^R wird das Spiegelwort zu w bezeichnet.

$$w^R = (x_1, x_2 ... x_n)^R = x_n ... x_2, x_1$$

Es gilt $|w| = |w^R| und |w|_x = |w^R|_x f\ddot{u}r alle x \in \sum$. Wenn $w = wR$ gilt, dann bezeichnet man w als Palindrom.

- $(abc)^R = cba$
- $(100111)^R = 111001$
- $\varepsilon^R = \varepsilon$
- $(Informatik\ ist\ spannend)^R = dnennaps\ tsi\ kitamrofnI$

Definition (Teilwort)

Wir sagen, dass v ein Teilwort (Infix) von w ist, wenn man w als

$$w = xvy$$

für beliebige Wörter x und y über $\sum$ schreiben kann

$$\underbrace{\boxed{\ x\ }\boxed{\quad v\quad}\boxed{\ y\ }}_{w}$$

Definition (echtes Teilwort)

Ein echtes Teilwort von w ist jedes Teilwort von w, das nicht identisch mit w ist (in diesem Falle ist x oder y nicht leer).

- ε, a, b, ab, abb, bb, $abba$, bba und ba sind die Teilwörter von $abba$.
- $abba$ ist kein echtes Teilwort von $abba$ (alle anderen ja).

In Programmiersprachen ist der Begriff substring gebräuchlich.

Präfix

Ein Wort v ist ein Präfix von w, wenn

$$w = xy$$

$$\underbrace{\boxed{\quad v \quad}\boxed{y}}_{w}$$

Ein echtes Präfix von w ist jedes Präfix von w, das nicht identisch mit w ist (in diesem Fall ist y leer).

- ε, a, ab, abb und $abba$ sind die Präfixe von $abba$.
- $abba$ ist kein echtes Präfix von $abba$ (alle anderen ja).

Definition (Suffix)

Ein Wort v ist ein Suffix von w, wenn

$$w = xv$$

$$\underbrace{\boxed{x}\boxed{\quad v \quad}}_{w}$$

Ein echtes Suffix von w ist jedes Suffix von w, das nicht identisch mit w ist (in diesem Fall ist x leer).

- $abba$, bba, ba, a und ε sind die Suffixe von $abba$.
- $abba$ ist kein echtes Suffix von $abba$ (alle anderen ja).

Definition (Menge aller Wörter der Länge k)

Die Menge aller Wörter der Länge k über einem Alphabet $\sum$ wird mit $\sum^k$ bezeichnet.

- Für $\Sigma = \{a, b, c\}$ ist $\Sigma^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$.
- Für $\{0, 1\}$ ist $\{0, 1\}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$.

Definition (Menge aller Wörter (Zeichenreihen))

Die Menge aller Wörter (Kleenesche Hülle) über einem Alphabet $\sum$ wird mit $\sum^*$ bezeichnet. $\sum + = \sum * \varepsilon$ ist die Menge aller nichtleeren Wörter (positive Hülle) über einem Alphabet $\sum$.

Regex definitionen ursprung von hier.

Für $\{0, 1\}$ ist $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$. Wörter aus $\{0, 1\}^*$ nennt man *Binärwörter*.

Eigenschaften

- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$
- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$
- $\Sigma^* = \Sigma^+ \cup \Sigma^0 = \Sigma^+ \cup \{\varepsilon\}$

Definition (Konkatenation)

Definition (Konkatenation) Seien x und y zwei beliebige Wörter. Dann steht

$x \circ y = xy := (x_1, x_2 ... x_n, y_1, y_2 ... y_m)$

für die Konkatenation (Verkettung) von x und y.

Seien $x = 01001$ und $y = 110$ zwei Wörter. Dann ist $xy = 01001110$ die Konkatenation der Wörter x und y.

Definition (Wortpotenzen)

Sei x ein Wort über einem Alphabet $\sum$. Für alle $n \in N$ sind Wortpotenzen wie folgt definiert:

$$\begin{aligned} x^0 &:= \varepsilon \\ x^{n+1} &:= x^n \circ x = x^n x \end{aligned}$$

$a^3 = a^2 a = a^1 aa = a^0 aaa = aaa$

$bbababababbaaaabab = b^2(ab)^4 ba^4 bab = b(ba)^4 b^2 a^3 (ab)^2$
$abbabbabbabbabbabbabbabba = a(bba)^9$

Definition (Sprache)

Eine Teilmenge $L \subseteq \sum^*$ von Wörtern über einem Alphabet $\sum$ wird als Sprache über $\sum$ bezeichnet.

- *Deutsch* ist eine Sprache über dem Alphabet der lateinischen Buchstaben, Leerzeichen, Kommata, Punkte ...
- *Programmiersprachen* (wie C) sind Sprachen über dem Alphabet des ASCII-Zeichensatzes.
- $\{\varepsilon, 10, 01, 1100, 1010, 1001, 0110, 0011, \ldots\}$ ist die Sprache der Wörter über $\{0, 1\}$ mit der gleichen Anzahl von Nullen und Einsen.

Anmerkungen:

- Sprachen können aus unendlich vielen Wörtern bestehen.
- Wörter müssen aus einem festen, endlichen Alphabet gebildet werden.
- Wörter selber haben eine endliche Länge.

Definition (Konkatenation von Sprachen)

Sind $A \subset \sum^*$ und $B \subset \tau^*$ beliebige Sprachen, dann wird die Menge

$$AB = uv|u \in Aundv \in B$$

Die Sprachen A und B sind wie folgt gegeben:

- A enthält alle Binärwörter, die mit 1 beginnen.
- B enthält alle Binärwörter, die mit 0 enden.

Welche der folgenden Wörter sind Elemente von AB?

ε ✗	10 ✓	01010 ✗
1010 ✓	11 ✗	1100110010 ✓

Wie kann man die Elemente von AB einfach beschreiben?

$$AB = \text{Menge der geraden Binärzahlen ohne Null}$$

Reguläre Ausdrücke

Reguläre Ausdrücke sind Wörter, die Sprachen beschreiben, also eine Möglichkeit (gewisse) Sprachen endlich zu repräsentieren.

- Die Syntax der regulären Ausdrücke befasst sich mit der Frage, welche Form diese Wörter haben.
- In der Semantik der regulären Ausdrücke wird erklärt, wie man reguläre Ausdrücke als Sprachen interpretiert.

Gegeben: Das Wort 101 über dem Aphabet $\Sigma = \{0, 1, 2, ..., 9\}$

- Die Syntax beschreibt, wie die Symbole des Alphabets zu einem Wort angeordnet bzw. aneinandergereiht werden.
- Aus der Semantik geht hervor, was diese Zeichenreihe bedeutet: Z.B. die Zahl 101 im Zehnersystem, die Zahl 5 im Dualsystem oder einfach nur eine Folge von Symbolen usw.

Ein regulärer Ausdruck, der die Sprache aller Binärwörter der Länge 4 beschreibt:

$$\underbrace{(0|1)}_{\text{0 oder 1}}\ \underbrace{(0|1)}_{\text{nochmals}}\ \underbrace{(0|1)}_{\text{dreimal}}\ \underbrace{(0|1)}_{\text{genug}}$$

Ein passender regulärer Ausdruck ist also

$$(0|1)(0|1)(0|1)(0|1)$$

Ein regulärer Ausdruck für die Sprache der Binärwörter, die das Teilwort 00 enthalten:

$$\underbrace{(0|1)^*}_{\text{0 oder 1 beliebig oft}}\quad \underbrace{00}_{\text{das Teilwort}}\quad \underbrace{(0|1)^*}_{\text{0 oder 1 beliebig oft}}$$

Ein passender regulärer Ausdruck ist also

$$(0|1)^*00(0|1)^*$$

Definition (Reguläre Ausdrücke)

Es sei $\sum$ ein beliebiges Alphabet. Die Sprache $RA\sum$ der regulären Ausdrücke über $\sum$ ist wie folgt definiert:

- $\varnothing, \epsilon \in \mathsf{RA}_\Sigma$
- $\Sigma \subset \mathsf{RA}_\Sigma$
- $R \in \mathsf{RA}_\Sigma \Rightarrow (R^*) \in \mathsf{RA}_\Sigma$
- $R, S \in \mathsf{RA}_\Sigma \Rightarrow (RS) \in \mathsf{RA}_\Sigma$
- $R, S \in \mathsf{RA}_\Sigma \Rightarrow (R|S) \in \mathsf{RA}_\Sigma$

Erläuterungen zur Definition

- Die Sonderzeichen ε und $\varnothing$ sind reguläre Ausdrücke.
- Jedes Symbol aus dem Alphabet $\sum$ ist auch ein regulärer Ausdruck über $\sum$.

- Ist R ein regulärer Ausdruck über $\sum$, dann ist auch (R^*) ein regulärer Ausdruck über $\sum$.
- Sind R und S reguläre Ausdrücke über $\sum$, dann sind auch (RS) und $(R|S)$ Ausdrücke über $\sum$.

Eigenschaften und Konventionen:

–

- Die **Menge** RA$_\Sigma$ **der regulären Ausdrücke** über dem Alphabet Σ ist **eine Sprache** über dem Alphabet $\{\varnothing, \epsilon, {}^*, (,), |\} \cup \Sigma$.

  Elemente von RA$_\Sigma$ sind z.B. $\varnothing, \epsilon, a, a^*, a|b, aba, ..$
- Der Lesbarkeit halber werden "überflüssige" Klammern weggelassen.
- Damit reguläre Ausdrücke auch mit (teilweise) weggelassenen Klammen eindeutig lesbar bleiben, gilt folgende Rangfolge der Operatoren:
  a) "*" vor "Konkatenation" und
  b) "Konkatenation" vor "|".

  Der Ausdruck $ab^*|c$ wird beispielsweise als $((a(b^*))|c)$ gelesen.

Einige reguläre Ausdrücke über dem Alphabet $\{a, b\}$.

- a^*b
- $(aa)^*b^*aba$
- $(a|(ab))^*$
- $(ab)|(ba)$
- $a(b(ba))|b$

Reguläre Sprachen

Satz (Rechenregeln für reguläre Ausdrücke)

- L(R|S) = L(S|R)
- L(R(ST)) = L((RS)T)
- L(R|(S|T)) = L((R|S)|T)
- L(R(S|T)) = L(RS|RT)
- L((R*)*) = L(R*)
- L(R|R) = L(R)

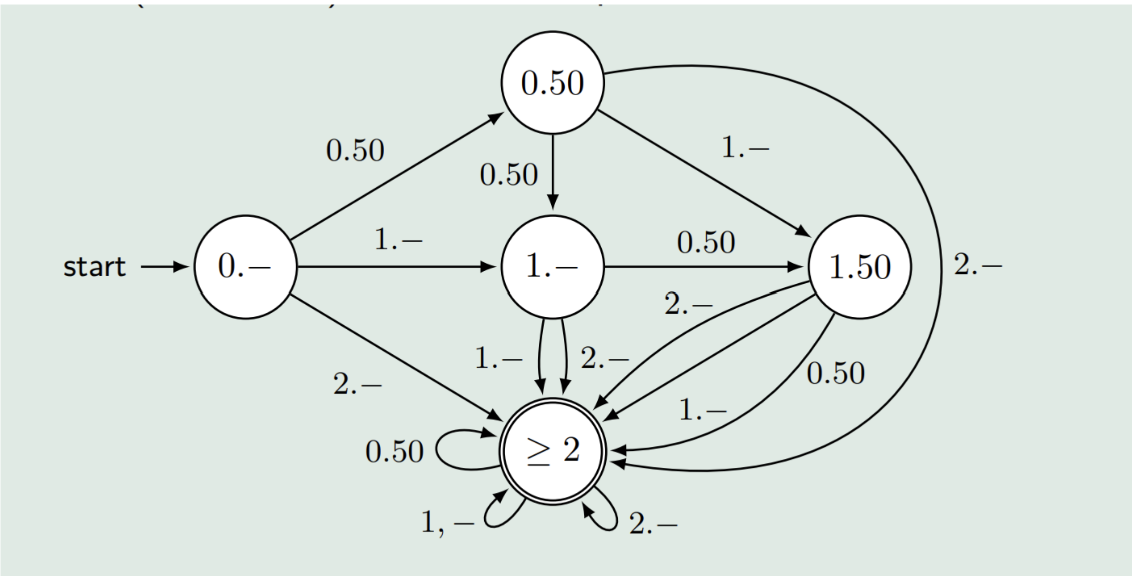
Anwendungen von regulären Ausdrücken:

- Mustersuche in Texten
- Lexikalische Analyse (in Compilern); Erkennung von Schlüsselwörtern ("Token")
- Syntax Test (bei einer einfachen Syntax)

Endliche Automaten

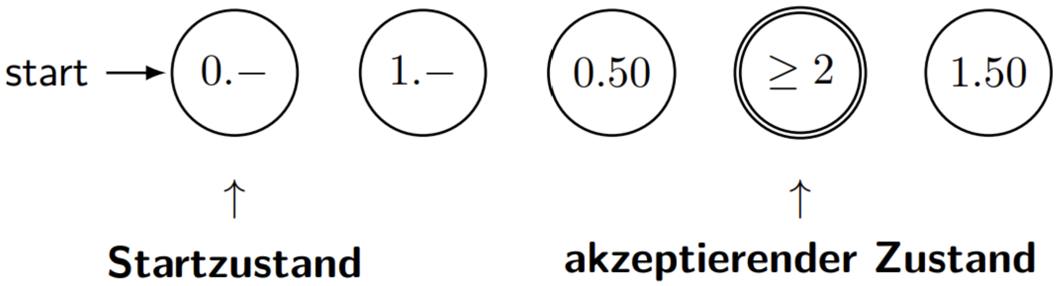
Beispiel (Einstiegsaufgabe: Eintrittskarte Schwimmbad)

Kosten 2.- (mindestens), Automat akzeptiert 0.50, 1.- und 2.-



Ein **endlicher Automat** besteht aus (elementare Bausteine):

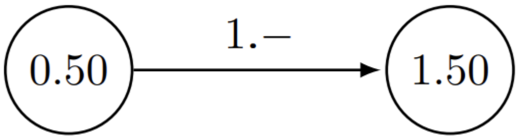
Zuständen



Eingabealphabet

0.50, 1.-,2.-

Übergangsfunktionen

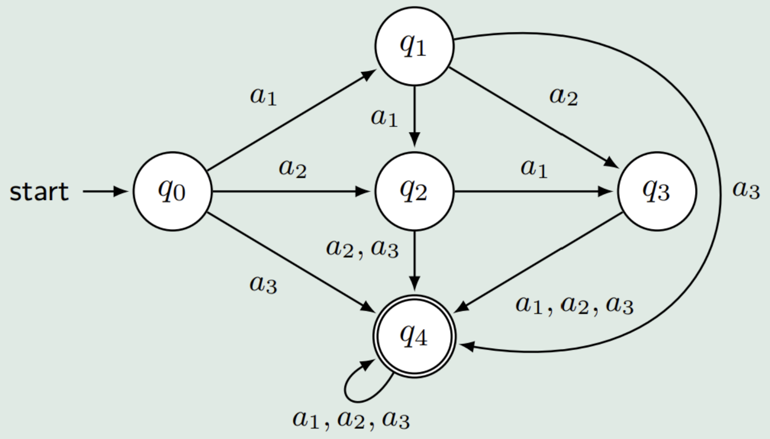


Wie wird die Eingabe eingegeben? -> Der endliche Automat liest das Wort von links nach rechts

Wieviel Speicher steht zur Verfügung? Wie geht man mit dem Speicher um? -> Es gibt keinen Speicher. -> Variablen dürfen nicht benutzt werden. -> Der einzige (gespeicherte) Information ist der aktuelle Zustand.

Wie wird die Ausgabe bestimmt (und ausgegeben)? -> Die Ausgabe erfolgt über akzeptierende Zustände.

Automat A:



Zustände:
$q_0 =$ Start
$q_1 = 0.50$
$q_2 = 1.–$
$q_3 = 1.50$
$q_4 \geq 2$

Eingabealphabet:
$a_1 = 0.50$
$a_2 = 1.–$
$a_3 = 2.–$

Übergangsfunktionen:
$\delta(q_0, a_1) = q_1, \ \ldots$

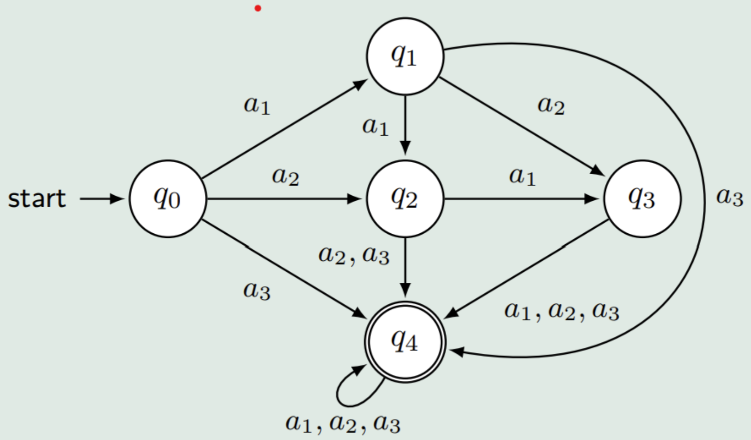
Definition (Endlicher Automat)

Ein (deterministischer) endlicher Automat (EA) ist ein Quintupel

$$M = (Q, \Sigma, \delta, q_0, F)$$

- endlichen Menge von Zuständen $Q = \{q_0, q_1, \ldots, q_n\}$ $(n \in \mathbb{N})$
- Eingabealphabet $\Sigma = \{a_1, a_2, \ldots, a_m\}$ $(m \in \mathbb{N})$
- Übergangsfunktion $\delta : Q \times N \to Q$
- Startzustand $q_0 \in Q$
- Menge der akzeptierenden Zustände $F \subseteq Q$

Automat A:



$\delta(q_0, a_1) = q_1$
$\delta(q_0, a_2) = q_2$
$\delta(q_0, a_3) = q_4$
$\delta(q_1, a_1) = q_2$
$\delta(q_1, a_2) = q_3$
$\delta(q_1, a_3) = q_4$
$\delta(q_2, a_1) = q_3$
$\delta(q_2, a_2) = q_4$
$\delta(q_2, a_3) = q_4$
$\delta(q_3, a_1) = q_4$
$\delta(q_3, a_2) = q_4$
$\delta(q_3, a_3) = q_4$

$\delta(q, a) = p$ bedeutet: EA wechselt zu p, falls in q Symbol a gelesen wird.

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

- Zustandsmenge $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- Eingabealphabet $\Sigma = \{a_1, a_2, a_3\}$
- Startzustand q_0
- akzeptierende Zustände $F = \{q_4\}$
- Übergangsfunktion δ:

	Eingabe		
Zustand	a_1	a_2	a_3
q_0	q_1	q_2	q_4
q_1	q_2	q_3	q_4
q_2	q_3	q_4	q_4
q_3	q_4	q_4	q_4
q_4	q_4	q_4	q_4

Sind NEAs mächtiger als DEAs?

Es gibt einen DEA, der die Sprache L akzeptiert.

vs

Es gibt einen NEA, der die Sprache L akzeptiert.

Jeder DEA ist ein NEA.

Teilmengenkonstruktion (siehe nächste Folie)

Beweiskonstruktion.

Sei $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ ein NEA

Der dazu äquivalente DEA $D = (Q_D, \Sigma, \delta_D, q_0, F_D)$ wird konstruiert durch:

$$Q_D = \mathcal{P}(Q_N)$$

(Menge aller Teilmengen von Q_N)

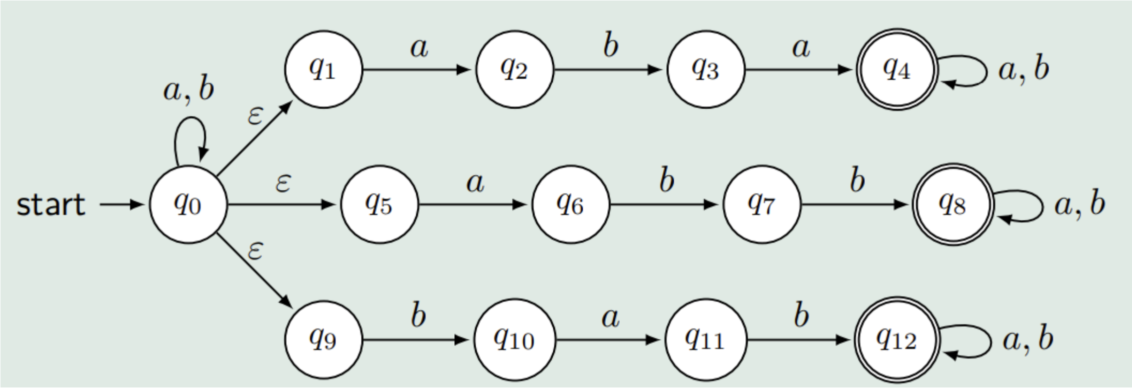
$$F_D = \{S \in Q_D \mid S \cap F_N \neq \emptyset\}$$

(alle Mengen aus Q_D, die mindestens einen akzeptierenden Zustand aus F_N enthalten.)

$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$ für $S \in Q_D$ und $a \in \Sigma$ (Menge aller Zustände von D, die von den Zuständen aus S durch Lesen von a erreichbar sind.)

NEAs mit ϵ -Übergängen

Suche nach einem von mehreren Mustern: $L_5 = \{w \in \{a, b\}^* \mid w$ enthält eines der Teilwörter $aba, abb, bab\}$



Ein nichtdeterministischer endlicher Automat mit ε-Übergängen (ε-NEA) wird beschrieben durch

$$M = (Q, \Sigma, \delta, q_0, F)$$

wobei Q, Σ, q_0 und F wie beim deterministischen endlichen Automaten definiert sind und die Übergangsfunktion δ definiert ist als.

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

Äquivalenz DEA und RA

Satz (Gleichmächtigkeit von RA und DEA)

Es gibt einen DEA, der $\iff$ Es gibt einen RA, der die die Sprache L akzeptiert. Sprache L akzeptiert.

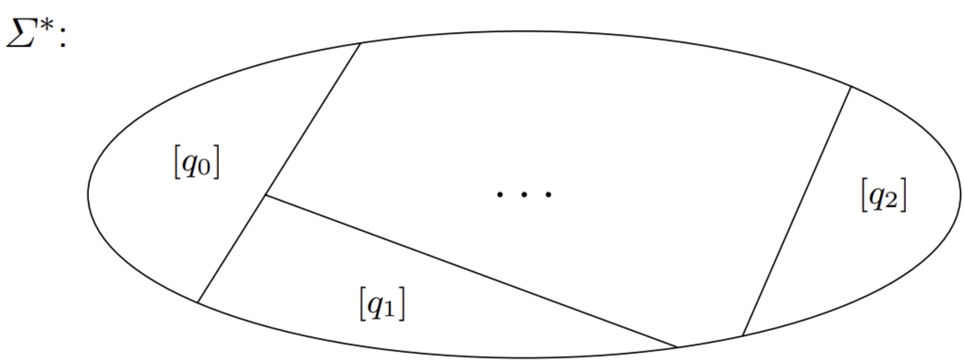
Beweis

- Dynamische Programmierung: Für jedes Paar von Zuständen p, q regulären Ausdrück finden, der alle Wörter beschreibt, die von p nach q führen.
- RA in einen speziellen NEA umwandeln, der auch spontane Übergänge (ohne ein Eingabesymbol zu lesen) zulässt. Diese sogenannten ε-NEAs können in NEAs und somit durch Teilmengenkonstruktion in DEAs umgewandelt werden.

Die Klasse der **regulären Sprachen** beinhaltet alle Sprachen, die von einem endlichen Automaten akzeptier t werden. Jede dieser Sprachen wird regulär genannt.

Zustandsklassen

$$\text{Klasse } [p] = \{w \in \Sigma^* \mid M \text{ endet nach dem Lesen von } w \text{ in } p\}$$



Die Menge alle Wörter Σ^* wird von den Zustandsklassen $[p_0]$ bis $[p_n]$ partitioniert.

Eigenschaften der Klassen:

- Jedes Wort landet in einem Zustand:

$$\Sigma^* = \bigcup_{p \in Q} [p]$$

- Kein Wort landet nach dem Lesen in zwei Zuständen:

$$[p] \cap [q] = \emptyset, \text{ für alle } p \neq q, p, q \in Q$$

- Von M akzeptierte Sprache:

$$L(M) = \bigcup_{p \in F} [p]$$

Grenzen endlicher Automaten

Wenn ein EA M nach dem Lesen zweier Präfixe x und y im gleichen Zustand landet, kann er nicht mehr zwischen x und y unterscheiden.

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein EA und x und y zwei beliebige Wörter aus Σ^*, so dass $x, y \in [p]$. Dann gilt für alle Wörter $z \in \Sigma^*$

$$xz \in L(M) \quad \iff \quad yz \in L(M).$$

Endlichen Automaten definieren: 5 Tupel Übergang skizieren zustände Konfiguration eines Automaten; eine momentaufnahem zustand und rest des eingabewortes (Q0,aabba) Berechnung start, rechnungsschritte…., endkonfiguration (erfolg oder nein)

Kontextfreie Grammatiken

Einführung und Definition

Kontextfreie Grammatik für die nicht-reguläre Sprache $\{0^n 1^n \mid n \in \mathbb{N}\}$

$$A \to 0A1$$
$$A \to \varepsilon$$

Eine Ableitung des Wortes 000111 in der Grammatik:

$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000111$

Eine kontextfreie Grammatik G (KFG) ist ein 4-Tupel (N, Σ, P, A)

- N ist das Alphabet der Nichtterminale (Variablen)
- Σ ist das Alphabet der Terminale.
- P ist eine endliche Menge von Produktionen (Regeln). Jede Produktion hat die Form

$$X \to \beta$$

mit Kopf $X \in N$ und Rumpf $\beta \in (N \cup \Sigma)^*$.

- A ist das Startsymbol, wobei $A \in N$.

Definition (Satzform) Ein Wort $\beta \in (N \cup \Sigma)^*$ nennen wir Satzform.

Ableitungsschritt, Ableitung

Seien α, β und γ Satzformen und $A \to \gamma$ eine Produktion.

Durch einen Ableitungsschritt wird eine Satzform $\alpha A \beta$ durch die Anwendung der Produktion $A \to \gamma$ in die Satzform $\alpha\gamma\beta$ abgeleitet. Das notieren wir mit:

$$\alpha A \beta \Rightarrow \alpha\gamma\beta$$

Eine Ableitung ist eine Folge von Ableitungsschritten, so dass aus einer Satzform α das Wort w abgeleitet wird.

$$\alpha \Rightarrow \ldots \Rightarrow w$$

Ein Wort $w \in \Sigma^*$ ist in einer kontextfreien Grammatik $G = (N, \Sigma, P, A)$ ableitbar, falls es eine Ableitung in G gibt, die mit dem Startsymbol A beginnt und mit dem Wort w endet. Dafür schreiben wir

$$A \overset{*}{\Rightarrow} w$$

Wir sagen auch, dass w von A erzeugt oder generiert wird.

Rechtseitige und linksseitige Ableitungen

- Eine linksseitige Ableitung ersetzt bei jedem Ableitungsschritt das Nichtterminal, das am weitesten links in der Satzform auftritt
- Eine rechtsseitige Ableitung ersetzt bei jedem Ableitungsschritt das Nichtterminal, das am weitesten rechts in der Satzform auftritt.

Sprache eine KFG

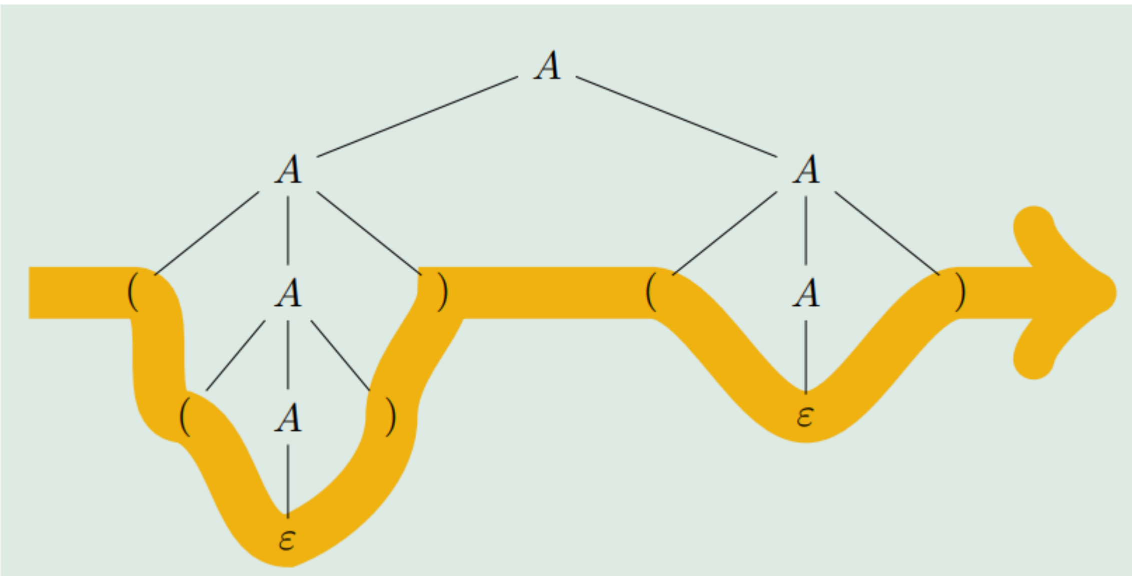
Die von G erzeugte Sprache $L(G)$ beinhaltet alle Wörter, die in G aus dem Startsymbol A ableitbar sind.

$$L(G) = \left\{ w \in \Sigma^* \mid A \overset{*}{\Rightarrow} w \right\}$$

Wenn es für eine Sprache L eine kontextfreie Grammatik G gibt mit $L = L(G)$, dann nennen wir L eine kontextfreie Sprache.

Ableitungsbaum

Ein **Ableitungsbaum** ist eine graphische Darstellung einer Ableitung. Beispiel (Ableitungsbaum für das Wort (())() i



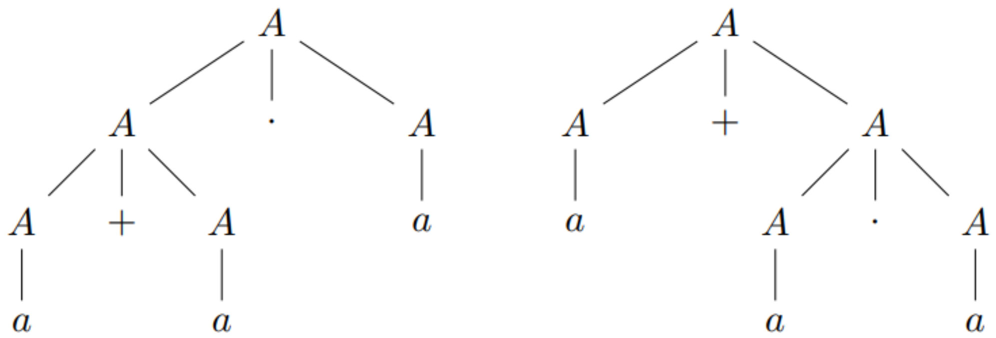
Mehrdeutigkeit (von KFG)

Beispiel (Kontextfreie Grammatik für einfache arithmetische Ausdrücke)

$$G_3 = \{\{A\}, \{a, \cdot, +, (,)\}, P, A\}, \text{ mit } P = \{A \to A + A | A \cdot A | (A) \mid a\}$$

(a steht vereinfachend für eine beliebige Binärzahl)

Das Wort $a + a \cdot a$ kann durch zwei verschiedene Ableitungsbäume dargestellt werden:



Definition (Mehrdeutige kontextfreie Grammatik) > Eine kontextfreie Grammatik nennen wir mehrdeutig, wenn es ein Wort gibt, das mehrere Ableitungsbäume besitzt.

Definition (Inhärent mehrdeutig)

> Eine kontextfreie Sprache, für die alle Grammatiken mehrdeutig sind, heisst inhärent mehrdeutig.

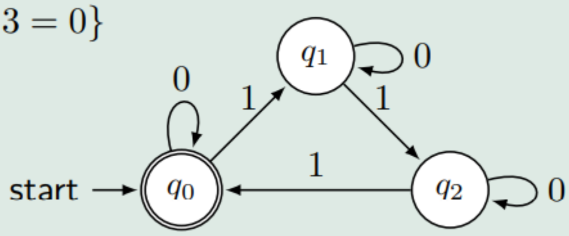
Die kontextfreien Sprachen enthalten die regulären Sprachen.

Zusammenhang von KFG und regulären Ausdrücken

Jede reguläre Sprache kann durch eine kontextfreie Grammatik beschrieben werden. Sei L eine reguläre Sprache. Dann gibt es einen DEA $M = (Q, \Sigma, \delta, q_0, F)$ mit $L(M) = L$ Dann können wir eine KFG für L wie folgt bauen: 1 Für jeden Zustand q_i gibt es ein Nichtterminal Q_i. 2 Für jede Transition $\delta(q_i, a) = q_j$ erstellen wir die Produktion $Q_i \to aQ_j$. 3 Für jeden akzeptierenden Zustand $q_i \in F$ erstellen wir die Produktion $Q_i \to \varepsilon$. 4 Das Nichtterminal Q_0 wird zum Startsymbol.

Beispiel

$L_5 = \{ w \in \{0,1\}^* \mid |w|_1 \mod 3 = 0\}$



Nichtterminale:

Q_0, Q_1, Q_2

Produktionen:

$$Q_0 \to 0Q_0 \mid 1Q_1 \mid \varepsilon$$
$$Q_1 \to 0Q_1 \mid 1Q_2$$
$$Q_2 \to 0Q_2 \mid 1Q_0$$

Beispiel für Ableitung von $w = 10011$:

$$Q_0 \Rightarrow 1Q_1 \Rightarrow 10Q_1 \Rightarrow 100Q_1 \Rightarrow 1001Q_2 \Rightarrow 10011Q_0 \Rightarrow 10011$$

Techniken für den Entwurf von kontextfreien Grammatiken:

- Komplexe KFGs können oft in mehrere einfachere KFGs aufgeteilt werden und danach mit der Regel $A \to A_1 \,|A_2| \ldots \mid A_k$ kombiniert werden.
- Um eine KFG für eine reguläre Sprache zu erstellen, kann zuerst ein DEA erstellt werden und dieser dann in eine KFG umgewandelt werden.
- Kontextfreie Sprachen enthalten manchmal Teilwörter, die voneinander „abhängig" sind. Eine KFG für diese Situation kann mit einer Regel $R \to uRv$ behandelt werden.
- Komplexere Sprachen sind meist rekursiv aufgebaut. Steht zum Beispiel das Nichtterminal A für einen Ausdruck, kann A wiederum überall dort verwendet werden, wo dieser Ausdruck erlaubt ist.