Satz (1): Zu jedem RA lässt sich ein EA angeben, der die gleiche Sprache beschreibt.

Beweis: Wir geben die Konstruktionsprinzipien an:

(1) Für elementare (nicht zus.gesetzte) RA:
Sei $x \in T$, dann ist x RA mit einer Sprache, die durch den EA

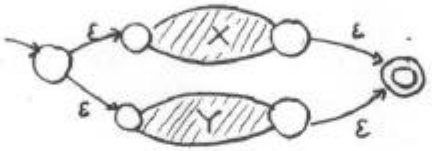
beschrieben wird.

ε ist RA mit EA:

Automat für leere Sprache

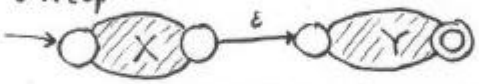
(2) Seien X und Y RA, die durch die EA

Für [X|Y] akzeptiert



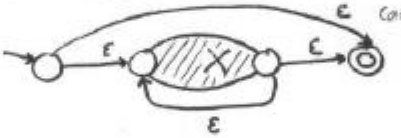
die gleiche Sprache.

Für XY akzeptiert



die gleiche Sprache.

Zu X^*:

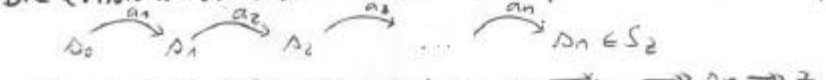


Die Überlegungen zeigen: Zu RA können wir einen EA angeben der die gleiche formale Sprache beschreibt – vorausgesetzt wir finden bei zusammengesetzten RA EA für die Unterausdrücke. Vollständige Induktion über die Anzahl der Symbole in einem RA liefert die Behauptung. □

Wichtiges Beweis-Prinzip in der Informatik! Zu jeder Formel Ausdruck konstruierbar

Satz (2a): Zu jedem det. ε-freien EA können wir eine Ch-3-G angeben, die die gleiche Sprache beschreibt.

Beweis: (konstruiere Ch-3-G)
gegeben EA $A = (S, T, s_0, S_z, \delta)$
O.B.d.A. gehe eine Kante nach s_0. Jeder Kante (d.h. jedem Übergang $\delta(s,a) = s'$ ordnen wir eine Regel d. Grammatik zu: wir verwenden Zustände als Nichtterminalzeichen.
Für Übergänge vom Anfangszustand s_0 aus: $\delta(s_0, a) = s_i$ verwenden wir die Regel $a \to s_i$.
Für Übergänge $\delta(s_i, a) = s_j$ ($s_i \xrightarrow{a} s_j$) führen wir die Regel $s_i a \to s_j$ ein.
Zu den Zuständen als Nichtterminalzeichen nehmen wir noch das Sonderzeichen, die „Wurzel" z hinzu. (o.B.d.A sei $z \in S$)
Für jeden Zustand $s_i \in S_z$ nehmen wir die Regel $s_i \to z$ hinzu.
Die entstehende Grammatik akzeptiert die gleiche Sprache.



$a_1 \dots a_n \Rightarrow s_0 a_2 \dots a_n \Rightarrow s_2 a_3 \dots a_n \Rightarrow \dots \Rightarrow s_n \Rightarrow z$ Grammatik

→ als Hülle

Satz: Zu jeder Ch-3-G ex. ein EA, der die gleiche Sprache akzeptiert.

Beweis: o.B.d.A. sei die Ch-3-G linkslinear (rechtslinear: $A \to wA'$, $A, A' \in N$, $w \in T^*$)

(2b) und alle linkslinearen Regeln von der Form $Aa \to A'$ mit $a \in T$. Wir geben einen EA an (sei $A_0 \notin N \cup T$):

$A \to w_1 \dots w_n \to A_n$
$A \to w_1 \to A_1$
$A_1 w_2 \to A_2$
$\vdots$
$A_{n-1} w_n \to A'$

Zustände	$S = N \cup \{A_0\}$
Eingangszeichen	T
Anfangszustand	A_0
Endzustände	$\{Z\}$

Übergangsfunktion:
$\delta(A,a) = \{A' \in S : Aa \to A' \text{ Regel}\}$ für $A \in N, a \in T$.
$\delta(A,\varepsilon) = \{A' \in S : A \to A' \text{ Regel}\}$ für $A \in N$
$\delta(A_0,a) = \{A' \in S : a \to A' \text{ Regel}\}$ für $A \in N, a \in T$.

Es entsteht ein Automat, der die gleiche Sprache wie die Grammatik beschreibt. Jede Ableitung in der Grammatik entspricht einer Folge von Zustandsübergängen und umgekehrt. □

Fazit: Ein EA beschreibt Sprache L $\overset{(2a)(2b)}{\Longleftrightarrow}$ Eine Ch-3-G beschreibt Sprache L.
$\Updownarrow$ (1)
Ein RA beschreibt Sprache L ⟸ folgt aus Transitivität

Fazit: (1) Die Beschreibungsmittel EA, RA, Ch-3-G sind gleich mächtig; beschreiben die gleiche Klasse formaler Sprachen.
(2) Die Übergänge sind konstruktiv: zu jedem (EA, RA, Ch-3-G) können wir systematisch (durch einen Algorithmus) einen RA (Ch-3-G, EA) angeben, der die gleiche formale Sprache beschreibt.
Solche Sprachen heißen regulär.

Wir wenden uns nun der Frage zu, wie wir für eine vorgegebene formale Sprache erkennen können, dass sie regulär ist.
Wir beginnen mit einem einfachen Beispiel für eine nichtreguläre Sprache $L_{ab} = \{a^n b^n : n \in \mathbb{N} \setminus \{0\}\}$
Diese Sprache können wir ohne Probleme durch eine Ch-2-G beschreiben:

$T = \{a, b\}$ Regeln: $ab \to Z$
$N = \{Z\}$ $aZb \to Z$
Z Wurzel

Behauptung: Die Sprache L_{ab} ist nicht regulär. — d.h. es ex. kein EA oder RA der die Sprache beschr.
Beweis: Wir zeigen, dass kein EA existiert, der L_{ab} akzeptiert.
Annahme: Ein solcher EA existiert.
Widerspruchsbeweis:
(1) Der endl. Automat hat nur endl. viele Zustände (sei k die Anzahl der Zustände).
(2) $a^n b^n$, $a^m b^m$ sind im Sprachschatz, d.h. es existieren Zustände A_n, A_m, A_Z

$A_0 \xrightarrow{a^n} {}^* A_n \wedge A_n \xrightarrow{b^n} {}^* A_Z$
$A_0 \xrightarrow{a^m} {}^* A_m \wedge A_m \xrightarrow{b^m} {}^* A_Z$

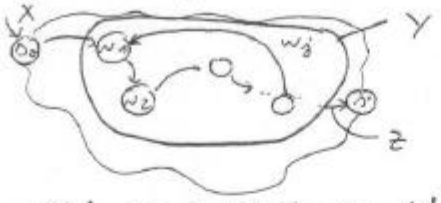
Falls $A_n = A_m$ gilt, werden auch Wörter $a^n b^m$ $a^m b^n$ akzeptiert. Da nur endl. viele Zustände (genau k Zustände) existieren, muss es Zahlen n und m geben mit $(n \neq m)$ und $A_n = A_m$. ↯ Widerspruch □

Problem bei EA: EA können nur endliche Information speichern, können nicht unbeschränkt zählen.

Sprachen mit Klammerstrukturen (beliebiges Nesten von öffnenden und schließenden Klammern) sind nicht regulär.

Satz: Pumping Lemma für reguläre Sprachen

Zu jeder regulären Sprache L ex. eine Zahl $n \in \mathbb{N}$, so dass für alle Wörter $w \in L$ mit $|w| \geq n$ gilt:
w lässt sich in Wörter $x, y, z \in T^*$ zerlegen: $w = x \cdot y \cdot z$
mit $|y| \geq 1$, $|x \cdot y| \leq n$,
$x \cdot y^i \cdot z \in L$ für alle $i \in \mathbb{N} \setminus \{0\}$



Dadurch dass es nur endl. viele Zustände gibt, gibt es Zyklen.

Beweis: L regulär! $\Rightarrow$ existiert ein det., endlicher, ε-freier Automat, der L akzeptiert. Sei n die Anzahl seiner Zustände, $w \in L$.
Es gilt: $s_0 \xrightarrow{w} s_z$.
Dabei durchläuft der Automat $|w|+1$ Zustände. Gilt $|w| \geq n$, so tritt ein Zustand mindestens 2-mal auf: d.h.

$$s_0 \xrightarrow{x} s_k \wedge s_k \xrightarrow{y} s_k \wedge s_k \xrightarrow{z} s_z,$$

dann gilt auch (i-mal Zyklus durchlaufen)

$$s_0 \xrightarrow{x} s_k \wedge s_k \xrightarrow{y^i} s_k \wedge s_k \xrightarrow{z} s_z \quad \square$$

Damit lässt sich auch zeigen: $L_{a,b}$ ist nicht regulär.

1.3.5. Minimale Automaten

Wir zeigen nun, wie wir für eine reguläre Sprache einen minimalen Automaten (kleinste Anzahl von Zuständen) angeben können.
Sei $L \subseteq T^*$; wir definieren eine Äquivalenzrelation $\approx_L$ auf T^* durch (sei $v, w \in L^*$):

$$v \approx_L w \iff_{\text{def}} \forall u \in T^*: v \cdot u \in L \iff w \cdot u \in L$$

Dadurch erhalten wir Äquivalenzklassen $[v]_L = \{w \in T^* : w \approx_L v\}$

Satz: (Myhill-Nerode)
L ist genau dann regulär, wenn die Menge der Äquivalenzklassen endlich ist.

Beweisidee: Jede Äquivalenzklasse definiert einen Zustand in einem EA, der die Sprache akzeptiert. Umgekehrt: jeder Zustand in einem EA entspricht einer Äquivalenzklasse. $\square$

Die Beweisidee liefert bereits einen Hinweis, wie wir einen ε-freien, det., endlichen Automat mit totaler Übergangsfunktion konstruieren können, der die Sprache L akzeptiert, und eine minimale Anzahl von Zuständen hat.
Wir zeigen nun, wie wir aus einem gegebenen endl., ε-freien, det. Automaten konstruieren indem wir bestimmte Zustände zusammenlegen.

1. Schritt: Wir entfernen alle Zustände, die vom Anfangszustand aus nicht erreichbar sind. (neue Zustandsmenge S')
2. Schritt: Wir konstruieren eine Folge von Prädikaten auf Zuständen: $p_i : S' \times S' \to \mathbb{B}$

$$p_i(s_1, s_2) : \forall w \in T^* : |w| \leq i \Rightarrow (\exists z \in S_z : s_1 \xrightarrow{w} z) \iff (\exists z \in S_z : s_2 \xrightarrow{w} z)$$

p_i lässt sich induktiv definieren: $p_0(s_1, s_2) = (s_1 \in S_z \iff s_2 \in S_z)$ (ε-Freiheit vorausgesetzt)

$$p_{i+1}(s_1, s_2) : p_i(s_1, s_2) \wedge \forall x \in T : p_i(\delta(s_1, x), \delta(s_2, x))$$

Da der Automat endlich ist, existiert ein k mit $p_k = p_{k+1}$. Das Prädikat p_k definiert uns dann, welche Zustände der Automaten äquivalent sind und zu einem Zustand (im minimalen Automaten) verschmolzen werden können.