„Automatentheorie und formale Sprachen“

Gehalten von Prof. Dr. Thomas im SS04

Eine studentische Mitschrift von

Benedikt Westermann und Daniel Steinberger

Letze Änderung: 7. September 2004

Wir danken Corinna Habets, Gregor Fabritius und Jan Newger, daß sie in Situationen, in denen wir nicht schnell genug Grafiken oder Ähnliches auf unseren Notebooks einfangen konnten, immer eine hilfreiche Mitschrift für uns zur Verfügung hatten. Sowie Sascha Wiedenfeld für progressives Korrekturlesen.
Vielen Dank auch für die Korrekturvorschläge, die uns per Mail erreicht haben!

Bitte meldet es uns, falls ihr Fehler in dieser Mitschrift findet!
Wir schreiben all das hier für uns und euch auf. Also tut also euren Gönnern und euren Kommilitonen etwas Gutes, und tragt dazu bei, ein möglichst fehlerfreies Script zu produzieren. Ihr könnt uns erreichen unter:

`atfs@beneficium.de` oder
`Daniel.Steinberger@gmx.de`.

Inhaltsverzeichnis

1 Endliche Automaten **4**
1.1 Einführung . . . 4
1.2 Vier Typen endlicher Automaten . . . 4
1.3 Definition des DEA . . . 6
1.4 Induktive Definition von δ^* . . . 7
1.5 Definition eines NEA . . . 7
1.5.1 Weitere Beispiele für DEA's und NEA's . . . 7
1.6 Ein Lauf auf einem Wort . . . 8
1.6.1 Beispiele für Läufe . . . 8
1.7 Erreichbarkeitsmenge . . . 9
1.7.1 Beispiele zur Erreichbarkeitsmenge . . . 9
1.7.2 Update der Erreichbarkeitsmenge . . . 9
1.8 Äquivalenz von NEA und DEA . . . 9
1.8.1 Beispiel für einen Potenzmengenautomaten . . . 9
1.8.2 Allgemeine Konstruktion eines NEA $\mathcal{A} \to$ DEA $\mathcal{B}$ . . . 10
1.9 Allgemeine Konstruktion von DEA $\to$ NEA . . . 11
1.10 Äquivalenz von Automaten . . . 11
1.10.1 Beispiel für ε-NEA $\to$ NEA: . . . 12
1.10.2 Kompensationsregel: . . . 13
1.10.3 Zusammenfassung: . . . 15
1.11 Verknüpfung von Automaten, reguläre Ausdrücke . . . 16
1.11.1 Verknüpfungsproblem (für NEA's) . . . 16
1.11.2 Produktautomat . . . 17

2 Reguläre Ausdrücke **18**
2.1 Automaten $\leftrightarrow$ Reguläre Ausdrücke . . . 19
2.1.1 $\mathsf{RA}_\Sigma \to$ Automaten . . . 20
2.1.2 Von NEA's zu regulärem Ausdruck: . . . 21
2.1.3 Umgang mit VNEAs: . . . 23
2.1.4 Eliminationsverfahren . . . 23
2.1.5 Von NEA's zu regulären Ausdrücke: Alternatives Verfahren . . . 25

3 Tragweite der regulären Sprachen **29**
3.0.6 Pumping Lemma . . . 30

4 Algorithmen über Automaten **33**
4.1 Algorithmische Probleme: . . . 33

5 Weitere Konstruktionen **37**
5.1 Automaten mit Ausgabe . . . 37
5.1.1 Eigenschaften verallgemeinert sequentieller Funktionen . . . 40

5.1.2 Allgemeine Konstruktion 41
5.1.3 Satz von Ginsburg und Rose 42
5.2 Endliche Zwei-Wege-Automaten 42

6 Grammatiken **44**
6.1 Vereinfachung kontextfreier Grammatiken 53
6.2 Ableitungsbäume 54

7 Kellerautomaten **63**

8 Unentscheidbarkeitsresultate **69**
8.1 Postsches Korrespondenzproblem 70

9 Erweiterte Sprachdefinition **76**
9.1 Turingmaschinen als Akzeptoren 76
9.2 Ergänzende Resultat zur Chomsky Hierarchie 80

1 Endliche Automaten

20.4.'04

1.1 Einführung

Mathematik:

- Zahlen
- geometrische Figuren
- Mengen & Funktionen

Informatik:

- Wörter
- Wortmengen
- Wortfunktionen

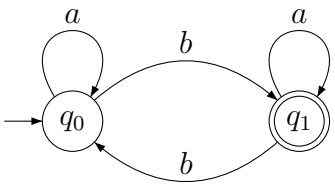


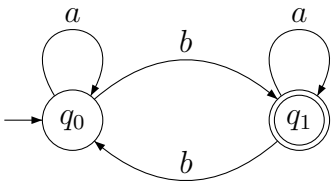
Abbildung 1: Ein kleiner endlicher Automat

Die akzeptierte Sprache des Automaten ist $L(A) = \{w \mid w \text{ hat ungeradzahlig viele } b\}$

1.2 Vier Typen endlicher Automaten

Motivation, Beispiele:

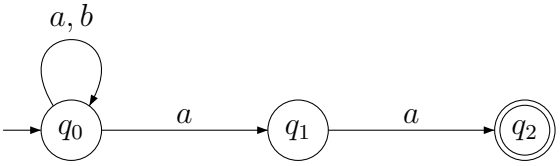
- **Deterministischer endlicher Automat (DEA)**
  $\mathcal{A}_1$:



Typische Vorkommen: Hardware (FlipFlops), Textverarbeitung (reguläre Sprachen)

- **Nichtdeterministischer endlicher Automat (NEA)**

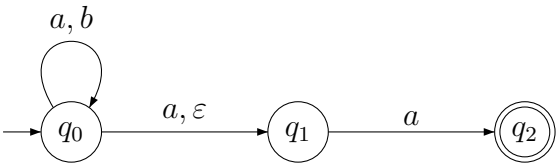
  $\mathcal{A}_2$:



  Standardanwendung: Systembeschreibung

- **NEA mit $\varepsilon-$Transitionen (ε-NEA's)**

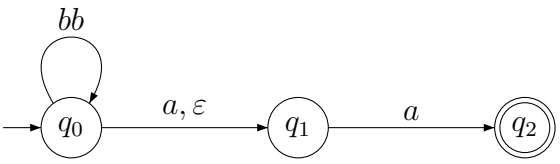
  $\mathcal{A}_3$:



  Standardanwendung: Syntaxdiagramme

- **NEA mit Worttransitionen**

  $\mathcal{A}_4$:



Die durch $\mathcal{A}_1$ akzeptierten Wörter sind die Beschriftungen der Wege vom Anfangs- in den Endzustand.

- $\mathcal{A}_1$: akzeptiert $w \Leftrightarrow w$ hat ungeradzahlig viele b
- $\mathcal{A}_2$: akzeptiert $w \Leftrightarrow w$ endet mit aa
- $\mathcal{A}_3$: akzeptiert $w \Leftrightarrow w$ endet mit a oder aa (zweiter Fall durch den ersten schon gegeben)
- $\mathcal{A}_4$: akzeptiert $w \Leftrightarrow w$ enthält eine gerade Anzahl von b's und endet mit a oder aa

Plan:

- genaue Einführung der vier Typen von Automaten
- Jeweils Festlegung der akzeptierten Sprachen
- Vergleiche

1.3 Definition des DEA

Ein DEA hat folgende Form:

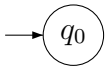
$\mathcal{A} = (Q,\ \Sigma,\ q_0,\ \delta,\ F)$ mit $q_0 \in Q$, $F \subseteq Q$, $\delta : Q \times E \to Q$

Dabei bezeichnet Q eine endliche Zustandsmenge, Σ das Eingabealphabet, q_0 den Anfangszustand und F die Endzustände.

Konventionen für die graphische Darstellung:

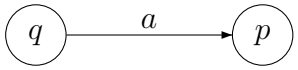
- Zustände durch Knoten dargestellt ($q_i \in Q$):

- Anfangszustand ($q_0 \in Q$):



- Endzustände werden umkreist ($q \in F$):

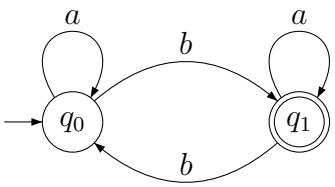
- Im Fall $\delta(q,\ a) = p$ füge einen Pfeil ein:



Erweitere δ zur Funktion $\delta^* : Q \times \Sigma^* \to Q$

$\delta^*(p[\in Q],\ w[\in \Sigma^*])$ = der Zustand, in den der Automat von p aus mit den Eingabebuchstaben von w (in der Reihenfolge gemäß w) gelangt.

Beispiel an einem DEA $\mathcal{A}$:



$$\begin{aligned} \delta^*(q_0, babab) &= q_1 \\ \delta^*(q_1, \varepsilon) &= q_1 \\ \delta^*(q_0, baba) &= q_0 \end{aligned}$$

1.4 Induktive Definition von δ^*

$\delta^*(p,\ \varepsilon) = p$
$\delta^*(p,\ wa) = \delta(\delta^*(p,\ w),\ a)$

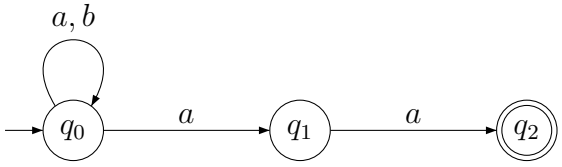
- $\mathcal{A}$ akzeptiert das Wort $w \Leftrightarrow \delta^*(q_0,\ w) \in F$
- $\delta^*(q_0,\ w) \in F$ bedeutet umgangssprachlich: „von q_0 aus gelangt $\mathcal{A}$ bei Eingabewort w in einen Endzustand“
- Die von $\mathcal{A}$ akzeptierte Sprache $L(\mathcal{A}) = \{w \in \Sigma^* \mid A \text{ akzeptiert } w\}$. Eine Sprache L heißt DEA-akzeptierbar, falls ein DEA $\mathcal{A}$ existiert mit $L = L(\mathcal{A})$.

1.5 Definition eines NEA

Ein NEA hat die Form:
$\mathcal{A} = (Q,\ \Sigma,\ q_0,\ \Delta,\ F)$ mit $Q,\ \Sigma,\ q_0,\ F$ wie bei einem DEA. Für das Δ gilt: $\Delta \subseteq Q \times \Sigma \times Q$.

$(p, w, q) \in \Delta$ besagt: „von p ist mit w ein Übergang nach q möglich“.

Beispiel am NEA $\mathcal{A}_2$:

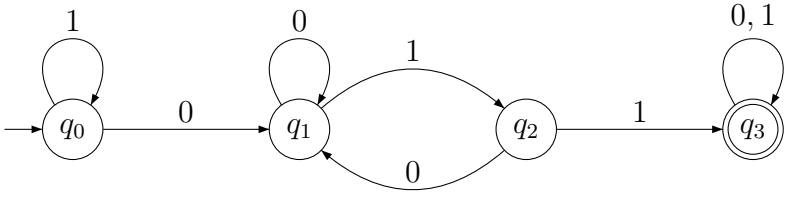


$Q = \{q_0,\ q_1,\ q_2\}$
$F = \{q_2\}$
$\Delta = \{(q_0,\ a,\ q_0),\ (q_0,\ b,\ q_0),\ (q_0,\ a,\ q_1),\ (q_1,\ a,\ q_2)\}$

1.5.1 Weitere Beispiele für DEA's und NEA's

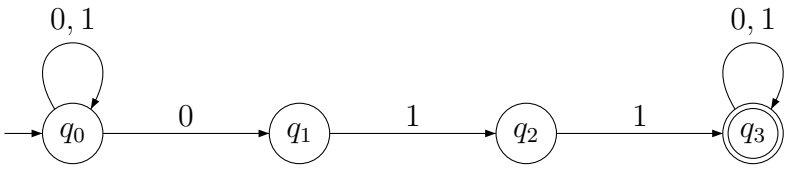
23.4.'04

- **DEA $\mathcal{A}_5$ ($\Sigma = \{0,\ 1\}$):**



$L(\mathcal{A}_5)$ = Menge der Bitwörter mit einem Infix 011

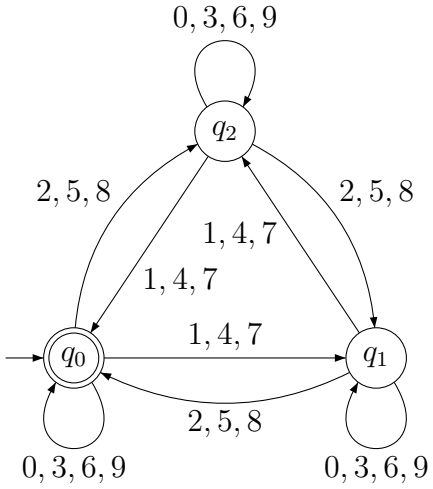
- **NEA $\mathcal{A}_6$ ($\Sigma = \{0,\ 1\}$):**



$L(\mathcal{A}_6) = L(\mathcal{A}_5)$ = Menge der Bitwörter mit einem Infix 011

- **DEA $\mathcal{A}_7$ ($\Sigma = \{0,\ \ldots,\ 9\}$):**

  Idee: Aufsummieren der Reste von 3



$L(\mathcal{A}_7)$ = Menge der Dezimalzahlen, die durch 3 teilbar sind

1.6 Ein Lauf auf einem Wort

Gegeben sei ein NEA $\mathcal{A} = (Q,\ \Sigma,\ q_0,\ \Delta,\ F)$ mit $(p,\ a,\ q) \in \Delta$. Ein Lauf auf einem Wort $w = a_1 a_2 \ldots a_n$ ($\mathcal{A} : p \xrightarrow{w} q$) ist eine Zustandsfolge $p_0,\ p_1,\ \ldots,\ p_n$.
$\mathcal{A}$ akzeptiert $w = a_1 \ldots a_n \Leftrightarrow$ es existiert ein Lauf von $\mathcal{A}$ auf w, der von q_0 zu einem Endzustand führt.

1.6.1 Beispiele für Läufe

Beispiel für Läufe auf dem NEA $\mathcal{A}_6$:

Läufe	0	1	1	0	1
L_1	q_0	q_0	q_0	q_0	q_0
L_2	q_0	q_1	q_2	q_3	q_3

Kurznotation: $\mathcal{A}$ akzeptiert $w \Leftrightarrow \mathcal{A} : q_0 \xrightarrow{w} q$ mit $q \in F$

1.7 Erreichbarkeitsmenge

Definition: Erreichbarkeitsmenge von $\mathcal{A}$ mit w
$E_{\mathcal{A}}(w) = \{q \in Q \mid \mathcal{A} : q_0 \xrightarrow{w} q\} \Leftrightarrow$ Menge der von q über w erreichbaren Zustände

1.7.1 Beispiele zur Erreichbarkeitsmenge

Beispiele für den NEA $\mathcal{A}_6$:

- $E_{\mathcal{A}_6}(\varepsilon) = \{q_0\}$
- $E_{\mathcal{A}_6}(0) = \{q_0,\ q_1\}$
- $E_{\mathcal{A}_6}(01) = \{q_0,\ q_2\}$
- $E_{\mathcal{A}_6}(01101) = \{q_0,\ q_3,\ q_2\}$

Bemerkung: A akzeptiert $w \Leftrightarrow E_{\mathcal{A}}(w)$ enthält einen Endzustand $\Leftrightarrow E_{\mathcal{A}}(w) \cap F \neq \emptyset$.

1.7.2 Update der Erreichbarkeitsmenge

Gegeben $E_{\mathcal{A}}(w) = P$ und ein $a \in \Sigma$, dann ist $E_{\mathcal{A}}(wa) = \{q \in Q \mid \exists\ p \in P,\ (p, a, q) \in \Delta\}$.

1.8 Äquivalenz von NEA und DEA

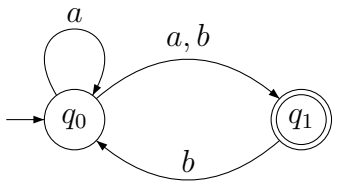
27.4.'04

Satz:
Zu jedem NEA $\mathcal{A}$ kann man einen äquivalenten DEA $\mathcal{B}$ angeben. $\mathcal{A}$ und $\mathcal{B}$ heißen äquivalent, falls gilt $L(\mathcal{A}) = L(\mathcal{B})$.

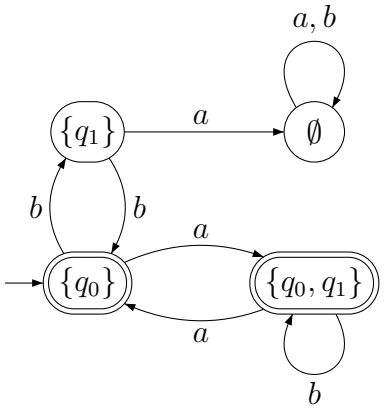
Beweis:
Gegeben ist ein NEA $\mathcal{A} = (Q,\ \Sigma,\ q_0,\ \Delta,\ F)$
Gesucht ist ein DEA $\mathcal{B} = (Q',\ \Sigma,\ q_0',\ \delta',\ F')$

1.8.1 Beispiel für einen Potenzmengenautomaten

NEA $\mathcal{A}$ ($\Sigma = \{a,\ b\}$):



Dazu der Potenzmengenautomat $\mathcal{B}$ ($\Sigma = \{a,\ b\}$):



1.8.2 Allgemeine Konstruktion eines NEA $\mathcal{A}$ → DEA $\mathcal{B}$

Konstruktion von $\mathcal{B}$ allgemein:
Sei $\mathcal{A} = (Q,\ \Sigma,\ q_0,\ \Delta,\ F)$ ein NEA

Definiere DEA $\mathcal{B}$ wie folgt:
$\mathcal{B} = (Q',\ \Sigma,\ q_0',\ \delta',\ F')$ mit

- $Q' = \mathcal{P}(Q)$ [der Potenzmenge von Q]
- $q_0' = \{q_0\}$
- $\delta'(P,\ a) = \{q \in Q \mid \exists p \in P,\ (p, a, q) \in \Delta\}$
- $F' = \{P \subset Q \mid P \cap F \neq \emptyset\}$

$\mathcal{B}$ ist äquivalent zu $\mathcal{A}$, d.h. für alle $w \in L(\mathcal{A})$ gilt:

$$\begin{aligned} \delta'^*(\{q_0\},\ w) \in F' &\Leftrightarrow \delta'^*(\{q_0\},\ w) \cap F \neq \emptyset \quad (\text{da } \mathcal{A} \text{ das Wort } w \text{ akzeptiert}) \\ &\Leftrightarrow \mathcal{A} \text{ akzeptiert } w \\ &\Leftrightarrow w \in L(\mathcal{A}) \end{aligned}$$

Bemerkung zur Konstruktion:

- Es genügt, sich auf $P \subseteq Q$ zu beschränken, die man ausgehend von $\{q_0\}$ über Wörter w erreichen kann.
- Es gibt (über $\Sigma = \{a,\ b\}$) eine Familie von Sprachen L_n mit:
  - L_n lässt sich von NEA mit $n + 1$ Zuständen akzeptieren
  - Jeder DEA, der L_n akzeptiert, hat $\geq 2^n$ Zustände

Bemerkung:
$\delta'^*(\{q\},\ w)$ = Menge der in $\mathcal{A}$ von q aus mit w erreichbaren Zustände

Beweis durch Induktion über den Aufbau von w:

Induktions-Anfang:

$w = \varepsilon$

$\delta'^*(\{q\},\ \varepsilon) = \{q\}$

Induktions-Schritt: $v = wa$

Induktionsvoraussetzung: Behauptung für w: $\delta'^*(\{q\},\ \varepsilon) = \{q\}$

$\delta'^*(\{q\},\ v)$ = Menge der von $\{q\}$ aus mit wa erreichbaren Zustände

1.9 Allgemeine Konstruktion von DEA → NEA

Gegeben DEA $\mathcal{A} = (Q,\ \Sigma,\ q_0,\ \delta,\ F)$. Wähle NEA $\mathcal{B} = (Q,\ \Sigma,\ q_0,\ \Delta,\ F)$, wobei:

$$\Delta := \{(p, a, q) \mid \delta(p,\ a) = q\}$$

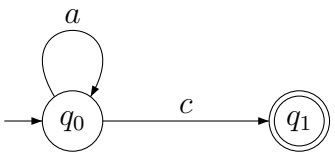
1.10 Äquivalenz von NEA, ε-NEA und NEA mit Worttransitionen

Erinnerung:
In einem ε-NEA sind Transitionen $(p,\ \varepsilon,\ q)$ erlaubt $\Rightarrow \Delta \subset Q \times (\Sigma \cup \{\varepsilon\}) \times Q$

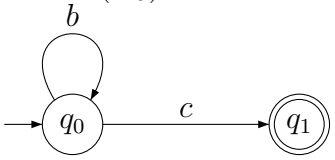
Motivation:
ε-Transitionen helfen bei der Strukturierung von Automaten

Beispiel:
$L_1 = \{w \mid w$ beginnt mit (eventuell leerer) Folge von a, anschließend folgt $c\}$
$L_2 = \{w \mid w$ beginnt mit (eventuell leerer) Folge von b, anschließend folgt $c\}$

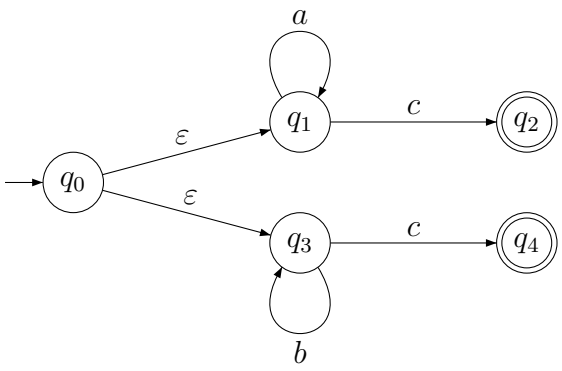
NEA $\mathcal{A}_8$ $(\Sigma = \{a,\ b,\ c\})$ mit$L_1 = L(\mathcal{A}_8)$:



NEA $\mathcal{A}_9$ $(\Sigma = \{a,\ b,\ c\})$ mit $L_2 = L(\mathcal{A}_9)$:



ε**-NEA** $\mathcal{A}_{10}$ $(\Sigma = \{a,\ b,\ c\})$ mit $L_3 = L(\mathcal{A}_8) \cup L(\mathcal{A}_9) = L(\mathcal{A}_{10})$:



Motivation:

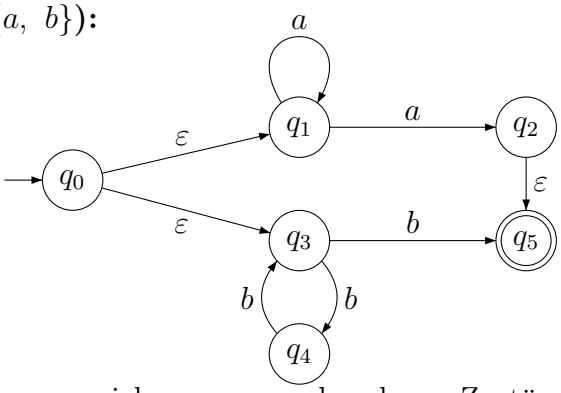
- Praktische Verwendung in Syntaxdiagrammen
- Modellierung paralleler Prozesse:

  Unterscheidung zwischen sichtbaren und „stillen“ Aktionen.

Satz:
Zu jedem ε-NEA $\mathcal{A}$ kann man einen NEA $\mathcal{B}$ mit $L(\mathcal{A}) = L(\mathcal{B})$ konstruieren.

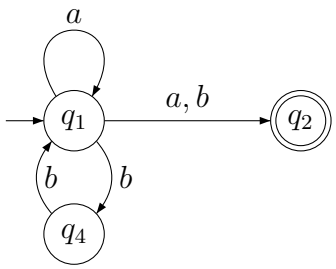
1.10.1 Beispiel für ε-NEA $\rightarrow$ NEA:

ε-NEA: $\mathcal{A}$ ($\Sigma = \{a,\ b\}$):



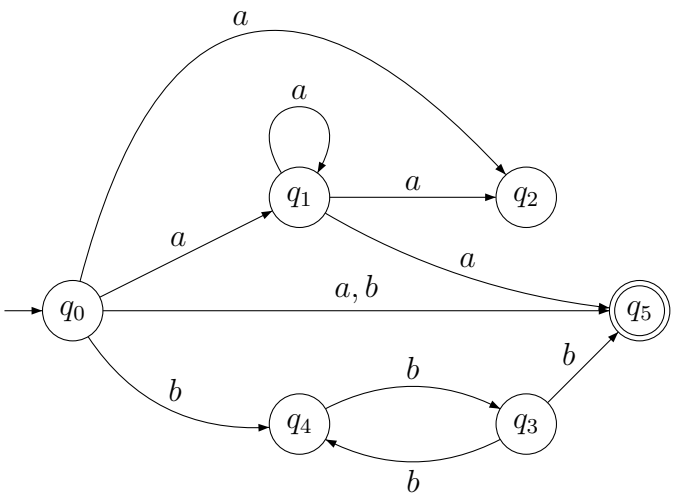
1. **Ansatz:** Zusammenziehen von ε-verbundenen Zuständen

   ε-NEA $\mathcal{A}$ ($\Sigma = \{a,\ b\}$) konstruiert durch Zusammenziehen der ε-Kanten:[**Automat Falsch!!**]



2. **Ansatz:** Streichen der ε-Transitionen und Kompensation durch neue Buchstaben-Transitionen

   Zu $\mathcal{A}$ passender NEA $\mathcal{B}$ ($\Sigma = \{a,\ b\}$):



1.10.2 Kompensationsregel:

Gibt es im gegebenen ε-NEA einen Pfad $p \xrightarrow{\varepsilon} q \xrightarrow{a} q' \xrightarrow{\varepsilon} r$, dann füge im neuen NEA die Transition $(p,\ a,\ r)$ hinzu.

Sonderfälle: $p = q$, $q' = r$

Beweis:

1. Etappe: Definition des NEA $\mathcal{B}$
2. Etappe: Korrektheit
3. Etappe: Algorithmische Berechnung von $\mathcal{B}$

Erste Etappe: Definition des NEA

Gegeben: ε-NEA $\mathcal{A} = (Q,\ \Sigma,\ q_0,\ \Delta,\ F)$, dann definiere NEA $\mathcal{B} = (Q,\ \Sigma,\ q_0,\ \Delta',\ F')$ mit:

$$\Delta' = \{(p,\ a,\ r) \mid \exists\ q, q' \in Q,\ (q,\ a,\ q') \in \Delta,\ \mathcal{A} : p \xrightarrow{\varepsilon} q,\ \mathcal{A} : q' \xrightarrow{\varepsilon} r\}$$

$$F' = \begin{cases} F \cup \{q_0\} & \text{falls } \mathcal{A} : q_0 \xrightarrow{\varepsilon} q,\ q \in F \\ F & \text{sonst} \end{cases}$$

Dann:

(a) $\mathcal{A} : q_0 \xrightarrow{\varepsilon} q,\ q \in F \Leftrightarrow q_0 \in F'$

(b) $\mathcal{A} : p \xrightarrow{a} q \Leftrightarrow (p,\ a,\ q) \in \Delta'$

Zweite Etappe: Zeige: $\mathcal{A}$ akzeptiert $w \Leftrightarrow \mathcal{B}$ akzeptiert w

Fall (a): $w = \varepsilon$:

$$\begin{aligned} \mathcal{A} \text{ akzeptiert } w &\Leftrightarrow \mathcal{A}\text{: } q_0 \xrightarrow{\varepsilon} q,\ q \in F \\ &\Leftrightarrow q_0 \in F' \\ &\Leftrightarrow \mathcal{B} \text{ akzeptiert } w \end{aligned}$$

Fall (b): $w = a_1 \dots a_n$ $(n \geq 1)$

$$\begin{aligned} \mathcal{A} \text{ akzeptiert } w &\Leftrightarrow \mathcal{A}\text{: } q_0 \xrightarrow{w} q,\ q \in F \\ &\Leftrightarrow \text{es existieren Zustände } p_1,\ \dots,\ p_{n-1} \text{ mit } \mathcal{A} : q_0 \xrightarrow{a_1} p_1, \\ &\quad\ \mathcal{A} : p_1 \xrightarrow{a_2} p_2, \dots, \mathcal{A} : p_{n-1} \xrightarrow{a_n} q,\ q \in F \\ &\Leftrightarrow \text{nach Konstruktion von } \Delta' \text{ existieren Transitionen} \\ &\quad\ (q_0,\ a_1,\ p_1), (p_1,\ a_2,\ p_2), \dots, (p_{n-1},\ a_n,\ q) \in \Delta' \\ &\Leftrightarrow \mathcal{B} \text{ akzeptiert } a_1 \dots a_n = w \text{ (da } F \subseteq F') \end{aligned}$$

(Für die Rückrichtung benutze, dass $F' = F \cup \{q_0\}$ und Zustand $q \neq q_0$)

Dritte Etappe: Algorithmische Berechnung

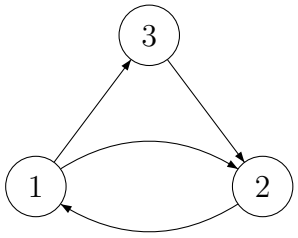
Zur algorithmischen Bestimmung von Δ', F' genügt ein Verfahren, dass zu $p,\ q$ jeweils $\mathcal{A} : p \xrightarrow{\varepsilon} q$ entscheidet, d.h. ob in $\mathcal{A}$ ein ε-Pfad von p nach q existiert.

Methode: streiche alle Buchstaben-Transitionen aus $\mathcal{A}$, somit entsteht der Transitionsgraph $G(V,\ E)$ allein mit den ε-Transitionen.

Aufgabe: Bestimmung von $(p,\ q)$, so dass ε-Pfad von p nach q existiert.

Verfahren: (Floyd-Warshall)

Beispiel:



$$\text{Adjazenzmatrix } A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Gesucht ist eine Matrix A^* mit $a^*_{ij} = 1 \Leftrightarrow$ in G existiert Pfad von i nach j.

Berechne für $k = 0, \dots, n$ (über die Knoten $1, \dots, n$) die Adjazenzmatrix des Graphen G_k mit einer Kante von i nach j, falls in G ein Pfad von i nach j existiert mit höchstens den Zwischenknoten $1, \dots, k$.

Für $k = 0$:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

Automatentypen:

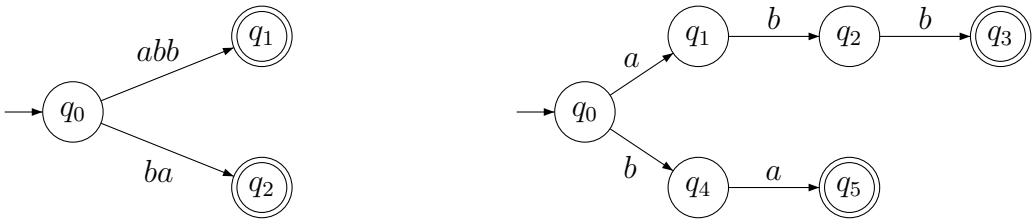
1. DEA
2. NEA
3. ε-NEA
4. NEA mit Worttransitionen

Gezeigt:

- Jeder NEA ist äquivalent zu einem DEA (Potenzmengenkonstruktion)
- Jeder ε-NEA ist äquivalent zu einem NEA (Elimination der ε-Transitionen)

Definition:
Ein NEA mit Worttransitionen hat die Form $\mathcal{A} = (Q,\ \Sigma,\ q_0,\ \Delta,\ F)$, wobei Q, Σ, q_0, F wie bei NEA's und $\Delta \subseteq Q \times \Sigma^* \times Q$ ist endlich. Die Transition hat die Form $(p,\ u,\ q)$. Akzeptieren analog zum NEA.

Triviales Beispiel:
$L = \{abb,\ ba\}$ akzeptiert durch einen NEA mit Worttransitionen bzw. einen DEA



Bemerkung:
Jeder NEA mit Worttransitionen ist äquivalent zu einem ε-NEA.

Allgemeine Konstruktion:
Ersetze jede Worttransition $(p,\ a_1 \dots a_n,\ q)$ mit $n \geq 2$ durch die Transitionen (p, a_1, p_1), (p_1, a_2, p_2), $\dots$, $(p_{n-1},\ a_n,\ q)$ mit jeweils neuen Hilfszuständen $p_1, \dots, p_{n-1}$. Daraus ergibt sich ein zu einem gegebenen NEA mit Worttransitionen äquivalenter ε-NEA.

1.10.3 Zusammenfassung:

Mit den Automatentypen DEA, NEA, ε-NEA, NEA mit Worttransitionen lassen sich die gleichen Sprachen definieren, und man kann algorithmisch von einem Automaten eines Typs zu einem Äquivalenten eines anderen Typs übergehen.

1.11 Verknüpfung von Automaten, reguläre Ausdrücke

Vorbereitung: Sprachoperationen für Sprachen über Σ

1. **Boolsche Operationen:**

   $\cup$, $\cap$, $\overline{L} := \Sigma^* - L$

2. **Verkettung (Konkatenation):**

   $L_1 \cdot L_2 = \{w \in \Sigma^* \mid w \text{ zerlegbar als } w = uv \text{ mit } u \in L_1,\ v \in L_2\}$

3. **Iteration (Kleene-Stern)**

   $L^* = \{w \in \Sigma^* \mid w = \varepsilon \text{ oder } w \text{ zerlegbar als } w = u_1 \dots u_n \text{ und } u_1, \dots, u_n \in L\}$

1.11.1 Verknüpfungsproblem (für NEA's)

Gegeben 2 NEAs $\mathcal{A}_1$, $\mathcal{A}_2$ mit $L_1 = L(\mathcal{A}_1)$ und $L_2 = L(\mathcal{A}_2)$ und Sprachoperation op.

Finde NEA $\mathcal{A}$, so dass $L(\mathcal{A}) = L_1 \ \mathsf{op}\ L_2$.

Zu den Boolschen Operationen:

Satz:

Zu DEA's $\mathcal{A}_1$, $\mathcal{A}_2$ über Σ kann man konstruieren:

- einen DEA $\mathcal{A}'$ mit $L(\mathcal{A}') = \Sigma^* \setminus L(\mathcal{A}_1)$ (Komplement)
- einen DEA $\mathcal{A}''$ mit $L(\mathcal{A}'') = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ (Schnitt)

Zum Komplement:

Gegeben $\mathcal{A}_1 = (Q,\ \Sigma,\ q_0,\ \delta,\ F)$. Definiere $\mathcal{A}' = (Q,\ \Sigma,\ q_0,\ \delta,\ Q \setminus F)$.

Dann $w \in L(\mathcal{A}') \Leftrightarrow \delta^*(q_0,\ w) \in Q \setminus F \Leftrightarrow \delta^*(q_0,\ w) \notin F \Leftrightarrow \mathcal{A}_1$ akzeptiert w nicht.

Zum Durchschnitt: Idee anhand Beispiel:

Abbildung 2: Der Automat $\mathcal{A}_1$

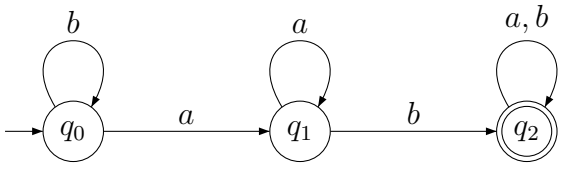
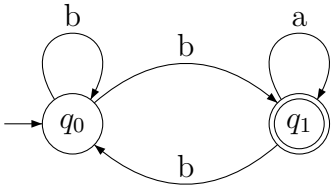
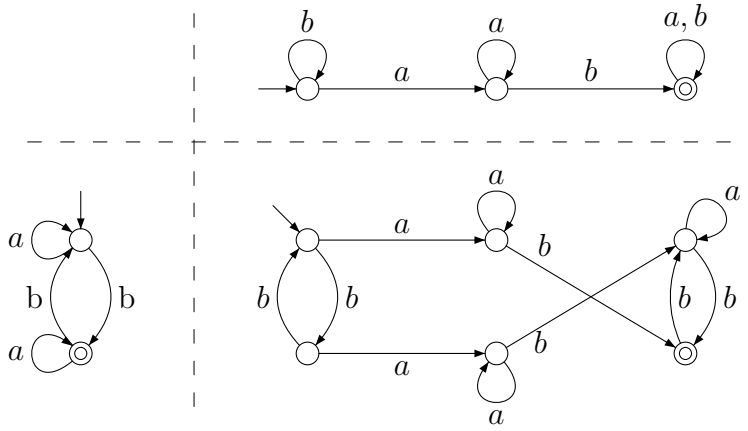


Abbildung 3: Der Automat $\mathcal{A}_2$



1.11.2 Produktautomat



Ziel:
$L_1,\ L_2 \subset \Sigma^*$ sind DEA-akzeptierbar $\Rightarrow L_1 \cap L_2$ DEA-akzeptierbar

Gegeben: 2 DEA's:
$\mathcal{A}_1 = (Q_1,\ \Sigma,\ q_{01},\ \delta_1,\ F_1)$ akzeptiert L_1 und $\mathcal{A}_2 = (Q_2,\ \Sigma,\ q_{02},\ \delta_2,\ F_2)$ akzeptiert L_2

Gesucht:
Ein DEA $\mathcal{A}$, der $L_1 \cap L_2$ akzeptiert. Definiere $\mathcal{A}$ als Produktautomaten von $\mathcal{A}_1$ und $\mathcal{A}_2$.
$\mathcal{A} = (Q_1 \times Q_2,\ \Sigma,\ (q_{01},\ q_{02}),\ \delta,\ F)$ mit $\delta((p,\ q),\ a) = (\delta_1(p,\ a),\ \delta_2(q,\ a))$ und $F = F_1 \times F_2$.

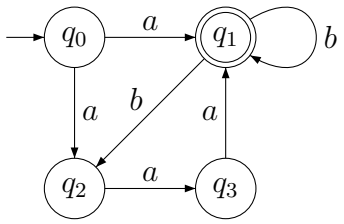
Dann gilt $\delta^*((q_{01},\ q_{02}),\ w) = (\delta^*(q_{01},\ w),\ \delta^*(q_{02},\ w)) \in F \Leftrightarrow F = F_1 \times F_2$. Also $\mathcal{A}$ akzeptiert $w \Leftrightarrow \delta_1^*(q_{01}, w) \in F_1$, $\delta_2^*(q_{02},\ w) \in F_2 \Leftrightarrow \mathcal{A}_1$ akzeptiert w und $\mathcal{A}_2$ akzeptiert w.

Analoge Konstruktion für NEA's

$\mathcal{A}_1 = (Q_1,\ \Sigma,\ q_{01},\ \Delta_1,\ F_1)$
$\mathcal{A}_2 = (Q_2,\ \Sigma,\ q_{02},\ \Delta_2,\ F_2)$
$\mathcal{A} := (Q_1 \times Q_2,\ \Sigma,\ (q_{01},\ q_{02}),\ \Delta,\ F = (F_1 \times F_2))$ mit
$((p_1,\ p_2),\ a,\ (q_1,\ q_2)) \in \Delta \Leftrightarrow (p_1,\ a,\ q_1) \in \Delta_1,\ (p_2,\ a,\ q_2) \in \Delta_2$

2 Reguläre Ausdrücke

Abbildung 4: Beispielautomat $\mathcal{A}$



Regulärer Ausdruck von $\mathcal{A}$: $(a + aaa)(b^*baa)^*b^*$

$\mathcal{A}$ akzeptiert genau dann wenn:
w beginnt mit einem oder mit drei a, gefolgt von eventuell leerer Folge von Blöcken, jeweils aus mindestens einem b und dann zwei a's (aa) und abgeschlossen durch einen eventuell leeren Block von b's.

Definition (Syntax der regulären Ausdrücke):
Die regulären Ausdrücke über Σ entstehen aus den atomaren Ausdrücken $\emptyset$, ε, a $(a \in \Sigma)$ mit den Operationssymbolen $+$, $\cdot$ (zweistellig) und * (einstellig).

Klammerung wie üblich:

- * bindet stärker als $+$
- $\cdot$ bindet stärker als $+$
- $ab^* + b == (a \cdot (b^*)) + b$
- $\cdot$ kann wegfallen

Definition: (Semantik der regulären Ausdrücke)
Zum regulären Ausdruck r wird $L(r)$, die durch r definierte Sprache festgelegt:

- $L(\emptyset) =$ leere Sprache
- $L(\varepsilon) = \{\varepsilon\}$
- $L(a) = \{a\}$
- $L(r + s) = L(r) \cup L(s)$
- $L(r \cdot s) = L(r) \cdot L(s)$
- $L(r^*) = (L(r))^*$

Erlaubte Notation $w \in r$ steht für $w \in L(r)$ z.B. $w \in ba^*$

r und s heißen äquivalent, falls $L(r) = L(s)$. Notation $r = s$.

Bemerkung 1:

$r + s \simeq s + r$ (klar, denn $L(r) \cup L(s) = L(s) \cup L(r)$)

$r \cdot s \not\simeq s \cdot r$

(Gegen-)Beispiel: $r = a^*, s = b^*$

Spezialfall: Mit $r = \varepsilon = s$ ist $\cdot$ ausnahmsweise kommutativ

Bemerkung 2:

$(a + b)^*(a^*b)^*a^*)$ (WTF ist das hier??)

Zeige ganz allgemein: $(K \cup L)^* = (K^*L)^*K^*$

$w \in (K \cup L)^*$ ist zerlegbar als $w = w_1 \dots w_n$ mit $w_i \in K \cup L$

Fall 1: Kein $w_i \in L$. Dann $w \in K^*$, also auch aus $(K^*L)^* \cdot K^*$

Fall 2: $w_i \in L$:

Gruppiere die Segmente w_i jeweils bis zum nächsten $w_k \in L$. Erhalte Blöcke in $K^* \cdot L$. Nach letzten Block folgen eventuell noch K-Segmente. Also $w \in (K^*L)^* \cdot K^*$.

2.1 Automaten ↔ Reguläre Ausdrücke

Die Menge RA_Σ der reguläre Ausdrücke über Σ entsteht aus $\emptyset$, ε, a ($a \in \Sigma$) mit den Operationen + (Vereinigung), $\cdot$ (Verkettung), * (Iteration (Kleene-Stern)).

$$r \to L(r)$$

Satz von Kleene:
Zu jedem regulären Ausdruck kann man einen äquivalenten NEA konstruieren, und zu jedem NEA kann man einen äquivalenten regulären Ausdruck konstruieren.

Eine durch einen regulären Ausdruck definierbare Sprache heißt **regulär**.

„regulär“ bedeutet:

- durch regulären Ausdruck definierbar
- DEA-akzeptierbar
- NEA-akzeptierbar

Beweis:

Beweismethode:

Induktion über den Aufbau der regulären Ausdrücke.

Teilaufgaben:

Finde Automaten für $\emptyset,\ \varepsilon,\ a$ (Induktionsanfang).

Induktionsschritt:

Induktions-Voraussetzung:
Gegeben Automaten $\mathcal{A}_1,\ \mathcal{B}_2$ für $r_1,\ r_2$

Induktions-Bedingung:
finde Automaten $\mathcal{A}_+,\ \mathcal{A}_\bullet,\ \mathcal{A}_*$

- für $r_1 + r_2$
- bzw. $r_1 \cdot r_2$
- bzw. r_1^*

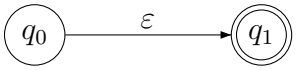
Wir arbeiten mit ε-NEA's, die zudem genau einen Endzustand (verschieden vom Anfangszustand) aufweisen.

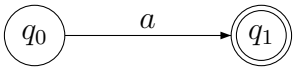
2.1.1 $\mathsf{RA}_\Sigma \rightarrow$ Automaten

11.5.'04

Induktionsanfang

- $\emptyset$: 

- ε: 

- **a:** 

Induktionsschritt:
Gegeben seien $\mathcal{A}_1$ und $\mathcal{A}_2$ als ε-NEA's mit genau einem Endzustand $L(\mathcal{A}_1) = L(r_1)$ bzw. $L(\mathcal{A}_2) = L(r_2)$.
$L(A_*) = L(r_1^*) \overset{I.V.}{=} (L(A_1))^*$ [siehe auch Folie im Netz]

Abbildung 5: A_+ für $r_1 + r_2$ (Sprache $L(r_1) \cup L(r_2)$)

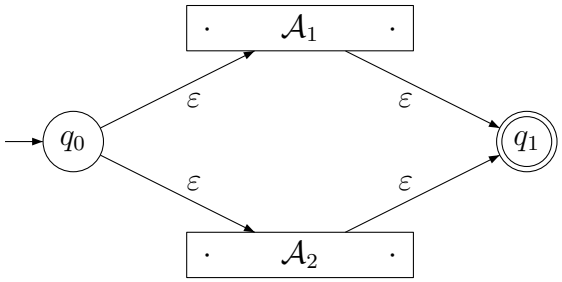
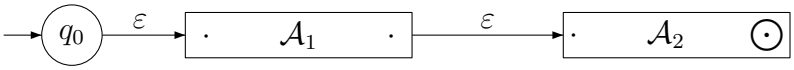
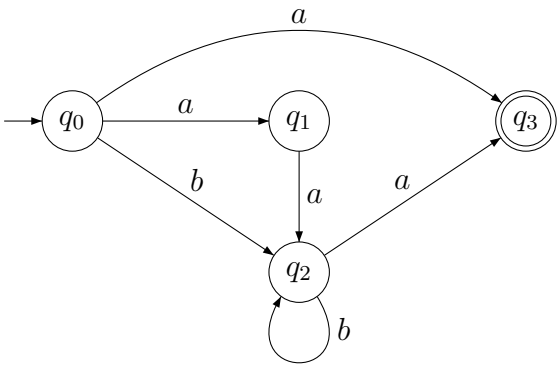


Abbildung 6: $A_\bullet$ für $r_1 \cdot r_2$

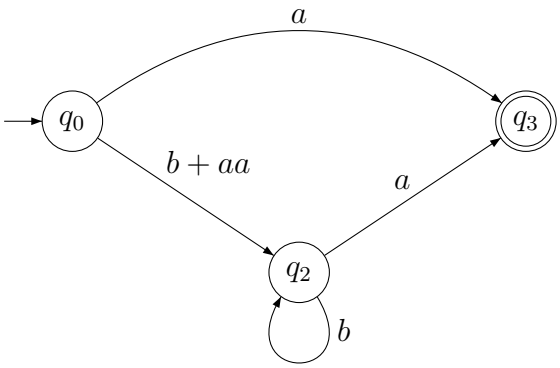


2.1.2 Von NEA's zu regulärem Ausdruck:

Beispiel:



Eliminiere q_1, korrigiere durch Neubeschriftung der Transitionen $q_1 \to q_2$



Eliminiere q_2

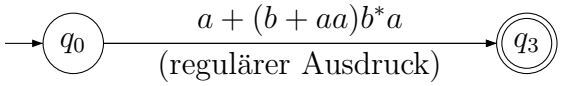
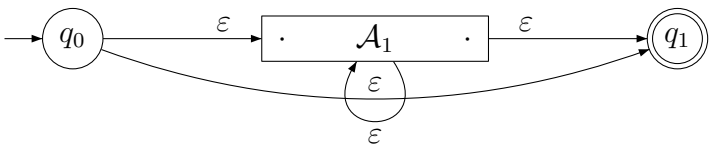


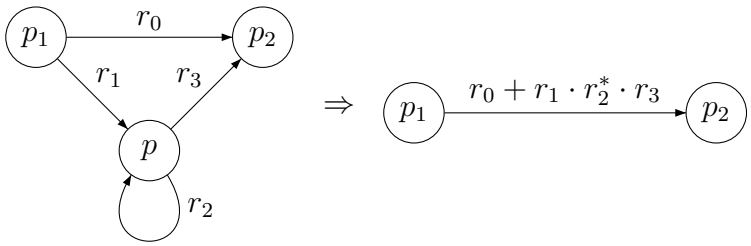
Abbildung 7: A_* für r_1^*



Idee: Eliminiere Zustände, nutze reguläre Ausdrücke als Transitionsbeschriftungen.

Allgemeines Verfahren

1. Führe „verallgemeinerte NEA's" ein (VNEA's) mit
   - Anfangszustand q_{start} und Endzustand q_{end}
   - Transitionsbeschriftung durch reguläre Ausdrücke
2. Eliminiere Zustände $p \neq q_{\text{start}}, q_{\text{end}}$ gemäß Regel:



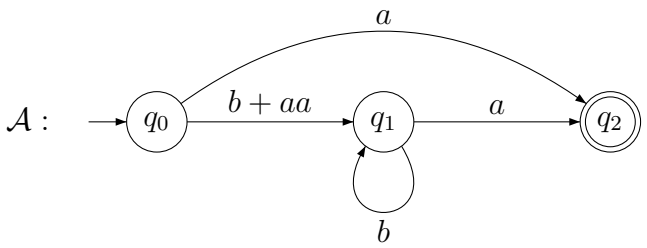
Definition:
Ein verallgemeinerter NEA über Σ (VNEA) hat die Form $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ mit

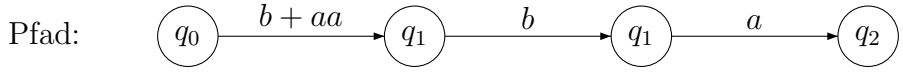
- $q_0 = q_{\text{start}}$
- $F = \{q_{\text{end}}\}$
- $\Delta \subseteq Q \times \mathsf{RA}_\Sigma \times Q$ (p, r, q) mit regulärem Ausdruck r.

Eine Transitionsfolge $(p_0, r_1, p_1) \ldots (p_{k-1}, k_k, p_k)$ hat Beschriftung $r_1, \ldots, r_k$

Definiere Wortmenge: $L(r_1 \cdot \ldots \cdot r_k)$
$L(\mathcal{A}) =$ Menge der Wörter, die sich als Vereinigung der Wortmengen zu den Pfaden von q_{start} zu q_{end} ergibt.

Beispiel:





$L(\mathcal{A}) = a + (b + aa)b^*a$

2.1.3 Umgang mit VNEAs:

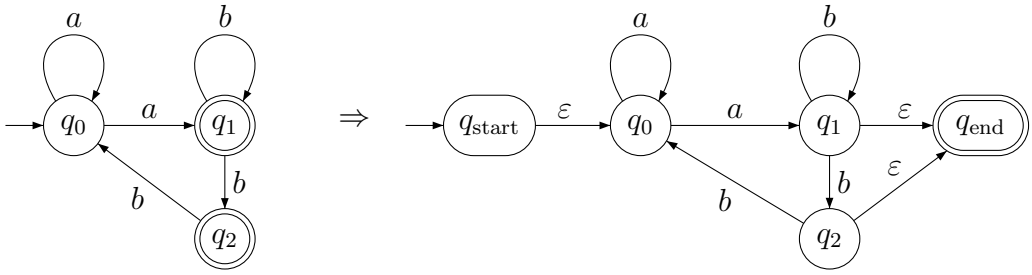
Unterstelle jeweils eine einzige Transition von p nach q

- Fasse mehrere (mit Beschriftungen $r_1, \ldots, r_k$) zusammen (Beschriftung $r_1 + \ldots + r_k$)

- Fehlende Transitionen (von p nach q) denkt man sich, ergänzt mit Beschriftung $\emptyset$

2.1.4 Eliminationsverfahren

1. Von NEA gehe zu VNEA mit neuen q_{start}, q_{end} über

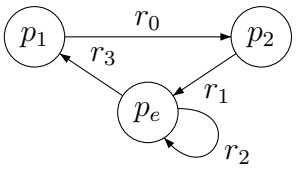


Allgemein:

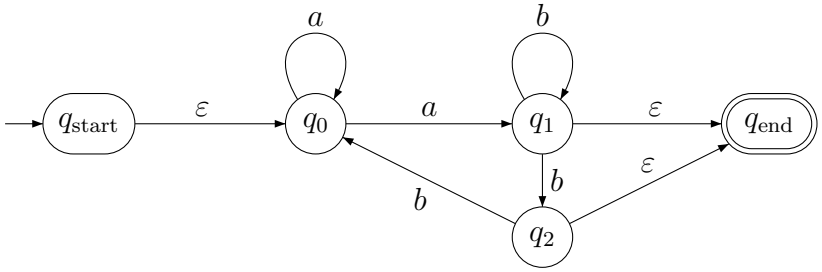
Von $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ gehe über zu $\mathcal{B} = (Q \cup \{q_{\text{start}}, q_{\text{end}}\}, \Sigma, q_{\text{start}}, \Delta', q_{\text{end}})$ mit $\Delta' = \Delta \cup \{(q_{\text{start}}, \varepsilon, q_0)\} \cup \{(q, \varepsilon, q_{\text{end}}) \mid q \in F\}$

2. Solange der vorliegende VNEA noch einen Q-Zustand hat

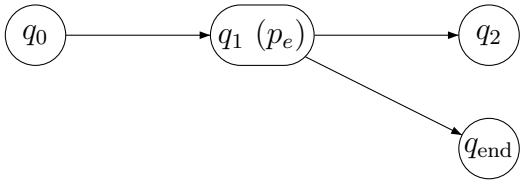
   - wähle zu eliminierenden Zustand $p_e \in Q$ aus
   - für alle Paare (p_1, p_2) mit Transitionen von p_1 nach p_e und von p_e nach p_2
     - bestimme Beschriftungen von
     - streiche p_e und p_e-Transitionen
     - ersetze r_0 durch $r_0 + r_1 r_2^* r_3$

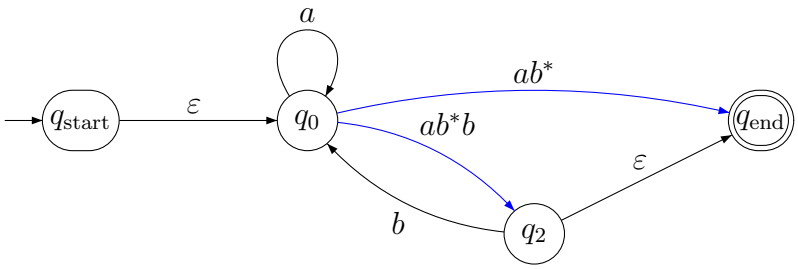


Beispiel:

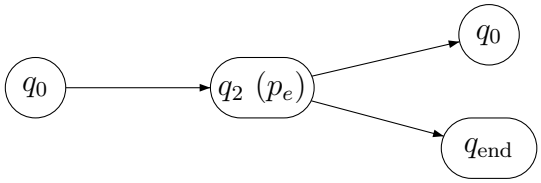


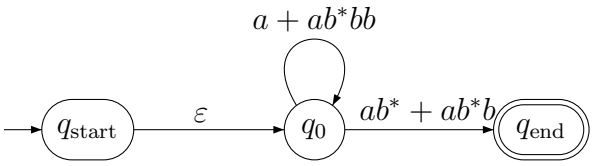
Eliminiere q_1:
zu betrachtende Situation:



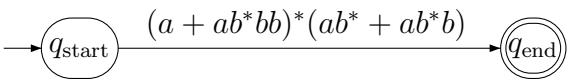


Eliminiere q_2:
zu beachtende Situation:





Eliminiere q_0:



Korrektheitsaussage

1. Der aus dem NEA konstruierte VNEA mit q_{start} und q_{end} ist äquivalent zum gegebenen NEA (trivial)

2. Der Eliminationsalgorithmus terminiert und liefert VNEA (als Endergebnis) der äquivalent zum Ausgangs-VNEA ist.

   Nötig: **Lemma:**

   Ein Eliminationsschritt von $\mathcal{A}'$ nach $\mathcal{A}''$ erfüllt $L(\mathcal{A}') = L(\mathcal{A}'')$

Zeitaufwand:
$|Q|$ Iterationen, $|Q|^2$ Zustandspaare $\Rightarrow O(|Q|^3)$

Satz von Kleene:
Eine Sprache ist NEA-akzeptierbar genau dann, wenn sie durch einen regulären Ausdruck definierbar ist.

2.1.5 Von NEA's zu regulären Ausdrücke: Alternatives Verfahren

Kleene-Floyd-Warshall bei Automaten: McNaughton, Yamada

Idee: NEA $\mathcal{A} = (Q,\ \Sigma,\ q_0,\ \Delta,\ F)$ mit $Q = \{1, \ldots, n\}$ und $q_0 = F = \{j_1, \ldots, j_m\}$.

$\mathcal{A}_{ij} = (Q, \Sigma, i, \Delta, j)$
$L_{ij} = L(\mathcal{A}_{ij})$ = Menge der Beschriftungen der Pfade von i nach j

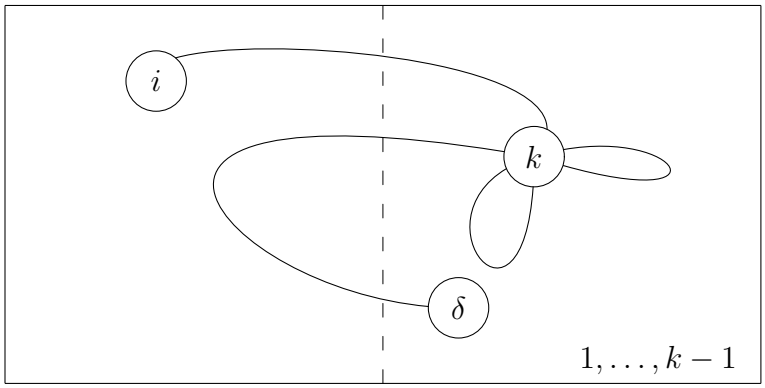
Bemerkung:
$L(\mathcal{A}) = \bigcup j \in F$ d.h. $L_{1j} = L_{1j_1} \cup \ldots \cup L_{1j_m}$

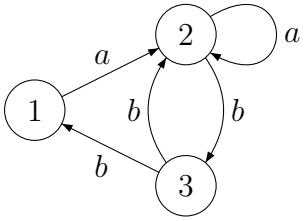
Definition:
Sei $k = 0, \ldots, n$, dann ist L_{ij}^k = Menge der Beschriftungen der Pfade von i nach j mit Zwischenzuständen höchstens in $\{1, \ldots, k\}$. $[L_{ij}^n = L_{ij}]$

Ansatz:
Für $k = 0, \ldots, n$ reguläre Ausdrücke r_{ij}^k für L_{ij}^k. Dann $L(\mathcal{A})$ definiert durch $r_{1j_1}^n + \ldots + r_{1j_m}^n$.
Bestimmung von r_{ij}^0: (Fall: keine Zwischenknoten)

$$(*)\ r_{ij}^0 = \begin{cases} \varepsilon & \text{falls } i = j,\ \text{keine Transition } (i,\ a,\ i = j) \in \Delta \\ \varepsilon + a_1 + \ldots + a_m & \text{falls } i = j,\ \text{Transitionen } (i,\ a_1,\ i),\ \ldots,\ (i, a_m, i) \in \Delta \\ \emptyset & \text{falls } i \neq j,\ \text{keine Transition } (i,\ a,\ j) \in \Delta \\ a_1 + \ldots + a_m & \text{falls } i \neq j,\ \text{Transition } (i,\ a_1,\ j),\ \ldots,\ (i,\ a_m,\ j) \in \Delta \end{cases}$$

$(**)\ k > 1$: $r_{ij}^k = r_{ij}^{k-1} + r_{ik}^{k-1} \cdot (r_{kk}^{k-1})^* \cdot r_{kj}^{k-1}$



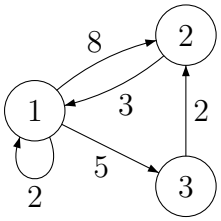


Analyse Matrix $\binom{k}{r_{ij}}$ für $k = 0, \ldots, 3$:

$$\binom{0}{r_{ij}} = \begin{pmatrix} \varepsilon & a & \emptyset \\ \emptyset & \varepsilon + a & b \\ b & b & \varepsilon \end{pmatrix}$$

$$\binom{1}{r_{ij}} = \begin{pmatrix} \varepsilon & a & \emptyset \\ \emptyset & \varepsilon + a & b \\ b & b + ba & \varepsilon \end{pmatrix}$$

$r_{11}^1 = r_{11}^0 + r_{11}^0 (r_{11}^0)^* r_{11}^0$
$r_{32}^1 = r_{32}^0 + r_{31}^0 (r_{11}^0)^* r_{12}^0 = b + b \cdot \varepsilon^* \cdot a$



Kostenmatrix $\begin{pmatrix} 2 & 8 & 5 \\ 3 & \infty & \infty \\ \infty & 2 & \infty \end{pmatrix}$

c_{ij}^k = Minimalkosten eines Pfades von i nach j mit Zwischenknoten $\leq k$

$c_{ij}^0 = \begin{cases} c_{ij} & i \neq j \\ 0 & i = j \end{cases}$

$c_{ij}^k = \min\{c_{ij}^{k-1}, c_{ik}^{k-1} + c_{kj}^{k-1}\}$

Andere Situation:

Minimale Kosten von Pfaden in gerichteten Graphen (V, E), Kostenfunktion

$$c : V \times V \to R_+ \cup \{\infty\}$$

Kleene:

L durch regulären Ausdruck definierbar $\Leftrightarrow$ durch NEA akzeptierbar

„⇒“

Induktion über Aufbau der regulären Ausdrücke

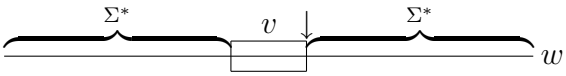
„⇐“

1. Eliminationsverfahren in VNEA
2. Kleene - Floyd - Warshall (McNaughton, Yamada)

Nachbemerkungen zu „⇒“:

1. **Pattern Matching**

   Pattern spezifiziert durch regulären Ausdruck r sucht in Text w nach „Vorkommen des Pattern“ (Existenz eines Infixes v von w mit $v \in L(r)$)



   Aufgabe: testen, ob $w \in \Sigma^* \cdot L(r) \cdot \Sigma^*$ gilt.

   Implementierung des Tests:

   $\Sigma^* \cdot r \cdot \Sigma^*$
   $\to \varepsilon - NEA$
   $\to NEA$
   $\to DEA$

   (fehlt: Optimierung des DEA auf möglichst wenige Zustände)

   Notation der regulären Ausdrücke in UNIX (Unix: grep, lex, sed, etc.):

- Σ: Buchstabenmengen mit Rückgriff auf ASCII-Reihenfolge: [a-z][0-9]
- $a_1 + \ldots + a_n$: $[a_1 a_2 \ldots a_n]$ [ist es nicht „·“ statt „+“??]
- $+$: $|$
- $\varepsilon + r$: $r?$
- $r^* r$: r^+ ($L^+ = L^* \setminus \{\varepsilon\}$
- $rrrr$: $r\{4\}$

Beispiel:

Format von Telefon-Nummern:

- (+49)0241-98765
- +49-241-98765
- 0049-241-98765

Ausdruck:

```
(\(\+49\)0|\+49\-|0049\-)241\-[1-9][0-9]*
```

2. Verallgemeinerte reguläre Ausdrücke entstehen aus Standardausdrücken durch Hinzunahme von

   - $\sim$ (einstellig) für Komplement
   - $\cap$ (zweistellig) für Durchschnitt

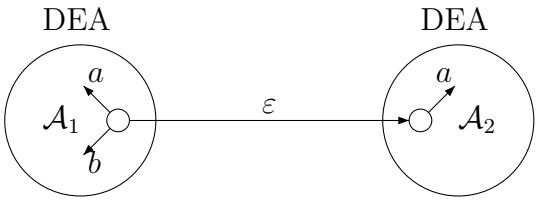
   Boolsche Operationen: $\sim$, $+$, $\cap$

   Beispiel: Worteigenschaft

   - „weder abb noch bba kommen als Infix vor“
     $\sim ((\Sigma^* abb \Sigma^*) \cap (\Sigma^* bba \Sigma^*))$ für $\Sigma = \{a, b, c\}$
   - „Zwischen je zwei a, die nur durch b's und c's getrennt sind, kommt höchstens ein b vor“
     $\sim (\Sigma^* a B a \Sigma^*)$ mit $B = (b + c)^* \cap \Sigma^* b \Sigma^* b \Sigma^*$

Satz:
Jeder verallgemeinerte reguläre Ausdruck ist äquivalent zu einem NEA

Beweis: (induktiv über Aufbau der VRA)
Anfang wie früher: $\emptyset$, ε, a
Schritt: $+$, $\cdot$, $*$, $\cap$

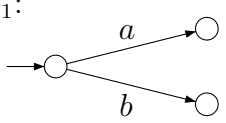


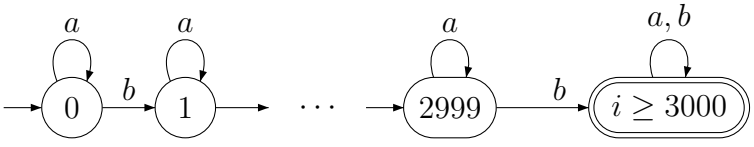
3 Tragweite der regulären Sprachen

Beispiel:

- $L_1 = \{w \in \{a,b\}* \mid$ erster und letzter Buchstabe existieren und sind gleich$\}$
- $L_2 = \{w \in \{a,b\}* \mid$ in w mindestens 3000 b's$\}$
- $L_3 = \{w \in \{0,\ldots,9\}* \mid w$ stellt durch 7 teilbare Dezimalzahl dar$\}$
- $L_4 = \{w \in \{a,b\}* \mid$ in w genauso viele a wie $b\}$
- $L_5 = \{w \in \{a,b\}* \mid w = v$ für ein geeignetes $v\}$
- $L_6 =$ Menge der voll geklammerten arithmetischen Ausdrücke, die man aus 0, 1 mit $+, \cdot$ bilden kann [Beispiel: $(0 + (((1 + 0) \cdot (1 + 1)) + 1))$]

Intuitiver Ansatz:
L regulär $\Leftrightarrow$ Mitgliedschaft eines Wort in L kann man „DEA-mäßig“ testen, nämlich durch einmaliges Durchlesen mit festem endlichen Speicher.

- L_1 regulär: „merke den ersten Buchstaben“
  L_1:
  

- L_2 regulär: Zustände 0, 1, …, 2999, ≥ 3000 (3001 viele Zustände)
  

- L_3 regulär: Übungsaufgaben: 7 Zustände genügen

- L_4: Man benötigt „Zählervariable“ mit unbeschränktem Wertebereich

  Bemerkung: L_4 ist nicht regulär

  Beweis:

  Annahme L_4 ist durch einen NEA akzeptierbar etwa $\mathcal{A} = (Q,\ \{a, b\},\ q_0,\ \Delta,\ F)$ etwa mit n Zuständen.

  Betrachte $\mathcal{A}$ auf $w = a^n \cdot b^n \in L_4$

  In einem akzeptierenden Lauf von $\mathcal{A}$ auf w ist spätestens nach dem letzten a eine Zustandswiederholung aufgetreten, etwa mit Zustand p:

  $\mathcal{A}$: $q_0 \xrightarrow{a^k} p \xrightarrow{a^j} p \xrightarrow{a^i} q \xrightarrow{b^n} q'$ mit $k + j + i = n,\ j > 0$

  Neues Input-Wort mit neuem Lauf:

  $\mathcal{A}$: $q_0 \xrightarrow{a^k} p \xrightarrow{a^i} q \xrightarrow{b^n} q' \in F$

  $\mathcal{A}$ akzeptiert $a^{k+i} < n$ und b^n mit $k + i = n - j$

  Widerspruch dazu, dass $\mathcal{A}$ die Sprache L_4 akzeptiert

- L_5: **Behauptung:** L_5 ist nicht regulär

  Beweis:

  Annahme L_5 durch NEA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ akzeptierbar, etwa mit n Zuständen.

  Betrachte $\mathcal{A}$ auf dem Input $a^n b a^n b \in L_5$.

  In einem akzeptierende Lauf von $\mathcal{A}$ auf $a^n b a^n b$ tritt eine Zustandswiederholung auf:

  $\mathcal{A}$: $q_0 \xrightarrow{a^k} p \xrightarrow{a^j} p \xrightarrow{a^i} q \xrightarrow{ba^nb} q' \in F$ mit $k + j + i = n, j \geq 0$

  $\mathcal{A}$: $q_0 \xrightarrow{a^k} p \xrightarrow{a^i} q \xrightarrow{ba^nb} q' \in F$ mit $a^{k+i} b a^n b \notin L_5$ Widerspruch!

  Beachte:

  Auch akzeptierender Lauf durch Interaktion des a^j-Teils:

  $\mathcal{A}$: $q_0 \xrightarrow{a^k} p \xrightarrow{a^j} p \xrightarrow{a^j} p \xrightarrow{a^i} q \xrightarrow{ba^nb} q' \in F$

3.0.6 Pumping Lemma

Sei L regulär. Dann gibt es ein $m \geq 1$, so dass jedes Wort $w \in L$ mit $|w| \geq n$ zerlegbar ist als $w = xyz$ mit $y \neq \varepsilon$ und $xy^iz \in L$ für alle $i \geq 0$ wählen kann.

- In den Anwendungen kann man $n =$ Zustandszahl eines NEA für L wählen
- Zu der Zerlegung $w = xyz$ kann man $|xy| \leq n$ verlangen

Beweis:
L sei regulär, also akzeptierbar durch einen NEA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ mit n Zuständen.
Zeige Pumping Lemma für dieses n. Sei $w \in L$ mit $|w| \geq n$. Betrachte akzeptierenden Lauf von $\mathcal{A}$ auf w, spätestens nach dem n-ten Buchstaben ist eine Zustandswiederholung aufgetreten, etwa mit p.

$$\mathcal{A} : q_0 \xrightarrow{x} p \xrightarrow{y} p \xrightarrow{z} q \in F$$

Wähle

- x so, dass p erstmals erreicht
- y so, dass p danach erstmals wieder erreicht
- $p,\ x,\ y$ so wählbar, dass $|xy| \leq n$
- $y \neq \varepsilon$

Dann ist auch $\mathcal{A}$: $q_0 \xrightarrow{x} p \xrightarrow{y^i} p \xrightarrow{z} q \in F$ für $i \geq 0$. Also $xy^iz \in L$ für $i \geq 0$.
Anwendung:
Sprache AA = Menge der voll geklammerten arithmetischen Ausdrücke, die aus 0, 1 mit $+, \cdot$ entstehen.

Behauptung: AA sei nicht regulär

Annahme:
Sei AA regulär, dann wähle n gemäß Pumping Lemma.
Betrachte $w = (\ldots((1 + 1) + 1) + 1) \in AA$. Nach Pumping Lemma wähle Zerlegung $w = xyz$, insbesondere $|xy| \leq n$

$$\underbrace{\underbrace{(\ldots(}_{k}\underbrace{(\ldots(}_{l}\underbrace{(\ldots(}_{m}\underbrace{1+1)\ldots+1)}_{n}}_{z}}_{y}}_{x}$$

Nach dem Pumping Lemma wäre $xz \in AA$ mit weniger „(“-Klammern als „)“-Klammern, aber $xz \notin AA$. Widerspruch!

Alternativer Ansatz:

Definition:
Zu $L \subseteq \Sigma*$ (beliebig) definiere $u \sim_L v :\Leftrightarrow \forall\ w \in \Sigma\ (uw \in L \Leftrightarrow vw \in L)$

Bemerkung:

- $\sim_L$ ist Äquivalenzrelation ?über? Σ^*
- $u \sim_L u$

- $u \sim_L v \Leftrightarrow v \sim_L u$
- $u_1 \sim_L u_2$, $u_2 \sim_L u_3 \Rightarrow u_1 \sim_L u_3$
- Äquivalenzklasse von u: $[u]_L$
- $index(\sim_L)$ = Anzahl der $\sim_L$ Klassen

Satz:
Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein DEA, der L akzeptiert. Dann: $index(\sim_L) \leq |Q|$.

Beweis:
Wähle aus der $\sim_L$-Klasse ein Wort u und betrachte $\delta^*(q_0, u)$

Zeige:
Alle diese Zustände sind verschieden, hierzu genügt: $\delta^*(q_0, u) = \delta^*(q_0, v) \Rightarrow u \sim_L v$

$$\begin{array}{lll} w \text{ beliebig: } uw \in L & \Leftrightarrow & \mathcal{A} \text{ akzeptiert } uw\text{: } \delta^*(\delta^*(q_0, u), w) \in F \\ & \Leftrightarrow & \delta^*(\delta^*(q_0, v), w) \in F \\ & \Leftrightarrow & \mathcal{A} \text{ akzeptiert } vw \in L. \end{array}$$

Folgerung 1:

Hat $\sim_L$ unendlich viele Äquivalenzklassen, dann ist L nicht regulär.

Folgerung 2:

Sind $u_1, \ldots, u_n$ paarweise nicht $\sim_L$-äquivalent, und wird durch DEA mit n Zuständen akzeptiert dann sind $[u_1]_L, \ldots, [u_n]_L$ alle $\sim_L$-Klassen.

Beispiel 1:

$L = AA$. Betrachte $u_1 = (, u_2 = ((, \ldots, u_i = (^i$

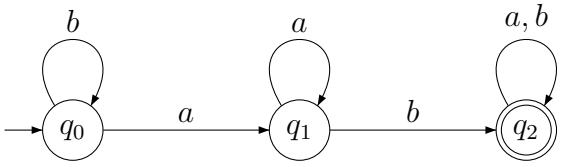
Behauptung:

Ist $i \neq j$, $u_i \nsim_L u_j$ (Mit Folgerung 1 also: L nicht regulär).

Finde u_i, u_j jeweils w: $u_i w \in L$, $u_j w \in L$ (oder umgekehrt)

$u_i = \underbrace{(\ldots(}_{i}$, nun wählen wir $w = \underbrace{0+0)+)\ldots+0)}_{i}$, somit ist $u_i w \in L$, aber für $u_j = \underbrace{(\ldots(}_{j}$ ist $u_j w \notin L$.

Beispiel 2: $L = \{w \in \{a, b\}^* \mid w \text{ hat Infix } ab\}$



- $\varepsilon \not\sim_L a$: Wähle $w = b$, dann ist $\varepsilon w \notin L$, aber $aw \in L$
- $\varepsilon \sim_L b$: $\varepsilon w \in L \Leftrightarrow$ in w ist Infix $ab \Leftrightarrow bw \in L$
- $a \sim_L aa$: $aw \in L \Leftrightarrow$ in w ist Infix ab oder w beginnt mit b
- $a \not\sim_L ab$: $w = \varepsilon$, dann ist $aw \notin L$, aber $abw \in L$
- $\varepsilon \not\sim_L ab$

Daraus ergeben sich die Äquivalenzklassen: $[\varepsilon]_L$, $[a]_L$ und $[ab]_L$

4 Algorithmen über Automaten

4.1 Algorithmische Probleme:

1. **Nicht-Leerheitsproblem:**
   Gegeben NEA/DEA $\mathcal{A}$, gilt $L(\mathcal{A}) \neq \emptyset$?
2. - **Inklusionsproblem**:
     Gegeben DEA's $\mathcal{A}$, $\mathcal{B}$, gilt $L(\mathcal{A}) \subseteq L(\mathcal{B})$?
   - **Äquivalenzproblem**:
     Gegeben DEA's, $\mathcal{A}$, $\mathcal{B}$, gilt $L(\mathcal{A}) = L(\mathcal{B})$?
3. **Minimierungsproblem:**
   Gegeben DEA $\mathcal{A}$, finde einen zu $\mathcal{A}$ äquivalenten endlichen Automat möglichst kleiner Zustandszahl.

Zu 1: (Nicht-Leerheitsproblem)

Satz: Das Nicht-Leerheitsproblem für NEA's $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ ist entscheidbar in $O(|Q + |\Delta|)$.
(12.0)

Beweis: Es gilt: $L(\mathcal{A}) \neq \emptyset$

$\Longleftrightarrow$ es existiert ein Lauf von $\mathcal{A}$, der in q_0 beginnt und in einem F-Zustand endet.

$\overset{*}{\Longleftrightarrow}$ es existiert im Transitionsgraph (Q, Δ) einen Pfad von q_0 in die Menge F.

Teste die zweite Äquivalenzbedingung (*) mit Tiefensuche oder Breitensuche von q_0 im Transitionsgraph (Q, Δ). Überprüfe, ob in der berechneten Liste der erreichbaren Zustände ein F-Zustand enthalten ist. Wenn

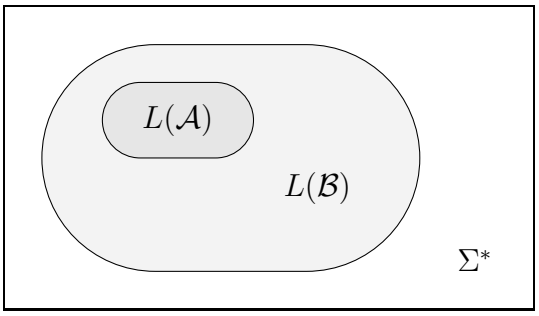
ja: Ausgabe $L(\mathcal{A}) \neq \emptyset$
nein: Ausgabe $L(\mathcal{A}) = \emptyset$

Zu 2: (Inklusions- / Äquivalenzproblem)

Satz:

Das Inklusionsproblem und das Äquivalenzproblem sind für DEA's $\mathcal{A}$, $\mathcal{B}$ (über Σ) ist entscheidbar im Polynomzeit.

Beweis:



Es gilt:

$$\mathcal{A} \subseteq L(\mathcal{B}) \Leftrightarrow \cap(\Sigma^* - L(\mathcal{B}) = \emptyset$$

Algorithmus:

- Bilde zu $\mathcal{B}$ den Komplement-DEA $\mathcal{B}'$
  (Vertausche Endzustände mit den Nichtendzuständen)
- Bilde zu $\mathcal{A}$, $\mathcal{B}'$ den Produkt-DEA $\mathcal{C} := \mathcal{A} \times \mathcal{B}'$
- Teste mit vorigem Satz, ob $L(\mathcal{C}) = \emptyset$

Komplexität für $n = max(|Q_\mathcal{A}|, |Q_\mathcal{B}|)$ polynomial.

Bemerkung:

Erster Schritt ist aufwendig für den Fall der NEA's (Potenzmengenkonstruktion)

Zur Lösung der Äquivalenzproblems für $\mathcal{A}$ und $\mathcal{B}$, überprüfe:

- $L(\mathcal{A}) \subseteq L(\mathcal{B})$
- $L(\mathcal{B}) \subseteq L(\mathcal{A})$

mit Inklusionstest

Anwendung des Inklusionstest:

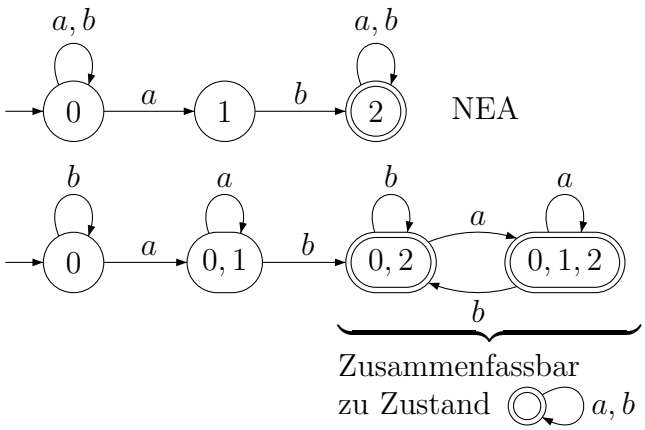
Algorithmische Verifikation von Programmen, mit endlichen Zustandsraum.

Beispiele:

- Protokolle (Mutal Exclusion)
- Steuerungsprogramme
- Programme modelliert durch endlichen Automaten $\mathcal{A}_p$
- Spezifikation: Anforderung an die Programmläufe, etwa beschrieben durch Logik-Formel, regulärer Ausdruck, Automat $\mathcal{A}_s$
- Korrektheit des Programms bezüglich der Spezifikation
- $L(\mathcal{A}_p) \subseteq L(\mathcal{A}_s)$
- Modell-Checking-Problem (lösbar durch Inklusionstest)

Zu 3: (Minimierungsproblem)

Beispiel: Potenzmengenkonstruktion



Ansatz zur Minimierung von DEA's

1. Elimination der Zustände, die von q_0 aus nicht erreichbar sind
2. Zusammenfassen „äquivalenter" Zustände

Erster Schritt: Äquivalenz von Zuständen

Gegeben ist der Automat DEA

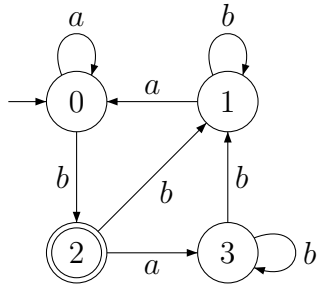
$\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ mit $p \sim_{\mathcal{A}} q$: $\Longleftrightarrow [\forall\ w \in \Sigma^*\ \delta^*(p, w) \in F \Leftrightarrow \delta^*(q, w) \in F]$

„von p und q aus werden die gleichen Wörter akzeptiert" (12.3)

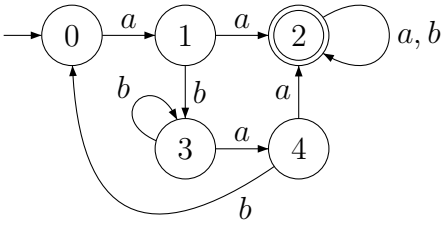
Äquivalenzklasse von p: $\widetilde{p}$

Beispiel: Bestimmung nicht-äquivalenter Zustände

Zu p, q finde w mit $\delta^*(p, w) \in F$, $\delta * (q, w) \notin F$ oder umgekehrt



Beispiel:



	0	1	2	3
1	x_2			
2	x_1	x_2		
3		x_2	x_1	
4	x_2		x_1	x_2

Trennbarkeit über das leere Wort ε (x_1 erster Durchgang)
Trennbarkeit über a (x_2 zweiter Durchgang)

- $0 \not\sim_{\mathcal{A}} 2$, mit $w = \varepsilon$
- $0 \not\sim_{\mathcal{A}} 1$, mit $w = b$
- $1 \not\sim_{\mathcal{A}} 3$, mit $w = ab$

Bemerkung:

$p \sim_{\mathcal{A}} q$, $a \in \Sigma$, $\delta(p,a) \sim_{\mathcal{A}} \delta(q,a)$

Beweis:

Zeige $\delta(p,a) \not\sim_{\mathcal{A}} \delta(q,a) \Rightarrow p \not\sim_{\mathcal{A}} q$

Nach Voraussetzung existiert w mit $\underbrace{\delta^*(\delta(p,a),w)}_{\delta^*(p,aw)} \in F$ und $\underbrace{\delta^*(\delta(q,a),w)}_{\delta^*(q,aw)} \notin F$.

Folglich $p \not\sim_{\mathcal{A}} q$, mit aw

Definition (reduzierter DEA):
Zu $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ definiere $\widetilde{\mathcal{A}} = (\widetilde{Q}, \Sigma, \widetilde{q_0}, \widetilde{\delta}, \widetilde{F})$

$\widetilde{Q}$ = Menge der $\sim_{\mathcal{A}}$-Klassen, allgemein $P \subseteq Q : \widetilde{P} = \{\widetilde{p} \mid p \in P\}$ mit $\widetilde{\delta}(\widetilde{p}, a) := \widetilde{\delta(p, a)}$.
$\widetilde{\delta}$ ist wohldefiniert: $p \sim_{\mathcal{A}} q \Rightarrow \delta(p, a) \sim_{\mathcal{A}} \delta(q, a)$

Behauptung:
$\widetilde{\mathcal{A}}$ ist äquivalent zu $\mathcal{A}$

Vorbereitung:
$p \sim_{\mathcal{A}} q$, $w \in \Sigma^*$ $\delta^*(p, w) \sim_{\mathcal{A}} \delta^*(q, w)$ (einfache Induktion über den Aufbau der Wörter)

Beweis:

$$\begin{array}{lll} \mathcal{A} \text{ akzeptiert } w & \iff & \delta^*(q_0, w) \in F \\ & \iff & \widetilde{\delta^*(q_0, w)} \in \widetilde{F} \\ & \overset{\text{Vorbereitung}}{\iff} & \widetilde{\delta}^*(\widetilde{q_0}, w) \in \widetilde{F} \\ & \iff & \widetilde{\mathcal{A}} \text{ akzeptiert } w. \end{array}$$

Bestimmung von $\widetilde{\mathcal{A}}$

Beispiel: (12.4)

Ansatz:
Bestimme die Paare nicht-äquivalenter Zustände nach Wörtern wachsender Länge p, q trennbar über w: $\delta^*(p, w) \in F$, $\delta^*(q, w) \notin F$ oder umgekehrt
(12.5)

5 Weitere Konstruktionen

1. Automaten mit Ausgabe
2. Zwei-Wege-Automaten

5.1 Automaten mit Ausgabe

Erweiterung von DEA um Ausgaben-Komponente, zwecks Berechnung von Wortfunktionen $f : \Sigma^* \to \Gamma^*$

Idee
Nehme Transitionen: $p \xrightarrow{a/v} q$, d.h. bei Eingabe a gebe Wort v aus.
Formal verwende neben Transitionsfunktion δ eine Ausgabefunktion:

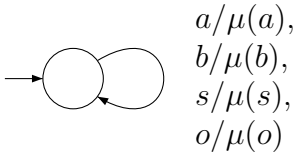
- λ: $Q \times \Sigma \to \Gamma^*$
- $\lambda(p, a) = v$

Beispiel 1: Morsekodierung $\mu : \Sigma^*$

- $\mu(a) = \cdot -_{\sqcup}$
- $\mu(b) = - \cdots_{\sqcup} \cdots$
- $\mu(s) = \cdots_{\sqcup}$
- $\mu(o) = - - -_{\sqcup}$

$\mu(a_1 \dots a_n) = \mu(a_1) \dots \mu(a_n)$

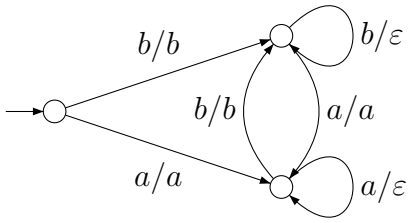
μ ist Homomorphismus (spezifizierbar allein durch Werte auf Σ)



Beispiel 2: (Stotterreduktion)

$\Sigma = \Gamma = a, b$

w^- entstehe aus w durch Zusammenziehen der maximalen Infixe a^i bzw. b^i $(i > 0)$ auf a bzw. b. D.h. beispielsweise $(\underline{aab}\,\underline{aabb}\,ab)^- = ababab$.



Definition: Eine verallgemeinerte sequentielle Maschine (Generalized Sequential Machine, GSM) hat die Form:

$\mathcal{A} = (Q, \Sigma, \Gamma, q_0, \delta, \lambda)$ mit:

- Q, q_0, δ wie bei DEA'
- Σ ist das Eingabealphabet
- Γ ist das Ausgabealphabet
- $\lambda : Q \times \Sigma \to \Gamma^*$ [Für $\lambda : Q \times \Sigma \to \Gamma$ spricht man von sequentiellen Maschinen]

Graphische Darstellung $p \xrightarrow{a/v} q$ für $\delta(p, a) = q$ und $\lambda(p, a) = v$.

Erweitere λ zu $\lambda^* : Q \times \Sigma^* \to \Gamma^*$.

$\lambda^*(p, w)$ = Ausgabe von p bei Eingabe w.

Definition induktiv:

- $\lambda^*(p, \varepsilon) = \varepsilon$
- $\lambda^*(p, wa) = \lambda^*(p, w)\lambda(\delta^*(p, w), a)$

Die durch $\mathcal{A}$ berechnete Funktion ist $f_{\mathcal{A}} : \Sigma^* \to \Gamma^*$, gegeben durch $f_{\mathcal{A}}(w) = \lambda^*(q_0, w)$

f heißt verallgemeinert sequentiell, falls $f = f_{\mathcal{A}}$ für eine GSM $\mathcal{A}$.

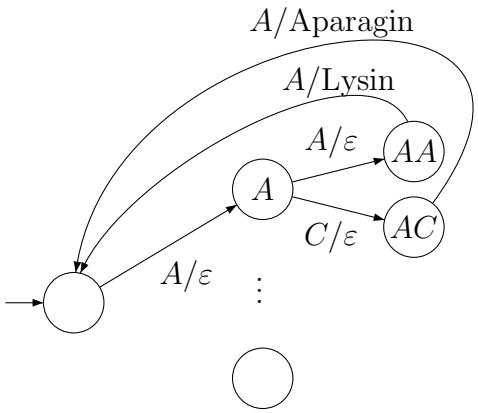
Beispiel 1: (RNA-Aminosäuren-Übersetzer) Sei nun

- $\Sigma = \{\underbrace{A}_{\text{Ademin}}, \underbrace{C}_{\text{Cytosin}}, \underbrace{G}_{\text{Guanin}}, \underbrace{U}_{\text{Uracile}}\}$
- $\Gamma = \{\text{Adanin, Lencin, Glycin, } \dots\}$ (Alphabet der Aminosäuren)

und die Kodierungsfunktion:

- c: $AAA \mapsto$ Lysin, $AAC \mapsto$ Asparagin

Die Realisierung durch GSM:

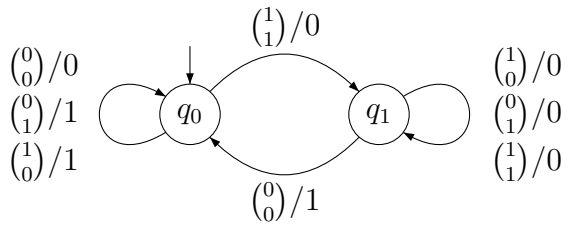


Beispiel 2: (Binäre Addition)

$$\Sigma = \{\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}\}$$

$$\begin{array}{l} 011101 \\ 001001 \\ \hline 100110 \end{array} \leftarrow$$

GSM für Addition gespiegelter Binärzahlen (mit wenigstens einer führenden Null):



5.1.1 Eigenschaften verallgemeinert sequentieller Funktionen

1. Sei $f : \Sigma^* \to \Gamma^*$ verallgemeinert sequentiell. Dann existiert $k \geq 1$ mit $|f(w)| \leq k \cdot |w|$ für alle $w \in \Sigma^*$ („f ist linear längenbeschränkt“).

   In GSM für f betrachte maximale Länge eines Wortes $\lambda(p, a)$. Sei diese Länge k. In jedem Schritt (Verarbeitung eines $a \in \Sigma$) liefert GSM höchstens k Ausgabebuchstaben. Also $|f(w)| \leq k \cdot |w|$.

   Anwendungsbeispiel:

   Transformation Dualzahlen $\to$ Unärzahlen

   - $f : \{0, 1\}^* \to \{|\}^*$
   - $f(100) = ||||$

   f nicht linear längenbeschränkt, also nicht verallgemeinert sequentiell.

2. Sei $f : \Sigma^* \to \Gamma^*$ verallgemeinert sequentiell. Ein Wert $f(uv)$ für das Argument uv hat jeweils $f(u)$ als Präfix („f ist präfixtreu“). Klar aus der Definition von GSM-berechenbaren Funktionen.

   Anwendungsbeispiel:

   $f : \{a\}^* \to \{a\}^*$

   $f(a^i) = \begin{cases} \varepsilon & \text{falls } i \text{ gerade} \\ a & \text{falls } i \text{ ungerade} \end{cases}$

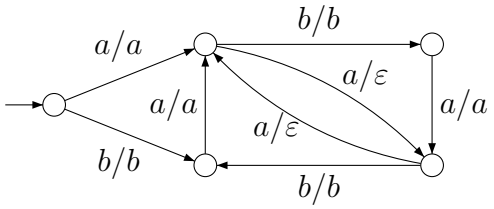
   f nicht präfixtreu (also nicht verallgemeinert sequenziell), nehme:

   - $u = v = a$
   - $uv = aa$
   - $f(uv) = \varepsilon$
   - $f(u) = a$

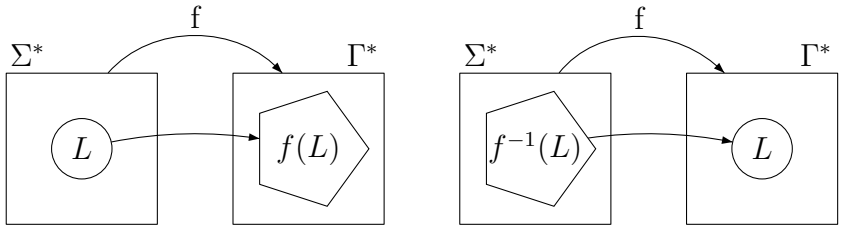
   $f(uv)$ hat $f(u)$ nicht als Präfix $\Rightarrow$ f ist nicht präfixtreu.

3. (a) $f : \Sigma^* \to \Gamma^*$ verallgemeinert sequenziell:
$L \subseteq \Sigma^*$ regulär $\Rightarrow f(L) \subseteq \Gamma^*$ regulär

Beispiel: Stotterreduktion f: $L = \{w \mid |w|_a \text{ gerade}\}$



(b) $L \subseteq \Gamma^*$ regulär $\Rightarrow f^{-1}(L) \subseteq \Sigma^*$ regulär



5.1.2 Allgemeine Konstruktion

8.6.'04

$\mathcal{A}_1 = (Q_1,\ \Sigma,\ \Gamma,\ q_{01},\ \delta_1,\ \lambda_1)$ GSM f
$\mathcal{A}_2 = (Q_2,\ \Sigma,\ \Gamma,\ q_{02},\ \delta_2,\ F_2)$ DEA für L
Bilde Produktautomat (NEA mit Worttransitionen)
$\mathcal{A} = (Q_1 \times Q_2,\ \Gamma,\ (q_{01}, q_{02}),\ \Delta,\ Q_1 \times F_2)$
Führe Worttransition ein
$((p_1, p_2),\ w,\ (q_1, q_2)) \in \Delta \Leftrightarrow$ existiert $a \in \Sigma$ mit:

- $\delta_1(p_1, a) = q_1$
- $\lambda_1((p_1, p_2), a) = w$
- $\delta_2(p_2, a) = q_2$

Dann gilt:
$\mathcal{A} : (q_{01}, q_{02}) \xrightarrow{u} Q_1 \times F_2 \Leftrightarrow$ existiert Eingabewort w mit $f_{\mathcal{A}_1(w)} = u$, d.h. $\mathcal{A}_2$ akzeptiert w ($w \in L$).

Anwendungsbeispiel: (Binäre Multiplikation)
$f_\bullet : \{\binom{0}{0}, \binom{0}{1}, \binom{1}{0}, \binom{1}{1}\}^* \to \{0,1\}^*$

$$\begin{array}{r} 1000 \cdot 0111 \\ \hline 1000 \\ 1000 \\ 1000 \\ \hline 111000 \end{array}$$

gespiegelt: $\underbrace{\binom{0}{1}^n \binom{1}{0}}_{\text{reguläre Sprache}} \mapsto 0^n 1^n$

$L = (\binom{0}{1}^+ \binom{1}{0})$. Also $f(L)$ nicht regulär und $f_\bullet$ nicht verallgemeinert sequentiell.

5.1.3 Satz von Ginsburg und Rose

(a), (b), (c) treffen auf f zu $\Leftrightarrow$ f ist verallgemeinert sequenziell

5.2 Endliche Zwei-Wege-Automaten

Motivation:

- bestimmte Sprachen „rückwärts gerichtet“
- Speicher-Einsparung

Fragen:

- Mächtigkeit?
- Effizienz?

Definition (2DEA/2NEA):
Ein deterministischer/nichtdeterministischer endlicher 2-Wege(2DEA/2NEA) Automat hat die Form:
$\mathcal{A} = (Q,\ \Sigma,\ q_0,\ \delta,\ F)$ bzw. $\mathcal{A} = (Q,\ \Sigma,\ q_0,\ \Delta,\ F)$ mit:

- Q: endliche Zustandsmenge
- Σ: Eingabealphabet
- $q_0 \in Q$: Anfangszustand
- $F \subseteq Q$: Endzustandsmenge
- für 2DEA: $\delta :\ Q \times (\Sigma \dot\cup \{¢,\ \$\} \to Q \times l, r$: die Transitionsfunktion
- für 2NEA: $\Delta \subseteq Q \times \{¢, \$\} \times Q \times \{l, r\}$: die Transitionsrelation

Konfiguration auf Eingabe $w_1 \dots w_n$: $\mathcal{A}$ **akzeptiert** w, falls es einen **Lauf** gibt:

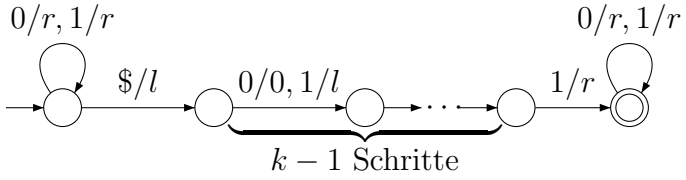
$$(p_0, i_0),\ (p_1, i_1), \dots,\ (p_k, i_k) \in K^+$$

- $p_0 = q_0$
- $i_0 = 1$
- $p_k \in F$
- $i_k = n + 1 \forall j \in \{0, \ldots, k-1\}$
- Für DEA: $\delta(p_j, w_{i_j}) = (p_{j+1}, m)$
- Für NEA: $(p_j, w_{i_j}, p_{j+1}, n) \in \Delta$

$$i_{j+1} = \begin{cases} i_j + 1 & \text{falls } m = r \\ i_j - 1 & \text{falls } m = l \end{cases}$$

$L(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A} \text{ akzeptiert } w\}$

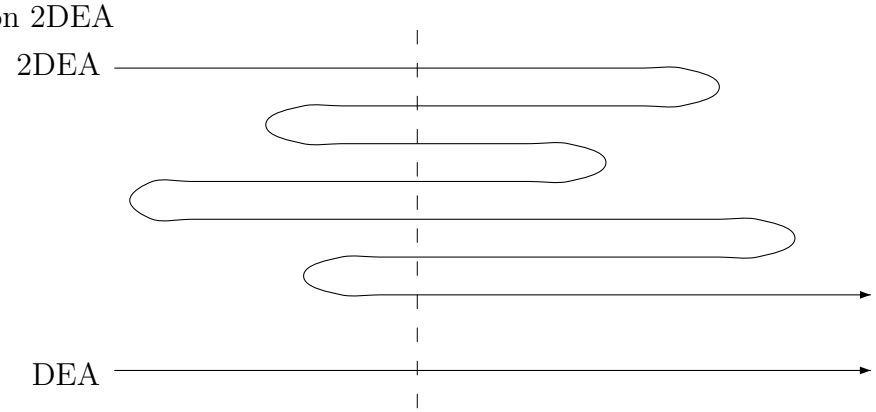
Beispiel:
$L_k = \{w \in \{0,1\}^* \mid k - \text{letzter Buchstabe ist } 1\}$



Bemerkung: fehlende Transitionen führen zu einer Senke

Satz:
Zu jedem 2 DEA/2NEA existiert ein äquivalenter DEA bzw. NEA

Beweis: für 2DEA → DEA
Gegeben 2DEA $\mathcal{A} = (Q,\ \varepsilon,\ q_0,\ \delta,\ F)$, ohne Beschränkung der Allgemeinheit $s \notin Q$.
Lauf von 2DEA



DEA $\mathcal{A}' = (Q', \Sigma, q_0', \delta', F')$

- $Q' = (Q \cup \{\perp\})^{Q \dot\cup \{s\}} = \{f \mid f : Q \dot\cup s \to Q \cup \{\perp\}\}$

- $f \in Q' = \begin{cases} f(s) & \text{Zustand beim ersten Erreichen der Position} \\ f(q) = p & \text{falls in } q \text{ nach links, in } p \text{ zurück} \\ q' & \text{falls: } \delta(q, e) = (q', r) \end{cases}$
- $f(q) = \begin{cases} q' & \text{falls } \delta(q, a) = (q', r) \\ \widehat{q} & \text{falls } \delta(q, a) = (q', l) \text{ und Linkslauf von } q^* \text{ zu } \widehat{q} \text{ existiert} \\ \perp & \text{sonst} \end{cases}$

Linkslauf ist eine Folge $p_1, r_1, \ldots, p_k, r_k$
$f(p_i) = r_i$
$\delta(r_i, a) = (p_{i+1}, l)$
$\delta(r_k, a) = (p_k, r)$

6 Grammatiken

- Automaten

  Test auf Input(-wort) mit Antwort ja/nein

  Definiere Sprache: Menge der Wörter, der mit Ergebnis „ja“ getestet werden können

- Grammatiken

  Erzeugung von Wörter(aus Anfangswort)

  Definiere Sprache: Menge der erzeugbaren Wörter.

- **Beispiel**

  AA = Menge der voll geklammerten Arithmischen Ausdrücke, die man aus 0, 1 mit $+$ und $\cdot$ erzeugen kann, z.B: $((0 + 1) \cdot 1)$.

  Erinnerung: AA ist nicht regulär.

  Definition mit BNF-Regeln:

  $\langle AA \rangle ::= 0 \mid 1 \mid (\langle AA \rangle + \langle AA \rangle) \mid (\langle AA \rangle \cdot \langle AA \rangle)$

  Herleitung(Erzeugung) von $((0 + 1) \cdot 1)$

  $$\begin{array}{ll} & \langle AA \rangle \\ \vdash & (\langle AA \rangle \cdot \langle AA \rangle) \\ \vdash & ((\langle AA \rangle + \langle AA \rangle) \cdot \langle AA \rangle) \\ \vdash & ((0 + \langle AA \rangle) \cdot \langle AA \rangle) \\ \vdash^2 & ((0 + 1) \cdot 1) \end{array}$$

Definition
Eine Chomsky-Grammatik hat die Form $G = (N, \Sigma, P, S)$ mit:

- N ist endliche Menge von „Nichtterminalsymbolen“ („Variablen“)

- Σ Terminalsymbol-Alphabet(disjunkt zu N)
- P endliche Menge von „Produktionen“ („Regeln“)
  $P \subseteq (\Sigma \cup N)^* N (N \cup \Sigma^*) \times (N \cup \Sigma)^*$
- $S \in N$ („Startsymbol“)
  Notation: $(\alpha, \beta) \in P$, dafür $\alpha \to \beta \in P$
  In BNF: $\alpha ::= \beta$

Symbole in $N : A, B, C, \ldots, S$
Symbole in $\Sigma : a, b, c$
Wörter über $N \cup \Sigma : \alpha, \beta, \gamma \ldots$ (Satzform) Wörter über $\Sigma : u, v, w, \ldots$

Regelanwendung für $G = (N,\ \Sigma,\ P,\ S)$
$\alpha \vdash_G \beta$: es existiert $\gamma, \delta, \alpha_1, \beta_1$ mit:
$\alpha = \gamma\alpha_1\delta$, $\beta = \gamma\beta_1\delta$, $\alpha_1 \to \beta_1 \in P$

Beispiel: $\underbrace{(}_{\gamma}\underbrace{\langle AA\rangle}_{\alpha_1}\underbrace{\cdot\langle AA\rangle}_{\gamma}\underbrace{)}_{\gamma} \vdash \underbrace{((}_{\gamma}\underbrace{\langle AA\rangle + \langle AA\rangle}_{\beta_1}\underbrace{)}_{\gamma}\underbrace{\cdot\langle AA\rangle}_{\gamma}\underbrace{)}_{\gamma}$

Notationen:
$\alpha \vdash_G^n \beta :\Leftrightarrow$ es existieren $\alpha_0, \ldots, \alpha_n$ mit $\alpha_0 = \alpha$, $\alpha_i \vdash_G \alpha_{i+1}$, $\alpha_n = \beta$ $(i < n)$ analog: $\alpha \vdash_G^{\leq n} \beta$
[hier fehlt etwas fehlt etwas!!!]
Die durch G erzeugte Sprache ist:
$L(G) = \{w \in \Sigma^* \mid SL_G^* w\}$
Beispiel:

- G_1 :
  - $S \to abc \mid aAbc$
  - $Ab \to bA$
  - $Ac \to Bbcc$
  - $bB \to Bb$
  - $aB \to aaA \mid aa$ [hier: $N = \{A, B, S\}$, $\Sigma = \{a, b, c\}$]
- G_2: $S \to 0 \mid 1 \mid (S + S) \mid (S \cdot S)$
- G_2':
  - $S \to aB \mid bA$
  - $A \to a \mid aS \mid bAA$
  - $B \to b \mid bS \mid aBB$

- G_3:
  - $S \to aA \mid a$
  - $A \to aA \mid a \mid bB$
  - $B \to aB \mid bA \mid b$

Ableitungen:

$G_1 : S \vdash_{G_1} aAbc \vdash_{G_1} abAc \vdash_{G_1} abBbcc \vdash_{G_1} (*)aBbbcc \vdash_{G_1} aabbcc$
$G_2' : S \vdash_{G_2'} aB \vdash_{G_2'} abS \vdash_{G_2'} abbA \vdash_{G_2'} abbaS \vdash_{G_2'} abbaaB \vdash_{G_2'} abbaab$
$G_3 : S \vdash_{G_3} aA \vdash_{G_3} aaA \vdash_{G_3} aabB \vdash_{G_3} aabbA \vdash_{G_3} aabba$

Bemerkung zu G_1, G_2', G_3:

1. $L(G_1) = \{a^n b^n c^n \mid n \geq 1\}$
2. $L(G_2') = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$
3. $L(G_3) = w \in \{a, b^* \mid w$ beginnt mit a und hat geradzahlig viele $b\}$

zu 1:

„$\supseteq$“: Wir zeigen durch Induktion für $i \geq 1$: $S \vdash^* a^i B b^{i+1} c^{i+1}$ $\forall n \geq 1$

(Dann auch herleitbar a^{i+1} b^{i+1} c^{i+1} ohnehin $S \vdash abc$)

$i = 1$:

Nehme obige Ableitung bis (*)

Induktionsschritt I.V. $a^i B b^{i+1} c^{i+1}$ herleitbar

Finde Ableitung von $a^{i+1} B b^{i+2} c^{i+2}$

Beginne mit S $\vdash \ldots \vdash a^i B b^{i+1} c^{i+1}$

Verlängere: $\vdash a^{i+1} A b^{i+1} c^{i+1}$

$(i+1)\times$: $Ab \to bA \vdash^{i+1} a^{i+1} b^{i+1} A c^{i+1} \vdash a^{i+1} B b c^{i+2} L^{i+1} a^{i+1} B b^{i+2} c^{i+2}$

„$\subseteq$“: Zeige $S \vdash^* w \Rightarrow w = a^n b^n c^n$ für ein $n \geq 1$

Falls Ableitung Länge 1: $S \vdash w$ $w = abc$ sonst $S \vdash aAbc \vdash \ldots \vdash (*)aBbbcc$ von einem Wort der Form $a^i B b^{i+1} c^{i+1}$ direkt erzeugbar: $a^{i+1} b^{i+1} c^{i+1}$ oder $a^{i+1} A b^{i+1} c^{i+1}$ von dort aus eindeutig bis $a^{i+2} b^{i+2} c^{i+2}$

Also Terminalwörter nur der gewünschten Form ableitbar.

Zu G_2':
Zeige durch Induktion über $|w|$:

1. $S \vdash^* w \Leftrightarrow |w|_a = |w|_b$

2. $A \vdash^* w \Leftrightarrow |w|_a = |w|_b + 1$
3. $B \vdash^* w \Leftrightarrow |w|_a + 1 = |w|_b$

(Dann fertig) **Induktionsanfang** $|w| = 1$:

1. (beide Seiten falsche) $\surd$
2. $\surd$
3. $\surd$

Induktionsvoraussetzung: 1, 2, 3 gelten für $|w| = k + 1$ (gegeben)

Induktions Behauptung:

Zeige 1 (2, 3 analog)

Es gelte $S \vdash^* w$ z.B. $S \vdash aB \vdash \ldots \vdash \underbrace{av}_{w} aB \to c$ [Fall $S + bA \vdash \ldots \vdash bv$]

Also B $\vdash^* v$ mit $|v| = k$

Nach Induktionsvoraussetzung (3) $|v|_a + 1 = |v|_b$

Also $|w|_a = |w|_b$

$\Leftarrow$ Gegeben $|w| = k+1$, $|w|_a = |w|_b$ z.B. $w = av$ [$w = bv$ analog] Dann $|v|_b = |v|_a + 1$

mit Induktionsvoraussetzung (3) $\Leftarrow$: $B \vdash^* v$

Also $S \vdash aB \vdash^* av = w$

Definition
Eine Grammatik $G = (N, \Sigma, P, S)$ heißt

- vom Typ 0 im allgemeinen Fall
- vom Typ 1 (Kontextsensitiv), falls für jede Regel $\alpha \to \beta$ gilt $|\alpha| \leq |\beta|$
- vom Typ 2 (kontextfrei), falls jede Regel die Form $A \to \beta$ hat
- vom Typ 3 (rechtslinear), wenn jede Regel die Form $A \to, A \to w, A \to wB, w \neq \varepsilon$

Bei Typ 1 - 3 verlangen wir die sogenannte ε-Bedingung:
Regel $S \to \varepsilon$ erlaubt, aber dann darf S auf keiner Rechten Regelseite vorkommen
[Also $S \vdash^* \varepsilon$ nur direkt mit $S \to \varepsilon$ möglich]

Grammatiken $G = (N, \Sigma, P, S)$

ε-Bedingung:
ε auf rechter Seite in $S \to \epsilon$ zugelassen, dann also S auf keiner rechten Regelseite.

Typ 0: keine weitere Einschränkung

Typ 1: In $\alpha \to \beta$ gilt $|\alpha| \leq |\beta|$ (kontextsensitive Grammatik) z.B. $Aa \to \beta_1\ Ab \to \beta_2$

Typ 2: Regeln $A \to \beta$ (kontextfreie Grammatiken) z.B. $Aa \to CD$

Typ 3: Regeln $A \to w, A \to wB$ (rechtslineare Grammatik)

[Diese Einteilung wurde eingeführt von Noam Chomsky (*7.12.1928, Philadelphia)]

Definition
Eine Sprache L ist vom Typ i $(i = 0, \ldots, 3)$, wenn es eine Grammatik G vom Typ i gibt mit $L = L(G)$.

Bemerkung: Nach Definition gilt:

$$L \underbrace{\text{vom Typ 3}}_{\text{rechtslinear}} \Rightarrow L \underbrace{\text{vom Typ 2}}_{\text{kontextfrei}} \Rightarrow L \underbrace{\text{vom Typ1}}_{\text{kontextsensitiv}} \Rightarrow L \underbrace{\text{vom Typ 0}}_{\text{rekursiv aufzählbar}}$$

Später: Keine der Umkehrungen gilt.

Satz:
L ist rechtslinear $\Leftrightarrow$ L regulär

Beweis:

„$\Rightarrow$“: Sei G rechtslinear $L = L(G)$, $G = (N, \Sigma, P, S)$

$w \in L(G) \Leftrightarrow S \vdash w_1 B_1 \vdash w_1 w_2 B_2 + \ldots \vdash w_1 \ldots w_{n-1} B_{n-1} + \underbrace{w_1 \ldots w_n}_{w}$

$B_i \to w_{i+1} B\{i+1\} \in P$ $(1 \leq i < n-1)$
$B_{n-1} \to w_n \in P$

Nutze Ableitung als Lauf eines NEA mit Worttransitionen

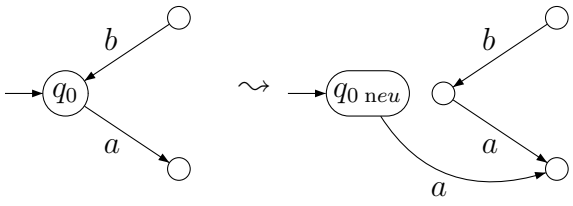
$\mathcal{A} = (N \cup \{\Omega\}, \Sigma, S, \Delta, \{\Omega\})$ mit:

- $(A, w, B) \in \Delta :\Leftrightarrow A \to wB \in P$
- $(A, w, \Omega) \in \Delta :\Leftrightarrow A \to w \in P$

Dann $L(\mathcal{N}) = L(G)$

„$\Leftarrow$“: Sei NEA $\mathcal{N} = (Q, \Sigma, q_0, \Delta, F)$

o.B.d.A. keine Transition nach q_0 vorhanden



$\mathcal{N}$ akzeptiert $w = a_1 \dots a_n \Leftrightarrow$ es existiert ein Lauf $q_0 a_1 q_1 \dots a_n q_n \in F$.

Definiere $G = (N, \Sigma, P, S)$ durch $N = Q, S = q_0$

$p \to aq \in P :\Leftarrow (p, a, q) \in \Delta$

Außerdem $q_0 \to \varepsilon$ falls $q_0 \in F$ (ε-Bedingung erfüllt)

$p \to a \in P \Leftrightarrow (p, a, q)$ mit $q \in F$

2. Kontextfreie Grammatiken: Elementare Eigenschaften

(a) Normalformen, Vereinfachungsregeln

(b) Ableitungsbäume und Anwendungen

$$\begin{array}{ll} A \to w & A \to a \\ A \to wB & A \to BC \end{array}$$

Definition: Eine kontextfreie Grammatik liegt in Chomsky-Normalform (CNF) vor, wenn sie nur Regeln der Form

- $X \to a$
- $X \to YZ$

hat, und gegebenenfalls auch $S \to \varepsilon$

Satz: Jede kontextfreie Grammatik kann man in eine äquivalente Grammatik in CNF überführen.

Beweis: Verfahren in 3 Etappen:

1. Auftreten von Terminalsymbolen nur in Regeln $X \to a$
   [Es verbleiben dann Regeln: $X \to X_1 \dots X_m$ $(m \geq 1)$]
2. Elimination von Regeln $X \to Y$
3. Elimination von Regeln $X \to X_1 \dots X_m$ mit $m > 2$

Etappe 1

- Führe für jedes Terminalsymbol a eine Zugehörige Variable X_a ein.
- Ersetze in allen Regeln a durch X_a.
- Füge die Regel $X_a \to a$ hinzu.
- Erhalte aus G damit die Grammatik G':
  $S \vdash_G^* a_1 \dots a_n \Leftrightarrow S \vdash_{G'}^* X_{a_1} \dots X_{a_n} \vdash_{G'}^n a_1 \dots a_n$

Etappe 2

Elimination der Regeln $X \to Y$:

(*) Bestimme alle Ableitungen $X_1 \vdash X_2 \vdash \dots X_k \vdash \alpha$ mit

- $\alpha \notin N$ oder $|\alpha| \geq 2$ oder $\alpha \in \Sigma$
- $X_i \neq X_j \ \forall \ i, j \in 1 \dots k, \ i \neq j$
- $k \leq |N|$

Alle endlich vielen Ableitungen (*) algorithmisch herstellbar

Lösche alle Regeln $X \to Y$ und füge alle Regeln $X_1 \to \alpha$ gemäß (*) hinzu.

Erhalte aus G' die Grammatik G''

Behauptung: $L(G) = L(G'')$

Beispiel:

$S \to ASA \mid aB$

$A \to B \mid S \mid a$

$B \to b$

1. Etappe:

$S \to ASA \mid X_aB$

$A \to B \mid S \mid X_a$

$B \to X_b$

$X_a \to a$

$X_b \to b$

2. Etappe:

Ableitungen gemäß (*) $A \begin{cases} \vdash B \vdash X_b \vdash b \\ \vdash X_a \vdash a \\ \vdash S \begin{cases} \vdash ASA \\ \vdash X_aB \end{cases} \end{cases}$

$B \vdash X_b \vdash b$

neue Regeln:

$A \to b$

$A \to ASA$

$A \to X_a B$

$A \to a$

$B \to b$

Ergebnis:

$S \to ASA \mid X_a B$

$A \to B \mid ASA \mid X_a B \mid a$

$B \to b$

$X_a \to a$

$\to X_b \to b$

3. Etappe: Elimination der Regel $X \to X_1 \dots X_m$ mit $m > 2$

Beispiel:

$S \to ASA$

$S \to AS_1$

$S_1 \to SA$

$A \to ASA$

$A \to AS_2$

$S_2 \to SA$

Allgemein: Führe für jede Regel $X \to X_1 \dots X_m$ jeweils neue Hilfsvariablen $Y_1 \dots Y_{m-2}$ ein. Ersetze $X \to X_1 \dots X_m$ durch

$X \to X_1 Y_1$

$Y_1 \to X_2 Y_2$

$\vdots$

$Y_{m-3} \to X_{m-2} Y_{m-2}$

$Y_{m-2} \to X_{m-1} X_m$

Erhalte aus G'' die Grammatik G''' in CNF. $L(G'') = L(G''')$

Somit ist G in CNF überführt.

Beweis zu $L(G'') = L(G''')$: Zeige für jedes Nichtterminalsymbol Z von G'':

$Z \vdash^*_{G''} w \Leftrightarrow Z \vdash^*_{G'''} w$

„⇒“: für $n \geq 1$:

$Z \vdash_{G''}^{\leq n} w \Rightarrow Z \vdash_{G'''}^{*} w$

Induktionsanfang $n = 1$:

$Z \vdash_{G''}^{\leq 1} w$ nur mit Regel $Z \to a$ $(= w)$ möglich auch in G''' möglich

Induktionsschritt:

Zeige $Z \vdash_{G''}^{\leq n+1} w \Rightarrow Z \vdash_{G'''}^{*} w$

G'' Ableitung: $Z \vdash_{G''} X_1 \dots \underbrace{X_m}_{m \geq 2} \vdash_{G''}^{*} \underbrace{n_1 n_2 \dots n_m}_{w} X_i \vdash_{G''}^{\leq n} n_i$

Nach I.V. gilt $X_1 \dots X_m \vdash_{G'''}^{*} \underbrace{n_1 \dots n_m}_{w}$.

Für die Ableitung in Richtung $Z \vdash X_1 \dots X_m$ nehme vorhandene Regel im Fall $m = 2$ und neuen Regel gemäß Definition von G''' im Fall $m > 2$.

„⇐“: Zeige: $Z \vdash_{G'''} \dots \vdash_{G'''} w \Rightarrow Z \vdash_{G''}^{*} w$

In G'''-Ableitung ersetze auftretende Hilfsvariablen (die in Reihenfolge gemäß Konstruktion müssen) durch zugrunde G''-Regel. $\alpha X \beta \vdash \alpha X_1 Y_1 \beta \vdash \dots X_2 Y_2$

$X \to a$

$X \to a$

$X \to YZ$

$X \to uYv$

Definition

Eine Grammatik mit Regeln der Form $X \to u$, $X \to uYv$ (neben $S \to \varepsilon$) heißt linear.

Beispiel: $\{a^n b^n \mid n > 0\}$ ist linear

Nehme Grammatik $S \to ab \mid aSb$

Behauptung (ohne Beweis): Es gibt kontextfreie Sprachen, die nicht linear sind. **Beispiel:** $\{a^n b^n c^m d^m \mid m, n > 0\}$

$S \to AB$

$A \to ab \mid aAb$

$B \to cd \mid cBd$

Eine kontextfreie Grammatik ist in Greibach-Normalform, wenn ihre Regeln die Form $X \to aY_1 \dots Y_m$ $(m \geq 0)$ haben (eventuell außerdem $S \to \varepsilon$).

Satz: Jede kontextfreie Grammatik kann man in eine äquivalente Grammatik Greibach-Normalform überführen.

6.1 Vereinfachung kontextfreier Grammatiken

Definition: eine Variable X einer kontextfreien Grammatik $G = (N, \Sigma, P, S)$ heißt

- erreichbar, wenn eine Satzform $\alpha X \beta$ in G ableitbar ist, kurz $S \vdash_G^* \alpha X \beta$
- terminierend, falls $X \vdash_G^* u$ mit $u \in \Sigma^*$

Bemerkung: Streicht man aus G alle Regeln, in denen eine nicht-erreichbare oder nicht-terminierende Variable auftritt und erhält so die Grammatik G', dann gilt $L(G) = L(G')$.

Man nennt G' reduzierte Grammatik zu G.

Zur Bestimmung der terminierende und erreichbaren Variablen.

Beispiel:

$\underline{S} \to A\underline{aB} \mid \underline{aB}$

$A \to AA \mid A\underline{bb}$

$\underline{B} \to S\underline{cc} \mid A \mid \underline{DD}$

$C \to \underline{Ea}S \mid \underline{SS}$

$\underline{D} \to \underline{S}A\underline{B} \mid \underline{bE}$

$\underline{E} \to \underline{aab}$

Unterstreiche sukzessiv terminierende Variablen.

Algorithmus zur Bestimmung der terminierenden Variablen:

1. Markiere (durch Unterstreichen) auf den rechten Regelseiten alle Terminalsymbole
2. Solange Regel $X \to \alpha$ mit X unmarkiert und α voll markiert existiert: markiere X überall

Da nur endlich viele Variablen existieren, terminiert der Algorithmus.

Behauptung: Die bei Termination des Algorithmus markierten Variablen sind genau die terminierenden.

Folgerung: Das Nicht-Leerheitsproblem für kontextfreie Grammatiken „Gegeben G, gilt $L(G) \neq \emptyset$?“ ist entscheidbar.

Beachte: $L(G) \notin \emptyset \Leftrightarrow S \vdash_G^* w$ für ein $w \Leftrightarrow S$ terminierende Variable $\underbrace{\Leftrightarrow}_{Algorithmus}$ im Markierungsalgorithmus wird S unterstrichen

Korrektheit: Algorithmus markiert $X \Leftrightarrow X$ terminierend 25.6.'04

„⇒“: Jede markierte Variable ist terminierend

Zeige: Wird X im n-ten Durchlauf (durch 1., 2.) markiert dann X terminierend

Induktion über $n \geq 1$:

Induktionsanfang: $n = 1$

Regel $X \to w$ vorhanden. Also X terminierend.

$n + 1$: Gemäß Algorithmus existiert Regel $X \to \alpha$, wobei in α jede Variable in $\leq n$ Durchläufen markiert ist. Nach I.V.: jede dieser Variablen terminiert.

Also mit $X \to \alpha$ auch X terminierend.

„⇐“: Jede terminierende Variable wird markiert.

Annahme: Existiert Variable X, die terminierend ist, aber nicht markiert wird.

Wähle solches X mit kürzester Ableitung eines Terminalwortes.

Betrachte derartige Ableitung: $X \vdash \alpha \vdash \ldots \vdash w$

Jede Variable in α terminierend, nach Wahl der Ableitung auch markiert. Also wird Algorithmus anhand Regel $X \to \alpha \Rightarrow X$ markieren. Widerspruch!

Analog bestimmt man die erreichbaren Variablen (von S aus):

- Markiere zunächst S
- Markiere jeweils X, falls X auf rechter Seite einer Regel $y \to \alpha x$ vorkommt, so dass y markiert ist.

6.2 Ableitungsbäume

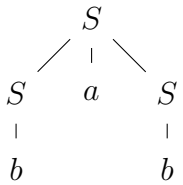
Beispiel:

$S \to b \mid SaS$ [hier Richtung ablesbar: erst links, dann rechts]

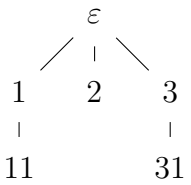
$S \vdash SaS \vdash baS \vdash bab$

$S \vdash SaS \vdash Sab \vdash bab$

Ableitungsbaum: Baum t [in Ableitungsbaum keine Richtung ablesbar]



Baumbereich: $dom(t)$ [für Domain]



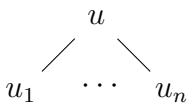
Vorbereitung: Ein höchstens k-verzweigter Baumbereich ist eine Menge $D \subseteq \{1, \ldots, k\}^*$ [D für Domain] so dass D präfixabgeschlossen ist ($uv \in D \Rightarrow u \in D$).

$u_j \in D,\ 1 \leq i < j \Rightarrow u_i \in D$

Ein W-beschrifteter Baum ist eine Abbildung $t : dom(t) \to W$ (so dass $t(u) =$ Beschriftung von u durch Wert $\in W$)

Ein Ableitungsbaum zur Grammatik $G = (N, \Sigma, P, S)$ ist ein $(N \cup \Sigma)$-beschrifteter Baum mit:

- $t(\varepsilon) \in N$
- Hat der Knoten $n \in dom(t)$ die Söhne $u_1, \ldots, u_n$, dann $t(u) \to t(u_1) \ldots t(u_n)$. [Regel in P]



- $Yield(t) \in \Sigma^*$

Terminologie:

Wurzel: ε

Blatt: Knoten ohne Nachfolger

Pfad: Folge $u_0 \ldots u_r$ mit $u_0 = \varepsilon$, u_r Blatt und u_{i+1} Sohn von u_i

Länge (des oben beschriebenen Pfades): r

Höhe von t: Länge eines längsten Pfades

Front: Folge der Blätter von links nach rechts

Yield (Ergebnis): Folge der Beschriftungen der Blätter von links nach rechts

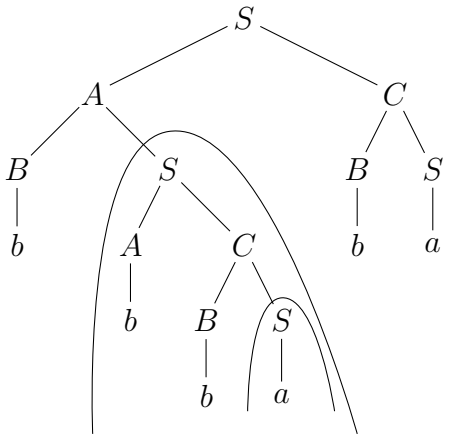
Lemma:

Ist G kontextfreie Grammatik, so gilt $(w \neq \varepsilon)$ $w \in L(G) \Leftrightarrow$ es existiert ein Ableitungsbaum t zu G mit $t(\varepsilon) = S$, $Yield(t) = w$.

Höhe $h \Rightarrow |Yield| \leq k^h$

$|Yield| > k^h \Rightarrow$ Höhe $> h$ 29.6.'04

$S \to AC \mid a$
$A \to BS \mid b$
$B \to b$
$C \to BS$



Ableitungslemma: $G = (N, \Sigma, P, S)$

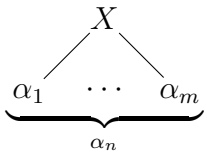
$S \vdash_G^* \iff$ es existiert Ableitungsbaum zu G mit Wurzelbeschriftung S und Frontbeschriftung w.

Beweis: Induktion über $n \geq 1$:

Wenn $X \vdash_G^{\leq n} w \Rightarrow$ existiert Ableitungsbaum mit Wurzel X und Frontbeschriftung α_n.

$n = 1$**:**

$X \vdash w$, Regel $X \to w$ vorhanden



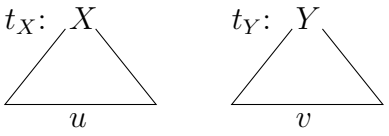
$n + 1$**:** Gelte $X \vdash_G^{\leq n+1} w$

Also Regel $X \to \alpha$ vorhanden: $X \vdash \alpha \vdash^{\leq n} w$

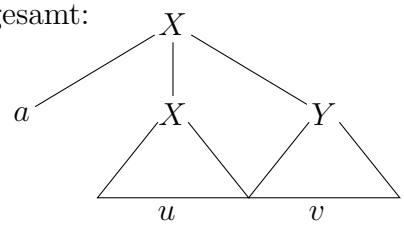
Beispiel: $\alpha = aXY$

Dann $w = auv$, $X \vdash^{\leq n} u, Y \vdash^{\leq n} v$

I.V. liefert Ableitungsbäume



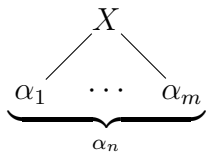
Insgesamt:



„$\Leftarrow$“

Ist t ein Ableitungsbaum der Höhe n mit Wurzelbeschriftung X, Frontbeschriftung $w \Rightarrow X \vdash^* w$ (für $n \geq 1$)

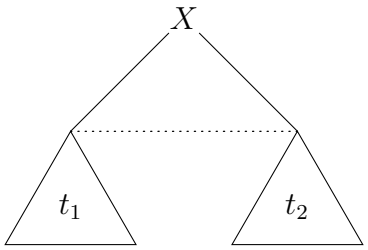
$n = 1$**:** Baum hat Form



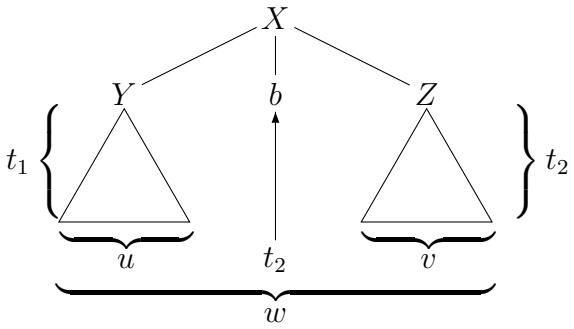
auf Grund der Regel $X \to \alpha_1 \dots \alpha_m$

Also $X \vdash \alpha_1 \dots \alpha_m$

$n + 1$**:** Baum hat Form:



Beispiel: $k = 3$



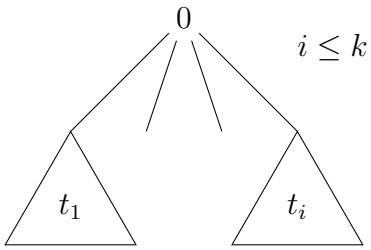
Nach I.V. $Y \vdash^* n$ $Z \vdash^* v$ Also $X \vdash YbZ \vdash^* \underbrace{ubv}_{w}$

Verzweigungslemma: Hat ein $\leq$ k-verzweigter Baum die Höhe m, so hat er $\leq k^m$ Blätter

Beweis: Induktion über m

$m = 0$ Blattzahl: $1 = k^0$

$m + 1$ Baum hat Form



mit Höhe$(t_i) \leq m$

I.V. liefert jeweils $\leq k^m$ Blätter für jedes der t_i Gesamtzahl der Blätter: $\leq i * k^m \leq k^{m+1}$

Konsequenz: hat $\leq k$-verzweigter Baum $> k^m$ Blätter, dann ist die Höhe $> m$

Pumping Lemma für kontextfreie Sprachen: ohne Regeln der Form $X \to Y$

Dann existiert eine Zahl n, so dass jedes Wort $z \in L(G)$ mit $|z| > n$ zerlegbar ist in der Form $z = uvwxy$ mit:

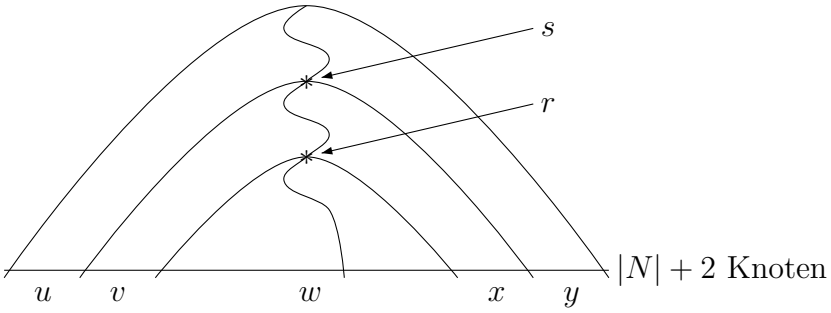
- $uv^iwx^iy \in L(G)$ für $i \geq 0$
- $vx \neq \varepsilon$ (v oder x oder beiden $\neq \varepsilon$)
- $|vwx| \leq n$

[Zusatz: Man kann $n = k^{|N|+1}$ wählen mit $k =$ max. Länge einer rechten Regelsätze]

Beweis: Setze n wie im Zusatz fest. Betrachte $z \in L(G)$ mit $|z| \geq n > k^{|N|+1}$. Nach Verzweigungslemma hat z einen Ableitungsbaum aber Höhe $> |N| + 1$

Wähle Pfad der Länge größer als $|N| + 1$, mit mehr als $|N| + 2$ Knoten. Somit sind mehr als $|N| + 1$ durch Variablen beschriftet. Somit tritt eine Variable X zweifach auf.

Wähle von Front aus auf Pfad die erste Stelle s, wo eine schon zuvor aufgetretene Variable wieder auftritt.



Erhalte Zerlegung des Ableitungsbaumes an diesen Stellen s und r. Betrachte die Unterbäume $t|s$, $t|r$. (*) $t|s$ hat die Höhe $\leq |N| + 1$

Frontbeschriftung von $t|r$ sei w

Frontbeschriftung von $t|s$ sei vwx

Frontbeschriftung von t insgesamt sei $z = uvwxy$

- Klar gemäß Zerlegung: $uv^iwx^iy \in L(G)$
- Wäre $v = x = \varepsilon$, dann besteht der Ableitungsabschnitt zwischen den beiden Punkten r, s nur aus Schritten mit Regeln $X_1 \to X_2$ (verboten)
- wegen (*) $|vwx| \leq \underbrace{k^{|N|+1}}_{n}$

1. Anwendung: $\{a^nb^nc^n | n \geq 1\}$ ist nicht kontextfrei

Beweis: Angenommen L ist kontextfrei und erzeugt etwa durch G.

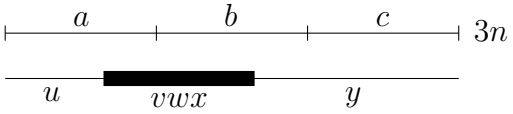
(gemäß Beweis CNF ohne Regel $X \to Y$)

Wähle n gemäß Pumping Lemma.

Betrachte $z = a^nb^nc^n$ für dieses n: $z \in L(G)$, $|z| > n$

Nach Pumping Lemma wähle Zerlegung $z = uvwxy$ mit

- $uv^iwx^iy \in L$ für $i \geq 0$
- $vx \neq \varepsilon$
- $|vwx| \leq n$



1. Fall Alle c sind im Teil y
2. Fall Alle a sind im Teil u

Im 1. Fall:

wx hat wenigstens ein a oder ein b. Mit uv^iwx^iy für $i > 1$ erhalten wir also mehr a oder mehr b aber keine neuen c.

Also in diesem Fall $uv^iwx^iy \notin L(G)$. **Widerspruch** zur 1. Behauptung des Pumping Lemma.

2. Fall analog

Folgerung: Die Klasse der kontextfreien Sprachen ist nicht unter Durchschnitt abgeschlossen. [D.h. es existieren $L_1,\ L_2$ kontextfrei, mit $L_1 \cap L_2$ nicht kontextfrei.]

Beweis: Betrachte

$L_1 = \{\underline{a^n b^n} c^k \mid n, k > 0\}$ und
$L_2 = \{a^n \underline{b^k c^k} \mid n, k > 0\}$ kontextfrei.

$L_1 \cap L_2 = \{a^n b^n c^n \mid n > 0\}$ ist nicht kontextfrei.

L_1 ist kontextfrei:

$$S \to AC$$
$$A \to ab \mid aAb$$
$$C \to c \mid cC$$

L_2 analog.

Bemerkung: Sind $L_1,\ L_2$ kontextfrei, dann ist auch $L_1 \cup L_2$ kontextfrei.

Folgerung: Abschluss der Klasse der kontextfreien Sprachen unter Komplement ($\overline{\ }$) trifft nicht zu, sonst könnte man mit den Operationen $\cup$ und $\overline{\ }$ sowie der deMorgan'schen Regel die Abgeschlossenheit unter $\cap$ folgern, die nicht zutrifft.

weitere Anwendung: Sprachen über $\Sigma = \{a\}$. Sprache

$$L \subseteq \{a\}^* \quad \sim \quad \text{Zahlmenge } M \subseteq \mathbb{N}$$
$$a^i \in L \quad \Leftrightarrow \quad i \in M$$

Definition: M heißt arithmetische Progression, falls $M = \{k_0 + i \cdot k \mid i \geq 0\}$ für feste Zahlen $k_0, k,\ k > 0$.

Satz: Sei $M \subseteq \mathbb{N}$ eine Menge natürlicher Zahlen, die keine (unendliche) arithmetische Progression als Teilmenge enthält. Dann ist die Sprache $\{a^i \mid i \in M\}$ nicht kontextfrei.

[Umkehrschluß gilt und ist eine praktische Anwendung.]

2.7.'04 **Wir zeigen:**

Wenn eine Sprache $L \subseteq \{a\}^*$ kontextfrei ist, so enthält L eine arithmetische Progression.

Nach Pumping-Lemma wähle für L und betrachte $a^m \in L$ $(m > n)$. Das Pumping-Lemma liefert eine Zerlegung $a^m = uvwxy$ mit $uv^i wx^i y \in L$ und $vx \neq \varepsilon$.

Wähle k_0 so, daß $uwy = a^{k_o}$ und k so, daß $vx = a^k$. Dann ist $uvwxy = a^{k_0+k} \in L$ und $uv^i wx^i y = a^{k_0+i\cdot k} \in L$, also erhalten wir eine arithmetische Progression in L.

Anwendung:

- $\{a^i \mid i$ ist Zweierpotenz$\}$ ist nicht kontextfrei.

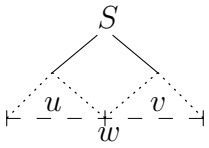
- $\{a^i \mid i \text{ ist Primzahl}\}$ ist nicht kontextfrei.

Wortproblem für kontextfreie Sprachen:

Gegeben: kontextfreie Grammatik $G = (N, \Sigma, P, S)$, Wort $w \in \Sigma^*$.

Frage: $w \in L(G)$?

Naive Idee: divide and conquer



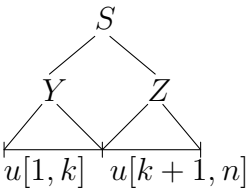
Probieransatz: Sei G in CNF (Chonsky-Normalform). Dann werden in Ableitung von $w = a_1 \dots a_n$ werden $n-1$ Schritte mit $X \to YZ$ verwendet (diese liefern n Nichtterminalsymbole). Zusätzlich außerdem n Anwendungen von Regeln $X \to a$.

Man kann alle Möglichkeiten von $2n-1$ Regelanwendungen durchprobieren.

Problem: Im Allgemeinen ist die Anzahl der Möglichkeiten exponentiell in n.

CYK-Algorithmus (Cooke, Younger, Kasami):

Zu $w = a_1 \dots a_n$ sei $w[i,j] = a_i \dots a_j$



$S \to YZ$

$N_{ij} := \{X \in N \mid X \vdash_G^* w[i,j]\}$

Idee des CYK-Algorithmus: Betrachte Segmente $w[i,j]$ nach wachsender Länge. Bestimme jeweils die Menge N_{ij} der Nichtterminalsymbole, aus denen man $w[i,j]$ ableiten kann.

Beginne dabei mir den Segmenten $w[i,i]$, d.h. mit den Einzelbuchstaben. Prüfe anschließend, ob S zu $N_{i,n}$ gehört (also zu den Nichtterminalsymbolen, aus denen $w = w[1,n]$ ableitbar ist).

Bestimmung der $N_{i,j}$:

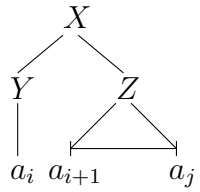
- Fall $i = j$: $X \in N_{ii} \Leftrightarrow X \vdash^* a_i \Leftrightarrow X \to a_i$ ist in P vorhanden.

- Fall $i < j$: $X \in N_{ij}$ $w[i,k] \Leftrightarrow X \vdash YZ$ mit $Y \vdash^* w[i,k]$ und $Z \vdash^* w[k+1,j]$ für geeignetes $k \in \{i, \ldots, j-1\}$
  Also: $X \in N_{ij} \Leftrightarrow$ es gibt Regel $X \to YZ$ und $k \in \{i, \ldots, j-1\}$ mit $Y \in N_{ik}$ und $Z \in N_{k+1,j}$

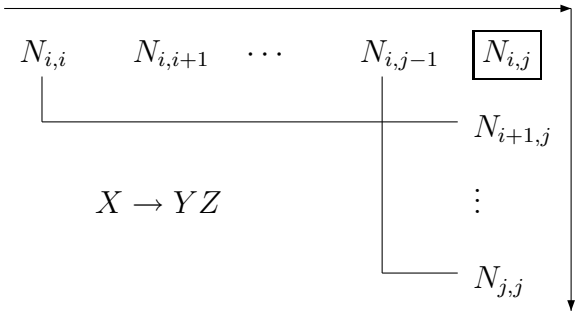
[dazu Folie „CYK-Algorithmus“, im Netz erreichbar]

Durchführung mittels Tabelle: Schema: Für ein Terminalwort der Länge n bestimmen wir die Mengen N_{ij} als Einträge einer $n \times n$-Matrix. Eintrag N_{ij} erfordert Rückgriff auf

- N_{ii} und $N_{i+1,j}$ (Fall $k = i$) Y in N_{ii}, $Z \in N_{i+1,j}$



- $N_{i,j+1}$ und $N_{i+2,j}$ (Fall $k = i+1$)
- $N_{i,j-1}$ und N_{ij} (Fall $k = j-1$)



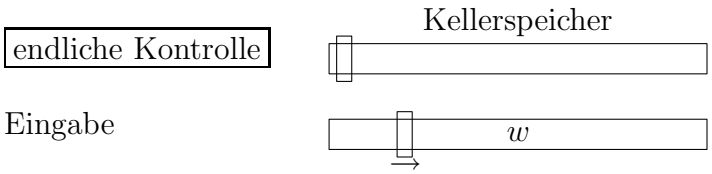
Beispiel: Gegeben: $G : S \to SA \mid a, A \to BS, B \to BB \mid BS \mid b \mid c$ $w = abaaba$

i/j	1	2	3	4	5	6
1	S	$\emptyset$	S	S	$\emptyset$	S
2		B	A, B	A, B	B	A, B
3			S	$\emptyset$	$\emptyset$	$\emptyset$
4				S	$\emptyset$	S
5					B	A, B
6						S

da $S \in N_{1,6} \Rightarrow w \in L(G)$

$$\frac{\text{endliche Automaten}}{\text{reguläre Sprachen}} = \frac{?}{\text{kontextfreie Sprachen}}$$

6.7.'04



7 Kellerautomaten(Pushdown Automaten)

Endliche Automaten mit Kellerspeicher(Stack). Zugriff auf den Speicher nur an der Spitze.

Beispiel: $L = \{a^i b^i \mid i > 0\}$

Akzeptieren durch:

- Aufbau des Kellers: Für jedes a lege ein Symbol (z.B. |) auf Keller ab
- Abbau des Kellers: Für jedes b nehme ein | vom Keller
- Am Wortende überprüfe, ob der Kellerboden erreicht.

Definition

Ein **Kellerautomat** (Pushdown-Automat, PDA) hat die Form:

$$\mathcal{A} = (Q, \Sigma, \Gamma, q_0, z_0, \Delta, F)$$

- endliche Zustandsmenge $Q, q_0 \in Q, F \subseteq Q$
- Eingabealphabet Σ, Kelleralphabet Γ, Kellerstartsymbol $z_0 \in \Gamma$
- Transitionen $(p, {}^{a}/_{\varepsilon}, v, q)$
  „in p, bei Lesen des Eingabebuchstaben a (bzw. ohne Verarbeitung des Eingabebuchstaben) und z als Kellerspitze, ersetze Z durch $v \in \Gamma^*$ und gehe nach q“
  D.h. $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times \Gamma^* \times Q$ ist endlich.

Beispiel: (siehe oben, für $\{a^i b^i \mid i > 0\}$)

$\mathcal{A}_0 = (Q, \Sigma, \Gamma, q_0, z_0, \Delta, F)$

- $Q = \{q_0, q_1, q_2\}$
- $F = \{q_2\}$

- $\Sigma = \{a, b\}$
- $\Gamma = \{z_0, z\}$ $\qquad z \sim |$
- Δ mit folgenden Transitionen
  1. (q_0, a, z_0, zz_0, q_0)
  2. (q_0, a, z, zz, q_0)
  3. $(q_0, b, z, \varepsilon, q_1)$
  4. $(q_0, b, z, \varepsilon, q_1)$
  5. $(q_1, \varepsilon, z_0, z_0, q_2)$

Beschreibung der Arbeitsweise eines PDA $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, z_0, \Delta, F)$

Konfiguration ist ein Tupel Zustand, Kellerinhalt, noch zu lesender Input)

$K(\kappa) = (qu, w) \in Q \cdot \Gamma^* \times \Sigma^*$

Definition der Übergangsrelation „$\vdash$“:

- $(pZu, aw) \vdash (qvu, w)$ falls $(p, a, Z, v, q) \in \Delta$
- $(pZu, w) \vdash (qvu, w)$ falls $(p, \varepsilon, z, r, q) \in \Delta$

$\kappa \vdash^n \kappa' :\Leftrightarrow$ es existiert $\kappa_0, \ldots, \kappa_n$ mit $\kappa_0 = \kappa, \kappa_i \vdash \kappa_{i+1} (i < n), \kappa_n = \kappa'$

$\kappa \vdash^* \kappa' :\Leftrightarrow$ es existiert $n \geq 0$ mit $\kappa \vdash^n \kappa'$

$\mathcal{A}$ akzeptiert $w :\Leftrightarrow (q_0 z_0, w) \vdash^* (qu, \varepsilon)$ für ein $q \in F$, $u \in \Gamma^*$ beliebig

$L(\mathcal{A}) := \{w \in \Sigma^* \mid \mathcal{A} \text{ akzeptiert } w\}$

Beispiel: $L(\mathcal{A}_0) = \{a^i b^i \mid i > 0\}$

Bemerkung:

„$\supseteq$“: Zu $a^i b^i$ betrachte folgende Konfigurationsfolge (Berechnung):

$$(q_0 z_0, a^i b^i) \underbrace{\vdash}_{(1)} (q_0 z z_0, a^{i-1} b^i) \underbrace{\vdash^*}_{(i-1) \text{ mal } (2)} (q_0 z^i z_0, b^i) \underbrace{\vdash}_{(3)} (q_1 z^{i-1} z_0, b^{i-1})$$

$$\underbrace{\vdash^*}_{(i-1) \text{ mal } (4)} (q_1 z_0, \varepsilon) \underbrace{\vdash}_{(5)} (q_2, z_0, \varepsilon)$$

„$\subseteq$“: $\mathcal{A}_0$ kann von $(q_0 z_0, w)$ nur in eine akzeptierende Konfiguration kommen $(q_2 u, \varepsilon)$, wenn Transition (1)…(5) in dieser Reihenfolge verwendet werden, (1),(3),(5) genau einmal. (2), (4) eventuell öfter oder gar nicht aber gleich häufig. Insgesamt hat Input die Form $a^i b^i$.

Beispiel: $(q_0 z_0, a^3 b^7) \vdash^* (q_1 z_0, b^4) \vdash (q_2 z_0, b^4)$, somit kann man nicht zu $(q_2 z_0, \varepsilon)$ kommen.

Beispiel: PAL $= \{ww^R \mid w \in \{a,b\}^*\}$

Idee $(abbbba)$

Kelleraufbau Lege die gelesenen Buchstaben im Keller ab

$(q_0 z_0, abbbbba) \vdash^3 (q_0, bbaz_0, bba)$

Kellerabbau Vergleich des Eingabebuchstaben mit Kellerspitze

$\vdash^3 (q_1, z_0, \varepsilon)$

Umsetzung als PDA: $\mathcal{A}_1 = (Q, \underbrace{\Sigma}_{\{a,b\}}, \Gamma, q_0, z_0, \Delta, F)$

- $Q = \{q_0, q_1, q_2\}$
- $F = \{q_2\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{z_0, a, b\}$
- Δ mit folgenden Transitionen
  - $(q_0, a, *, a*, q_0)* \in \Gamma$
  - $(q_0, b, *, b*, q_0)$
  - $(q_0, a, a, \varepsilon, q_1)$
  - $(q_0, b, b, \varepsilon, q_1)$
  - $(q_0, \varepsilon, z_0, z_0, q_1)$
  - $(q_1, a, a, \varepsilon, q_1)$
  - $(q_1, b, b, \varepsilon, q_1)$
  - $(q_1, \varepsilon, z_0, z_0, q_1)$

$L(\mathcal{B}_1) = Pal$

Beweis analog zu $\mathcal{A}_0$

Determinitische PDA's erlauben in jeder Konfiguration höchstens die Anwendung einer einzigen Transition: für jedes Paar $(p, Z) \in Q \times \Gamma$ gibt es höchstens eine passende Transition $(p, a/\varepsilon, Z, v, q)$
Nur eine Option erlaubt: ε-Schritt oder Buchstaben-Schritt.

Später: Pal nicht durch DPDA akzeptierbar.

Variante: Akzeptieren mit leerem Keller (Verzicht auf Komponente F)

PDA $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, Z_0, \Delta)$ akzeptiert w mit leeren Keller, falls

$$(q_0 Z_0, w) \vdash^* (q, \underbrace{\varepsilon}_{\text{Keller}}, \underbrace{\varepsilon}_{\text{Input ist abgearbeitet}})$$

$N(\mathcal{A}) := \{w \in \Sigma \mid \mathcal{A} \text{ akzeptiert } w \text{ mit leeren Keller}\}$

Satz: L ist PDA-akzeptierbar $\Leftrightarrow$ L ist PDA-akzeptierbar mit leerem Keller.

Beweis:

„$\Rightarrow$“ Übung

„$\Leftarrow$“ Gegeben PDA $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, Z_0, \Delta)$

Konstruiere $\mathcal{A}'$ so, dass die Feststelung „leerer Keller erreicht“ den übergang in einen eingeführten Endzustand q_f ermöglicht

$F = \{q_f\}$

Implementierung: Neues Kellersymbol y_0

Erster Schritt stellt Kellerinahlt $Z_0 Y_0$ her. Dann Durchführung $\mathcal{A}$-Schritt, bis Z_0 gelöscht wird. Dann mit $(q, \varepsilon, y_0, y_0, q_f)$ in den Endzustand wechseln. Erster Schritt wird durch gesonderten Anfangszustand erzwungen.

Ziel: L PDA-akzeptierbar $\Leftrightarrow$ L kontextfrei

Zuerst: von Grammatiken zu PDA's ($\Rightarrow$)

Vorbereitung: Linksableitungen

Eine Ableitung heißt Linksableitung, wenn in jedem Schritt das jeweils am weitesten links stehende NTS ersetzt wird.

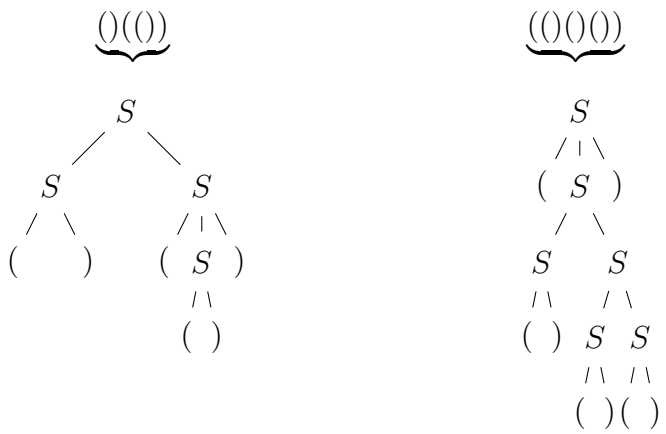
Bemerkung: $S \vdash^* w \Rightarrow w$ aus S mit Linksableitung

Beweis: Bilde Ableitungsbaum nach Vorlesung $S \vdash^* w$ und wende Regeln gemäß Tiefensuche durch den Baum an

Beispiel: $\Sigma = \{(,)\}$

Sprach der Klammerwörter erzeugt durch:

$$S \to () \mid (S) \mid SS$$



Linksableitung:

$$S \vdash SS \vdash ()S \vdash ()(S) \vdash ()(())$$

9.7.'04

Satz: Zu jeder kontextfreien Grammatik G kann man einen PDA $\mathcal{A}$ konstruieren mit $L(G) = N(\mathcal{A})$, und umgekehrt existiert zu jedem PDA $\mathcal{A}$ eine kontextfreie Grammatik G mit $L(G) = N(\mathcal{A})$.

Beispiel: $S \to () \mid (S) \mid SS$

Linksableitung:

$S \vdash (S) \vdash (SS) \vdash ((S)S) \vdash (((S) \vdash ((()())$

Restinput	Keller
(((())())	S
(((())())	$(S$
((())())	$S)$
((())())	$SS)$
((())())	$(S)S)$
())())	$S)S)$
())())	$())S)$
())	$S)$
())	$())$

Beweis: Von G zu PDA $\mathcal{A}$:

Start mit S im Keller. Erzeuge im Keller Satzformen von Linksableitung und lösche dann Terminalsymbole auf dem Kellerspitze und im Restinput bei Übereinstimmung.

Zu $G = (N, \Sigma, P, S)$ definiere PDA $\mathcal{A} = (Q, \Sigma, \Gamma, z_0, q_0, \Delta)$ mit $Q = \{q_0\}$, $\Gamma = N \cup \Sigma$, $z_0 = S$

Δ mit folgenden Transitionen:

- $(q_0, \varepsilon, X, \beta, q_0)$ wenn $X \to \beta$ in P
- $(q_0, a, a, \varepsilon, q_0)$ für jedes $a \in \Sigma$

Behauptung: $N(\mathcal{A}) = L(G)$

Zeige für jeden Kellerinhalt α, der nicht mit einem Terminalsymbol beginnt:

$$(q_0, S, u) \vdash_{\mathcal{A}}^* (q_0, \alpha, \varepsilon) \;\Leftrightarrow\; S \vdash_G^* u\alpha$$

Dann fertig mit $\alpha = \varepsilon$ $(u \in N(\mathcal{A})) \Leftrightarrow u \in L(G))$

„$\Rightarrow$“: $(q_0, S, u) \vdash_{\mathcal{A}}^n (q_0, \alpha, \varepsilon) \Rightarrow S \vdash_G^* u\alpha$

Induktion:

$n = 0$**:** $S = \alpha,\ u = \varepsilon\ \surd$

$n + 1$**:** Gelte $(q_0 S, u) \vdash_{\mathcal{A}}^{n+1} (q_0\alpha, \varepsilon)$

Isoliere letzten Schritt.

Fall1: letzter Schritt mit Transition für $X \to \beta$

$(q_0, S, u) \vdash_{\mathcal{A}}^n (q_0, X\alpha', \varepsilon) \vdash_{\mathcal{A}} (q_0, \beta\alpha', \varepsilon)$ mit (*) $\beta\alpha' = \alpha$

I.V. $S \vdash_G^* uX\alpha' \vdash u\beta\alpha' \underbrace{=}_{*} u\alpha$

Fall2: u-Schritt, $X \to \beta$ $u = \alpha' a : (q_0 S, \underbrace{aa}_{u} \vdash_{\mathcal{A}}^n (q_0\alpha, \varepsilon)$

[Hier fehlt leider etwas(und es ist glaube ich nicht 100% korrekt, wer es hat, bitte melden!]

I.V. $S \vdash_G^* ua\alpha = u\alpha$

Gelte $S \vdash_G^n u\alpha$. Zeige $(q_0 S, u) \vdash_{\mathcal{A}}^* (q_0\alpha, \varepsilon)$

Induktion über n**:**

„$\Leftarrow$“: $n = 0$**:** $u = \varepsilon, \alpha = S$

$n + 1$**:** Gelte $S \vdash_G^{n+1} u\alpha$ mit Regel $X \to \beta$ am Ende

Dann $S \vdash_G^n vX\gamma \vdash_G \underbrace{v\beta\gamma}_{u\alpha}$.

Also v ist Präfix von u. (α beginnt mit Nichtterminal, oder $\alpha = \varepsilon$) $(vw = u)$

I.V.: $(q_0, S, v) \vdash_{\mathcal{A}}^* (q_0, X\gamma, \varepsilon) \underbrace{\vdash_{\mathcal{A}}}_{Def\mathcal{A}} (q_0, \beta\gamma, \varepsilon)$

$(q_0 S, \underbrace{vw}_{u}) \vdash_{\mathcal{A}}^* (q_0, \beta\gamma, w) = (q_0, w\alpha, w) \vdash_{\mathcal{A}}^* (q_0, \alpha, \varepsilon)$

Bemerkungen zur Transformation $G \to \mathcal{A}$:

Fall der Greibach-Normalform $X \to aY_1 \dots Y_m$ liefert
Transitionen $(q_0, a, X, Y_1 \dots Y_m, q_0)$

Also: In PDA's sind ε-Transitionen vermeidbar.

Illustration: Aussagenlogischen Ausdrücke in polnischer Notation.

$S \to x0 \mid \ldots \mid x9 \mid \neg S \mid \wedge SS \mid \vee SS$

$S \to xZ \mid \neg S \mid \wedge SS \mid \vee SS \qquad Z \to 0 \mid \ldots \mid 9$

Sei der Proz. S.:

- bei Eingabe x rufe Z auf
- bei Eingabe $\wedge, \vee$ rufe SS auf
- bei Eingabe $\neg$ rufe S auf

Implementierung durch iterative Programm mit Stack wird durch die PDA Konstruktion geliefert. Anfangs ist S auf Keller, gemäß Eingabe $x, \neg\wedge, \vee$ ersetze Kellerspitze S durch Z, S, SS.

Von PDA's zu Grammatiken

Gegeben sei $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, z_0, \Delta)$.

Wir führen Nichtterminalsymbole $[pXq]$ ein, mit $p, q \in Q, X \in N$

Erreiche $[pXq] \vdash^*_{G?} u \Leftrightarrow \mathcal{A}$ kann mit Eingabe a von p nach qw unter Löschen des X vom Keller gelangen. $(pX, u) \vdash^*_{\mathcal{A}} (q, \varepsilon)$

Verwende zusätzlich Startsymbol S.

Regeln:

$S \to [q_0 Z_0 q]$ für jedes $q \in Q$

[Beispiel: PDA-Transition (p,a,Z,XY,q). PDA kann dann durch Lesen von in p'-Zustand kommen und danach mit lesen der Resteingabe X, Y von Kellerspitze löschen]

$[pZq] \to a[p'Xp_1][p_1Yq]$

Allgemein:

Für Transition $(p, a/\varepsilon, Z, Z_1 \ldots Z_m, q)$ führe ein: $[pZq] \to v[p'Z_1p_1] \ldots [p_{n+1}Z_mp_m]$

8 Unentscheidbarkeitsresultate

16.7.'04

Entscheidungsprobleme:

- DKG Durchschnittsproblem für kontextfreie Grammatiken

  Gegeben kontextfreie Grammatik. G_1, G_2 über Σ, gilt $L(G_1) \cap L(G_2) \neq \emptyset$?

- UKG Universalitätsproblem für kontextfreie Grammatiken.

  Gegeben kontextfreie Grammatik G, etwa über Σ, gilt $L(G) = \Sigma^*$

- ÄQU Äquivalenzproblem für kontextfreie Grammatiken.

  Gegeben kontextfreie Grammatiken G_1, G_2 über gemeinsamen Term alphabet Σ, gilt $L(G_1) = L(G_2)$

Ein Entscheidungsproblem P ist gegeben als ein Paar $(Inst(P), Pos(P))$ mit $Inst(P)$ (Instanzen zu P) Menge der Eingabe(durch Wörter kodierbar).
$Pos(P) \subseteq Inst(P)$ (Menge der Instanzen mit positiver Antwort)

Erinnerung:
$P = (Inst(P), Pos(P))$ ist entscheidbar, falls ein Algorithmus existiert, der zu Eingabe $p \in Inst(p)$ mit „ja“ für $p \in Pos(P)$, und sonst mit „nein“ terminiert.

8.1 Postsches Korrespondenzproblem

Post's Correspondance Problem (PCP) oder auch Modifiziertes Korrespondenzproblem (MPCP)

Gegeben: Wortlisten $X = (x_1, \ldots, x_k), Y = (y_1, \ldots, y_k)$ über dem gleichen Alphabet Σ.
Gibt es eine Indexfolge $i_1, \ldots, i_k$ $(1 \leq i_j \leq k)$ mit $x_{i_1}, \ldots, x_{i_n} = y_{i_1}, \ldots, y_{i_n}$?
(bei MPCP: Folge $1, i_2, \ldots, i_n$ mit $x_1, \ldots, x_{i_n} = y_1, \ldots, y_{i_n}$)

Solche Folge: „Lösung für X,Y“

Beispiel: $X = (abaaa, aba, a)$

$Y = (ab, b, aaa)$

Lösung zu X, Y für MPCP: $(1, 3, 3, 2)$

X: $abaaaaaab$

Y: $abaaaaaab$

Satz (von Post): MPCP (und PCP) unentscheidbar.

Reduktionsmethode: für Probleme $P = (Inst(P), Pos(P))$ und $Q = (Inst(Q), Pos(Q))$

$P \leq Q$ („P auf Q reduzierbar“): $\Leftrightarrow$ es existiert eine berechenbare Funktion
$F : Inst(P) \to Inst(Q)$ mit:

$p \in Pos(P) \Leftrightarrow F(p) \in Pos(Q)$

Lemma: Wenn P unentscheidbar und $P \leq Q \Rightarrow Q$ unentscheidbar.

Satz 1: DKG unenscheidbar

Beweis: Zeige MPCP $\leq$ DKG

Suche Transformation $(X,Y) \mapsto (G_1^{X,Y}, G_2^{X,Y})$

$$\boxed{(X,Y) \text{ hat Lösung} \Leftrightarrow L(G_1^{X,Y} \cap G_2^{X,Y} \neq \emptyset}$$

Konstruktion von $G_1^{X,Y}, G_2^{X,Y}$:

Gegeben $X = (x_1, \ldots, x_k)$, $Y = (y_1, \ldots, y_k)$ über Σ

$\Sigma_1 := \Sigma \cup \underbrace{\{\underline{1}, \ldots, \underline{k}\}}_{K} \cup \{\$\}$

$L_1^{X,Y} = \{v\$v^R \mid v \in \Sigma^+ \cdot K^+\}$

$G_1^{X,Y}$ ($\Sigma = \{a,b\}, K = \{\underline{1}, \underline{2}\}$):

$$S \to aSa|bSb|aAa|bAb$$
$$A \to \underline{1}A\underline{1} \mid \underline{2}A\underline{2} \mid \underline{1}\$\underline{1} \mid \underline{2}\$\underline{2}$$

Zu $L_2^{X,Y}, G_2^{X,Y}$

X-Code sei ein Wort $x_1 x_{i_2} \ldots x_{i_n} \underline{i_n} \ldots \underline{i_2}\underline{1}$

Y-Code sei ein Wort $y_1 y_{i_2} \ldots y_{i_n} \underline{i_n} \ldots \underline{i_2}\underline{1}$

$L_2^{X,Y} = \{v\$u^R \mid v \text{ ist } X\text{-Code}, u \text{ ist } Y\text{-Code}\}$

$G_2^{X,Y}$:

$$S \to A'\$B'$$
$$A' \to x_1\underline{1} \mid x_1 A\underline{1}$$
$$A \to x_1\underline{1} \mid x_2\underline{2} \mid x_1 A\underline{1} \mid x_2 A\underline{2}$$
$$B' \to \underline{1}y_1^R \mid \underline{1}By_1^R$$
$$B \to \underline{1}y_1^R \mid \underline{2}y_2^R \mid \underline{1}By_1^R \mid \underline{2}By_2^R$$

Für MPCP $\leq$ DKG fehlt:

(X,Y) hat eine Lösung $1, i_2, \ldots, i_n \Leftrightarrow L(G_1^{X,Y}) \cap L(G_2^{X,Y}) \neq \emptyset$

„$\Rightarrow$“ Gelte: $x_1 x_{i_2} \ldots x_{i_n} = y_1 y_{i_2} \ldots y_{i_n}$

Dann: $x_1 \ldots x_{i_n} \underline{i_n} \ldots \underline{i_2}\underline{1} \underbrace{=}_{+} y_1 \ldots y_{i_n} \underline{i_n} \ldots \underline{i_2}\underline{1}$

Betrachte

$x_1 \ldots x_{i_n} \underline{i_n} \ldots \underline{i_2}\underline{1}\$(y_1 \ldots y_{i_n} \underline{i_n} \ldots \underline{i_2}\underline{1})^R$

- $\in L_2^{X,Y}$ klar nach Def.
- $\in L_1^{X,Y}$ nach Def mit (+)

„$\Leftarrow$“ Gelte $w \in L_1^{X,Y} \cap L_2^{X,Y}$

Wegen $w \in L_2^{X,Y}$: $w = v\$u^R$ für X-Code v und Y-Code u gilt:

$$w = x_1 x_{i_2} \ldots x_{i_n} \underline{i_n} \ldots \underline{i_2}\underline{1}\$(y_1 y_{j_2} \ldots y_{j_n} \underline{j_n} \ldots \underline{j_2}\underline{1})^R$$

Da $w \in L_1^{X,Y}$ gilt:

- $n = r$
- $i_2 = j_2, \ldots, i_n = j_n = j_r$
- $x_1 x_{i_2} \ldots x_{i_n} = y_1 y_{i_2} \ldots y_{i_{r(i_n)}}$

Also $1, i_2 \ldots i_n$ Lösung zu X, Y.

Satz 2: UKG unentscheidbar

Beweis: Zeige DKG $\leq$ UKG

Bekannt (X, Y) hat eine Lösung $\Leftrightarrow L_1^{X,Y} \cap L_2^{X,Y} \neq \emptyset$.
Für $\Sigma_1 = \Sigma \cup K \cup \{\$\}$ gilt: $\Leftrightarrow \underbrace{\Sigma_1^* \setminus (L_1^{X,Y} \cap L_2^{X,Y})}_{(\Sigma_1^* \setminus L_1^{X,Y}) \cup (\Sigma_1^* \setminus L_2^{X,Y})} \neq \Sigma_1^*$

Es fehlt: Angabe von G mit $L(G) = (\Sigma_1^* \setminus L_1^{X,Y}) \cup (\Sigma_1^* \setminus L_2^{X,Y})$
finde G_1, G_2 für Einzelsprachen.

Zu G_1: $w \notin L_1^{X,Y} \Leftrightarrow$

1. $w \notin \Sigma^+ K^+ \$ K^+ \Sigma^+$ oder
2. 1. gilt oder vor/nach \$ ungleiche Länge oder
3. 1.,2. efüllt, aber vor/nach \$ ungleiche Buchstaben in gleichem Abstand

1., 2., 3. durch PDA's testbar, insgesamt G_1 wie gewünscht konstruierbar.

zu G_2: $w \notin L_2^{X,Y}$
$\Leftrightarrow w \notin X$-Code $(Y$-Code$)^R \Leftrightarrow$ 1. $w \notin \Sigma^+ K^+ \$ K^+ \Sigma^+$ oder 2. vor \$ kein X-Code oder nach \$ kein $(Y$-Code$)^R$
PDA für Test „u kein X-Code“ $\neq \underbrace{x_1, x_{i_2} \ldots x_{i_n}}\, \underline{i_n} \ldots \underline{i_2 1}$

Schreibe $x_1 \ldots x_{i_n}$ auf Keller, Kellerinhalt $x_{i_n}^R \ldots x_1^R$, Überprüfung anhand der Indizies $\underline{i_n} \ldots \underline{1}$ Akzeptiere, wenn „Fehler“ gefunden

20.7.'04 **Satz:** Folgende Probleme sind unentscheidbar:

DKG: Gegeben kfG G_1, G_2 über gemeinsamen Terminalalphabet Σ, gilt $L(G_1) \cap L(G_2) \neq \emptyset$?

UKG: Gegeben kontextfreie Grammatik G über Σ, gilt $L(G) = \Sigma^*$?

Beweisansatz: PCP $\leq$ DKG

PCP $\leq$ UKG

$(X, Y)| \rightarrow L_1^{X,Y}, L_2^{X,Y}$ mit $G_1^{X,Y} G_2^{X,Y}$

$L_1^{X,Y} = \{v\$v^R \mid v \in \Sigma^+ K^+\}$

$L_2^{X,Y} = \{v\$u^R \mid \text{v X-Code, u Y-Code}\}$

(X, Y) hat Lösung $\Leftrightarrow L_1^{X,Y} \cap L_2^{X,Y} \neq \emptyset$

Folgerung aus Unentscheidbarkeit von UKG:

Das Äquivalenzproblem für kontextfreie Grammatiken ist unenscheidbar.

Beweis: Betrachte $\Sigma = \{a, b\}$

$G_0 : S \to \varepsilon \mid A$
$A \to a \mid b \mid aA \mid bA$

Das UKG ist das spezielle Äquivalenzproblem

Gegeben G, gilt $L(G) = \underbrace{L(G_0)}_{\Sigma^*}$

Da UKG unenscheidbar, Äquivalenzproblem unentscheidbar.

Satz: Das Problem „Gegeben kontextfreie Grammatik G, ist $L(G)$ regulär?“ ist unentscheidbar.

Beweis: Benutze Reduktion PCP $\leq$ DKG und zeigen

$L_1^{X,Y} \cap L_2^{X,Y} \neq \emptyset \Leftrightarrow L_1^{X,Y} \cap L_2^{X,Y}$ nicht kontextfrei

Es genügt „$\Rightarrow$“ zu zeigen:

Sei $L_1^{X,Y} \cap L_2^{X,Y} \neq \emptyset$.

Annahme $L_1^{X,Y} \cap L_2^{X,Y}$ kontextfrei.

Betrachte kürzestes Wort im Druchschnitt

$\underline{uv}\$\underline{v^R u^R}$ mit $u \in \Sigma^+$, $v \in K^+$

$u^i v^i \$ (v^R)^i (u^R)^i \in L_1^{X,Y} \cap L_2^{X,Y}$

$L : \underbrace{\underbrace{L_1^{X,Y} \cap L_2^{X,Y}}_{\text{kontextfrei}} \cap \underbrace{u^* v^* \$ (v^R)^* (u^R)^*}_{\text{regulär}}}_{\text{kontextfrei (Beweis folgt)}} = \{u^i v^i \$ (v^R)^i (u^R)^i \mid i \geq 0\}\ [\approx \{a^i b^i c^i d^i \mid i \geq 0\}]$

Pumping Lemma für kontextfreie Sprachen: L nicht kontextfrei. Widerspruch!

Lemma: L kontextfrei, R regulär $\Rightarrow L \cap R$ kontextfrei

Beweis: Sei

- $\mathcal{A}$ PDA, der L akzeptiert
- $\mathcal{A}'$ DEA, der R akzeptiert

$\mathcal{A} = (Q, \Sigma, \Gamma, q_0, Z_0, \Delta, F)$
$\mathcal{A}' = (Q', \Sigma, q_0, \delta', F')$

Bilde Produkt-PDA $\mathcal{C} = (Q \times Q', \Sigma, \Gamma, (q_0, q_0'), Z_0, \Delta', F \times F')$ mit:
$((p, p'), a/\varepsilon, Z, v, (q, q')) \in \Delta' :\Leftrightarrow (p, a/\varepsilon, Z, v, q) \in \Delta$ und $\delta(p', a) = q'/p' = q'$

Synopse

Typ		Wortproblem	Nichtleerheitsproblem	Äquivalenzproblem
3	DEA's	+	+	+
2	PDA's/kf.G	+	+	-
1	Kont. sens. Gr.	$\oplus$		-
0	allg. Gr.			

Erinnerung: Kontextsensitive Grammatiken haben Regeln $\alpha \to \beta$ mit $|\alpha| \leq |\beta|$ (außer Regel $S \to \varepsilon$)

Satz: Das Wortproblem für kontextsensensitive Grammatik „Gegeben kontextsensitive Grammatik G, Wort w, gilt $w \in L(G)$?“ ist entscheidbar.

Beweis: In Ableitung eines Wortes w ($\neq \varepsilon$) kommen nur Satzformen der Länge $\leq |w|$ vor.

Bilde zu G, w einen Graphen (V, E) mit

- V = Menge aller Wörter über $(\Sigma \cup N)$ der Länge $\leq |w|$
- $E = \{(\alpha, \beta) \mid \alpha \vdash_G \beta\}$

(V, E) effektiv bestimmbar, da V endlich.

Dann gilt $w \in L(G) \Leftrightarrow S \vdash_G^* w \Leftrightarrow \underbrace{\text{in } (V, E) \text{ existiert Pfad von } S \text{ nach } w}_{\text{effektiv entscheidbar}}$

Kurz gesagt: Jede kontexstsensitive Sprache ist entscheidbar

Satz: Nicht jede entscheidbare Sprache ist kontextsensitiv

Beweis: Benutze eine Auflistung aller kontextfreien Grammatiken über $\Sigma = \{a, b\}$

Kodiere eine Grammatik durch Auflistung der Regeln, hierbei seien Nichtterminale kodiert in Form X Dezimalzahl

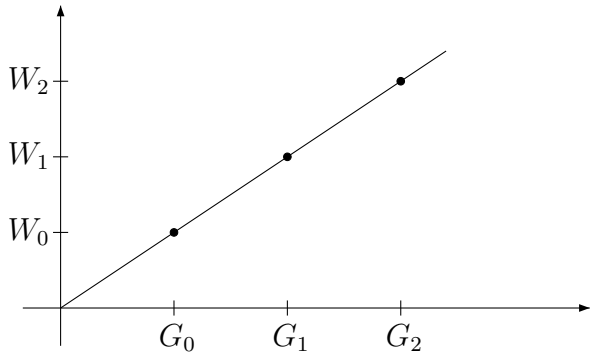
Alphabet: $\{X, 0, \ldots, 9, \to, ;, a, b, \underline{\varepsilon}\}$ genügt.

Benutze die kanonische Reihenfolge der Wörter über diesem Alphabet zur Aufzählung der Grammatiken:

$$G_0, G_1, G_2, G_3, \ldots$$

Wörter über $\{a, b\}$ ebenfalls auflistbar in kanonischer Reihenfolge:

$$w_0, w_1, w_2, w_3, \ldots$$



$(**) D := \{w_i \mid w_i \notin L(G_i)\}$

1. D ist enschteidbar
2. D ist nicht kontextsensitiv

zu 1) Gegeben w, entscheide ob $w \in D$:

Finde das i mit $w = w_i$.

Bilde G_i.

Nach Enscheidbarkeit des Wortproblems kann man testen, ob $w_i \in L(G_i)$. Damit klar, ob $w_i \in D$

zu 2) **Annahme:** D ist kontextsensitiv.

Dann existiert G_n aus Liste der konstextsensitiven Grammatiken welche D erzeugt.

$(*) D = L(G_n)$

Betrachte w_n

$w_n \in L(G_n) \underbrace{\Leftrightarrow}_{(*)} w_n \in D \underbrace{\Leftrightarrow}_{(**)} w_n \notin L(G_n)$

Widerspruch!

9 Erweiterte Formen der Sprachdefinition

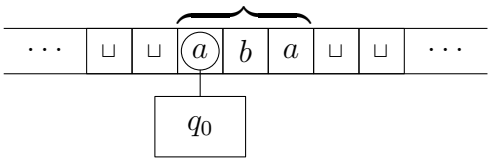
9.1 Turingmaschinen als Akzeptoren

$$\frac{\text{DEA's}}{\text{reg. Sprachen}} = \frac{\text{PDA}}{\text{kontextfreie Sprachen}} = \frac{\text{X}}{\text{kontextsensitive Sprachen}} = \frac{\text{Y}}{\text{Typ0-Sprachen}}$$

Definition der Turingmaschinen (TM):

TM hat das Format $\mathcal{A}$ mit

- Q der Zustandsmenge
- Σ dem Eingabealphabet
- Γ dem Arbeitsalphabet
- q_0 dem Anfangszustand
- Δ
- q_+ dem akzeptierenden Zustand



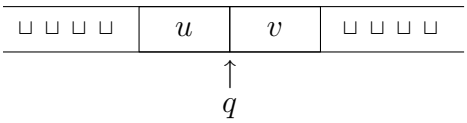
mit $\Sigma \cup \{\sqcup\} \subseteq \Gamma$

Transition in Δ hat die Form:

$$(p, a, a', L/N/R, q)$$

„Wenn man sich im Zustand p befindet und auf dem Arbeitsband ein a gelesen wird, so ersetze a durch a' und bewege den Kopf ein Feld nach links/rechts/garnicht und gehe in Zustand q“

Konfiguration hat die Form uqv mit $u, v \in \Gamma^*, q \in Q$



Für Konfigurationen C, C' gelte $C \vdash C'$ falls C' durch Anwendung einer Transition aus C entsteht.

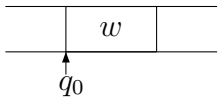
Beispiel: Sei $C =_\sqcup abq_\sqcup b$ und sei $(q,_\sqcup, a, R, p) \in \Delta$ so folgt:

$$C' =_\sqcup abapb$$

[Die Vollständige Defintion einer Konfiguration einer TM wird in Netz gestellt.]

$\mathcal{A}$ **akzeptiert** $w \in \Sigma^* \Leftrightarrow$ es existiert eine Konfigurationsfolge $C_0, \ldots, C_k$ mit folgenden Eigenschaften:

- $C_0 = q_0 w$
- $C_i \vdash C_{i+1}$



- C_k ist „akzeptierende Stopkonfiguration“ der Form nq_+v und kein weiterer Schritt durchführbar

$L(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A} \text{ akzeptiert w}\}$ die von TM akzeptierte Sprache

Einschränkung des Modells: Linear beschränkte Automaten (LBA)

Ein LBA ist eine TM mit folgenden Zusatzkonventionen:

- Γ enthält „[“ und „]“
- TM kann „[“ und „]“ weder verändern noch überschreiben.

Realisierung durch folgende Einschränkung der Transitionen.

$(p,],], L, q)$ - - - - - - - - -]
$(p, [, [, R, q)$ [- - - - - - - - -

$L(\mathcal{A})$ definiert wie zuvor

Satz:

a) L ist durch TM akzeptierbar $\Leftrightarrow$ L ist vom Typ0 (durch allgemeine Grammatik erzeugbar)

b) L ist durch LBA akzeptierbar $\Leftrightarrow$ L ist von Typ1 (durch kontextsensitive Grammatik erzeugbar)

Satz: 23.7.'04

a) Eine Sprache ist vom Typ 0 $\Leftrightarrow$ L wird durch eine TM akzeptiert

b) Eine Sprache ist kontextsensitiv $\Leftrightarrow$ L wird durch LBA akzeptiert

Beweis zu a):

„$\Rightarrow$“ Gegeben Grammatik $G = (N, \Sigma, P, S)$

Gesuchte TM tut folgendes:

Behalte die Eingabe b auf dem Band und erzeuge (z.B.) rechts davon, ausgehend von S, sukzessiv Satzformen durch Regelanwendungen. Nach Herstellung neuer Satzformen jeweils, ob Terminalwort vorhanden erzeugt und in diesem Falle Vergleich mit w.

Bei Gleichheit der Satzformen mit w, Stop mit Akzeptieren.

„⇐“ Gegeben sei eine Turingmaschine $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, \Delta, q_+)$

Gesuchte Grammatik muss w erzeugen können $\Leftrightarrow$ $\mathcal{A}$ das Wort w akzeptiert.

Idee: Stelle $a_1 \dots a_n$ nicht deterministisch her.

Gehe zu Satzform $\Big[\underbrace{Q_0}_{Nichtdet.} \binom{a_1}{a_1}\binom{a_2}{a_2}\binom{a_3}{a_3}\dots\binom{a_n}{a_n}\Big]$

Simuliere Arbeitsweise von $\mathcal{A}$ und erhalte Sazformen wie:

$$\left[\binom{\sqcup}{b}\binom{a_1}{a_1}\binom{a_2}{c}Q_1\binom{a_3}{a}\dots\binom{a_n}{a_5}\binom{\sqcup}{\sqcup}\binom{\sqcup}{b}\right]$$

Bei Erreichen von Q_+ und Fehlen einer Folgekonf. (TM $\mathcal{A}$ akzeptiert)

Übergang zu $a_1 \dots a_n$

Details:

1. Phase $\rightarrow A]$ $\quad A \rightarrow A \underbrace{\binom{a}{a}}_{a\in\Sigma} \mid [Q_0$
2. Phase Für TM-Trans. $(q_1, a, b, R, q_3) \Rightarrow$ **Regel** $Q_1\binom{*}{a} \rightarrow \binom{*}{b}Q_3$
3. Phase
   $C\binom{\sqcup}{*} \rightarrow C$
   $\binom{a}{*}^{C}\rightarrow aC)$

Beweis zu b):

„⇒“ von kontextsensitiven Grammatik zu TM

Auf. konf. der TM $q_0a_1 \dots a_n$

Übergang zu $[q\binom{a_1}{S}\binom{a_2}{\sqcup}\dots\binom{a_n}{sqcup}]$

Erzeugung der Satzformen in 2. Komp.

Akz. bei Inschrift $[\binom{a_1}{a_1}\dots\binom{a_n}{a_n}]$

„⇐“ Von LBA zu kontextsensitiven Grammatiken.

Stelle anfangssymbol her:

$\underbrace{([\underbrace{Q_0\binom{a_1}{a_1}}\ \underbrace{\text{Nicht T.}\binom{a_2}{a_2}\dots\binom{a_n}{a_n}}_{Nichtterminal-Symbol}])}_{n \text{ Symbole}}$ Analog wie Typ0

13.0

$\underbrace{L \text{ vom Typ } 0}_{\text{aufzählbar}} \Leftrightarrow L$ durch TM akz. $\underbrace{L \text{ vom Typ } 1}_{kontextsenitiv} \Leftrightarrow L$ durch LBA akz.

zu Typ 0 - Sprachen

Eine Aufzählungs-TM ist eine deterministische TM, die auf dem leeren Band gestartet wird und einen Zustand q! hat. Bei Besuch von q! wird das Wort vom aktuellen Arbeitsfeld vis zum nächsten $_sqcup$ ausschlielich als Ausgabe gewertet.

Die aufgezählte Sprache = Menge der so ausgegeben Wörter.

Satz

L vom Typ 0 $\Leftrightarrow$ L ist durch Aufzählung-TM erzeugbar.

Beweis:

$\Rightarrow$ Gegeben Grammatik G für L

Aufzählungs TM erzeugt sukzessiv Ableitungen

$$S \vdash \alpha_1 \vdash \alpha_2 \vdash ... \vdash \alpha_k$$

für wachsendes k. (Für festes k jeweils alle solchen Ableitungen herstellbar) Bei Erzeugen eines Terminalworts ($\alpha_k \in \Sigma^*$)

Ausgabe durch TM.

$\Leftrightarrow$ Gegeben Aufzählung TM

Finde akzeptierende TM.

Arbeitsweise zu gegebenen Eingabe w

(14.0)

Lasse auf freiem Band laufen und bei Ausgabe eines Wortes wird Test auf Gleichheit mit w durchgeführt. In diesem Fall: Stop und Akzeptiere.

Entscheidungsprobleme

Typ	Automatenmodell	Wortproblem	Nichtleerheits-problem	Äquivalenz-problem
3	NEA/DEA	+	+	+
2	PDA	+	+	−
1	LBA	+	−	−
0	TM	$\underbrace{-}_{\mathrm{B}u\mathrm{K}}$	−	−

(Tabelle auch 14.1)

Satz

Das Nichtleerheitsproblem für Typ 1 - Sprachen ist unentscheidbar

Beweis benutzt die Charakterisierung der Typ1 -Sprachen durch LBA's

Reduktion MPCP $\leq$ Nichtleerheitsproblem(LBA)

Hierzu Transformation von MPCP- Instanz

$$X = (x_1, \ldots, x_k) \qquad Y = (y_1, \ldots, y_k)$$

in LBA $\mathcal{A}_{X,Y}$ mit (X, Y) hat Lösung $(1, i_2, \ldots, i_n)$

$$[x_1 x_{i_2} \ldots x_{i_n} = y_1 y_{i_2} \ldots y_{i_n}] \Leftrightarrow L(\mathcal{A}_{X,Y}) \neq \emptyset$$

Arbeitsweise von $\mathcal{A}_{X,Y}$ auf Eingabe w: $\mathcal{A}_{X,Y}$ prüft, ob w in Form $w = x_1 x_{i_2} \ldots x_{i_n}$ und $w = y_1 y_{i_2} \ldots y_{i_n}$ zerlegbar.

Idee zur Umsetzung:

- X = (abaaa,ab,a)
- Y = (ab,b,aa)
- Lösung: 1, 3,3 2

$$w = \overline{\underline{a}} b \underline{a} a a \overline{\underline{a}} a a \underline{b}]$$

(14.2)

Details Arbeitsalpabet: Σ, dazu $\overline{a}$, $\underline{a}$, $\overline{\underline{a}}$ für $a \in \Sigma$

1. $\mathcal{A}_{X,Y}$ platziert $\underline{\ }$ auf 1. Buchstaben und gehe in Zustand [1]. $[i]$ verlangt Abarbeitung der nächsten Buchstaben der Eingabe mit x_i, y_i
2. Im Zustand $[i]$ versuche Marke $\overline{\ }$ über das Eingabesehgment x_i ($\underline{\ }$ über y_i) zu verschieben.
   - Falls Verschiebung fehlschlägt, stoppen ohne zu akzeptieren
   - Falls Verschiebung in beiden Fällen bis zum Wortende reicht, stop mit Akz.
3. Nichtdet. Übergang in einen neues $[i]$ zurück zu 2.

Dann: $(X; Y)$ hat Lösung mit (Lösungswort w) $\Leftrightarrow \mathcal{A}_{X,Y} \neq \emptyset$ (nämlich $w \in L(\mathcal{A}_{X,Y})$

9.2 Ergänzende Resultat zur Chomsky Hierarchie

(14.3)

NEA	PDA	TM
TYP3	Typ2	Typ0

Satz (Büchi 1964)

Die Menge der Kellerinhalte, die ein PDA (über irgendwelche Inputwörter) errechen kann, ist r**regulär**.

Einfacher Fall: Keller expandiert nur (14.4)

NEA löscht den Keller in umgekehrter Reihenfolge der PDA- Schritte. (14.5)

Satz von Chomsky und Schützenberger

$\underbrace{\text{Doppelklammersprache}}$ über Alphabet DD_2 (siehe 14.5)

Dycksprache D_2

$DD_2 = \{),], (,]\}$

D_2 enthält z.B. []([]), ([[()]])

Grammatik: $S \to () \mid [] \mid (S) \mid [S] \mid SS$

Jede kontextfreie Sprache $L \supseteq \Sigma^* = \{a_1, \ldots, a_n\}$ ist darstellbar als

$$L = f(D_2 \cap R)$$

mit $D_2, R \subseteq DD_2$ und R regulär f: $[\underbrace{(\ldots(}_{i}[\to a_i/\varepsilon\ ]\underbrace{)\ldots)}_{i}] \to \varepsilon$

(14.6)

2-PDA's: PDA's mit zwei Keller

(14.7)

Transition $(p, a/\varepsilon, X, Y, \alpha, \beta, q)$

Eine Turingmachine kann man durch einen 2-PDA simulieren.

TM Konfguration (14.8)

Ein paralleles Modell Zellulare Automaten 1-dimensionaler Fall

(14.9)

$\mathcal{A} = (Q, q_0, \delta, q_+)$ $\delta : Q \times Q \times Q \to Q$ bisherige Zustände von linkem Nachbarn, selbst, rechten Nachbarn $\mapsto$ neuer eigener Zustand

Konfiguration:

ZZ-Folge von Zuständen $\ldots \overbrace{p_{-2}p_{-1}}\, p_0 p_1 p_2 \ldots$ übergang simultan in allen Komponenten. Standardfalle: $\ldots q_0 q_0 q_0 \quad q_0 q_0 q_0\ q_0$ Ruhezustand $\delta(q_0, q_0, q_0) = q_0$

Akzeptieren von $w = a_1 \ldots a_n$

Deklariere Buchstaben als besondere Zustände und beginne mit Konfiguration

$$\underbrace{\quad}_{q_0}[a_1 a_2 \ldots a_n]\underbrace{\quad}_{q_0}$$

w wird akzeptiert, wenn in Konfigurationsfolge irgendwann q_+ (an irgendeiner Stelle) auftaucht.

Satz

(a) Jede reguläre Sprache ist zellular akzeptierbar

(b) Die Sprache $\{a^n b^n c^n\}$ ist zellular akzeptierbar

Beweis

(a) Simuliere DEA durch Zell. Automaten

DEA auf $w = abb$ akzeptiert mit Lauf $p_0p_1p_2p_1 \in F$

ZA:

- $[a\ b\ b\ ]$
- $[p_1\ p_2\ b\ ]$
- $[p_1\ p_2\ p_1\ ]$
- $[p_1\ p_2\ q_+]$ (da geht einen pfeil von (1,1) -> (4,4) (a ->])

(b)

Test auf Eingabeform $\underbrace{a \ldots a}_{n}\underbrace{b}_{n}\underbrace{c}_{n}]$ Siehe(14.10)