Theoretische Informatik 1

Zusammenfassung zum Modul Formale Sprachen und Automatentheorie

Simon König

Inhaltsverzeichnis

I Formale Sprachen und Automatentheorie **4**

1 Formale Sprachen – Einstieg **5**

2 Alle Sprachen **6**

3 Rekursiv aufzählbare Sprachen $(\mathrm{r.e.})$: Typ-0 **7**

4 Entscheidbare Sprachen (REC): **8**

5 Kontextsensitive Sprachen (CSL): Typ-1 **9**
5.1 Algorithmus zur Entscheidbarkeit des Wortproblems: . . . 9
5.2 Automatenmodell LBA: . . . 9
5.3 Beispiele: . . . 9

6 Kontextfreie Sprachen (CFL): Typ-2 **10**
6.1 Automatenmodell PDA: . . . 10
6.2 Sätze zu den kontextfreien Sprachen: . . . 11
6.2.1 Pumping-Lemma für Typ-2: . . . 11
6.3 Chomsky-Normalform . . . 11
6.3.1 Umformungsalgorithmus: . . . 11
6.4 Greibach-Normalform . . . 12
6.4.1 Umformungsalgorithmus: . . . 12
6.5 Beispiele: . . . 13

7 Deterministisch Kontextfreie Sprachen (DCFL): **14**
7.1 Automatenmodell DPDA: . . . 14
7.2 Sätze zu den deterministisch kontextfreien Sprachen: . . . 14
7.3 Beispiele: . . . 15

8 Reguläre Sprachen (REG): Typ-3 **16**
8.1 Automatenmodell DEA: . . . 16
8.2 Automatenmodell NEA: . . . 16
8.3 Reguläre Ausdrücke . . . 17
8.4 Sätze zu den regulären Sprachen: . . . 17
8.4.1 Pumping-Lemma für Typ-3 . . . 17
8.4.2 Myhill-Nerode-Äquivalenz . . . 18
8.4.3 Erkennung durch Monoide – Syntaktisches Monoid . . . 18
8.5 Beispiele: . . . 19

9 Endliche Sprachen (FIN): **20**
9.1 Beispiele: . . . 20

10 Algorithmen **21**
10.1 Konstruktionsalgorithmus für Minimal-DEAs: 21
10.2 CYK-Algorithmus zur Lösung des Wortproblems für Typ-2 22

11 Anwedungen der Sätze **23**
11.1 Pumping-Lemma für Typ-3 23
11.2 Myhill-Nerode-Äquivalenz 23
11.3 Pumping-Lemma für Typ-2 23

1

Formale Sprachen und Automatentheorie

1: Formale Sprachen – Einstieg

Definition 1.1: Alphabet

Als Alphabet bezeichnen wir eine endliche, nichtleere Menge, deren Elemente Buchstaben genannt werden.
Dieses wird üblich mit Σ bezeichnet.

Definition 1.2: Freies Monoid über Σ

Ein Monoid ist eine Menge mit einer assoziativen Verknüpfung und einem neutralen Element. Die Menge aller endlichen Zeichenketten, die sich aus Elementen von Σ bilden lassen bilden mit der *Konkatenation* ein Monoid $\Sigma^\star$.
Das leere Wort (ϵ) bildet das neutrale Element.

Definition 1.3: Grammatik

Eine Grammatik ist ein Quadrupel:

$$G = (V, \Sigma, P, S)$$

V Eine Menge an Zeichen, den Variablen

Σ Das Alphabet, $V \cap \Sigma = \varnothing$

P Die Produktionsmenge, $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^\star$

S Das Startsymbol oder die Startvariable, $S \in V$

2: Alle Sprachen

3: Rekursiv aufzählbare Sprachen $(\mathrm{r.e.})$: Typ-0

Nicht entscheidbar in allen Kategorien, Abschluss unter Stern, Vereinigung, Schnitt und Konkatenation.

4: Entscheidbare Sprachen (REC):

(REC) $\subset$ (r.e.)

5: Kontextsensitive Sprachen (CSL): Typ-1

$$(\mathrm{CSL}) \subset (\mathrm{REC}) \subset (\mathrm{r.e.})$$

Wortproblem entscheidbar. Abschluss unter allen Operationen.

5.1 Algorithmus zur Entscheidbarkeit des Wortproblems:

Die Funktion $\mathrm{Abl}_n(X)$ wird iteriert angewendet, bis sich entweder X nicht mehr ändert ($w \notin L(G)$) oder das gesuchte Wort w in X enthalten ist ($w \in L(G)$).
Dabei ist n die Länge des gesuchten Worts w, also $|w|$.
Die Funktion $\mathrm{Abl}_n(X)$ ist für eine Grammatik G wie folgt definiert:

$$\mathrm{Abl}_n(X) := X \cup \{w \in (V \cup \Sigma)^\star \mid |w| \leq n \wedge \exists y \in X : y \Rightarrow_G w\}$$

5.2 Automatenmodell LBA:

Die Kontextsensitiven Sprachen werden von linear beschränkten Turingmaschinen akzeptiert.

5.3 Beispiele:

- $L_1 = \{a^n b^n c^n \mid n \geq 1\}$ (drei und mehr gleiche Exponenten)
-

6: Kontextfreie Sprachen (CFL): Typ-2

$$(\mathrm{CFL}) \subset (\mathrm{CSL}) \subset (\mathrm{REC}) \subset (\mathrm{r.e.})$$

Wort- und Leerheitsproblem entscheidbar.

6.1 Automatenmodell PDA:

Der Push Down Automata (Kellerautomat) ist ähnlich definiert wie ein nichtdeterministischer endlicher Automat (Siehe Abschnitt 8.2).
Der PDA jedoch hat einen entscheidenden Unterschied, er hat einen sog. *Kellerspeicher*, in dem Informationen zwischengespeichert werden können.

Ein Kellerautomat ist ein 6-Tupel

$$M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$$

Z endliche Zustandsmenge

Σ Eingabealphabet

Γ Kelleralphabet

δ Überführungsfunktion $\delta : Z \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathrm{Pot}_e(Z \times \Gamma^\star)$

z_0 Startzustand, $z_0 \in Z$

$\#$ Keller-Bottom-Symbol $\# \in \Gamma \setminus \{\Sigma\}$

Akzeptierte Sprache eines nichtdeterministischen PDA:

$$N(M) \coloneqq \{w \in \Sigma^\star \mid \exists z \in Z : (z_0, w, \#) \vdash^\star (z, \epsilon, \epsilon)\}$$

dies wird auch als *Akzeptieren durch leeren Keller* bezeichnet.
Insbesondere beim deterministischen Kellerautomaten gibt es eine zweite Definition der akzeptierten Sprache, beide Definitionen sind äquivalent (Siehe Abschnitt 7.1).

Konfiguration des PDA: Eine Konfiguration ist jedes Element k aus der Menge $Z \times \Sigma^\star \times \Gamma^\star$
Einen Konfigurationsübergang stellt man durch $\vdash$ dar.
Die aktuelle Konfiguration sei $(z, a_1 a_2 \dots a_n, A_1 \dots A_m)$, möglich sind jetzt die Übergänge aus $\delta(z, a_1, A_1)$ und $\delta(z, \epsilon, A_1)$.
Ein möglicher Übergang ist also für $(z', B_1 \dots B_k) \in \delta(z, a_1, A_1)$:

$$(z, a_1 a_2 \dots a_n, A_1 \dots A_m) \vdash (z', a_2 \dots a_n, B_1 \dots B_k A_2 \dots A_m)$$

6.2 Sätze zu den kontextfreien Sprachen:

- Alle Typ-2 Sprachen über einem einelementigen Alphabet sind bereits regulär.
- Die Klasse der kontextfreien Sprachen ist abgeschlossen unter Sternoperation, Vereinigung und Konkatenation.
  Zur Vereinigung: $G_1 = (V_1, \Sigma, P_1, S_1), G_2 = (V_2, \Sigma, P_2, S_2), V_1 \cap V_2 = \varnothing, S \notin V_1 \cup V_2$.
  $$G = (V_1 \cup V_2, \Sigma, P_1 \cup P_2 \cup \{(S, S_1), (S, S_2)\}, S) \rightsquigarrow L(G) = L(G_1) \cup L(G_2)$$
- Das Wortproblem ($\mathcal{O}(n^3)$) sowie das Leerheitsproblem sind entscheidbar (Siehe Abschnitt 10.2).
- Die Klasse der Typ-2 Sprachen ist identisch zu der durch EBNF beschreibbaren Sprachen.

6.2.1 Pumping-Lemma für Typ-2:

Sei $L \subseteq \Sigma^\star$ eine kontextfreie Sprache, dann gibt es eine Zahl n so, dass für alle $z \in L$ mit $|z| \geq n$ eine Zerlegung mit $z = uvwxy$ in $u, v, w, x, y \in \Sigma^\star$ exisitert für die die drei Bedingungen erfüllt sind:

- $|vx| \geq 1$
- $|vwx| \leq n$
- $\forall i \in N : uv^iwx^iy \in L$

6.3 Chomsky-Normalform

Eine Typ-2 Grammatik (V, Σ, P, S) ist in Chomsky-Normalform (CNF), wenn gilt:

$$\forall (u, v) \in P : v \in V^2 \cup \Sigma$$

Zu jeder Typ-2 Grammatik existiert eine Grammatik G' in CNF für die gilt $L(G) = L(G')$!
Für alle Ableitungen in CNF gilt: Die Ableitung eines Wortes der Länge n benötigt genau $2n - 1$ Schritte!

6.3.1 Umformungsalgorithmus:

1. Zunächst wollen wir erreichen, dass folgendes gilt: $(u, v) \in P \Rightarrow (|v| > 1 \vee v \in \Sigma)$
   (a) **Ringableitungen entfernen:**
   Eine Ringableitung liegt vor, wenn es Variablen $A_1, \ldots A_r$ gibt, die sich im Kreis in einander ableiten lassen, d.h. es gibt Regeln $A_i \to A_{i+1}$ und $A_r \to A_1$.
   Um dies loszuwerden, werden alle Variablen A_i durch eine neue Variable A ersetzt. Überflüssige Regeln wie $A \to A$ werden gelöscht.
   (b) **Variablen anordnen:**
   Man legt eine Ordnung der Variablen fest: $V = \{B_1, B_2, \ldots, B_n\}$, hierfür muss gelten:
   $$A_i \to A_j \in P \Leftrightarrow i < j$$
   Falls dies nicht gilt, müssen Abkürzungen verwendet werden, also alle Produktionen von A_j werden eingesetzt:
   $$P = (P \setminus \{A_i \to A_j\}) \cup \{(A_i, w) \mid (A_j, w) \in P\}$$

2. Jetzt gilt für jede Regel $(u, v) \in P$ entweder $v \in \Sigma$ oder $|v| \geq 2$.

   Für letztere Regeln werden nun **Pseudoterminale** eingeführt. Es werden neue Variablen und Produktionen für jedes Terminalsymbol hinzugefügt, z.B. $V_a \to a$.

3. **Letzter Schritt:** Alle rechten Seiten mit $|v| > 2$ müssen nun noch auf Länge 2 gekürzt werden. Hierfür werden wiederum neue Variablen eingefügt:

   $$A \to C_1C_2C_3$$

   Wird gekürzt zu

   $$A \to C_1D_1$$
   $$D_1 \to C_2C_3$$

6.4 Greibach-Normalform

Eine Typ-2 Grammatik (V, Σ, P, S) ist in Greibach-Normalform (GNF), wenn gilt:

$$\forall (u, v) \in P : v \in \Sigma V^\star$$

Zu jeder Typ-2 Grammatik existiert eine Grammatik G' in GNF für die gilt $L(G) = L(G')$!

6.4.1 Umformungsalgorithmus:

1. **Der erste Algorithmus:**

   Der erste Algorithmus hat zum Ziel, dass alle Produktionen einer bestimmten Ordnung unterliegen. Hierfür werden zunächst die Variablen angeordnet:

   $$V = \{A_1, A_2, \ldots, A_m\}$$

   Nun sollen nur Regeln $A_i \to A_j\alpha$ in P sein, wenn $i < j$ gilt.

   Um dies zu erreichen wird solange Eingesetzt bis keine Regeln dieser Form vorliegen.

   Für alle $A_i \to A_j\alpha \in P$ mit $i > j$ werden alle Produktionsregeln

   $$A_j \to \beta_1 | \ldots | \beta_r$$

   Eingesetzt zu

   $$A_i \to \beta_1\alpha | \ldots | \beta_r\alpha$$

   Und schließlich die Regel $A_i \to A_j\alpha$ aus P gestrichen.

2. **Beseitigung von Linksrekursion:**

   Alle Produktionsregeln sind von der Form:

   $$A \to A\alpha_1 | \ldots | A\alpha_k | \beta_1 | \ldots | \beta_l$$

   Diese können durch diese $2k + 2l$ Regeln ersetzt werden:

   $$A \to \beta_1 | \ldots | \beta_l$$
   $$A \to \beta_1 B | \ldots | \beta_l B$$
   $$B \to \alpha_1 | \ldots | \alpha_k$$
   $$B \to \alpha_1 B | \ldots | \alpha_k B$$

   Nun sind keinerlei Linksrekursionen mehr vorhanden!

3. **Der zweite Algorithmus:**

   Der zweite Algorithmus hat zum Ziel, dass alle rechten Seiten mit einem Terminalsymbol beginnen.

   Da die Regeln mit A_m auf der linken Seite nur in ein Terminal übergehen können, da sie am Ende der Variablenanordnung stehen müssen wir nur alle A_m-Produktionen bei den A_{m-1}-Regeln einsetzen und so weiter.

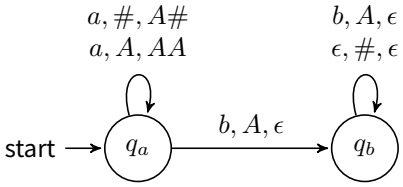
4. **Der letzte Schritt:**

   Als letztes müssen wir überprüfen ob die B-Regeln aus der Beseitigung der Linksrekursionen die gewünschte Form haben. Da aber alle B-Produktionen entweder mit einem A_i oder einem Terminal beginnen, muss wieder nur eingesetzt werden.

   Schließlich müssen die Terminalsymbole, die in allen Produktionen weiter hinten auftreten durch Pseudoterminale ersetzt werden.

6.5 Beispiele:

- $L_1 = \left\{ ww^R \,\middle|\, w \in \Sigma^\star \right\}$ (unmarkierte Palindrome)
- Korrekt geklammerte arithmetische Ausdrücke (Dyck-Wörter)
- Kellerautomat, der die Sprache $L_2 = \{a^n b^n \mid n \geq 1\}$ akzeptiert:



Konfigurationsübergänge bei Eingabewort $aaabb$:

$$\begin{aligned}(z_a, aaabb, \#) &\vdash (z_a, aabb, A\#)\\ &\vdash (z_a, abb, A\#)\\ &\vdash (z_a, bb, AAA\#)\\ &\vdash (z_a, b, AA\#)\\ &\vdash (z_a, \epsilon, A\#)\\ &\rightsquigarrow aaabb \notin N(M) = L_2\end{aligned}$$

7: Deterministisch Kontextfreie Sprachen (DCFL):

$$(\mathrm{DCFL}) \subset (\mathrm{CFL}) \subset (\mathrm{CSL}) \subset (\mathrm{REC}) \subset (\mathrm{r.e.})$$

Wort-, Leerheits- und Äquivalenzproblem entscheidbar. Abschluss nur unter Komplement.
Beschreib- und erkennbar durch einen endlichen deterministischen Kellerautomaten (DPDA).

7.1 Automatenmodell DPDA:

Der deterministische Kellerautomat ist ähnlich definiert wie ein der nichtdeterministische (Siehe Abschnitt 6.1).
Der Unterschied zum PDA liegt dabei, dass beim DPDA in jeder Situation nur ein Übergang möglich sein darf,

$$\forall z \in Z, a \in \Sigma, A \in \Gamma : |\delta(z, a, A)| + |\delta(z, \epsilon, A)| \leq 1$$

und der DPDA akzeptiert nicht durch leeren Keller sondern durch Endzustände.

Ein deterministischer Kellerautomat ist ein 7-Tupel

$$M = (Z, \Sigma, \Gamma, \delta, z_0, \#, E)$$

Z endliche Zustandsmenge

Σ Eingabealphabet

Γ Kelleralphabet

δ Überführungsfunktion $\delta : Z \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to Z \times \Gamma^\star$

z_0 Startzustand, $z_0 \in Z$

$\#$ Keller-Bottom-Symbol $\# \in \Gamma \setminus \{\Sigma\}$

E Endzustandsmenge $E \subseteq Z$

Akzeptierte Sprache eines deterministischen PDA:

$$N(M) \coloneqq \{w \in \Sigma^\star \mid \exists e \in E, V \in \Gamma^\star : (z_0, w, \#) \vdash^\star (e, \epsilon, V)\}$$

dies wird auch als *Akzeptieren durch Endzustand* bezeichnet.
Beide Akzeptanzarten, (durch Endzustand und leeren Keller, siehe Abschnitt 6.1) sind äquivalent.

7.2 Sätze zu den deterministisch kontextfreien Sprachen:

- Eine deterministisch kontextfreie Sprache geschnitten mit einer regulären Sprache ist wieder eine deterministisch kontextfreie Sprache.

  $$L_1 \in (\mathrm{DCFL}), L_2 \in (\mathrm{REG}) \Rightarrow L_1 \cap L_2 \in (\mathrm{DCFL})$$

7.3 Beispiele:

- $L_1 = \left\{ w\$w^R \,\middle|\, w \in \Sigma^\star \right\}$ (markierte Palindrome)

8: Reguläre Sprachen (REG): Typ-3

$$(\mathrm{REG}) \subset (\mathrm{DCFL}) \subset (\mathrm{CFL}) \subset (\mathrm{CSL}) \subset (\mathrm{REC}) \subset (\mathrm{r.e.})$$

8.1 Automatenmodell DEA:

Ein deterministischer endlicher Automat ist ein 5-Tupel

$$M = (Z, \Sigma, \delta, z_0, E)$$

Z endliche Zustandsmenge

Σ Eingabealphabet

δ Überführungsfunktion $\delta : Z \times \Sigma \to Z$

z_0 Startzustand, $z_0 \in Z$

E Endzustandsmenge, $E \subseteq Z$

Es lässt sich außerdem eine erweiterte Funktion $\hat{\delta}$ definieren:

$$\hat{\delta} : Z \times \Sigma^\star \to Z$$

Mit den folgenden Eigenschaften:

$$\hat{\delta}(z, \epsilon) = z$$
$$\hat{\delta}(z, ax) = \hat{\delta}(\delta(z, a), x)$$

Die von einem deterministischen Automaten M akzeptierte Sprache ist

$$T(M) = \left\{ w \in \Sigma^\star \,\middle|\, \hat{\delta}(z_0, w) \in E \right\}$$

8.2 Automatenmodell NEA:

Ein nichtdeterministischer endlicher Automat ist ein 5-Tupel

$$M = (Z, \Sigma, \delta, S, E)$$

Z endliche Zustandsmenge

Σ Eingabealphabet

δ Überführungsfunktion $\delta : Z \times \Sigma \to \mathrm{Pot}(Z)$

S Startzustandsmenge, $S \subseteq Z$

E Endzustandsmenge, $E \subseteq Z$

Der NEA ist formal stärker als der DEA, sie akzeptieren jedoch beide die selbe Sprachklasse.
Die akzeptierte Sprache eines nichtdeterministischen endlichen Automaten ist:

$$T(M) = \left\{ w \in \Sigma^\star \;\middle|\; \hat{\delta}(S, w) \cap E \neq \varnothing \right\}$$

8.3 Reguläre Ausdrücke

Die regulären Sprachen lassen sich zusätzlich zu den zwei Automatenmodellen auch durch sog. reguläre Ausdrücke beschreiben. Eine Definition für die Syntax der regulären Ausdrücke ist:

- $\varnothing$ und ϵ sind reguläre Ausdrücke.
- a ist ein regulärer Ausdruck für alle $a \in \Sigma$
- Wenn α und β eguläre Ausdrücke sind, dann sind auch $\alpha\beta$, $(\alpha|\beta)$ und $(\alpha)^\star$ reguläre Ausdrücke.

Die Semantik der regulären Ausdrücke ist ebenso induktiv bestimmt:

- $L(\varnothing) = \varnothing$ und $L(\epsilon) = \{\epsilon\}$
- $L(a) = \{a\}$ für jedes $a \in \Sigma$
- $L(\alpha\beta) = L(\alpha)L(\beta)$, $L(\alpha|\beta) = L(\alpha) \cup L(\beta)$, $L((\alpha)^\star) = L(\alpha)^\star$

8.4 Sätze zu den regulären Sprachen:

- Typ-3 Sprachen können *nicht* inhärent mehrdeutig sein, da sich zu jeder Sprache ein Minimalautomat bilden lässt.
- Die Klasse der Typ-3 Sprachen ist unter allen boole'schen Operationen, Sternoperation und der Konkatenation abgeschlossen.
- Für reguläre Sprachen ist das Wortproblem (in Linearzeit), das Leerheitsproblem, das Äquivalenzproblem sowie das Schnittproblem entscheidbar.
- Alle Typ-2 Sprachen über einem einelementigen Alphabet sind bereits regulär.

8.4.1 Pumping-Lemma für Typ-3

Für jede reguläre Sprache L gibt es ein $n \in \mathbb{N}$, so dass für jedes $x \in L$ mit $|x| \geq n$ eine Zerlegung in drei Teile exisitert: $x = uvw$, so dass die drei Bedingungen erfüllt sind:

- $|v| \geq 1$
- $|uv| \leq n$
- $\forall i \in \mathbb{N} : uv^i w \in L$ („Pump-Bedingung“)

Gilt die Negation dieser Aussage, also

$$\forall n \in \mathbb{N} : \exists x \in L, |x| \geq n : \forall u, v, w \in \Sigma^\star, x = uvw, |v| \geq 1, |uv| \leq n : \exists i \in \mathbb{N} : uv^i w \notin L$$

so ist L nicht regulär!

ABER: Das Pumping-Lemma gibt keine Charakterisierung der Typ-3 Sprachen an! Es gibt auch Sprachen, die nicht Typ-3 sind, die Aussage des Lemmas aber trotzdem erfüllen!
Das Pumping-Lemma gibt also nur eine Möglichkeit, zu Beweisen, dass eine Sprache *nicht* regulär ist! (Siehe Abschnitt 11.1)

8.4.2 Myhill-Nerode-Äquivalenz

Mit der Myhill-Nerode-Äquivalenz ist es möglich nachzuweisen, ob eine Sprache regulär ist.

$$x \mathrm{R}_L y \iff [\forall w \in \Sigma^\star : xw \in L \Leftrightarrow yw \in L]$$

Bzw. anhand eines DEA (dies führt zu einer Verfeinerung von R_L)

$$x \mathrm{R}_M y \iff [\delta(z_0, x) = \delta(z_0, y)]$$

Es gilt:

$$x \mathrm{R}_M y \Rightarrow \forall w \in \Sigma^\star : \delta(z_0, xw) = \delta(z_0, yw) \Rightarrow x \mathrm{R}_L y$$

Die Sprache $L \subseteq \Sigma^\star$ ist genau dann regulär, wenn der Index der Myhill-Nerode-Äquivalenz R_L endlich ist.

8.4.3 Erkennung durch Monoide – Syntaktisches Monoid

Sei $L \subseteq \Sigma^\star$ eine formale Sprache und M ein Monoid.
M erkennt L, wenn eine Teilmenge $A \subseteq M$ und ein Homomorphismus $\varphi : \Sigma^\star \to M$ existiert, so dass gilt:

$L = \varphi^{-1}(A)$	d.h. $w \in L \Leftrightarrow \varphi(w) \in A$
$L = \varphi^{-1}(\varphi(L))$	d.h. $w \in L \Leftrightarrow \varphi(w) \in \varphi(L)$

Weiter kann man für eine konkrete Sprache L die *syntaktische Kongruenz* definieren:

$$w_1 \equiv_L w_2 \iff [\forall x, y \in \Sigma^\star : xw_1y \in L \Leftrightarrow xw_2y \in L]$$

Basierend auf dieser Kongruenz definieren wir das Quotientenmonoid der Kongruenz, dessen Elemente die Äquivalenzklassen sind.
Das Quotientenmonoid oder auch syntaktisches Monoid bezüglich der syntaktischen Kongruenz wird mit

$$\mathrm{Synt}(L) \coloneqq (\Sigma^\star / \equiv_L)$$

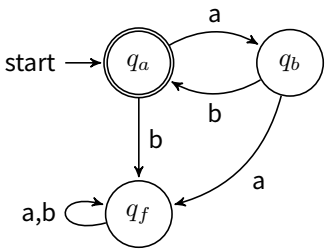
bezeichnet.
Für jede Sprache L gibt es ein syntaktisches Monoid das L mit dem Homomorphismus

$$\varphi : L \to \mathrm{Synt}(L), w \mapsto [w]$$

erkennt. Ist $|\mathrm{Synt}(L)|$ endlich, so ist L regulär bzw. erkennbar.

8.5 Beispiele:

- $L_1 = \Sigma^\star$
- $L_2 = L(a(a|b)^\star bb(ab)^\star)$
- Automat M mit $L_3 = T(M) = \{(ab)^n \mid n \in \mathbb{N}_0\}$:



9: Endliche Sprachen (FIN):

$$(\mathrm{FIN}) \subset (\mathrm{REG}) \subset (\mathrm{DCFL}) \subset (\mathrm{CFL}) \subset (\mathrm{CSL}) \subset (\mathrm{REC}) \subset (\mathrm{r.e.})$$

Alle endlichen Sprachen sind regulär.

9.1 Beispiele:

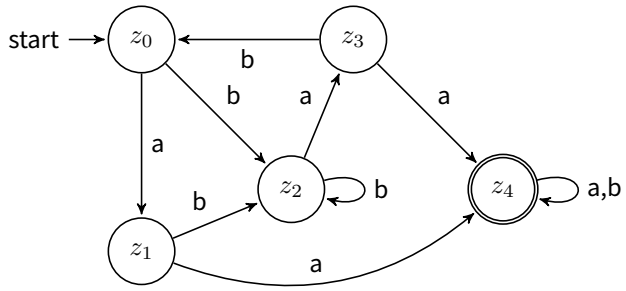
- $L_1 = \varnothing$ (leere Sprache ist endlich)
- $L_2 = \Sigma$ (nur die Buchstaben)
- $L_3 = \{aaa, baba\}$ (z.B. nur zwei Wörter)

10: Algorithmen

10.1 Konstruktionsalgorithmus für Minimal-DEAs:

Mit dem Beweis zur Myhill-Nerode-Äquivalenz wird ein Automat definiert, dieser ist isomorph zum Minimalautomaten. Der Index der Myhill-Nerode-Äquivalenz ist genau die Anzahl der Zustände des Minimalautomaten.
Man kann mit einem einfachen Algorithmus aus einem beliebigen DEA den Minimalautomaten erzeugen:
Wir ermitteln algorithmisch, welche Zustände nicht äquivalent sind und verschmelzen die übrig bleibenden. Nicht äquivalent sind Zustände, bei denen bei Eingabe eines Worts vom einen aus ein Endzustand erreicht wird, vom anderen jedoch nicht.
So sind im ersten Schritt Zustandspaare aus Endzustand und Nichtendzustand nicht äquivalent und werden markiert.

Am Beispiel:



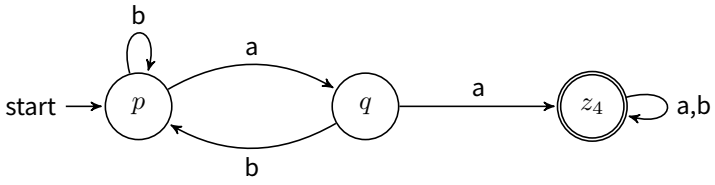
Die Paare $\{z_i, z_4\}$ mit $i = 0, 1, 2, 3$ werden markiert:

z_1				
z_2				
z_3				
z_4	ϵ	ϵ	ϵ	ϵ
	z_0	z_1	z_2	z_3

Durch Testen der Zustandspaare erhält man die untenstehende Tabelle. Hierbei wurden Zeugen für die Antivalenz eingetragen.

z_1	a			
z_2		a		
z_3	a		a	
z_4	ϵ	ϵ	ϵ	ϵ
	z_0	z_1	z_2	z_3

Damit lassen sich Zustände zusammenfassen: $p = \{z_0, z_2\}, q = \{z_1, z_3\}$. Der Minimalautomat ist also:



$$T(M) = \{waaw' \mid w, w' \in \{a, b\}^\star\}$$

10.2 CYK-Algorithmus zur Lösung des Wortproblems für Typ-2

Mit dem CYK-Algorithmus ist das Wortproblem für Typ-2 in $\mathcal{O}(n^3)$ entscheidbar.
Hierfür werden alle Ableitungsmöglichkeiten in einer Tabelle geordnet dargestellt. Ist am Ende die Startvariable als Startknoten für die Ableitung möglich, so ist das Wort in L.

Am Beispiel:

Produktionsregeln der Grammatik (CNF):

$$S \to AX|YB, A \to XA|AB|a, B \to XY|BB, X \to YA|a, Y \to XX|b$$

Eingabewort: $abbaab$

Länge	a	b	b	a	a	b
1	$\{A, X\}$	$\{Y\}$	$\{Y\}$	$\{A, X\}$	$\{A, X\}$	$\{Y\}$
2	$\{B\}$	$\varnothing$	$\{X\}$	$\{A, S, Y\}$	$\{B\}$	
3	$\varnothing$	$\varnothing$	$\{A, Y, X\}$	$\{A\}$		
4	$\varnothing$	$\{X\}$	$\{B, X\}$			
5	$\{S, Y\}$	$\{B, S\}$				
6	$\{A, B\}$					

Da die unterste Zelle nun $\{A, B\}$ enthält, ist $w = abbaab \notin L$

11: Anwedungen der Sätze

11.1 Pumping-Lemma für Typ-3

Für die Sprache $L = \{a^n b^n \mid n \geq 1\}$:

Zunächst wählt man ein Wort x, mit $|x| \geq n$:

$$x = a^n b^n \in L, \quad |a^n b^n| = 2n > n$$

Nun zu einer beliebigen Zerlegung $x = uvw$, für die die Bedingungen gelten:

$$x = uvw = a^n b^n$$
$$u = a^{n-l-k} \quad v = a^l \quad w = a^k b^n \text{ mit } l \geq 1$$
$$x = (a^{n-l-k})(a^l)(a^k b^n)$$

Pumpt man nun v mit $v = 0$:

$$x = (a^{n-l-k})(a^l)^i(a^k b^n) = (a^{n-l-k})(a^l)^0(a^k b^n)$$
$$x = (a^{n-l-k})(a^k b^n) = a^{n-l} b^n \notin L, \text{ da } l \geq 1$$

11.2 Myhill-Nerode-Äquivalenz

Für die Sprache $L = \{w \in \{a,b\}^\star \mid w \text{ enthält das Teilwort } abb\}$:

Finden der Äquivalenzklassen:

$$[\epsilon] = \{\epsilon, b, bb, \ldots\} = \{b^n \mid n \in N_0\}$$
$$[a] = \{b^n a^m \mid n \in \mathbb{N}_0, m \in \mathbb{N}^+\}$$
$$[ab] = \{w \in \{a,b\}^\star \mid w \text{ enthält das Teilwort } ab \text{ aber nicht das Teilwort } abb\}$$
$$[abb] = \{wabbw' \mid w, w' \in \{a,b\}^\star\}$$

11.3 Pumping-Lemma für Typ-2

- Für die Sprache $L = \{a^n b^n c^n \mid n \geq 1\}$:
  Zunächst wählt man ein Wort x, mit $|x| \geq n$:

  $$x = a^n b^n c^n \in L, \quad |a^n b^n c^n| = 3n > n$$

Nun zu einer beliebigen Zerlegung $x = uvwxy$, für die die Bedingungen gelten. Da $|vwx| \leq n$, können v und x nur maximal zwei unterschiedliche Buchstaben beinhalten, niemals jedoch a, b und c.

Damit kann uv^iwx^iy nicht in L sein für ein $i \neq 1$. □

- Für die Sprache $L = \{a^n \mid n \text{ ist eine Quadratzahl}\}$:

  Zunächst wählt man ein Wort x, mit $|x| \geq n$:

  $$x = a^{n^2} \in L$$

  Bei jeder Zerlegung $x = uvwxy$ sind nur as in den Teilwörtern v und x, die gepumpt werden.

  Betrachtet man die Länge $|vx| = r$, so gilt:

  $$|uv^iwx^iy| = n^2 + r(i-1)$$

  Insbesondere gilt für das Wort

  $$x' = uv^2wx^2y = a^s$$
  $$|x'| = n^2 + r = s$$

  Das hieße, $n^2 + r$ müsste eine Quadratzahl sein damit x' wiederum in L läge. Für r gilt aber die Ausgangsbedinung des Pumping-Lemmas!

  $$|vwx| = r + |w| \leq n$$

  s kann damit unmöglich eine Quadratzahl sein. □