Automaten und Formale Sprachen

SoSe 2007 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

$\boxed{\textbf{Automaten und Formale Sprachen}}$ Gesamtübersicht

- Organisatorisches
- Einführung
- **Endliche Automaten und reguläre Sprachen**
- Kontextfreie Grammatiken und kontrextfreie Sprachen
- Chomsky-Hierarchie

Endliche Automaten und reguläre Sprachen

1. Deterministische endliche Automaten

2. Nichtdeterministische endliche Automaten

3. Reguläre Ausdrücke

4. Nichtreguläre Sprachen

5. **Algorithmen mit / für endliche Automaten**

Wann ist nun ein DEA $\mathcal{A}$ nicht minimal ?

- Wenn es nicht-erreichbare Zustände gibt, d.h. es gibt q mit $(q_0, y) \vdash_{\mathcal{A}}^{*} (q, \lambda)$ für kein Wort $y \in \Sigma^*$.
  Im Folgenden: $\mathcal{A}$ hat nur erreichbare Zustände! (s.u.)

- Wenn es Zustände $q \neq q'$ gibt mit

  $$\forall w \exists p, p' \in Q : |\{p, p'\} \cap F| \neq 1 \Longrightarrow ((q, w) \vdash_{\mathcal{A}}^{*} (p, \lambda) \Longleftrightarrow (q', w) \vdash_{\mathcal{A}}^{*} (p', \lambda))$$

  d.h. q und q' sind nicht *trennbar*, sondern *äquivalent*.

Es bezeichne $[q]$ die Menge aller Zustände, die zu q äquivalent sind.

Eigenschaften äquivalenter Zustände

1. Sind q und q' äquivalent, dann auch $\delta(q, a)$ und $\delta(q', a)$, denn $(\delta(q, a), w) = (q, aw)$ und $(q', aw) = (\delta(q', a), w)$.

2. Sind q und q' äquivalent, dann gilt $q \in F \iff q' \in F$.

Zur Konstruktion des Minimalautomaten I

Definiere zu $A = (Q, \Sigma, \delta, q_0, F)$ neuen Automaten $A_{\equiv} = (Q_{\equiv}, \Sigma, \delta_{\equiv}, [q_0], F_{\equiv})$ mit

- Anfangszustand $[q_0]$
- Endzuständen $F_{\equiv} := \{[q] \mid q \in F\}$
- Übergangsfunktion $\delta_{\equiv}([q], a) := [\delta(q, a)]$

Mit A hat auch $A_{\equiv}$ hat keine nicht-erreichbaren Zustände.
Betrachte $f : Q \to Q_{\equiv}$ mit $f(q) := [q]$. Aus den aufgeführten Eigenschaften folgt:

Satz: f ist Automatenmorphismus; und damit gilt $L(A) = L(A_{\equiv})$.

Zur Konstruktion des Minimalautomaten II

Satz: $A_{\Box}$ isomorph zum Minimalautomaten.

Beweis: Vergleiche $\equiv_{A_{\Box}}$ und $x \equiv_L y$ für $L := L(A)$:

- $\equiv_{A_{\Box}}$ ist Verfeinerung von $\equiv_L$, da $L = L(A_{\Box})$.
- Sei $x \equiv_L y$. Da $L = L(A)$, gilt für q_x mit $(q_0, x) \vdash_A^* (q_x, \lambda)$ und für q_y mit $(q_0, y) \vdash_A^* (q_y, \lambda)$: $[q_x] = [q_y]$. Daher gilt:

$$([q_0], x) \vdash_{A_{\Box}}^* ([q_x], \lambda) \quad \wedge \quad ([q_0], y) \vdash_{A_{\Box}}^* ([q_x], \lambda).$$

  $\rightsquigarrow x \equiv_{A_{\Box}} y$.

Konstruktion des Minimalautomaten III

Gegeben sei DEA $A = (Q, \Sigma, \delta, q_0, F)$.

Schritt (a): Bestimme die Menge der von q_0 erreichbaren Zustände E!
Bezeichne E_i die Menge der in $\leq i$ Schritten erreichbaren Zustände.

- Setze $E_0 = \{q_0\}$ (und $E_{-1} := \emptyset$)
- Wiederhole
$$E_{i+1} = E_i \cup \{\delta(q, a) \mid q \in E_i \setminus E_{i-1}, a \in E\}$$
  bis erstmals $E_i = E_{i+1}$ gilt.
- Dann ist $E = E_i$.
- Entferne die Zustände $Q \setminus E$ aus dem Automaten.

Alternative Darstellung

Hinweis: reflexive transitive Hülle der 1-Einschritt-Erreichbarkeitsrelation

Genauer: Definiere zu DEA $A = (Q, \Sigma, \delta, q_0, F)$ die *1-Schritt-Zustandserreichbarkeitsrelation* $R = \{(p, q) \in Q \times Q \mid \exists a \in \Sigma : \delta(p, a) = q\}$.
Ist R^* die reflexive transitive Hülle von R, so ist

$$\{q \in Q \mid (q_0, q) \in R^*\}$$

die Menge der von q_0 erreichbaren Zustände.

Frage: Welches Verfahren ist besser ?!

Konstruktion des Minimalautomaten IV

Schritt (b): Bestimme die Äquivalenzrelation $\equiv_{\mathcal{A}}$ im nach (a) verkleinerten Automaten wie folgt mit folgenden *Markierungsalgorithmus*:

- Verwende eine Tabelle aller ungeordneten Zustandspaare $\{q, q'\}$ mit $q \neq q'$.
- Markiere alle Paare $\{q, q'\}$ als nicht-äquivalent, bei denen $|\{q, q'\} \cap F| = 1$.
- Wiederhole, solange noch Änderungen in der Tabelle entstehen:

  Für jedes nicht-markierte Paar $\{q, q'\}$ und jedes $a \in \Sigma$
  Teste, ob $(\ \delta(q, a), \delta(q', a)\ )$ bereits markiert ist.
  Wenn ja $\rightsquigarrow$ markiere $\{q, q'\}$.

- Alle am Ende nicht-markierten Paare sind äquivalent!

Gesamtaufwand (mit geeigneten Datenstrukturen und $k = |E|$ und $n = |Q|$, ohne Beweis):

$$O(k \cdot n^2)$$

Ein Beispiel:

In VL 3 haben wir zu $L = \{a, aa, ab, abb\}$ den *Präfixbaumakzeptor* konstruiert:

δ	a	b
$\to Q_0$	Q_1	$\emptyset$
$Q_1 \to$	Q_2	Q_3
$Q_2 \to$	$\emptyset$	$\emptyset$
$Q_3 \to$	$\emptyset$	Q_4
$Q_4 \to$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

Runde	neue markierte Paare
0	$M_0 = \{\{Q_i, \emptyset\}, \{Q_i, Q_0\} \mid i = 1, 2, 3, 4\}$
1	$\{\{Q_1, Q_2\}\}$ denn $\{\delta(Q_1, a), \delta(Q_2, a)\} \in M_0$
1	$\{\{Q_1, Q_3\}, \{Q_1, Q_4\}, \{Q_2, Q_3\}, \{Q_3, Q_4\}\}$
2	$\emptyset$
	übriggebliebene unmarkierte Paare
	$\{\{Q_2, Q_4\}\}$

Der Minimalautomat für $L = \{a, aa, ab, abb\}$ ist daher:

δ	a	b
$\rightarrow Q_0$	Q_1	$\emptyset$
$Q_1 \rightarrow$	Q_2	Q_3
$Q_2 \rightarrow$	$\emptyset$	$\emptyset$
$Q_3 \rightarrow$	$\emptyset$	Q_2
$\emptyset$	$\emptyset$	$\emptyset$

Andere Sprechweise: *Verschmelzung* der Zustände Q_2 und Q_4.

Weitere Fragen an vorgegebenen DEA $\mathcal{A}$: (evtl. zweiter DEA $\mathcal{A}'$)

- Ist $\mathrm{L}(\mathcal{A}) = \emptyset$? *Leerheitsproblem*
- Ist $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}')$? *Äquivalenzproblem*
- Ist $\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{A}')$? *Teilmengenproblem*
- Ist $\mathrm{L}(\mathcal{A})$ endlich ? *Endlichkeitsproblem*

Leerheitsproblem

Wir haben schon zwei Methoden kennen gelernt, die Menge E der erreichbaren Zustände zu berechnen. Die vom Automaten beschriebene Sprache ist leer gdw. E keine Endzustände enthält.

Betrachte zu DEA $A = (Q, \Sigma, \delta, q_0, F)$ die
erweiterte 1-Schritt-Zustandserreichbarkeitsrelation

$$R = \{(p, q) \in Q \times Q \mid \exists a \in \Sigma : \delta(p, a) = q\} \cup F \times \{q_f\},$$

wobei $q_f \notin Q$ und mit $Q' = Q \cup \{q_f\}$ gilt $R \subset Q' \times Q'$.
$L(A) = \emptyset$ gdw. $(q_0, q_f) \notin R^*$.
Die Existenz einer Punkt-zu-Punkt-Verbindung kann sogar in Linearzeit $O(|Q|)$ berechnet werden. (z.B.: Dijkstras Algorithmus)

Teilmengen- und Äquivalenzproblem

Beobachte: $L(A) \subseteq L(A')$ gdw. $L(A) \setminus L(A') = \emptyset$.

Daher:

1. Aus gegebenen DEAs A und A' berechne DEA A'' mit $L(A'') = L(A) \setminus L(A')$.
Dies geht direkt mit *Produktautomatenkonstruktion* wie auf Monoidebene erläutert.

2. Entscheide ob $L(A'') = \emptyset$ mit vorher skizziertem Verfahren.

Wegen $L(A) = L(A')$ gdw. $L(A) \subseteq L(A')$ und $L(A') \subseteq L(A)$ folgt damit die Entscheidbarkeit des Äquivalenzproblems.

Endlichkeitsproblem zu DEA $A = (Q, \Sigma, \delta, q_0, F)$

Wie im Beweis zum Pumping-Lemma sieht man:
Ist $L(A)$ unendlich, so gibt es einen Zustand q, einen (evtl. leeren) Weg vom Anfangszustand q_0 nach q, einen nicht-leeren Weg von q nach q und einen (evtl. leeren) Weg von q zu einem Endzustand.
Die Umkehrung gilt sogar trivialer Weise!
Bezeichnet R die 1-Schritt-Zustandserreichbarkeitsrelation, so berechne
E': die Menge der Zustände, die sowohl erreichbar als auch *co-erreichbar* sind (d.h., für alle $q \in E'$ gilt: $(q_0, q) \in R^*$ und $\exists q_f \in F : (q, q_f) \in R^*$).
Dann gilt: $L(A)$ ist unendlich gdw. $\exists q \in E' : (q, q) \in R^+$.

EA zur Mustersuche (Pattern Matching)

Beispiel: Finde Vorkommen des Musters (Pattern)

$$p = ababac$$

in einem Text $t \in \{a, b, c\}^*$.

Wir haben schon früher gesehen:
NEAs sind nützlich für diese Aufgabe.

In RA-artiger Notation beobachten wir:
Lemma: $t \in \Sigma^*$ enthält das Muster p gdw. $t \in \Sigma^*\{p\}\Sigma^*$.

Klar: Die Bedingung lässt sich sofort in NEA umsetzen.
Frage: Wie lassen sich hierzu DEAs nutzen ?

DEA zur Mustersuche

Vorteil wäre: Linearzeitalgorithmus zur Mustersuche.
Dagegen naiv: quadratischer Algorithmus zur Mustersuche; nämlich
Problem: Zurücksetzen bei “falschem Alarm”.

Ziel: Vermeide Potenzautomatenkonstruktion.

Wie geht das ?

Einige Hilfsbegriffe

u heißt *Teilwort* von $x \in \Sigma^*$ gdw. $x \in \Sigma^*\{u\}\Sigma^*$.

Mustersuche ist also die Suche nach Teilwörtern.

u heißt *Präfix* oder *Anfangswort* von $x \in \Sigma^*$ gdw. $x \in \{u\}\Sigma^*$.

u heißt *Suffix* oder *Endwort* von $x \in \Sigma^*$ gdw. $x \in \Sigma^*\{u\}$.

Ein Teilwort / Präfix / Suffix u von x heißt *echt* gdw. $\ell(u) < \ell(x)$.

Ein echtes Teilwort u von x, das sowohl Präfix als auch Suffix von x ist, heißt *Rand (der Breite $\ell(u)$)* von x.

Beispiel: Sei $x = abacab$.

Die echten Präfixe von x sind $\lambda, a, ab, aba, abac, abaca$;

die echten Suffixe von x sind $\lambda, b, ab, cab, acab, bacab$.

Ränder von x sind λ, ab; der Rand ab hat die Breite 2.

DEA-Konstruktionsidee für Mustersuche nach Knuth/Morris/Pratt (Matjasewitsch)

Frage: Wo darf DEA nach einem Mismatch wieder “einsetzen” ?

Idee: Verwende die im bisher gelesenen Präfix des Musters steckende Info !

Betrachte

Pos.	0	1	2	3	4	5	6	7	8
t	a	b	c	a	b	c	a	b	d
p	a	b	c	a	b	d			
Forts.				a	b	c	a	b	d

Die Symbole an den Positionen 0, . . . , 4 haben übereingestimmt. Der Vergleich $c - d$ an Position 5 ergibt einen Mismatch. Das Muster kann bis Position 3 weitergeschoben werden, und der Vergleich wird ab Position 5 des Textes fortgesetzt.

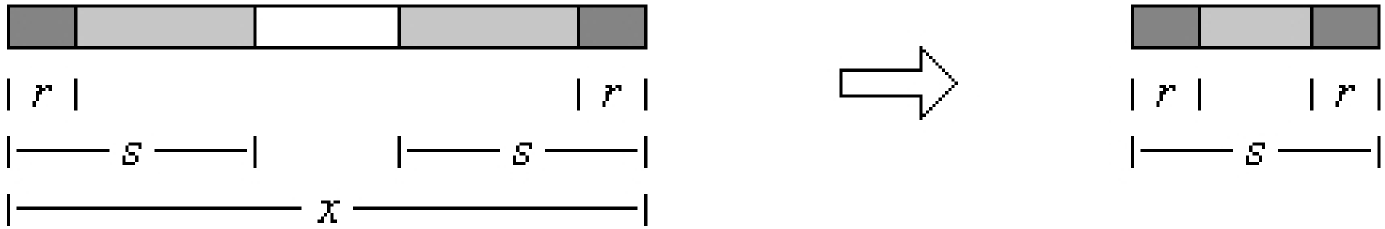
Wie weit dürfen wir schieben ?

Die *Schiebedistanz* richtet sich nach dem breitesten Rand des übereinstimmenden Präfixes des Musters.
Im Beispiel ist das übereinstimmende Präfix $abcab$; es hat die Länge $j = 5$.
Sein breitester Rand ist ab mit der Breite $b = 2$.
Die Schiebedistanz beträgt $j - b = 5 - 2 = 3$.

Die in der *Vorlaufphase* zu gewinnende Information besteht also darin, für jedes Präfix des Musters die Länge seines breitesten Randes zu bestimmen.

Eine wichtige Beobachtung für die Vorlaufphase:

Lemma: Seien r, s Ränder eines Wortes x mit $\ell(r) < \ell(s)$. Dann ist r ein Rand von s.



Beweis: Das Bild zeigt schematisch x mit den Rändern r und s.

Als Rand von x ist r Präfix von x und damit, weil kürzer als s, auch echtes Präfix von s.

Aber r ist auch Suffix von x und damit echtes Suffix von s. Also ist r Rand von s.

Ist s der breiteste Rand von x, so ergibt sich der nächstschmälere Rand r von x als breitester Rand von s usw.

Ein weiterer wichtiger Begriff

Sei $x \in \Sigma^*$ und $a \in \Sigma$. Ein Rand r von x lässt sich durch a *fortsetzen*, wenn ra Rand von xa ist.

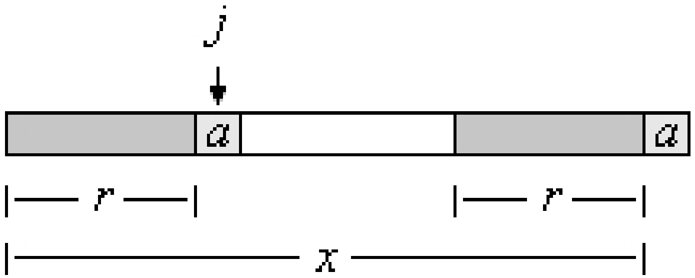
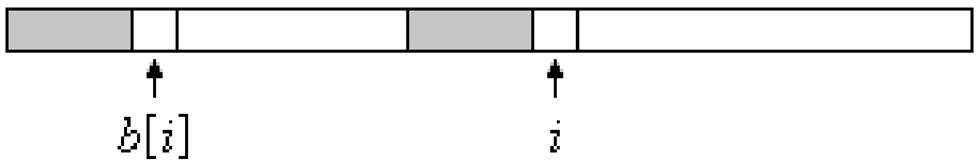


Bild $\leadsto$ Ein Rand r der Breite j von x lässt sich durch a fortsetzen, wenn $x[j] = a$.

Die Vorlaufphase

In der Vorlaufphase wird ein Array b der Länge $m + 1$ berechnet.
Der Eintrag $b[i]$ enthält für jedes Präfix der Länge i des Musters die *Breite seines breitesten Randes* $(i = 0, \ldots, m)$.
Das Präfix λ der Länge $i = 0$ hat keinen Rand; daher wird $b[0] = -1$ gesetzt.



Sind die Werte $b[0], \ldots, b[i]$ bereits bekannt, so ergibt sich $b[i+1]$, indem geprüft wird, ob sich ein Rand des Präfixes $p_0 \ldots p_{i-1}$ durch p_i fortsetzen lässt.
Dies ist der Fall, wenn $p_{b[i]} = p_i$ ist (Bild !).
Die zu prüfenden Ränder ergeben sich nach obigem Lemma in absteigender Breite aus den Werten $b[i]$, $b[b[i]]$ usw.

Ein Beispiel

Beispiel: Für das Muster $p = ababaa$ ergeben sich die Randbreiten im Array b wie folgt. Beispielsweise ist $b[5] = 3$, weil das Präfix $ababa$ der Länge 5 einen Rand der Breite 3 hat.

j	0	1	2	3	4	5	6
$p[j]$	a	b	a	b	a	a	
$b[j]$	-1	0	0	1	2	3	1

Die Vorlaufphase: In C-Code:

```
void kmpPreprocess()
{
    int i=0, j=-1;
    b[i]=j;
    while (i<m)
    {
        while (j>=0 && p[i]!=p[j]) j=b[j];
        i++; j++;
        b[i]=j;
    }
}
```

Knuth-Morris-Pratt Such–Algorithmus

Es werden sogar <u>alle</u> Treffer gemeldet.

```
void kmpSearch()
{
    int i=0, j=0;
    while (i<n)
    {
        while (j>=0 && t[i]!=p[j]) j=b[j];
        i++; j++;
        if (j==m)
        {
            report(i-j);
            j=b[j];
        }
    }
}
```

Sehen Sie den DEA ?

Ein Beispiel mit $p = ababaa$.

a	b	a	b	b	a	b	a	a	…	
a	b	a	b	a	a					
		a	b	a	b	a	a			
				a	b	a	b	a	a	
					a	b	a	b	a	a
							a	b	a	…

j	0	1	2	3	4	5	6
$p[j]$	a	b	a	b	a	a	
$b[j]$	-1	0	0	1	2	3	1