Vorlesungsmitschrift zur Vorlesung Theoretische Informatik I vom 23. Juni 2000

Christian Franz

Inhaltsverzeichnis

Wiederholung: Vorlesung vom 09.06.2000

In der letzten Vorlesung wurde eine algebraische Charakterisierung der regulären Sprachen vorgenommen. Dabei wurde die Relation $\equiv_L$ und der Begriff der Äquivalenzklasse bezüglich $\equiv_L$ eingeführt. Zum Schluß der Vorlesung wurde der Satz von Myhill/Nerode bewiesen. Er besagt, daß die Sprache L genau dann regulär ist, wenn der Index von $\equiv_L$ endlich ist.

Es sollen nun noch einige Beispiele für die Bildung von Äquivalenzklassen $\equiv_L$ zu einer gegebenen Sprache vorgestellt werden:

Beispiele für Äquivalenzklassen

Wdh.
Definition: Äquivalenzklasse bzgl. $\equiv_L$
$[u] := \{v \in X^* \mid u \equiv_L v\} \quad u \in X^*$

Definition: Äquivalenzrelation $\equiv_L$ (Nerode-Äquivalenz zur Sprache L)
$u \equiv_L v \Leftrightarrow \forall w \in X^* (u \cdot w \in L \Leftrightarrow v \cdot w)$

1) $L = \{x \in \{0, 1\}^* \mid x \text{ endet mit } 00\}$

   Offenbar gilt:

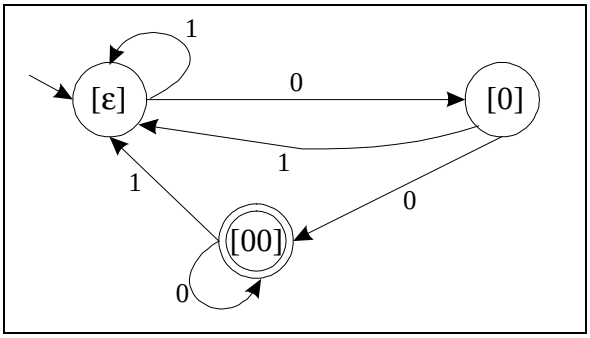
   $[\varepsilon] = \{w \mid w \in \{0, 1\}^* \wedge w \text{ endet nicht mit } 0\}$

   $[0] = \{w \mid w \in \{0, 1\}^* \wedge w \text{ endet mit } 0, \text{ aber nicht mit } 00\}$

   $[00] = \{w \mid w \in \{0, 1\}^* \wedge w \text{ endet mit } 00\}$

   Das sind alle Äquivalenzklassen von $\equiv_L$, also:
   $X^* = \{0, 1\}^* = [\varepsilon] \cup [0] \cup [00]$

   Man kann nun mittels des in Satz T gezeigten Verfahrens einen endlichen deterministischen Akzeptor mit Hilfe der Äquivalenzklassen konstruieren:



2) $L = \{a^n \cdot b^n \mid n \geq 0\}$

Dies ist ein Standardbeispiel für eine Nichtreguläre Sprache, da man zeigen kann, dass $\equiv_L$ keinen endlichen Index hat.

$[\varepsilon] \neq [a]$, denn $\varepsilon \underline{ab} \in L$, aber $a\underline{ab} \notin L$

$[a] \neq [aa]$, denn $a\underline{b} \in L$, aber $aa\underline{b} \notin L$

$[aa] \neq [aaa]$, denn $aa\underline{bb} \in L$, aber $aaa\underline{bb} \notin L$

allgemein:
$[a^i] \neq [a^j]$, falls $i \neq j$, denn $a^i b^i \in L$, aber $a^j b^i \notin L$

Man erhält also unendlich viele verschiedene Äquivalenzklassen, d. h. der Index von $\equiv_L$ ist nicht endlich. Daraus folgt, daß L nicht regulär ist.

4.5. Minimierung von Automaten

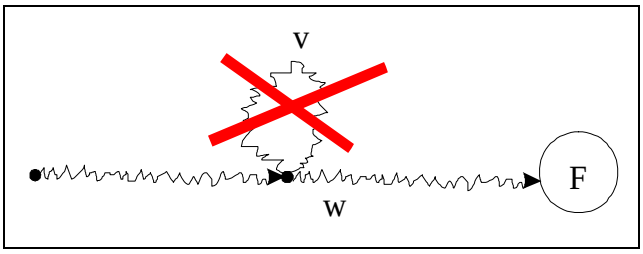
Myhill-Nerode:
Der Äquivalenzklassenautomat ist der Automat mit der kleinsten Anzahl von Zuständen, der sog. *Minimalautomat*.
Denn der Beweis besagt:
Für jeden beliebigen Automaten A´, der die Sprache L erkennt gilt:
$K_{L(A')} \subseteq K_L = K_{A_0}$
wobei A_0 der Minimalakzeptor ist.

Das heißt, die dem Automaten A´ zugeordnete Menge der Äquivalenzklassen $K_{L(A')}$ ist eine Teilmenge der Menge der Äquivalenzklassen $K_L = K_{A_0}$. Das bedeutet, daß die Zahl der Zustände von A´ größer oder gleich der von M_0 ist.

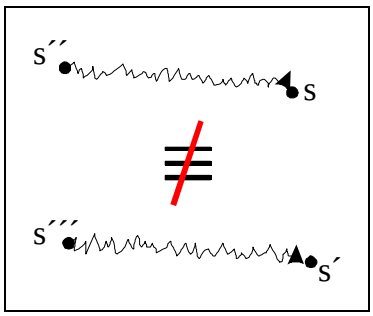
Definition U:
Sei $A = (X, S, \delta, s_0, F)$ ein endl. Akzeptor. Zwei Zustände $s, s' \in S$ heißen *äquivalent*, wenn für alle $w \in X^*$ gilt: $\delta(s, w) \in F \Leftrightarrow \delta(s', w) \in F$.

Bemerkungen:

a) Offenbar können s und s´ verschmolzen werden ohne daß sich L(A) ändert. In diesem Fall verkleinert sich die Anzahl der Zustände des Automaten.
b) Man kann sich bei der Eingabe der Wörter w auf die beschränken, für die $|w| \leq |S|$ gilt, da aus Wörtern dieser Länge bereits alle Informationen für Definition U gewonnen werden können. Dies liegt daran, daß bei allen Wörtern $>|S|$ Zyklen auftreten, die aus w gelöscht werden können.



c) Wenn $s \in F$ und $s' \notin F$ so gilt: s ist nicht äquivalent zu s′. Begründung: Setze $w = \varepsilon$ in Definition U.



Es soll nun ein allgemeiner Algorithmus für die Minimierung von Automaten entwickelt werden:

Idee:
Ausgehend von den Endzuständen teste die Äquivalenz der Zustände mit inkrementeller Verlängerung der betrachteten Wörter.

Eingabe:
Ein deterministischer endlicher Akzeptor $A = (X, S, \delta, s_0, F)$, bei dem alle Zustände bereits entfernt sind, die vom Startzustand nicht erreichbar sind.

Ausgabe:
Angabe, welche Zustände von A zu verschmelzen sind, um den minimalen Automaten zu erhalten.

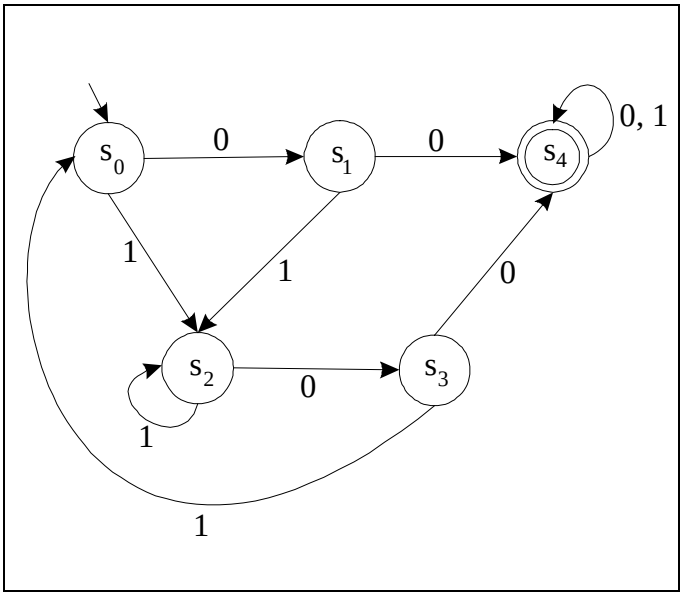
Methode zur Minimierung von Automaten:

1) Bilde Tabelle aller Zustandspaare $\{s, s'\}$ mit $s \neq s'$ von A.
2) Markiere alle Paare $\{s, s'\}$ mit $s \in F$ und $s' \notin F$, oder umgekehrt.
3) Für jedes noch unmarkierte Paar $\{s, s'\}$ und jedes $a \in X$ teste, ob $\{\delta(s, a), \delta(s', a)\}$ bereits markiert ist. Wenn ja, markiere auch $\{s, s'\}$.
4) Wiederhole den Schritt 3) bis die Tabelle keine Änderung mehr ergibt.
5) Alle jetzt noch unmarkierten Paare können jeweils zu einem Zustand verschmolzen werden.

Komplexität des Algorithmus: $O(|S|^2)$ bei geeigneter Implementierung (siehe Hopcroft/Ullmann)

Beispiel für Minimierung von Automaten

Es sei der folgende deterministische Akzeptor gegeben:



Bilde Tabelle aller Zustandspaare (Schritt 1) und markiere alle Paare $\{s, s'\}$ mit $s \in F$ und $s' \notin F$ oder umgekehrt (Schritt 2):

s_1				
s_2				
s_3				
s_4	*	*	*	*
	s_0	s_1	s_2	s_3

Nach Schritt 3 und 4 sieht die Tabelle wie folgt aus:

s_1	*			
s_2		*		
s_3	*		*	
s_4	*	*	*	*
	s_0	s_1	s_2	s_3

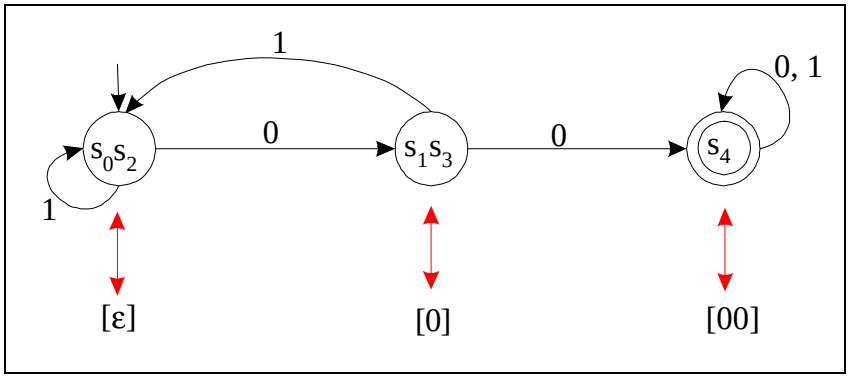
denn:

$\{s_0, s_1\} \xrightarrow{0} \{s_1, s_4\}$ markieren
$\{s_0, s_2\} \xrightarrow{0} \{s_1, s_3\}$
$\{s_0, s_3\} \xrightarrow{0} \{s_1, s_4\}$ markieren
$\{s_1, s_2\} \xrightarrow{0} \{s_3, s_4\}$ markieren
$\{s_1, s_3\} \xrightarrow{0} \{s_4\}$
$\{s_2, s_3\} \xrightarrow{0} \{s_3, s_4\}$ markieren

$\{s_0, s_1\} \xrightarrow{1} \{s_2\}$
$\{s_0, s_2\} \xrightarrow{1} \{s_2\}$
$\{s_0, s_3\} \xrightarrow{1} \{s_0, s_2\}$
$\{s_1, s_2\} \xrightarrow{1} \{s_2\}$
$\{s_1, s_3\} \xrightarrow{1} \{s_0, s_2\}$
$\{s_2, s_3\} \xrightarrow{1} \{s_0, s_2\}$

Die Zustandspaare $\{s_0, s_2\}$ und $\{s_1, s_3\}$ bleiben unmarkiert. Das heißt sie können verschmolzen werden (Schritt 5).

Damit ergibt sich folgender minimaler Automat:



Man sieht, daß dieser Automat die Sprache L = {x∈{0, 1}*|in x kommt 00 vor} beschreibt. Zusätzlich wurden in der oberen Grafik die entsprechenden Äquivalenzklassen eingezeichnet.

Satz V[1]:
Zu jedem deterministischen endlichen Akzeptor A gibt es einen bis auf Isomorphie eindeutig bestimmten minimalen deterministischen endlichen Akzeptor A´ mit L(A) = L(A´).

„Isomorphie“
hier: Gleichgestaltigkeit, Identität bis auf Umbenennung der Zustände

4.6. Pumping-Lemma für reguläre Sprachen

Betrachtet man die regulären Sprachen so gibt es nur einen einzigen Operator, um mit einem regulären Ausdruck unendlich viele Wörter zu beschreiben: die Sternbildung. Also muß sich in diesen Wörtern eine bestimmte Buchstabenkette beliebig oft wiederholen.
Dieselbe Beobachtung kann man bei Automaten machen, die mit nur endlich vielen Zuständen eine unendliche Sprache akzeptieren können. Auch sie müssen Schleifen enthalten.
Diese Beobachtung wird in dem folgenden Pumping-Lemma formalisiert, welches ein zentrales Hilfsmittel zum Nachweis ist, daß eine Sprache nicht regulär ist.

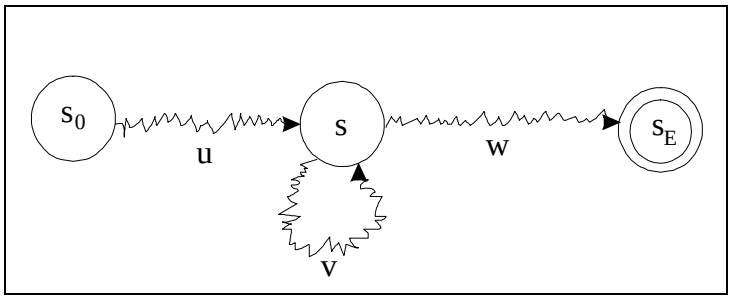
Satz W:
Sei X ein Alphabet. Zu jeder regulären Sprache R⊆X* gibt es ein n∈ℕ, so daß für alle Wörter z∈R mit |z|≥n gilt: Es gibt eine Zerlegung z = uvw mit u, v, w∈X*, v≠ε, |uv|≤n, so daß für alle i≥0 gilt: uv^{i}w∈R.

Formal:
∀R∈Reg∃n∈ℕ∀z∈R,|z|≥n∃u, v, w∈X*:z = uvw ∧ |uv|≤n: ∀i∈ℕ$_0$: uv^{i}w∈R

[1] Ein Beweis hierfür findet sich in [Erk/Priese]

Idee:
Es existiert eine Pumpstelle in allen Wörtern z∈R ab einer gewissen Länge n.

1) endliche Akzeptoren: Um unbeschränkte Wörter zu erkennen, muß ein Akzeptor in Zyklen laufen.



2) reguläre Ausdrücke: Der einzige Konstruktor, um unendliche Sprachen zu erzeugen ist die Sternbildung. Folglich enthalten sehr lange Wörter viele Wiederholungen ein- und desselben Teilworts.

Beispiel für das Pumping-Lemma

$L = \{a^n b^n | n \in \mathbb{N}\}$

Behauptung:
L ist nicht regulär.

Beweis:
Annahme: L ist regulär. Dann gibt es ein Zahl $n \in \mathbb{N}$, so daß sich alle Wörter $z \in L$ der Länge $\geq n$ wie im Pumping-Lemma beschrieben zerlegen lassen. Sei uvw die Zerlegung des Wortes, dann sind folgende 3 Fälle zu beachten:

1) $u \in \{a\}^*$, $v \in \{a\}^*$, $w \in \{b\}^* \Rightarrow$ beim „Pumpen“ ist $uv^i w \notin L$
2) $u \in \{a\}^*$, $v \in \{b\}^*$, $w \in \{b\}^* \Rightarrow$ analog
3) $u \in \{a\}^*$, $v \in \{a\}^*\{b\}^*$, $w \in \{b\}^* \Rightarrow uv^i w \notin L$

Stichwortverzeichnis

Literaturverzeichnis

[Schöning] Schöning, U., „Theoretische Informatik – kurzgefasst“, Spektrum Akademischer Verlag GmbH, 1997

[Erk/Priese] Erk, K., Priese, L., „Theoretische Informatik – Eine umfassende Einführung“, Springer-Verlag, 2000

[Kozen] Kozen, D. C., „Automata and Computability“, Springer-Verlag, 1997