Formale Aspekte endlicher Automaten entdecken

Katharina Wendlandt[1] Matthias Wendlandt[2]

Abstract: Automatentheorie, bzw. formale Sprachen, sind ein fester Bestandteil der meisten Curricula für das Unterrichtsfach Informatik in der Oberstufe. Insbesondere die regulären Sprachen zeichnen sich durch ihre außergewöhnlichen formalen Eigenschaften aus und machen damit endliche Automaten zu einem häufig genutzten Werkzeug in vielen Bereichen auch außerhalb der Informatik. Für die Vermittlung der grundlegenden Arbeitsweise endlicher Automaten, bzw. Mealy- und Moore-Automaten, gibt es bereits zahlreiche, anschauliche Unterrichtskonzepte. Doch insbesondere die formalen Eigenschaften der Automaten (wie beispielsweise Minimierung) begründen den Erfolg und das breite Anwendungsspektrum endlicher Automaten. In der Schule werden die formalen Eigenschaften jedoch häufig ausschließlich abstrakt oder gar nicht behandelt. In unserem Beitrag möchten wir einen Weg zeigen, wie diese Eigenschaft selbstständig und entdeckend von den Schüler*innen erarbeitet werden können.

Keywords: Minimierung, endliche Automaten, entdecken

1 Einleitung

Formale Sprachen werden in der Schule in der Regel exemplarisch durch reguläre Sprachen thematisiert. Auch wenn die regulären Sprachen ein formales Konzept sind, spielen sie in vielen praktischen Bereichen eine wichtige Rolle. So werden sie beispielsweise bei der Textsuche, der Definition von Wertebereichen für Formulare, Konsolenbefehlen und in der lexikalischen Analyse verwendet. Die Bedeutung der Sprachfamilie der regulären Sprachen für die Praxis liegt in deren Abschlusseigenschaften bezüglich sämtlicher Operationen. Für reguläre Sprachen sind sowohl Leere als auch die Universalität und insbesondere das Äquivalenzproblem entscheidbar. Es ist keine andere Sprachfamilie mit ähnlichen, für die Praxis wünschenswerten Eigenschaften bekannt (vgl.[Sc92], Abschnitt 1.5). Die regulären Sprachen lassen sich durch drei formale Beschreibungssysteme charakterisieren: durch endliche Automaten, durch reguläre Grammatiken und durch reguläre Ausdrücke. Diese drei Formalismen sind äquivalent bezüglich ihrer Beschreibungsmächtigkeit. Während Grammatiken und Ausdrücke rein beschreibend sind, arbeiten endliche Automaten auswertend und sind deswegen von besonderer praktischer Bedeutung. Darüber hinaus können endliche Automaten minimiert werden (siehe [Sc92]). Diese herausragende Eigenschaft endlicher Automaten ermöglicht es, den Äquivalenztest und damit auch die Validierung auf eine sehr

[1] Justus-Liebig-Universität Gießen, II. Physikalisches Institut, Heinrich-Buff-Ring 16, 35392 Gießen, Deutschland, katharina.wendlandt@exp2.physik.uni-giessen.de
[2] Justus-Liebig-Universität, Institut für Informatik, Arndtstr. 2, 35392 Gießen, Deutschland, matthias.wendlandt@informatik.uni-giessen.de

doi:10.18420/infos2021_p250

einfache und schnelle Art und Weise durchzuführen. Auch wenn andere Automatenmodelle wie Kellerautomaten, linearbeschränkte Automaten oder sogar Turingmaschinen viel ausdrucksmächtiger sind, so ist die Minimierung für alle diese Modelle nicht möglich (siehe [HMU11]).

Im Schulunterricht werden sowohl die Minimierung als auch andere formale Eigenschaften endlicher Automaten fast ausschließlich nur am Rande thematisiert. Die Thematik der Minimierung beschränkt sich dabei häufig nur auf die Betrachtung des Minimierungs-Algorithmus' endlicher Automaten (oder auch Mealy- und Moore-Automaten). Die Idee, die dem Algorithmus zugrunde liegt, ist die Bildung von Äquivalenzklassen aus der Zustandsmenge. Aus den einzelnen Arbeitsschritten ist diese Idee jedoch nicht direkt ersichtlich. Insbesondere Schüler*innen können den Algorithmus blind anwenden, ohne die Prinzipien, die im Hintergrund wirken, zu verstehen. Das erklärte Ziel der Schule und insbesondere des Informatikunterrichts ist es jedoch nicht, die Schüler*innen zu Anwender*innen zu erziehen. Vielmehr sollen sie allgemeine Prinzipien aus konkreten Sachverhalten extrahieren und dadurch verstehen können. Sie sollen lernen, Ideen und Prinzipien zu kombinieren, um dadurch Lösungsvorschläge für neue Probleme entwickeln zu können (vgl. [Sc93]).

Im Folgenden zeigen wir eine Methode auf, die diese sehr abstrakte Idee des Minimierens auf eine entdeckende Art und Weise vermittelt. Sie ermöglicht es den Schüler*innen, hinter die Idee der Minimierung zu blicken und ermöglicht dadurch ein tiefes Verständnis dieses Lernbereichs, das auch den Transfer der Prinzipien auf neue Probleme zulässt. Ansätze zum Einsatz von entdeckendem und forschendem Lernen im Informatikunterricht wurden bereits in [RAH05] und den LOG IN Beiträgen [Di11] und [LW11] anhand von Themen der praktischen Informatik beschrieben. Der Ansatz, den wir gewählt haben, beruht vor allem auf dem Aufstellen von subjektiven Modellen (Theorien, Vermutungen). Näheres zu diesem Ansatz kann in [Pe99] nachgelesen werden.

2 Minimierung endlicher Automaten

Für die Minimierung endlicher Automaten gibt es im Wesentlichen zwei verschiedene Techniken: Die erste Technik beruht auf einem Algorithmus des theoretischen Informatikers John Brzozowski [Br62]. Zu einem gegebenen, endlichen Automaten A wird, unter Verwendung der üblichen Technik [HMU11], ein endlicher Automat A' konstruiert, der die gespiegelte Sprache $L(A)^R$ erkennt. Diese Technik führt dazu, dass der Automat A' automatisch auch minimal ist. Konstruiert man nun auf dieselbe Art und Weise erneut einen Automaten, der die gespiegelte Sprache von A' erkennt, erhält man einen minimalen Automaten, der die Sprache $(L(A)^R)^R = L(A)$ erkennt. Diese Technik des Minimierens endlicher Automaten ist auf den ersten Blick sehr verwunderlich und die notwendige Idee ist auf den ersten Blick nicht intuitiv erkennbar. Für den Schulunterricht ist diese Technik somit nicht zu empfehlen, da ein tiefer gehendes Verständnis und viel Hintergrundwissen benötigt wird. Als für die Schule geeigneter stellt sich die zweite Technik heraus, die auf

der Bildung von Äquivalenzklassen bezüglich der Nerode-Relation beruht. Häufig werden Äquivalenzklassen sehr formal eingeführt und darüber auch die Minimierung sehr stark formalisiert. Doch die Idee, die hinter der Minimierung steckt, ist sehr intuitiv.

Zunächst betrachten wir den Äquivalenzbegriff näher: Formal ausgedrückt sind zwei Zustände p und q eines Automaten $A = (Q, \Sigma, \delta, q_0, F)$ äquivalent bezüglich der Nerode-Relation, in Zeichen $p \sim q$, wenn gilt $\forall w \in \Sigma^*\colon \delta(p, w) \in F \Leftrightarrow \delta(q, w) \in F$. Zwei Zustände sind also genau dann äquivalent, wenn aus ihnen die gleichen Zeichenketten erkannt werden können. So sind im folgenden Beispiel (Abbildung 1) die Zustände q_7 und q_8 äquivalent, weil aus beiden einzig eine runde schließende Klammer erkannt wird. Ebenfalls sind die Zustände q_5 und q_6 äquivalent, da aus beiden einzig das leere Word (beide sind Endzustände) erkannt wird und zusätzlich eine öffnende und eine schließende Klammer. Die Zustände q_3 und q_4 sind allerdings nicht äquivalent, da aus q_3 die Zeichenketten f und $f()$ erkannt werden und aus Zustand q_4 die Zeichenketten r und $r()$.

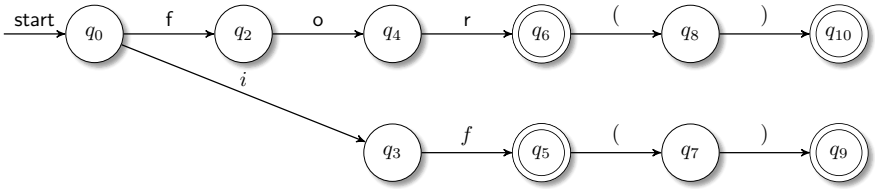

Abb. 1: Deterministischer endlicher Automat

Der Minimierungs-Algorithmus beruht nun auf dem Prinzip, dass Zustände, die äquivalent sind, verschmolzen werden. Der obige Automat kann also minimiert werden zu folgendem Automaten (Abbildung 2).

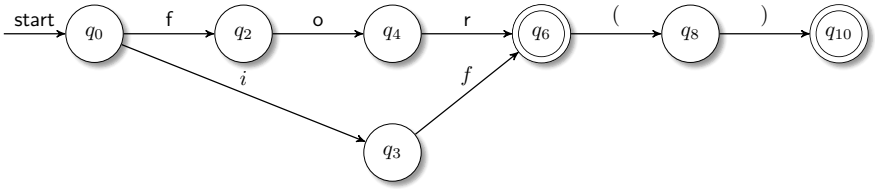

Abb. 2: Minimaler, deterministischer, endlicher Automat

Auch wenn dieses Vorgehen in obigem Beispiel sehr intuitiv aussieht, ist es dies im Allgemeinen nicht (siehe Abbildung 6).

Der sogenannte Table-Filling-Algorithmus zur Minimierung arbeitet folgendermaßen: Zu einem gegebenen Automaten $A = (Q, \Sigma, \delta, s_0, F)$ bestimmt der Algorithmus die Mengen der zu verschmelzenden Zustände, indem zunächst eine Tabelle aller Zustandspaare erstellt wird. In einem ersten Schritt werden alle diejenigen Zustandspaare markiert, die aus einem Nicht-Endzustand und einem Endzustand bestehen. Anschließend wird für jedes Zustandspaar $(p, q) \in Q \times Q$ und jedes Eingabesymbol $a \in \Sigma$ geprüft, ob das Zustandspaar $(\delta(p, a), \delta(q, a))$ in der Tabelle bereits markiert ist. Ist dies der Fall, so

wird auch das Paar (p, q) markiert. Dieser Schritt wird nun so lange wiederholt, bis sich keine neuen Markierungen mehr ergeben. Danach können die Zustände eines unmarkierten Zustandspaares zu einem einzigen Zustand verschmolzen werden. Für eine ausführliche Beschreibung des Table-Filling-Algorithmus siehe Anhang A.

3 Minimierung entdecken - Der Verlauf

Im Unterrichtsprozess soll vor allem darauf geachtet werden, dass die Schüler*innen sehr selbstständig arbeiten und Schritt für Schritt eine subjektive Theorie (subjektive Vermutung) darüber aufstellen, was es bedeutet, dass zwei Zustände eines Automaten äquivalent sind, und wie diese Äquivalenz in einem Minimierungsalgorithmus genutzt werden kann. Im gesamten Unterrichtsprozess gibt es sowohl Lösungskarten als auch Tippkarten, um den Lernprozess der Schüler*innen sehr selbstständig zu gestalten. Der Ablauf in den Aufgabenstellungen ist dabei immer ähnlich. Zunächst wird das Problem, geleitet durch eine Fragestellung, aufgeworfen. Anschließend versuchen die Schüler*innen verschiedene Zustandspaare zu verschmelzen. Diese Phase sowie die darauf folgende Reflexionsphase werden durch Hilfekarten und Lösungskarten unterstützt. Der durchgeführte Unterricht wurde entlang eines „Smiley-Automaten“ aufgebaut (Abbildung 3).

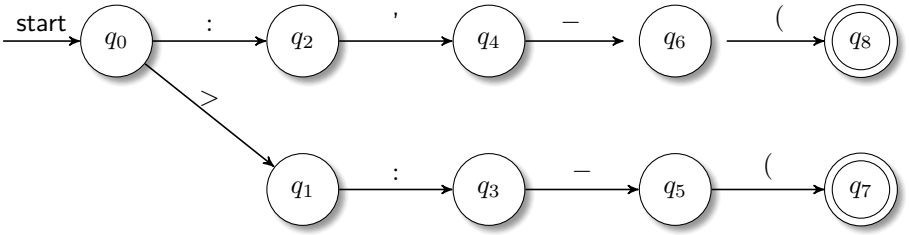

Abb. 3: Erster Beispielautomat

Die Schüler*innen bestimmen zunächst, welche Sprache von dem gegebenen Automaten erkannt wird. Anschließend werden sie gezielt auf die Betrachtung der Zustände q_7 und q_8 hingewiesen. Durch das Streichen eines der beiden Zustände und das geschickte Umlenken der Kanten kann man einen der Zustände einsparen. Anschließend überprüfen die Schüler*innen, ob dies auch für andere Zustandspaare wie beispielsweise $((q_5, q_6), (q_3, q_4))$ möglich ist. Explizit wird auf die Zustände q_1 und q_2 hingewiesen, welche nicht verschmolzen werden können. Es folgt die Reflexion darüber, wann zwei Zustände verschmolzen werden können und wann nicht. Diese Phase ist besonders wichtig. Für die Schüler*innen ergibt sich ein erstes Bild, was zum Verschmelzen führen könnte und was nicht. Ihre Theorie halten die Schüler*innen schriftlich und unterstützt durch Beispiele fest. Zum Abschluss geben Sie eine Einschätzung darüber, wann Zustände verschmolzen werden können.

Das zweite Beispiel (Abbildung 4) geht genauer darauf ein, dass Endzustände und Nicht-Endzustände nicht verschmolzen werden können. Dies ist gleichzeitig der erste Schritt im

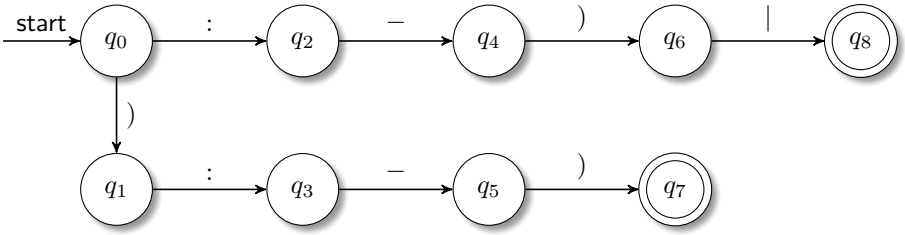

Abb. 4: Zweiter Beispielautomat

Table-Filling-Algorithmus. Die Vorgehensweise ist wie im ersten Beispiel. Zunächst notieren die Schüler*innen die erkannte Sprache. Explizit wird die Frage nach dem Zustandspaar (q_6, q_7) gestellt. Können diese beiden Zustände verschmolzen werden? Die Reflexion wird durch folgende Frage unterstützt: Kann ein Endzustand mit einem Nicht-Endzustand verschmolzen werden? Damit ist ein zweiter wesentlicher Baustein für das Verständnis des Table-Filling-Algorithmus gelegt.

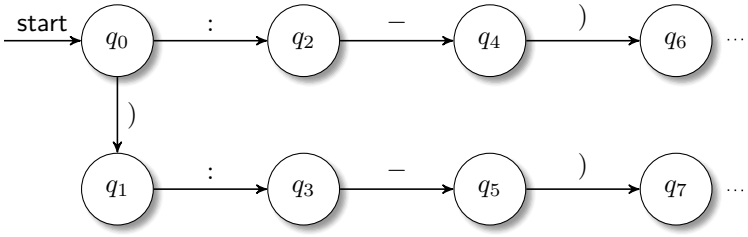

Abb. 5: Dritter Beispielautomat

Der nächste Schritt (Abbildung 5) thematisiert die wiederholende Abfrage des Table-Filling-Algorithmus nach den Folgezuständen zweier Zustände, die untersucht werden sollen (Schritt 3 im Table-Filling-Algorithmus). In diesem Beispiel wurde bewusst auf eine Darstellung des kompletten Automaten verzichtet. An den Zuständen q_6 und q_7 wurde er gewissermaßen „abgeschnitten“. Die Schüler*innen werden durch die folgende Frage in ihrem Entdecken unterstützt: Wenn man bereits weiß, dass die Zustände q_6 und q_7 nicht verschmolzen werden können, kann man dann die Zustände q_4 und q_5 verschmelzen? Dabei sollen die Schüler*innen den Automaten mit gleichen und ungleichen Teilen erweitern, um sich eine Theorie zu erschließen. Auch dieser letzte Schritt wird dann reflektiert.

Zum Abschluss erhalten die Schüler*innen dann die Beschreibung des Table-Filling-Algorithmus' und einen Beispielautomaten, auf den sie den Algorithmus anwenden sollen. Es folgt eine Reflexion: Welche Erkenntnisse aus den Aufgaben 1-3 findet man im Algorithmus wieder? Die Ergebnisse werden zum Abschluss zusammengetragen.

4 Erfahrungen und Ausblick

Die Unterrichtseinheit wurden zunächst mit Studierenden, die den Minimierungsalgorithmus bereits in der Vorlesung kennen gelernt hatten, durchgeführt. Die Rückmeldung dieser Studierenden war eindeutig. Nach ihrer Aussage hatten sie den Minimierungsalgorithmus bisher nur „blind“ angewendet und um damit die nötigen Übungspunkte zu erhalten. Das Prinzip hinter dem Algorithmus war ihnen unklar. Die Einheit führte, laut ihrer Aussagen, dazu, dass sie nicht nur die Minimierung besser verstanden hatten, sondern vor allem auch die eher formale Sichtweise von Äquivalenzklassen und die damit verbundenen Beweise der Vorlesung besser nachvollziehen konnten.

Die Schüler*innen der Testklasse dagegen hatten keine Vorkenntnisse im Bereich der Minimierung von endlichen Automaten. Sie hatten deterministische und nichtdeterministische endliche Automaten bereits kennen gelernt und sie zur Beschreibung von Sprachen genutzt. Nach dem Unterricht konnte die überwiegende Mehrheit der Schüler*innen auf Nachfrage anhand von Beispielen die Minimierung und den Begriff der Äquivalenz erklären. Außerdem war es ihnen möglich, den Algorithmus korrekt anzuwenden.

Die durchweg positiven Erfahrungen, diesen theoretischen Inhalt entdecken zu lassen, führt uns zu zwei Vorhaben: Zum einen stellt sich die Frage, ob diese Idee auch auf andere, theoretischen Eigenschaften übertragen werden kann wie zum Beispiel auf die Determinisierung von nichtdeterministischen endlichen Automaten. Außerdem könnte eine empirische Studie über den tatsächlich messbaren Nutzen dieser Herangehensweise Aufschluss geben.

A Theoretische Grundlagen

Zur Veranschaulichung wählen wir folgenden Beispielautomaten. Der Automat hat fünf Zustände.

Die einzelnen Schritte des Table-Filling-Algorithmus’ sind:

1. Stelle eine Tabelle aller Zustandspaare auf. Obwohl der Automat fünf Zustände hat, muss die Tabelle lediglich vier Zeilen und vier Spalten haben, da ein Zustand nicht mit sich selbst verglichen werden muss. Außerdem kann an der Diagonalen gekürzt werden, da die Relation symmetrisch ist.

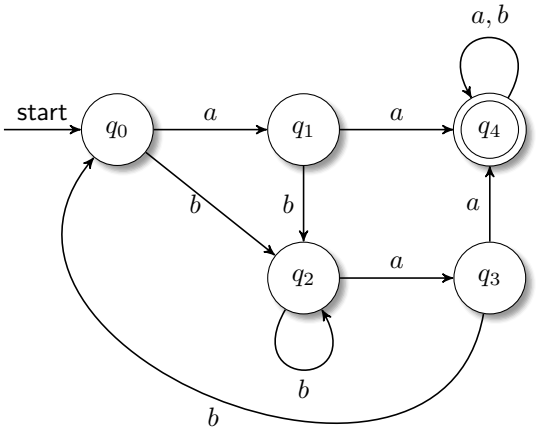

Abb. 6: Deterministischer endlicher Automat

Beispiel

	q_0	q_1	q_2	q_3
q_4				
q_3				
q_2				
q_1				

2. Markiere alle Paare aus Nicht-Endzuständen und Endzuständen.

Beispiel

	q_0	q_1	q_2	q_3
q_4	x	x	x	x
q_3				
q_2				
q_1				

3. Teste für jedes Paar aus Zuständen p, q und jedes Eingabezeichen x, ob das Paar $\delta(p,x), \delta(q,x)$ markiert ist. Wenn ja, dann markiere auch das Paar p, q.

Beispiel

	q_0	q_1	q_2	q_3
q_4	x	x	x	x
q_3	x			
q_2				
q_1				

4. Wiederhole Schritt 3, bis sich keine Änderung in der Tabelle mehr ergibt.

Beispiel

	q_0	q_1	q_2	q_3
q_4	x	x	x	x
q_3	x		x	
q_2		x		
q_1	x			

5. Alle jetzt noch unmarkierten Paare können jeweils zu einem Zustand veschmolzen werden.

Der Algorithmus nimmt Markierungen vor, wenn zwei Zustände nicht äquivalent sein können. Endzustände und Nichtendzustände werden in Schritt 2 markiert, da mindestens das leere Wort die Zustände voneinander trennt. In Schritt 3 werden dann aufbauend auf diesen Markierungen weitere Markierungen vorgenommen. Wenn ein Zeichen x aus einem Zustandspaar (p,q) in ein Zustandspaar (s,t) führt und (s,t) markiert ist, dann wird auch (p,q) markiert, da es ein trennendes Wort w für (s,t) gibt und somit xw ein trennendes Wort für (p,q) ist.

Zum Abschluss werden sowohl die Zustände q_0 und q_2 als auch die Zustände q_1 und q_3 verschmolzen (siehe Abbildung 7).

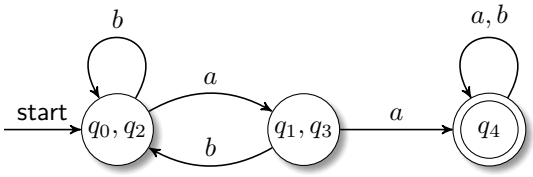

Abb. 7: Deterministischer endlicher Automat

Literatur

[Br62] Brzozowski, J. A.: Canonical regular expressions and minimal state graphs for definite events. Mathematical Theory of Automata/, S. 529–561, 1962.

[Di11] Diethelm, I.: Wie forschend-entdeckendes Lernen gelingen kann - Forschendes und entdeckendes Lernen in Kontexten zu Datenschutz, Internet und Urheberrecht. LOG IN 31/168, S. 28–34, 2011.

[HMU11] Hopcroft, J. E.; Motwani, R.; Ullman, J. D.: Einführung in Automatentheorie, formale Sprachen und Berechenbarkeit. Pearson Studium, 2011.

[LW11] Leonhardt, T.; Westram, H.: Das Projekt go4IT! - Mit Robotern forschend und entdeckend lernen. LOG IN 31/168, S. 56–61, 2011.

[Pe99] Peterßen, W. H.: Kleines Methoden-Lexikon. Oldenbourg, 1999.

[RAH05] Reichert, R.; Arnold, R.; Hartmann, W.: Entdeckendes Lernen im Informatik-Unterricht. In (Friedrich, S., Hrsg.): Unterrichtskonzepte für informatische Bildung, INFOS 2005, 11. GI-Fachtagung Informatik und Schule, 28.-30. September 2005 an der TU Dresden. Bd. P-60. LNI, GI, S. 197–205, 2005.

[Sc92] Schöning, U.: Theoretische Informatik kurz gefasst. BI-Wissenschaftsverlag, 1992.

[Sc93] Schwill, A.: Fundamentale Ideen der Informatik. Zentralblatt für Didaktik der Mathematik 25/, S. 20–31, 1993.