WWW.PEARSON-STUDIUM.DE
CWS
COMPANION WEBSITE

it informatik

John E. Hopcroft
Rajeev Motwani
Jeffrey D. Ullman

Einführung in Automatentheorie, Formale Sprachen und Berechenbarkeit

3., aktualisierte Auflage

ADDISON-WESLEY

PEARSON Studium

John E. Hopcroft,
Rajeev Motwani,
Jeffrey D. Ullman

Einführung in Automatentheorie, Formale Sprachen und Berechenbarkeit

3., aktualisierte Auflage

PEARSON
Studium

ein Imprint von Pearson Education
München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Daraus folgt: Wenn einer dieser Fälle zutrifft, dann ist $h(w)$ nicht in L enthalten. Zudem gilt, dass w die Form *baba ... ba* hat, sofern keiner der Fälle (1) bis (4) zutrifft.

Um das zu sehen, wollen wir annehmen, dass keine der Bedingungen (1) bis (4) erfüllt ist. Dann wissen wir auf Grund von (1), dass w mit b beginnen muss, und aus (2) geht hervor, dass w auf a enden muss. Aus (3) und (4) können wir schließen, dass sich die Symbole a und b in w abwechseln müssen. Folglich ist die logische ODER-Verknüpfung von (1) bis (4) äquivalent mit der Aussage »w hat nicht die Form *baba ... ba*«. Wir haben bewiesen, dass die ODER-Verknüpfung von (1) bis (4) impliziert, dass $h(w)$ nicht in L enthalten ist. Diese Aussage stellt die Umkehrung der gewünschten Aussage dar: »Wenn $h(w)$ in L enthalten ist, dann hat w die Form *baba ... ba*.« ■

Wer werden als Nächstes beweisen, dass der inverse Homomorphismus einer regulären Sprache auch regulär ist, und dann zeigen, wie dieser Satz eingesetzt werden kann.

Satz 4.8 Wenn h ein Homomorphismus von Alphabet Σ nach Alphabet T und L eine reguläre Sprache über T ist, dann ist auch $h^{-1}(L)$ eine reguläre Sprache.

Beweis ■ Der Beweis beginnt mit einem DEA A für L. Wir konstruieren aus A und h unter Verwendung des in Abbildung 4.6 dargestellten Plans einen DEA für $h^{-1}(L)$. Dieser DEA verwendet die Zustände von A, übersetzt die Eingabesymbole jedoch gemäß h, bevor er den Übergang in den nächsten Zustand vollzieht.

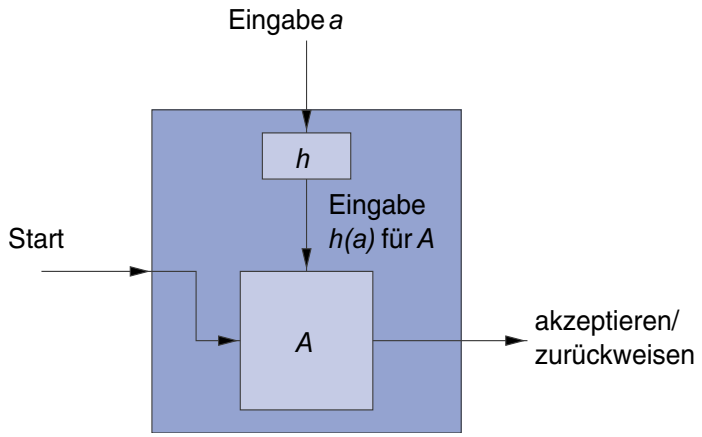

Abbildung 4.6: Der DEA für $h^{-1}(L)$ wendet h auf seine Eingabe an und simuliert dann den DEA für L

Formal ausgedrückt, L sei $L(A)$, wobei DEA $A = (Q, T, \delta, q_0, F)$. Definiere einen DEA

$$B = (Q, \Sigma, \gamma, q_0, F),$$

dessen Übergangsfunktion γ nach der Regel $\gamma(q, a) = \hat{\delta}(q, h(a))$ gebildet wird. Das heißt, die Übergänge, die B auf die Eingabe a hin durchführt, sind ein Ergebnis der Folge von Übergängen, die A auf die Zeichenreihe $h(a)$ hin vollzieht. Bedenken Sie, dass es sich bei $h(a)$ um ε, um ein Symbol oder um eine Zeichenreihe handeln kann. Die Übergangsfunktion $\hat{\delta}$ ist allerdings so definiert, dass sie alle diese Fälle behandeln kann.

Mit einer einfachen Induktion über $|w|$ lässt sich zeigen, dass $\hat{\gamma}(q_0, w) = \hat{\delta}(q_0, h(w))$. Da A und B über dieselben akzeptierenden Zustände verfügen, akzeptiert B die Zeichenreihe w genau dann, wenn A $h(w)$ akzeptiert. Anders ausgedrückt, B akzeptiert genau die Zeichenreihen, die in $h^{-1}(L)$ enthalten sind. ■

Beispiel 4.9 In diesem Beispiel werden wir inverse Homomorphismen und einige andere Abschluss-Eigenschaften regulärer Mengen verwenden, um eine Merkwürdigkeit in Bezug auf endliche Automaten zu beweisen. Angenommen, wir forderten, dass ein DEA jeden Zustand mindestens einmal besucht, wenn er seine Eingabe akzeptiert. Genauer gesagt, angenommen, $A = (Q, \Sigma, \delta, q_0, F)$ sei ein DEA, und uns interessiert die Sprache L aller Zeichenreihen w aus Σ^*, derart dass $\hat{\delta}(q_0, w)$ in F enthalten ist und es zudem für jeden Zustand q aus Q ein Präfix x_q von w gibt, derart dass $\hat{\delta}(q_0, x_q) = q$. Ist L regulär? Wir können dies beweisen, allerdings ist die Konstruktion kompliziert.

Zuerst beginnen wir mit der Sprache M, die $L(A)$ ist, d. h. die Menge der Zeichenreihen, die der DEA A in der üblichen Weise akzeptiert, also ungeachtet dessen, welche Zustände er während der Verarbeitung der Eingabe besucht. Beachten Sie, dass $L \subseteq M$, da die Definition von L den in $L(A)$ enthaltenen Zeichenreihen eine zusätzliche Bedingung auferlegt. Unser Beweis, dass L regulär ist, beginnt mit einem inversen Homomorphismus, bei dem die Zustände von A in die Eingabesymbole eingesetzt werden. Genauer gesagt, wir wollen ein neues Alphabet T definieren, das aus Symbolen besteht, die man sich als Tripel $[paq]$ vorstellen kann, wobei gilt:

1. p und q sind in Q enthaltene Zustände.
2. a ist ein in Σ enthaltenes Symbol.
3. $\delta(p, a) = q$.

Wir können die in T enthaltenen Symbole als Repräsentationen der Zustandsübergänge von A betrachten. Es ist unbedingt zu beachten, dass die Notation $[paq]$ ein einziges Symbol darstellt und es sich nicht um die Verkettung von drei Symbolen handelt. Wir könnten auch einzelne Buchstaben zur Benennung verwenden, doch dann wäre ihre Beziehung zu p, q und a schwer zu beschreiben.

Nun definieren wir den Homomorphismus $h([paq]) = a$ für alle p, a und q. Das heißt, h entfernt die Zustandskomponenten aus jedem der in T enthaltenen Symbole, sodass nur das Symbol aus Σ übrig bleibt. Unser erster Schritt zum Beweis der Regularität von L besteht in der Konstruktion der Sprache $L_1 = h^{-1}(M)$. Da M regulär ist, ist nach Satz 4.8 auch L_1 regulär. Bei den in L_1 enthaltenen Zeichenreihen handelt es sich um

die Zeichenreihen aus M, deren einzelnen Symbolen ein Zustandspaar angehängt wurde und die so einen Zustandsübergang repräsentieren.

Betrachten Sie zur Veranschaulichung den Automaten mit zwei Zuständen in Abbildung 4.4 (a). Das Alphabet Σ ist {0, 1}, und das Alphabet T besteht aus den vier Symbolen $[p0q]$, $[q0q]$, $[p1p]$ und $[q1q]$. Es gibt beispielsweise für die Eingabe 0 einen Übergang von Zustand p nach q, und daher ist $[p0q]$ eines der in T enthaltenen Symbole. Die Zeichenreihe 101 wird vom Automaten akzeptiert. Die Anwendung von h^{-1} auf diese Zeichenreihe ergibt $2^3 = 8$ Zeichenreihen, wobei $[p1p][p0q][q1q]$ und $[q1q][q0q][p1p]$ zwei Beispiele für diese Zeichenreihen sind.

Wir werden nun L aus L_1 konstruieren, indem wir eine Reihe weiterer Operationen ausführen, die die Regularität erhalten. Unser erstes Ziel besteht darin, all jene Zeichenreihen aus L_1 zu eliminieren, die Zustandsfolgen nicht korrekt beschreiben. Das heißt, wir können uns vorstellen, dass ein Symbol wie $[paq]$ Folgendes aussagt: Der Automat war im Zustand p, las die Eingabe a und ging daraufhin in den Zustand q über. Die Symbolfolge muss drei Bedingungen erfüllen, wenn sie eine akzeptierende Berechnung von A repräsentieren soll:

1. Das erste Symbol muss als ersten Zustand q_0 enthalten, den Startzustand von A.
2. Jeder Übergang muss da beginnen, wo der letzte aufgehört hat. Das heißt, in einem Symbol muss der erste Zustand gleich dem zweiten Zustand des vorherigen Symbols sein.
3. Der zweite Zustand des letzten Symbols muss in F enthalten sein. Diese Bedingung wird tatsächlich schon garantiert, wenn wir (1) und (2) erzwingen, da wir wissen, dass jede in L_1 enthaltene Zeichenreihe aus einer von A akzeptierten Zeichenreihe resultiert.

Abbildung 4.7 zeigt den Plan zur Konstruktion von L.

Wir erzwingen (1), indem wir den Durchschnitt von L_1 und der Menge von Zeichenreihen bilden, die für ein Symbol a und einen Zustand q mit einem Symbol der Form $[q_0aq]$ beginnen. Das heißt, sei E_1 der Ausdruck $[q_0a_1q_1] + [q_0a_2q_2] + \ldots$, wobei a_iq_i alle Paare in $\Sigma \times Q$ umfasst, derart dass $\delta(q_0, a_i) = q_i$, dann sei $L_2 = L_1 \cap L(E_1T^*)$. Da E_1T^* ein regulärer Ausdruck ist, der alle Zeichenreihen aus T^* beschreibt, die mit dem Startzustand beginnen, enthält L_2 alle Zeichenreihen, die durch die Anwendung von h^{-1} auf die Sprache M gebildet werden und den Startzustand als erste Komponente des ersten Symbols enthalten; d. h. L_2 erfüllt (1).

Um Bedingung (2) zu erzwingen, ist es am einfachsten, alle Zeichenreihen, die diese Bedingung nicht erfüllen, von L_2 (mit dem Operator für die Mengendifferenz) abzuziehen. Sei E_2 der reguläre Ausdruck, der aus der Summe (Vereinigung) der Verkettungen aller Symbolpaare besteht, die nicht zusammenpassen, d. h. Paare der Form $[paq][rbs]$, wobei $q \neq r$; dann ist $T^*E_2T^*$ ein regulärer Ausdruck, der alle Zeichenreihen beschreibt, die die Bedingung (2) nicht erfüllen.

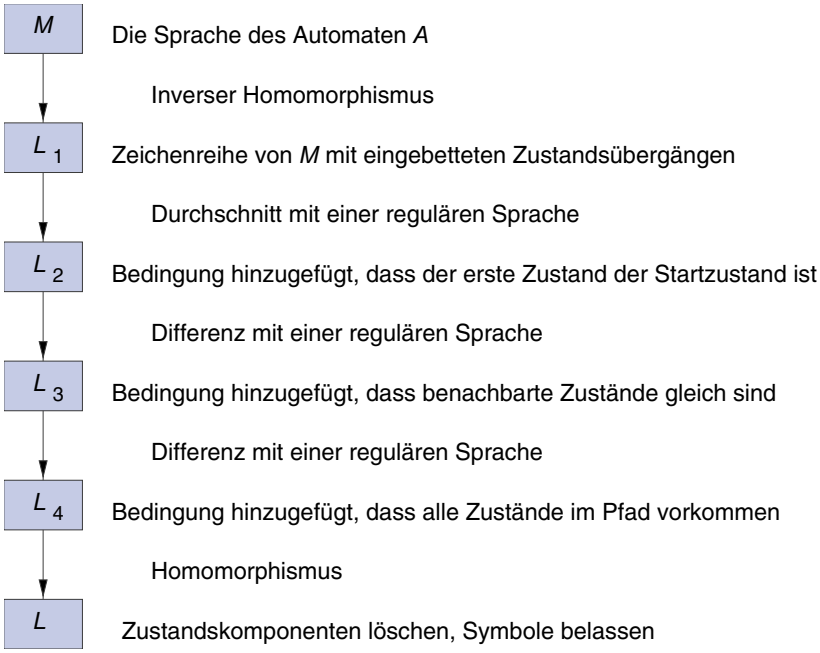

Abbildung 4.7: Wie die Sprache *L* mithilfe von Operationen, die die Regularität von Sprachen erhalten, aus der Sprache *M* konstruiert wird

Wir können nun definieren $L_3 = L_2 - L(T^*E_2T^*)$. Die Zeichenreihen in L_3 erfüllen die Bedingung (1), weil die Zeichenreihen in L_2 mit dem Startsymbol beginnen müssen. Sie erfüllen Bedingung (2), weil die Subtraktion von $L(T^*E_2T^*)$ jede Zeichenreihe entfernt, die diese Bedingung verletzt. Schließlich erfüllen sie Bedingung (3), dass der letzte Zustand ein akzeptierender sein muss, weil wir lediglich mit Zeichenreihen aus M begonnen haben, die alle zu einem akzeptierenden Zustand von A führen. Infolgedessen besteht L_3 aus den Zeichenreihen aus M, in denen die Zustände der akzeptierenden Berechnung als Bestandteile in die einzelnen Symbole eingebettet sind. Beachten Sie, dass die Sprache L_3 regulär ist, weil auf die reguläre Menge M nur eine Folge von Operationen (inverser Homomorphismus, Durchschnitt und Mengendifferenz) angewandt wird, die die Regularität erhalten.

Unser Ziel bestand ursprünglich darin, nur jene Zeichenreihen aus M zu akzeptieren, während deren akzeptierender Berechnung jeder Zustand besucht wurde. Wir können diese Bedingung durch weitere Anwendungen des Mengendifferenzoperators erzwingen. Das heißt, für jeden Zustand q sei E_q der reguläre Ausdruck, der die Summe aller in T enthaltenen Symbole beschreibt, derart dass q weder an der ersten noch an der zweiten Position vorkommt. Wenn wir $L(E_q^*)$ von L_3 subtrahieren, dann erhalten wir die Zeichenreihen, die eine akzeptierende Berechnung von A repräsentieren und die mindestens einmal zum Zustand q führen. Wenn wir von L_3 alle Sprachen $L(E_q^*)$ für q aus Q subtrahieren, dann erhalten wir die akzeptierenden Berechnungen von A, die

alle Zustände berühren. Wir nennen diese Sprache L_4. Nach Satz 4.5 wissen wir, dass auch L_4 eine reguläre Sprache ist.

Der letzte Schritt besteht darin, durch Beseitigung der Zustandskomponenten L aus L_4 zu konstruieren. Das heißt, $L = h(L_4)$. L ist die Menge der in Σ^* enthaltenen Zeichenreihen, die von A akzeptiert werden und während ihrer Berechnung mindestens einmal zu jedem Zustand von A führen. Da reguläre Sprachen bezüglich Homomorphismen abgeschlossen sind, wissen wir, dass L regulär ist. ■

4.2.5 Übungen zum Abschnitt 4.2

Übung 4.2.1 Angenommen, h ist ein Homomorphismus vom Alphabet {0, 1, 2} zum Alphabet {a, b} und wie folgt definiert: $h(0) = a$; $h(1) = ab$ und $h(2) = ba$.

* a) Was ergibt $h(0120)$?

b) Was ergibt $h(21120)$?

* c) Wenn L die Sprache $L(\mathbf{01}^*\mathbf{2})$ ist, was ist dann $h(L)$?

d) Wenn L die Sprache $L(\mathbf{0} + \mathbf{12})$ ist, was ist dann $h(L)$?

* e) Angenommen, L ist die Sprache $\{ababa\}$, d. h. die Sprache, die lediglich aus der einen Zeichenreihe *ababa* besteht. Was ist dann $h^{-1}(L)$?

! f) Wenn L die Sprache $L(\mathbf{a}(\mathbf{ba})^*)$ ist, was ist dann $h^{-1}(L)$?

*! **Übung 4.2.2** Wenn L eine Sprache und a ein Symbol ist, dann ist L/a, der Quotient von L und a, die Menge der Zeichenreihen w, derart dass wa in L enthalten ist. Wenn beispielsweise $L = \{a, aab, baa\}$, dann ist $L/a = \{\varepsilon, ba\}$. Beweisen Sie, dass L/a regulär ist, wenn L regulär ist. *Hinweis*: Beginnen Sie mit einem DEA für L und betrachten Sie die Menge der akzeptierenden Zustände.

! **Übung 4.2.3** Wenn L eine Sprache und a ein Symbol ist, dann ist $a\backslash L$ die Menge der Zeichenreihen w, derart dass aw in L enthalten ist. Wenn beispielsweise $L = \{a, aab, baa\}$, dann ist $a\backslash L = \{\varepsilon, ab\}$. Beweisen Sie, dass $a\backslash L$ regulär ist, wenn L regulär ist. *Hinweis*: Denken Sie daran, dass reguläre Sprachen bezüglich der Spiegelung und der Quotientenoperation aus 4.2.2 abgeschlossen sind.

! **Übung 4.2.4** Welche der folgenden Identitäten sind wahr?

a) $(L/a)a = L$ (die linke Seite repräsentiert die Verkettung der Sprachen L/a und $\{a\}$)

b) $a(a\backslash L) = L$ (hier ist wieder eine Verkettung mit {a} gemeint)

c) $(La)/a = L$

d) $a\backslash(aL) = L$

Übung 4.2.5 Die in Übung 4.2.3 beschriebene Operation wird gelegentlich auch als »Ableitung« betrachtet und $a\backslash L$ wird wie folgt dargestellt: $\frac{dL}{da}$. Diese Ableitungen werden auf reguläre Ausdrücke in ähnlicher Weise angewandt, wie normale Ableitungen auf arithmetische Ausdrücke. Wenn R ein regulärer Ausdruck ist, dann soll $\frac{dR}{da}$ dasselbe bedeuten wie $\frac{dL}{da}$, wenn $L = L(R)$.

a) Zeigen Sie, dass $\frac{d(R+S)}{da} = \frac{dR}{da} + \frac{dS}{da}$.

*! b) Formulieren Sie die Regel für die Ableitung von RS. *Hinweis*: Sie müssen zwei Fälle unterscheiden: Wenn $L(R)$ die leere Zeichenfolge ε enthält und wenn nicht. Diese Regel entspricht nicht ganz der »Produktregel« für gewöhnliche Ableitungen, ähnelt ihr jedoch.

! c) Formulieren Sie die Regel für die Ableitung einer Hülle, d. h. für $\frac{d(R^*)}{da}$.

d) Verwenden Sie die Regeln aus (a) bis (c), um die Ableitung für den regulären Ausdruck $(\mathbf{0} + \mathbf{1})^*\mathbf{011}$ in Bezug auf 0 und 1 zu finden.

* e) Charakterisieren Sie diejenigen Sprachen L, für die gilt $\frac{dL}{d0} = \emptyset$.

*! f) Charakterisieren Sie diejenigen Sprachen L, für die gilt $\frac{dL}{d0} = L$.

! **Übung 4.2.6** Zeigen Sie, dass die regulären Sprachen bezüglich der folgenden Operationen abgeschlossen sind:

a) $min(L) = \{w \mid w$ ist in L enthalten, aber kein eigentliches Präfix von w ist in L enthalten$\}$.

b) $max(L) = \{w \mid w$ ist in L enthalten, und für kein anderes x als ε ist wx in L enthalten$\}$.

c) $init(L) = \{w \mid$ für ein x ist wx in L enthalten$\}$.*

Hinweis: Wie bei Übung 4.2.2 ist es am einfachsten mit einem DEA für L zu beginnen und eine Konstruktion anzugeben, um die gewünschte Sprache zu erhalten.

! **Übung 4.2.7** Wenn $w = a_1a_2 \ldots a_n$ und $x = b_1b_2 \ldots b_n$ Zeichenreihen derselben Länge sind, definieren Sie $alt(w, x)$ als die Zeichenreihe, in der die Symbole von w und x abwechseln, wobei mit w begonnen werden soll, d. h. $a_1b_1a_2b_2 \ldots a_nb_n$. Wenn L und M Sprachen sind, definieren Sie $alt(L, M)$ als die Menge der Zeichenreihen der Form $alt(w, x)$, wobei w eine beliebige Zeichenreihe aus L und x eine beliebige Zeichenreihe aus M derselben Länge ist. Beweisen Sie, dass $alt(L, M)$ regulär ist, wenn L und M regulär sind.

*!! **Übung 4.2.8** Sei L eine Sprache. Definieren Sie $half(L)$ als die Menge der ersten Hälften der in L enthaltenen Zeichenreihen, d. h. $\{w \mid$ für ein x mit $|x| = |w|$ ist wx in $L\}$. Wenn z. B. $L = \{\varepsilon, 0010, 011, 010110\}$, dann ist $half(L) = \{\varepsilon, 00, 010\}$. Beachten Sie, dass Zeichenreihen mit einer ungeraden Anzahl von Zeichen nicht zu $half(L)$ beitragen. Beweisen Sie, dass $half(L)$ eine reguläre Sprache ist, wenn L regulär ist.

---

* x ist eine Zeichenreihe.

!! **Übung 4.2.9** Wir können Übung 4.2.8 für eine Reihe von Funktionen verallgemeinern, die die Größe des Anteils an den Zeichenreihen festlegen. Wenn f eine Funktion ganzer Zahlen ist, definieren Sie $f(L)$ wie folgt: $\{w \mid$ für ein x mit $|x| = f(|w|)$ ist wx in $L\}$. Die Operation *half* entspricht z. B. der Identitätsfunktion $f(n) = n$, da die Definition von *half*(L) die Bedingung $|x| = |w|$ enthält. Zeigen Sie, dass $f(L)$ eine reguläre Sprache ist, wenn L eine reguläre Sprache ist und wenn f für eine der folgenden Funktionen steht:

a) $f(n) = 2n$ (d. h. nimm das erste Drittel der Zeichenreihen)

b) $f(n) = n^2$ (d. h. die Anzahl der zu übernehmenden Zeichen ist gleich der Quadratwurzel der Anzahl der Zeichen, die nicht übernommen werden)

c) $f(n) = 2^n$ (d. h. die Anzahl der zu übernehmenden Zeichen ist gleich dem dualen Logarithmus der Anzahl der Zeichen, die nicht übernommen werden)

!! **Übung 4.2.10** Angenommen, L ist irgendeine Sprache, die nicht unbedingt regulär zu sein braucht und deren Alphabet gleich {0} ist; d. h. die Zeichenreihen von L bestehen ausschließlich aus Nullen. Beweisen Sie, dass L^* regulär ist. *Hinweis*: Dieser Satz mag auf den ersten Blick absurd scheinen. Ein Beispiel kann Ihnen vielleicht verdeutlichen, warum er wahr ist. Betrachten Sie die Sprache $L = \{0^i \mid i$ ist eine Primzahl$\}$, von der wir aus Beispiel 4.2 wissen, dass sie nicht regulär ist. Die Zeichenreihen 00 und 000 sind in L enthalten, da 2 und 3 Primzahlen sind. Folglich können wir für $j \geq 2$ zeigen, dass 0^j in L^* enthalten ist. Wenn j gerade ist, dann verwenden wir j/2-mal 00, und wenn j ungerade ist, dann verwenden wir einmal 000 und $(j - 3)$/2-mal 00. Daraus ergibt sich $L^* = \mathbf{000}^* \cup \{\varepsilon\}$.

!! **Übung 4.2.11** Zeigen Sie, dass die regulären Sprachen bezüglich der folgenden Operation abgeschlossen sind: *cycle*(L) = $\{w \mid$ wir können w in der Form $w = xy$ schreiben, sodass yx in L enthalten ist$\}$. Wenn z. B. $L = \{01, 011\}$, dann ist *cycle*(L) = {01, 10, 011, 110, 101}. *Hinweis*: Beginnen Sie mit einem DEA für L und konstruieren Sie einen ε-NEA für *cycle*(L).

!! **Übung 4.2.12** Sei $w_1 = a_0a_0a_1$ und $w_i = w_{i-1}w_{i-1}a_i$ für alle $i > 1$. Beispielsweise ist $w_3 = a_0a_0a_1a_0a_0a_1a_2a_0a_0a_1a_0a_0a_1a_2a_3$. Der kürzeste reguläre Ausdruck für die Sprache $L_n = \{w_n\}$, d. h. die Sprache, die aus der Zeichenreihe w_n besteht, ist die Zeichenreihe w_n, und die Länge dieses Ausdrucks ist $2^{n+1} - 1$. Wenn wir jedoch den Durchschnittsoperator zulassen, dann können wir einen Ausdruck für L_n formulieren, dessen Länge $O(n^2)$ ist. Finden Sie einen solchen Ausdruck. *Hinweis*: Finden Sie n Sprachen, die jeweils durch einen regulären Ausdruck der Länge $O(n)$ beschrieben werden und deren Schnittmenge L_n ist.

! **Übung 4.2.13** Die Abschluss-Eigenschaften erleichtern Beweise, dass bestimmte Sprachen nicht regulär sind. Beginnen Sie mit der Tatsache, dass die Sprache

$$L_{0n1n} = \{0^n1^n \mid n \geq 0\}$$

keine reguläre Menge ist. Beweisen Sie, dass die folgenden Sprachen nicht regulär sind, indem Sie sie unter Verwendung von Operationen, die bekanntermaßen die Regularität erhalten, in L_{0n1n} transformieren:

* a) $[0^i1^j \mid i \neq j\}$

b) $[0^n1^m2^{n-m} \mid n \geq m \geq 0\}$

Übung 4.2.14 In Satz 4.4 beschrieben wir die »Produktkonstruktion«, mit der aus zwei DEAs ein DEA konstruiert wurde, dessen Sprache in der Schnittmenge der Sprachen der ersten beiden DEAs enthalten ist.

a) Zeigen Sie, wie die Produktkonstruktion für NEAs (ohne ε-Übergänge) durchgeführt wird.

! b) Zeigen Sie, wie die Produktkonstruktion für ε-NEAs durchgeführt wird.

* c) Zeigen Sie, in welcher Weise die Produktkonstruktion modifiziert werden muss, damit der resultierende DEA die Differenz der Sprachen der beiden gegebenen DEAs akzeptiert.

d) Zeigen Sie, in welcher Weise die Produktkonstruktion modifiziert werden muss, damit der resultierende DEA die Vereinigung der Sprachen der beiden gegebenen DEAs akzeptiert.

Übung 4.2.15 Im Beweis von Satz 4.4 behaupteten wir, dass durch Induktion über die Länge von w bewiesen werden könnte, dass

$$\hat{\delta}((q_L,q_M), w) = (\hat{\delta}_L(q_L, w), \hat{\delta}_M(q_M, w)).$$

Führen Sie diesen induktiven Beweis.

Übung 4.2.16 Vervollständigen Sie den Beweis von Satz 4.7, indem Sie die Fälle behandeln, in denen E eine Verkettung von zwei Teilausdrücken bzw. die Hülle eines Ausdrucks darstellt.

Übung 4.2.17 In Satz 4.8 haben wir einen Induktionsbeweis über die Länge von w ausgelassen, nämlich dass $\hat{\gamma}(q_0, w) = \hat{\delta}(q_0, h(w))$. Beweisen Sie diese Aussage.

4.3 Entscheidbarkeits-Eigenschaften regulärer Sprachen

In diesem Abschnitt betrachten wir, wie man wichtige Fragen zu regulären Sprachen beantworten kann. Zuerst müssen wir darauf eingehen, was es bedeutet, eine Frage zu einer Sprache zu stellen. Sprachen sind normalerweise unendlich, und man kann also im Allgemeinen die Zeichenreihen einer Sprache nicht jemandem zeigen und dann eine Frage stellen, zu deren Beantwortung die unendliche Menge von Zeichenreihen untersucht werden müsste. Stattdessen präsentieren wir eine Sprache, indem wir eine der endlichen Repräsentationen angeben: einen DEA, NEA, ε-NEA oder regulären Ausdruck.

Natürlich wird die auf diese Weise beschriebene Sprache regulär sein, und es gibt tatsächlich gar keine Möglichkeit, eine völlig beliebige Sprache endlich zu repräsentieren. In späteren Kapiteln werden wir endliche Verfahren zur Darstellung von weiteren Sprachen neben den regulären kennen lernen, sodass wir Fragen zu Sprachen im Hinblick auf diese allgemeineren Klassen betrachten können. Für viele Fragen gibt es allerdings nur für die Klasse der regulären Sprachen Algorithmen. Diese Fragen werden »unentscheidbar« (es gibt keinen Algorithmus zu deren Beantwortung), wenn sie zu »ausdrucksstärkeren« Notationen (d. h. Notationen, mit denen umfangreichere Mengen von Sprachen beschrieben werden können) gestellt werden, als zu den Repräsentationen, die wir für reguläre Sprachen entwickelt haben.

Wir beginnen unser Studium der Algorithmen für Fragen zu regulären Sprachen, indem wir Möglichkeiten betrachten, wie die Repräsentation einer Sprache in eine andere Repräsentation derselben Sprache umgewandelt werden kann. Insbesondere werden wir die Zeitkomplexität der Algorithmen betrachten, die diese Umwandlungen durchführen. Anschließend erörtern wir einige grundlegende Fragen zu Sprachen:

1. Ist die beschriebene Sprache leer?
2. Ist eine bestimmte Zeichenreihe w in der beschriebenen Sprache?
3. Beschreiben zwei Repräsentationen einer Sprache tatsächlich dieselbe Sprache? Diese Frage wird häufig als **»Äquivalenz«** von Sprachen bezeichnet.

4.3.1 Wechsel zwischen Repräsentationen

Wir wissen, dass wir jede der vier Repräsentationen von regulären Sprachen in jede der drei anderen Repräsentationen umwandeln können. Abbildung 3.1 hat die Pfade gezeigt, die von einer Repräsentation zu den anderen führen. Obwohl es Algorithmen für die verschiedenen Umwandlungen gibt, sind wir nicht nur an der Möglichkeit einer Umwandlung, sondern auch an dem damit verbundenen Zeitaufwand interessiert. Insbesondere ist es wichtig, zwischen Algorithmen, die einen exponentiellen Zeitaufwand (als Funktion der Größe ihrer Eingabe) erfordern und daher nur für relativ kleine Beispiele verwendbar sind, und solchen, die einen bezüglich der Größe ihrer Eingabe linearen, quadratischen oder in geringem Umfang polynomiellen Zeitaufwand erfordern. Letztere Algorithmen sind in dem Sinn »realistisch«, als wir erwarten, dass sie auch für umfangreiche Beispiele ausführbar sind. Wir werden die Zeitkomplexität aller hier erörterten Umwandlungen betrachten.

NEAs in DEAs umwandeln

Wenn wir mit einem NEA oder einem ε-NEA beginnen und ihn in einen DEA umwandeln, dann kann der Zeitaufwand in der Anzahl der Zustände des NEA exponentiell sein. Die Berechnung der ε-Hülle von n Zuständen nimmt $O(n^3)$ Zeit in Anspruch. Wir müssen von jedem der n Zustände allen mit ε beschrifteten Pfeilen nachgehen. Wenn es n Zustände gibt, dann kann es nicht mehr als n^2 Pfeile geben. Überlegte

Buchführung und gut entworfene Datenstrukturen können sicherstellen, dass die Erforschung von jedem Zustand aus in $O(n^2)$ Zeit erfolgt. Mit einem transitiven Hüllenbildungsalgorithmus wie dem Warshall-Algorithmus kann die gesamte ε-Hülle sogar in einem Durchgang berechnet werden.[2]

Nachdem die ε-Hülle berechnet wurde, können wir mithilfe der Teilmengenkonstruktion den äquivalenten DEA berechnen. Hierbei ergeben die Zustände des DEA, deren Anzahl bis zu 2^n betragen kann, die dominanten Kosten. Wir können unter Zuhilfenahme der ε-Hüllendaten und der Übergangstabelle des NEA in $O(n^3)$ Zeit für jeden Zustand die Übergänge für alle Eingabesymbole berechnen. Angenommen, wir möchten für den DEA $\delta(\{q_1, q_2, ..., q_k\}, a)$ berechnen. Von jedem Zustand q_1 aus können auf mit ε beschrifteten Pfaden n Zustände erreichbar sein, und jeder dieser Zustände kann bis zu n Pfeile mit der Beschriftung a besitzen. Indem wir ein mit Zuständen indiziertes Array anlegen, können wir die Vereinigung von bis zu n Mengen von bis zu n Zuständen in einem zu n^2 proportionalen Zeitraum berechnen.

Auf diese Weise können wir für jeden Zustand q_i die Menge der Zustände berechnen, die auf einem Pfad mit der Beschriftung a (möglicherweise einschließlich Pfeilen mit der Beschriftung ε) erreicht werden können. Da $k \leq n$, gibt es höchstens n Zustände zu behandeln. Wir berechnen die erreichbaren Zustände für jeden Zustand in $O(n^2)$ Zeit. Folglich beträgt der gesamte Zeitraum, der zur Berechnung von erreichbaren Zuständen aufgewendet werden muss, $O(n^3)$. Die Vereinigung der Mengen von erreichbaren Zuständen erfordert nur $O(n^2)$ zusätzliche Zeit, und wir schließen daraus, dass die Berechnung eines DEA-Übergangs $O(n^3)$ Zeit erfordert.

Beachten Sie, dass die Anzahl der Eingabesymbole konstant ist und nicht von n abhängt. Folglich berücksichtigen wir in dieser und anderen Schätzungen der Ausführungszeit die Anzahl der Eingabesymbole nicht als Faktor. Die Größe des Eingabealphabets beeinflusst den konstanten Faktor, der in der »O«-Notation verborgen ist, aber sonst nichts.

Unsere Schlussfolgerung ist, dass die Ausführungszeit einer NEA-in-DEA-Umwandlung (einschließlich NEAs mit ε-Übergängen) $O(n^3 2^n)$ beträgt. Nun ist es in der Praxis häufig so, dass eine viel geringere Anzahl von Zuständen als 2^n erzeugt wird, oft sogar nur n Zustände. Wir können also den oberen Grenzwert für die Ausführungszeit mit $O(n^3 s)$ angeben, wobei s die Anzahl der Zustände angibt, die der DEA tatsächlich besitzt.

DEAs in NEAs umwandeln

Diese Umwandlung ist einfach und erfordert $O(n)$ Zeit für einen DEA mit n Zuständen. Wir müssen lediglich die Übergangstabelle des DEA abändern, indem wir die Zustände in geschweifte Klammern setzen, um sie als Mengen zu kennzeichnen, und eine Spalte für ε hinzufügen, wenn es sich um einen ε-NEA handelt. Da wir die

2 Transitive Hüllenbildungsalgorithmen werden in A. V. Aho, J. E. Hopcroft und J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, 1984, näher behandelt.

Copyright