5 Entscheidbarkeit des Leerheitsproblems

Nach den Überlegungen in Kapitel 1 muss nur noch gezeigt werden, dass sich algorithmisch feststellen lässt, ob eine von einem endlichen Automaten erkannte Sprache leer ist oder nicht, um Korrektheit nachzuweisen. Denn die Sprache des Automaten, der das Sytem modelliert, enthält alle möglichen Abläufe und arbeitet korrekt, wenn kein möglicher Ablauf verboten ist. Wenn also die verbotenen Sequenzen auch durch einen endlichen Automaten erkannt werden, muss man nach Kapitel 4 nur den Produktautomaten konstruieren und diesen auf Leerheit seiner Sprache testen.

Sei $A = (Z, I, d, s_o, F)$ ein deterministischer Automat. Sei

$$H(A) = \{s \in Z \mid d^*(s, w) \in F \text{ für ein } w \in I^*\}$$

die Menge aller Zustände, die zu Endzuständen führen. Dann gilt offensichtlich:

$$L(A) \neq \emptyset \text{ gdw. } s_0 \in H(A).$$

$H(A)$ lässt sich folgermaßen iterieren:

$H_0 = F$ und $H_{i+1} = H_i \cup \{s \in Z \mid d(s, x) \in H_i \text{ für ein } x \in I\}$ sowie $H(A) = H_m$ für das kleinste m mit $H_{m+1} = H_m$.

Nach Definition gilt: $F = H_0 \subseteq H_1 \subseteq \cdots \subseteq H_i \subseteq H_{i+1} \subseteq \cdots \subseteq Z$. Da Z endlich ist, können nur endlich viele dieser Inklusionen echt sein, so dass es m geben muss. Dann gilt auch $H_{m+k} = H_m$ für alle $k \in \mathbb{N}$ (einfache Induktion über k). Daraus folgt die gewünschte Gleichheit. Denn mit Induktion über i ergibt sich $H_i \subseteq H(A)$:

IA: $H_0 = F \subseteq H(A)$ wegen $d^*(s, \lambda) = s \in F$ für alle $s \in F$.
IS: $H_{i+1} = H_i \cup \{s \in Z \mid d(s, x) \in H_i \text{ für ein } x \in I\}$
$\subseteq H(A) \cup \{s \in Z \mid d(s, x) \in H(A) \text{ für ein } x \in I\}$
$= H(A) \cup \{s \in Z \mid d^*((d)s, x), w)) \in F \text{ für ein } w \in I^*, x \in I\}$
$= H(A) \cup \{s \in Z \mid d^*(s, xw) \in F \text{ für ein } xw \in I^*\}$
$\subseteq H(A) \cup H(A) = H(A)$.
Insbesondere gilt $H_m \subseteq H(A)$.

Betrachte umgekehrt $s \in H(A)$. Dann gibt es $w \in I^*$ mit $d^*(s, w) \in F$. Das impliziert $s \in H_{length(w)}$, wie eine einfache Induktion ergibt. Daraus folgt wie gewünscht: $s \in H_{length(w)} \subseteq H_m$.

6 Reguläre Sprachen und reguläre Ausdrücke

Neben dem Erkennen von Sprachen ist es interessant, ihre Kompositionalität oder Modularität zu untersuchen. Dabei geht es darum, wie sich umfangreiche und komplizierte Sprachen aus einfachen Bausteinen aufbauen lassen. Diese Frage ist auch für die Entwicklung großer Spezifikationen und informationstechnischer Systeme von zentraler Bedeutung. Es zeigt sich, dass sich die von endlichen Automaten erkannten Sprachen in besonders einfacher Weise modular aufbauen lassen.

Dieses Verhalten wird in drei Schritten verwirklicht. Im Abschnitt 6.1 wird zuerst eine modulare Komposition von Sprachen eingeführt, die zu den sogenannten regulären Sprachen führt und von endlichen Automaten realisiert werden kann. Im Abschnitt 6.2 wird dann in einem nicht ganz einfachen Beiweis hergeleitet, dass jede von einem endlichen Automaten erkannte Sprachen schon selbst regulär ist, sich also modular aufbauen lässt. Im Abschnitt 6.3 schließlich werden reguläre Ausdrücke eingeführt, die eine syntaktische Beschreibung regulärer Sprachen liefern.

6.1 Reguläre Operationen

Die in Abbildung 4 gezeigten drei endlichen Automaten erkennen (a) die leere Menge $\emptyset$, (b) die einelementige Sprache $\{\lambda\}$, die das leere Wort enthält, bzw. (c) die einelementige Sprache $\{x\}$, die das Wort x der Länge 1 enthält. Diese Sprachen dienen als kleinste, unzerlegbare Sprachmoduln.

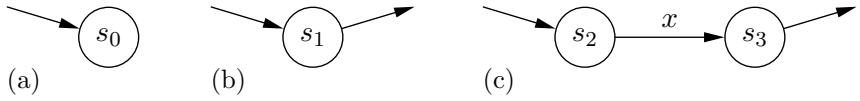

Abbildung 4: Endliche Automaten für atomare Sprachen: (a) $\emptyset$ (b) $\{\lambda\}$ (c) $\{x\}$ mit $x \in I$

Sind L, L_1 und L_2 Sprachen zum Alphabet I, die von endlichen Automaten erkannt werden, so werden auch die zusammengesetzten Sprachen $L_1 \cup L_2$, $L_1 L_2$ und L^* von endlichen Automaten erkannt. Die dabei verwendeten Sprachoperationen sind die Vereinigung, die Konkatenation ($L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1,\, w_2 \in L_2\}$) und die Kleene-Hülle ($L^* = \bigcup_{i \in \mathbb{N}} L^i$ mit $L^0 = \{\lambda\}$ und $L^{i+1} = L^i L$).

Bezeichnet $\mathcal{L}_{AUT(I)}$ die Klasse aller Sprachen über dem Alphabet I, die von endlichen Automaten erkannt werden, so hat $\mathcal{L}_{AUT(I)}$ nach den obigen Überlegungen folgende Eigenschaften:

(i) $\emptyset, \{\lambda\}, \{x\} \in \mathcal{L}_{AUT(I)}$ für $x \in I$,
(ii) $L, L_1, L_2 \in \mathcal{L}_{AUT(I)}$ impliziert $L_1 \cup L_2,\ L_1 L_2,\ L^* \in \mathcal{L}_{AUT(I)}$.

Die kleinste Klasse $\mathcal{L}_{REG(I)}$ von Sprachen mit diesen Eigenschaften sind die sogenannten *regulären Sprachen*, die durch endlich-maliges Anwenden der Sprachoperationen in (ii), beginnend mit den Sprachen in (i), entstehen. Die regulären Sprachen sind also vollständig modular aufgebaut. Insbesondere gilt: $\mathcal{L}_{REG(I)} \subseteq \mathcal{L}_{AUT(I)}$.

6.2 Endliche Automaten erkennen reguläre Sprachen

Bemerkenswerterweise kann auch die Umkehrung nachgewiesen werden, so dass sich die von endlichen Automaten erkannten Sprachen als regulär erweisen und somit aus den atomaren Sprachmoduln allein durch endlich viele Vereinigungen, Konkatenationen und Kleene-Hüllen-Bildungen aufgebaut werden können.

Theorem 2
Sei $A = (Z, I, d, s_0, F)$ ein endlicher Automat. Dann ist $L(A)$ regulär.

Beweis.
Ohne Beschränkung der Allgemeinheit kann $Z = \{1, \ldots, n\}$ und $s_0 = 1$ angenommen werden. Dann ist für $i, j \in Z$ und $k \in \mathbb{N}$ die Sprache $L^k_{i,j}$ aller Wörter, die im Zustandsgraphen von i nach j führen, ohne zwischendurch Zustände jenseits von k zu besuchen, definiert als:

$$L^k_{i,j} = \{w \in I^* \mid j \in d^*(i,w),\ d^*(i,u) \subseteq \{1,\ldots,k\} \text{ für alle } u \text{ mit } w = uv,\ w \neq u \neq \lambda\}.$$

Mit Induktion über k kann gezeigt werden, dass $L^k_{i,j}$ für alle $i, j \in Z$ und $k \in \mathbb{N}$ regulär ist.

IA: Für $k = 0$ und $i \neq j$ enthält $L^k_{i,j}$ die Eingaben, die direkt von i nach j führen, weil alle Zustände jenseits von 0 liegen und deshalb nicht besucht werden dürfen. D.h. $L^0_{i,j} = \{x \in I \mid j \in d(i,x)\}$. Diese Sprache ist entweder leer oder die endliche Vereinigung von atomaren regulären Sprachen und deshalb selbst regulär.

Für $k = 0$ und $i = j$ gehört in jedem Fall noch λ zur Sprache, d.h. $L^0_{i,i} = \{\lambda\} \cup \{x \in I \mid i \in d(i,x)\}$, was sich analog als regulär erweist, weil $\{\lambda\}$ regulär ist.

IV: Als Induktionsvoraussetzung wird von der Regularität von $L^k_{i,j}$ für alle i und j ausgegangen.

IS: Betrachte nun im Induktionsschluss $L^{k+1}_{i,j}$ bzw. ein Element w daraus. Dieses Wort induziert einen Weg von i nach j im Zustandsgraph. Sei $l_0 \cdots l_p$ die durchlaufene Sequenz von Zuständen mit $i = l_0$ und $j = l_p$. Kommt der Zustand $k+1$ gar nicht in $l_1 \cdots l_{p-1}$ vor, so liegt w in $L^k_{i,j}$. Ansonsten kommt der Zustand $k+1$ m-mal darin vor mit $0 < m < p$, so dass die Sequenz folgende Form hat:

$$l_0 \cdots l_{p_1-1}(k+1)l_{p_1+1} \cdots l_{p_2-1}(k+1) \cdots (k+1)l_{p_m+1} \cdots l_p,$$

wobei $p_1 < p_2 < \cdots < p_m$ die Stellen sind, wo $k+1$ auftritt. Insbesondere kommt in den Abschnitten $l_1 \cdots l_{p_1-1}, l_{p_1+1} \cdots l_{p_2-1}, \ldots, l_{p_m+1} \cdots l_{p-1}$ der Zustand $k+1$ nicht vor, sondern nur die Zustände $1, \ldots, k$. Es stellt sich also heraus, dass $w = w_0 w_1 \ldots w_m$ für Wörter $w_0 \in L^k_{i,k+1}$, $w_1, \ldots, w_{m-1} \in L^k_{k+1,k+1}$ und $w_m \in L^k_{k+1,j}$, d.h. $w \in L^k_{i,k+1}(L^k_{k+1,k+1})^* L^k_{k+1,j}$. Damit ist gezeigt, dass

$$L^{k+1}_{i,j} \subseteq L^k_{i,j} \cup L^k_{i,k+1}(L^k_{k+1,k+1})^* L^k_{k+1,j}.$$

Nach Definition von $L^k_{i,j}$ gilt auch die umgekehrte Inklusion, so dass insgesamt

$$L^{k+1}_{i,j} = L^k_{i,j} \cup L^k_{i,k+1}(L^k_{k+1,k+1})^* L^k_{k+1,j}$$

folgt. Nach Induktionsvoraussetzung sind $L^k_{i,j}$, $L^k_{i,k+1}$, $L^k_{k+1,k+1}$ und $L^k_{k+1,j}$ regulär, so dass auch die Kleene-Hülle der dritten Sprache und die Konkatenation mit den anderen beiden Sprachen sowie die Vereinigung mit der ersten Sprache regulär sind. Also ist $L^{k+1}_{i,j}$ regulär, was zu zeigen war.

Die Sprachen $L^k_{i,j}$ sind also für alle $i, j \in Z$ und $k \in \mathbb{N}$ regulär. Die von A erkannte Sprache ist eine Vereinigung endlich vieler dieser Sprachen, denn es gilt $L(A) = \bigcup_{j \in F} L^n_{1,j}$. Somit ist auch $L(A)$ regulär, wie behauptet. □

6.3 Reguläre Ausdrücke

Reguläre Ausdrücke sind, analog zu arithmetischen Ausdrücken, aus Konstanten, Operationen und Klammern aufgebaute Zeichenketten, die einen Wert beschreiben. Der Wert soll in diesem Fall allerdings keine Zahl sein, sondern eine Sprache. Demzufolge müssen die Konstanten möglichst einfache Sprachen repräsentieren und die Operationen müssen es erlauben, aus diesen Sprachen komplexere zu bilden.

In der einen oder anderen Form dürften reguläre Ausdrücke den meisten schon einmal über den Weg gelaufen sein, da sie in Editoren mit komfortablen Suchfunktionen und in UNIX-Kommandos wie z.B. grep, find und sed Verwendung finden.

Sei I ein Alphabet mit $\mathit{lambda}, \mathit{empty}, +, \circ, *, (,) \notin I$. Die Menge $\mathit{REX}(I)$ der *regulären Ausdrücke über* I ist rekursiv definiert durch:

(i) $\mathit{empty}, \mathit{lambda} \in \mathit{REX}(I)$,
(ii) $I \subseteq \mathit{REX}(I)$ und
(iii) für alle $r, r_1, r_2 \in \mathit{REX}(I)$ sind auch die Wörter $(r_1 + r_2)$, $(r_1 \circ r_2)$ und (r^*) in $\mathit{REX}(I)$.

Jedem regulären Ausdruck r über I wird wie folgt eine Sprache $L(r)$ zugeordnet:

(i) $L(\mathit{empty}) = \emptyset$, $L(\mathit{lambda}) = \{\lambda\}$ und $L(x) = \{x\}$ für alle $x \in I$,
(ii) für alle $r, r_1, r_2 \in \mathit{REX}(I)$ sei
  (1) $L((r_1 + r_2)) = L(r_1) \cup L(r_2)$,
  (2) $L((r_1 \circ r_2)) = L(r_1)L(r_2) = \{w_1 w_2 \mid w_1 \in L(r_1), w_2 \in L(r_2)\}$ und
  (3) $L((r^*)) = L(r)^* = \{w_1 \cdots w_n \mid w_1, \ldots, w_n \in L(r), n \in \mathbb{N}\}$.

Bemerkungen

1. Nach der Interpretation regulärer Ausdrücke steht $+$ für die Vereinigung zweier Sprachen, $\circ$ für deren Konkatenation und $*$ für die Iteration einer Sprache. Die Verwendung der Symbole $+$ und $\circ$ für Vereinigung und Konkatenation hat einen guten Grund: Abstrakt gesehen, verhalten sich diese Operationen wie Addition und Multiplikation (genauer gesagt, die Menge der Wörter bildet mit diesen Operationen einen Semiring). Dabei ist $\emptyset$ das neutrale Element der Addition (die "Null") und $\{\lambda\}$ ist das der Multiplikation (die "Eins"). Wie gewohnt ergibt die Multiplikation mit Null immer Null: $\emptyset L = \emptyset = L\emptyset$. Beide Operationen sind offenbar assoziativ, aber nur die Vereinigung ist auch kommutativ, denn $L_1 L_2$ ist natürlich im allgemeinen nicht dasselbe wie $L_2 L_1$. Wie man sich leicht überzeugen kann, gelten auch die Distributivgesetze: $L(L_1 \cup L_2) = LL_1 \cup LL_2$ und $(L_1 \cup L_2)L = L_1 L \cup L_2 L$. Damit sind alle Zutaten beisammen, die man für einen Semiring benötigt (zu einem "echten" Ring fehlen die inversen Elemente bezüglich der Addition). Übrigens hat auch der Kleene-Stern einige interessante Eigenschaften. Insbesondere ist er idempotent, $L^{**} = L^*$, was praktisch direkt aus der Definition folgt.
2. Um reguläre Ausdrücke übersichtlicher zu machen, wird üblicherweise von der Konvention Gebrauch gemacht, dass $*$ stärker bindet als $\circ$ und dies wiederum stärker als $+$. Klammern, die nach dieser Konvention (oder der Assoziativität von $+$ und $\circ$) zur Vermeidung von Mehrdeutigkeiten unnötig sind, können weggelassen werden. Außerdem lässt man das Symbol $\circ$ oft weg, analog zur Schreibweise bei arithmetischen Ausdrücken, wo man ja auch oft xy für $x \cdot y$ schreibt. Demnach kann man beispielsweise statt $(((a \circ b)^*) \circ (c \circ (c^*)))$ auch kurz $(ab)^* cc^*$ schreiben.
3. Gelegentlich wird in der Literatur ein regulärer Ausdruck r angegeben, wenn eigentlich $L(r)$ gemeint ist, sofern dies aus dem Kontext genügend klar hervorgeht. Dies sollte aber nicht darüber hinwegtäuschen, dass r und $L(r)$ zwei grundverschiedene Dinge sind: r ist eine Zeichenkette, während $L(r)$ eine Sprache ist.

Aus der Definition der regulären Ausdrücke und ihrer Interpretation sowie der Definition der regulären Sprachen folgt unmittelbar, dass eine Sprache $L \subseteq I^*$ genau dann regulär ist, wenn es einen regulären Ausdruck $r \in REX(I)$ mit $L = L(r)$ gibt. Reguläre Ausdrücke stellen somit einen weiteren Formalismus neben endlichen Automaten dar, um reguläre Sprachen zu spezifizieren. Formal gilt:

$$\mathcal{L}_{REG(I)} = \mathcal{L}_{AUT(I)} = \mathcal{L}_{REX(I)},$$

wobei $\mathcal{L}_{REX(I)} = \{L(r) \mid r \in REX(I)\}$ alle durch Interpretation der regulären Ausdrücke über I erhaltenen Sprachen enthält.

7 Pumping-Lemma für erkannte Sprachen

Endliche Automaten haben als Modellierungskonzept viele brauchbare Eigenschaften: Ihre Sprache lässt sich schnell erkennen, sie besitzen neben der graphisch-visuellen auch eine textuell-kompositionelle Beschreibung, und sie erlauben automatische Korrektheitsbeweise. Sie wären ein ideales Modellierungsinstrument, wenn es nicht vieles Interessante gäbe, was mit ihnen nicht modelliert werden kann. Um das einsehen zu können, soll zunächst eine Eigenschaft aller von endlichen Automaten erkannten Sprachen bewiesen werden. Wenn es eine Sprache gibt, die diese Eigenschaft nicht hat, so gibt es demnach keinen endlichen Automaten, der sie erkennt.

Es wird ein Pumping Lemma gezeigt, das aussagt, dass genügend lange Wörter einer von einem endlichen Automaten erkannten Sprache ein Teilwort besitzen, das beliebig oft wiederholt werden kann, ohne dass man dabei die Sprache verlässt.

Theorem 3 (Erstes Pumping-Lemma)
Sei L eine von einem endlichen Automaten erkannte Sprache.

Dann existiert eine natürliche Zahl $p \in \mathbb{N}$ derart, dass jedes Wort $w \in L$ mit $\mathit{length}(w) \geq p$ zerlegt werden kann in drei Teilwörter $w = xyz$ mit $p \geq \mathit{length}(y) > 0$ und $p > \mathit{length}(x)$ und dass $xy^iz \in L$ ist für alle $i \geq 0$.

Beweis.
Nach Voraussetzung existiert ein Automat $A = (Z, I, d, s_0, F)$, der L erkennt. Wähle dann p als Anzahl der Zustände von A ($p = \#Z$). Gibt man nun ein Wort $w = x_1 \cdots x_n \in L$ ($x_i \in I$) mit $n \geq p$ in A ein, so erhält man durch buchstabenweises Abarbeiten eine Folge $s_0 \cdots s_n$ von Zuständen mit der Eigenschaft

$$s_i \in d(s_{i-1}, x_i) \text{ für } i = 1, \ldots, n.$$

Wegen $n \geq p$ gibt es unter den ersten $p+1$ Zuständen $s_0, \ldots, s_p$ zwei gleiche, etwa $s_j = s_k$ mit $0 \leq j < k \leq p$. Das liefert die gewünschte Zerlegung von w, denn mit $x = x_1 \cdots x_j$ kommt man von s_0 nach s_j, mit $y = x_{j+1} \cdots x_k$ von s_j nach $s_k = s_j$, was man dann aber auch immer wiederholen oder auslassen kann, und von s_k gelangt man mit $z = x_{k+1} \cdots x_n$ nach $s_n \in F$. Insgesamt erreicht man also mit den Wörtern xy^iz für $i \geq 0$ von s_0 den Endzustand s_n. Das aber bedeutet gerade $xy^iz \in L$, wie behauptet.

Nach Konstruktion ist $\mathit{length}(x) = j < p$ und $\mathit{length}(y) = k - j$ mit $0 \leq \mathit{length}(y) \leq p$.

□

Beispiel: Anwendung des Pumping-Lemmas

1. Das Pumping-Lemma kann benutzt werden, um zu zeigen, dass eine Sprache von keinem endlichen Automaten erkannt wird.

Betrachte die Sprache $L_{balance} = \{a^n b^n \mid n \in \mathbb{N}\}$. Angenommen, sie wird von einem endlichen Automaten erkannt. Sei dann $p \in \mathbb{N}$ die Konstante aus dem Pumping-Lemma, und wähle $w = a^p b^p \in L_{balance}$. Nach dem Pumping-Lemma kann w in drei Teilwörter $w = xyz$ mit $y \neq \lambda$ zerlegt werden, so dass $xy^i z \in L_{balance}$ für alle $i \in \mathbb{N}$. Es gibt drei Fälle, wie y in w liegen kann:

(i) y liegt in der ersten Hälfte von w, d.h. $y = a^k$ für ein $k > 0$.
Dies kann nicht sein, denn $xy^0 z = xz = a^{p-k} b^p \notin L_{balance}$ für $k \neq 0$.

(ii) y liegt in der Mitte von w, d.h. $y = a^k b^l$ für $k, l > 0$.
Auch das ist nicht möglich, denn $xy^2 z = a^p b^l a^k b^p \notin L_{balance}$ für $k \neq 0 \neq l$.

(iii) y liegt in der zweiten Hälfte von w, d.h. $y = b^k$ für ein $k > 0$.
Dies ist analog zu Fall (i) ausgeschlossen.

Also gibt es keine Zerlegung von w mit den geforderten Eigenschaften, was bedeutet, dass die Annahme falsch gewesen sein muss.

2. Man kann das Pumping-Lemma *nicht* verwenden, um zu zeigen, dass es für eine gegebene Sprache einen erkennenden Automaten gibt.
Betrachte die Sprache $L' = \{w \in \{a, b\}^* \mid count_a(w) = count_b(w)\}$. In jedem nichtleeren Wort $w \in L'$ gibt es eine Stelle, in der ein a auf ein b trifft. Demnach gibt es eine Zerlegung $w = xyz$ mit $y = ab$ oder $y = ba$, und weiter gilt für alle $i \in \mathbb{N}$, dass $xy^i z \in L'$. Es wäre aber falsch, daraus zu schließen, dass es einen endlichen Automaten gibt, der L' erkennt.
Tatsächlich gibt es keinen solchen Automaten. Intuitiv liegt das daran, dass beim Abarbeiten eines Eingabewortes eine beliebig große Differenz zwischen der Anzahl der gelesenen a's und der der b's entstehen kann, aber in den endlich vielen Zuständen eines endlichen Automaten kann maximal nur eine beschränkt große Differenz gespeichert werden. Der formale Beweis muss an dieser Stelle geschuldet bleiben.