ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
http://www.ita.inf.ethz.ch/theoInf18

Lösungsvorschläge – Blatt 9

Zürich, 23. November 2018

Lösung zu Aufgabe 25

(a) Die Sprache L_1 lässt sich schreiben als $L_1 = L_{11} \cup L_{12}$ mit

$$L_{11} = \{x \in \{a,b\}^* \mid (|x|_a + 2|x|_b) \bmod 3 = 2\}$$

und

$$L_{12} = \{x \in \{a,b\}^* \mid x \text{ beginnt und endet mit } aa\}\,.$$

Wir konstruieren zunächst zwei reguläre Grammatiken G_{11} und G_{12} für die Sprachen L_{11} und L_{12}.

Die Grammatik $G_{11} = (\{X_0, X_1, X_2\}, \{a,b\}, P_{11}, X_0)$ mit

$$P_{11} = \{X_0 \to aX_1 \mid bX_2,\ X_1 \to aX_2 \mid bX_0,\\ X_2 \to aX_0 \mid bX_1 \mid \lambda\}$$

generiert die Sprache L_{11}. Die Idee ist dabei, dass das Nichtterminal X_i genau dann erzeugt wird, wenn das damit erzeugte Anfangsstück x des Terminalworts die Bedingung $(|x|_a + 2|x|_b) \bmod 3 = i$ erfüllt. Die Regel $X_2 \to \lambda$ stellt sicher, dass die Ableitung genau dann enden kann, wenn das dadurch erzeugte Terminalwort in der Sprache L_{11} liegt.

Die Grammatik $G_{12} = (\{Y, Z\}, \{a,b\}, P_{12}, Y)$ mit

$$P_{12} = \{Y \to aa \mid aaa \mid aaZ,\ Z \to aZ \mid bZ \mid aa\}$$

generiert die Sprache L_{12}. Die ersten beiden Regeln generieren die beiden kurzen Wörter, in denen sich das aa am Anfang des Wortes und am Ende des Wortes überlappen. Mit den anderen Regeln können alle längeren Wörter aus L_{12} generiert werden: Die Regel $Y \to aaZ$ erzeugt das Präfix aa, mit den Regeln $Z \to aZ$ und $Z \to bZ$ kann ein beliebiger Mittelteil erzeugt werden und die Regel $Z \to aa$ generiert schliesslich das Suffix aa.

Eine Grammatik für L_1 erhalten wir nun, indem wir ein neues Startsymbol S hinzunehmen und die beiden neuen Regeln $S \to X_0$ und $S \to Y$ einführen. Damit kann im ersten Ableitungsschritt entschieden werden, ob ein Wort aus L_{11} oder aus L_{12}

erzeugt werden soll. Diese Konstruktion setzt voraus, dass die Nichtterminalmengen in G_{11} und G_{12} disjunkt sind, was hier offenbar gilt. Es ergibt sich insgesamt also die Grammatik $G_1 = (\{S, X_0, X_1, X_2, Y, Z\}, \{a, b\}, P_1, S)$ mit

$$\begin{aligned} P_1 = \{ & S \to X_0 \mid Y, \\ & X_0 \to aX_1 \mid bX_2,\ X_1 \to aX_2 \mid bX_0,\ X_2 \to aX_0 \mid bX_1 \mid \lambda, \\ & Y \to aa \mid aaa \mid aaZ,\ Z \to aZ \mid bZ \mid aa \}\,. \end{aligned}$$

(b) Die Grammatik G_2 erzeugt die Sprache

$$L_2 = \{0w0 \mid w \in \{0,1\}^*\} \cup \{1w1 \mid w \in \{1\}^*\}\,.$$

Dies lässt sich wie folgt begründen. Bei Anwendung der Regel $S \to 0X0$ werden zunächst zwei Nullen am Anfang und Ende des Wortes erzeugt. Danach lässt sich in einer ersten Phase der Ableitung zwischen den beiden Nullen mit den Regeln $X \to AX$ und $X \to BX$ ein beliebiges Nichtterminalwort aus den Buchstaben A und B erzeugen. Mit der Umwandlung des Nichtterminals X in das Nichtterminal Z mit Hilfe der Regel $X \to Z$ beginnt dann die zweite Phase der Ableitung. Hier werden in einem Durchlauf von rechts nach links mit den Regeln $AZ \to Z1$ und $BZ \to Z0$ alle As durch Einsen und alle Bs durch Nullen ersetzt. Die Regel $0Z \to 0$ stellt sicher, dass die Ableitung zu einem Terminalwort genau dann beendet werden kann, wenn keine As und Bs mehr übrig sind. Wenn eine Ableitung mit Anwendung der Regel $S \to 1Y1$ begonnen wird, werden damit zunächst zwei Einsen am Anfang und Ende des Wortes erzeugt. Die weitere Ableitung lässt sich auch dann wieder in zwei Phasen unterteilen. In der ersten Phase lässt sich mit der Regel $Y \to AY$ eine beliebige Anzahl von Nichtterminalen A erzeugen. Die erste Phase wird durch Anwendung der Regel $Y \to Z$ beendet. In der zweiten Phase können dann mit Hilfe der Regel $AZ \to Z1$ die As durch Einsen ersetzt werden, und die Regel $1Z \to 1$ stellt wiederum sicher, dass ein Terminalwort abgeleitet werden kann.

Die Sprache L_2 ist offensichtlich regulär und kann durch die reguläre Grammatik $G_2' = (\{S, X, Y\}, \{0, 1\}, P_2', S)$ erzeugt werden, wobei

$$P_2' = \{S \to 0X \mid 1Y,\ X \to 0X \mid 1X \mid 0,\ Y \to 1Y \mid 1\}\,.$$

In G_2' kann zunächst mit der Regel $S \to 0X$ die 0 am Anfang des Wortes erzeugt werden. Danach erzeugt die Anwendung der Regeln $X \to 0X$ und $X \to 1X$ ein beliebiges Wort über $\{0, 1\}$ und die letzte Regel $X \to 0$ erzeugt die Null am Ende des Wortes. Mit der Regel $S \to 1Y$ kann eine Eins am Anfang des Wortes erzeugt werden. Danach erzeugt die Anwendung der Regel $Y \to 1Y$ eine beliebige Anzahl von Einsen und die letzte Regel $Y \to 1$ erzeugt in diesem Fall die Eins am Ende des Wortes.

Lösung zu Aufgabe 26

Wir verwenden das Verfahren aus dem Abschnitt 3.2.1 des Buchs *Einführung in Automatentheorie, Formale Sprachen und Berechenbarkeit* von Hopcroft et al., um den Automaten in einen äquivalenten regulären Ausdruck umzuwandeln. Dieses Verfahren basiert auf dem Entwurfsprinzip der dynamischen Programmierung und berechnet für jedes Paar von Zuständen (i, j) reguläre Teilausdrücke $\alpha_{ij}^{(k)}$, die die Menge aller Wörter beschreiben, die

den Automaten vom Zustand i in den Zustand j führen und als Zwischenzustände nur die Zustände $1, \ldots, k$ verwenden.
Für die Initialisierung ergeben sich die folgenden Ausdrücke:

$$\alpha_{11}^{(0)} = \lambda$$
$$\alpha_{12}^{(0)} = a$$
$$\alpha_{13}^{(0)} = b$$
$$\alpha_{21}^{(0)} = a$$
$$\alpha_{22}^{(0)} = \lambda$$
$$\alpha_{23}^{(0)} = b$$
$$\alpha_{31}^{(0)} = b$$
$$\alpha_{32}^{(0)} = a$$
$$\alpha_{33}^{(0)} = \lambda$$

Mit Anwendung der Formel

$$\alpha_{ij}^{(k)} = \alpha_{ij}^{(k-1)} + \alpha_{ik}^{(k-1)} \cdot \left(\alpha_{kk}^{(k-1)}\right)^* \cdot \alpha_{kj}^{(k-1)}$$

für $k = 1$ können wir nun die Teilausdrücke für Wege durch den Automaten mit Zwischenzustand 1 herleiten. Dabei werden wir die hergeleiteten Ausdrücke soweit wie möglich vereinfachen. Es ergibt sich die folgende Tabelle:

	direktes Einsetzen in die Formel	vereinfachter Ausdruck
$\alpha_{11}^{(1)}$	$\lambda + \lambda(\lambda)^*\lambda$	λ
$\alpha_{12}^{(1)}$	$a + \lambda(\lambda)^*a$	a
$\alpha_{13}^{(1)}$	$b + \lambda(\lambda)^*b$	b
$\alpha_{21}^{(1)}$	$a + a(\lambda)^*\lambda$	a
$\alpha_{22}^{(1)}$	$\lambda + a(\lambda)^*a$	$\lambda + aa$
$\alpha_{23}^{(1)}$	$b + a(\lambda)^*b$	$b + ab = (\lambda + a)b$
$\alpha_{31}^{(1)}$	$b + b(\lambda)^*\lambda$	b
$\alpha_{32}^{(1)}$	$a + b(\lambda)^*a$	$a + ba = (\lambda + b)a$
$\alpha_{33}^{(1)}$	$\lambda + b(\lambda)^*b$	$\lambda + bb$

Diese Ausdrücke können wir nun wiederum verwenden, um die Ausdrücke für die Zwischenzustände 1 und 2 zu berechnen. Wir berechnen hier nur diejenigen Ausdrücke, die wir später noch brauchen werden. Es ergibt sich hier die folgende Tabelle:

	direktes Einsetzen in die Formel	vereinfachter Ausdruck
$\alpha_{13}^{(2)}$	$b + a(\lambda + aa)^*((\lambda + a)b)$	$b + aa^*b = a^*b$
$\alpha_{32}^{(2)}$	$(a + ba) + (a + ba)(\lambda + aa)^*(\lambda + aa)$	$(a + ba)(aa)^*$
$\alpha_{33}^{(2)}$	$(\lambda + bb) + ((\lambda + b)a)(\lambda + aa)^*((\lambda + a)b))$	$\lambda + (a + b)a^*b$

Dabei folgt die Vereinfachung von $\alpha_{33}^{(2)}$ aus der folgenden Rechnung:

$$\alpha_{33}^{(2)} = (\lambda + bb) + ((\lambda + b)a)(\lambda + aa)^*((\lambda + a)b))$$

$$
\begin{aligned}
&= \lambda + bb + (\lambda + b)aa^*b \\
&= \lambda + bb + aa^*b + baa^*b \\
&= \lambda + aa^*b + ba^*b \\
&= \lambda + (a + b)a^*b
\end{aligned}
$$

In der letzten Iteration, in der alle Zustände als Zwischenzustände zugelassen sind, müssen wir nicht mehr alle Ausdrücke berechnen. Der zu A äquivalente Ausdruck α ergibt sich als

$$\alpha = \alpha_{13}^{(3)},$$

da dies der einzige Weg ist, vom Anfangszustand in einen akzeptierenden Zustand zu gelangen. Mit Anwendung der Formel ergibt sich:

$$\alpha = \alpha_{13}^{(3)} = a^*b + a^*b(\lambda + (a + b)a^*b)^*(\lambda + (a + b)a^*b)$$

oder vereinfacht

$$\alpha = \alpha_{13}^{(3)} = a^*b((a + b)a^*b)^* .$$

Lösung zu Aufgabe 27

Die Grammatik $G = (\{S, A, B, X\}, \{0, 1, 2\}, P, S)$ mit

$$P = \{S \to ABS2 \mid X,\ ABX \to 0X1,\ AB0 \to 0AB,\ X \to \lambda\}$$

erzeugt die Sprache L. Sie arbeitet nach der folgenden Idee: Mit der Regel $S \to ABS2$ werden zunächst gleich viele As, Bs und Zweien erzeugt. Die As und Bs dienen hier als Platzhalter für Nullen und Einsen. Mit der Regel $ABX \to 0X1$ wird ein Paar AB in eine Null und eine Eins umgewandelt. Die Regel $AB0 \to 0AB$ dient nun dazu, die erzeugte Null über alle As und Bs hinweg nach links zu schieben. Erst wenn die erzeugte Null mindestens einmal nach links verschoben wurde, ist die Regel $ABX \to 0X1$ wieder anwendbar. Falls die Regel $X \to \lambda$ angewendet wird, obwohl noch As oder Bs vorhanden sind, dann lässt sich kein Terminalwort erzeugen, dies ist also keine gültige Ableitung. Mit $S \Rightarrow X \Rightarrow \lambda$ lässt sich auch das leere Wort ableiten. Eine Ableitung für das Wort 000111222 kann wie folgt aussehen:

$$
\begin{aligned}
S &\Rightarrow ABS2 \Rightarrow ABABS22 \Rightarrow ABABABS222 \Rightarrow ABABABX222 \\
&\Rightarrow ABAB0X1222 \Rightarrow AB0ABX1222 \Rightarrow 0ABABX1222 \Rightarrow 0AB0X11222 \\
&\Rightarrow 00ABX11222 \Rightarrow 000X111222 \Rightarrow 000111222 .
\end{aligned}
$$