Übungen zur Vorlesung Grundbegriffe der Theoretischen Informatik

technische universität dortmund

Thomas Schwentick

Gaetano Geck, Lutz Oettershagen,
Christopher Spinrath, Marco Wilhelm

Lehrstuhl 1
Logik in der Informatik

SoSe 2018 | **Übungsblatt 3** | **26.04.2018**

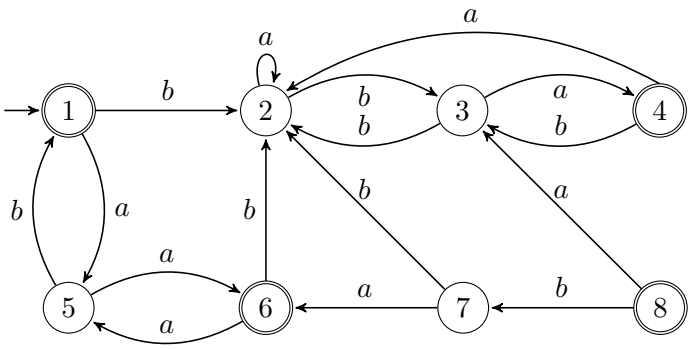
Abgabe bis spätestens am Donnerstag, 03.05.2018,

- (vor der Vorlesung) im HG II, HS 3, oder
- in die Briefkästen im Durchgangsflur, der die 1. Etage der OH 12 mit dem Erdgeschoss der OH 14 verbindet.

Beachten Sie die Schließzeiten der Gebäude!
Ansonsten gelten die Hinweise von Blatt 1.

Aufgabe 3.1 [Minimierung endlicher Automaten] — 4 Punkte

Bestimmen Sie zu nachfolgend angegebenem DFA einen äquivalenten minimalen DFA unter Verwendung des in der Vorlesung vorgestellten Algorithmus. Geben Sie dabei für jedes markierte Zustandspaar in einer Tabelle, wie in der Vorlesung, an, in welchem Durchlauf es markiert wurde. In einem Durchlauf sollen dabei genau die Zustandspaare markiert werden, die aufgrund von Markierungen im vorherigen Durchlauf (zum ersten Mal) markiert werden.
Zeichnen Sie anschließend den resultierenden Minimalautomaten.



Aufgabe 3.2 [Nerode-Äquivalenzklassen] — 4 Punkte

Sei $L = \{w \in \{a,b\}^* \mid |w| > 1 \text{ und der vorletzte Buchstabe in } w \text{ ist ein } b\}$.

a) Geben Sie für jede Äquivalenzklasse der Nerode-Relation $\sim_L$ einen Repräsentanten an. Geben Sie außerdem für je zwei verschiedene dieser Repräsentanten x_i und x_j ein Wort z_{ij} an, das bezeugt, dass x_i und x_j verschiedene Äquivalenzklassen repräsentieren. Es soll also gelten $x_i z_{ij} \in L \not\Leftrightarrow x_j z_{ij} \in L$ für alle Repräsentanten x_i, x_j mit $x_i \neq x_j$. **(2,5 Punkte)**

b) Geben Sie einen minimalen DFA $\mathcal{A}$ an, so dass $L(\mathcal{A}) = L$ gilt. Begründen Sie sowohl, dass $\mathcal{A}$ die Sprache L entscheidet, als auch, dass $\mathcal{A}$ minimal ist. **(1,5 Punkte)**

Hinweis

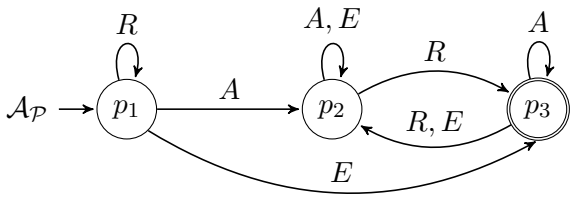
Die Nerode-Relation $\sim_L$ hat genau vier Äquivalenzklassen. Sie können diesen Fakt in dieser Aufgabe ohne Begründung benutzen.

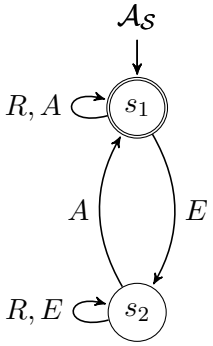
Aufgabe 3.3 [Verifikation mit endlichen Automaten] — 7 Punkte

In dieser Aufgabe wollen wir mit Hilfe der Automatentheorie verifizieren, ob ein gegebenes Programm $\mathcal{P}$ eine Spezifikation $\mathcal{S}$ erfüllt. Dazu nehmen wir stark vereinfachend an, dass ein Programm zu jedem Zeitpunkt genau eine der drei folgenden Aktionen ausführen kann:

- Es kann einen Rechenschritt durchführen;
- Es kann eine Ausgabe machen; oder
- Es kann eine Eingabe gelesen werden.

Ein *Programmablauf* ist eine Folge von Aktionen. Wir nehmen vereinfachend an, dass jedes Programm terminiert; jeder Programmablauf ist also eine *endliche* Folge. Ein Programm erfüllt eine Spezifikation genau dann, wenn jeder Programmablauf des Programms die Bedingungen der Spezifikation erfüllt.
Da Programmabläufe endliche Folgen sind, können wir sie durch Wörter über dem Alphabet $\{R, A, E\}$ modellieren, wobei R für einen Rechenschritt, A für eine Ausgabe und E für eine Eingabe stehen soll. Die Programmabläufe eines Programms können dann durch einen DFA modelliert werden: Der Automat akzeptiert genau die Wörter über $\{R, A, E\}$, die Programmabläufe des Programms darstellen. Ebenso lässt sich eine Spezifikation durch einen DFA modellieren: Es werden genau die Wörter akzeptiert, die Programmabläufe darstellen, die die Spezifikation erfüllen.
Der im Folgenden angegebene Automat $\mathcal{A}_\mathcal{P}$ entscheidet die Sprache aller Wörter, die Programmabläufe eines Beispielprogramms $\mathcal{P}$ darstellen. Weiterhin wird die Spezifikation $\mathcal{S}$, die fordert, dass nach jeder gelesenen Eingabe irgendwann eine Ausgabe erfolgen muss, gerade durch den Automaten $\mathcal{A}_\mathcal{S}$ modelliert.





a) Konstruieren Sie einen Produktautomaten $\mathcal{A}_\mathcal{S} \times \mathcal{A}_\mathcal{P}$. Wählen Sie dabei die Menge der akzeptierenden Zustände so, dass der Produktautomat für die Frage, ob das Programm $\mathcal{P}$ die Spezifikation $\mathcal{S}$ erfüllt, hilfreich ist. **(3 Punkte)**

b) Begründen sie mit Hilfe des in Aufgabenteil a) konstruierten Produktautomaten, ob das Programm $\mathcal{P}$ die Spezifikation $\mathcal{S}$ erfüllt. **(1 Punkt)**

c) Geben Sie einen Algorithmus in Pseudo-Code an, der als Eingabe zwei DFAs $\mathcal{A}_{\mathcal{P}'}$ und $\mathcal{A}_{\mathcal{S}'}$ über dem Alphabet $\{R, A, E\}$ erhält und entscheidet, ob das Programm $\mathcal{P}'$ die Spezifikation $\mathcal{S}'$ erfüllt. Bestimmen Sie außerdem die asymptotische Laufzeitschranke (in $\mathcal{O}$-Notation) Ihres Algorithmus.

Sie können die Algorithmen und Abschlusseigenschaften der Vorlesung als Bausteine in ihrem Algorithmus verwenden.

Hinweis

Zur Angabe der Algorithmen können Sie sich an den Algorithmen in Abschnitt 5.3 der Vorlesung orientieren.

(2 Punkte)

d) Wie kann Ihr Algorithmus abgewandelt werden, wenn die Spezifikation durch einen regulären Ausdruck statt einem DFA angegeben wird? Hat dies Auswirkungen auf die Laufzeit? Wenn ja, geben Sie eine neue Laufzeitschranke an. **(1 Punkt)**

Zusatzaufgabe [Kleine NFAs] **2 Punkte**

Wir wissen, dass es für jede reguläre Sprache einen minimalen DFA gibt, der diese Sprache entscheidet. Außerdem wurde in der Vorlesung ein Verfahren vorgestellt, solch einen minimalen DFA systematisch zu konstruieren. Aber gilt dies auch für NFAs?

a) Seien p, q zwei verschiedene Primzahlen mit $p, q < 1336 < pq$. Wir betrachten die Sprache $L = \{a^n \mid n \not\equiv_{pq} 1336\}$, also die Sprache aller Wörter über dem unären Alphabet $\{a\}$ deren Länge modulo pq *nicht* gleich 1336 ist. Gibt es einen NFA mit $p + q + 1$ Zuständen, der L entscheidet? **(2 Punkte)**

b) Sei $K = \{a^n \mid n \neq 1336\}$. Gibt es einen NFA mit weniger als 999 Zuständen, der K entscheidet?

Hinweis

Abgaben für Teilaufgabe b) richten Sie bitte per E-Mail an Christopher Spinrath. Die ersten drei korrekten und nachvollziehbaren Lösungen werden von Thomas Zeume mit einer Süßigkeit aus dem Lehrstuhlvorrat belohnt. Die Abgabefrist ist das Ende des Semesters.