Prof. Javier Esparza
Kevin Kappelmann, Simon Roßkopf, Lukas Stevens
Technische Universität München
Lehrstuhl für Theoretische Informatik

# Einführung in die Theoretische Informatik

Sommersemester 2023 – Hausaufgabenblatt 2

- Bitte beachten Sie, dass in dieser Vorlesung generell Antworten mit Begründung gefordert werden, solange die Aufgabe nicht explizit das Gegenteil sagt.
- Zum Bestehen diese Blattes müssen Sie 60% der Punkte erreichen.

**AT-Aufgabe H2.1.** (*Dem Tutor, dem die Studis vertrauen*) 0 Punkte

Bearbeiten Sie folgende Aufgaben in Automata Tutor (Aufgaben H2.1 a–e). Konstruieren Sie reguläre Ausdrücke für die folgenden Sprachen über dem Alphabet $\Sigma := \{a, b\}$. Beachten Sie dabei, dass die Bedingungen ab Aufgabenteil (b) sukzessiv kombiniert werden, die Sprache für (d) soll also beispielsweise die Bedingungen von (b) und (c) erfüllen.

(a) Geben Sie einen regulären Ausdruck für $\{(ab)^n a(ba)^n : n \in \mathbb{N}\}$ an.

(b) Die Sprache aller Wörter, die nicht mit einem $a$ beginnen.

(c) Zusätzlich müssen die Wörter mit einem $b$ enden.

(d) Zusätzlich müssen die Wörter eine ungerade Anzahl von $a$s haben.

(e) Zusätzlich dürfen die Wörter $bb$ nicht enthalten.

*Lösungsskizze.*

(a) $(abab)^*a$

(b) $\epsilon \mid b(a \mid b)^*$

(c) $b \mid b(a \mid b)^*b$

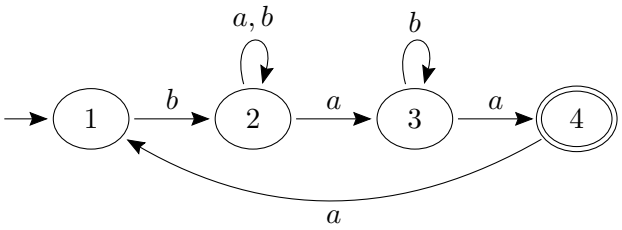
(d) $bb^*a(ab^*a \mid b)^*b$

(e) $ba((b \mid \epsilon)a(b \mid \epsilon)a)^*b$

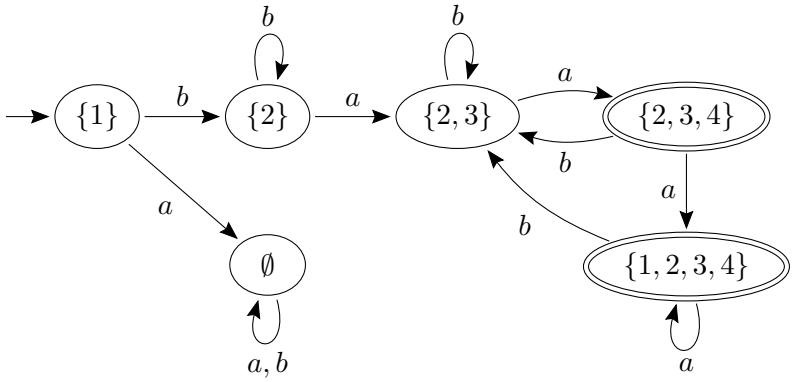
**Aufgabe H2.2.** (*Potenzmengenkonstruktion*) 3 Punkte

Konvertieren Sie den folgenden NFA über dem Alphabet $\Sigma = \{a, b\}$ zu einem DFA, indem Sie die Potenzmengenkonstruktion aus der Vorlesung verwenden.

**Hinweis:** Es genügt, nur die vom Startzustand erreichbaren Zustände zu konstruieren.




*Lösungsskizze.*




**Aufgabe H2.3.** (*Kein n-de in Sicht*) 3 + 3 Punkte

(a) Dr. Evilsparza hat sich durch eure letzten Bemühungen nicht einschüchtern lassen. Er glaubt immer noch fest daran, möglichst viel Unheil mit seinen $n$-DFAs (siehe H1.4) über die Studierendenschaft bringen zu können. Doch mutig wie ihr seid, gebt ihr natürlich nicht auf, sondern reitet auf zum zweiten Gefecht!

Weisen Sie Dr. Evilsparza in die Schranken, indem Sie zeigen, dass jede von einem $n$-DFA erkannte Sprache von einem DFA erkannt werden.

Geben Sie hierür eine formale Übersetzung von $n$-DFAs zu DFAs an. In anderen Worten: gegeben ein $n$-DFA $M = (Q, \Sigma, \delta, q_0, F)$, konstruieren Sie einen DFA $M' = (Q', \Sigma, \delta', q_0', F')$ mit $L(M) = L(M')$ indem Sie die Komponenten $Q', \delta', q_0', F'$ geeignet definieren (vgl. Ü2.5 b–c).

Die Korrektheit der Übersetzung müssen Sie nicht beweisen, allerdings informell erläutern.

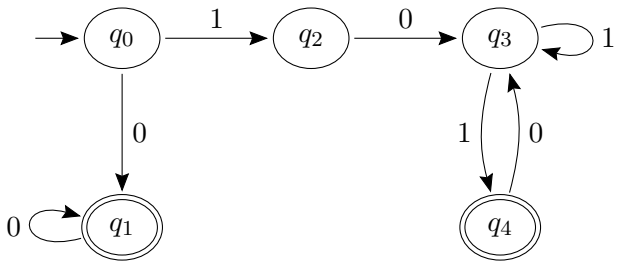
(b) Oh nein, der arme Dr. Evilsparza hat nun all seine Endzustände im Gefecht mit seinen $n$-DFAs verloren. Es bleibt ihm nur noch ein einziger Endzustand, den er in einem NFA einbauen möchte. Er nennt diese Konstruktion *SINGLE-NFAs*. In anderen Worten: ein SINGLE-NFA ist ein NFA, der genau einen Endzustand besitzt.

Heitern Sie Dr. Evilsparza auf, indem Sie zeigen, dass jede reguläre Sprache $L$ mit $\epsilon \notin L$ von einem SINGLE-NFA erkannt werden kann.

Geben Sie hierfür eine formale Übersetzung von NFAs zu SINGLE-NFAs an, wobei Sie annehmen dürfen, dass das leere Wort nicht in der Sprache des NFAs liegt. In anderen Worten: gegeben ein NFA $M = (Q, \Sigma, \delta, q_0, F)$ mit $\epsilon \notin L(M)$, konstruieren Sie einen SINGLE-NFA $M' = (Q', \Sigma, \delta', q_0', \{q_f'\})$ mit $L(M) = L(M')$ indem Sie die Komponenten $Q', \delta', q_0', q_f'$ geeignet definieren (vgl. Ü2.5 b–c).

Die Korrektheit der Übersetzung müssen Sie nicht beweisen, allerdings informell erläutern.

Wenden Sie anschließend Ihre Konstruktion auf den folgenden NFA an:



*Lösungsskizze.*

(a) Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein $n$-DFA. Wir konstruieren einen DFA $M'$ mit $L(M) = L(M')$. Die Konstruktionsidee liegt darin, $n+1$ Kopien von $M$ zu erzeugen und bei jedem Besuch eines Zustandes in $F$ in die nächste Kopie zu wechseln. Sobald wir in der $n+1$-ten Kopie angelangt sind, können wir akzeptieren. Formal definieren wir $M' \coloneqq (Q', \Sigma, \delta', q_0', F')$ mit

$$Q' \coloneqq \{q^i \mid q \in Q, 0 \leq i \leq n\}$$
$$F' \coloneqq \{q^n \mid q \in Q\}$$
$$\delta'(q^i, a) = \begin{cases} p^i, & \text{falls } i < n \text{ und } p = \delta(q,a) \notin F \\ p^{i+1}, & \text{falls } i < n \text{ und } p = \delta(q,a) \in F \\ p^n, & \text{sonst, wobei } p = \delta(q,a) \end{cases}$$
$$q_0' \coloneqq \begin{cases} q_0^0, & \text{falls } q_0 \notin F \\ q_0^1, & \text{sonst} \end{cases}$$

(b) Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein NFA. Wir konstruieren einen SINGLE-NFA $M'$ mit $L(M) = L(M')$. Die Konstruktionsidee liegt darin

(1) den Automaten $M$ zu kopieren,

(2) alle Endzustände zu normalen Zuständen zu machen,

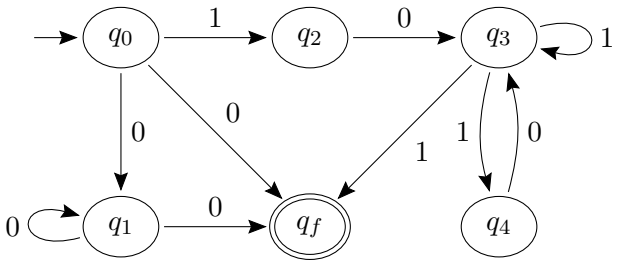
(3) einen neuen Endzustand einzuführen und

(4) Übergänge in alte Endzustände zusätzlich nichtdeterministisch in den neuen Endzustand zu überführen.

Formal definieren wir $M' \coloneqq (Q', \Sigma, \delta', q_0, \{q_f\})$ mit $q_f \notin Q$ und

$$Q' \coloneqq Q \cup \{q_f\}$$
$$\delta'(q,a) = \begin{cases} \delta(q,a), & \text{falls } \delta(q,a) \cap F = \emptyset \\ \delta(q,a) \cup \{q_f\}, & \text{sonst} \end{cases}$$

Anwendung auf den Beispielautomaten:




**Hausaufgabe H2.4.** (*Regex-Rekursion*) 3 + 3 Punkte

In den Übungsaufgaben haben wir eine Funktion definiert, die für einen regulären Ausdruck $r$ bestimmt, ob die Sprache $L(r)$ unendlich viele Wörter enthält. Nun wollen wir eine Funktion $\mathsf{contains}(\mathsf{a}, r)$ definieren, die für einen Buchstaben $\mathsf{a} \in \Sigma$ und regulären Ausdruck $r$ berechnet, ob $\mathsf{a}$ in jedem Wort aus $L(r)$ vorkommt. Für alle $w \in L(r)$ soll also gelten, dass $(\exists u, v \in \Sigma^*.\ w = u\mathsf{a}v) =: P(w)$.

(a) Geben Sie eine rekursive Funktion an, indem Sie das folgende Gerüst vervollständigen.

- $\mathsf{contains}(\mathsf{a}, \emptyset) = \ldots$
- $\mathsf{contains}(\mathsf{a}, \epsilon) = \ldots$
- $\mathsf{contains}(\mathsf{a}, \mathsf{x}) = \ldots$
- $\mathsf{contains}(\mathsf{a}, r^*) = \ldots$
- $\mathsf{contains}(\mathsf{a}, r_1|r_2) = \ldots$
- $\mathsf{contains}(\mathsf{a}, r_1 r_2) = \ldots$

(b) Beweisen Sie die Korrektheit ihrer Funktion mit struktureller Induktion. Kennzeichnen Sie dabei die Induktionshypothesen und deren Anwendungen deutlich.

*Lösungsskizze.*

(a)
- $\mathsf{contains}(\mathsf{a}, \emptyset) = \mathsf{true}$
- $\mathsf{contains}(\mathsf{a}, \epsilon) = \mathsf{false}$
- $\mathsf{contains}(\mathsf{a}, \mathsf{x}) = (\mathsf{a} = \mathsf{x})$
- $\mathsf{contains}(\mathsf{a}, r_1 \,|\, r_2) = \mathsf{contains}(\mathsf{a}, r_1) \wedge \mathsf{contains}(\mathsf{a}, r_2)$
- $\mathsf{contains}(\mathsf{a}, r_1 r_2) = \mathsf{contains}(\mathsf{a}, r_1) \vee \mathsf{contains}(\mathsf{a}, r_2)$
- $\mathsf{contains}(\mathsf{a}, r^*) = \mathsf{false}$

(b) **Fall** $r = \emptyset$
Es gilt $\mathsf{contains}(\mathsf{a}, \emptyset) = \mathsf{true} \iff \forall w \in L(\emptyset).\ P(w)$, da $L(\emptyset) = \emptyset$.

**Fall** $r = \epsilon$
Wir haben $L(r) = \{\varepsilon\}$ und $\mathsf{contains}(a, \epsilon) = \mathsf{false} \iff P(\varepsilon) \iff \forall w \in L(\epsilon).\ P(w)$.

**Fall** $r = \mathsf{x}$
Wenn $\mathsf{a} = \mathsf{x}$, dann gilt $L(\mathsf{x}) = \{\mathsf{a}\}$ und wir haben

$$\mathsf{contains}(\mathsf{a}, \mathsf{x}) = \mathsf{true} \iff P(\mathsf{a}) \iff \forall w \in L(\mathsf{x}).\ P(w).$$

Ansonsten haben wir

$$\mathsf{contains}(\mathsf{a}, \mathsf{x}) = \mathsf{false} \iff P(\mathsf{x}) \iff \forall w \in L(\mathsf{x}).\ P(w),$$

da $L(\mathsf{x}) = \{\mathsf{x}\}$.

**Fall** $r = s^*$
Dann gilt $\varepsilon \in L(s^*)$ und da $P(\varepsilon) = \mathsf{false}$, ist die Definition $\mathsf{contains}(a, r^*) = \mathsf{false}$ korrekt.

**Fall** $r = r_1 \,|\, r_2$
Als Induktionshypothesen erhalten wir $\mathsf{contains}(\mathsf{a}, r_1) \iff \forall w \in L(r_1).\ P(w)$ und $\mathsf{contains}(\mathsf{a}, r_2) \iff \forall w \in L(r_2).\ P(w)$. Damit haben wir

$$\begin{aligned}
\mathsf{contains}(\mathsf{a}, r_1 \,|\, r_2) &\iff \mathsf{contains}(\mathsf{a}, r_1) \wedge \mathsf{contains}(\mathsf{a}, r_2) \\
&\overset{IH}{\iff} (\forall w \in L(r_1).\ P(w)) \wedge (\forall w \in L(r_2).\ P(w)) \\
&\iff \forall w \in L(r_1) \cup L(r_2).\ P(w) \\
&\iff \forall w \in L(r_1 \,|\, r_2).\ P(w).
\end{aligned}$$

**Fall** $r = r_1 r_2$
Als Induktionshypothesen erhalten wir $\mathsf{contains}(\mathsf{a}, r_1) \iff \forall w \in L(r_1).\ P(w)$ und

$\mathsf{contains}(\mathsf{a}, r_2) \iff \forall w \in L(r_2).\ P(w)$. Wenn $L(r_1) = \emptyset$ oder $L(r_2) = \emptyset$, dann gilt $L(r_1 r_2) = \emptyset$ und die Korrektheit folgt analog zum Fall $r = \emptyset$. Ansonsten gilt

$$
\begin{aligned}
\mathsf{contains}(\mathsf{a}, r_1 r_2) &\iff \mathsf{contains}(\mathsf{a}, r_1) \vee \mathsf{contains}(\mathsf{a}, r_2) \\
&\overset{IH}{\iff} (\forall w \in L(r_1).\ \exists u, v.\ w = u\mathsf{a}v) \vee (\forall w \in L(r_2).\ \exists u, v.\ w = u\mathsf{a}v) \\
&\iff (\forall w \in L(r_1 r_2).\ \exists y \in L(r_2).\ \exists u, v.\ w = u\mathsf{a}vy) \vee \\
&\qquad (\forall w \in L(r_1 r_2).\ \exists y \in L(r_1).\ \exists u, v.\ w = yu\mathsf{a}v) \\
&\iff (\forall w \in L(r_1 r_2).\ \exists u, v.\ w = u\mathsf{a}v) \\
&\qquad \vee (\forall w \in L(r_1 r_2).\ \exists u, v.\ w = u\mathsf{a}v) \\
&\iff \forall w \in L(r_1 r_2).\ \exists u, v.\ w = u\mathsf{a}v
\end{aligned}
$$

Dabei gilt die dritte Äquivalenz aufgrund der Annahmen $L(r_1) \neq \emptyset$ und $L(r_2) \neq \emptyset$.