Prof. Dr. Jasmin Blanchette
Elisabeth Lempa
Luca Maio

Ludwig-Maximilians-Universität München
Institut für Informatik
Besprechung 08.05.2025 bis 12.05.2025
Abgabe bis 19.05.2025, 10:00 Uhr

Lösungsvorschlag zur Übung 2 zur Vorlesung

Formale Sprachen und Komplexität

Wenn Sie Automaten angeben, tun Sie dies immer in Form eines Zustandsgraphen. Andere Formen der Darstellung (z.B. als Liste von Übergängen) werden nicht gewertet, da sie sehr viel aufwändiger zu korrigieren sind. Vergessen Sie nicht, im Zustandsgraphen Start- und Endzustände zu markieren.

FSK2-1 *Grammatiken und Elimination von ε-Produktionen*

a) Geben Sie kontextfreie Grammatiken (ggf. mit ε-Produktionen) an, die folgende Sprachen über dem Alphabet $\Sigma = \{a, b\}$ erkennen:

i) $L_1 = \{caw \mid c \in \Sigma, w \in \Sigma^*\}$

LÖSUNGSVORSCHLAG:
$G_1 = (\{S, C, W\}, \Sigma, P_1, S)$ mit

$$P_1 = \{S \to CaW,\ C \to a,\ C \to b,\ W \to \varepsilon,\ W \to aW,\ W \to bW\}$$

ii) $L_2 = \{aawbb \mid w \in \Sigma^*\}$

LÖSUNGSVORSCHLAG:
$G_2 = (\{S, W\}, \Sigma, \{S \to aaWbb,\ W \to \varepsilon,\ W \to aW,\ W \to bW\}, S)$

b) Betrachten Sie die Grammatik $G = (\{A, B, C\}, \{a, b\}, P, A)$ mit Produktionen

$$P = \{A \to aBbC,\ A \to AaA,\ B \to \varepsilon,\ B \to CC,\ C \to BB\}$$

Geben Sie eine zu G äquivalente Grammatik G' ohne ε-Produktionen an. Verwenden Sie den Algorithmus zur Elimination von ε-Produktionen aus der Vorlesung und geben Sie die Zwischenschritte Ihrer Berechnung an. (Das ermöglicht uns, Ihnen bei kleinen Fehlern noch Teilpunkte zu geben.)

LÖSUNGSVORSCHLAG:

- Finde die Menge W der Variablen D mit $D \Rightarrow^* \varepsilon$, wobei D nicht das Startsymbol ist.

- Initial $W := \{B\}$, da $B \to \varepsilon$ und $B \neq A$.
- Wegen $C \to BB$ und $B \in W$ muss $C \in W$ sein, also $W := \{B, C\}$.
- Wegen $B \to CC$ und $C \in W$ muss $B \in W$ sein, aber das war es auch vorher schon.
- Es gibt keine weiteren Produktionen der Form $D \to E_1 \dots E_n$ mit $E_1, \dots, E_n \in W$, also ist $W = \{B, C\}$.

- Entferne Produktionen der Form $D \to \varepsilon$, wobei D nicht das Startsymbol ist:

$$P' := \{A \to aBbC,\ A \to AaA,\ B \to CC,\ C \to BB\}$$

- Für Produktionen der Form $D \to uEv \in P'$ mit $|uv| > 0$ und $E \in W$, füge $D \to uv$ zu P' hinzu.
  - Wegen $B \to CC \in P'$: $P' := P' \cup \{B \to C\}$.
  - Wegen $C \to BB \in P'$: $P' := P' \cup \{C \to B\}$.
  - Wegen $A \to aBbC \in P'$: $P' := P' \cup \{A \to abC\}$.
  - Wegen $A \to aBbC \in P'$: $P' := P' \cup \{A \to aBb\}$.
  - Wegen $A \to aBb \in P'$: $P' := P' \cup \{A \to ab\}$.
  - Wegen $A \to abC \in P'$: $P' := P' \cup \{A \to ab\}$, aber diese Produktion war bereits enthalten.
  - Es gibt keine weiteren Produktionen der gewünschten Form in P', also ist

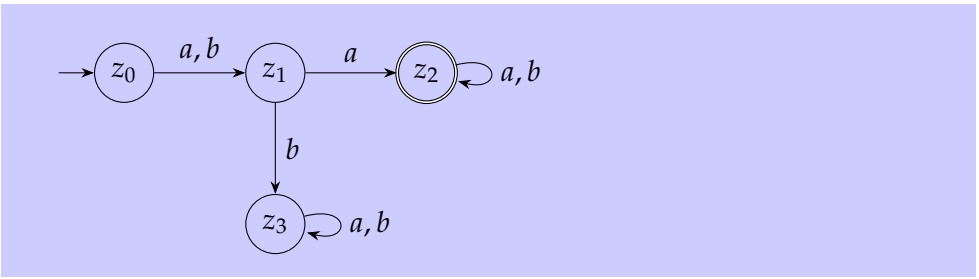
$$P' = \{A \to aBbC,\ A \to AaA,\ B \to CC,\ C \to BB,\ B \to C,\ C \to B,\ A \to abC,\ A \to aBb,\ A \to ab\}$$

FSK2-2 ***DFAs und Minimierung***

a) Geben Sie DFAs an, die folgende Sprachen über dem Alphabet $\Sigma = \{a, b\}$ erkennen:

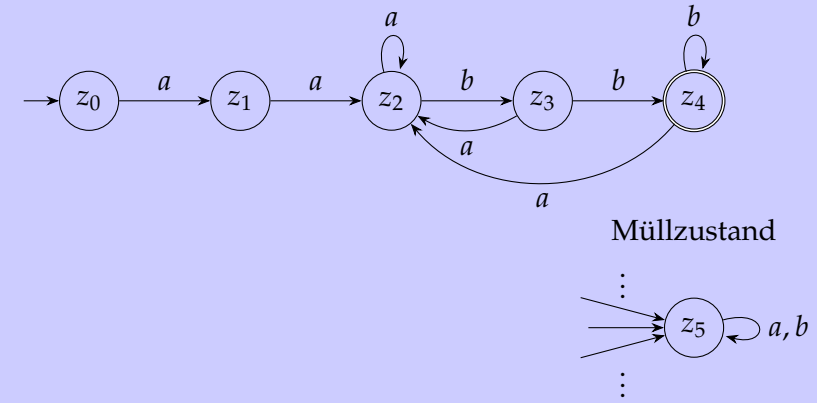
i) $L_1 = \{caw \mid c \in \Sigma, w \in \Sigma^*\}$

LÖSUNGSVORSCHLAG:



ii) $L_2 = \{aawbb \mid w \in \Sigma^*\}$

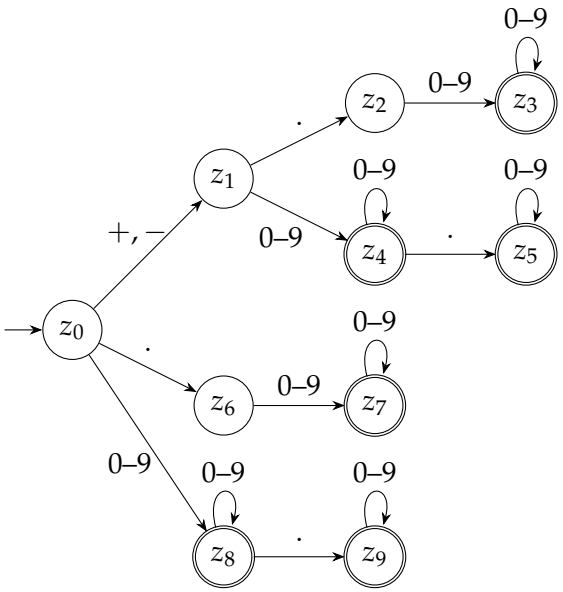
LÖSUNGSVORSCHLAG:

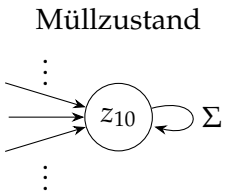


(Das sind die gleichen Sprachen wie in Aufgabe FSK2-1.)

b) Minimieren Sie die folgenden DFAs. Verwenden Sie die tabellarische Variante des Algorithmus zur Minimierung von DFAs aus der Vorlesung (nicht die graphische Variante und nicht den Algorithmus aus vorherigen Jahren). Geben Sie die Partitionstabelle und den minimalen DFA an.

i) DFA A_2 über dem Alphabet $\Sigma = \{+, -, ., 0, \ldots, 9\}$ (bekannt aus der Vorlesung):



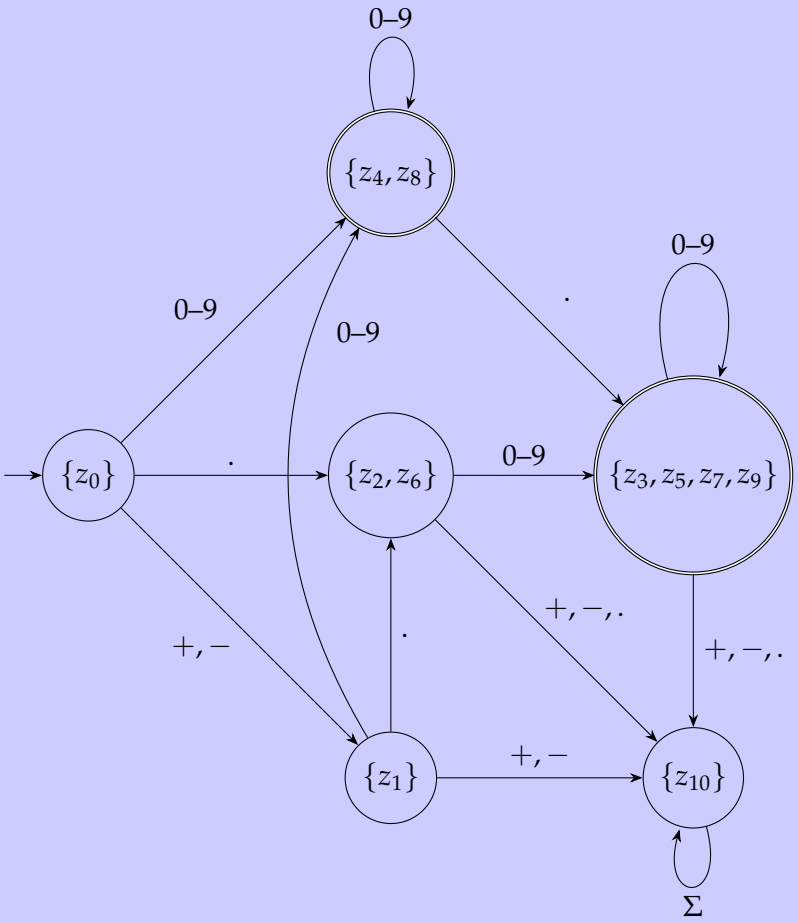


LÖSUNGSVORSCHLAG:
Partitionstabelle:

z_0	z_1	z_2	z_6	z_{10}	z_3	z_4	z_5	z_7	z_8	z_9	
z_0	z_1	z_2	z_6	z_{10}	z_3	z_4	z_5	z_7	z_8	z_9	mit 0
z_0	z_1	z_2	z_6	z_{10}	z_3	z_4	z_5	z_7	z_8	z_9	mit +
z_0	z_1	z_2	z_6	z_{10}	z_3	z_5	z_7	z_9	z_4	z_8	mit .

Sie dürfen (und sollten vernünftigerweise) Mengen von Symbolen, die immer zusammen vorkommen, für diesen Algorithmus als ein Symbol betrachten. Beispielsweise müssen Sie bei dem gegebenen Automaten nicht separat prüfen, ob 0, 1, 2, …, 9 zu einer Partitionierung führen; es genügt, ein Symbol aus dieser Menge zu prüfen.

Aus der Partitionierung ergibt sich der Minimalautomat:



(In diesem Fall ist der ursprünliche Automat womöglich besser lesbar und verständlich als der Minimalautomat. Für viele Algorithmen und Beweise sind Minimalautomaten aber trotzdem wichtig.)

FSK2-3 ***Kleine Automaten***

a) Sei A_1 ein DFA mit Alphabet Σ und genau einem Zustand. Zeigen oder widerlegen Sie: Es ist entweder $L(A_1) = \Sigma^*$ oder $L(A_1) = \emptyset$.

LÖSUNGSVORSCHLAG:
Stimmt. Der Automat hat genau einen Zustand z_0 (der also auch Startzustand ist) und muss Übergänge $z_0 \xrightarrow{a} z_0$ für jedes $a \in \Sigma$ haben. Wenn z_0 ein Endzustand ist, akzeptiert A_1 somit jedes Wort in Σ^*. Wenn z_0 kein Endzustand ist, akzeptiert A_1 kein Wort.

b) Zeigen Sie: Für jeden DFA A mit Alphabet $\Sigma = \{a, b\}$ und genau vier Zuständen gilt: Wenn für jede natürliche Zahl $n \geq 1$ das Wort $a^{n^2} \in L(A)$ ist, dann ist auch $a^{12} \in L(A)$.

LÖSUNGSVORSCHLAG:
In A muss es mindestens eine Schleife aus a-Übergängen geben, die einen Endzustand enthält, da a-Sequenzen mit Länge > 4 erkannt werden, aber A nur 4 Zustände hat.
Sei z_0 der Startzustand von A. Sei $z_w = \widetilde{\delta}(z_0, a^{12})$ der Zustand, in dem wir uns nach Lesen von a^{12} befinden. Dieser Zustand z_w muss Teil einer Schleife sein, da A nur 4 Zustände hat. Sei $s \in \{1, 2, 3, 4\}$ die Länge dieser Schleife. Es gilt somit für jedes $k \in \mathbb{N}$: $\widetilde{\delta}(z_w, a^{ks}) = z_w$.
Wir zeigen, dass z_w ein Endzustand ist. Fallunterscheidung über s:

- Für $s = 1$ gilt $\widetilde{\delta}(z_0, a^{4^2}) = \widetilde{\delta}(z_0, a^{12+4}) = \widetilde{\delta}(z_w, a^{4 \cdot 1}) = z_w$
- Für $s = 2$ gilt $\widetilde{\delta}(z_0, a^{4^2}) = \widetilde{\delta}(z_0, a^{12+4}) = \widetilde{\delta}(z_w, a^{2 \cdot 2}) = z_w$
- Für $s = 3$ gilt $\widetilde{\delta}(z_0, a^{6^2}) = \widetilde{\delta}(z_0, a^{12+24}) = \widetilde{\delta}(z_w, a^{8 \cdot 3}) = z_w$
- Für $s = 4$ gilt $\widetilde{\delta}(z_0, a^{4^2}) = \widetilde{\delta}(z_0, a^{12+4}) = \widetilde{\delta}(z_w, a^{1 \cdot 4}) = z_w$

In allen Fällen gibt es also ein $n \in \mathbb{N}_{>0}$ mit $\widetilde{\delta}(z_0, a^{n^2}) = z_w$, sodass z_w ein Endzustand sein muss. Damit ist $a^{12} \in L(A)$. (Übrigens ist 12 die kleinste Nicht-Quadratzahl mit dieser Eigenschaft.)

FSK2-4 ***Grammatik-Konkatenation***

Seien G_1 und G_2 Typ i-Grammatiken (für $i \in \{0, \ldots, 3\}$), sodass $\varepsilon \notin L(G_1)$ und $\varepsilon \notin L(G_2)$.

Zeigen oder widerlegen Sie für alle i: Es gibt eine Grammatik H vom Typ i, sodass $L(H) = L(G_1)L(G_2)$. (Die Sprache von H ist also die Konkatenation der Sprachen von G_1 und G_2.)

LÖSUNGSVORSCHLAG:
Ja, dies gilt für alle i. Sei $G_1 = (V_1, \Sigma_1, P_1, S_1)$ und $G_2 = (V_2, \Sigma_2, P_2, S_2)$.
Es liegt nahe, die Alphabete, Variablen und Produktionsregeln von G_1 und G_2 zu vereinigen, und ein neues Startsymbol S_H einzuführen. Mit einer zusätzlichen Regel: $S_H \to S_1 S_2$ können dann Satzformen hergeleitet werden, von der aus man mit den Regeln in P_1 aus S_1 genau die Wörter aus $L(G_1)$ herleiten kann, und mit den Regeln in P_2 aus S_2 genau die Wörter aus $L(G_2)$.
Für Grammatiken der Typen 0, 1 und 2 könnte man also $H = (V_H, \Sigma_H, P_H, S_H)$

wie folgt konstruieren:

- Sei S_H eine neue Variable (d.h. $S_H \notin V_1 \cup V_2$)
- $V_H = V_1 \cup V_2 \cup \{S_H\}$
- $\Sigma_H = \Sigma_1 \cup \Sigma_2$
- $P_H = P_1 \cup P_2 \cup \{S_H \to S_1 S_2\}$

Es gibt allerdings ein Problem, falls die Mengen der Variablen und Alphabete nicht disjunkt sind. Um das zu beheben, verwenden wir folgende Konstruktion:

- Führe für jedes $A \in V_1 \cap (V_2 \cup \Sigma_2)$ eine neue Variable A' ein und ersetze A in G_1 durch A' (sowohl in V_1 als auch in P_1 und eventuell in S_1).
- Führe analog für jedes $A \in V_2 \cap \Sigma_1$ eine neue Variable A' ein und ersetze A in G_2 durch A'.
- Führe für jedes $a \in \Sigma_1 \cap \Sigma_2$, welches auf der linken Seite einer Produktionsregel von P_1 vorkommt, eine Variable V_a ein, ersetze jedes a in P_1 durch V_a und füge die Produktionsregel $V_a \to a$ zu P_1 hinzu.
- Führe analog für jedes $a \in \Sigma_1 \cap \Sigma_2$, welches auf der linken Seite einer Produktionsregel von P_2 vorkommt, eine Variable V'_a ein.

Bei Typ-3-Grammatiken stoßen wir noch auf ein weiteres Problem: Sie erlauben keine Regel der Form $S_H \to S_1 S_2$. Wir können aber die Konstruktion wie folgt abändern:

- $S_H = S_1$
- $V_H = V_1 \cup V_2$
- $\Sigma_H = \Sigma_1 \cup \Sigma_2$
- $P_H = P_n \cup P_c \cup P_2$
  wobei $P_n = \{A \to aB \mid (A \to aB) \in P_1\}$
  und $P_c = \{A \to aS' \mid (A \to a) \in P_1\}$

Produktionsregeln aus P_1 der Form $A \to a$ (also solche die in G_1 dazu dienen den letzten Buchstaben eines Wortes herzuleiten) werden umgeschrieben zu $A \to aS_2$. Dadurch kann nach dem letzten Buchstaben der von den Regeln aus G_1 erzeugt wird mit dem Startsymbol von G_2 und den entsprechenden Regeln weitergemacht werden. (Alternativ hätten wir bei Typ-3-Sprachen auch einfach über die Abschlusseigenschaften argumentieren können.)