Theoretische Informatik 1
Übungsblatt 5

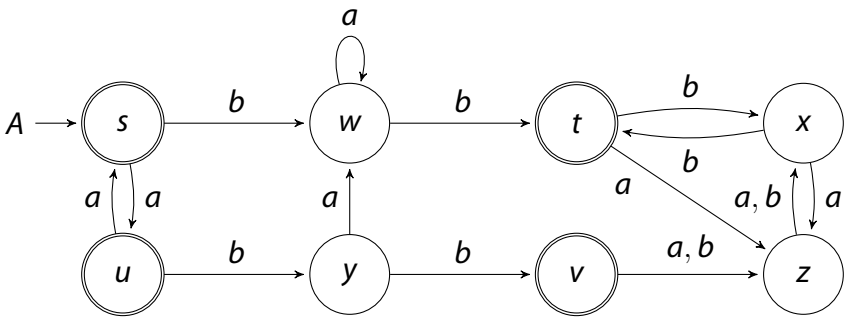
René Maseli
Thomas Haas
TU Braunschweig
Wintersemester 2023/24

Ausgabe: 2024-01-08 Abgabe: 2024-01-18 23:59

Geben Sie Ihre Lösungen bis Donnerstag, 18.01.2024 23:59 Uhr, im Vips-Verzeichnis der StudIP-Veranstaltung ab. Fertigen Sie dazu ihre Hausaufgaben direkt in .pdf Form an oder scannen ihre handschriftlichen Hausaufgaben ein. Geben Sie in Gruppen von 4 Personen ab und geben Sie **alle** Gruppenmitglieder mit **Matrikelnummer, Namen und Studiengang** an.

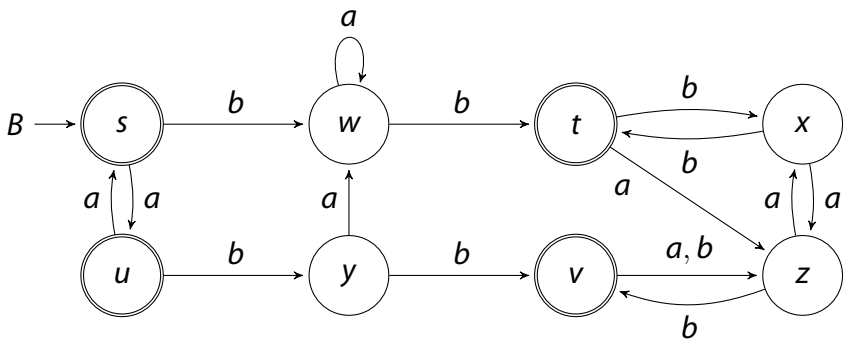
Hausaufgabe 1: Table-Filling-Algprithmus [12 Punkte]

Betrachten Sie den folgenden DFA A.



a) [5 Punkte] Zeigen Sie, dass A minimal ist, indem Sie den Table-Filling-Algorithmus anwenden. Füllen Sie Zellen mit 0, wenn das jeweilige Zustandspaar initial zu trennen ist, und ansonsten mit der Nummer der Iteration, in welcher das Paar erstmals getrennt wird.
Hinweis: Notieren Sie, während Sie Ihre Tabelle füllen, in welcher Reihenfolge Sie Zustände unterscheiden, z.B. werden zu Beginn finale Zustände abgetrennt: $\{s,t,u,v\} \not\sim_A \{w,x,y,z\}$, und in Iteration 1 können deshalb $\{s,u\} \not\sim_A \{t,v\}$ getrennt werden, usw.

Betrachten Sie nun den DFA B, welcher sich von A nur in einer Transition unterscheidet.



b) [5 Punkte] Nutzen Sie den Table-Filling-Algorithmus, um den minimalen DFA $B_{\min}$ mit $\mathcal{L}(B_{\min}) = \mathcal{L}(B)$ zu finden. Zeichnen Sie den Zustandsgraphen von $B_{\min}$.

c) [2 Punkte] Geben Sie alle Äquivalenzklassen der Nerode-Rechtskongruenz von $\mathcal{L}(B)$ mit jeweils mindestens einem Repräsentanten an.

Hausaufgabe 2: Pumping-Lemma für reguläre Sprachen [9 Punkte]

Sei $\Sigma = \{a, b\}$ ein Alphabet. Für jedes Wort w beschreibe $|w|_a$ die Anzahl der Vorkommen von a in w. $|w|_b$ sei analog definiert.

Beweisen Sie unter Verwendung des Pumping-Lemmas, dass die folgenden Sprachen nicht regulär sind.

a) [2 Punkte] $L_1 = \{ w \in \{a, b\}^* \mid |w|_b + 7 > |w|_a \}$

b) [3 Punkte] $L_2 = \{ xb^m y \in \{a, b\}^* \mid \exists n \in \mathbb{N} : |y| = n \text{ und } x \in (a^* b)^n \text{ und } m \geq 2 \}$

c) [2 Punkte] $L_3 = \{ a^n b^m \mid n < 42 \text{ oder } m < n \}$

d) [2 Punkte] $L_4 = \{ w \in \{a, b\}^* \mid |w|_a \neq |w|_b \}$

Hinweis zu d): Überlegen Sie sich folgendes: für eine gegebene Zahl $n \in \mathbb{N}$, welche Zahl ist durch jede Zahl $\leq n$ teilbar?

Hausaufgabe 3: Ersetzungssysteme [9 Punkte]

Sei $\Sigma = \{a, b\}$ ein Alphabet. Geben Sie kontextfreie Grammatiken G_1, G_2, G_3 und G_4 an, welche die folgenden Sprachen ableiten:

a) [1 Punkt] $L_1 = \{ a^n b^m w \mid w \in \Sigma^* \text{ und } m > 2 \text{ und } |w|_a = n \}$.

b) [1 Punkt] $L_2 = \{ w \in \Sigma^* \mid |w|_a < |w|_b \}$.

c) [1 Punkt] $L_3 = \{ w \in \Sigma^* \mid \forall u, v\colon\ w = u.v \Rightarrow |v|_a \leq |v|_b \}$.

d) [2 Punkte] $L_4 = \{ b^m M^m \mid m \in \mathbb{N} \}$, wobei $M = \{ a^n b^n \mid n \in \mathbb{N} \}$ bekanntermaßen kontextfrei ist (siehe Bsp. 8.18 im Skript). Z.B. $bbabaabb \in L_4$.

Ein Kontrollpfad eines While-Programms kann als ein Weg im Kontrollfluss-Graphen verstanden werden, d.h. eine Block-Folge, in der alle aufeinanderfolgenden Paare durch eine Fluss-Kante verbunden sind. In der Programmanalyse bezeichnet man sie üblicherweise als Pfade (Wege ohne Kreise), weil man z.B. auch deren Index in der Folge beachtet.

Betrachten Sie die folgenden Programme P_5 und P_6 jeweils mit Blöcken $B = \{0,1,2,3,4,5,6,7\}$.

P_5:

```
[x := 0]^0
while [x^2 < y]^1 do
    [z := x + 1]^2
    [x := z^2 + z]^3
end while
if [x^2 = y]^4 then
    [z := 1]^5
else
    [z := y]^6
end if
[x := z]^7
```

P_6:

```
[x := 0]^0
while [x < 2^4]^1 do
    [y := 3x + 2]^2
    while [y < 5x]^3 do
        [y := y + 2]^4
        if [3x < y]^5 then
            [x := x + 1]^6
        end if
    end while
end while
[x := x − 14]^7
```

e) [2 Punkte] Geben Sie eine **rechtslineare** Grammatik G_5 über dem Alphabet $\Sigma = B$ an, welche genau die Kontrollpfade von P_5 produziert, die in Block 0 beginnen und in Block 7 enden. Z.B. ist $01457 \in \mathcal{L}(G_a)$ und $01231467 \in \mathcal{L}(G_5)$, sind gültig, aber z.B. $01234567 \notin \mathcal{L}(G_5)$ darf nicht akzeptiert werden.

f) [2 Punkte] Konstruieren Sie eine rechtslineare Grammatik G_6 über dem Alphabet $\Sigma = B$, welche genau die Kontrollpfade von P_6 produziert, die in Block 0 beginnen und in Block 7 enden. Z.B. ist $017 \in \mathcal{L}(G_6)$ das kleinste Wort der Sprache. Ein weiteres Element ist z.B. $0123456317 \in \mathcal{L}(G_6)$, aber z.B. $01234567 \notin \mathcal{L}(G_6)$ darf nicht ableitbar sein.

Hausaufgabe 4: Lineare Grammatiken [8 Punkte]

Beweisen Sie, dass die regulären Sprachen genau die Sprachen sind, die als $\mathcal{L}(G)$ für eine rechtlineare Grammatik G auftreten.

a) [4 Punkte] Erklären Sie, wie man zu einem gegebenen NFA A eine rechtslineare Grammatik G mit $\mathcal{L}(G) = \mathcal{L}(A)$ konstruieren kann.

b) [4 Punkte] Erklären Sie, wie man zu einer gegebenen rechtslineare Grammatik G einen NFA A mit $\mathcal{L}(G) = \mathcal{L}(A)$ konstruieren kann.

Bemerkung: Ein analoges Resultat gilt auch für linkslineare Grammatiken, deshalb spricht man in beiden Fällen von **regulären** Grammatiken.

Übungsaufgabe 5:

Wir betrachten Programme nur mit boolschen Variablen. Hier werden Ausdrücke $e \in \text{Exp}$ nichtdeterministisch auf Variabel-Zuständen $\sigma \in \{0,1\}^V$ ausgewertet: Für $v \in \{0,1\}$ soll $S_v(e)$ die Menge der Variablen-Zustände sein, auf denen e den Wert v zurückgeben kann.

Für Variable $x \in V$ ist $S_v(x) = \{ \sigma \in \{0,1\}^V \mid \sigma(x) = v \}$. Beispielsweise gilt auch $\sigma \in S_v(y \text{ oder nicht } z)$ genau dann, wenn $v = \max(\sigma(y), 1 - \sigma(z)) \in \{0,1\}$ ist. Außerdem gibt es den sogenannten *havoc*-Ausdruck $*$, welcher nicht-deterministisch sowohl 0 als auch 1 zurückgeben kann: $S_0(*) = S_1(*) = \{0,1\}^V$.

Sei V die Menge der Variabeln und B die Menge der Blöcke des boolschen Programms P. Einige Wörter aus $\{s\} \cup (V \times \{0,1\})$ entsprechen (endlichen) Ausführungen von P. Es gibt einen NFA $\langle (B \times \{0,1\}^V), \langle b_0, 0^V \rangle, \to, \{f\} \times \{0,1\}^V \rangle$, dessen Sprache exakt diese Wörter enthält. Dieser startet beim initialen Block $i \in B$ mit allen Variablen auf Wert 0 und akzeptiert auf dem (einzigen) finalen Block $f \in B$, ungeachtet der Werte.

Die Transition $\langle b, \sigma \rangle \xrightarrow{\langle x,v \rangle} \langle c, \tau \rangle$ existiert genau dann, wenn für jedes $y \in V \setminus \{x\}$ die Gleichheit $\sigma(y) = \tau(y)$ gilt, $\tau(x) = v$ ist, $b = [x := e]^\ell$ eine Zuweisung ist, c der (nach Programmrichtung eindeutige) Nachfolger von b ist und wenn $\sigma \in S_v(e)$.

Die Transition $\langle b, \sigma \rangle \xrightarrow{s} \langle c, \tau \rangle$ existiert genau dann, wenn $\sigma = \tau$ gilt, $b = [e]^\ell$ die Bedingung einer Verzweigung oder Schleife ist, und entweder

- c der erste else-Block oder falls dieser nicht existiert, der nächste Block ist und $\sigma \in S_0(e)$ ist.
- c der erste then-Block oder der erste innere Schleifenblock ist und $\sigma \in S_1(e)$ ist.

a) Betrachten Sie das folgende boolsche Programm P mit den Blöcken $B = \{b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$ und den Variablen $V = \{x, y, z\}$.

$[x := *]^0$
while $[$ nicht x oder nicht $z]^1$ **do**
  $[y :=$ nicht x und nicht $z]^2$
  $[x := *]^3$
  if $[y]^4$ **then**
    $[x :=$ nicht $x]^5$
  end if
  $[z := x]^6$
end while
$[\text{skip}]^7$

Konstruieren Sie einen endlichen Automaten A_P, welcher Wörter aus $\{s\} \cup V \times \{0, 1\}$ akzeptiert: $\Sigma = \{s, x0, x1, y0, y1, z0, z1\}$.

$\varepsilon \notin \mathcal{L}(A_P)$, da jede Ausführung mindestens durch Block b_0 führen muss.

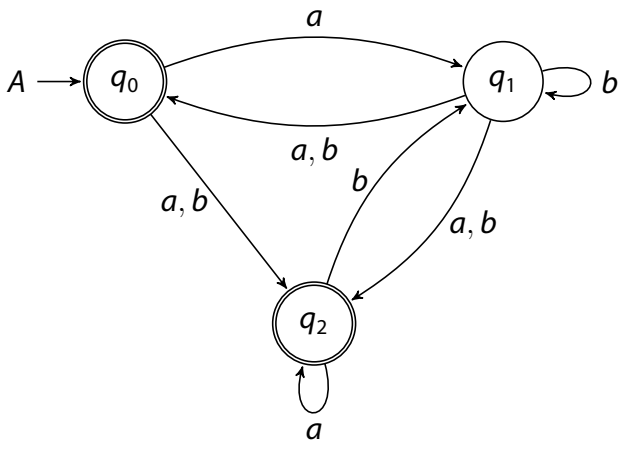
$x1.s \notin \mathcal{L}(A_P)$, da jede Ausführung mit $z = 0$ startet und deswegen mindestens einmal die Schleife durchlaufen muss.

$x1.s.y0.x1.s.z1.s \in \mathcal{L}(A_P)$, da die Ausführungen zuerst 1 und dann 0 lesen können, wodurch die Austrittsbedingung erfüllt wird.

b) Ist Ihr Automat A_P *partiell* deterministisch (indem alle fehlenden Kanten zu einem neuen Zustand $\varnothing$ führen)?

Übungsaufgabe 6:

Betrachten Sie den folgenden NFA A über $\{a, b\}$.



a) Konstruieren Sie einen zu A sprachäquivalenten DFA $\mathcal{P}(A)$ unter Verwendung der Rabin-Scott-Potenzmengenkonstruktion. Stellen Sie sicher, dass $\mathcal{P}(A)$ keine unerreichbaren Zustände enthält.

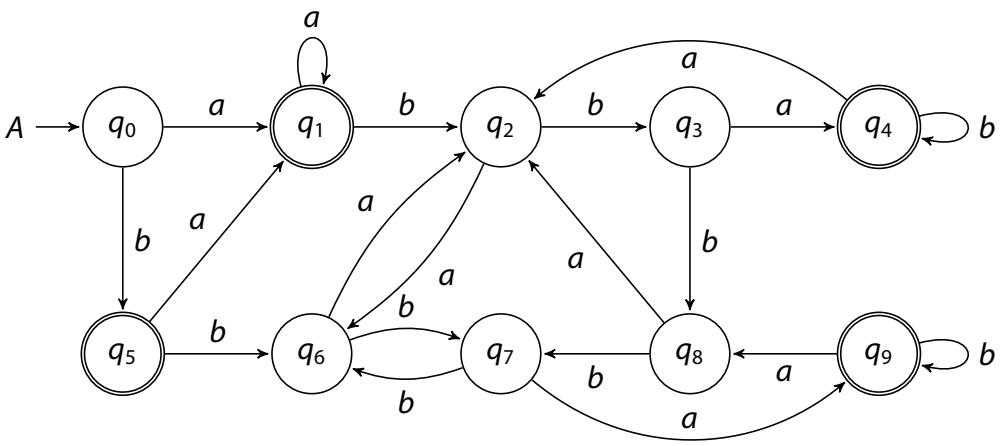
b) Bestimmen Sie alle ~-Äquivalenzklassen auf den Zuständen von $\mathcal{P}(A)$ unter Verwendung des Table-Fillings-Algorithmus aus der Vorlesung. Geben Sie an, in welcher Reihenfolge Sie die Zellen in der Tabelle markiert haben.

c) Geben Sie den minimalen DFA B für $\mathcal{L}(A)$ an. Verwenden Sie hierzu die ~-Äquivalenzklassen.

d) Bestimmen Sie alle Äquivalenzklassen der Nerode-Rechtskongruenz $\equiv_{\mathcal{L}(A)}$.

Übungsaufgabe 7:

Betrachten Sie den folgenden NFA A über $\{a, b\}$.



a) Bestimmen Sie auf den Zuständen von A alle ~-Äquivalenzklassen unter Verwendung des Table-Fillings-Algorithmus aus der Vorlesung. Geben Sie an, in welcher Reihenfolge Sie die Zellen in der Tabelle markiert haben.

b) Geben Sie den minimalen DFA B für $\mathcal{L}(A)$ an. Verwenden Sie hierzu die ~-Äquivalenzklassen.

c) Bestimmen Sie alle Äquivalenzklassen der Nerode-Rechtskongruenz $\equiv_{\mathcal{L}(A)}$. Stellen Sie $\mathcal{L}(A)$ als Vereinigung einer bestimmten Teilmenge dieser Äquivalenzklassen dar.