Prof. Javier Esparza
Philipp Czerner, Martin Helfrich

Technische Universität München
Lehrstuhl für Theoretische Informatik

Einführung in die Theoretische Informatik

Sommersemester 2021 – Hausaufgabenblatt 1

- Beachten Sie die Abgabemodalitäten auf der Vorlesungswebsite!
- Sei $\Phi := \{\{1\}, \{2,3\}, \{4,5\}, \{6\}\}$. Nach dem Abgabedatum werden wir für jede Menge $A \in \Phi$ eine zufällige Aufgabe $a \in A$ wählen und korrigieren, d.h. Aufgaben H1.1 und H1.6 werden sicher korrigiert, während zwischen Aufgaben H1.2 und H1.3 eine Münze entscheidet.
- Es werden diese Aufgaben korrigiert: H1.1, H1.2, H1.4, H1.6
- Wenn Sie einen Beweis aufstellen, von dem Sie wissen, das einzelne Schritte problematisch oder unvollständig sind, merken Sie dies bitte in Ihrer Lösung an, damit wir das bei der Korrektur positiv berücksichtigen können.
- Es gilt $0 \in \mathbb{N}$.

Aufgabe H1.1. (*DFA/NFA Konstruktion*) 1+1+1 Punkte

Diese Hausaufgabe wird mit Automata Tutor bearbeitet und abgegeben. Falls Sie es noch nicht bereits gemacht haben, folgen Sie den Schritten in Ü1.2, um ein Konto zu erstellen. Achten Sie darauf, dass Sie sich, wie dort beschrieben, mit Ihrer TUM-Kennung anmelden. Ansonsten können wir Ihnen die Punkte nicht gutschreiben.

Bearbeiten Sie die Hausaufgaben H1.1 (a–c). **Achtung:** Während Sie für die Aufgaben aus dem Übungsblatt beliebig viele Versuche hatten, haben Sie für jede Hausaufgabe nur 5 Versuche. Sie bekommen nur dann einen Punkt, wenn Sie die Aufgabe nach 5 Versuchen vollständig (also mit 10/10 Punkten) gelöst haben.

Aufgabe H1.2. (*Sprachen*) 1+1+1+1 Punkte

Sei $\Sigma := \{a, b\}$, $A := \{a, ba, aab\}$, $B := \{\varepsilon, ab\}$. Bestimmen Sie folgende Mengen:

(a) $AB \setminus A$ (b) $A^3\emptyset$ (c) AB^0 (d) $\emptyset^* \times B^2$

Lösungsskizze.

(a) $\{baab, aabab\}$
(b) $\emptyset$
(c) A
(d) $\{(\varepsilon, \varepsilon), (\varepsilon, ab), (\varepsilon, abab)\}$

Aufgabe H1.3. (*Lug und Trug*) 1+1+1+1 Punkte

Sei $\Sigma \neq \emptyset$ ein Alphabet, und $A, B, C \subseteq \Sigma^*$ Sprachen über Σ. Folgende Aussagen sind falsch; geben Sie jeweils ein Gegenbeispiel an.

(a) $A^+ \setminus A = A^2A^*$
(b) $A \subseteq B \Leftrightarrow A^* \subseteq B^*$
(c) $|AB| = |BA|$
(d) $AB \setminus AC = A(B \setminus C)$

Lösungsskizze. In allen Gegenbeispielen verwenden wir $\Sigma := \{a, b\}$.

(a) $A := \{\varepsilon\}$ oder auch $A := \{a, aa\}$.
(b) $A := \{aa\}, B := \{a\}$, womit $A^* \subseteq B^*$ gilt, $A \subseteq B$ aber nicht.
(c) $A := \{a, aa\}, B := \{b, ab\}$, d.h. $AB = \{ab, aab, aaab\}$ und $BA = \{ba, baa, aba, abaa\}$.

(d) $A := \{a, aa\}, B := \{ab\}, C := \{b\}$, also gelten $AB \setminus AC = \{aaab\}$ und $A(B \setminus C) = \{aab, aaab\}$.

Aufgabe H1.4. (*Teilbarkeit*) 2+3 Punkte

Der kleine Theodor lernt gerade in der Schule, wie mal Zahlen malnimmt und teilt. Damit er mehr Zeit zum Malen hat, will er seine Hausaufgaben vollautomatisch lösen.

Sei $\Sigma := \{\texttt{0}, ..., \texttt{9}\}$. Für ein Wort $w_1 w_2 ... w_n = w \in \Sigma^*$ bezeichnen wir mit $(w)_{10} := \sum_{i=1}^{n} w_i \cdot 10^{n-i}$ die Zahl, die man erhält, wenn man w zur Basis 10 interpretiert (d.h. $(x)_{10} \in \mathbb{N}$ und es gilt z.B. $(\texttt{42})_{10} = 42$ oder $(\varepsilon)_{10} = 0$).

Sei $k \in \mathbb{N}, k \geq 2$. Wir definieren die Sprache der durch k teilbaren Dezimalzahlen als $L_k := \{w \in \Sigma^* : (w)_{10} = i \cdot k,\ i \in \mathbb{N}\}$.

(a) Geben Sie einen deterministischen endlichen Automaten (DFA) an, der L_3 akzeptiert, und begründen Sie, dass ihr Automat korrekt ist, also genau die Wörter in L_3 akzeptiert.

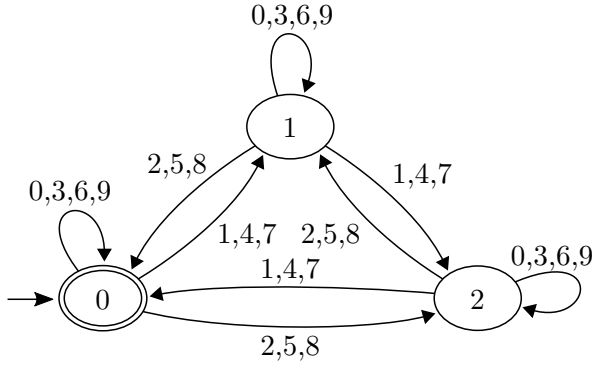
(b) Geben Sie einen DFA an, der L_k akzeptiert, und beweisen Sie seine Korrektheit. *Anmerkung:* k ist ein Parameter, Sie müssen also die Zustände und die Übergangsfunktion in Abhängigkeit von k angeben.

Hinweis: Es genügt, Teil (b) sinnvoll zu bearbeiten. Die Punkte für (a) erhalten Sie dann automatisch.

Lösungsskizze. **(a)** Es ist bekannt, dass eine Zahl genau dann durch 3 teilbar ist, wenn ihre Quersumme durch 3 teilbar ist. Kurze Begründung, da Grundschulen diese Aussage häufig ohne Beweis angeben: Sei $w_1 w_2 ... w_n = w \in \Sigma^*$ ein Wort. Dann gilt

$$(w)_{10} \bmod 3 = \sum_{i=1}^{n} w_i \cdot 10^{n-i} \bmod 3 = \sum_{i=1}^{n} w_i \cdot (10 \bmod 3)^{n-i} \bmod 3 = \sum_{i=1}^{n} w_i \bmod 3$$

Um festzustellen, ob die Quersumme durch 3 teilbar ist, genügt es uns, sich die Quersumme modulo 3 zu merken. Wir verwenden also Zustände $Q = \{0, 1, 2\}$, Startzustand $q_0 = 0$, akzeptierenden Zustände $\{0\}$, und Übergänge δ wie folgt.



Insbesondere gilt $\delta(q, c) = (q + c) \bmod 3$ für alle Zustände q und Zeichen $c \in \{\texttt{0}, ..., \texttt{9}\}$.
(b) Einfach die Quersumme zu betrachten genügt hier nicht, aber wir können ein ähnliches Verfahren verwenden. Sei $M_k := (Q, \Sigma, \delta, q_0, F)$ ein DFA, mit Zuständen $Q := \{0, ..., k-1\}$, Startzustand $q_0 := 0$, akzeptierenden Zuständen $F = \{0\}$, und Übergängen $\delta(q, c) := (10q + c) \bmod k$ für jedes $q \in Q, c \in \Sigma$.

Um zu beweisen, dass $L(M_k) = L_k$ gilt, zeigen wir die Aussage $\hat{\delta}(q_0, w) = (w)_{10} \bmod k$ mit vollständiger Induktion über die Wortlänge. Die Aussage, die wir beweisen wollen,

ist also: für jedes $n \in \mathbb{N}$ gilt, dass alle Wörter $w \in \Sigma^n$ mit Länge $|w| = n$ die Gleichung $\hat{\delta}(q_0, w) = (w)_{10} \bmod k$ erfüllen.
Induktionsbasis. Für $n = 0$ kann w nur das leere Wort sein, und es gilt

$$\hat{\delta}(q_0, \varepsilon) = q_0 = 0 = (\varepsilon)_{10} \bmod k$$

Induktionsschritt. Angenommen, die Aussage gilt für ein bestimmtes $n \in \mathbb{N}$, es gilt also $\hat{\delta}(q_0, w) = (w)_{10} \bmod k$ für alle Wörter w mit Länge n. Wir müssen nun zeigen, dass diese Gleichung auch für alle Wörter der Länge $n+1$ gilt. Sei also $w \in \Sigma^{n+1}$ ein Wort der Länge $n+1$. Wir können w schreiben als $w = uc$, wobei $u \in \Sigma^n$ ein Wort der Länge n ist und $c \in \Sigma$ ein Zeichen. Es gilt also

$$\begin{aligned}\hat{\delta}(q_0, w) = \hat{\delta}(q_0, uc) &\overset{(1)}{=} \delta(\hat{\delta}(q_0, u), c) \overset{\text{(IA)}}{=} \delta((u)_{10} \bmod k, c) \\ &\overset{(2)}{=} (10(u)_{10} + c) \bmod k \overset{(3)}{=} (uc)_{10} \bmod k = (w)_{10} \bmod k\end{aligned}$$

In Schritt (1) haben wir die Definition von $\hat{\delta}$ verwendet, für (2) unsere Konstruktion von δ, und (3) folgt aus der Definition von $(w)_{10}$.

Aufgabe H1.5. (*Buchstabenfhler*) 2+2+1 Punkte

Der Superschurke Dr. Evilsparza hat sich ins Campusnetzwerk gehackt und es ist ihm gelungen, aus mehreren Programmen Fragmente zu löschen! Sie sind Teil des Cybersecurityteams, das versucht das Ausmaß des Schadens zu bestimmen und die beschädigten Programme zu reparieren.

Sei $\Sigma := \{a, b\}$ ein Alphabet und $w \in \Sigma^*$ ein Wort. Mit w^R bezeichnen wir die Umdrehung von w, also $(w_1 w_2 ... w_n)^R := w_n w_{n-1} ... w_1$ (hier sind $w_1, w_2, ..., w_n \in \Sigma$) die Buchstaben von w. Beispielsweise gilt also $(abbaaa)^R = aaabba$.

(a) Sei $L := \{w \in \Sigma^* : w = w^R\}$ die Menge der Palindrome über Σ. Wir definieren folgende kontextfreie Grammatik $G = (\{S\}, \Sigma, P, S)$ mit Produktionen $P := \{S \to aSa \mid bSb \mid \varepsilon\}$.

Gilt $L(G) = L$? Wenn ja, beweisen Sie es, wenn nein, geben Sie ein Gegenbeispiel an und „reparieren“ Sie G, konstruieren Sie also eine Grammatik G' mit $L(G') = L$, indem Sie neue Produktionen zu G hinzufügen.

(b) Sei G eine beliebige Grammatik, und G' eine Grammatik, die durch das Entfernen von einer beliebigen Produktion aus G entsteht. Beweisen Sie $L(G') \subseteq L(G)$, oder widerlegen Sie es durch Angabe eines Gegenbeispiels.

(c) Sei G wieder eine beliebige Grammatik, und G' eine Grammatik, die man erhält, wenn man von G ein beliebiges Nichtterminalzeichen aus der rechten Seite einer beliebigen Produktion löscht. Beweisen oder widerlegen Sie: $L(G') \subseteq L(G)$

Lösungsskizze.

(a) $L(G) = L$ gilt *nicht*, da $a \in L$, aber nicht über G produziert werden kann, weil jedes Wort $w \in (\{S\} \cup \Sigma)^*$ das aus S abgeleitet werden kann, d.h. $S \to_G^* w$, eine gerade Anzahl an a's hat. Wir reparieren G, indem wir $S \to a \mid b$ hinzufügen.

(b) Die Aussage ist wahr: Sei $w \in L(G')$ beliebig, und S das Startsymbol von G. Nach Definition von $L(G')$ gilt also $S \to_{G'}^* w$. Da wir für G' aber nur eine Produktion entfernt haben, gilt $S \to_G^* w$ ebenfalls, und somit $w \in L(G)$.

(c) Falsch! Ein Gegenbeispiel wäre $G = (\{S\}, \Sigma, P, S)$ mit $P = \{S \to aS\}$. Es ist nicht möglich, in G ein Wort produzieren, das nur aus Terminalzeichen besteht, also $L(G) = \emptyset$. Wenn wir allerdings die Produktionsregel zu $S \to a$ verändern, erhalten wir $L(G') = \{a\}$.

Bonusaufgabe H1.6. (*Thue-Morse-Sequenz*) 1+4 Bonuspunkte

Die Thue-Morse-Sequenz über dem Alphabet $\Sigma := \{0, 1\}$ kann wie folgt definiert werden. Wir setzen $w_0 := 0$, und für $n \in \mathbb{N}$ definieren wir $w_{n+1} := w_n \overline{w_n}$, wobei $\overline{w}$ das Einerkomplement des Binärwortes $w \in \Sigma^*$ bezeichnet, z.B. $\overline{100111} = 011000$. Die ersten Folgenglieder sind also $w_0 = 0$, $w_1 = 01$, $w_2 = 0110$, und $w_3 = 01101001$.

(a) Überprüfen Sie $(w_5)_2 \bmod 7 = 4$, wobei $(w)_2$ – analog zu Aufgabe H1.4 – den Wert von w zur Basis 2 bezeichnet. Geben Sie Ihren vollständigen Rechenweg an.

(b) Bestimmen Sie $(w_{2021})_2 \bmod 19$. Beschreiben Sie ihr Vorgehen.

Hinweis: Für Teilaufgabe (b) dürfen sie einen Computer zu Hilfe nehmen. Falls Sie dies tun, beschreiben Sie bitte Ihren Ansatz in *natürlicher Sprache*. Sie können (müssen aber nicht), ihrer Lösung Programmcode beifügen, jedoch steht es den Korrektoren frei, diesen zu ignorieren. Zur Kontrolle: $(w_{42})_2 \bmod 19 = 2, (w_{43})_2 \bmod 19 = 10$.

Lösungsskizze. **(a)** Wir verwenden quasi den DFA $M_7 = (Q, \Sigma, \delta, q_0, F)$ aus Aufgabe H1.4, gehen aber von Basis 10 zu Basis 2. Es gilt also $Q = \{0, ..., 6\}$, $q_0 = 0$, und $\delta(q, c) = (2q + c) \bmod 7$ für alle Zustände $q \in Q$ und Zeichen $c \in \Sigma$. Analog zum dortigen Beweis, gilt für jedes $q \in Q, w \in \Sigma^*$, dass $\hat{\delta}(q_0, w) = (w)_2 \bmod 7$.

Wir können also insbesondere den Rest $(w_5)_2 \bmod 7$ ablesen, indem wir den Zustand betrachten, den der Automat nach Einlesen von w_6 erreicht:

Wort w_5	0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1
Zustand	0 0 1 3 6 6 5 3 0 1 2 4 2 4 2 5 3 0 0 0 1 2 5 4 1 2 5 4 1 3 6 5 4

(b) Hier verwenden wir wieder einen DFA $M_{19} = (Q, \Sigma, \delta, q_0, F)$ aus Aufgabe H1.4. Das Wort w_{2021} hat allerdings Länge 2^{2021}, eine Zahl, die zu groß ist, um ihre Binärdarstellung in diesem Universum zu speichern. Wir müssen uns also ein geschicktes Verfahren überlegen, um M_{19} auf w_{2021} auszuführen, ohne jedes Zeichen einzeln durchzugehen.

Hierzu merken wir uns für jedes n nicht nur, in welchen Zustand M_{19} landet, wenn wir in Zustand 0 starten und w_n einlesen, sondern wir merken uns auch die Endzustände für jeden anderen Startzustand. In anderen Worten sind wir an der Funktion $f_n : Q \to Q$ interessiert, die $f_n(q) = \hat{\delta}(q, w_n)$ erfüllt. Analog dazu definieren wir uns g_n für $\overline{w_n}$, also $g_n(q) = \hat{\delta}(q, \overline{w_n})$.

Nach Definition von w_{n+1} gilt $w_{n+1} = w_n \overline{w_n}$, wir können also (nach Definition von $\hat{\delta}$) folgendes schreiben:

$$f_{n+1}(q) = \hat{\delta}(q, w_{n+1}) = \hat{\delta}(q, w_n \overline{w_n}) = \hat{\delta}(\hat{\delta}(q, w_n), \overline{w_n}) = \hat{\delta}(f_n(q), \overline{w_n}) = g_n(f_n(q))$$

Somit gilt $f_{n+1} = g_n \circ f_n$. Mit dem gleichen Argument zeigen wir auch $g_{n+1} = f_n \circ g_n$ (beachten Sie hier, dass $\overline{w_{n+1}} = \overline{w_n} w_n$). Die Startwerte f_0, g_0 ergeben sich direkt aus δ, z.B. $f_0(q) = \delta(q, 0)$. Es genügt nun, $f_{2021}(0)$ auszurechnen, etwa mit folgenden Python-Code:

```
f = [(2*i)  %19 for i in range(19)]
g = [(2*i+1)%19 for i in range(19)]
```

```
for it in range(2021):
    f_new = [g[f[i]] for i in range(19)]
    g_new = [f[g[i]] for i in range(19)]
    f, g = f_new, g_new
print(f[0])
```

Das Ergebnis ist $(w_{2021})_2 \bmod 19 = 15$.
Alternativlösung: Es gibt auch eine Lösung, die Automaten nicht verwendet: Vorab betrachten wir zunächst die Folge $a_0 := 2$, $a_{n+1} := a_n^2 \bmod 19$. Offensichtlich gilt $a_n = 2^{2^n} \bmod 19$; die ersten Folgenglieder sind

n	0	1	2	3	4	5	6	7
a_n	2	4	16	9	5	6	17	4

Insbesondere gilt $a_1 = a_7$, damit ist die Folge periodisch.

Wir wollen nun die rekursive Definition $w_{n+1} = w_n\overline{w_n}$, die auf der Binärdarstellung operiert, über arithmetische Operationen darstellen. Setzen wir also $x_n := (w_n)_2$. Um dem Binärwort w_n genau $l := |w_n| = 2^n$ Nullen anzuhängen, müssen wir den Wert x_n mit 2^l multiplizieren. Außerdem können wir das Einerkomplement von w_n bilden, indem wir $(2^l - 1) - x_n$ rechnen. (Binär dargestellt besteht $2^l - 1$ aus l Einsen.) Insgesamt können wir also x_{n+1} folgendermaßen ausrechnen:

$$x_{n+1} = 2^{2^n} x_n + (2^{2^n} - 1) - x_n = (2^{2^n} - 1)(x_n + 1)$$

Schließlich können wir $y_n := x_n \bmod 19$ definieren, und obige Gleichung ergibt

$$y_{n+1} = (2^{2^n} - 1)(x_n + 1) \bmod 19 = (a_n - 1)(y_n + 1) \bmod 19$$

Der gesuchte Wert y_{2021} lässt sich leicht berechnen:

```
def a(n):
    if n == 0: return 2
    return [4, 16, 9, 5, 6, 17][(n-1)%6]
y = 0
for i in range(2021):
    y = (a(i)-1) * (y+1) % 19
print(y)
```

Anmerkung: Da a_n 6-periodisch ist, und y_n nur 19 mögliche Werte hat, ist y_n auch periodisch, mit Periode höchstens $6 \cdot 19 = 114$. Tatsächlich ist y_n 108-periodisch und somit gilt $y_{2021} = y_{77}$.