PD Dr. Jan Johannsen
Elisabeth Lempa
Luca Maio

Ludwig-Maximilians-Universität München
Institut für Informatik
Besprechung 09.07.2026 bis 13.07.2026
Abgabe bis 21.07.2026, 14:00 Uhr

Lösungsvorschlag zur Übung 11 zur Vorlesung

# Formale Sprachen und Komplexität

**FSK11-1** ***Satz von Rice***

Sei $M = (Z, \{a, b\}, \Gamma, \delta, z_0, \square, E)$ eine deterministische Turingmaschine. Welche der folgenden Fragestellungen zu $M$ sind entscheidbar?

Zeigen Sie die Unentscheidbarkeit unentscheidbarer Fragestellungen mittels des Satzes von Rice. Bei entscheidbaren Fragestellungen erläutern Sie, wie die charakteristische Funktion berechnet wird.

**LÖSUNGSVORSCHLAG:**
Wiederholung:

**Satz von Rice**: Sei $\mathcal{R}$ die Klasse aller turingberechenbaren Funktionen. Sei $\mathcal{S}$ eine beliebige Teilmenge, sodass $\emptyset \subset \mathcal{S} \subset \mathcal{R}$. Dann ist die Sprache

$$C(\mathcal{S}) = \{w_M \mid \text{die von } M \text{ berechnete Funktion liegt in } \mathcal{S}\}$$

unentscheidbar.

a) Die Funktion die von $M$ berechnet wird hat mindestens 3 Fixpunkte. Ein Fixpunkt einer Funktion $f : D \to D$ ist ein $x \in D$, sodass $f(x) = x$.

**LÖSUNGSVORSCHLAG:** Das Problem ist unentscheidbar:

Sei

$$\mathcal{S} = \{f \in \mathcal{R} \mid \exists x, y, z.\ x \neq y \wedge x \neq z \wedge y \neq z \wedge f(x) = x \wedge f(y) = y \wedge f(z) = z\},$$

also die Menge aller berechenbarer Funktionen die mindestens drei Fixpunkte haben.

Dann liegt die Funktion $f(x) = x$ in $\mathcal{S}$, da sie unendlich viele Fixpunkte hat, aber die Funktion $f(x) = x^2$ nicht, da sie nur zwei Fixpunkte hat (0 und 1).

Damit ist $\mathcal{S}$ nicht leer und auch nicht die Menge aller berechenbaren Funktionen.

Mit dem Satz von Rice ist

$$C(S) = \{w_M \mid \text{Die von } M \text{ berechnete Funktion hat mindestens drei Fixpunkte}\}$$

unentscheidbar.

b) Totalitätsproblem: Es gibt ein Wort $w \notin L(M)$.

**LÖSUNGSVORSCHLAG:** Das Problem ist unentscheidbar:

Sei $\mathcal{S}$ die Menge aller Funktionen, die an mindestens einer Stelle nicht definiert sind.

Dann liegt die Funktion $\Omega$ die überall undefiniert ist in $\mathcal{S}$, aber die Funktion $f(x) = x$ nicht.

$\mathcal{S}$ ist damit nicht leer und auch nicht die Menge aller berechenbaren Funktionen.

Dann ist $C(\mathcal{S}) = \{w_M \mid M \text{ akzeptiert mindestens für eine Eingabe nicht}\}$.

Mit dem Satz von Rice ist $C(\mathcal{S})$ unentscheidbar und damit ist das Totalitätsproblem (also ob eine Turingmaschine mindestens ein Wort nicht akzeptiert) unentscheidbar.

c) Hat M einen Müllzustand? Formal definieren wir einen Müllzustand hier als einen Zustand $z$, der kein Endzustand ist und für den gilt $\delta(z, a) \subseteq \{z\}$ für alle $a$.

**LÖSUNGSVORSCHLAG:** Das Problem ist entscheidbar. Prüfe für alle Zustände, die keine Endzustände sind, die Übergänge für alle (endlich viele) Eingabesymbole. Wenn keiner der Übergänge zu einem anderen Zustand führt beende und gib Wahr zurück, sonst gib nach Prüfen aller Nicht-Endzustände Falsch zurück.

d) $M$ terminiert auf jeder Eingabe nach zwischen 50 und 55 Schritten.

**LÖSUNGSVORSCHLAG:** Das Problem ist entscheidbar. Simuliere $M$ für bis zu 55 Schritten auf allen Eingaben mit einer Länge kleiner oder gleich 55 und falls $M$ vor 50 Schritten terminiert oder nach 55 Schritten nicht terminiert gib Falsch zurück, sonst gib nach Prüfen aller (endlich vieler) Optionen Wahr zurück.

**FSK11-2** ***Beweise prüfen***

In den folgenden Teilaufgaben betrachten wir jeweils einen Beweis, der einen Fehler enthält. Identifizieren Sie diesen Fehler (mit kurzer Begründung).

a) Beweisen oder widerlegen Sie: Die Sprache

$$D = \{w \in \{0,1\}^* \mid M_w \text{ akzeptiert } w \text{ nicht}\}$$

ist semi-entscheidbar.

**Beweis**:
$D$ ist semi-entscheidbar. Um das zu zeigen, konstruieren wir eine DTM $M$, die $D$ semi-entscheidet. Das heißt, dass $M$ für alle Eingaben $w \in D$ hält und für alle Eingaben $w \notin D$ nicht hält.

Angenommen, es gäbe so eine Turingmaschine $M$. Betrachte ein Wort $w \in \{0,1\}^*$.

- Wenn $w \in D$ ist, dann akzeptiert $M_w$ die Eingabe $w$. Somit akzeptiert auch $M$ das Wort $w$.
- Wenn $w \notin D$ ist, dann hält $M_w$ mit Eingabe $w$ nicht. Somit akzeptiert auch $M$ das Wort $w$ nicht.

$M$ semi-entscheidet also $D$.

**LÖSUNGSVORSCHLAG:**
Der Beweis enthält zwei Fehler:

- $w \in D$ bedeutet, dass $M_w$ die Eingabe $w$ *nicht* akzeptiert. Die Lösung geht in den zwei Stichpunkten aber davon aus, dass $w \in D$ das Gegenteil bedeutet.
- Selbst wenn wir diesen Fehler beheben, nimmt die Lösung an, dass ein $M$ mit der gewünschten Eigenschaft existiert, aber wir haben das nie gezeigt. Der Beweis ist also zirkulär: „Unter der Annahme, dass $M$ existiert, existiert $M$."

Tatsächlich ist $D$ nicht semi-entscheidbar, d.h. die Aussage ist falsch. Intuition: Die einzige Möglichkeit, zu testen, ob $M_w$ das Wort $w$ nicht akzeptiert, ist, $M_w$ auf $w$ auszuführen. Wenn $M_w$ dann beliebig lange läuft, kann man nie sagen, ob $M_w$ noch halten (und damit akzeptieren) wird oder nicht.

b) Sei $L_u = \{w\#x \mid w, x \in \{0,1\}^* \text{ und } x \in L(M_w)\}$. Diese Sprache ist semi-entscheidbar, aber nicht entscheidbar.

Zeigen Sie: Die Sprache $L_\text{r} = \{w \in \{0,1\}^* \mid L(M_w) \text{ ist regulär}\}$ ist unentscheidbar.

**Beweis**:
Wir reduzieren $L_\text{u}$ auf $L_\text{r}$. Da $L_\text{u}$ unentscheidbar ist, folgt daraus, dass $L_\text{r}$ unentscheidbar ist.

Sei $v \in \{0,1,\#\}^*$. Wir definieren die Reduktionsfunktion $f$ durch

$$f(v) = \begin{cases} w_{M_3} & \text{falls } v = w\#x \text{ und } M_w \text{ akzeptiert } x \\ w_{M_4} & \text{sonst} \end{cases}$$

Dabei ist $M_3$ eine Turingmaschine, die die reguläre Sprache $\{0,1\}^*$ akzeptiert. $M_4$ ist eine Turingmaschine, die die nichtreguläre Sprache $\{0^n1^n \mid n \in \mathbb{N}\}$ akzeptiert. $w_{M_i}$ ist die Binärkodierung der jeweiligen Turingmaschine.

$M_3$, $M_4$ und die Binärkodierungen sind offensichtlich berechenbar, also ist $f$ berechenbar (und offensichtlich total). Weiterhin gilt:

$$\begin{array}{ll} & v \in L_{\mathrm{u}} \\ \text{g.d.w.} & v = w\#x \text{ und } x \in L(M_w) \\ \text{g.d.w.} & M_{f(v)} \text{ akzeptiert eine reguläre Sprache} \\ \text{g.d.w.} & f(v) \in L_{\mathrm{r}} \end{array}$$

Somit ist $f$ eine valide Reduktionsfunktion und $L_{\mathrm{u}} \leq L_{\mathrm{r}}$.

**LÖSUNGSVORSCHLAG:**
Die Reduktionsfunktion $f$ ist nicht berechenbar. Um zu entscheiden, ob $f(v) = w_{M_3}$ oder $f(v) = w_{M_4}$, müssen wir entscheiden, ob $M_w$ das Wort $x$ akzeptiert. Das ist aber bekanntermaßen unentscheidbar.

**FSK11-3** ***PCP und DFA***

Betrachten Sie das folgende Verfahren $V$:

$V$ berechnet für eine PCP-Instanz $K$ über einem Alphabet $\Sigma$ (d. h. $K = ((x_1, y_1), \ldots, (x_k, y_k))$ mit $x_i, y_i \in \Sigma^+$) und einen deterministischen endlichen Automaten $A = (Z, \Sigma, \delta, z_0, E)$ einen Wahrheitswert.

Dafür wird zunächst die Menge $M \subseteq Z \times Z$ folgendermaßen iterativ bestimmt:

- Anfangs ist $M = \varnothing$
- In jedem Schritt wird für alle $(z_i, z_j) \in M$, sowie für $(z_0, z_0)$ (selbst, wenn es nicht in $M$ enthalten ist) und jedes Wortpaar/Spielstein $T = (x_l, y_l)$ aus $K$ bestimmt, welcher Zustand $z_{i'}$ von $z_i$ aus mit dem oberen Wort von $T$ erreicht wird (d.h. $z_{i'} = \widehat{\delta}(z_i, x_l)$) und welcher Zustand $z_{j'}$ von $z_j$ aus mit dem unteren Wort von $T$ erreicht wird (d.h. $z_{j'} = \widehat{\delta}(z_j, y_l)$).

  Das Tupel $(z_{i'}, z_{j'})$ wird zu $M$ hinzugefügt, wenn es nicht bereits in $M$ vorhanden ist.

- Wenn sich die Menge $M$ nicht mehr ändert, ist dies die engültige Gestalt von $M$ und die Berechnung von $M$ ist beendet.

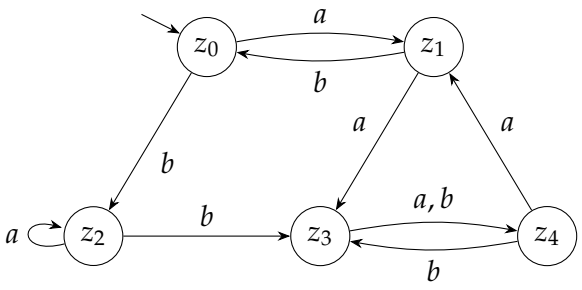
Gibt es nun einen Zustand $z \in Z$ mit $(z, z) \in M$, so wird „wahr“ zurückgegeben, ansonsten „falsch“.

**Hinweis:** Die Endzustandsmenge $E$ wird vom Verfahren $V$ nicht verwendet.

a) Berechnen Sie die Menge $M$ für die PCP-Instanz $K$:

$$\left\{ \begin{bmatrix} bba \\ bb \end{bmatrix}, \begin{bmatrix} abba \\ abbab \end{bmatrix}, \begin{bmatrix} aba \\ bab \end{bmatrix}, \begin{bmatrix} abaaba \\ bbb \end{bmatrix}, \begin{bmatrix} ba \\ bbaabbab \end{bmatrix} \right\}$$

und den Automaten $A$:




Was gibt das Verfahren zurück?

**LÖSUNGSVORSCHLAG:**
Zur Vereinfachung betrachten wir nur, welche Zustandspaare in jedem Schritt zu $M$ hinzugefügt werden, falls sie noch nicht enthalten sind.

- $M = \emptyset$
- Betrachte $(z_0, z_0)$:

$$\begin{aligned} T = (bba, bb) &: (z_4, z_3) \in M \\ T = (abba, abbab) &: (z_2, z_3) \in M \\ T = (aba, bab) &: (z_1, z_3) \in M \\ T = (abaaba, bbb) &: (z_1, z_4) \in M \\ T = (ba, bbaaabbab) &: (z_2, z_3) \in M \end{aligned}$$

- Betrachte $(z_4, z_3)$:

$$
\begin{aligned}
T = (bba, bb) &: (z_1, z_3) \in M \\
T = (abba, abbab) &: (z_2, z_0) \in M \\
T = (aba, bab) &: (z_1, z_0) \in M \\
T = (abaaba, bbb) &: (z_1, z_4) \in M \\
T = (ba, bbaaabbab) &: (z_4, z_3) \in M
\end{aligned}
$$

- Betrachte $(z_2, z_3)$:

$$
\begin{aligned}
T = (bba, bb) &: (z_1, z_3) \in M \\
T = (abba, abbab) &: (z_1, z_0) \in M \\
T = (aba, bab) &: (z_4, z_0) \in M \\
T = (abaaba, bbb) &: (z_1, z_4) \in M \\
T = (ba, bbaaabbab) &: (z_4, z_3) \in M
\end{aligned}
$$

  An dieser Stelle können wir beobachten, dass die rechten Seiten der *Ergebnis*-Tupel identisch ist zu denen für $(z_4, z_3)$. Dies liegt daran, dass das Verfahren die beiden Seiten der Tupel komplett unabhängig voneinander behandelt. Wir können also, um Arbeit zu sparen, falls wir für eine der Seiten des *Eingabe*-Tupels bereits die *Ergebnis*-Tupel kennen, die entsprechende Seite der *Ergebnis*-Tupel einfach ablesen, statt sie erneut zu berechnen.

- Betrachte $(z_1, z_3)$:

$$
\begin{aligned}
T = (bba, bb) &: (z_2, z_3) \in M \\
T = (abba, abbab) &: (z_4, z_0) \in M \\
T = (aba, bab) &: (z_1, z_0) \in M \\
T = (abaaba, bbb) &: (z_1, z_4) \in M \\
T = (ba, bbaaabbab) &: (z_1, z_3) \in M
\end{aligned}
$$

- Betrachte $(z_1, z_4)$:

$$
\begin{aligned}
T = (bba, bb) &: (z_2, z_4) \in M \\
T = (abba, abbab) &: (z_4, z_3) \in M \\
T = (aba, bab) &: (z_1, z_3) \in M \\
T = (abaaba, bbb) &: (z_1, z_3) \in M \\
T = (ba, bbaaabbab) &: (z_1, z_3) \in M
\end{aligned}
$$

An dieser Stelle sind alle Zustände $z_0, z_1, z_2, z_3, z_4$ mindestens einmal (auf der jeweils korrekten Seite für die folgenden Schritte) in einem der betrachteten Tupel vorgekommen. Wir können also ab hier immer ablesen statt zu berechnen.

- Betrachte $(z_2, z_0)$:

$$\begin{aligned} T = (bba, bb) &: (z_1, z_3) \in M \\ T = (abba, abbab) &: (z_1, z_3) \in M \\ T = (aba, bab) &: (z_4, z_3) \in M \\ T = (abaaba, bbb) &: (z_1, z_4) \in M \\ T = (ba, bbaaabbab) &: (z_4, z_3) \in M \end{aligned}$$

- Betrachte $(z_1, z_0)$:

$$\begin{aligned} T = (bba, bb) &: (z_2, z_3) \in M \\ T = (abba, abbab) &: (z_4, z_3) \in M \\ T = (aba, bab) &: (z_1, z_3) \in M \\ T = (abaaba, bbb) &: (z_1, z_4) \in M \\ T = (ba, bbaaabbab) &: (z_1, z_3) \in M \end{aligned}$$

- Betrachte $(z_4, z_0)$:

$$\begin{aligned} T = (bba, bb) &: (z_1, z_3) \in M \\ T = (abba, abbab) &: (z_2, z_3) \in M \\ T = (aba, bab) &: (z_1, z_3) \in M \\ T = (abaaba, bbb) &: (z_1, z_4) \in M \\ T = (ba, bbaaabbab) &: (z_4, z_3) \in M \end{aligned}$$

- Betrachte $(z_2, z_4)$:

$$\begin{aligned} T = (bba, bb) &: (z_1, z_4) \in M \\ T = (abba, abbab) &: (z_1, z_3) \in M \\ T = (aba, bab) &: (z_4, z_3) \in M \\ T = (abaaba, bbb) &: (z_1, z_3) \in M \\ T = (ba, bbaaabbab) &: (z_4, z_3) \in M \end{aligned}$$

Es ist damit: $M = \{(z_4, z_3), (z_2, z_3), (z_1, z_3), (z_1, z_4), (z_2, z_0), (z_1, z_0), (z_4, z_0), (z_2, z_4)\}$
Das Verfahren gibt also „falsch“ zurück.

b) Wenn $V$ auf einer PCP-Instanz $K$ gelaufen ist, was kann man dann aus einer „wahr“- und was aus einer „falsch“-Antwort für die PCP-Instanz schlussfolgern?

Bestimmen Sie mit Begründung getrennt für den „wahr“- und „falsch“-Fall die richtige Antwortkategorie:

- $K$ hat sicher eine Lösung
- $K$ hat sicher keine Lösung
- Man kann damit noch nicht sicher sagen, ob $K$ eine Lösung hat

**LÖSUNGSVORSCHLAG:**
Das Verfahren $V$ betrachtet für alle Folgen von Indizes $i_1, \ldots, i_n$, welche Zustände von $x_{i_1} \cdots x_{i_n}$ und $y_{i_1} \cdots y_{i_n}$ in $A$ erreicht werden. Dabei wird „wahr“ zurückgegeben, wenn es eine Folge $i_1, \ldots, i_n$ gibt, bei der $x_{i_1} \cdots x_{i_n}$ und $y_{i_1} \cdots y_{i_n}$ den gleichen Zustand erreichen. Andernfalls wird „falsch“ zurückgegeben.
„falsch“ bedeutet deshalb, dass $K$ sicher keine Lösung hat:
Wenn $x_{i_1} \cdots x_{i_n}$ nie den gleichen Zustand wie $y_{i_1} \cdots y_{i_n}$ erreicht, so gilt sicherlich $x_{i_1} \cdots x_{i_n} \neq y_{i_1} \cdots y_{i_n}$ für alle Folgen von Indizes $i_1, \ldots, i_n$.
„wahr“ bedeutet dagegen, dass man noch nicht sicher sagen kann, ob $K$ eine Lösung hat: Hierfür kann z. B. ein Automaten mit nur einem Zustand und eine unlösbare PCP-Instanz betrachtet werden. Es ist leicht zu sehen, dass der Algorithmus hier „wahr“ zurückgibt, obwohl die Instanz nicht lösbar ist. Der Algorithmus gibt aber auch bei einer lösbaren PCP-Instanz „wahr“ zurück.

c) Zeigen Sie, dass es PCP-Instanzen $K$ gibt, die keine Lösung haben, bei denen es aber dennoch keinen Automaten $A$ gibt, sodass $V(K, A)$ mit „falsch“ antwortet.

**Hinweis:** Die Unentscheidbarkeit von PCP kann Ihnen beim Beweis helfen

**LÖSUNGSVORSCHLAG:**
Wir führen den Beweis durch Widerspruch und nehmen hierfür an, dass es für jede PCP-Instanz einen Automaten gibt, sodass der Algorithmus mit „falsch“ antwortet, falls die Instanz keine Lösung hat.
Wir können nun sowohl alle Automaten, wie auch alle PCP-Instanzen rekursiv aufzählen. Da der Algorithmus berechenbar ist, könnten wir so die Menge aller PCP-Instanzen ohne Lösung rekursiv aufzählen.
Es wäre somit das Komplement von PCP rekursiv aufzählbar, also semientscheidbar. Zusammen damit, dass, wie aus der Vorlesung bekannt, PCP

semi-entscheidbar ist, wäre also PCP entscheidbar. Dies steht im Widerspruch dazu, dass PCP, wie aus der Vorlesung bekannt, durch Reduktion des Halteproblems auf PCP, unentscheidbar ist.