

Probeklausur

Theoretische Grundlagen der Informatik

Universität: Technische Universität Berlin
Kurs/Modul: Theoretische Grundlagen der Informatik
Bearbeitungszeit: 180 Minuten
Erstellungsdatum: September 19, 2025



Zielorientierte Lerninhalte, kostenlos!
Entdecke zugeschnittene Materialien für deine Kurse:

<https://study.AllWeCanLearn.com>

Theoretische Grundlagen der Informatik

Aufgabe 1. Formale Sprachen und Grammatiken

Bearbeitungszeit: 180 Minuten.

(a) Gegeben sei die Grammatik G_1 mit Regelsatz

$$S \rightarrow AB \mid \varepsilon, \quad A \rightarrow aA \mid a, \quad B \rightarrow bB \mid b.$$

Bestimmen Sie, ob das Wort $w = aabb$ in $L(G_1)$ liegt. Falls ja, geben Sie eine Ableitung an.**(b)** Die Grammatik

$$S \rightarrow aSb \mid bSa \mid SS \mid \varepsilon$$

generiert alle Wörter mit gleich vielen

 $(\#a = \#b)$) über dem Alphabet $\{a, b\}$. Geben Sie ein konkretes Wort $w_1 \in L(S)$ an und geben Sie eine Begründung, warum $w_1 \in L(S)$ liegt.**(c)** Endlicher Automat Gegeben sei der deterministische endliche Automat A mit

$$Q = \{q_0, q_1, q_2\}, \quad \Sigma = \{a, b\}, \quad q_0 \text{ Start}, \quad F = \{q_2\},$$

und Übergänge

$$\delta(q_0, a) = q_0, \quad \delta(q_0, b) = q_1, \quad \delta(q_1, a) = q_2, \quad \delta(q_1, b) = q_0, \quad \delta(q_2, a) = q_2, \quad \delta(q_2, b) = q_2.$$

Geben Sie an, ob das Wort $w = aab$ von A akzeptiert wird. Falls ja, geben Sie eine akzeptierende Folge von Zuständen an.**(d)** Kontextfreie Grammatik, Nicht-Regularität Geben Sie ein Beispiel einer kontextfreien Grammatik an, die nicht regulär ist, und begründen Sie kurz, warum ihre Sprache nicht regulär sein kann. Als Beispiel genügt eine Grammatik mit

$$S \rightarrow aSb \mid \varepsilon.$$

Aufgabe 2. Endliche Automaten, Regularität und Transformationen

(a) Gegeben sei der DFA D mit

$$Q = \{p_0, p_1, p_2\}, \quad \Sigma = \{a, b\}, \quad p_0 \text{ Start}, \quad F = \{p_2\},$$

Übergänge

$$\delta(p_0, a) = p_0, \quad \delta(p_0, b) = p_1, \quad \delta(p_1, a) = p_2, \quad \delta(p_1, b) = p_0, \quad \delta(p_2, a) = p_2, \quad \delta(p_2, b) = p_2.$$

Geben Sie an, ob der Wort $w = abbab$ akzeptiert wird. Falls ja, liefern Sie eine akzeptierende Folge von Zuständen.

(b) Regulärer Ausdruck Geben Sie einen regulären Ausdruck an, der $L(D)$ beschreibt.

(c) Abschluss-Eigenschaften Nennen Sie drei Operationen, unter denen reguläre Sprachen abgeschlossen sind, und geben Sie jeweils eine kurze Begründung an.

Aufgabe 3. Turingmaschinen und Berechenbarkeit

- (a) Skizzieren Sie eine Turingmaschine M , die prüft, ob eine gegebene Binärzeichenfolge eine Zweierpotenz ist. Beschreiben Sie den grundlegenden Arbeitsablauf ohne formale Formalisierung.
- (b) Halteproblem Formulieren Sie das Halteproblem für eine gegebene TMP/Turingmaschine T und erklären Sie, warum dieses Problem allgemein unentscheidbar ist.
- (c) Palindrom-Erkennung Beschreiben Sie in zwei Sätzen, wie eine Turingmaschine M' prüft, ob eine gegebene Zeichenfolge auf dem Eingabeband ein Palindrom ist. Geben Sie keine Lösung an.

Aufgabe 4. Komplexität, P, NP und NP-Vollständigkeit

- (a) Kurze Definitionen Geben Sie formale kurze Definitionen von - P: Probleme, die deterministisch in Polynomzeit gelöst werden, - NP: Probleme, die in Nichtdeterministischer Polynomzeit verifiziert werden können.
- (b) Beispiel und Reduktion Nennen Sie ein NP-vollständiges Problem und skizzieren Sie in zwei bis drei Sätzen, wie eine polynomielle Reduktion von diesem Problem auf ein anderes NP-schweres Problem aufgebaut werden könnte.
- (c) Klassische Beispiele Nennen Sie je drei Beispiele für Probleme, die mit hoher Wahrscheinlichkeit in P liegen, in NP liegen, und NP-vollständig sind. Geben Sie keine Belege oder Beweise.

Lösungen

Aufgabe 1. Formale Sprachen und Grammatiken**Lösungen:**

(a) $w = aabb$ gehört zu $L(G_1)$. Begründung: G_1 erzeugt Strings der Form $a^m b^n$ mit $m \geq 1$, $n \geq 1$ bzw. ε . Hier ist $m = 2$ und $n = 2$. Eine zulässige Ableitung (linkstrukturiert) lautet

$$S \Rightarrow AB \Rightarrow aAB \Rightarrow aaB \Rightarrow aabB \Rightarrow aabb.$$

(b) $S \rightarrow aSb \mid bSa \mid SS \mid \varepsilon$ erzeugt alle Wörter mit $\#a = \#b$ über $\{a, b\}$. **Beispiel:** $w_1 = ab \in L(S)$. Ableitung:

$$S \Rightarrow aSb \Rightarrow a\varepsilon b = ab.$$

Begründung: Für jedes entstehende Wort gilt $\#a = \#b$; Umgekehrt kann man für jedes Wort mit $\#a = \#b$ durch geeignete Anwendungen von SS und Basisfall ε eine Ableitung konstruieren.

(c) $A = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $F = \{q_2\}$ mit

$$\delta(q_0, a) = q_0, \quad \delta(q_0, b) = q_1, \quad \delta(q_1, a) = q_2, \quad \delta(q_1, b) = q_0, \quad \delta(q_2, a) = q_2, \quad \delta(q_2, b) = q_2.$$

Wort $w = aab$ wird nicht akzeptiert. Start in q_0 :

$$a \rightarrow q_0, \quad a \rightarrow q_0, \quad b \rightarrow q_1.$$

Endzustand q_1 (nicht final). Somit wird w von A nicht akzeptiert.

(d) Eine kontextfreie Grammatik, die nicht regulär ist, z. B. $S \rightarrow aSb \mid \varepsilon$. $L(S) = \{a^n b^n \mid n \geq 0\}$ ist nicht regulär (Begründung in Kürze durch Pumping-Lemma; jede Pumping-Lemma-Argumentation verhindert eine Regulärität, da $a^n b^n$ beim Zerlegen in xyz nicht zyklisch in der Länge des Pumpens bleibt).

Aufgabe 2. Endliche Automaten, Regularität und Transformationen

(a) D akzeptiert $w = abbab \implies$ Nein. Ausführung:

$$p_0 \xrightarrow{a} p_0 \xrightarrow{b} p_1 \xrightarrow{b} p_0 \xrightarrow{a} p_0 \xrightarrow{b} p_1.$$

Endzustand p_1 ist nicht final, daher wird w nicht akzeptiert.

(b) $L(D) = \{a^*ba(a|b)^*\}$. *Begründung:* Um in den akzeptierenden Zustand p_2 zu gelangen, muss der Pfad von p_0 nach p_2 über p_1 führen, d. h. vor dem ersten Übergang $p_0 \rightarrow p_1$ muss es eine (beliebige) Sequenz von a 's geben, dann ein b von p_0 nach p_1 , anschließend ein a von p_1 nach p_2 . Nachdem man in p_2 ist, bleibt man dort und liest beliebige Symbole. Die linke Seite ist damit a^*ba gefolgt von beliebigem Rest $(a|b)^*$.

(c) Drei Abschluss-Eigenschaften regulärer Sprachen: - Vereinigung: Falls L_1, L_2 regulär, dann $L_1 \cup L_2$ regulär. Konstruktion z. B. über NFA mit Parallelzuständen. - Konkatenation: Falls L_1, L_2 regulär, dann L_1L_2 regulär. Konstruktion über Produkt-Automat oder NFA mit -Übergängen. - Kleene-Stern (*): Falls L regulär, dann L^* regulär. Konstruktion durch Hinzufügen eines Startzustands mit -Verbindungen zum Endzustand.

Aufgabe 3. Turingmaschinen und Berechenbarkeit

(a) Skizze einer Turingmaschine M , die prüft, ob eine Binärzahl eine Zweierpotenz ist (ohne führende Nullen). *Grob-Working*: - Prüfe, dass die erste Ziffer eine 1 ist, und dass danach nur 0 folgt (sonst Abbruch). - Markiere das erste 1 als verarbeitet (z. B. durch Ersetzen mit X) und scanne den Rest aktiv nach weiteren Einsen; falls eine weitere Eins gefunden wird, ablehnen. - Andernfalls akzeptieren. Der Kernpunkt: eine Potenz von zwei hat genau eine 1 in ihrer Binärdarstellung.

(b) Halteproblem (Halting Problem). **Formulierung**: Gegeben eine Turingmaschine T und Eingabe w , entscheidet, ob T auf w terminiert (hält). **Begründung der Unentscheidbarkeit**: Annahme eines Entscheidungsverfahrens $H(T, w)$ existiere; man konstruiert eine Maschine D , die H nutzt und sich selbst als Eingabe gibt, wobei Widersprüche auftreten (Diagonalisierung). Daraus folgt, dass kein allgemeines Halte-Entscheidungssystem existieren kann.

(c) Palindrom-Erkennung. **Zwei Sätze**: Eine Turingmaschine M' prüft, ob das Eingabewort ein Palindrom ist, indem sie das erste und das letzte Symbol vergleicht (und ggf. beide Symbole markiert), dann den Innenkern rekursiv oder iterativ weiter prüft. Falls ein Paar ungleich ist, lehnt sie ab; sonst fährt sie fort, bis der gesamte Wortkern geprüft ist und akzeptiert.

Aufgabe 4. Komplexität, P, NP und NP-Vollständigkeit

(a) Kurze Definitionen: - P : Probleme, die deterministisch in Polynomzeit gelöst werden können. - NP : Probleme, deren Lösungen in polynomieller Zeit verifiziert (oder äquivalent: von einer NP-zertifizierenden nichtdeterministischen Turingmaschine in Polynomzeit entschieden) werden können.

(b) Beispiel und Reduktion: *NP-vollständiges Problem* ist z. B. SAT. Eine Polynomzeit-Reduktion von SAT auf ein anderes NP-schweres Problem erfolgt, indem man eine Instanz von SAT in eine äquivalente Instanz eines anderen NP-vollständigen Problems transformiert, wobei Gültigkeit der Belegung erhalten bleibt. *Konkret*: Man reduziert SAT auf 3-SAT, indem lange Klauseln in 3-CNF-Klauseln zerlegt werden (mit zusätzlichen Variablen), sodass die ursprüngliche Erfüllbarkeit erhalten bleibt. Die Transformationsregel ist polynomiell, die äquivalente Belegung sorgt dafür, dass Lösungen erhalten bleiben.

(c) Klassische Beispiele (ohne Belege): - In P : Kürzeste Pfade (Dijkstra-Algorithmus), Minimaler Spannbaum (Kruskal/Prim), Topologisches Sortieren. - In NP : Hamiltonianer Pfad, Entscheidbare Version von Subset Sum, Graph Coloring (mit variablem k). - NP-Vollständig: SAT, 3-SAT, Clique (bzw. Hamiltonian Cycle).