

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

Bearbeitungszeit: 180 Minuten.

Aufgabe 1.

(a) Definieren Sie eine Sprache über dem Alphabet $\{a, b\}$ durch

$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ enthält eine gerade Anzahl von } a\text{'s} \}.$$

Geben Sie eine deterministische endliche Automaton(M) an, der L_1 akzeptiert, oder beschreiben Sie eine äquivalente reguläre Grammatik.

(b) Seien $\Sigma = \{a, b\}$ und $L_2 = \{ a^n b^n \mid n \geq 0 \}$. Geben Sie eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ an, die L_2 erzeugt. Definieren Sie Startsymbol, Produktionen und erläutern Sie die Erzeugung grob.

(c) Beweisen Sie mittels Pumping-Lemma, dass $L_3 = \{ a^n b^n c^n \mid n \geq 0 \}$ nicht kontextfrei ist.

(d) Skizzieren Sie die Umformung der Grammatik G aus Teil (b) in Chomsky-Normalform. Nennen Sie grob die notwendigen Schritte und welche Typen von Produktionen entstehen können.

Aufgabe 2.

(a) Seien $\Sigma = \{0, 1\}$ und $L_4 = \{w \in \{0, 1\}^* \mid \text{Anzahl der Einsen in } w \text{ ist durch } 3 \text{ teilbar}\}$. Geben Sie einen deterministischen endlichen Automaten M mit Zustandsmenge und Übergangsrelation an, der L_4 akzeptiert.

(b) Zeigen Sie, dass L_4 auch von einem Kellerautomaten akzeptiert wird, und skizzieren Sie eine Stack-Verifikation, die dies ermöglicht.

(c) Geben Sie eine Sprache L_5 an, die von einem Kellerautomaten akzeptiert wird, aber nicht von einem endlichen Automaten. Begründen Sie Ihre Wahl.

(d) Diskutieren Sie, ob die folgende Sprache von einem endlichen Automaten akzeptiert werden kann:

$$L_6 = \{w \in \{a, b\}^* \mid \#a(w) = \#b(w)\}.$$

Begründen Sie Ihre Einschätzung.

Aufgabe 3.

(a) Beschreiben Sie eine Turingmaschine M , die entscheidet

$$L_7 = \{ w \in \{0, 1\}^* \mid w \text{ enthält genau zwei Einsen} \}.$$

Geben Sie das Eingabeformat an und die groben Berechnungsschritte.

(b) Zeigen Sie, dass das Halteproblem HALT unentscheidbar ist. Skizzieren Sie die zentrale Idee der Begründung, ohne vollständige Beweise zu liefern.

(c) Erläutern Sie die Idee einer Reduktion von SAT auf 3-SAT, und warum Reduktionen zentral für NP-Vollständigkeit sind. Geben Sie eine grobe Transformationsidee an.

(d) Diskutieren Sie, wie die Begriffe P, NP und NP-Vollständigkeit im Konzept der theoretischen Informatik interpretiert werden und welche praktischen Auswirkungen NP-Vollständigkeit auf Lösungsansätze hat.

Aufgabe 4.

- (a) Definieren Sie die Klassen P, NP und NP-Vollständigkeit. Formulieren Sie präzise, wie eine Reduktion zwischen Problemen durchzuführen ist.
- (b) Beweisen Sie, dass das Problem 3-SAT NP-vollständig ist, indem Sie eine Reduktion von SAT auf 3-SAT verwenden. Skizzieren Sie die Transformation der Klauseln und die Erhaltung der Lösbarkeit.
- (c) Nennen Sie zwei typische NP-vollständige Probleme und geben Sie jeweils eine grobe Reduktionsidee an.
- (d) Diskutieren Sie kurz, wie $P \subseteq NP$ interpretiert wird und welche praktischen Implikationen NP-Vollständigkeit für algorithmische Lösungsansätze hat.

Lösungen

Bearbeitungszeit: 180 Minuten.

Lösung 1.(a).

Deterministischer endlicher Automat M (oder äquivalente reguläre Grammatik).

- DFA M: $M = (Q, \Sigma, \delta, q_0, F)$ mit - $Q = \{q_0, q_1\}$, - $\Sigma = \{a, b\}$, - Startzustand q_0 , - Endzustand $F = \{q_0\}$, - Übergangsfunktion δ : - $\delta(q_0, a) = q_1$, $\delta(q_0, b) = q_0$, - $\delta(q_1, a) = q_0$, $\delta(q_1, b) = q_1$.

Begründung: Der Zustand encodiert Parität der Anzahl der vorkommenden a . Sind a 's gerade gezählt, befindet sich der Automat im Zustand q_0 und akzeptiert. Jeder durchlaufene a wechselt die Parität, b beeinflusst die Parität nicht.

- Äquivalente reguläre Grammatik $G = (V, \Sigma, P, S)$: - $V = \{S, A\}$, $\Sigma = \{a, b\}$, Startsymbol S . - Produktionen P : - $S \rightarrow bS \mid aA$, - $A \rightarrow bA \mid aS$. - Erklärung: S steht für gerade Anzahl von a 's, A für ungerade Anzahl. Jede Produktion erhöht bzw. bewahrt die Parität entsprechend; akzeptiert werden alle erzeugten Sätze, die in S enden.

Lösung 1.(b).

Kontextfreie Grammatik $G = (V, \Sigma, P, S)$ für $L_2 = \{a^n b^n \mid n \geq 0\}$.

- $\Sigma = \{a, b\}$, $V = \{S\}$, Startsymbol S . - Produktionen P : - $S \rightarrow aSb \mid \varepsilon$. - Erläuterung: Jede Ableitung erzeugt gleich viele a 's am Anfang und b 's am Ende; damit ist genau L_2 erzeugt.

Lösung 1.(c).

Pumping-Lemma: Nicht-KF (für $L_3 = \{a^n b^n c^n \mid n \geq 0\}$).

- Angenommen, L_3 sei kontextfrei und p sei eine Pumping-Länge. - Wähle $w = a^p b^p c^p \in L_3$. - Nach Pumping-Lemma lässt sich $w = xyz$ mit $|xy| \leq p$, $|y| > 0$ schreiben, sodass $xy^i z \in L_3$ für alle $i \geq 0$. - Da $|xy| \leq p$ gilt, besteht y nur aus Buchstaben aus dem Prefix a^p ; also $y = a^k$ mit $k > 0$. - Wählen wir $i = 0$: $xz = a^{p-k} b^p c^p$. - Die Anzahl der a 's ist ungerade oder ungleich der Anzahl der b 's bzw. c 's; damit gilt $xz \notin L_3$. - Widerspruch. Also ist L_3 nicht kontextfrei.

Lösung 1.(d).

Skizzierte Umformung von G in Chomsky-Normalform (CNF).

- Ausgangs-Grammatik $G = (\{S\}, \{a, b\}, P, S)$ mit $P = \{S \rightarrow aSb, S \rightarrow \varepsilon\}$. - Schritte (grobe Skizze): 1) Neuer Startsymbol S_0 einführen, um ε abzubilden: $S_0 \rightarrow S \mid \varepsilon$ (Beibehaltung von ε am Start). 2) Terminale auf Nichtterminale abschreiben: $A \rightarrow a$, $B \rightarrow b$. 3) Die Regel $S \rightarrow aSb$ in CNF überführen: - Zunächst $S \rightarrow ASB$ (mit A und B wie oben). - Dann eine Zwischenregel einführen: $S \rightarrow AT$, $T \rightarrow SB$. 4) Unit-Produktionen entfernen (hier zielt man darauf ab, startseitig den Übergang direkt in CNF zu haben). Eine mögliche CNF-Form (eine mögliche, grobe CNF-Variante) lautet: - $S_0 \rightarrow AT \mid \varepsilon$ - $T \rightarrow SB$ - $S \rightarrow AT$ - $A \rightarrow a$ - $B \rightarrow b$

- Typen von Produktionen in CNF: $A \rightarrow BC$ oder $A \rightarrow a$; (Zustzlich Startregel $S_0 \rightarrow \varepsilon$ zulässig, falls ε zur Sprache gehört.)

- Hinweis: Die obige Skizze illustriert die typischen Schritte (Elimination von ε -Produktionen, Beseitigung von Unit-Produktionen, Ersetzung von Terminalsymbolen durch Nichtterminale, Zerlegung langer RHS in binäre Produktionen). Eine vollständige CNF-Umformung erfordert die vollständige Eliminierung sämtlicher Einheiten und eine rigorose Umformung aller Produktionen in die binäre Form; das hier

dargestellte Muster verdeutlicht jedoch die Art der Umformungen und die entstehenden Produktionen.

Lösung 2.(a).

Deterministischer endlicher Automat für $L_4 = \{w \in \{0,1\}^* \mid \#1(w) \equiv 0 \pmod{3}\}$.

- **DFA M mit Zuständen $Q = \{q_0, q_1, q_2\}$, Startzustand q_0 , Endzustand $F = \{q_0\}$, Alphabet $\Sigma = \{0,1\}$.** - **Übergänge (Modulo-3-Zähler der Einsen):** - von q_0 : $\delta(q_0, 0) = q_0$, $\delta(q_0, 1) = q_1$ - von q_1 : $\delta(q_1, 0) = q_1$, $\delta(q_1, 1) = q_2$ - von q_2 : $\delta(q_2, 0) = q_2$, $\delta(q_2, 1) = q_0$ - **Begründung:** Die Anzahl der Einsen wird modulo 3 gezählt; akzeptiert wird, wenn der Rest 0 ist (also in q_0).

Lösung 2.(b).

Kellerautomat (Pushdown-Automat, PDA) für L_4 (reguläre Sprache).

- Da L_4 regulär ist, akzeptiert ein PDA diese Sprache auch durch endlichen Anteil des Stack-Verhaltens. - Eine naheliegende PDA-Implementierung: - **Stackalphabet:** $\{Z, R_0, R_1, R_2\}$ mit Startstapelzeichen Z und Startzustand q . - **Zustandsmatrix (eine einfache Form):** - Für Input 0: der Stackinhalt bleibt unverändert (Top bleibt gleich). - Für Input 1: Top-Symbol aktualisiert gemäß Restklasse: $R_0 \rightarrow R_1$, $R_1 \rightarrow R_2$, $R_2 \rightarrow R_0 \pmod{3}$. - Startzustand q mit initialem Stack ZR_0 (Z als Bodensymbol, top ist R_0). - **Abschlussbedingung:** Akzeptieren, wenn der Input vollständig gelesen ist und der Top des Stacks R_0 ist (entspricht Rest 0). - **Explanation:** Der PDA zählt die Anzahl der Einsen modulo 3 auf dem Stack, während Nullen das Zählen bzw. den Stack nicht verändern. Am Ende akzeptiert er, wenn der Rest 0 ist.

Lösung 2.(c).

L5: Sprache, die von einem PDA akzeptiert wird, aber nicht von einem DFA.

- **Vorschlag:** $L_5 = \{a^n b^n \mid n \geq 0\}$. - **Begründung:** - L_5 ist kontextfrei; es gibt eine PDA, die genau diese Gleichheit von a - und b -Anzahlen mittels Stack prüfen kann (Push für jedes a , Pop für jedes b ; am Ende leerer Stack akzeptiert). - L_5 ist jedoch nicht regulär (Beispiel mittels Pumping-Lemma: wähle $w = a^p b^p$; dem Pumping-Lemma widerspricht, da das Pumpen von Teilen aus dem Anfangsblock die Gleichheit von Zählungen zerstört). - **Begründung für Wahl:** Diese Sprache illustriert einen klassischen Unterschied zwischen Regularität und Kontextfreiheit – von einem PDA akzeptierbar, aber nicht von einem DFA.

Lösung 2.(d).

Kann L_6 von einem DFA akzeptiert werden? $L_6 = \{w \in \{a,b\}^* \mid \#a(w) = \#b(w)\}$.

- **Nein. L_6 ist nicht regulär.** - **Begründung (Pumping Lemma):** Sei p die Pumping-Länge. Betrachte $w = a^p b^p \in L_6$. Zerlege $w = xyz$ mit $|xy| \leq p$ und $|y| > 0$. Dann enthält y nur a 's. Wähle $i = 0$: $xz = a^{p-|y|} b^p$; hier gilt $\#a \neq \#b$. Damit ist $xz \notin L_6$, Widerspruch. Also ist L_6 nicht regulär.

Lösung 3.(a).

Turingmaschine M , die genau zwei Einsen akzeptiert:

- Eingabeformat: ein Band mit $w \in \{0,1\}^*$ ohne zusätzliche Beschränkungen (andere Zeichen als 0/1 werden verworfen/abgelehnt). - Vorgehen in drei Phasen (Markieren und Zählen): 1) Suche nach der ersten Eins: - Kopf wandert links, bis eine 1 gefunden wird. - schreibe ein Markierungssymbol (z. B. X) über diese 1; wenn keine 1 gefunden wird, akzeptiert/verweigert entsprechend. 2) Suche nach der zweiten Eins: - Kopf fortsetzen, Nullen ignorieren, bis die zweite 1 gefunden wird. - schreibe ein weiteres Markierungssymbol (z. B. Y) über diese zweite 1. 3) Prüfung, ob exakt zwei Einsen vorhanden sind: - Kopf fährt zum rechten Rand und prüft, ob danach noch eine weitere 1 vorkommt. - Wenn noch eine 1 gefunden wird, lehnt ab; ansonsten akzeptiert. - Akzeptieren: Der Ringzustand/Endzustand wird erreicht, wenn genau zwei Einsen markiert sind und danach keine weiteren Einsen mehr erscheinen.

Hinweis: Dies ist eine typische, kompakte Beschreibung eines Ein-Tape-Türingautomaten mit Markierungen, der die Eigenschaft „genau zwei Einsen“ überprüft.

Lösung 3.(b).

Halteproblem HALT ist unentscheidbar.

- Standardbeweis durch Reduktion von Diagonalargumenten: Angenommen, HALT sei entscheidbar durch eine Turingmaschine H . - Konstruktion einer Turingmaschine D (Diagonal), die als Eingabe die Beschreibung einer Turingmaschine M erhält und sie selbst auf Eingabe D verhält, führt zu Widerspruch. - Die Kernaussage: Widerspruch entsteht, indem man $D(D)$ betrachtet und zeigt, dass D genau dann hält, wenn D nicht hält – ein klassischer Beweis durch Reduktion, der die Unentscheidbarkeit von HALT zeigt.

Lösung 3.(c).

Reduktion von SAT auf 3-SAT (grobe Transformationsidee).

- Ziel: Jede Klausel einer CNF-Form ϕ in eine Menge 3-literaler Klauseln umformen, so dass ϕ lösbar ist genau dann ϕ' lösbar. - Vorgehen (Standardtechnik): - Für eine Klausel mit Länge $k > 3$ füge neue Variablen $y_1, \dots, y_{k-3}$ ein und ersetze die Klausel

$$(l_1 \vee l_2 \vee \dots \vee l_k)$$

durch eine Konjunktion von 3-literal-Klauseln:

$$(l_1 \vee l_2 \vee y_1) \wedge (\neg y_1 \vee l_3 \vee y_2) \wedge (\neg y_2 \vee l_4 \vee y_3) \wedge \dots \wedge (\neg y_{k-3} \vee l_k).$$

- Klauseln mit $k = 1$ oder $k = 2$ werden entsprechend durch kleine Umformungen (Beibehaltung oder Hinzufügen von Dummy-Variablen) angepasst, damit alle Klauseln drei Literale haben. - Eigenschaft: Die Umformung ist in polynomieller Zeit; ϕ ist genau dann satisfizierbar, wenn ϕ' satisfizierbar.

Lösung 3.(d).

Interpretation von P, NP, NP-Vollständigkeit; praktische Auswirkungen.

- P-Theorie: Losbare Probleme in polynomialer Zeit durch deterministische Turingmaschinen. - NP: Nichtdeterministische Polynomialzeit-Lösungen – von außen

überprüfbar, ob eine gegebene Lösung korrekt ist. - NP-Vollständigkeit: Ein Problem ist NP-vollständig, wenn es in NP liegt und jedes Problem in NP in polynomieller Zeit darauf reduzierbar ist. - Praktische Auswirkungen: - Falls $P = NP$, gibt es keine generischen polynomialzeitlichen Algorithmen für NP-vollständige Probleme. - In Praxis werden oft heuristische, approximative oder spezialisierte Algorithmen eingesetzt; viele NP-vollständige Probleme treten in der Praxis mit typischen Strukturen auf, die effektive Lösungen ermöglichen. - NP-Vollständigkeit motiviert die Entwicklung von Grenztheorie, FPT-Analysen, Parameterelbst-Optimierungen und problem-spezifischen Heuristiken.

Lösung 4.(a).

Definitionen und Reduktionsbegriff.

- Klassen: P, NP und NP-Vollständigkeit. - Reduktion: Sei A, B Sprachen. Eine Reduktion $A \leq_p B$ bedeutet, dass es eine polynomiell berechenbare Funktion f gibt mit

$$\forall x : x \in A \text{ gl. } f(x) \in B.$$

- Reduktion erklärt, wie die Lösung eines Problems in Polynomialzeit in ein anderes Problem übertragen werden kann.

Lösung 4.(b).

Beweis, dass 3-SAT NP-vollständig ist (Reduktion von SAT auf 3-SAT).

- Vorgehen: Gegeben sei eine beliebige SAT-Formel ϕ in konjunktiver Normalform (CNF). Alle Klauseln fester Länge ≥ 3 werden schrittweise in 3-literal-Klauseln transformiert, wie in 3.(c) beschrieben. - Transformationsfunktion f ist polynomiell berechenbar. - Erhaltung der Lösbarkeit: ϕ ist erfüllbar genau dann, wenn $\phi' = f(\phi)$ erfüllbar ist. - Somit ist 3-SAT NP-vollständig.

Lösung 4.(c).

Zwei typische NP-vollständige Probleme und grobe Reduktionsideen.

- Beispiel 1: Vertex-Cover - Eingabe: Graph $G = (V, E)$ und Zahl k . - Frage: Gibt es eine Teilmenge $C \subseteq V$ mit $|C| \leq k$, sodass jede Kante mindestens einen Endknoten in C hat? - Reduktionsidee: Von einer FORMEL (z. B. 3-SAT) zu Vertex-Cover; mittels standardisiertem Gadget-Design wird eine Belegung erzeugt, die genau dann existiert, wenn die Klauseln erfüllbar sind. - Beispiel 2: Clique - Eingabe: Graph G und Zahl k . - Frage: Existiert eine k -Clique in G ? - Reduktionsidee: Aus SAT/3-SAT oder Vertex-Cover lassen sich entsprechende Graphen bauen, so dass eine erfüllende Belegung bzw. eine kleine Vertex-Cover-Schnittstelle eine k -Clique ergibt. (Grobe Idee: Konstruieren eines Graphen, in dem Clique-Existenz mit einer logischen Bedingung verknüpft ist.)

Lösung 4.(d).

Interpretation von $P \subseteq NP$; Implikationen.

- Interpretation: Alle in P lösbaren Probleme liegen auch in NP (weil Polynomialzeit-Lösungen von Zertifikaten geprüft werden können; Null-Zertifikate existieren). Die Frage, ob P gleich NP ist, bleibt offen. - Praktische Implikationen: - Falls $P = NP$, gäbe es mittlere, generische polynomialzeitliche Algorithmen für NP-vollständige

Probleme. - Ist P ungleich NP , bleiben NP -vollständige Probleme schwer; Fokus liegt auf heuristischen Methoden, Approximationen, FPT-Algorithmen (parametrisiert) und Problem-spezifischen Strukturen, die in der Praxis genutzt werden. - Theoretische Erkenntnisse beeinflussen Lösungsansätze, Bewertung von Algorithmen und Verständnis von Komplexität in der Informatik.