UNIVERSITÄT DES SAARLANDES
FR 6.2 – Informatik
Lehrstuhl für Kryptographie und Sicherheit
Birgit Pfitzmann
Ahmad-Reza Sadeghi
Sandra Steinbrecher

Klausur mit Lösungsvorschlägen zu Automaten, Berechenbarkeit und Komplexitätstheorie (Informatik 3) (WS 2000/2001)

Zur Klausur waren keine Hilfsmittel zugelassen. Bearbeitungszeit waren 180 Minuten (im Mittel je Klausurpunkt eine Minute Bearbeitungszeit vorgesehen). Die Lösungen sind kursiv gedruckt.

1. Aufgabe *(Reguläre Ausdrücke und endliche Automaten)* $(4+5+5+16 = 30$ Punkte)

a) Geben Sie die formale Definition eines nichtdeterministischen endlichen Automaten an.

Siehe Def. 1.17: 5-*Tupel* $(Q, \Sigma, \delta, q_0, F)$ *mit*

- Q *endliche Menge (von Zuständen),*
- Σ *endliches Alphabet (Menge),*
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$ *Übergangsfunktion,*
- $q_0 \in Q$ *Startzustand,*
- $F \subseteq Q$ *Menge von Endzuständen.*

b) Geben Sie das Pumpinglemma für reguläre Sprachen an.

Siehe Theorem 1.37: Sei A reguläre Sprache, dann $\exists$ Pumpingzahl p, so dass $\forall s \in A$ mit $|s| \geq p$ gilt, dass s aufgeteilt werden kann in $s = xyz$ mit:

(a) $\forall i \geq 0$: $xy^iz \in A$,
(b) $|y| > 0$,
(c) $|xy| \leq p$.

c) Wie ergibt sich die Pumpingzahl p im Beweis des Pumpinglemmas? Geben Sie eine informelle Begründung an.

Siehe Beweis(idee) zu Theorem 1.37: Die Pumpingzahl p ist die Anzahl der Zustände eines Automaten, der die gegebene reguläre Sprache akzeptiert. Wenn $|s| > p$, muss bei der Abarbeitung von s also ein Zustand mehrfach durchlaufen werden. Um diesen Zustand ergibt sich ein Kreis, in dem ein Teilwort y von s abgearbeitet wird, was in einem Wort in der regulären Sprache also beliebig oft vorkommen darf.

d) Sind die folgenden Sprachen über $\Sigma = \{0, 1, 2\}$ regulär?

- $L_1 = \{w \in \Sigma^* \mid \text{Abstand von jeweils zwei 0en in } w \text{ ist durch 3 teilbar}\}$.
- $L_2 = \{w \in \Sigma^* \mid w = 0^i1^*2^i \text{ und } i \in \mathbb{N} \text{ ist durch 3 teilbar}\}$.

Wenn ja, geben Sie die Definition eines endlichen Automaten (mit Zustandsübergängen als Bild) an, der sie akzeptiert, **und** einen regulären Ausdruck, der sie beschreibt. Wenn nein, beweisen Sie die Nichtregularität mithilfe des Pumpinglemmas.

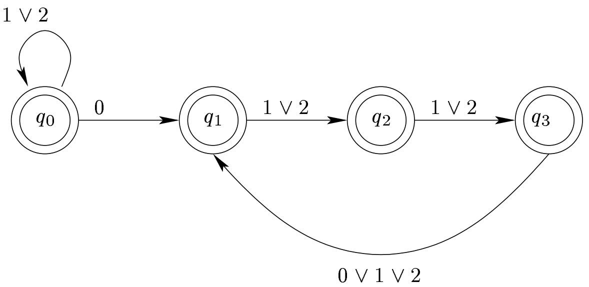
- *L_1 ist regulär:*

  Möglicher regulärer Ausdruck: $(1 \cup 2)^*(0 \cup \epsilon)\left((1 \cup 2)(1 \cup 2)(0 \cup 1 \cup 2)\right)^*(1 \cup 2)^*$

  NEA: $(Q, \Sigma, \delta, q_0, F)$ *mit*

  - $Q = \{q_0, q_1, q_2, q_3\}$
  - $\Sigma = \{0, 1, 2\}$
  - $F = Q$

  und δ gegeben durch



- *L_2 ist nicht regulär:*

  Kurze Lösung (gemischt aus Example 1.38 und 1.39): *Angenommen, L_2 sei regulär, dann würde das Pumpinglemma gelten. Sei p Pumpingzahl und $s = 0^{p'}2^{p'} \in L_2$ mit $p' = 3 \cdot \lceil \frac{p}{3} \rceil$. Bei der Einteilung von s in $s = xyz$ mit $|y| > 0$ und $|xy| \leq p$ kann y nur Nullen enthalten. Also gilt $w' = xy^2z \notin L_2$, da w' mehr 0en als 2en enthält. Widerspruch.*

  Zweite Lösung (genau wie Example 1.38): *Angenommen, L_2 sei regulär, dann würde das Pumpinglemma gelten. Sei p Pumpingzahl und $s = 0^{p'}2^{p'} \in L_2$ mit $p' = 3 \cdot \lceil \frac{\lceil p/2 \rceil}{3} \rceil$. Es gilt $|s| \geq p$. Unabhängig davon wie s in $s = xyz$ mit $|y| > 0$ und $|xy| \leq p$ unterteilt wird, zeigen wir, dass $\exists\, i \geq 0$ mit $xy^iz \notin L_2$.*

  - *Wenn y nur 0en enthält, gilt $w' = xy^2z \notin L_2$, da w' mehr 0en als 2en enthält.*
  - *Wenn y nur 2en enthält, gilt ebenfalls $w' = xy^2z \notin L_2$, da w' mehr 2en als 0en enthält.*
  - *Wenn y die Form $0^{k_0}2^{k_2}$ hat (mit $k_0, k_2 > 0$), gilt ebenfalls $w' = xy^2z \notin L_2$, da w' den Teilstring $0^{k_0}2^{k_2}0^{k_0}2^{k_2}$ enthält.*

2. Aufgabe *(Kontextfreie Grammatiken)* $(6 + 5 + 15 + 4 = 30$ Punkte)

a) Wir betrachten die Grammatik $G = (V, \Sigma, R, E)$ zur Erzeugung korrekter arithmetischer Ausdrücke. Dabei sind $V = \{E, F\}$, $\Sigma = \{a, b, c, d, +, -, \cdot, (,)\}$ und

$$\begin{aligned} R = \{E &\rightarrow E + E \mid E - E \mid F, \\ F &\rightarrow F \cdot F \mid (E) \mid a \mid b \mid c \mid d\}. \end{aligned}$$

Geben Sie vollständige Ableitungen (keine Ableitungsbäume) für $(a+b-c)\cdot d$ und $(d-((c+b)\cdot a)$ an, sofern sie in $L(G)$ liegen, andernfalls eine kurze Begründung.

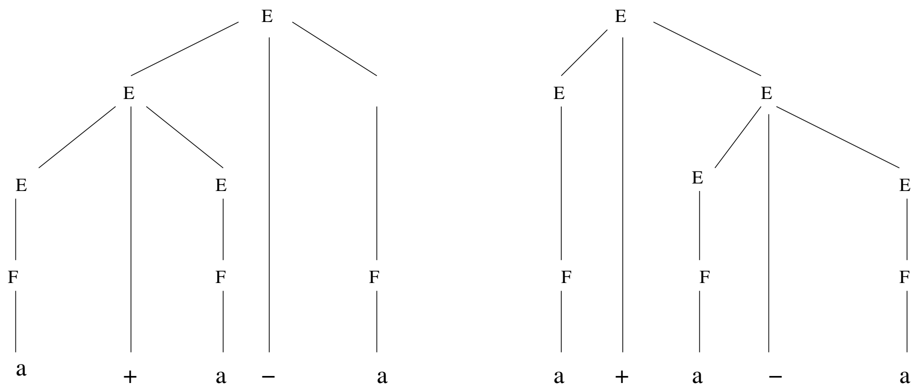
- *Ableitung für* $(a+b-c)\cdot d$*:*

$$\begin{aligned} E &\rightarrow F \\ &\rightarrow F\cdot F \\ &\rightarrow (E)\cdot F \\ &\rightarrow (E)\cdot d \\ &\rightarrow (E-E)\cdot d \\ &\rightarrow (E+E-E)\cdot d \\ &\rightarrow (F+E-E)\cdot d \\ &\rightarrow (F+F-E)\cdot d \\ &\rightarrow (F+F-F)\cdot d \\ &\rightarrow (a+b-c)\cdot d \end{aligned}$$

- $(c-((a+b)\cdot d)$ *wird von der Grammatik nicht erzeugt, da es 3 öffnende, aber nur 2 schließende Klammern hat. Klammern werden in* G *aber nur als Paar aus* (*und*) *erzeugt (von einer einzigen Regel).*

b) Ist die Grammatik G aus Teil a) ein- oder mehrdeutig? Begründen Sie Ihre Antwort kurz.

G ist mehrdeutig. Beispiel ist $a+a-a$ *(vgl. Figure 2.3) mit folgenden Ableitungsbäumen:*



c) Sind die folgenden Sprachen über $\Sigma = \{x, y, z\}$ kontextfrei?

- $L_1 = \{x^i y^j z^k \mid i, j, k \in \mathbb{N}_0 \wedge i = j = k\}$.
- $L_2 = \{x^i y^j z^k \mid i, j, k \in \mathbb{N}_0 \wedge i = j\}$.
- $L_3 = \{x^i y^j z^k \mid i, j, k \in \mathbb{N}_0 \wedge j = k\}$.

Wenn ja, geben Sie die Definition eines Kellerautomaten an, der sie akzeptiert (mit Zustandsübergängen als Bild). Wenn nein, beweisen Sie die Nichtkontextfreiheit mithilfe des Pumpinglemmas.

- *L_1 ist nicht kontextfrei (vgl. Example 2.20):*

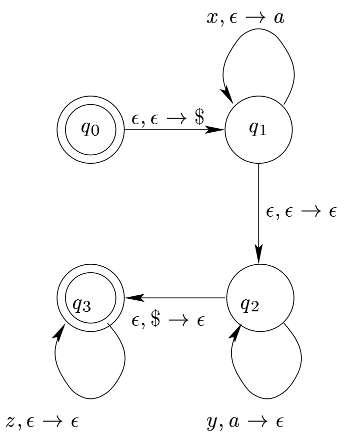
  Angenommen, L_1 sei kontextfrei, dann würde das Pumpinglemma gelten. Sei p die Pumpinglänge und $s = x^p y^p z^p \in L_1$. Unabhängig davon wie s in $s = abcde$ mit $|bd| > 0$ und $|bcd| \leq p$ unterteilt wird, zeigen wir, dass $\exists\ i \geq 0$ mit $ab^icd^ie \notin L_1$.

  - *Wenn b oder d mehr als ein Alphabetsymbol x, y oder z enthält, dann gilt $ab^2cd^2e \notin L_1$, da es nicht mehr die korrekte Form $x^iy^jz^k$ hat.*
  - *Wenn b oder d jeweils höchstens ein Alphabetsymbol x, y oder z enthalten, dann gilt $ab^2cd^2e \notin L$, da es nicht mehr die gleiche Anzahl jedes Symbols enthält. (Mindestens eins wird vermehrt, mindestens eins bleibt.)*

- *L_2 ist kontextfrei: Folgendes $(Q, \Sigma, \Theta, \delta, q_0, F)$ akzeptiert L_2:*

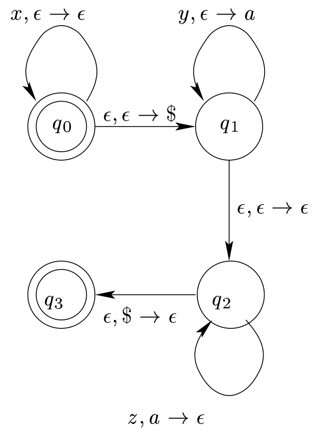
- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{x, y, z\}$,
- $\Theta = \{\$, a\}$,
- $F = \{q_1, q_4\}$

und δ definiert durch folgendes Bild:



- *L_3 ist kontextfrei: Folgendes $(Q, \Sigma, \Theta, \delta, q_0, F)$ akzeptiert L_3:*
  - $Q = \{q_0, q_1, q_2, q_3\}$
  - $\Sigma = \{x, y, z\}$,
  - $\Theta = \{\$, a\}$,
  - $F = \{q_1, q_4\}$

  und δ definiert durch folgendes Bild:



d) Ist die Klasse der kontextfreien Sprachen unter Schnittbildung $\cap$ abgeschlossen? Beweisen Sie Ihre Antwort.

Nein. Gegenbeispiel ist der vorige Aufgabenteil: L_2 und L_3 sind kontextfrei, $L_1 = L_2 \cap L_3$ aber nicht.

3. Aufgabe *(Turingmaschinen und Laufzeitanalyse)* $(5 + 13 + 7 + 5 = 30$ Punkte)

a) Geben Sie die formale Definition einer n-Band-Turingmaschine an.

Vgl. Def. 3.1 und Kap. 3.2: Eine n-Band-Turingmaschine ist ein 7-Tupel $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ mit

- *Q Zustandsmenge,*
- *Σ Eingabealphabet mit $\_ \notin \Sigma$ (wobei $\_$ das Leerzeichen sei),*
- *Γ Bandalphabet mit $\_ \in \Gamma$ und $\Sigma \subset \Gamma$,*
- *$\delta : Q \times \Gamma^n \to Q \times \Gamma^n \times \{L, R\}^n$ Übergangsfunktion.*

- $q_0 \in Q$ *Startzustand,*
- $q_{accept} \in Q$ *akzeptierender Endzustand,*
- $q_{reject} \in Q$ *ablehnender Endzustand mit* $q_{reject} \neq q_{accept}$.

b) Beschreiben Sie (keine formale Definition) eine 2-Band-Turingmaschine, die die unäre Darstellung einer Zahl aus $\mathbb{N}_0$ in ihre binäre verwandelt. Auf Band 1 steht die Eingabe und auf Band 2 steht am Ende die korrekte Ausgabe. Versuchen Sie das (lange) Eingabeband read-only zu behandeln. (Es gibt eine solche Lösung, aber wenn Ihnen keine einfällt, nehmen Sie eine andere.)

Anm: Die Beschreibung sollte aber auch nicht nur 2 Sätze haben, sondern halbwegs ausführlicher Pseudocode entsprechend der Punktzahl.

Die Grundidee der folgenden Maschine ist, auf Band 2 einen Zähler zu führen, der bei jeder 1 auf Band 1 um eins erhöht wird.

Auf Band 1 *steht die Eingabe in unärer Darstellung, also in der Form* 1^n *mit* $n \in \mathbb{N}_0$

(a) *Zunächst wird die Eingabe auf Band* 1 *geprüft, ob sie sie nur aus* 1*en besteht. Wenn ja, gehe zurück zum Bandanfang. Wenn nicht, endet die TM im Fehlerzustand.*

(b) *Ist die Eingabe leer (d.h. vorn auf Band 1 steht* $\_$*), so schreiben wir* 0 *auf Band* 2 *und die Maschine hält.*

(c) *Andernfalls schreiben wir* \$0 *auf Band* 2*.*

(d) *Wiederhole folgendes, solange auf Band 1 eine 1 gelesen wird:*
   - *Der Kopf auf Band* 2 *fährt zum rechtesten Zeichen (d.h. dem Zeichen vor dem ersten* $\_$*, das ist die niederwertigste Stelle des Zählers).*
   - *Solange er dort Einsen liest, kippt er sie auf 0 und geht ein Feld nach links (Überträge).*
   - *Die erste auftretende* 0 *wird durch* 1 *ersetzt. Wird keine* 0 *gefunden, sondern zuerst der \$ (d.h. das linke Bandende), so wird der Zähler (d.h. der Inhalt von Band 2 außer dem \$) um eins nach rechts verschoben, wobei eine* 1 *vorn angehängt wird.*
   - *Fahre auf Band 1 um eins nach rechts.*

(e) *Nun wird noch der \$ auf Band 2 gelöscht und der Inhalt um eins nach links geschoben.*

c) Analysieren Sie kurz Platz- und Zeitkomplexität Ihrer Maschine aus Teil b) und geben Sie das Ergebnis in möglichst genauer O-Notation an. Wenn Sie das Eingabeband read-only behandeln konnten, gehen Sie davon aus.

Sei n *die Eingabelänge.*

- *Die Platzkomplexität von Band* 1 *ist* n*, die von Band* 2 *ist* $\log(n)$*. Wird die angegebene TM dementsprechend als TM mit getrenntem Read-only-Eingabeband und Arbeitsband aufgefasst, dann ist die Platzkomplexität* $O(\log(n))$*. (Andernfalls ist sie* $n + \log(n) = O(n)$*.)*
- *Die Zeitkomplexität von Schritt (b) und (d) ist O(1), von Schritt (c)* $O(n)$*.*

  In Schritt (e) wird n *mal etwas wiederholt. Dabei wird jeweils maximal 2 mal der Zähler der Länge* $\leq \log(n)$ *durchfahren.*

  Insgesamt sind dies also $O(n \log(n))$ *Schritte.*

d) Wie verändert sich die Zeit- und Platzkomplexität der Maschine aus Teil b), wenn man sie in der üblichen Weise in eine 1-Band-Maschine wandelt?

Vgl. Theorem 7.8: Für $t(n) \geq n$ *wird bei der Zeitkomplexität aus* $O(t(n))$ *dann* $O(t^2(n))$*, d.h. hier* $O(n^2 \log(n)^2)$*. Bei nur einem Band stehen Ein- und Ausgabe auf einem Band durch ein Trennzeichen getrennt, als Platzkomplexität ergibt sich die Summe beider Bänder wie oben bereits summiert.*

4. Aufgabe *(Berechenbarkeit)* $(5 + 9 + 4 + 12 = 30$ Punkte)

a) Definieren Sie folgende Begriffe:

- entscheidbar (englisch "decidable"),

- abzählbar (englisch "countable").

Definitionen:

- *Eine Sprache A heißt entscheidbar, wenn es eine Turingmaschine M gibt mit $A = L(M)$ und $\bar{A} = L_{reject}(M)$.*
  Oder: *... , M gibt, die immer anhält und für die $A = L(M)$.*
- *Eine Menge A heißt abzählbar, wenn sie endlich oder gleichmächtig mit $\mathbb{N}$ ist.*

b) Hieße "entscheidbar" dasselbe, wenn man es, statt mittels Turingmaschinen, mittels folgenden Maschinen- bzw. Algorithmenklassen definieren würde:

- C-Programme?
- Kellerautomaten?
- Registermaschinen (d.h. Prozessoren, erweitert um potentiell unendlichen Speicher)?

Bitte jeweils in einem Satz grob begründen.

- *Ja. Man kann Turingmaschinen in C simulieren. (Siehe z.B. praktische Übungsaufgabe.) Umgekehrt kann man die C-Programme auf Registermaschinen ausführen, und diese wieder auf Turingmaschinen simulieren, siehe 3. Punkt.* ***Anm.:*** *Auch ok, aber kein inhaltlicher Grund: "Alle normalen Programmiersprachen haben nach der Church'schen These dieselbe Ausdrucksfähigkeit".*
- *Nein. Z.B. ist die Sprache L_1 aus Aufgabe 2 nicht kontextfrei, also nicht von Kellerautomaten akzeptierbar, aber offensichtlich entscheidbar.*
- *Ja. In der Vorlesung wurde explizit gezeigt, wie man Registermaschinen auf Turingmaschinen simuliert, und mit Punkt 1 erhält man einen Ringschluss gegenseitiger Simulation.*

c) Geben Sie ein unentscheidbares Problem aus der Vorlesung an, d.h. Name und Definition. (Ohne Begründung.)

Z.B. das Wortproblem für Turingmaschinen:

$$A_{TM} := \{\langle M, w\rangle \mid M \text{ ist Turingmaschine, } w \text{ Wort, } w \in L(M)\}.$$

d) Ist folgendes Problem entscheidbar? (Mit Beweis.)

$$\begin{aligned} LEQ_{TM} \quad = \quad & \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ Turingmaschinen und} \\ & \forall n \in \mathbb{N} : \big(f_{M_1}(n) \le f_{M_2}(n) \text{ oder } M_1, M_2 \text{ halten auf } n \text{ beide nicht}\big)\}. \end{aligned}$$

Nein. Nehmen wir an, eine Turingmaschine M_{LEQ} entscheide es doch.

Lösung 1: *Wir zeigen einen Widerspruch zum Satz von Rice. Dieser besagt, dass keine nichttriviale Eigenschaft der von einer Turingmaschine berechneten Funktion entscheidbar ist. Allerdings haben wir hier* zwei *Turingmaschinen. Daher gehen wir so vor:*

Wir definieren folgendes Problem:

$$LEQ\_ZERO_{TM} = \{\langle M\rangle \mid M \text{ ist Turingmaschine und } \forall n \in \mathbb{N} : f_M(n) \le 0\}.$$

Diese könnten wir so entscheiden: Sei M_2 eine feste Maschine, die immer sofort 0 ausgibt.

M_{LEQ_ZERO}: Bei Eingabe $\langle M\rangle$:

(a) Rufe M_{LEQ} mit Eingabe $\langle M, M_2\rangle$ auf.

(b) Akzeptiere, wenn dies akzeptiert.

Diese Entscheidung ist richtig: $M_{LEQ\_ZERO}(\langle M\rangle)$ akzeptiert genau dann, wenn $\forall n : f_M(n) \le f_{M_2}(n) = 0$ (da M_2 immer hält).

Nach dem Satz von Rice ist LEQ_ZERO_{TM} aber unentscheidbar. Widerspruch.

5. Aufgabe *(Überblick über die Sprachklassen)* $(6+4+4+12+4=30$ Punkte$)$

a) Geben Sie die Definitionen der folgenden Sprachklassen an (kurz, aber präzis):

- NP.
- L.
- NPSPACE.

Z.B.:

$$\begin{aligned} NP &= \bigcup_k NTIME(n^k); \\ L &= SPACE(\log(n)); \\ NPSPACE &= \bigcup_k NSPACE(n^k). \end{aligned}$$

Oder: *NP ist die Menge der Sprachen, die von einer nichtdeterministischen Turingmaschinen in polynomialer Zeit entschieden werden, usw.*

b) Zeigen Sie, dass für jede Funktion $t : \mathbb{N} \to \mathbb{N}$ gilt: $TIME(t) \subseteq SPACE(t)$.

Für jedes Problem in $TIME(t)$ gibt es eine Turingmaschine M, die es in $O(t(n))$ Schritten löst. In x Schritten können höchstens $x+1$ Zellen besucht werden, also hat M den Platzverbrauch $O(t(n))$ und das Problem liegt damit in $SPACE(t)$.

c) Geben Sie Savitch's Theorem an.

Vgl. Theorem 8.5: Für jede Funktion $f : \mathbb{N} \to \mathbb{N}$ mit $f(n) \geq n$ (bzw. $\log(n)$) gilt:

$$NSPACE(f(n)) \subseteq SPACE(f^2(n)).$$

d) Geben Sie den Sprachklassen

- P,
- NP,
- EXPTIME,
- L,
- NL,
- NPSPACE,
- PSPACE

mit den Relationen '=', '⊂' und '⊆' eine totale Ordnung (z.B. $A = B \subset C \subseteq D = E \subset F$). Verwenden Sie '⊆' nur, wenn nicht bewiesen ist, ob '=' oder '⊂' gilt.

Geben Sie zu jeder Relation einen Satz Begründung an, oder z.B. den Namen eines Theorems. (Weshalb nichts Stärkeres bewiesen ist, müssen Sie natürlich nicht begründen.)

Vgl. Kap. 8.2 und 8.6:

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME.$$

Begründung:

(a) *1. und 3.: Deterministische Maschinen sind ein Spezialfall von nichtdeterministischen.*

(b) *2.: Explizit in Kap. 8 bewiesen. (PATH NL-vollständig und klar in P.)*

(c) *5.: Savitch*

(d) *4.: $NP \subseteq NPSPACE$ analog Teil b). Dann folgt es mit 5.*

(e) *6.: $PSPACE \subseteq EXPTIME$ da nur exponentiell mehr Rechenzeit als Platz. (Bzw. "Lemma 8.0".) Dann folgt es mit 5.*

e) Welche Art von Komplexität wird für kryptographische Probleme benötigt? (Etwa einen Satz und eine Formel schreiben.)

Ein Problem sollte fast überall schwer sein, d.h. außer für einen vernachlässigbar kleinen Anteil der Eingaben. 'Vernachlässigbar klein' heißt, ein Angreifer soll mit Wahrscheinlichkeit $P(l) \leq \frac{1}{poly(l)}$ *Erfolg haben.*

Anm. *(volle Punktzahl auch ohne dies): Dabei heißt* $P(l) \leq \frac{1}{poly(l)}$*, dass für alle Polynome* Q *ein* n_0 *existiert, so dass* $\forall n > n_0 : P(l) \leq 1/Q(k)$*.* l *ist ein sogenannter Sicherheitsparameter.*

6. Aufgabe *(NP-Vollständigkeit)* $((3+6+6)+(4+11) = 30$ Punkte)

a) Seien L_1 und L_2 Sprachen über Σ.

i) Geben Sie die formale Definition für polynomielle Reduktion von L_1 nach L_2 an.

Siehe Def. 7.24:
Eine Sprache L_1 *heißt polynomzeit-reduzierbar auf eine Sprache* L_2 $(L_1 \leq_P L_2)$*, wenn eine in polynomieller Zeit berechenbare Funktion* $f : \Sigma^* \to \Sigma^*$ *existiert, so dass* $\forall w \in \Sigma^*$ *gilt:*

$$w \in L_1 \;\Leftrightarrow\; f(w) \in L_2.$$

ii) Es gilt $L_1 \leq_P L_2$ und L_2 sei lösbar in Rechenzeit $t(n)$ für alle Eingaben der Länge n. Welche Aussage können Sie über die Rechenzeit von L_1 machen?

Sei $p(n)$ *die Rechenzeit für die Umwandlungsfunktion* f *(siehe Teil i)), und sei* M_2 *die Maschine, die* L_2 *entscheidet.*
Wenn $|w| = n$*, dann folgt* $|f(w)| \leq p(|w|)$*. Die Laufzeit von* M_2 *auf* $f(w)$ *ist also* $\leq t(p(n))$*. Insgesamt ergibt sich eine Laufzeit* $\leq p(n) + t(p(n))$*; i.allg. ist das* $O(t(p(n)))$*.*
Anm.: Es könnte auch eine effizientere Lösung existieren als die mittels L_2*, deshalb* $\leq$*.*
Achtung: *An die Verlängerung der Eingabe hatte fast niemand gedacht!*

iii) Sei L_1 NP-vollständig; beweisen Sie:

$$L_1 \notin P \;\Rightarrow\; \forall L_2 \text{ NP-vollständig: } L_2 \notin P.$$

(Hinweis: Setzen Sie folgenden Satz als bewiesen voraus: $L_1 \leq_p L_2 \wedge L_2 \in P \Rightarrow L_1 \in P$.)

Annahme: Es existiert ein NP-vollständiges L_2 *mit* $L_2 \in P$*. Nach Definition der NP-Vollständigkeit folgt* $L_1 \leq_P L_2$*, da* $L_1 \in NP$*. Mit dem Hinweis gilt dann auch* $L_1 \in P$*, im Widerspruch zur Voraussetzung.*

b) Herr Schlau ist Professor für Informatik an der Universität Obermeierhof. In seinem Institut arbeiten k wissenschaftliche MitarbeiterInnen und m Bremser. Für die anstehende Klausur bereitet Herr Schlau eine Aufgabenliste $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ aus n verschiedenen Aufgaben vor. Dann legt er für jede Aufgabe a_i mit der Abbildung $p : \mathcal{A} \to \mathbb{N}$ die Punktzahl $p(a_i)$ fest. Nun stellt er seine Mitarbeiter vor folgende Probleme:

Π_1: Aus der Aufgabenliste $\mathcal{A}$ soll eine Klausur $\mathcal{K}$ mit Punktzahl $B \in \mathbb{N}$ zusammengestellt werden.

Π_2: Um potentielle Täuschungsversuche zu verringern, sollen die Klausurteilnehmer abwechselnd verschiedene Klausuren $\mathcal{K}_1$ bzw. $\mathcal{K}_2$ erhalten. Dazu soll die Aufgabenliste $\mathcal{A}$ vollständig in 2 komplett verschiedene Klausuren $\mathcal{K}_1$ und $\mathcal{K}_2$ mit gleicher Punktzahl aufgeteilt werden.

Beantworten Sie nun folgende Fragen:

i) Beschreiben Sie die Entscheidungsvarianten der Probleme Π_1 und Π_2 formal. (D.h. in der Form $\Pi_i = \{\langle \ldots \rangle \mid \ldots\}$.)

$$\Pi_1 \;=\; \{\langle \mathcal{A}, p, B\rangle \;\mid\; \exists \mathcal{A}^* \subseteq \mathcal{A} \textit{ mit } \sum_{a \in \mathcal{A}^*} p(a) = B\}.$$

$$\Pi_2 \;=\; \{\langle \mathcal{A}, p\rangle \;\mid\; \exists \mathcal{A}^* \subseteq \mathcal{A} \textit{ mit } \sum_{a \in \mathcal{A}^*} p(a) = \sum_{a \notin \mathcal{A}^*} p(a)\}.$$

Noch besser ist es, wenn in der Bedingung auch die Wertebereiche angegeben sind: $\mathcal{A}$ endliche Menge, $p : \mathcal{A} \to \mathbb{N}$, $B \in \mathbb{N}$.

Achtung: *Viele von Ihnen haben statt dessen eine Menge von $\mathcal{A}^*$ betrachtet. Intuitiv müssen aber links von "$|$" immer Eingaben für den gesuchten Algorithmus stehen und rechts die Lösbarkeitsbedingung.*

Anm.: *Π_1 ist genau das Subset-Sum-Problem (Theorem 7.21) und Π_2 das Partition-Problem (Übung 11.2).*

ii) Nehmen Sie an, Π_1 sei NP-vollständig. Zeigen Sie, dass dann auch Π_2 NP-vollständig ist.

Teil 1: *Zu zeigen $\Pi_2 \in NP$.*
Wir können zwei Klausuren (d.h. eine Menge $\mathcal{A}^* \subseteq \mathcal{A}$) raten und deren Korrektheit in polynomieller Zeit verifizieren. (Summen ausrechnen und vergleichen; die einzelnen $p(a)$ sind explizit Teile der Eingabe.)

Teil 2: *Zu zeigen $\Pi_1 \leq_P \Pi_2$. (Dann ist nach einem Satz aus der Vorlesung auch Π_2 NP-hart.)*

1) *Konstruktion der Funktion f (mit den Eigenschaften aus Teil a iii)): Sei $\pi_1 = \langle \mathcal{A}_1, p, B \rangle$ eine Probleminstanz für Π_1 mit $\mathcal{A}_1 = \{a_1, \ldots, a_n\}$.*
Idee: *Wenn es eine Teilmenge $\mathcal{A}_1^*$ gibt, die sich zu B summiert, hat der Rest die Summe $S - B$, wenn*
$$S := \sum_{a \in \mathcal{A}_1} p(a)$$
die Gesamtsumme ist. Um auf ein Problem mit einer Zerlegung in zwei gleich bepunktete Mengen zu kommen, ergänzen wir beide Hälften um je ein Element, so dass beide Hälften die Summe $S + 1$ bekommen. (Das ist die kleinste Möglichkeit, wo die Ergänzungen immer positiv sind.)
Konkrete Definition von $\pi_2 = f(\pi_1)$: *Wir setzen $\mathcal{A}_2 := \mathcal{A}_1 \cup \{a_{n+1}, a_{n+2}\}$ für beliebige Strings a_{n+1}, a_{n+2} (Hilfsaufgaben) und*
$$\begin{aligned} p(a_{n+1}) &= S - B + 1, \\ p(a_{n+2}) &= B + 1. \end{aligned}$$
Damit ist $\pi_2 = \langle \mathcal{A}_2, p \rangle$. (Da die Abbildung p nur erweitert wird, ist es ok, denselben Namen zu verwenden.)
2) *Offensichtlich ist f in polynomieller Zeit berechenbar.*
3) *Beh.: $\pi_1 \in \Pi_1 \Rightarrow \pi_2 \in \Pi_2$.*
Sei $\mathcal{A}_1^* \subseteq \mathcal{A}_1$ eine Lösung für π_1, also $\sum_{a \in \mathcal{A}_1^*} p(a) = B$. Dann ist $\mathcal{A}_2^* = \mathcal{A}_1^* \cup \{a_{n+1}\}$ eine Lösung für π_2, denn
$$\sum_{a \in \mathcal{A}_2^*} p(a) = \sum_{a \in \mathcal{A}_1^*} p(a) + p(a_{n+1}) = B + (S - B + 1) = S + 1$$
und
$$\sum_{a \notin \mathcal{A}_2^*} p(a) = \sum_{a \in \mathcal{A}_1 \setminus \mathcal{A}_1^*} p(a) + p(a_{n+2}) = (S - B) + B + 1 = S + 1.$$
4) *Beh.: $\pi_2 \in \Pi_2 \Rightarrow \pi_1 \in \Pi_1$.*
Sei $\mathcal{A}_2^* \subseteq \mathcal{A}_2$ eine Lösung für π_2. Es gilt $S_2 := \sum_{a \in \mathcal{A}_2} p(a) = 2S+2$, also $\sum_{a \in \mathcal{A}_2^*} p(a) = S + 1$. Damit kann nur eins von a_{n+1} und a_{n+2} in $\mathcal{A}_2^*$ liegen, da deren Summe $S + 2$ ist.
ObdA sei das a_{n+1} (sonst vertausche $\mathcal{A}_2^*$ und sein Komplement). Dann ist $\mathcal{A}_1^* = \mathcal{A}_2^* \setminus \{a_{n+1}\}$ eine Lösung für π_1. (Noch genauer kann man das wie in Teil 3) nachrechnen.)