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

—MIT MUSTERLÖSUNG—
Nachklausur zu
Automaten, Berechenbarkeit und Komplexitätstheorie (Informatik 3)
(WS 2000/2001)

Name, Vorname : ....

Matrikelnummer : ....

Zur Nachklausur:

- Es sind **keine Hilfsmittel** zugelassen.
- Benutzen Sie nur das von uns bereit gestellte Papier mit Ihrem Namen – auch als Schmierblätter (bitte gesondert kennzeichnen!). Sollte das Papier nicht ausreichen, lassen Sie sich von der Aufsicht weiteres geben und kennzeichnen Sie es bitte auch mit Namen und Matrikelnummer.
- Achten Sie auf lesbare Schrift, verständliche Begründungen, formale Definitionen gemäß Vorlesung (Sipser) und korrekte Beschriftung mit Aufgabennummern.
- Legen Sie Personal– und Studentenausweis offen vor sich hin.
- Bei vorübergehendem Verlassen des Raumes sind alle Unterlagen bei der Aufsicht abzugeben.
- Als Anhaltspunkt zur Bearbeitung sollte dienen, dass 1 Klausurpunkt etwa 1 Bearbeitungsminute entspricht.

Bei der Abgabe:

- Bitte geben Sie alle auf Ihrem Tisch befindlichen Blätter ab (inkl. Schmierblätter).
- Lassen Sie Aufgabenblatt und alle beschriebenen Blätter von der Aufsicht heften!

Unterschreiben Sie hier: **Viel Erfolg!**

....

Aufgabe	1	2	3	4	5	6	Σ
Punkte							
Kürzel							

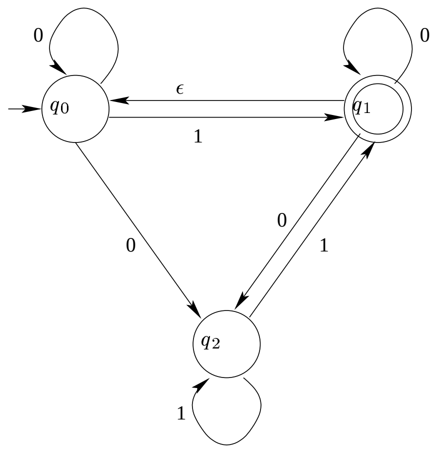
1. Aufgabe *(Endliche Automaten und reguläre Ausdrücke)* $(4+4+10+5+7=30$ Punkte$)$

a) Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein deterministischer endlicher Automat. Wie ist definiert, dass M ein Wort w akzeptiert?

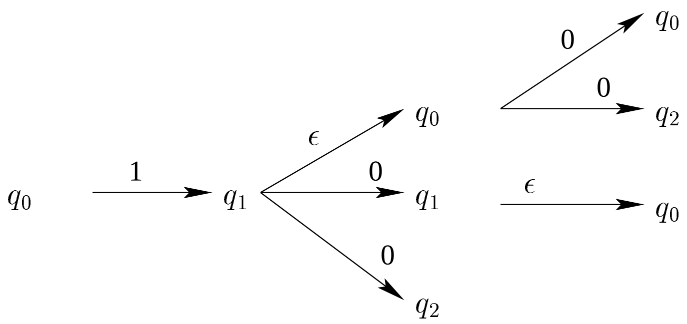
Sipser S. 40: Sei $w = w_1 \cdots w_n$. Es wird akzeptiert, wenn eine Folge von Zuständen $r_0, \ldots, r_n \in Q$ existiert mit

- *$r_0 = q_0$ und*
- *$\delta(r_i, w_{i+1}) = r_{i+1}$ für $i = 0, \ldots, n-1$ und*
- *$r_n \in F$.*

b) Gegeben sei der nichtdeterministische endliche Automat $(\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$ mit δ repräsentiert durch:



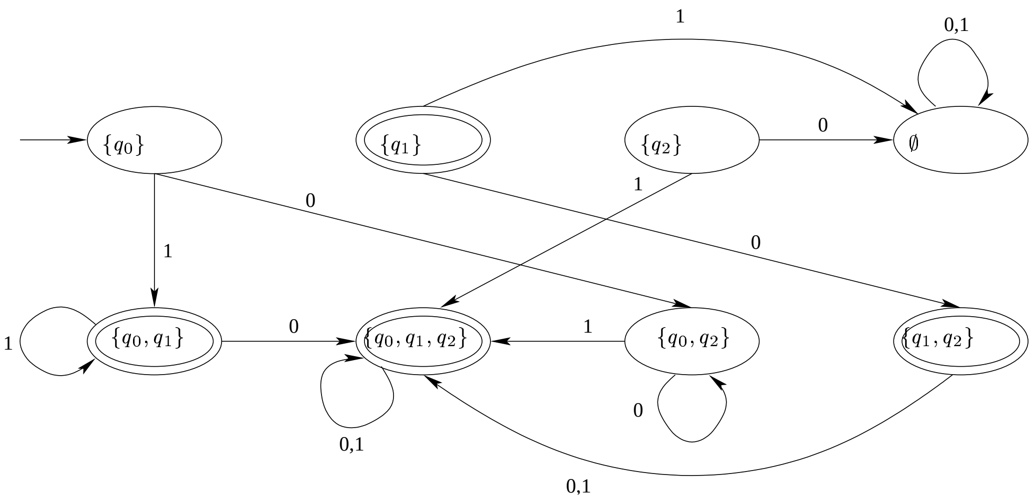
Welche Zustandsfolgen können bei der Eingabe 10 durchlaufen werden? Wird diese Eingabe akzeptiert?



Die Eingabe wird im Endzustand q_1 akzeptiert.

c) Wandeln Sie den NEA aus b) mit der allgemeinen Methode aus der Vorlesung in einen äquivalenten deterministischen Automaten um. Dabei reicht die Angabe des Ergebnis (Bild und Tupel).

Der entsprechende DEA ist $Q' = (\mathcal{P}(Q), \{0, 1\}, \delta', \{q_0\}, F')$ mit δ' repräsentiert durch:



d) Geben Sie einen möglichst einfachen regulären Ausdruck für die reguläre Sprache an, die die Automaten aus Teil b) und c) akzeptieren. (Hinweis: Es gibt tatsächlich einen sehr einfachen.).

$0^*1(0 \vee 1)^*$

e) Zeigen Sie, dass zu jedem nichtdeterministischen endlichen Automaten ein äquivalenter nichtdeterministischer Automat mit nur einem Endzustand existiert. Geben Sie dazu einen entsprechenden NEA an und begründen Sie die Äquivalenz.

Beweisskizze: Gegeben sei ein NEA $M = (Q, \Sigma, \delta, q_0, F)$ *mit* $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$. *Dann gibt es auch einen NEA* $M' = (Q \cup \{q_F\}, \Sigma, \delta', q_0, q_F)$ *mit*

$$\delta'(q,a) = \begin{cases} \delta(q,a) & \textit{falls } (q,a) \in Q \times \Sigma_\epsilon, \\ q_F & \textit{falls } (q,a) \in F \times \{\epsilon\}. \end{cases}.$$

Von diesem NEA M' *müssen wir zeigen, dass er äquivalent zu* M *ist, d.h.* $\forall w \in \Sigma^*$ *gilt:*

$$w \textit{ wird von } M \textit{ akzeptiert} \iff w \textit{ wird von } M' \textit{ akzeptiert}.$$

$\Rightarrow$: *Wenn* w *von* M *akzeptiert wird, gibt es eine Aufteilung von* w *und eine Folge von Zuständen, die die Aufteilung von* w *abarbeitet und zu einem Endzustand in* F *führt. Dann gibt es aber offensichtlich auch eine Aufteilung von* w*, die mit* ϵ *endet, sonst aber identisch zur vorigen Aufteilung ist. Die Abarbeitung dieser Aufteilung mit* M' *geht über einen Zustand in* F *und führt dann mittels einem* ϵ*-Übergang zu* q_F*, womit* w *auch von* M' *akzeptiert wird.*

$\Leftarrow$: *Wenn* w *von* M' *akzeptiert wird, gibt es eine Aufteilung von* w*, die mit* ϵ *endet (nur so kann man in* q_F *gelangen) und eine Folge von Zuständen in* Q*, die die Aufteilung von* w *ohne* ϵ *abarbeitet, dann in einem Zustand aus* F *ist (nur von dort kann man zu* q_F *gelangen) und dann mittels* ϵ *zu* q_F *führt. Damit wird* w *aber auch von* M *akzeptiert, da die vorige Aufteilung von* w *ohne* ϵ *in einem Zustand aus* F *endet.*

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

a) Geben Sie die formale Definition einer kontextfreien Grammatik (kfG) an. Vergessen Sie dabei evtl. Anforderungen an Bestandteile der kfG nicht.

Sipser Def. 2.1: Eine kontextfreie Grammatik ist ein 4-Tupel (V, Σ, R, S) *mit*

- V *endliche Menge, genannt Variablen.*
- Σ *endliche Menge, genannt Terminale, mit* $V \cap \Sigma = \emptyset$.
- R *endliche Menge von Regeln der Form* $A \to \alpha$ *mit* $A \in V$ *und* $\alpha \in (V \cup \Sigma)^*$.
- $S \in V$ *Startvariable.*

b) Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein deterministischer endlicher Automat, der die Sprache $L = L(M)$ akzeptiert. Konstruieren Sie eine kontextfreie Grammatik G, die die gleiche Sprache L erzeugt.

(Sipser S. 97 oben). $G = (V, \Sigma, R, S)$ *mit*

- $V = Q$ *mit* $V \cap \Sigma = \emptyset$,
- $S = q_0$,
- $R = \{q \to ap \mid a \in \Sigma, q, p \in Q, \delta(q, a) = p\} \cup \{q \to \epsilon \mid q \in F\}$

c) Sind die folgenden Sprachen über $\Sigma = \{0, 1, \#\}$ kontextfrei?

- $L_1 = \{x\#y \mid x, y \in \{0,1\}^* \ \wedge \ x$ Teilstring von $y\}$.
- $L_2 = \{x\#y \mid x, y \in \{0,1\}^* \ \wedge \ x^R$ Teilstring von $y\}$.

Wenn ja, geben Sie die Definition einer kontextfreien Grammatik an, die sie erzeugt. Wenn nein, beweisen Sie die Nichtkontextfreiheit mithilfe des Pumpinglemmas.

- *Beh.:* L_1 *ist nicht kontextfrei.*

  Beweis durch Pumpinglemma (ähnlich Example 2.22):

  Wenn L kontextfrei ist, $\exists$ *Pumpingzahl p. Sei* $w = 0^p1^p\#0^p1^p \in L_1$ *mit* $|w| = 4p + 1 \geq p$. *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 bcd ganz im Bereich vor dem # liegt, dann hat* ab^2cd^2e *mehr Zeichen vor dem # als dahinter, da* $|bd| > 0$. *Damit kann die geforderte Teilstringbeziehung nicht mehr erfüllt sein und* $ab^2cd^2e \notin L_1$.
  - *Wenn bcd ganz im Bereich hinter dem # liegt, dann hat* ab^0cd^0e *weniger Zeichen hinter dem # als davor, da* $|bd| > 0$. *Damit kann die geforderte Teilstringbeziehung nicht mehr erfüllt sein und* $ab^0cd^0e \notin L_1$.

  bcd muss also die Form $0^{p_0}\#1^{p_1}$ *mit* $p_0 + p_1 \leq p - 1$ *haben. Dann tritt einer der folgenden Fälle ein.*

  - *b oder d enthält #, dann enthält* ab^0cd^0e *kein # mehr und* $ab^0cd^0e \notin L_1$.
  - *b und d enthalten beide # nicht, damit hat c die Form* $0^{q_0}\#1^{q_1}$ *mit* $q_0 < p_0$ *oder* $q_1 < p_1$, *da* $|bd| > 0$:
    - *Falls* $q_0 < p_0$, *gilt* $|b| > 0$ *und* ab^2cd^2e *enthält mehr* 0*en vor dem # als dahinter. Damit kann die geforderte Teilstringbeziehung nicht mehr erfüllt sein und* $ab^2cd^2e \notin L_1$.
    - *Falls* $q_1 < p_1$, *gilt* $|d| > 0$ *und* ab^0cd^0e *enthält weniger* 0*en hinter dem # als davor. Damit kann die geforderte Teilstringbeziehung nicht mehr erfüllt sein und* $ab^0cd^0e \notin L_1$.

  Für alle auftretenden Zerlegungen wurde ein Widerspruch zum Pumpinglemma gezeigt, damit ist L_1 *nicht kontextfrei.*

- L_2 *wird von der Grammatik* $G_2 = (V, \Sigma, R, S)$ *mit* $V = \{S, A, B, C\}$ *und*

$$
\begin{aligned}
R = \{S &\to AC \\
A &\to 0A0 \mid 1A1 \mid B \\
B &\to \#C \\
C &\to 0C \mid 1C \mid \epsilon\}
\end{aligned}
$$

  erzeugt.

d) Begründen Sie kurz, dass die Klasse der kontextfreien Sprachen unter der Stern-Operation * abgeschlossen ist. Gegeben eine Sprache L, konstruieren Sie dazu eine kfG, von der Sie vermuten, dass sie L^* erzeugt, und begründen sie dann, dass alle von dieser erzeugten Wörter in L^* liegen und umgekehrt.

Beweis:

Sei L eine kfS, die durch die kfG $G = (V, \Sigma, R, S)$ erzeugt wird. Da wir Variablen umbenennen können, ohne die erzeugte Sprache zu verändern, nehmen wir an, dass $S_* \notin V$. Nun konstruieren wir eine Grammatik $G_* = (V \cup \{S_*\}, \Sigma, R_*, S_*)$ mit $R_* = R \cup \{S_* \to SS_* \mid \epsilon\}$. G_* erzeugt eine Sprache L_*; wir müssen nun $L_* = L^*$ zeigen.

$\subseteq$: *$w = \epsilon \in L^*$ wird auch von G_* mittels $S_* \to \epsilon$ erzeugt. Sei nun $w = w_1 \dots w_n$ mit $n \in \mathbb{N}$ und $w_j \in L\ \forall j$ mit $1 \leq j \leq n$. Damit wurde w_j von einer Ableitung $S \Rightarrow_G^* w_j$ erzeugt. $w_1 \dots w_n$ kann somit aus S^n erzeugt werden. S^n kann von G_* leicht mittels n-facher Verkettung der Ableitung $S_* \to SS_*$ und dann $S_* \to \epsilon$ erzeugt werden. Also gilt $L^* \subseteq L(G_*)$.*

$\supseteq$: *Für die umgekehrte Richtung sei $w \in L_*$. Dann hat w eine Ableitung $S_* \Longrightarrow_{G_*}^* w$, für die es folgende Möglichkeiten gibt:*

- *Sie ist $S_* \to \epsilon$ und $w = \epsilon \in L^*$.*
- *Sie beginnt mit $S_* \to SS_*$. Aus S_* können nur weitere S oder ϵ erzeugt werden. Dadurch ergibt sich die Ableitung $S \Rightarrow_{G^*}^* S^i$ mit $i \in \mathbb{N}$. Aus S wiederum können nur Wörter aus L erzeugt werden. Dies führt zu $S \Rightarrow_{G^*}^* S^i \Rightarrow_G^* w_1 \dots w_n$ mit $w_j \in L$ und damit hat w die Form $w = w_1 \dots w_n$ und $w \in L^*$.*

Also gilt $L(G_*) \subseteq L^*$.

Damit gilt $L(G_*) = L^*$.

3. Aufgabe *(Turingmaschinen)* $(8 + 3 + 6 + 13 = 30$ Punkte)

a) Gegeben sei $L = \{a^i b^j \mid i \geq j \geq 0\}$ über $\Sigma = \{a, b\}$. Konstruieren Sie eine 1-Band-TM, die L akzeptiert: Geben Sie dazu Ihre Idee in einem Satz an und beschreiben Sie Ihre Vorgehensweise dann detailliert in Pseudocode in mehreren Schritten.

Idee ist abwechselndes Markieren der a und b und Kontrolle, von welchem zuerst alle markiert sind.

(a) Prüfe, ob die Eingabe auf dem Band die Form $a^i b^j$ hat: Fahre nach rechts bis zum 1. Zeichen $\neq a$. Falls das nicht b oder Blank ist, lehne ab. Dann fahre bis zum 1. Zeichen $\neq b$. Falls das nicht Blank ist, lehne ab. Fahre zurück.

(b) Solange noch a's und b's auf dem Band unmarkiert sind:

1) Markiere das erste unmarkierte a,

2) Laufe zum ersten unmarkierten b und markiere es.

3) Laufe zurück zum am weitesten links stehenden unmarkierten a.

(c) Wenn vor Markierung aller b's alle a's markiert sind, lehne Wort ab.

(d) Sonst sind noch a's unmarkiert und Wort wird akzeptiert.

b) Analysieren Sie kurz die Zeitkomplexität Ihrer Maschine aus Teil a) und geben Sie das Ergebnis in möglichst genauer O-Notation an.

Bei jedem Markiervorgang werden jeweils ein a und ein b markiert, dazu werden $O(n)$ Zeichen durchlaufen. Insgesamt werden $O(\min(i,j)) = O(n)$ solcher Durchgänge benötigt (dann ist eines der beiden Zeichen komplett markiert), womit die Gesamtlaufzeit sich als $O(n^2)$ ergibt.

c) Eine 2-dimensionale 1-Feld-TM ist eine TM mit einem Feld anstelle eines Bandes. Das Feld ist nach oben und links beschränkt, aber nach unten und rechts unbeschränkt. Auf dem Feld operiert analog zur 1-Band-TM ein Kopf, der abhängig von der Konfiguration der TM nach rechts, links, oben und unten wandern und Zeichen verändern kann.
Definieren Sie 2-dimensionale 1-Feld-TM formal analog zur bekannten Definition von 1-Band-TM aus der Vorlesung.

Eine 2-dimensionale 1-Feld-TM 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 $\Sigma \subset \Gamma$,*
- *$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, O, U\}$ Übergangsfunktion.*
- *$q_0 \in Q$ Startzustand,*
- *$q_{accept} \in Q$ akzeptierender Endzustand,*
- *$q_{reject} \in Q$ ablehnender Endzustand mit $q_{reject} \neq q_{accept}$.*

d) Zeigen Sie, dass jede 2-dimensionale 1-Feld-TM eine äquivalente k-Band-TM für geeignetes aber festes k besitzt. (Ohne Benutzung der Church'schen These.) Gibt es auch eine äquivalente 1-Band-TM?

Sei $M_F = (Q_F, \Sigma, \Gamma_F, \delta_F, qF_0, qF_{accept}, qF_{reject})$ die 1-Feld-TM. Wir konstruieren eine äquivalente 3-Band-TM $M_B = (Q_B, \Sigma, \Gamma_B^3, \delta_B, qB_0, qB_{accept}, qB_{reject})$.

***Idee:** Auf dem Feld von M_B stehen zu jedem Zeitpunkt jeweils nur in m Zeilen und n Spalten von $\_$ verschiedene Symbole. Wir betrachten dieses $(m \times n)$-Teilfeld*

a_{11}	$\cdots$	a_{1n}
$\vdots$	$\ddots$	$\vdots$
a_{m1}	$\cdots$	a_{mn}

(mit $a_{ij} \in \Gamma$ für $1 \leq i \leq m$, $1 \leq j \leq n$). (Manche a_{ij} können dabei $\_$ sein.) Auf dem ersten Band der 2-Band-TM repräsentieren wir das obige Feld als

$$\star\star\, a_{11} \cdots a_{1n} \star \cdots \star\, a_{m1} \cdots a_{mn} \star\star.$$

D.h., ein $\star$ symbolisiert einen Zeilenwechsel und $\star\star$ Anfang und Ende des benutzten Teilfeldes.

Einzelheiten:

- *Es ist $\Gamma_B := \Gamma_F \cup \{\star\}$, wobei oBdA $\star \notin \Gamma_F$.*
- *Zu Beginn bringt M_B seine Eingabe in passende Form. Wir nehmen an, dass Eingaben von M_F in der 1. Zeile des Feldes stehen. M_B hängt also an seine Eingabe einfach $\star\star$ an und fügt durch Shiften vorne (Makro aus der Vorlesung) ebenfalls $\star\star$ ein.*
- *Eine horizontale Bewegung von M_F im $(m \times n)$-Teilfeld simuliert M_B durch einfache Links- oder Rechtsbewegung.*
- *Für eine vertikale Bewegung auf dem Feld von M_F muss der Kopf von M_B auf dem ersten Band hingegen um $n + 1$ Stellen nach rechts bzw. links bewegt werden. Zum Zählen der Bewegung wird das 2. Band von M_B benutzt. Auf dem dritten Band stehen zum Vergleich die aktuellen Größen des benutzten Teilfeldes m und n.*
- *Stößt der Bandkopf von M_B bei einer der obigen Links- oder Oben-Bewegungen auf ein $\star$ auf dem Band, so bewegt er den Kopf wieder nach rechts bzw. unten, da in diesem Fall, der Kopf von M_F an seinen Rand stößt und nach Definition unserer TM keine Bewegung ausführt, sondern stehenbleibt.*
- *Bewegt sich der Kopf von M_F nach unten aus dem $(m \times n)$-Teilfeld hinaus, so muss M_B eine Zeile hinzufügen: Es merkt das am Überfahren von $\star\star$. Auf dem 1. Band ersetzt es $\star\star$ durch $\star a_{(m+1),1} \cdots a_{(m+1),n} \star\star$. Dabei sind alle $a_{(m+1),i} = \_$. Anschlieŝend muss er den Bandkopf wieder auf die Stelle zurückbewegen, an der er sich aus seiner Darstellung des Teilfeldes herausbewegt hatte. Die entsprechende Stelle wird auf dem 2. Band mitgezählt.*
- *Bewegt sich der Kopf von M_F nach rechts aus dem $(m \times n)$-Teilfeld hinaus, so muss M_B eine Spalte einfügen, d.h. in jeder Zeile ein Blank anhängen. Es bemerkt die Situation, indem es beim Fahren nach rechts auf $\star$ trifft. Dann verschiebt es vor jedem $\star$ (außer dem zweiten in $\star\star$) den Rest seines Bandes eins nach rechts. (Makro aus der Vorlesung.) Anschlieŝend muss er den Bandkopf wieder auf die Stelle zurückbewegen, an der er sich aus seiner Darstellung des Teilfeldes herausbewegt hatte. Die entsprechende Stelle wird auf dem 2. Band mitgezählt.*
- *Da jede k-Band-TM eine äquivalente 1-Band-TM besitzt, hat auch die 1-Feld-TM eine äquivalente 1-Band-TM.*

4. Aufgabe *(Berechenbarkeit)* (5 + 5 + 6 + 4 + 10 = 30 Punkte)

a) Definieren Sie folgende Begriffe:

- semi-entscheidbar (englisch "Turing-recognizable"),
- berechenbar.

- *Eine Sprache A heißt semi-entscheidbar, wenn es eine Turingmaschine M gibt mit $A = L(M)$.*
- *(Definition 5.12.) Eine Funktion $f : \Sigma^* \to \Sigma^*$ heißt berechenbar, wenn es eine Turingmaschine M gibt, die bei jeder Eingabe w mit dem Ergebnis $f(w)$ auf dem Band hält.*

b) Gibt es unentscheidbare Probleme, die in

- P
- NP

liegen? (Je 1 Satz Begründung.)

- *P: Nein. Dafür muss M sogar in polynomialer Zeit entscheiden.*
- *NP: Nein, dann existiert eine nichtdeterministische Maschine M, die das Problem entscheidet, und bei Berechenbarkeit macht Nichtdeterminismus keinen Unterschied. (Theorem 3.10.)*

c) Geben Sie den Satz von Rice an. (Möglichst präzis.)

Informell: Bei Eingabe einer Turingmaschine ist keine nichttriviale Eigenschaft der davon akzeptierten Sprache entscheidbar. (Oder auch "der berechneten Funktion").

Genauer: Für alle nichttrivialen Eigenschaften $E \subseteq \mathcal{P}(\Sigma^*)$ ist die Menge

$$RICE_E := \{\langle M\rangle | M \textit{ ist } TM \land L(M) \in E\}$$

unentscheidbar. Dabei heißt E nichttrivial, wenn in E und $\bar{E}$ je mindestens eine semi-entscheidbare Sprache liegt.

d) Was bedeutet Turing-Reduzierbarkeit (in Zeichen $\leq_T$)?

Für Sprachen A, B gilt $A \leq_T B$, wenn A relativ zu B entscheidbar ist, d.h. wenn es eine Orakel-Turingmaschine gibt, die A entscheidet, wenn sie ein Orakel für B bekommt. Eine Orakelmaschine entspricht allgemeinen Unterprogrammaufrufen, d.h. sie darf beliebig viele Anfragen w an das Orakel stellen (d.h. auf ein spezielles Band schreiben). Im nächsten Schritt steht dort jeweils die korrekte Lösung, d.h. ob $w \in B$.

e) Ist das Postsche Korrespondenzproblem mit 3-reihigen Dominosteinen entscheidbar? Formal:

$$\begin{aligned} \mathit{3PCP} \;=\; & \{k \in \mathbb{N}, (a_1, b_1, c_1), \ldots, (a_k, b_k, c_k) \in ((\Sigma^*)^3)^k \\ & \mid \exists l \in \mathbb{N}, (i_1, \ldots, i_l) \in \mathbb{N}^l : a_{i_1} \cdots a_{i_l} = b_{i_1} \cdots b_{i_l} = c_{i_1} \cdots c_{i_l}\}. \end{aligned}$$

Nein.

Idee: *Das normale, 2-reihige PCP ist ein Spezialfall hiervon, und das ist schon unentscheidbar.*

Genauer: *Wir nehmen an, eine Turingmaschine M_3 entscheide 3PCP. Wir konstruieren eine Turingmaschine M_2, die PCP entscheiden soll:*

M_2: Bei Eingabe $k \in \mathbb{N}, (a_1, b_1), \ldots, (a_k, b_k) \in ((\Sigma^*)^2)^k$:

- *Setze $c_i := b_i$ für $i := 1, \ldots, k$.*
- *Rufe M_3 mit der Eingabe $(k, (a_1, b_1, c_1), \ldots, (a_k, b_k, c_k))$ auf.*

- *Akzeptiere genau dann, wenn M_3 akzeptiert.*

Offensichtlich ist M_2 eine immer haltende Turingmaschine. Wir müssen noch zeigen, dass ihre Entscheidung richtig ist:

a) Wenn $(k, (a_1, b_1), \ldots, (a_k, b_k))$ lösbar ist, dann auch $(k, (a_1, b_1, c_1), \ldots, (a_k, b_k, c_k))$, denn die Lösung $l, (i_1, \ldots, i_l)$ erfüllt auch $b_{i_1} \cdots b_{i_l} = c_{i_1} \cdots c_{i_l}$ nach Konstruktion der c_i.

b) Jede Lösung $l, (i_1, \ldots, i_l)$ von $(k, (a_1, b_1, c_1), \ldots, (a_k, b_k, c_k))$ ist auch eine von $(k, (a_1, b_1), \ldots, (a_k, b_k))$, denn die Bedingung hierfür ist ein Spezialfall der anderen.

5. Aufgabe *(Überblick über die Sprachklassen)* $(4 + 6 + 6 + 4 \cdot 3 + 2 = 30$ Punkte$)$

a) Geben Sie die formalen Definitionen an, wann eine Sprache NP-hart bzw. NP-vollständig ist.

Siehe Def. 7.27: Eine Sprache B heißt NP-hart, wenn $\forall A \in NP$ gilt $A \leq_P B$.

Sie heißt NP-vollständig, wenn sie in NP liegt und NP-hart ist.

b) Zeichnen Sie ein Mengendiagramm für die Klassen P, NP, und *NP-VOLLSTÄNDIG* beim derzeitigen Wissensstand. Begründen Sie alle 3 Einordnungen kurz.

(Bild 7.4 aus den zusätzlichen Notizen.): Kreis für NP außen, darin disjunkte Kreise für P und NP-VOLLSTÄNDIG.

- *$P \subseteq NP$, da deterministische Maschinen auch nichtdeterministisch.*
- *NP-VOLLSTÄNDIG $\subseteq NP$ nach Definition (siehe Teil a).*
- *Der Schnitt von P und NP-VOLLSTÄNDIG ist leer, außer wenn $P = NP$ wäre.*

c) Sind die Definitionen in Teil a) davon abhängig, ob man als Maschinenmodell Ein- oder Mehrbandmaschinen verwendet? (Mit Begründung.)

Nein.

Jedes auf einer (deterministischen) Mehrbandmaschine in Zeit $t(n)$ lösbare Problem ist auf Einbandmaschinen in Zeit $t(n)^2$ lösbar (Theorem 7.8). Wenn $t(n)$ ein Polynom ist, dann auch $t(n)^2$. (Die Umkehrung ist klar, Einbandmaschinen sind Spezialfall von Mehrbandmaschinen.)

Das Gleichbleiben der Definition von NP folgt nun direkt in der Definition mit polynomialen (deterministischen) Verifizierern. Auch folgt, dass die Definition von $A \leq_P B$ gleich bleibt. Damit bleiben dann auch die obigen Definitionen gleich, die ja nur mit NP und $\leq_P$ ausgedrückt sind.

d) Ordnen Sie die folgenden Sprachen/Probleme mit einer kurzen Begründung in die kleinstmögliche Klasse ein. (Intuitiv also: was ist die kleinstmögliche Komplexität bzw. der einfachste mögliche Maschinentyp?)

i) Gegeben ist die Bevölkerung eines Ortes mit Angaben, wer wen kennt, und eine bestimmte Person P, von der Sie etwas wollen. Gibt es eine Kette von Personen von Ihnen zu dieser Person, d.h. Sie kennen jemand, der jemand kennt, der $\ldots$, der P kennt?

Dies ist das Problem $PATH$ aus der Vorlesung (Erreichbarkeit oder Pfadsuche in Graphen. Es liegt in NL. (Es liegt erst recht in P.)

ii) 400 Erstsemester sollen ein SML-Programm für den ggT schreiben. Sie sind Bremser. Statt das alles zu korrigieren, möchten Sie ein Programm schreiben, das die (digital abgegebenen) Lösungen prüft.

Das Problem ist unentscheidbar, denn es fällt unter den Satz von Rice: Die Eigenschaft ist, die Funktion "ggT" zu berechnen.

iii) Sie schreiben einen Text in Latex. Darin haben Sie eine Variable a verwendet. Diese wollen Sie in b umbenennen, d.h. Sie wollen alle "a" in Formel-Umgebungen durch "b" ersetzen. Nehmen Sie vereinfacht an, Formel-Umgebungen in Latex würden immer mit "\$" begonnen und beendet, und ansonsten kämen keine "\$" vor.

Dies ist mit einem endlichen Automaten (Transduktor) lösbar. (Man braucht nur Zustände für in Formel / nicht in Formel.)

iv) Das Leitungsnetz einer großen Firma soll ständig auf physische Schäden überprüft werden. Dazu sollen Leitungsprüfer in einige Knotenpunkte des Netzes eingebaut werden. Ein solches Gerät kann alle an diesem Knoten angeschlossenen Leitungen prüfen. Sie sollen vorweg entscheiden, ob eine bestimmte Anzahl x von Geräten genügt.

Dies ist genau das Problem $VERTEX - COVER$ aus der Vorlesung. Es ist NP-vollständig.

e) Was ist die Klasse BPP?

(Sipser Def. 10.4). Es ist die Klasse der Probleme, die man mit einem probabilistischen polynomialen Algorithmus mit Fehlerwahrscheinlichkeit $\leq 1/3$ lösen kann.

6. Aufgabe *(Platzkomplexität)* $((6+2)+8+11+3 = 30$ Punkte)

a) Sei M eine immer haltende 1-Band Turingmaschine mit Bandalphabet Γ, Zustandsmenge Q und Platzbedarf $f(n)$ für eine Eingabe w der Länge n.

i) Berechnen Sie eine möglichst genaue Zeitschranke für M (als Funktion von n).

ii) Leiten Sie daraus eine Schranke in möglichst genauer O-Notation ab.

i) Die Zeitschranke ergibt sich als die Anzahl C möglicher Konfigurationen von M: Da T immer hält, kann sich keine Konfiguration wiederholen, sonst würde T in eine Endlosschleife geraten.

- *Es gibt $|Q|$ mögliche Zustände.*
- *Für die Kopfposition gibt es $f(n)$ Möglichkeiten.*
- *Jedes der $f(n)$ eventuell benutzten Bandfelder kann jedes der $|\Gamma|$ Zeichen enthalten.*

Also gilt $C = |Q| f(n) |\Gamma|^{f(n)}$.

ii) Da $|Q|$ und $|\Gamma|$ Konstanten sind, folgt $C \leq f(n) 2^{O(f(n))} \leq 2^{O(f(n))}$. Der letzte Schritt gilt, weil immer $f(n) < 2^{f(n)}$ (für $f(n) > 0$).

b) Betrachten Sie folgendes 2-Personen-Spiel:

$\mathbf{G}_{flip}$**:** Gegeben sei eine Formel ϕ mit n Variablen $x_1, x_2, \ldots, x_n$ und $\phi[0, \ldots, 0] = 0$, $\phi[1, \ldots, 1] = 1$. Sie ist in 2DNF-Form, d.h. sie ist eine ODER-Verknüpfung von Teilformeln, und jede Teilformel ist eine UND-Verknüpfung von höchstens 2 Literalen (Bsp. $(x_1 \wedge x_2) \vee (x_2 \wedge \overline{x}_1) \vee x_3)$.
Alle Variablen von ϕ sind zunächst auf 0 gesetzt. Die Spieler flippen abwechselnd je eine Variable ihrer Wahl auf 1. Wer zuerst die Formel ϕ wahr macht, verliert.

Das zugehörige Entscheidungsproblem ist, ob der anfangende Spieler I eine Gewinnstrategie hat. Es heiße auch G_{flip}. Entwerfen Sie einen PSPACE-Algorithmus zum Entscheiden von G_{flip}.

Hinweise: Sie müssen die Korrektheit nicht begründen, aber die PSPACE-Eigenschaft. Seien Sie nicht verwirrt, wenn Sie nicht alle Annahmen über die Formel benötigen, diese sind z.T. für Teil c).

Idee: *Wie bei allen Spielen ist die Idee, rekursiv den Baum der möglichen Spielabläufe abzuarbeiten. Wir müssen also einen Algorithmus entwerfen, den wir auch mitten im Spiel rekursiv aufrufen können. Dann ist ϕ gegeben und eine beliebige Belegung B der Variablen (formal $B : \{x_1, \ldots, x_n\} \to \{0, 1\}$).*

Genauer : *Der folgende Algorithmus soll G_{flip} entscheiden, wenn er mit ϕ und der Nullbelegung aufgerufen wird:*

M_{flip} = "Bei Eingabe $< \phi, B >$:

- *Wenn $\phi[B] = 1$, gib accept aus. (Spieler I gewinnt, da er schon gar nicht mehr ziehen muss.)*
- *Sonst: Für alle x_i mit $B(x_i) = 0$ probiere "flip x_i". D.h. rufe M_{flip} mit $< \phi, B' >$ auf, wobei $B' = B$ außer dass $B'(x_i) = 1$.*
- *Wenn all diese Aufrufe accept ergeben, gib reject aus, sonst accept."*

Platzbedarf: *Es gibt maximal n Rekursionsstufen (dann sind alle x_i mit 1 belegt). Auf jeder Rekursionsstufe muss im wesentlichen die aktuelle Belegung B gespeichert werden, die lineare Länge hat.*

(Nicht verlangt: Plus eine Konstante Anzahl von Variablen, z.B. den Schleifenzähler für die x_i. Und auf der untersten Rekursionsstufe evtl. noch einmal linear viel Platz für die Auswertung der Formel.)

c) Betrachten Sie zusätzlich folgendes Spiel (es kam in den Übungen als "Knotenbelegungsspiel" vor):

$\mathbf{G}_{mark}$: Gegeben sei ein ungerichteter Graph $G = (V, E)$ ohne isolierte Knoten. Die Spieler markieren abwechselnd einen Knoten mit den folgenden Regeln:

- Ein bereits markierter Knoten darf nicht mehr markiert werden.
- Alle Nachbarknoten eines markierten Knoten dürfen auch nicht markiert werden.

Wer als erster keinen Knoten mehr markieren kann, verliert.

Wieder ist das zugehörige Entscheidungsproblem, ob Spieler I eine Gewinnstrategie hat. Zeigen Sie

$$G_{mark} \leq_P G_{flip}.$$

Hinweis: Überlegen Sie, wie jeweils ein Spielzug in beiden Spielen einander entsprechen könnte. Und überlegen Sie, wie eine nicht mehr erlaubte Graphmarkierung durch eine Formel ausgedrückt werden kann.

Idee: *Jeder Knoten von G wird eine Variable, und Markieren des Knotens bedeutet Flippen der Variable von 0 auf 1.*

Damit sind Doppelmarkierungen sowieso ausgeschlossen. Eine verbotene Markierung von G ist also eine, in der zwei Nachbarknoten markiert sind. Nachbarknoten sind gerade die Paare in E. Bei der Umsetzung zwischen Graphen und Formeln wird die Formel "Markierung verboten" genau dann wahr, wenn das Spiel G_{mark} endet, d.h. es ist eine passende Formel für das Flipspiel.

Genauer: *Wir definieren eine Abbildung f, die Graphen in Formeln verwandelt, so dass Spieler I genau dann eine Gewinnstrategie in G_{mark} mit Graph G hat, wenn er das auch in G_{flip} mit Formel $\phi = f(G)$ hat.*

- *Konstruktion von f: Sei also ein Graph $G = (V, E)$ mit $V = \{x_1, x_2, \ldots x_n\}$ gegeben. Sei $\phi = f(G)$ folgende Formel:*

$$\bigvee_{(x_i, x_j) \in E} (x_i \wedge x_j).$$

  (Dabei steht $\bigvee$ für eine lange Disjunktion all dieser Terme.) Offensichtlich ist diese Formel von der in G_{flip} erlaubten Form.
- *Diese Abbildung ist offensichtlich in polynomialer Zeit berechenbar: Man nimmt nur eine leichte Syntaxumformung an E vor. ($\wedge$ statt Trennzeichen in den Paaren, $\vee$ statt Trennzeichen zwischen den Paaren.*
- *Korrektheit: Die möglichen Abläufe entsprechen sich Zug für Zug, wie in der Idee gesagt, d.h. Markieren von x_i in G entspricht Flippen von x_i auf 1 in ϕ. Wie im 2. Teil der Idee gesagt, wird dabei genau dann ein ungültig markierter Graph erreicht, wenn die Formel wahr wird, weil genau dann für eine Kante (x_i, x_j) sowohl $x_i = 1$ als auch $x_j = 1$ wird. In G_{mark} kann der Spieler nicht mehr markieren und verliert; in G_{flip} flippt der Spieler noch, verliert aber auch.*

d) Welche der folgenden beiden Aussagen schließen Sie aus Teil b) und c)?

- Wenn $\mathbf{G}_{flip}$ PSPACE-vollständig ist, ist auch $\mathbf{G}_{mark}$ PSPACE-vollständig.
- Wenn $\mathbf{G}_{mark}$ PSPACE-vollständig ist, ist auch $\mathbf{G}_{flip}$ PSPACE-vollständig.

Die zweite. Denn aus Teil c) folgt:

$$G_{mark} \textit{ PSPACE-hart} \quad \Rightarrow \quad G_{flip} \textit{ PSPACE-hart.}$$

Außerdem wissen wir aus Teil b), dass $G_{flip} \in PSPACE$.