Zusätzliche Übungsaufgaben für die Vorlesung "Berechenbarkeit und Komplexität"

Die folgenden Aufgaben sind ein Angebot für zusätzliche Vorbereitung von der Vorlesung und für die Prüfung "Berechenbarkeit und Komplexität". Sie sind kein offizielles Lernmittel von den Mitarbeitern des Lehrstuhls i1 oder von Prof. W. Thomas. Die Aufgaben haben in keiner Weise etwas mit dem Übungssystem oder der Prüfungsklausur zu tun. Es ist nicht gewährleistet, dass die genutzten Begriffe und Definitionen mit den Vorlesungsinhalten übereinstimmen. Ebenso besteht kein Anspruch auf Korrektheit oder Vollständigkeit der Lösungen für die Aufgaben. Insbesondere ist sämtlicher Inhalt dieses Dokuments nicht repräsentativ für die Inhalte der Klausuraufgaben oder der Bewertung dieser.

Die Aufgaben sind gedacht, um das eigene Wissen zu überprüfen und das Verständnis der Vorlesungsinhalte zu festigen. Im Vergleich zu typischen Aufgaben, die in einer Klausur vorkommen, sind die meisten Aufgaben hier zeitaufwändiger.
Besonders schwere Aufgaben, die über das Übungsniveau hinausgehen, sind mit einem Stern (*) markiert.

Verbesserungen, Fehlermeldungen und insbesondere eine Lösung zu Aufgabe 7.a.viii sind erwünscht.

Update 1 (25.02.2015)

- Aufgabe 2.a: Endzustand zu Q hinzugefügt.
- Aufgabe 2.a.i: Fehlerhaften Eintrag in der Tabelle behoben.
- Aufgabe 5.h: Formatierungsfehler behoben.
- Aufgabe 6.a: L_5 verändert.

Update 2 (20.03.2015)

- Aufgabe 1.h: Neu hinzugefügt.
- Aufgabe 5.h: Doppelte Aufgabenstellung entfernt.
- Aufgabe 6: Hinweis bzgl. q_0 und $\overline{q}$ hinzugefügt

Fehlermeldungen von: Besa Demiri, Stefanie Koß, Hong An Nguyen, Philipp Nolte, Sascha Welten, Henri Lotze

Algorithmen und Turing-Maschinen

Aufgabe 1: Wissensfragen und Allgemeines

a) Was bedeutet es, dass eine Turing-Maschine eine Funktion $f : \Sigma^* \to \Sigma^*$ berechnet?

b) Sei $M = (Q, \Sigma, \Gamma, q_0, B, \overline{q}, \delta)$ eine Turing-Maschine. Beschreiben Sie die Semantik von M, d.h. erklären Sie, was eine Konfiguration ist, welchen Status die Turing-Maschine zu Beginn einnimmt, wie sie sich in jedem Schritt verhält, wann sie terminiert und welche Ausgabe sie hat.

c) Können die Bänder einer Mehrband-TM in Spuren aufgeteilt sein? Können die Spuren einer Mehrspur-TM in Bänder aufgeteilt sein?

d) Gegeben ist eine TM M mit Eingabealphabet Σ. Wie klein kann $n \in \mathbb{N}$ gewählt werden, damit eine TM M' mit Eingabealphabet Σ' konstruiert werden kann, sodass $L(M) = L(M')$ und $|\Sigma'| = n$?

e) Was ist das Kodewort/die Gödelnummer einer TM? Nennen Sie eine Eigenschaft, die man aus der Existenz solcher Kodewörter schließen kann.

f) Welche Kardinalität haben die folgenden Mengen? Zur Auswahl stehen alle natürlichen Zahlen, unendlich abzählbar oder überabzählbar.

1. Σ^*
2. $\mathcal{P}(\Sigma^*)$ (Potenzmenge)
3. $\{L \subseteq \Sigma^* \mid \text{Es existiert eine TM, die } L \text{ entscheidet}\}$
4. $\{L \subseteq \Sigma^* \mid L \text{ ist regulär}\}$
5. $\{f \mid f : \Sigma^* \to \Sigma^*\}$
6. $\{f : \Sigma^* \to \Sigma^* \mid \text{Es existiert eine TM, die } f \text{ berechnet}\}$
7. $\{<M> \mid M \text{ entscheidet die Sprache } \emptyset\}$

g) Ordnen Sie die folgenden Rechenmodelle nach Mächtigkeit, d.h. A ist mächtiger als B, wenn eine Sprache L existiert, die von A entschieden werden kann, aber nicht von B.
Turing-Maschinen; RAM; WHILE-Programme; LOOP-Programme; Turing-Maschinen, die nur gerade Bandpositionen beschreiben (d.h. verändern); Deterministische endliche Automaten

h) Betrachten Sie für diese Aufgabe deterministische endliche Automaten (DEA / DFA). Ein DFA ist ein Tupel $A = (Q, \Sigma, q_0, \delta, F)$ mit $q_0 \in Q$, $F \subseteq Q$, $\delta : Q \times \Sigma \to Q$. Zu einem solchen Automaten A gibt es für jedes Wort $w = w_1 \dots w_n \in \Sigma^*$ einen "Lauf" $\rho : \{0, \dots, n\} \to Q$. Dabei ist $\rho(0) = q_0$ und für alle i: $\rho(i+1) = \delta(\rho(i), w_i)$.
A akzeptiert w, wenn der Lauf in einem Endzustand endet, also $\rho(|w|) \in F$. Die Sprache von A ist $L(A) = \{w \in \Sigma^* \mid A \text{ akzeptiert } w\}$.

Beschreiben Sie für einen beliebigen Automaten A eine *formale* Konstruktion einer TM M, die $L(A)$ entscheidet.

Aufgabe 2: Turing-Maschinen

a)

Die Turingmaschine M soll den XOR-Operator auf eine Eingabe anwenden. Das heißt, Eingaben haben die Form $x_0 x_1 \dots x_n \# y_0 y_1 \dots y_n$ für ein $n \geq 0$ und $x_i, y_i \in \{0, 1\}$. Die Ausgabe soll dann sein $(x_0 \oplus y_0) \dots (x_n \oplus y_n)$, wobei $x_i \oplus y_i = \begin{cases} 0 & \text{falls } y_i = x_i \\ 1 & \text{sonst} \end{cases}$. Zum Beispiel wird 001#100 zu der Ausgabe 101 transformiert.

M ist gegeben durch $M = (Q, \{0, 1, \#\}, \{0, 1, \#, B, a, b\}, B, q_0, \overline{q}, \delta)$, wobei $Q = \{q_0, q_1^0, q_1^1, q_2^0, q_2^1, q_3, q_4, q_5, \overline{q}\}$. δ ist allerdings nicht vollständig spezifiziert:

δ	0	1	#	a	b	B
q_0	q_1^0, B, R	q_1^1, B, R	$q_4, \#, N$	q_0, a, L	q_0, b, L	q_0, B, R
q_1^0	$q_1^0, 0, R$	$q_1^0, 1, R$	$q_2^0, \#, R$	q_2^0, a, R	q_2^0, b, R	q_3, B, L
q_1^1	$q_1^1, 0, R$	$q_1^1, 1, R$	$q_2^1, \#, R$	q_2^1, a, R	q_2^1, b, R	q_3, B, L
q_2^0	q_3, a, N	?	$q_2^0, \#, R$	q_2^0, a, R	q_2^0, b, R	$\overline{q}, 0, N$
q_2^1	q_3, b, N	?	$q_2^1, \#, R$	q_2^1, a, R	q_2^1, b, R	$\overline{q}, 0, N$
q_3	$q_3, 0, L$	$q_3, 1, L$	?	q_3, a, L	q_3, b, L	q_0, B, R
q_4	$q_4, 0, R$	$q_4, 1, R$	$q_4, \#, R$	$q_4, 0, R$	$q_4, 1, R$	q_5, B, R
q_5	$q_5, 0, L$	$q_5, 1, L$	$\overline{q}, \#, R$	$q_5, 0, L$	$q_5, 1, L$	q_5, B, R

i) In der Tabelle sind nicht alle Einträge angegeben. Weiterhin ist einer der vorhandenen Einträge fehlerhaft. Vervollständigen Sie die Tabelle und beheben Sie den Fehler, sodass sich die TM verhält wie beschrieben.
(*Hinweis*: der Fehler liegt im dritten Element des Eintrags, d.h. in der Bewegungsrichtung des Bandkopfes.)

ii) Führen Sie nun M beispielhaft auf der Eingabe 010#110 aus. Geben Sie dafür folgende zwei Teile des Laufes an:

- von der Startkonfiguration $q_0010\#110$ so lange, bis M wieder den Zustand q_0 erreicht
- von der Konfiguration $q_0\#baa$ so lange, bis M terminiert

Nutzen Sie dabei die Schreibweise $q_0010\#110 \vdash \ldots$.

iii) Identifizieren Sie die Einträge in δ, die nie "genutzt" werden, d.h. der Zustand in Kombination mit dem Bandsymbol werden nie erreicht. Beachten Sie, dass M *nur* Eingaben der oben beschriebenen Form erhält.

b)

i) Geben Sie die formale Definition einer Turing-Maschine M an, die für eine Eingabe $x \in \Sigma^* = \{0,1\}^*$ das Wort $y(x) = \begin{cases} \$(B\#)^{\frac{|x|}{2}}\$ & \text{falls } |x| \text{ gerade} \\ \$(B\#)^{\frac{|x|-1}{2}}B\$ & \text{sonst} \end{cases}$ ausgibt. Zum Beispiel $y(0010) = \$B\#B\#\$$.

ii) Geben Sie die Sprache L an, die von folgender Turing-Maschine $M = (\{q_0, q_1, \overline{q}\}, \{0,1\}, \{0,1,B\}, q_0, B, \overline{q}, \delta)$ entschieden wird.

δ	0	1	B
q_0	$q_0, 0, R$	$q_1, 1, R$	$\overline{q}, 0, N$
q_1	$\overline{q}, 0, N$	$q_1, 1, R$	$\overline{q}, 1, N$

iii) Berechnen sie das Kodewort / die Gödelnummer der TM, die in Aufgabe ii) gegeben ist.

c)*

Betrachten Sie das Modell der *Zweidimensionalen Turing-Maschine*:
$M = (Q, \Sigma, \Gamma, q_0, B, \overline{q}, \delta), \delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\} \times \{U, D, N\}$.
Das Band wird nicht mehr durch eine eindimensionale Funktion $\mathbb{Z} \to \Gamma$, sondern eine zweidimensionale $\mathbb{Z} \times \mathbb{Z} \to \Gamma$ beschrieben.
In der Anfangskonfiguration ist der Kopf an Position $(0,0)$ und die Eingabe ist auf $(0,0), (1,0), (2,0), \ldots$ geschrieben. Erreicht die Maschine den Endzustand an Kopfposition (x,y), so ist die Ausgabe $(x,y), (x+1,y), (x+2,y), \ldots$ bis zu einem Symbol aus $\Gamma \setminus \Sigma$.

Ist dieses Modell mächtiger als die üblichen Turing-Maschinen? Falls ja, geben sie eine Sprache L an, die von einer zweidimensionalen TM entschieden wird, aber von keiner eindimensionalen. Ansonsten beschrieben sie die Idee einer Konstruktion einer äquivalenten eindimensionalen Turing-Maschine aus einer zweidimensionalen.

Aufgabe 3: Kodierungen

a)

Ein wichtiges Objekt in der Logik und anderen Gebieten sind aussagenlogische Formeln AL. Für eine Menge VAR von Variablen ist AL folgendermaßen definiert:

- Jedes $X \in \text{VAR}$ ist eine Formel: $X \in \text{AL}$
- Wenn $\varphi \in \text{AL}$, dann auch $\neg\varphi \in \text{AL}$
- Wenn $\varphi, \psi \in \text{AL}$, dann auch $\varphi \wedge \psi \in \text{AL}$

Eine Interpretation für VAR ist eine Funktion $\mathcal{I} : \text{VAR} \to \{0,1\}$, d.h. eine Funktion, die jede Variable auf "true" oder "false" setzt. Eine Interpretation $\mathcal{I}$ erfüllt eine Formel $\varphi \in \text{AL}$ genau dann, wenn:

- $\varphi = X \in \text{VAR}$ und $\mathcal{I}(X) = 1$
- $\varphi = \neg\psi$ und $\mathcal{I}$ erfüllt ψ nicht
- $\varphi = \psi \wedge \theta$ und $\mathcal{I}$ erfüllt ψ und θ

Zum Beispiel erfüllt die Interpretation, die alle Variablen auf den Wert 1 setzt die Formel $\varphi_1 = X \wedge Y$, aber nicht die Formel $\varphi_2 = X \wedge \neg Y$.

i) Gegeben eine Formel $\varphi \in \text{AL}$, definieren Sie eine Kodierung dieser Formel als Zeichenkette. Nutzen Sie dafür das Alphabet $\Sigma = \{0, 1, \#\}$. Sie können annehmen, dass $\text{VAR} = \{X_0, X_1, \ldots, X_n\}$ für ein $n \in \mathbb{N}$. (*Hinweis*: Definieren Sie die Kodierung induktiv, d.h. für $\varphi = X$, $\varphi = \neg\psi$ und $\varphi = \psi \wedge \theta$. Beachten Sie außerdem, dass Ihre Kodierung Aufgabe ii) weniger oder mehr aufwändig machen kann.)

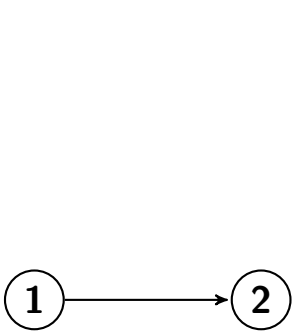
ii) Beschreiben Sie eine Turing-Maschine, die die Sprache L_{AL} entscheidet. Sie müssen keine vollständige Definition der TM angeben.

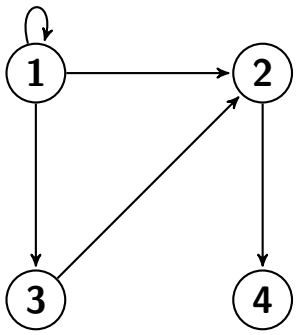
$L_{\text{AL}} = \{\text{code}(\varphi)\$\text{code}(\mathcal{I}) \mid \varphi \in \text{AL}, \mathcal{I} \text{ ist eine erfüllende Interpretation für } \varphi\}$
Dabei ist $\text{code}(\varphi)$ die von Ihnen beschriebene Kodierung und $\text{code}(\mathcal{I}) = \mathcal{I}(X_0)\mathcal{I}(X_1)\ldots\mathcal{I}(X_n)$, z.B. $\text{code}(\mathcal{I}) = 1^{n+1}$ für die Interpretation, die alle Variablen auf "true" setzt.

Geben Sie eine möglichst kleine obere Schranke für die Laufzeit ihrer Turing-Maschine an. Gehen Sie davon aus, dass n eine Konstante ist, $|\varphi|$, die Anzahl der Symbole in der Formel, aber variabel.

b)

Sei $G = (V, E)$ ein gerichteter Graph mit $V = \{v_1, \ldots, v_n\}$ und $E = \{e_1, \ldots, e_m\} \subseteq V \times V$. Im folgenden sind drei mögliche Kodierungen eines Graphen für das Alphabet $\Sigma = \{0, 1, \#\}$ gegeben und beispielhaft auf den linken abgebildeten Graphen angewandt.
Berechnen Sie die Kodierungen für den abgebildeten Graphen auf der rechten Seite. Geben Sie dann möglichst genaue untere und obere Schranken in Abhängigkeit von $|V|$ und/oder $|E|$ für die folgenden Kodierungen über $\Sigma = \{0, 1, \#\}$ an.





i) Adjazenzmatrix: $\text{code}(G) = d_{1,1}d_{1,2}\dots d_{1,n}d_{2,1}\dots d_{n,n}$,

wobei $d_{i,j} = \begin{cases} 1 & \text{falls } (v_i, v_j) \in E \\ 0 & \text{sonst} \end{cases}$.

Beispiel-Code: 0100

ii) Adjazenzliste: $\text{code}(G) = s_{1,1}s_{1,2}\dots s_{1,n}\#s_{2,1}\dots s_{2,n}\#\dots s_{n,n}$,

wobei $s_{i,j} = \begin{cases} \text{bin}(j)\# & \text{falls } (v_i, v_j) \in E \\ \epsilon & \text{sonst} \end{cases}$.

Beispiel-Code: 10##

iii) Kantenliste: $\text{code}(G) = \text{bin}(|V|)c_{1,1}c_{1,2}\dots c_{1,n}c_{2,1}\dots c_{n,n}$,

wobei $c_{i,j} = \begin{cases} \#\text{bin}(i)\#\text{bin}(j) & \text{falls } (v_i, v_j) \in E \\ \epsilon & \text{sonst} \end{cases}$.

Beispiel-Code: 10#01#10

Aufgabe 4: Alternative Rechenmodelle

a)

Geben Sie ein RAM-Programm, ein WHILE-Programm und ein LOOP-Programm an, die eine Zahl x als Eingabe erhalten und $\lceil\sqrt{x}\rceil$ ausgeben.

Berechenbarkeit

Aufgabe 5: Wissensfragen und Allgemeines

a) Was bedeutet es, wenn eine Sprache entscheidbar / semi-entscheidbar / aufzählbar / co-aufzählbar ist?

b) Definieren Sie das PKP und das MPKP.

c) Welche der folgenden Sprachen sind entscheidbar, aufzählbar und/oder co-aufzählbar?
$H, H_\epsilon, H_{\text{total}}, \text{SAT}, \text{PKP}, \emptyset$

d) Geben Sie eine unentscheidbare, co-aufzählbare Sprache an.

e) * Geben Sie eine unentscheidbare Sprache L über einem Alphabet $\Sigma \supseteq \{0,1\}$ an, deren Wörter keine TM-Kodierungen enthalten, d. h. für alle $w \in L$ und alle Aufteilungen $w = xyz$ mit $x, y, z \in \Sigma^*$ gilt, dass $y \neq <M>$ für alle Turing-Maschinen M.

f) Ein Rechenmodell, z.B. Turing-Maschinen, nennen wir "entscheidbar", wenn man für eine Instanz dieses Modells entscheiden kann, wie die Ausgabe für eine bestimmte Eingabe sein wird. Nennen Sie entscheidbare und unentscheidbare Rechenmodelle.

g) Zeigen Sie, dass Aufzählbarkeit und Semi-Entscheidbarkeit äquivalent sind. Beschreiben Sie dafür ausgehend von einem Aufzähler für L die Konstruktion einer TM, die L semi-entscheidet, und andersherum.

h) Analog zu nichtdeterministischen Automaten kann man nichtdeterministische Turing-Maschinen definieren: $M = (Q, \Sigma, \Gamma, q_0, B, \overline{q}, \Delta)$ mit den ersten sechs Attributen wie üblich und $\Delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R, N\})$. Ein Lauf von M auf einer Eingabe w ist eine Folge von Konfigurationen $K_0, \ldots,$ wobei K_0 die übliche Startkonfiguration ist und $K_i \vdash K_{i+1}$ nach einem Schritt aus Δ durchgeführt wurde.
Ein Lauf ist akzeptierend / verwerfend, analog zu deterministischen Turing-Maschinen. Eine nichtdet. TM akzeptiert w, wenn ein akzeptierender Lauf auf w existiert. Die TM verwirft w, wenn *alle* Läufe auf w verwerfend sind.

i) Wäre dieses Modell mächtiger als deterministische Turing-Maschinen, d.h. existiert eine unentscheidbare Sprache L, die von einer nichtdet. TM entschieden werden kann? Begründen Sie ihre Antwort kurz.

ii) * Begründen Sie Ihre Antwort von i) ausführlicher, d.h. geben Sie eine solche Sprache an oder zeigen Sie, dass eine solche Sprache nicht existieren kann.
(*Hinweis*: Verwenden Sie das *Lemma von König:* Sei G ein Graph, in dem jeder Knoten nur endlich viele Nachbarn hat, und in dem beliebig lange endliche Pfade existieren. Dann existiert in G ein unendlicher Pfad.)

i) Seien L_1, L_2 entscheidbare/semi-entscheidbare Sprachen. Für welche der folgenden verschiedenen Einschränkungen für die Sprache L ist L entscheidbar/semi-entscheidbar? Begründen Sie Ihre Antwort kurz.

- $L \subseteq L_1$
- $L \supseteq L_1$
- $L = L_1 \cap L_2$
- $L = L_1 \cup L_2$
- $L = \overline{L} = \Sigma^* \setminus L_1$
- $L_1 \leq L$ (Reduktion)
- $L \leq L_1$ (Reduktion)
- $L = L_1 \cdot L_2 = \{w \in \Sigma^* \mid \text{es existieren } v_1 \in L_1, v_2 \in L_2 \text{ sodass } w = v_1 v_2\}$

- $L = L_1^* = \{w \in \Sigma^* \mid$ es existieren $v_1, \ldots, v_n$ für ein $n \in \mathbb{N}$, sodass $w = v_1 v_2 \ldots v_n\}$
- $L = L_1 \Delta L_2 = \{w \in \Sigma^* \mid (w \in L_1$ und $w \notin L_2)$ oder $(w \notin L_1$ und $w \in L_2)\}$

j) Zeigen Sie, dass folgende Sprache unentscheidbar ist.
$L = \{w \in \Sigma^* \mid$ es existiert eine TM M sodass $<M> = w^{-1}$ und
M verwirft Eingabe $w\}$. Dabei ist w^{-1} das gespiegelte Wort w.
(*Hinweis*: Verwenden Sie ähnliche Argumente wie zum Beweis der Unentscheidbarkeit der Diagonalsprache.)

Aufgabe 6: Satz von Rice

a)

Welche der folgenden Sprachen sind (semi-/un)entscheidbar? Wenn eine Sprache entscheidbar oder aufzählbar ist, beschreiben Sie kurz einen passenden Algorithmus.
Nutzen Sie wenn möglich den Satz von Rice um die Unentscheidbarkeit zu beweisen. Falls der Satz nicht anwendbar ist, geben Sie die Voraussetzung an, die verletzt ist.

- $L_1 = \{<M> \mid M$ akzeptiert alle Eingaben$\}$
- $L_{2,n} = \{<M> \mid M$ akzeptiert alle Eingaben der Länge $n\}$
- $L_3 = \{<M>\#\text{bin}(n) \mid M$ akzeptiert alle Eingaben der Länge $n\}$
- $L_4 = \{<M> \mid M$ akzeptiert alle Eingaben, auf denen $\overline{q}$ nicht besucht wird$\}$
- $L_5 = \{<M> \mid M$ besucht q_0 auf allen Eingaben$\}$
- $L_6 = \{<M> \mid M$ besucht auf Eingabe ϵ jeden Zustand genau einmal$\}$
- $L_7 = \{<M> \mid M$ besucht auf Eingabe ϵ einen Zustand öfter als alle anderen Zustände$\}$
- * $L_8 = \{<M> \mid M$ akzeptiert die Eingabe ϵ und wenn $\overline{q}$ auf dieser Eingabe erreicht wird, steht auf dem Band eine Stelle rechts von der Kopfposition ein $B\}$

(*Hinweis*: Wie üblich stehen q_0 und $\overline{q}$ für den Start- und den Endzustand einer TM.)

b) *

Beweisen Sie folgenden Satz:
Sei $\mathcal{R}$ die Menge aller (partiell) berechenbaren Funktionen. Sei $\emptyset \subset S \subseteq \mathcal{R}$ und $w \in \Sigma^*$ ein Wort, sodass für alle $f \in S$ gilt: $f(w) = \perp$. Dann ist $L = \{<M> \mid f_M \in S\}$ nicht *aufzählbar*.

Aufgabe 7: Unterprogramme und Reduktion

a)

Zeigen Sie durch Reduktion oder Unterprogrammtechnik, dass die folgenden Programme unentscheidbar oder sogar nicht aufzählbar sind.

- $L_1 = \{<M> \mid M$ akzeptiert alle Wörter, die mit 1 enden$\}$
- $L_{2,g} = \{< M > \mid f_M = g\} = \{< M > \mid M$ berechnet die Funktion $g\}$ für eine partiell berechenbare Funktion $g : \Sigma^* \to \Sigma^* \cup \{\perp\}$
- $L_3 = \{<M> \mid M$ semi-entscheidet $H_\epsilon\}$
- $L_4 = \{<M>\#x\#y \mid$ auf Eingabe x hat M die Ausgabe $y\}$

- $L_5 = \{<M>|\ L(M)$ ist endlich$\}$
- $L_6 = \{<M>|\ M$ verwirft Eingabe $<M>\}$
- $L_7 = \{\text{code}(Q, \Sigma, \Gamma, B, \delta) \mid \delta : (Q \times \Gamma) \to (Q \times \Gamma \times \{L, N, R\})$, es existieren $q_1, q_2 \in Q, q_1 \neq q_2$, sodass $M' = (Q, \Sigma, \Gamma, q_1, B, q_2, \delta)$ auf allen Eingaben hält$\}$
- * $L_8 = \{< M >|\ M$ besucht auf Eingabe ϵ einen Zustand öfter als alle anderen Zustände$\}$ (Aufgabe 6.a.vii)

b) *

Zeigen Sie, dass Reduktion ein schwächeres Beweismittel als Unterprogrammtechnik ist, um Untentscheidbarkeit zu beweisen.

i) Zeigen Sie dafür zunächst, dass Reduktion höchstens so stark ist wie Unterprogrammtechnik. Für zwei Sprachen L_1, L_2, sei durch f gezeigt, dass $L_1 \leq L_2$, d.h. $f : \text{Input}(L_1) \to \text{Input}(L_2)$ ist berechenbar und $x \in L_1 \Leftrightarrow f(x) \in L_2$.
Unter der Annahme, dass L_1 unentscheidbar ist, zeigen sie mit *Unterprogrammtechnik*, dass L_2 unentscheidbar ist.

ii) Zeigen Sie nun durch Unterprogrammtechnik, dass die folgende Sprache L unentscheidbar ist. Nutzen Sie dafür H_ϵ.
$L = \{<M_1>\#<M_2>|$ Wenn M_1 und M_2 beide auf Eingabe ϵ halten, dann tun sie dies nach genau gleich vielen Schritten.$\}$

iii) Zeigen Sie nun, dass eine Reduktion $H_\epsilon \leq L$ nicht möglich ist.

Komplexität

Aufgabe 8: Wissensfragen und Allgemeines

a) Wie ist die Laufzeit einer deterministischen Turing-Maschine in Abhängigkeit von der Eingabelänge definiert? Wie ist die Laufzeit einer nicht-deterministischen Turing-Maschine in Abhängigkeit von der Eingabelänge definiert?

b) Definieren Sie das Problem 3-SAT.

c) Definieren Sie die Klassen `P` und `NP` und die Begriffen NP-schwer/-hart und NP-vollständig.

d) Nehmen Sie an, dass gilt $\texttt{P} \neq \texttt{NP} \neq \texttt{EXPTIME}$. Seien $L_1 \in \texttt{P}, L_2 \in \texttt{EXPTIME}$. Kann L_1 NP-schwer sein? Kann L_1 NP-vollständig sein? Kann L_2 NP-schwer sein? Kann L_2 NP-vollständig sein?

e) Für die folgenden drei Probleme, beschreiben Sie jeweils, welche Folgerungen die Existenz eines Algorithmus hätte, der das Problem in Polynomialzeit löst.
GRAPHISOMORHPISM, 2-SAT, 3-SAT

f) Was ist ein Polynomialzeit-Verifizierer?

g) Bzgl. der Laufzeit welchen Rechenmodells sind die Klassen `P` und `NP` definiert?

h) * So wie `P` die Rechenzeit eines Algorithmus betrachtet, kann man auch die Klasse `PSPACE` definieren, die den Nutzen von polynomial vielem Speicher erlaubt. Beweisen Sie die Aussagen $\texttt{NP} \subseteq \texttt{PSPACE} \subseteq \texttt{EXPTIME}$.

Aufgabe 9: P und NP

a)

Beschreiben Sie für die folgenden Probleme L_1, L_2 deterministische Turing-Maschinen, die die angegebene Laufzeit in Abhängigkeit der Eingabelänge n haben.
Beschreiben Sie weiterhin *nicht-deterministische* Turing-Maschinen, die die angegebene Laufzeit haben, oder begründen Sie, warum so eine NTM (vermutlich) nicht existiert. Sie brauchen keine formalen Beweise anzugeben.

i) $L_{xyz} = \{w \in \{0,1,\#\}^* \mid w = x\#y\#z, x \in \{0,1\}^m, y \in \{0,1\}^m, z \in \{0,1\}^m\}$
$L_1 = \{x\#y\#z \in L_{xyz} \mid \forall 1 \leq i \leq |x| : x_i \oplus y_i = z_i\}$,
wobei $\oplus$ der XOR-Operator ist, wie in Aufgabe 2.a.
Laufzeit (deterministisch): $\mathcal{O}(n^2)$
Laufzeit (nicht-deterministisch): $\mathcal{O}(n)$

ii) $L_2 = L_{xyz} \setminus L_1$
Laufzeit (deterministisch): $\mathcal{O}(n^2)$
Laufzeit (nicht-deterministisch): $\mathcal{O}(n)$

b)

Zeigen Sie, dass die Entscheidungsvarianten der folgenden Probleme in `NP` liegen, indem Sie ein Zertifikat und einen Verifizierer für eine Eingabe angeben. Geben Sie auch eine Kodierung für die Eingabe an. Begründen Sie kurz, warum die Länge des Zertifikats und die Laufzeit des Verifizierers polynomiell von der Eingabelänge abhängen.

i)

KNAPSACK (KP)
Eingabe: Eine Zahl $n \in \mathbb{N}$, eine Gewinnfunktion $v : \{1, \ldots, n\} \to \{1, \ldots, V\}$, eine Kostenfunktion $w : \{1, \ldots, n\} \to \{1, \ldots, W\}$, Maximalkosten $W \in \mathbb{N}$, Wunschgewinn $V \in \mathbb{N}$
Ausgabe: Eine Anzahlfunktion $m : \{1, \ldots, n\} \to \mathbb{N}$, sodass $\sum_{i=1}^{n} w(i)m(i) \leq W$ und $\sum_{i=1}^{n} v(i)m(i) \geq V$

ii)

SUBSETSUM
Eingabe: Eine Zahl $s \in \mathbb{Z}$, eine endliche Menge $M \subseteq \mathbb{Z}$
Ausgabe: Eine Teilmenge $M' \subseteq M$, sodass $\sum_{n \in M'} n = s$

iii)

CYCLICORDERING
Eingabe: Eine Zahl $n \in \mathbb{N}$, eine Menge $R \subseteq \{1, \ldots, n\}^3$
Ausgabe: Eine Permutation $(x_1, \ldots, x_n)$ von $\{1, \ldots, n\}$, sodass für alle $(x_i, x_j, x_k) \in R$ gilt: $i = k$ oder ($i < k$ und $j < i$) oder ($i < k$ und $k < j$) oder ($k < i$ und $i < j < k$)
(Betrachtet man die Folge x_i als zyklische Ordnung, so liegt für alle Tripel (a, b, c) c nicht zwischen a und b.)

Aufgabe 10: Polynomielle Reduktion

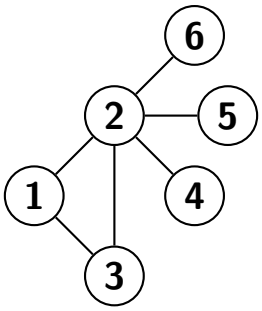
Zeigen Sie für die folgenden Probleme durch polynomielle Reduktion auf Ihnen bereits bekannte Probleme oder andere Probleme aus dieser Aufgabe, dass sie NP-schwer sind. Wenden Sie Ihre Konstruktion dann jeweils auf die angegebene Beispielinstanz an.

a)

VERTEXCOVER
Eingabe: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$.
Ausgabe: Existiert ein Vertex Cover C der Größe $|C| \leq k$? Ein Vertex Cover ist eine Teilmenge $C \subseteq V$, sodass alle Kanten $e \in E$ zu einem $v \in C$ inzident sind, d.h. $v \in e$.

Nutzen Sie zur Reduktion das Problem INDEPENDENTSET.

Führen Sie die Konstruktion beispielhaft für den folgenden Graphen und sowohl für $k = 2$ als auch für $k = 1$ aus.

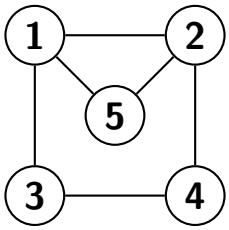


b)

SetCover
Eingabe: Eine Menge U (z.B. $U = \mathbb{N}$), eine Menge $\mathcal{S} \subseteq \mathcal{P}(U)$ mit $\bigcup \mathcal{S} = U$ und eine Zahl $k \in \mathbb{N}$.
Ausgabe: Existiert eine Menge $C \subseteq \mathcal{S}$ der Größe $|C| \leq k$, sodass $\bigcup C = U$?

Nutzen Sie zur Reduktion das Problem VertexCover.

Führen Sie die Konstruktion beispielhaft für den folgenden Graphen und sowohl für $k = 3$ als auch für $k = 2$ aus.



c)

GeneralizedAssignment
Eingabe: Eine Menge von Objekten $X = \{x_1, \ldots, x_n\}$, eine Menge von Behältern $B = \{b_1, \ldots, b_m\}$, eine Platzfunktion $s : B \to \mathbb{N}$, eine Gewinnfunktion $v : X \times B \to \{1, \ldots, V\}$, eine Gewichtsfunktion $w : X \times B \to \{1, \ldots, \max \text{Bild}(s)\}$, ein Wunschgewinn $V \in \mathbb{N}$
Ausgabe: Eine Anzahlfunktion $c : X \times B \to \mathbb{N}$, sodass für alle $1 \leq i \leq m$ gilt $\sum_{j=1}^{n} w(x_j, b_i)c(x_j, b_i) \leq w(i)$ (die Gewichte der Objekte in jedem Behälter überschreiten nicht den vorhandenen Platz) und $\sum_{i=1}^{m} \sum_{j=1}^{n} v(x_j, b_i)c(x_j, b_i) \geq V$ (der Gesamtwert aller Objekte ist mindestens V).

Nutzen Sie zur Reduktion das Problem Knapsack.

Führen Sie die Konstruktion auf der folgenden Instanz für jeweils $(W, V) = (10, 10)$ und $(W, V) = (9, 11)$ aus.
Anzahl der Objekte $n = 3$
Gewinnfunktion $v : 1 \mapsto 1, 2 \mapsto 3, 3 \mapsto 6$
Kostenfunktion $w : 1 \mapsto 2, 2 \mapsto 4, 3 \mapsto 5$

d)

MAX2-SAT
Eingabe: Eine aussagenlogische Formel φ in 2-KNF, d.h. $\varphi = \bigvee_{i=1}^{n} (x_i \vee y_i)$, und eine Zahl $k \in \mathbb{N}$.
Ausgabe: Existiert eine Belegung der Variablen, sodass mindestens k Klauseln von φ wahr werden?

Nutzen Sie zur Reduktion das Problem 3-SAT.

Führen Sie die Konstruktion beispielhaft auf den folgenden Formeln aus:
$\varphi_1 = (x \vee x \vee \overline{x})$
$\varphi_2 = (x \vee x \vee x) \wedge (\overline{x} \vee \overline{x} \vee \overline{x})$

e)

CLOSESTSTRING
Eingabe: Eine Menge von Zeichenketten $\{w_1, \ldots, w_n\} \subset \Sigma^m$ der Länge m und eine Zahl $k \in \mathbb{N}$.
Ausgabe: Existiert ein String $v \in \Sigma^m$ sodass die Hamming-Distanz zwischen v und allen w_i höchstens k ist $(d(v, w_i) \leq k)$?
Die *Hamming-Distanz* von zwei Zeichenketten gleicher Länge $x = x_1 \ldots x_m$, $y = y_1 \ldots y_m$ ist definiert als die Anzahl der Positionen, an denen die Worte sich unterscheiden. Das heißt $d(x, y) = |\{1 \leq i \leq m \mid x_i \neq y_i\}|$.

Nutzen Sie zur Reduktion das Problem 3-SAT.
Führen Sie die Konstruktion beispielhaft auf den folgenden Formeln aus:
$\varphi_1 = x \wedge (x \vee y \vee \overline{z})$
$\varphi_2 = (x \vee y) \wedge (\overline{x} \vee y) \wedge (x \vee \overline{y}) \wedge (\overline{x} \wedge \overline{y})$

f)

FEEDBACKARCSET
Eingabe: Ein gerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$.
Ausgabe: Existiert ein feedback arc set A der Größe $|A| \leq k$?
Ein feedback arc set (FAS) ist eine Menge von Kanten $A \subseteq E$ sodass der Graph $G' = (V, E \setminus A)$ keine Zykel enthält.

Nutzen Sie zur Reduktion das Problem VERTEX COVER.

Führen Sie die Konstruktion beispielhaft für den folgenden Graphen und sowohl für $k = 3$ als auch für $k = 2$ aus.