Aufgabe 1 Chomsky-Hierachie

(a) Zunächst gilt Typ-3 $\subseteq$ Typ-2 $\subseteq$ Typ-1 $\subseteq$ Typ-0

Typ-0 Grammatiken: Eine Grammatik G ist vom Typ-0, wenn keine Einschränkungen vorliegen.

Typ-1 Grammatiken: Eine Grammatik G heißt kontextsensitiv (Typ-1), wenn alle Produktionsregeln die folgende Form haben: $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$ mit $A \in V, \alpha_1, \alpha_2 \in (V \cup \Sigma)^*, \beta \in (V \cup \Sigma)^+$

Typ-2 Grammatiken: Eine Grammatik G heißt kontextfrei (Typ-2), wenn alle Produktionsregeln folgende Form haben: $A \to \beta$ mit $A \in V, \beta \in (V \cup \Sigma)^+$

Typ-3 Grammatiken: Eine Grammatik G heißt regulär (Typ-3), wenn alle Produktionsregeln folgende Form haben: $A \to aB$ oder $A \to a$ mit $A, B \in V, a \in \Sigma$

(b)
- L_1 ist eine Typ-2 Grammatik, da es mit Typ-3 nicht möglich ist, mehr als ein Nichtterminalsymbol pro Ableitungsschritt hinzuzufügen. Dies ist allerdings nötig, damit die Anzahl der as und bs gleichbliebt.
- L_2 ist eine Typ-3 Grammatik, da die Anzahl der as und bs nicht voneinander abhängt. Deswegen reicht es, wenn man pro Ableitungsschritt ein Terminalsymbol hinzufügt wird.
- L_3 ist eine Typ-1 Grammatik, da es mindestens eine Produktionsregel mit der Form $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$ geben muss, um die gleiche Anzahl von terminal Symbolen an unterschiedlichen Stellen zu erzeugen.

(c) $G_1 = (V, \Sigma, P, S)$
$V = \{S, A, B\}$
$\Sigma = \{a, b\}$
$P = \{S \to aAb, A \to aAb \mid B, B \to bBa \mid ba\}$
Da alle Produktionsregeln die Form $A \to \beta$ mit $A \in V, \beta \in (V \cup \Sigma)^+$ haben, ist die Grammatik vom Typ-2.

Aufgabe 2 Wiederholung: Chomsky-Grammatiken

(a) $L_1 = \{a^n b^m \mid n, m \in \mathbb{N}, n > 1\}$

(b) $L_2 = \{a^{2n} b^{3n} \mid n \in \mathbb{N}\}$

(c) aabbb $\in L_1 \cap L_2$

(d) $L_1 \cup L_2 = \{a^{2n} b^{3n} \mid n \in \mathbb{N}\}$ da $L_2 \subseteq L_1$

Aufgabe 3 Einführung Turingmaschine

(a) Im Binärsystem ist die Multiklikation mit 2 nichts anderes als eine Verschiebung der Bits um eine Stelle nach links.

1. Gehe nach rechts an das Ende der Zahl.
2. Schreibe eine '0' rechts neben die Zahl.
3. Gehe nach links bis zum Anfang der Zahl und wechsle in den Endzustand.

(b) Im Binärsystem ist die Multiklikation mit 4 eine Linksverschiebung um 2 Stellen. Entweder man führt die Turingmaschine aus (a) zweimal hintereinander aus oder man modifiziert die Turingmaschine aus (a) und führt den zweiten Schritt zweimal hintereinander aus.

Aufgabe 4 Mengen, Funktionen und Relationen

(a) 1. $\{n \in \mathbb{N} \mid n \mod 4 = 0 \wedge n \mod 100 = 0 \implies n \mod 400 = 0\}$ (Schaltjahre)
2. $\{(n,m) \in \mathbb{Z} \times \mathbb{Z} \mid m \neq 0\}$

(b) $\{1,2,3\} \cup \{x,z\} = \{1,2,3,x,z\}$
$\{1,2,3\} \cap \{x,z\} = \varnothing$
$\{1,2,3\} \setminus \{x,z\} = \{1,2,3\}$
$\{1,2,3\} \setminus \{2\} = \{1,3\}$
$\{1,2,3\} \times \{x,z\} = \{(1,x),(1,z),(2,x),(2,z),(3,x),(3,z)\}$
$\mathcal{P}(\{\varnothing\}) = \{\varnothing, \{\varnothing\}\}$
$\mathcal{P}(\mathcal{P}(\varnothing)) = \{\varnothing, \{\varnothing\}, \{\{\varnothing\}\}, \{\varnothing, \{\varnothing\}\}\}$

(c) 1. $R_f = \{(x,y) \in X \times Y \mid f(x) = y\}$
2. Eine Funktion $f_R : X \to Y$ legt für jedes $x \in X$ genau ein $y \in Y$ fest. Es gibt allerdings Relationen, wo mehrere y einem x zugewiesen werden:
$R_f = \{(x,y) \in X \times Y \mid x^2 + y^2 = 2^2\} = \{\dots,(2,2),(2,-2),\dots\}$
3. Es ist möglich jede Relation R als eine Funktion $g_R : X \to \mathcal{P}(Y)$ darzustellen, da durch eine Potenzmenge die Probleme der mehrfachen Zuweisung ($x \in X$ wird auf eine Menge mit Mächtigkeit > 1 abgebildet) und der fehlenden Zuweisung ($x \in X$ wird auf die leere Menge $\varnothing$ abgebildet) gelöst werden.

Aufgabe 5 Zählen von Turingmaschinen

(a) Für die deterministische TM gibt es ingesamt 18 verschiedene "rechte Seiten" der Übergangsfunktion ($|\{z_1,z_2,z_3\}| * |\{a,\Box\}| * |\{R,L,N\}| = 18$) und 6 verschiedene "linke Seiten" der Übergangsfunktion ($|\{z_1,z_2,z_3\}| * |\{a,\Box\}| = 6$) , Also gibt es ingesamt $18^6 = 34012224$ mögliche Übergangsfunktionen.

(b) Für die nichtdeterministische TM gibt es ingesamt 2^{18} verschiedene "rechte Seiten" der Übergangsfunktion ($|\mathcal{P}(\{z_a,z_b\} \times |\{x,y,\Box\} \times \{R,L,N\})| = 2^{18}$) und 6 verschiedene "linke Seiten" der Übergangsfunktion ($|\{z_a,z_b\}| * |\{x,y,\Box\}| = 6$). Es gibt also ingesamt $2^{18^6} = 262144^6 = 3,245 * 10^{32}$ mögliche Übergangsfunktionen.

(c) Die Überlegung aus (a) kann in eine verallgemeinerte Formel umgewandelt werden: Die Anzahl der Übergangsfunktionen kann mit $(3 * m * n)^{m*n}$ berechnet werden, wobei $m = |\Gamma|$ und $n = |Z|$.

(d) Die Überlegung aus (b) kann auch wieder in eine verallgemeinerte Formel umgewandelt werden: Die Anzahl der Übergangsfunktionen kann mit: $2^{(3mn)^{m*n}} = 2^{3m^2n^2}$ berechnet werden, wobei $m = |\Gamma|$ und $n = |Z|$.

Aufgabe 6 Was macht diese Turingmaschine?

(a) Für die Eingabe 'ab' durchläuft die TM folgende Schritte:
$z_0ab \vdash az_0b \vdash abz_0\Box \vdash az_sb\# \vdash a\#z_b\# \vdash a\#\#z_b\Box \vdash a\#z_r\#b \vdash az_s\#\#b \vdash z_sa\#\#b \vdash \#z_a\#\#b \vdash$
$\#\#z_a\#b \vdash \#\#\#z_ab \vdash \#\#\#bz_a\Box \vdash \#\#\#z_rba \vdash \#\#z_r\#ba \vdash \#z_s\#\#ba \vdash z_s\#\#\#ba \vdash z_s\Box\#\#\#ba \vdash$
$z_d\#\#\#ba \vdash z_d\#\#ba \vdash z_d\#ba \vdash z_dba \vdash z_Eba$

Für die Eingabe 'ab' : $z_0ab \vdash^* z_Eba$
Für die Eingabe 'aabb': $z_0aabb \vdash^* z_Ebbaa$.

(b) Die TM schreibt die Eingabe rückwärts auf das Band und löscht danach die ursprüngliche Eingabe.

(c) z_0: Es wird das rechte Ende der Eingabe mit einem #-Zeichen makiert. Danach wechselt die TM nach z_s.
z_s: Es wird vom rechten Ende des Wortes aus nach einem a oder b gesucht, welches mit # überschrieben wird. Bei einem a geht es mit z_a weiter, bei einem b geht es mit z_b weiter. Falls es kein a oder b mehr gibt, wechselt die Turingmaschine nach z_d.

z_a: Das gefunde a wird an das rechte Wortende geschrieben. Danach wechselt die TM nach z_r.
z_b: Das gefunde b wird an das rechte Wortende geschrieben. Danach wechselt die TM nach z_r.
z_r: Der Lesekopf wird wieder auf das ursprüngliche Ende der Eingabe gesetzt, dann wechselt die TM nach z_s.
z_d: Der Lesekopf wird auf das erste Zeichen des umgedrehten Wortes gesetzt und löscht dabei alle erzeugen #. Danach TM wechselt in den Endzustand z_e.
z_E: Die TM befindet sich im Endzustand und terminiert.

Aufgabe 7 Turingmaschine für eine reguläre Sprache

(a) $G = (V, \Sigma, P, S)$
$V = \{S, A, B\}$
$\Sigma = \{a, b\}$
$P = \{S \to bS \mid aA, A \to aA \mid ab\}$

(b) Man kann aa^*a in aaa^* umformen. Damit ist $L : b^*aaa^*b$.

$\delta(z_0, a) = (z_1, \square, R)$ z_0 erkennt b^* und wechselt bei einem a nach z_1 .
$\delta(z_0, b) = (z_0, \square, R)$
$\delta(z_0, 1) = (z_4, \square, R)$
$\delta(z_0, \square) = (z_E, \square, N)$

$\delta(z_1, a) = (z_2, \square, R)$ z_1 erkennt das zweite a und wechselt nach z_2. Ansonsten in den
$\delta(z_1, b) = (z_4, \square, R)$ Fehlerzustand z_4.
$\delta(z_1, 1) = (z_4, \square, R)$
$\delta(z_1, \square) = (z_E, \square, N)$

$\delta(z_2, a) = (z_2, \square, R)$ z_2 erkennt a^*, wechselt bei einem b nach z_3 und sonst in den Fehlerzustand z_4.
$\delta(z_2, b) = (z_3, \square, R)$
$\delta(z_2, 1) = (z_4, \square, R)$
$\delta(z_2, \square) = (z_E, \square, N)$

$\delta(z_3, a) = (z_4, \square, R)$ Falls das Band leer ist, wird eine '1' geschrieben und in den Endzustand
$\delta(z_3, b) = (z_4, \square, R)$ gewechselt, ansonsten in den Fehlerzustand z_4.
$\delta(z_3, 1) = (z_4, \square, R)$
$\delta(z_3, \square) = (z_E, 1, N)$

$\delta(z_4, a) = (z_4, \square, R)$ Fehlerzustand. Überschreibt Band mit $\square$ und wechselt in den Endzustand.
$\delta(z_4, b) = (z_4, \square, R)$
$\delta(z_4, 1) = (z_4, \square, R)$
$\delta(z_4, \square) = (z_E, \square, N)$

$\delta(z_E, a) = (z_E, a, N)$ Endzustand. Übergangsfunktion vorgegeben.
$\delta(z_E, b) = (z_E, b, N)$
$\delta(z_E, 1) = (z_E, 1, N)$
$\delta(z_E, \square) = (z_E, \square, N)$

Aufgabe 8 Wiederholung: Funktionen Teil II

(a) $f_a : \mathbb{R} \to \mathbb{R}, f_a(x) := 2$
Da $f_a(1) = f_a(2)$ aber $1 \neq 2$ also ist f_a nicht injektiv.
Nicht surjektiv da es z.B. für $y \in \mathbb{R} = 3$ kein $x \in \mathbb{R}$ mit $f_b(x) = 3$ gibt.

(b) $f_b : \mathbb{N} \to \mathbb{R}, f_b := x$
Aus $f_b(a) = f_b(b)$ folgt $a = b$ also ist $f_b(x)$ injektiv.
Nicht surjektiv da es z.B. für $y \in \mathbb{R} = 2,5$ es kein $x \in \mathbb{N}$ mit $f_b(x) = 2,5$ gibt.

(c) $f_c : \mathbb{R} \to \{2\}, f_c(x) := 2$
Da $f_c(1) = f_c(2)$ aber $1 \neq 2$ also ist f_c nicht injektiv.
f_c ist surjektiv, da es für alle $y \in \{2\}$ ein Urbild in $\mathbb{R}$ gibt.

(d) $f_d : \mathbb{R} \to \mathbb{R}, f_d(x) := x$
Aus $f_d(a) = f_d(b)$ folgt $a = b$ also ist $f_d(x)$ injektiv.
Für alle $y \in \mathbb{R} = Y$ existiert ein $x \in \mathbb{R} = X$, falls $X = Y$ und $f_d(x) = x$ also die Identität von x ist.

Aufgabe 9 Überabzahlbarkeit und Diagonalisierung

(a) Wenn I_1 abzählbar ist, kann man mit der Funktion $f_n : \mathbb{N}_0 \Rightarrow I_1$, wobei $f_n(x)$ die x. Dezimalstelle der n-ten Zahl angibt eine Liste erstellen, die alle Zahlen $\in I_1$ enthält.

n	$f_n(0)$	$f_n(1)$	$f_n(2)$	$f_n(3)$	$f_n(4)$	…
0	-0,	1	2	3	4	
1	0,	1	2	3	4	
2	-0,	2	3	4	5	
3	0,	2	3	4	5	
⋮						

Wenn man nun eine neue Zahl $\in I_1$ mit $g : \mathbb{N}_0 \Rightarrow \mathbb{N}_0$ mit $g(n) = \begin{cases} 0 & \text{falls } n = 0 \wedge f_0(0) = -0 \\ -0 & \text{falls } n = 0 \wedge f_0(0) = 0 \\ 1 & \text{falls } f_n(n) = 2 \\ 2 & \text{falls } f_n(n) \neq 2 \end{cases}$ definieren. Da die neue Zahl, per Definition nicht in der Liste enthalten seien kann, muss I_1 überabzählbar groß sein.

(b) $\varphi_r : I_r \Rightarrow I_1$, $\varphi_r(x) = \frac{x}{r}$ mit $\varphi^{-1} : I_1 \Rightarrow I_r$, $\varphi^{-1}(x) = rx$

(c) Die Bildmenge der Funktion $f(x) = \tan(x)$ auf dem Intervall $\left]-\frac{\pi}{2}, \frac{\pi}{2}\right[$ ist $\mathbb{R}$. Man kann für jedes $r \in \mathbb{R}$ eine Funktion $f_r : I_r \Rightarrow \mathbb{R}, f_r(x) = \tan(\frac{x}{r} * \frac{\pi}{2})$ definieren. f_r ist bijektiv da die Umkehrfunktion $f_r^{-1} : I_r \Rightarrow \mathbb{R}, f_r^{-1}(x) = \frac{2r}{\pi} \arctan(x)$ existiert.

(d) Es gibt genau dann eine bijektive Funktion $g_n : \mathbb{N}^n \Rightarrow \mathbb{R}$, wenn $\mathbb{N}^n$ und $\mathbb{R}$ gleich mächtig sind. Mit vollständiger Induktion kann bewiesen werden, das alle n-Tupel aus $\mathbb{N}$ gleichmächtig und damit abzählbar sind: Für $\mathbb{N}^2$ gibt es eine Funktion $f_2 : \mathbb{N} \times \mathbb{N} \Rightarrow \mathbb{N}_0$, $f_2(n,m) = \left(\sum_{i=0}^{m+n} i\right) + n$ Für $n = 3$ gibt es eine rekursive Funktion $f_3 : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \Rightarrow \mathbb{N}_0$, $f_3(m,n,k) = f_2(m, f_2(n,k))$. Für $n \geq 3$ kann diese Funktion zu $f_2 : \mathbb{N}^n \Rightarrow \mathbb{N}_0$, $f_n(m_1, \ldots, m_n) = f_2(m_1, f_{n-1}(m_2, \ldots, m_n))$ verallgemeinert werden. Da die Menge der reellen Zahlen aber überabzählbar ist, sind $\mathbb{N}^n$ und $\mathbb{R}$ nicht gleichmächtig, daraus folgt das es keine bijektive Funtion $g_n : \mathbb{N}^n \Rightarrow \mathbb{R}$ geben kann.

Aufgabe 10 Bandbeschränkt vs. Zeitbeschränkt

(a) Stimmt nur, wenn alle $b(n)$ besuchtet Stellen links oder rechts von der Startposition liegen und der Kopf niemals eine Bwegung in die andere Richtung macht, d.h. jede Stelle wird nur genau einmal besucht.

(b) Stimmt. Mit $t(n)$ Schritten kann sich der Lesekopf maximal $t(n)$-mal bewegen. Geht der Kopf immer nur in eine Richtung können maximal $t(n) + 1$ Stellen besucht werden. $t(n)$ Stellen rechts/links von der Startposition, plus die Startposition selber. Alle anderen Bewegungenskombinationen haben $< b(n)$ verschiedene besuchte Stellen, da mindestens eine Stellen doppelt besucht werden muss.

(c) Stimmt nicht. Da das Band unendlich lang ist, ist es möglich das die TM die ganze Zeit über nur nach links oder rechts läuft und sich deswegen keine Konfiguration wiederholt.

(d) Stimmt. Da es bei einer bandbeschränkter TM nur endlich viel Platz gibt und das Arbeitsalphabet eine endliche Menge ist, gibt es auch auch nur endlich viele Permutationen für die Konfiguration einer bandbeschränkten TM. Da aber unendlich viele Schritte gemacht werden, muss mindestens eine Konfiguration mehrmals erreicht werden.

(e) Stimmt. Wenn eine Konfiguration vor erreichen des Endzustandens mehrfach erreicht wird, bebeutet dass das die TM in einer Endlosschleife und kann nicht terminieren. Sie führt also unendlich viele Schritte aus.

Aufgabe 11 Zweiband-Turingmaschine

Sei $M = (\{z_1, z_E\}, \{a, b\}, \{a, b, \Box\}, \delta, z_1, \Box, \{z_E\})$ eine Mehrband-Turingmaschine mit $\delta : Z \times \Gamma^k \Rightarrow Z \times \Gamma^k \times \{L, R, N\}^k$ und $k = 2$ Bändern.

In z_1 wird die Eingabe von Band 1 von links nach rechts gelesen und nacheinander mit $\Box$ überschrieben. Gleichzeitig wird das gelesene Zeichen von nach rechts nach links auf das Ausgabeband geschrieben. Dadurch wird die Eingabe umgedreht. Sobald die Eingabe komplett abgearbeitet ist wechselt die TM in den Endzustand z_E

$\delta(z_1, (a, a)) = (z_1, (a, a), (N, N))$
$\delta(z_1, (a, b)) = (z_1, (a, b), (N, N))$
$\delta(z_1, (a, \Box)) = (z_1, (\Box, a), (R, L))$
$\delta(z_1, (b, a)) = (z_1, (b, a), (N, N))$
$\delta(z_1, (b, b)) = (z_1, (b, b), (N, N))$
$\delta(z_1, (b, \Box)) = (z_1, (\Box, b), (R, L))$
$\delta(z_1, (\Box, a)) = (z_1, (\Box, a), (N, N))$
$\delta(z_1, (\Box, b)) = (z_1, (\Box, b), (N, N))$
$\delta(z_1, (\Box, \Box)) = (z_E, (\Box, \Box), (N, N))$

z_E ist der Endzustand. Hier passiert nichts.
$\delta(z_E, (a, a)) = (z_E, (a, a), (N, N))$
$\delta(z_E, (a, b)) = (z_E, (a, b), (N, N))$
$\delta(z_E, (a, \Box)) = (z_E, (a\Box), (N, N))$
$\delta(z_E, (b, a)) = (z_E, (b, a), (N, N))$
$\delta(z_E, (b, b)) = (z_E, (b, b), (N, N))$
$\delta(z_E, (b, \Box)) = (z_E, (\Box, b), (N, N))$
$\delta(z_E, (\Box, a)) = (z_E, (\Box, a), (N, N))$
$\delta(z_E, (\Box, b)) = (z_E, (\Box, b), (N, N))$
$\delta(z_E, (\Box, \Box)) = (z_E, (\Box, \Box), (N, N))$

Aufgabe 12 Lineare Bandbeschrankung

Damit eine $c * n$ lange Bandbeschriftung auf eine Beschriftung der Länge n umgebaut werde kann, muss die Anzahl der Informationen pro Zellen erhöht werden. Dazu fasst man jeweils c-Zeichen zu einem neuen Zeichen zusammen, z. B. mit $c = 2$ und der Eingabe: $\Box \mid a \mid b \mid b \mid a \mid a \mid a \mid \Box \Rightarrow \Box a \mid bb \mid aa \mid a\Box$. Das ürsprüngliche Bandalphabet ($\Gamma = \{a, b, \Box\}$) wird zu $\Gamma' = \{aa, ab, a\Box, ba, bb, b\Box, \Box a, \Box b, \Box\Box\}$ umgebaut. Das Bandalphabet vergrößert sich also um einen Faktor von $\frac{|\Gamma'|}{|\Gamma|} = 3 = 3^{2-1}$. Für den allgemeinen Fall vergrößert sich das ursprüngliche Bandalphabet also um den Faktor $\mid \Gamma \mid^{c-1}$.

Aufgabe 13 LOOP-Programme

LOOP-Programm A berechnet $x_2 - x_1$, indem es x_1-mal 1 von x_2 abzieht.
LOOP-Programm B berechnet $x_1 - x_2$, indem es x_2-mal 1 von x_1 abzieht.
LOOP-Programm C berechnet $|x_1 - x_2|$. Indem es zuerst Programm B ausführt und das Ergebnis in x_3 speichert, für $x_2 \geq x_1$ ist $x_3 = 0$. Danach wird das Programm A ausgeführt und das Ergebnis in x_4 gespeichert, wobei $x_4 = 0$ falls $x_1 \geq x_2$ ist. Da nun entweder $x_3 = 0$ oder $x_4 = 0$ ist, kann man einfach $x_3 + x_4$ rechnen um die positive Differenz von x_1 und x_2 zu erhalten.

Aufgabe 14 GOTO-Programme

(a) $f_1 : \mathbb{N}_0 \times \mathbb{N}_0 \Rightarrow \mathbb{N}_0,\ f_1(a,b) = a + b$

$x_0 = x_1;$	Das Ergebnis $x_0 =$ a gesetzt.
M_1 : IF $x_2 = 0$ THEN GOTO M_2;	Fall $x_2 = 0$ ist muss nichts addiert werden.
$x_0 = x_0 + 1;$	Das Ergebnis x_0 wird um Eins erhöht.
$x_2 = x_2 - 1;$	Die Eingabe x_2 wird um Eins reduziert.
GOTO M_1;	Sprung zum Check für $x_2 = 0$.
M_2 : HALT;	

(b) $f_2 : \mathbb{N}_0 \times \mathbb{N}_0 \Rightarrow \mathbb{N}_0,\ f_2(a,b) = a - b$

$x_0 = x_1;$	Die Ausgabe $x_0 =$ a gesetzt.
M_1 : IF $x_2 = 0$ THEN GOTO M_2;	Falls $x_2 = 0$ ist muss nichts subtraiert werden.
$x_0 = x_0 - 1;$	Das Ergebnis x_0 wird um Eins reduziert.
$x_2 = x_2 - 1;$	Die Eingabe x_2 wird um Eins reduziert.
IF $x_0 = 0$ THEN GOTO M_2;	Abbruch, da es keien Zahlen < 0 gibt.
GOTO M_1;	Sprung zum Check für $x_2 = 0$.
M_2 : HALT;	

(c) $f_3 : \mathbb{N}_0 \times \mathbb{N}_0 \Rightarrow \mathbb{N}_0,\ f_3(a,b) = min(a,b)$

$x_3 = f_2(x_1, x_2);$	Die Differenz von x_1 und x_2 wird gebildet.
IF $x_3 = 0$ THEN GOTO M_1;	Wenn $x_3 = 0$ ist, ist $x_2 >= x_1$, also mit M_1 weiter.
$x_0 = x_2;$	x_2 ist die kleinere Zahl.
GOTO M_2;	Sprung zum Programmende.
$M_1 : x_0 = x_1;$	x_1 ist die kleinere Zahl.
M_2 : HALT;	

(d) $f_4 : \mathbb{N}_0 \times \mathbb{N}_0 \Rightarrow \mathbb{N}_0,\ f_4(a,b) = a$ MOD b

$M_1 : x_0 = f_2(x_1, x_2);$	Die Differenz von x_1 und x_2 wird gebildet.
IF $x_0 = 0$ THEN GOTO M_2;	Abbruchbedingung. Differenz $= 0$.
$x_1 = x_0;$	Die Differenz ($> x_2$) ist das neue a.
GOTO M_1;	Sprung zurück.
$M_2 : x_4 = f_2(x_1, x_2)$	Differenz x_1 und x_2.
$x_5 = f_2(x_2, x_1)$	Differenz x_2 und x_1.
IF $x_4 = 0$ THEN GOTO M_3;	Falls $x_1 - x_2 = 0$ ist ...
GOTO M_5;	
M_3 : IF $x_5 = 0$ THEN GOTO M_4;	und $x_2 - x_1 = 0 \Rightarrow x_1 = x_2$
$x_0 = x_1;$	Falls $x_1 \neq x_2$ steht in x_1 der modulo ...
GOTO M_5;	
$M_4 : x_0 = 0;$	ansonsten geht der Modulo glatt auf ($= 0$).
M_5 : HALT;	

(e) $f_5 : \mathbb{N}_0 \times \mathbb{N}_0 \Rightarrow \mathbb{N}_0,\ f_5(a, b) = a$ DIV b

M_1 : IF $x_1 = 0$ THEN GOTO M_3;	Abbruchbedingung: 0 div a ist immer 0.
$x_3 = f_4(x1, x2)$;	x_3 enthält x_1 mod x_2.
$x_3 = f_2(x1, x3)$;	Damit die Divsion glatt wird $x_3 = x_1 - x_3$
M_2 : IF $x_3 = 0$ THEN GOTO M_3;	Abbruchbedingung: x_2 passt nicht in x_1 .
$x_3 = f_2(x1, x2)$;	Division durch wiederholte Subtraktion.
$x_0 = x_0 + 1$;	x_0 zählt mit wie oft x_2 in x_1 passt.
$x_1 = x_3$;	Minuend aktualisieren.
GOTO M_1;	Sprung zum Check für $x_3 = 0$.
M_3 : HALT;	

Aufgabe 15 WHILE-Programme

(a) Für $x_1 = 3$ und $x_2 = 2$ gibt das Programm $x_0 = 12$ aus.

(b) Das Programm implementiert die Funktion $f : \mathbb{N} \Rightarrow \mathbb{N}$, $f(x_1, x_2) = 2^{x_2} * x_1$. Wobei die innere Schleife (Zeile 3-6) die Zahl verdoppelt und die äußere Schleife (Zeile 1-8) die Anzahl der Verdoppelungen vorgibt.

Aufgabe 16 WHILE-Programm

Das folgende WHILE-Programm berechnet $f(x_1, x_2) = ggT(x_1, x_2)$:

```
WHILE x1 ≠ x2 DO
    IF x2 − x1 = 0 THEN
        x1 := x1 − x2
    ELSE
        x2 := x2 − x1
    END;
END;
x0 := x1
```

Makro WHILE $x_1 \neq x_2$ DO A END:

```
IF x2 = x1 THEN x3 := 0 ELSE x3 := 1 END;
WHILE x3 ≠ 0 DO
    A;
    IF x2 = x1 THEN x3 := 0 ELSE x3 := 1 END;
END
```

Makro IF $x_2 = x_1$ THEN A ELSE B END:

```
x3 := x1 − x2;
x4 := x2 − x1;
x3 := x3 + x4;
IF x3 = 0 THEN A ELSE B END
```

Makro IF $x_1 = 0$ THEN A ELSE B END:

$x_2 := x_1;$
$x_3 := 1;$
$x_4 := 0;$
WHILE $x_2 \neq 0$ DO
    $x_2 := x_2 - 1;$
    $x_3 := 0;$
    $x_4 := 1$
END;
WHILE $x_3 \neq 0$ DO
    $x_3 := x_3 - 1;$
    A
END;
WHILE $x_4 \neq 0$ DO
    $x_4 := x_4 - 1;$
    B
END

Aufgabe 17 LOOP-Programm

(a) In einem LOOP-Programm wird die Anzahl der Schleifendurchläufe festgelegt, bevor die Schleife beginnt. Innerhalb der Schleife kann die Anzahl dann nicht mehr verändert werden, deshalb muss jede Schleife und damit auch jedes LOOP-Programm terminieren.

(b) Damit eine Programmiersparche turingmächtig ist, muss sie partielle Funktionen berechnen können. Da alle LOOP-Programme terminieren kann es keine partiellen Funktionen geben, also sind LOOP-Programme nicht turingmächtig.

(c) Das folgende LOOP-Programm berechnet die Funktion $f : \mathbb{N}_0 \Rightarrow \mathbb{N}_0$, $f(n) = \begin{cases} 0 & \text{, falls } n = 0 \\ \lceil \log_2(n) \rceil & \text{, sonst} \end{cases}$

$x_2 := 1;$
LOOP x_1 DO
    IF$(x_2 + 1) - x_1 = 0$ THEN
        $x_2 := 2 * x_2;$
        $x_3 := x_3 + 1;$
    END
END
$x_0 := x_3;$

Aufgabe 18 primitiv-rekursive Funktionen

(a) $f_1(0, y) = g(y) = id = \pi_1^1(y)$
$f_1(x + 1, y) = h(f_1(x, y), x, y) = dec(\pi_1^3(f_1(x, y), x, y))$

(b) $\chi_{\{0\}}(0) = 1$
$\chi_{\{0\}}(n + 1) = h(\chi_{\{0\}}(n), n) = 0$

(c) $f_2(0, y) = g(y) = f_3(y)$
$f_2(x + 1, y) = h(f_2(x, y), x, y) = y * \pi_1^3(f_2(x, y), x, y)$
$f_3(0) = 0$
$f_3(x + 1) = h(f_3(x), x) = 1$

Aufgabe 19 SKIP-Programm

(a) Damit ein GOTO-Programm durch ein SKIP-Programm simuliert werden kann, muss die komplette GOTO-Syntax durch die SKIP-Syntax ausgedrückt werden:

- Wertzuweisung: Die Syntax für die Wertezuweisung ist bei beiden Sprachen identisch.
- Unbedingter Sprung: $M_j:$ GOTO $M_i = \begin{cases} \text{SKIP } (i-j-1) & \text{, falls } i > j \\ \text{GOTOSTART; SKIP}(j-1) & \text{, falls } i = j \\ \text{GOTOSTART; SKIP}(j-i-1) & \text{, falls } i < j \end{cases}$
- Bedingter Sprung: Die Syntax ist fast identisch, nur GOTO muss durch SKIP ersetzt werden.
- Stopanweisung: HALT = SKIP(n); , wobei n = Anzahl der Zeilen im GOTO-Programm.

(b) Damit ein SKIP-Programm durch ein GOTO-Programm simuliert werden kann, muss die komplette SKIP-Syntax durch die GOTO-Syntax ausgedrückt werden:

- Wertzuweisung: Die Syntax für die Wertezuweisung ist bei beiden Sprachen identisch.
- Unbedingter Sprung: SKIP$(k) = \begin{cases} M_i : \text{GOTO } M_{i+k+1}; & \text{, falls } i+k+1 < n \\ M_i : \text{HALT}; & \text{, falls } i+k+1 \geq n \end{cases}$ n = Zeilenanzahl.
- Bedingter Sprung: Die Syntax ist fast identisch, nur SKIP muss durch GOTO ersetzt werden.
- Sprung zum Start: GOTOSTART = GOTO M_1;

Aufgabe 20 Totale Funktionen

Laut Aufgabe ist die Menge der gültigen Programme L rekursiv aufzählbar. Das bedeutet das es eine surjektive Funktion $F : \mathbb{N}_0 \Rightarrow L$ gibt, die einer Zahl n das n-te Programm in der Aufzählung zuordnet. Wenn man nun eine neues Programm definieren, mit $h : \mathbb{N}_0 \Rightarrow \mathbb{N}_0$, $h(x) = F(x) + 1$, welches die Ausgabe des x-ten Programmes + 1 berechnet. Diese Funktion ist laut Definition berechenbar und total, muss also in der Menge der gültigen Programme L sein. Sei jetzt i der Index von dem Programm $h(x)$ in der Aufzählung L. Dann würde $F(i) = h(i) = F(i) + 1$ gelten. Das führt allerdings zu einem Wiederspruch und daher kann $h(x)$ nicht Teil der Aufzählung sein. Das wiederspricht der 2. Eigenschaft des Sprache (jede total Funktion kann mit einem Programm der Sprache beschrieben werden), weshalb die Sprache so nicht existieren kann.

Aufgabe 21 primitiv-rekursive Funktionen

$\chi_{\{0\}} : \mathbb{N}_0 \Rightarrow \mathbb{N}_0,\ f(n) = \begin{cases} 1 & , x = 0 \\ 0 & , x > 0 \end{cases}$ siehe Blatt 5 Aufgabe 18.

(a) $leq(x, y) = \chi_{\{0\}}(x - y)$

(b) $geq(x, y) = \chi_{\{0\}}(y - x)$

(c) $eq(x, y) = mult(leq(x, y), geq(x, y))$

Aufgabe 22 μ-rekursive Funktionen

(a) Mit dem μ-Operator ist $\mu f_1 : \mathbb{N}_0^1 \Rightarrow \mathbb{N}_0$, $\mu f_1(y) = min\{n \mid f_1(n, y) = 0\}$, mit $f_1(x, y) = y - x$. f_1 ist genau dann = 0, wenn $y - n = 0 \Leftrightarrow y = n$ ist.
Also ist $\mu f_1 : \mathbb{N}_0^1 \Rightarrow \mathbb{N}_0$, $\mu f_1(x) = x$, also die Identitätsfunktion.

(b)
1. $\sqrt[x]{y} = n \Leftrightarrow y \leq n^x \Leftrightarrow y - n^x \leq 0$. Die Funktion bricht also ab, sobald $n^x \geq y$ ist.
   n^x ist primitiv-rekursiv, siehe Blatt 5 Aufgabe 18(c).
   $\mu f_{pot} : \mathbb{N}_0^2 \Rightarrow \mathbb{N}_0$, $\mu f_{pot}(x, y) = min\{n \mid f_{pot}(n, x, y) = 0\}$
   $f_{pot} : \mathbb{N}_0^3 \Rightarrow \mathbb{N}_0$, $f_{pot}(n, x, y) = y - n^x$

2. Für $n = x$ ist $1 - eq(x, n) = 0$ und $1 - eq(y, n) = 1$, also $(1 - eq(x, n)) * (1 - eq(y, n)) = 0$. Für $n = y$ analog. Da n hochgezählt wird tritt wird $f_{min}(n, x, y)$ zuerst bei dem kleineren Parameter = 0.
   $\mu f_{min} : \mathbb{N}_0^2 \Rightarrow \mathbb{N}_0$, $\mu f_{min}(x, y) = min\{n \mid f_{min}(n, x, y) = 0\}$
   $f_{min} : \mathbb{N}_0^3 \Rightarrow \mathbb{N}_0$, $f_{min}(n, x, y) = (1 - eq(x, n)) * (1 - eq(y, n))$

Aufgabe 23 Reduktionen I

Mit der Vorverarbeitung $f(a,b,c)$ ist es möglich mit der Maschine M_{sum} zuentscheiden ob $c = a - b$ gilt: $f : \mathbb{Z}^3 \Rightarrow \mathbb{Z}^3$, $f(a,b,c) = (a, -b, c)$, also $c' = c$, $b' = -b$ und $a' = a$.

Aufgabe 24 Reduktionen II

(a) Für alle Zahlen gilt: gerade Zahl + 1 = ungerade Zahl. Also kann man mit der Vorverarbeitung f_a das Problem A auf B reduzieren: $f_a : \mathbb{Z} \Rightarrow \mathbb{Z}$, $f_a(x) = x + 1$.

(b) Wenn $x \mod 11 = 0$ dann ist $x = 11n$, also ein Vielfaches von 11. Desweiteren ist $y = 11x$, also ein vielfaches von 121 und damit $y \mod 121 = 0$. Also kann mit f_b Problem A auf B reduziert werden: $f_b : \mathbb{Z} \Rightarrow \mathbb{Z}$, $f_b(x) = 11x$.

(c) Man kann ein Wort $w \in A$ in ein Wort $w' \in B$ mit folgenden Regeln umbauen:
$a \Rightarrow c :$ Jedes a wird durch ein c ersetzt.
$b \Rightarrow de :$ Jedes b wird durch de ersetzt.
Damit ist $\#_c(w') = \#_a(w)$, $\#_d(w') = \#_b(w)$ und $\#_e(w') = \#_b(w)$
also $2 * \#_b(w) = \#_d(w') + \#_e(w')$ und damit ist das Kriterium für Sprache B erfüllt.

Aufgabe 25 Entscheidbarkeit

(a) Man kann mit der Funktion f eine TM bauen, die charakteristische Funktion χ_L berechnet. Dabei geht die TM folgendermaße vor:

1. Die Eingabe ist $w \in \sum^*$.
2. Setze einen Counter $i = 0$.
3. Berechne f(i).
4. Falls $f(i) = w$, lösche das Band schreibe ein 1 und wechsel in den Endzustand.
5. Setze $i = i + 1$ und gehe nach 3. solange eine Eigenschaft erfüllt ist:
   (a) $|w| < |f(i)|$
   (b) $|w| = |f(i)|$ und es gibt ein $j \in \{1, \ldots, |w|\}$, sodass $\forall(k < j \in \mathbb{N} \mid w_k = f(i)_k \wedge w_j <_{\sum} f(i)_j)$
6. Lösche das Band, schreibe eine 0 und wechsel in den Endzustand.

(b) Man kann mit TM_{χ_L} eine TM bauen die f berechnet, dabei geht die TM folgendermaßen vor:

1. Die Eingabe ist $i \in \mathbb{N}_0$
2. Setze zwei Counter $j = 0$ und $l = 0$
3. Simuliere auf TM_{χ_L} alle Wörter mit Länge l in längen-lexikographischer Reihenfolge. Für jedes Wort $\in L$ erhöhe j um 1.
4. Falls j = i breche ab und gebe das letze erzeugte Wort zurück.
5. Setze $l = l + 1$ und gehe nach 3.

Aufgabe 26 Franz der Frisör (Diagonalisierung reloaded)

Aufgabe 27 Reduktionen

Aufgabe 28 Ein neues Halteproblem

Aufgabe 29 Abgeschlossenheit bezüglich Reduktionen

(a)

(b)

Aufgabe 30 Satz von Rice / Unentscheidbarkeit

Es ist zuzeigen, dass $L_0 \leq L_=$ gilt. Falls $L_0 \leq L_=$ gilt, kann man aus $L_=$ eine TM für L_0 bauen. Dies ist mit der Vorverarbeitungs Funktion $f(w) = w\#w_0$ möglich, wobei w_0 die Kodierung einer TM ist, die die konstante Nullfunktion berechnet. Da laut Satz von Rice L_0 unentscheidbar ist und $L_0 \leq L_=$ gilt, ist auch $L_=$ unentscheidbar.

Aufgabe 31 Postsches Korrespondenzproblem

(a) Eine mögliche Lösung ist : 3,1,4

$$\begin{array}{l|l|l} ab & aba & abaa \\ a & ab & abaa \end{array}$$

(b) Das MPCP gilt als gelöst, wenn die komplette Sequenz gleich ist, wenn man mit dem Paar (x_1, y_1) beginnt, unterscheiden sich die Sequenzen bereits im 1. Zeichen.

(c) [1]

Es gibt ingesamt K^n verschiedenen Indexfolgen. Da man in jedem Schritt eins der K-Wortpaare auswählen kann.

2.

$$\begin{aligned}
\sum_{n=1}^{N} \lambda(K,n) = \sum_{n=1}^{N} K^n &= K + K^2 + \cdots + K^N \\
s_n &= K + K^2 + \cdots + K^N && \mid *K \qquad (1) \\
Ks_n &= K^2 + K^3 + \cdots + K^{N+1} && \qquad (2) \\
\text{(1) - (2): } s_n - Ks_n &= K - K^{N+1} \\
s_n(1-K) &= K - K^{N+1} && \mid/(1-K) \\
s_n &= \frac{K - K^{N+1}}{1-K} \\
\sum_{n=1}^{N} \lambda(K,n) &= \frac{K - K^{N+1}}{1-K}
\end{aligned}$$

Aufgabe 32 Reduktion H $\leq$ MPCP

(a) Die Turingmaschine sucht vom Wortanfang aus ein b. Wenn die TM ein b gefunden hat wird der Kopf wieder auf den Start der Eingabe gesetzt und die Maschine wechselt in den Endzustand. Die TM terminiert bei allen Eingaben die mindestens ein b enthalten : $L(M) = \{a^*b^+a^*b^*\}$

(b) Da es sich um ein MPCP handelt ist $(x_1, y_1) = (\#, \#z_0b\#)$

1. Kopierregeln $= \{(a,a), (b,b), (\Box,\Box), (\#,\#)\}$
2. Überführugnsregeln $= \{(z_0b, z_1b), (z_E\Box, z_E\Box), (z_Ea, z_Ea), (z_Eb, z_Eb),$
   $(z_0\Box, \Box z_0), (z_0a, az_0), (z_1\Box, \Box z_E)$
   $(az_1a, z_1aa), (bz_1a, z_1ba), (\Box z_1a, z_1\Box a),$
   $(az_1b, z_1ab), (bz_1b, z_1bb), (\Box z_1b, z_1\Box b),$
   $(\#z_1a, \#z_1\Box a), (\#z_1b, \#z_1\Box b),$
   $(z_E\#, z_E\Box\#),$
   $(z_0\#, \Box z_0\#), (z_1\#, z_E\Box\#)$
3. Löschregeln $= \{(az_E, z_E), (z_Ea, z_E), (bz_E, z_E), (z_Eb, z_E), (\Box z_E, z_E), (z_E\Box, z_E)\}$
4. Abschlussregeln $= \{(z_E\#\#, \#)\}$

(c) Für die Eingabe 'b' durchläuft die TM folgende Konfigurationen: $z_0b \vdash z_1b \vdash z_1\Box b \vdash z_Eb$

(d) $(\#, \#z_0b\#), (z_0b, z_1b), (\#z_1b, \#z_1\Box b), (\#,\#), (z_1\Box, \Box z_E), (b,b), (\#,\#),$
$(\Box z_E, z_E), (b,b), (\#,\#)(z_Eb, z_E), (\#,\#)(z_E\#\#, \#)$

Aufgabe 33 Es weihnachtet sehr!

Ein dem Angebot wird eine Turingmaschine beschrieben. Das Band wird durch die Häuser simuliert, das Bandalphabet ist 0 (kein Geschenk im Haus) und 1 (Geschenk im Haus). Jeder Wichtel stellt eine Übergangsfunktion oder eine Abfolge von Übergangsfunktionen (Der Schlittenfahrer) dar. Das Problem ist der letze Punkt des Vertrages. Es soll ein Computerprogramm geben das entscheidet ob die Wichtel zurück kommen oder nicht. Wenn man das Angebot als eine Art Turingmaschine interpretiert, würde diese Programm das Halteproblem lösen. Das Halteprogramm ist allerdings nicht entscheidbar und deshalb kann es ein solchen Computerprogramm nicht geben. Herr M. E. Phisto ist also ein Betrüger.

Aufgabe 34 Eigenschaften von Turingmaschinen

(a) Entscheidbar. Es ist möglich die TM für 100 Schritte zu simulieren und dann zuüberprüfungen ob die TM sich in einem Endzustand befindet oder nicht.

(b) Entscheidbar. Man kann die TM simulieren und dabei die durchlaufenen Konfigurationen speichern. Dann tritt irgendwann einer der drei möglichen Fälle ein :

- Die Konfiguration der TM ist länger als 100 Zellen. $\Rightarrow$ Abbruch, Aussgabe falsch.
- Die TM terminiert mit einer Konfiguration < 100 Zellen. $\Rightarrow$ Aussage korrekt.
- Die TM terminiert nicht (= eine Konfiguration kommt mehrmals vor), aber alle Konfigurationen sind < 100 Zellen $\Rightarrow$ Abbruch. Aussage korrekt.

(c) Angenommen das Problem wäre entscheidbar, dann könnte man jede TM so umbauen das sie, beim Wechsel in den Endzustand ein "a" auf das Band schreibt. Mit einem solchen Umbau könnte man dann das Halteproblem entscheiden. Die TM hält mit einer Eingabe w an $\Leftrightarrow$ Die TM schreibt ein "a" auf das Band. Da das Halteproblem allerdings nicht entscheidbar ist, für diese Annahme zu einem Wiederspruch, also ist die Eigenschaft nicht entscheidbar.

Aufgabe 35 Eine nicht entscheidbare Sprache

(a) Es ist zuzeigen das $\overline{H_0} \leq P$: W_n sei die Kodierung einer TM die auf dem leeren Band hält. w sei die Eingabe für $\overline{H_0}$. Dann ist die Vorverarbeitungsfunktion $f(w) = W_n\#w$.
Also gilt $\overline{H_0} \leq P$ und damit ist P nicht semi-entscheidbar.

(b) Es ist zuzeigen das $\overline{H_0} \leq \overline{P}$: W_∞ sei die Kodierung einer TM die nicht auf dem leeren Band hält. w sei die Eingabe für $\overline{H_0}$. Dann ist die Vorverarbeitungsfunktion $f(w) = w\#W_\infty$.
Also gilt $\overline{H_0} \leq P$ und damit ist $\overline{P}$ nicht semi-entscheidbar.

Aufgabe 36 Die Landau-Notation

(a) Es sind ein N und ein C gesucht für die gilt $\forall x > N : \mid f(x) \mid \leq C \mid g(x) \mid$.
Versuch mit $N = 1$:

$$\begin{aligned}
\mid f(x) \mid =& \mid \sqrt{\mid x \mid} + 10 \mid = \sqrt{x} + 10 \mid \text{ Wegen } N = 1 \text{ und } x > N \\
&= x^{\frac{1}{2}} \leq C * x^3 \\
&\Leftrightarrow \frac{x^{\frac{1}{2}}}{x^3} + \frac{10}{x^3} \leq C \\
&\Leftrightarrow \frac{1}{\sqrt{x^5}} + \frac{10}{x^3} \leq C \\
&\Leftrightarrow 11 \leq C \mid \text{ Maximum der linken Seite ist bei } x = 1
\end{aligned}$$

Für $x > 1$ und $C \geq 11$ gilt die Ungleichung und damit ist $f \in O(g)$.

(b) Da $f(x)$ und $h(x)$ beide in $O(g)$ liegen muss es zwei N und C geben für die gilt:
$\forall x > N_f :\mid f(x) \mid \leq C_f \mid g(x) \mid$
$\forall x > N_h :\mid f(x) \mid \leq C_h \mid g(x) \mid$
Falls $\varphi \in O(g)$ muss folgendes gelten:

$$\begin{aligned} \forall x > N\colon\ \mid \varphi(x) \mid = \\ \mid f(x) + h(x) \mid &\leq \mid f(x) \mid + \mid h(x) \mid \\ &\leq C_f * \mid g(x) \mid + C_h * \mid g(x) \mid \\ &\leq (C_f + C_h) * \mid g(x) \mid \\ &= C * \mid g(x) \mid \ \text{ mit } C = C_f + C_h \end{aligned}$$

Das N für $\varphi(x)$ ist $N = \max(N_f, N_h)$

(c) Falls: $f \in O(g) \Rightarrow \mid f(x) \mid \leq C * \mid g(x) \mid$ also

$$\begin{aligned} \forall x > N\colon\ 3^x &\leq C * 2^x \\ \Leftrightarrow \frac{3^x}{2^x} &\leq C \\ \Leftrightarrow \left(\frac{3}{2}\right)^x &\leq C \end{aligned}$$

Da $\frac{3}{2} > 1$ ist das Wachstum nicht beschränkt und deshalb gilt: $f \notin O(g)$

(d) Sei $x \geq 1$, dann gilt für alle $i \in \{1, \ldots, n-1\} : x^i \leq x^n$

$$\begin{aligned} \mid f(x) \mid = \mid \sum_{i=0}^{n} k_i x^i \mid \quad &\leq \quad \sum_{i=0}^{n} \mid k_i \mid * \mid x^i \mid \\ = \sum_{i=0}^{n} \mid k_i \mid * x^i &\leq \left(\sum_{i=0}^{n} \mid k_i \mid\right) * x^n \\ C = \left(\sum_{i=0}^{n} \mid k_i \mid\right) \end{aligned}$$

Da es ein N und C gibt gilt $f \in O(x^n)$

Aufgabe 37 Unäres vs. logarithmisches Kostenmaß

(a) unäres Kostenmaß : $O(n * b) = O(n * \log_2(n))$

(b) logarithmisches Kostenmaß : $O(n * b) = O(2^b * b)$

Aufgabe 38 Rechenschritte einer Turing-Maschine

(a) Die TM interpretiert die Eingabe als natürliche Zahl in Binärdarstellung und dekrementiert sie so lange, bis 0 auf dem Band steht.

(b) Der worst-case triff für Eingaben auf, die nur aus Einsen bestehen. Für diesen Fall müssen ingesamt $2^n - 1$ Drekementierungen durchgeführt werden. Jede Dekrementation besteht aus maximal $2n$-Schritte (Wenn die Zahl die Form 100000... hat). Zusätzlichen wir die Zahl am Anfang und am Ende einmal komplett durchlaufen. Zusammen ergibt sich eine Gesamtlaufzeit von:
$O(2n + (2^n - 1)(2n)) = O(2n + 2n * 2^n - 2n) = O(2n * 2^n) = O(n2^n)$

Aufgabe 39 Polynomielle Laufzeiten

(a) Sei $P_B : \mathbb{N}_0 \Rightarrow \mathbb{N}_0$ das Polynom, welches die Laufzeit von B beschränkt und p_f das entsprechende Polynom für f. Sei w ein Wort der Länge n. Wir können die Länge von $f(w)$ mittels $p_f(n)$ abschätzen, da eine TM in polynomische Zeit höchsten polynomial viele Zeichen auf das Band schreiben kann. Ingesamt lässt sich A in polynomialer Zeit entscheiden und z war mit maximal $p_f(n) + p_B(p_f(n))$ Schritten.

(b) Ansatz analog zum Teil a), nur dass noch die Vorverwarbeitung von C dazu kommt, also ingesamt $p_f(n) + p_g(p_f(n)) + p_C(p_g(p_f(n)))$ Schritte.

(c) Mit der Vorverarbeitungsfunktion f wie folgt möglich:

- Man wählt zunächst 2 Wörter: $w_0 \notin B$ und $w_1 \in B$. Möglich da $B \neq \emptyset \wedge B \neq \sum *$
- Dann ist $f(w) = \begin{cases} w_0, w \notin A \\ w_1, w \in A \end{cases}$

Aufgabe 40 Kurze Fragen I (Entscheidbarkeit und Berechenbarkeit)

(a) Falsch, da jede entscheidbare Sprache auch semi-entscheidbar ist.

(b) Falsch, die charakterischte Funktion für die Sprache $\sum *$ ist $\chi_{\sum *} = 1$, welche offensichtlich berechenbar ist.

(c) Falsch, wegen dem Satz von Rice.

(d) Richtig, eine TM die die Identitätsfunktion berechnet, kann einfach direkt terminieren.

(e) Falsch. Beweis durch Gegenbeispiel: $Y = \emptyset$ und $X \neq \emptyset$

(f) Richtig, da jede det TM als eine nicht det TM aufgefasst werden kann und es für jede nicht det TM eine äquivalente det TM gibt.

Aufgabe 41 Weitere kurze Fragen

(a) Falsch, wäre H NP-vollständig also $H \in NP$ müsste H entscheidbar sein.

(b) Falsch, da $P \subseteq NP$ gitl, würde aus der Aussage sofort $P = NP$ folgen, was allerdings (noch) nicht bekannt ist.

(c) Richtig, man kann einen Faktor $t \in \mathbb{N}$ raten und dann in polynomialer Zeit überprüfuen ob $2 \leq t \ln k$ und ob n durch t teilbar ist.

(d) Richtig. Wenn man eine Aussagelogischeformel f in KNF negiert, dann ist $\neg f$ in DNF. Es gilt:

$$\begin{aligned} \text{F hat eine erfüllende Belegung } &\Leftrightarrow \neg f \text{ ist nicht die Tautologie} \\ f \in \text{KNF-SAT} &\Leftrightarrow \neg f \notin \text{DNF-TAU} \end{aligned} \tag{3}$$

$\Rightarrow$ Wäre DNF-TAU in polynomialer Zeit entscheidbar, dann könnte man auch KNF-SAT in polynomialer Zeit entscheiden und damit auch alle anderen Probleme in NP. Also wäre P = NP, was allerdings (noch) nicht bewiesen ist.

Aufgabe 42 CLIQUE $\leq_p$ INDEPENDENT - SET

(a) Sei $G = (V, E)$ ein ungerichteter Graph und k eine Zahl. Wir definieren $f(G, k) = (G', k)$ mit $G'(V, E')$ und $E' = \{(v_1, v_2) \in V \times V \mid v_1 \neq v_2 \wedge (v_1, v_2) \notin E\}$, also die Menge aller Kanten die nicht in E liegen. Diese Funktion f ist total und in polynomialzeit berechenbar.

$$(G, k) \in \text{CLIQUE} \Leftrightarrow f(G, k) \in \text{INDEPENDET - SET} \tag{4}$$

und damit ist INDEPENDET - SET NP-hart.

(b) Es ist möglich eine Menge M mit k Knoten zuraten und dann zu überprüfen ob $M \in IS$ ist. Es gibt maximal $\frac{n(n-1)}{2}$ Kanten, also ist das testen in polynomialer Zeit möglich. Zusammen mit dem Ergebnis aus (a) bedeutet das das INDEPENDENT - SET NP-vollständig ist

Aufgabe 43 VERTEX - COVER

(a) Sei $G = (V, E)$ ein ungerichteter Graph mit $n = \mid V \mid$ und $k \in \mathbb{N}$. Wir definieren $f(G, k) = (G, n-k)$. Diese Funktion f ist total und in polynomialzeit berechenbar.
Wir müssen zeigen: $(G, k) \in$ INDEPENDENT - SET $\Leftrightarrow f(G, k) \in$ VERTEX - COVER
"$\Rightarrow$": Sei $(G, k) \in IS$ und sei S ein IS der Größe k. Wir behaupten, dass $V \backslash S$ ein VC der Größe

$n - k$ ist, d. h. dass für alle Kanten $e \in E$ min ein Knoten der Kante in $V \backslash S$ liegt.
Angenommen es gäbe eine Kante $e' = \{v_1, v_2\} \in E$ mit $v_1 \notin V \backslash S$ und $v_2 \notin V \backslash S$, d.h. $v_1, v_2 \in S$. Eine solche Kante gibt es nicht, da $S \in IS$ ist.
"$\Leftarrow$": Sei $(G, n - k) \in VC$ und C ein VC der Größe $n - k$. Dann ist $V \backslash C$ ein IS der Größe k, d. h. es gibt keine Kannten zwischen ja 2 Knoten aus $V \backslash C$. Angenommen es gäbe eine Kanten mit $e = v_1, v_2 \in E$ mit $v_1, v_2 \in V \backslash C$, d.h. $v_1, v_2 \notin C$. Eine solche Kanten gibt es nicht, da C eine Knotenüberdeckung ist.
Daraus folgt das VC NP-hart ist.

(b) Es ist möglich eine Menge M mit k Knoten zuraten und dann zu überprüfen ob $M \in VC$ ist. Es gibt maximal $\frac{n(n-1)}{2}$ Kanten, also ist das testen in polynomialer Zeit möglich. Zusammen mit dem Ergebnis aus (a) bedeutet das das VERTEX - COVER NP-vollständig ist