

**Aufgabe 1** Chomsky-Hierarchie

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

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:  $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$  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:  $A \rightarrow \beta$  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:  $A \rightarrow aB$  oder  $A \rightarrow a$  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 gleichbleibt.
  - $L_2$  ist eine Typ-3 Grammatik, da die Anzahl der as und bs nicht voneinander abhängt. Deswegen reicht es, wenn man pro Ableitungsschritt ein Terminalsymbol hinzugefügt wird.
  - $L_3$  ist eine Typ-1 Grammatik, da es mindestens eine Produktionsregel mit der Form  $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$  geben muss, um die gleiche Anzahl von terminal Symbolen an unterschiedlichen Stellen zu erzeugen.

- (c)
- $G_1 = (V, \Sigma, P, S)$

$$V = \{S, A, B\}$$

$$\Sigma = \{a, b\}$$

$$P = \{S \rightarrow aAb, A \rightarrow aAb \mid B, B \rightarrow bBa \mid ba\}$$

Da alle Produktionsregeln die Form  $A \rightarrow \beta$  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 Multiplikation 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 wechsele in den Endzustand.
- (b) Im Binärsystem ist die Multiplikation 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.  $\{n \in \mathbb{N} \mid n \bmod 4 = 0 \wedge n \bmod 100 = 0 \implies n \bmod 400 = 0\}$  (Schaltjahre)  
 2.  $\{(n, m) \in \mathbb{Z} \times \mathbb{Z} \mid m \neq 0\}$
- (b)  $\{1, 2, 3\} \cup \{x, z\} = \{1, 2, 3, x, z\}$   
 $\{1, 2, 3\} \cap \{x, z\} = \emptyset$   
 $\{1, 2, 3\} \setminus \{x, z\} = \{1, 2, 3\}$   
 $\{1, 2, 3\} \setminus \{2\} = \{1, 3\}$   
 $\{1, 2, 3\} \times \{x, z\} = \{(1, x), (1, z), (2, x), (2, z), (3, x), (3, z)\}$   
 $\mathcal{P}(\{\emptyset\}) = \{\emptyset, \{\emptyset\}\}$   
 $\mathcal{P}(\mathcal{P}(\emptyset)) = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\emptyset, \{\emptyset\}\}\}$
- (c) 1.  $R_f = \{(x, y) \in X \times Y \mid f(x) = y\}$   
 2. Eine Funktion  $f_R : X \rightarrow Y$  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  $g_R : X \rightarrow \mathcal{P}(Y)$  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  $\emptyset$  abgebildet) gelöst werden.

**Aufgabe 5** Zählen von Turingmaschinen

- (a) Für die deterministische TM gibt es insgesamt 18 verschiedene "rechte Seiten" der Übergangsfunktion ( $|\{z_1, z_2, z_3\} * \{a, \square\} * \{R, L, N\}| = 18$ ) und 6 verschiedene "linke Seiten" der Übergangsfunktion ( $|\{z_1, z_2, z_3\} * \{a, \square\}| = 6$ ). Also gibt es insgesamt  $18^6 = 34012224$  mögliche Übergangsfunktionen.
- (b) Für die nichtdeterministische TM gibt es insgesamt  $2^{18}$  verschiedene "rechte Seiten" der Übergangsfunktion ( $|\mathcal{P}(\{z_a, z_b\} \times \{x, y, \square\} \times \{R, L, N\})| = 2^{18}$ ) und 6 verschiedene "linke Seiten" der Übergangsfunktion ( $|\{z_a, z_b\} * \{x, y, \square\}| = 6$ ). Es gibt also insgesamt  $2^{18^6} = 262144^6 = 3,245 * 10^{32}$  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:  $2^{(3mn)^{m * n}} = 2^{3m^2 n^2}$  berechnet werden, wobei  $m = |\Gamma|$  und  $n = |Z|$ .

**Aufgabe 6** Was macht diese Turingmaschine?

- (a) Für die Eingabe 'ab' durchläuft die TM folgende Schritte:  
 $z_0 ab \vdash a z_0 b \vdash ab z_0 \square \vdash a z_s b \# \vdash a \# z_b \# \vdash a \# \# z_b \square \vdash a \# z_r \# b \vdash a z_s \# \# b \vdash z_s a \# \# b \vdash \# z_a \# \# b \vdash \# \# z_a \# b \vdash \# \# \# z_a b \vdash \# \# \# b z_a \square \vdash \# \# \# z_r b a \vdash \# \# z_r \# b a \vdash \# z_s \# \# b a \vdash z_s \# \# \# b a \vdash z_s \square \# \# \# b a \vdash z_d \# \# \# b a \vdash z_d \# \# b a \vdash z_d \# b a \vdash z_d b a \vdash z_E b a$

Für die Eingabe 'ab':  $z_0 ab \vdash^* z_E ba$

Für die Eingabe 'aabb':  $z_0 aabb \vdash^* z_E bbaa$ .

- (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 markiert. 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 erzeugten #. 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)  $G = (V, \Sigma, P, S)$

$$V = \{S, A, B\}$$

$$\Sigma = \{a, b\}$$

$$P = \{S \rightarrow bS \mid aA, A \rightarrow aA \mid ab\}$$

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

$$\delta(z_0, a) = (z_1, \square, R) \quad z_0 \text{ erkennt } b^* \text{ und wechselt bei einem a nach } z_1.$$

$$\delta(z_0, b) = (z_0, \square, R)$$

$$\delta(z_0, 1) = (z_4, \square, R)$$

$$\delta(z_0, \square) = (z_E, \square, N)$$

$$\delta(z_1, a) = (z_2, \square, R) \quad z_1 \text{ erkennt das zweite a und wechselt nach } z_2. \text{ Ansonsten in den}$$

$$\delta(z_1, b) = (z_4, \square, R) \text{ Fehlerzustand } z_4.$$

$$\delta(z_1, 1) = (z_4, \square, R)$$

$$\delta(z_1, \square) = (z_E, \square, N)$$

$$\delta(z_2, a) = (z_2, \square, R) \quad z_2 \text{ erkennt } a^*, \text{ wechselt bei einem b nach } z_3 \text{ und sonst in den Fehlerzustand } z_4.$$

$$\delta(z_2, b) = (z_3, \square, R)$$

$$\delta(z_2, 1) = (z_4, \square, R)$$

$$\delta(z_2, \square) = (z_E, \square, N)$$

$$\delta(z_3, a) = (z_4, \square, R) \text{ Falls das Band leer ist, wird eine '1' geschrieben und in den Endzustand}$$

$$\delta(z_3, b) = (z_4, \square, R) \text{ gewechselt, ansonsten in den Fehlerzustand } z_4.$$

$$\delta(z_3, 1) = (z_4, \square, R)$$

$$\delta(z_3, \square) = (z_E, 1, N)$$

$$\delta(z_4, a) = (z_4, \square, R) \text{ Fehlerzustand. Überschreibt Band mit } \square \text{ und wechselt in den Endzustand.}$$

$$\delta(z_4, b) = (z_4, \square, R)$$

$$\delta(z_4, 1) = (z_4, \square, R)$$

$$\delta(z_4, \square) = (z_E, \square, N)$$

$$\delta(z_E, a) = (z_E, a, N) \text{ Endzustand. Übergangsfunktion vorgegeben.}$$

$$\delta(z_E, b) = (z_E, b, N)$$

$$\delta(z_E, 1) = (z_E, 1, N)$$

$$\delta(z_E, \square) = (z_E, \square, N)$$

### Aufgabe 8 Wiederholung: Funktionen Teil II

(a)  $f_a : \mathbb{R} \rightarrow \mathbb{R}, f_a(x) := 2$

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)  $f_b : \mathbb{N} \rightarrow \mathbb{R}, f_b := x$   
 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)  $f_c : \mathbb{R} \rightarrow \{2\}, f_c(x) := 2$   
 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)  $f_d : \mathbb{R} \rightarrow \mathbb{R}, f_d(x) := x$   
 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 Überabzählbarkeit 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	
⋮						

$$\text{Wenn man nun eine neue Zahl } \in I_1 \text{ mit } g : \mathbb{N}_0 \Rightarrow \mathbb{N}_0 \text{ mit } g(n) = \begin{cases} 0 & \text{falls } n = 0 \wedge f_0(0) = -0 \\ -0 & \text{falls } n = 0 \wedge f_0(0) = 0 \\ 1 & \text{falls } f_n(n) = 2 \\ 2 & \text{falls } f_n(n) \neq 2 \end{cases}$$

definieren. Da die neue Zahl, per Definition nicht in der Liste enthalten sein 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  $]-\frac{\pi}{2}, \frac{\pi}{2}[$  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, dass 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  $f_n : \mathbb{N}^n \Rightarrow \mathbb{N}_0, f_n(m_1, \dots, m_n) = f_2(m_1, f_{n-1}(m_2, \dots, m_n))$  verallgemeinert werden. Da die Menge der reellen Zahlen aber überabzählbar ist, sind  $\mathbb{N}^n$  und  $\mathbb{R}$  nicht gleichmächtig, daraus folgt dass es keine bijektive Funktion  $g_n : \mathbb{N}^n \Rightarrow \mathbb{R}$  geben kann.

### Aufgabe 10 Bandbeschränkt vs. Zeitbeschränkt

- (a) Stimmt nur, wenn alle  $b(n)$  besucht Stellen links oder rechts von der Startposition liegen und der Kopf niemals eine Bewegung 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 Bewegungskombinationen haben  $< b(n)$  verschiedene besuchte Stellen, da mindestens eine Stelle doppelt besucht werden muss.
- (c) Stimmt nicht. Da das Band unendlich lang ist, ist es möglich dass 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 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 Endzustands mehrfach erreicht wird, bedeutet das, dass die TM in einer Endlosschleife und kann nicht terminieren. Sie führt also unendlich viele Schritte aus.

### Aufgabe 11 Zweiband-Turingmaschine

Sei  $M = (\{z_1, z_E\}, \{a, b\}, \{a, b, \square\}, \delta, z_1, \square, \{z_E\})$  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  $\square$  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  $z_E$ .

$$\begin{aligned}\delta(z_1, (a, a)) &= (z_1, (a, a), (N, N)) \\ \delta(z_1, (a, b)) &= (z_1, (a, b), (N, N)) \\ \delta(z_1, (a, \square)) &= (z_1, (\square, a), (R, L)) \\ \delta(z_1, (b, a)) &= (z_1, (b, a), (N, N)) \\ \delta(z_1, (b, b)) &= (z_1, (b, b), (N, N)) \\ \delta(z_1, (b, \square)) &= (z_1, (\square, b), (R, L)) \\ \delta(z_1, (\square, a)) &= (z_1, (\square, a), (N, N)) \\ \delta(z_1, (\square, b)) &= (z_1, (\square, b), (N, N)) \\ \delta(z_1, (\square, \square)) &= (z_E, (\square, \square), (N, N))\end{aligned}$$

$z_E$  ist der Endzustand. Hier passiert nichts.

$$\begin{aligned}\delta(z_E, (a, a)) &= (z_E, (a, a), (N, N)) \\ \delta(z_E, (a, b)) &= (z_E, (a, b), (N, N)) \\ \delta(z_E, (a, \square)) &= (z_E, (a, \square), (N, N)) \\ \delta(z_E, (b, a)) &= (z_E, (b, a), (N, N)) \\ \delta(z_E, (b, b)) &= (z_E, (b, b), (N, N)) \\ \delta(z_E, (b, \square)) &= (z_E, (\square, b), (N, N)) \\ \delta(z_E, (\square, a)) &= (z_E, (\square, a), (N, N)) \\ \delta(z_E, (\square, b)) &= (z_E, (\square, b), (N, N)) \\ \delta(z_E, (\square, \square)) &= (z_E, (\square, \square), (N, N))\end{aligned}$$

### Aufgabe 12 Lineare Bandbeschränkung

Damit eine  $c \cdot n$  lange Bandbeschriftung auf eine Beschriftung der Länge  $n$  umgebaut werden 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:  $\square \mid a \mid b \mid b \mid a \mid a \mid a \mid \square \Rightarrow \square a \mid bb \mid aa \mid a \square$ . Das ursprüngliche Bandalphabet ( $\Gamma = \{a, b, \square\}$ ) wird zu  $\Gamma' = \{aa, ab, a\square, ba, bb, b\square, \square a, \square b, \square\square\}$  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  $|\Gamma|^{c-1}$ .

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

$x_0 = x_1$ ;  
 $M_1 : \text{IF } x_2 = 0 \text{ THEN GOTO } M_2$ ;  
 $x_0 = x_0 + 1$ ;  
 $x_2 = x_2 - 1$ ;  
 $\text{GOTO } M_1$ ;  
 $M_2 : \text{HALT}$ ;

Das Ergebnis  $x_0 = a$  gesetzt.  
 Fall  $x_2 = 0$  ist muss nichts addiert werden.  
 Das Ergebnis  $x_0$  wird um Eins erhöht.  
 Die Eingabe  $x_2$  wird um Eins reduziert.  
 Sprung zum Check für  $x_2 = 0$ .

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

$x_0 = x_1$ ;  
 $M_1 : \text{IF } x_2 = 0 \text{ THEN GOTO } M_2$ ;  
 $x_0 = x_0 - 1$ ;  
 $x_2 = x_2 - 1$ ;  
 $\text{IF } x_0 = 0 \text{ THEN GOTO } M_2$ ;  
 $\text{GOTO } M_1$ ;  
 $M_2 : \text{HALT}$ ;

Die Ausgabe  $x_0 = a$  gesetzt.  
 Falls  $x_2 = 0$  ist muss nichts subtrahiert werden.  
 Das Ergebnis  $x_0$  wird um Eins reduziert.  
 Die Eingabe  $x_2$  wird um Eins reduziert.  
 Abbruch, da es keine Zahlen  $< 0$  gibt.  
 Sprung zum Check für  $x_2 = 0$ .

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

$x_3 = f_2(x_1, x_2)$ ;  
 $\text{IF } x_3 = 0 \text{ THEN GOTO } M_1$ ;  
 $x_0 = x_2$ ;  
 $\text{GOTO } M_2$ ;  
 $M_1 : x_0 = x_1$ ;  
 $M_2 : \text{HALT}$ ;

Die Differenz von  $x_1$  und  $x_2$  wird gebildet.  
 Wenn  $x_3 = 0$  ist, ist  $x_2 \geq x_1$ , also mit  $M_1$  weiter.  
 $x_2$  ist die kleinere Zahl.  
 Sprung zum Programmende.  
 $x_1$  ist die kleinere Zahl.

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

$M_1 : x_0 = f_2(x_1, x_2)$ ;  
 $\text{IF } x_0 = 0 \text{ THEN GOTO } M_2$ ;  
 $x_1 = x_0$ ;  
 $\text{GOTO } M_1$ ;  
 $M_2 : x_4 = f_2(x_1, x_2)$   
 $x_5 = f_2(x_2, x_1)$   
 $\text{IF } x_4 = 0 \text{ THEN GOTO } M_3$ ;  
 $\text{GOTO } M_5$ ;  
 $M_3 : \text{IF } x_5 = 0 \text{ THEN GOTO } M_4$ ;  
 $x_0 = x_1$ ;  
 $\text{GOTO } M_5$ ;  
 $M_4 : x_0 = 0$ ;  
 $M_5 : \text{HALT}$ ;

Die Differenz von  $x_1$  und  $x_2$  wird gebildet.  
 Abbruchbedingung. Differenz = 0.  
 Die Differenz ( $> x_2$ ) ist das neue a.  
 Sprung zurück.  
 Differenz  $x_1$  und  $x_2$ .  
 Differenz  $x_2$  und  $x_1$ .  
 Falls  $x_1 - x_2 = 0$  ist ...  
 und  $x_2 - x_1 = 0 \Rightarrow x_1 = x_2$   
 Falls  $x_1 \neq x_2$  steht in  $x_1$  der modulo ...  
 ansonsten geht der Modulo glatt auf (= 0).

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

$M_1 : \text{IF } x_1 = 0 \text{ THEN GOTO } M_3;$	Abbruchbedingung: 0 div a ist immer 0.
$x_3 = f_4(x_1, x_2);$	$x_3$ enthält $x_1 \bmod x_2$ .
$x_3 = f_2(x_1, x_3);$	Damit die Division glatt wird $x_3 = x_1 - x_3$
$M_2 : \text{IF } x_3 = 0 \text{ THEN GOTO } M_3;$	Abbruchbedingung: $x_2$ passt nicht in $x_1$ .
$x_3 = f_2(x_1, x_2);$	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.
$\text{GOTO } M_1;$	Sprung zum Check für $x_3 = 0$ .
$M_3 : \text{HALT};$	

### 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  $x_1 \neq x_2$  DO
  IF  $x_2 - x_1 = 0$  THEN
     $x_1 := x_1 - x_2$ 
  ELSE
     $x_2 := x_2 - x_1$ 
  END;
END;
 $x_0 := x_1$ 

```

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

```

IF  $x_2 = x_1$  THEN  $x_3 := 0$  ELSE  $x_3 := 1$  END;
WHILE  $x_3 \neq 0$  DO
  A;
  IF  $x_2 = x_1$  THEN  $x_3 := 0$  ELSE  $x_3 := 1$  END;
END

```

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

```

 $x_3 := x_1 - x_2;$ 
 $x_4 := x_2 - x_1;$ 
 $x_3 := x_3 + x_4;$ 
IF  $x_3 = 0$  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 Programmiersprache 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:  $M_j : \text{GOTO } M_i = \begin{cases} \text{SKIP } (i - j - 1) & , \text{ falls } i > j \\ \text{GOTOSTART; SKIP}(j - 1) & , \text{ falls } i = j \\ \text{GOTOSTART; SKIP}(j - i - 1) & , \text{ falls } i < j \end{cases}$
  - Bedingter Sprung: Die Syntax ist fast identisch, nur GOTO muss durch SKIP ersetzt werden.
  - Stopanweisung:  $\text{HALT} = \text{SKIP}(n);$ , 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:  $\text{SKIP}(k) = \begin{cases} M_i : \text{GOTO } M_{i+k+1}; & , \text{ falls } i + k + 1 < n \\ M_i : \text{HALT}; & , \text{ falls } i + k + 1 \geq n \end{cases}$   $n$  = Zeilenanzahl.
  - Bedingter Sprung: Die Syntax ist fast identisch, nur SKIP muss durch GOTO ersetzt werden.
  - Sprung zum Start:  $\text{GOTOSTART} = \text{GOTO } M_1;$

**Aufgabe 20** Totale Funktionen

Laut Aufgabe ist die Menge der gültigen Programme  $L$  rekursiv aufzählbar. Das bedeutet, dass 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 ein neues Programm definiert, 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 Widerspruch und daher kann  $h(x)$  nicht Teil der Aufzählung sein. Das widerspricht der 2. Eigenschaft der Sprache (jede totale Funktion kann mit einem Programm der Sprache beschrieben werden), weshalb die Sprache so nicht existieren kann.

**Aufgabe 21** primitiv-rekursive Funktionen

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

- (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**  $\mu$ -rekursive Funktionen

- (a) Mit dem  $\mu$ -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.  $\sqrt[n]{y} = n \Leftrightarrow y \leq n^x \Leftrightarrow y - n^x \leq 0$ . 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  $f_{min}(n, x, y)$  zuerst bei dem kleineren Parameter  $= 0$ .  
 $\mu f_{min} : \mathbb{N}_0^3 \Rightarrow \mathbb{N}_0$ ,  $\mu f_{min}(x, y) = \min\{n \mid f_{min}(n, x, y) = 0\}$   
 $f_{min} : \mathbb{N}_0^3 \Rightarrow \mathbb{N}_0$ ,  $f_{min}(n, x, y) = (1 - eq(x, n)) * (1 - eq(y, n))$

**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  $x \bmod 11 = 0$  dann ist  $x = 11n$ , also ein Vielfaches von 11. Desweiteren ist  $y = 11x$ , also ein vielfaches von 121 und damit  $y \bmod 121 = 0$ . 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  $\chi_L$  berechnet. Dabei geht die TM folgendermaße vor:
  1. Die Eingabe ist  $w \in \Sigma^*$ .
  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  $j \in \{1, \dots, |w|\}$ , sodass  $\forall (k < j \in \mathbb{N} \mid w_k = f(i)_k \wedge w_j <_{\Sigma} f(i)_j)$
  6. Lösche das Band, schreibe eine 0 und wechsel in den Endzustand.
- (b) Man kann mit  $TM_{\chi_L}$  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  $TM_{\chi_L}$  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 letzte 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

$$\begin{array}{c} ab \mid aba \mid abaa \\ a \mid ab \mid abaa \end{array}$$

- (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 insgesamt  $K^n$  verschiedenen Indexfolgen. Da man in jedem Schritt eins der K-Wortpaare auswählen kann.

2.

$$\sum_{n=1}^N \lambda(K, n) = \sum_{n=1}^N K^n = K + K^2 + \dots + K^N$$

$$s_n = K + K^2 + \dots + K^N \quad | * K \quad (1)$$

$$Ks_n = K^2 + K^3 + \dots + K^{N+1} \quad (2)$$

$$(1) - (2): s_n - Ks_n = K - K^{N+1}$$

$$s_n(1 - K) = K - K^{N+1} \quad |/(1 - K)$$

$$s_n = \frac{K - K^{N+1}}{1 - K}$$

$$\sum_{n=1}^N \lambda(K, n) = \frac{K - K^{N+1}}{1 - K}$$

**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
- $(x_1, y_1) = (\#, \#z_0b\#)$

$$1. \text{ Kopierregeln} = \{(a, a), (b, b), (\square, \square), (\#, \#)\}$$

$$2. \text{ Überführungsregeln} = \{(z_0b, z_1b), (z_E\square, z_E\square), (z_Ea, z_Ea), (z_Eb, z_Eb), \\ (z_0\square, \square z_0), (z_0a, az_0), (z_1\square, \square z_E), \\ (az_1a, z_1aa), (bz_1a, z_1ba), (\square z_1a, z_1\square a), \\ (az_1b, z_1ab), (bz_1b, z_1bb), (\square z_1b, z_1\square b), \\ (\#z_1a, \#z_1\square a), (\#z_1b, \#z_1\square b), \\ (z_E\#, z_E\square\#), \\ (z_0\#, \square z_0\#), (z_1\#, z_E\square\#)$$

$$3. \text{ Löseregeln} = \{(az_E, z_E), (z_Ea, z_E), (bz_E, z_E), (z_Eb, z_E), (\square z_E, z_E), (z_E\square, z_E)\}$$

$$4. \text{ Abschlussregeln} = \{(z_E\#\#, \#)\}$$

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

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

**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 letzte 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 zu überprüfen 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, Ausgabe 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 Widerspruch, also ist die Eigenschaft nicht entscheidbar.

**Aufgabe 35** Eine nicht entscheidbare Sprache

- (a) Es ist zu zeigen 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 zu zeigen das  $\overline{H_0} \leq \overline{P}$ :  $W_\infty$  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  $\overline{H_0} \leq \overline{P}$  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  $\forall x > N : |f(x)| \leq C |g(x)|$ .  
Versuch mit  $N = 1$ :

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

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:

$$\forall x > N_f : |f(x)| \leq C_f |g(x)|$$

$$\forall x > N_h : |h(x)| \leq C_h |g(x)|$$

Falls  $\varphi \in O(g)$  muss folgendes gelten:

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

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

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

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

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  $i \in \{1, \dots, n-1\} : x^i \leq x^n$

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

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 trifft für Eingaben auf, die nur aus Einsen bestehen. Für diesen Fall müssen insgesamt  $2^n - 1$  Dekrementierungen durchgeführt werden. Jede Dekrementation besteht aus maximal  $2n$ -Schritten (Wenn die Zahl die Form 100000... hat). Zusätzlich wird 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 höchstens polynomial viele Zeichen auf das Band schreiben kann. Insgesamt 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 Vorverarbeitung von  $C$  dazu kommt, also insgesamt  $p_f(n) + p_g(p_f(n)) + p_C(p_g(p_f(n)))$  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  $B \neq \emptyset \wedge B \neq \Sigma^*$
  - 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 charakteristische Funktion für die Sprache  $\Sigma^*$  ist  $\chi_{\Sigma^*} = 1$ , 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 überprüfen ob  $2 \leq t \ln k$  und ob  $n$  durch  $t$  teilbar ist.
- (d) Richtig. Wenn man eine Aussagenlogischeformel  $f$  in KNF negiert, dann ist  $\neg f$  in DNF. Es gilt:

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

$\Rightarrow$  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.

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

und damit ist INDEPENDENT - 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  $n = |V|$  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:  $(G, k) \in \text{INDEPENDENT - SET} \Leftrightarrow f(G, k) \in \text{VERTEX - COVER}$   
" $\Rightarrow$ ": Sei  $(G, k) \in IS$  und sei  $S$  ein IS der Größe  $k$ . Wir behaupten, dass  $V \setminus S$  ein VC der Größe

$n - k$  ist, d. h. dass für alle Kanten  $e \in E$  min ein Knoten der Kante in  $V \setminus S$  liegt.

Angenommen es gäbe eine Kante  $e' = \{v_1, v_2\} \in E$  mit  $v_1 \notin V \setminus S$  und  $v_2 \notin V \setminus S$ , 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  $V \setminus C$  ein IS der Größe  $k$ , d. h. es gibt keine Kanten zwischen je 2 Knoten aus  $V \setminus C$ . Angenommen es gäbe eine Kante mit  $e = v_1, v_2 \in E$  mit  $v_1, v_2 \in V \setminus C$ , d.h.  $v_1, v_2 \notin C$ . Eine solche Kante 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 zu raten 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