Aufgabe 1 *Turing-Akzeptierbarkeit*

Die Turingmaschine soll für $\Sigma = \{a, b\}$ die Sprache $L_2 = \{\omega\omega | \omega \in \Sigma^*\}$ akzeptieren.

Konstruktion

Ich verwende im Folgenden eine nichtdeterministische 1-Band-Turing-Maschine.
Zunächst wird das erste Zeichen des Eingabewortes durch ein Blank ersetzt. Abhängig vom gelesenen Zeichen geht die TM entweder in einen Zustand über in dem sie nach rechts läuft und ein "a" sucht oder in einen Zustand in den sie nach rechts läuft und ein "b" sucht. Findet die TM das gesuchte Zeichen, so wird diese entweder überlesen und das nächste Vorkommen gesucht, oder die TM rät, dass es sich um das erste Zeichen der zweiten Worthälfte handelt. Zeichen die nicht dem gesuchten Zeichen etsprechen werden überlesen. Trifft die TM auf ein Blank, so wird das Wort nicht akzeptiert. Hat die TM das Zeichen als erste Zeichen des zweiten Worthälfte erkannt, so wird diese durch ein "X" ersetzt und nach links gelaufen, bis ein Blank erreicht wird. Anschlieend macht die TM einen Schritt nach rechts und liest das Zeichen an dieser Position. Handelt es sich um ein "X", d.h. die erste Worthälfte ist vollständig entfernt, so prüeft die TM, ob die zweite Worthälfte vollständig in "X" umgewandelt wurde. Ist dieses der Fall wird das Wort akzeptiert. Handelt es sich bei dem gelesenen Zeichen um ein "a" oder "b", so sucht die TM nach dem ersten Zeichen das auf die bereits in "X" umgewandelten Zeichen folgt. Entspricht dieses dem gelesenen Zeichen, so wird es durch ein "X" überschrieben und die TM läuft zum ersten Zeichen zurück und begint erneut. Entspricht das Zeichen nicht dem gelesenen Zeichen, so wird das Wort nicht akzeptiert.

Definition

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine 1-Band-Turing-Maschine, mit
$Z = \{z_0, z_a, z_b, z_2, z_3, z_{a'}, z_{b'}, z_4, z_E\}$,
$E = \{z_E\}$,
$\Sigma = \{a, b\}$,
$\Gamma = \{a, b, X, \square\}$,
und sei δ definiert durch:

Bei leerem Ausgangswortwort direkt in den Endzustand übergehen.
$\delta(z_0, \square) = \{(z_E, \square, N)\}$

Erste Zeichen des Wortes lesen und weiteres Verhalten bestimmen.
$\delta(z_0, a) = \{(z_a, \square, R)\}$
$\delta(z_0, b) = \{(z_b, \square, R)\}$

Erraten des Anfangs der zweiten Worthälfte. Wobei diese mit einem "a" beginnen muss.
$\delta(z_a, a) = \{(z_a, a, R)\}$
$\delta(z_a, b) = \{(z_a, b, R)\}$
$\delta(z_a, a) = \{(z_2, X, L)\}$

Erraten des Anfangs der zweiten Worthälfte. Wobei diese mit einem "b" beginnen muss.
$\delta(z_b, a) = \{(z_b, a, R)\}$
$\delta(z_b, b) = \{(z_b, b, R)\}$
$\delta(z_b, b) = \{(z_2, X, L)\}$

Nach links laufen bis zum ersten Zeichen des Wortes. $\delta(z_2, \gamma) = \{(z_2, \gamma, L)\}$ mit $\gamma \in \{a, b, X\}$
$\delta(z_2, \square) = \{(z_3, \square, R)\}$

Erste Zeichen des Wortes lesen und weiteres Verhalten bestimmen.
$\delta(z_3, a) = \{(z_{a'}, \square, R)\}$
$\delta(z_3, b) = \{(z_{b'}, \square, R)\}$
$\delta(z_3, X) = \{(z_4, X, R)\}$

Prüfen ob das erste noch nicht umgewandelte Zeichen der zweiten Worthälfte ein "a" ist. Ggf. das Zeichen durch "X" überschreiben.
$\delta(z_{a'}, X) = \{(z_{a'}, X, R)\}$
$\delta(z_{a'}, a) = \{(z_2, X, L)\}$

Prüfen ob das erste noch nicht umgewandelte Zeichen der zweiten Worthälfte ein “b“ ist. Ggf. das Zeichen durch “X“ überschreiben.
$\delta(z_{b'}, X) = \{(z_{b'}, X, R)\}$
$\delta(z_{b'}, b) = \{(z_2, X, L)\}$

Prüfen ob auch die rechte Worthälfte vollständig umgewandelt wurde.
$\delta(z_4, X) = \{(z_4, X, R)\}$
$\delta(z_4, \square) = \{(z, \square, N)\}$

$\delta(...) = \emptyset$ sonst.

Beispielableitung

$z_0abab \vdash$ (mit $\delta(z_0, a) = \{(z_a, \square, R)\}$)
$\square z_abab \vdash$ (mit $\delta(z_a, b) = \{(z_a, b, R)\}$)
$\square bz_aab \vdash$ (mit $\delta(z_a, a) = \{(z_2, X, L)\}$)
$\square z_2Xb \vdash$ (mit $\delta(z_2, b) = \{(z_2, b, L)\}$)
$z_2\square bXb \vdash$ (mit $\delta(z_2, \square) = \{(z_3, \square, R)\}$)
$\square z_3bXb \vdash$ (mit $\delta(z_3, b) = \{(z_{b'}, \square, R)\}$)
$\square\square z_{b'}Xb \vdash$ (mit $\delta(z_{b'}, X) = \{(z_{b'}, X, R)\}$)
$\square\square Xz_{b'}b \vdash$ (mit $\delta(z_{b'}, b) = \{(z_2, X, L)\}$)
$\square\square z_2XX \vdash$ (mit $\delta(z_2, X) = \{(z_2, X, L)\}$)
$\square z_2\square XX \vdash$ (mit $\delta(z_2, \square) = \{(z_3, \square, R)\}$)
$\square\square z_3XX \vdash$ (mit $\delta(z_3, X) = \{(z_4, X, R)\}$)
$\square\square Xz_4X \vdash$ (mit $\delta(z_4, X) = \{(z_4, X, R)\}$)
$\square\square XXz_4\square \vdash$ (mit $\delta(z_4, \square) = \{(z_E, \square, N)\}$)
$\square\square XXz_E\square$

Aufgabe 3 *Sprachen*

Zu Zeigen:

Es gibt nicht zu jeder Sprache eine Grammatik, welche diese erzeugt.

Beweis:

Eine Grammatik besteht nach Definition aus vier endlichen Mengen. Es handelt sich also um ein endliches Objekt. Folglich muss die Menge aller Grammatiken abzählbar sein. Man kann z.B. die Objekte aller Mengen der Grammatik konkartinieren und die so entstehenden Wörter zunächt nach Länge und dann innerhalb einer Längenklasse lexikographisch aufzählen.

Die Menge aller Sprachen ist die Potenzmenger der Menge aller Wörter. Da es sich bei der Menge aller Wörte bereits um eine abzählbar unendliche Menge handelt folgt, die Menge aller Sprachen muss überabzählbar sein. (vgl. Schning S. 189)

Aus beiden oben genanten Eigenschaften folgt, die Menge aller Grammatiken hat eine geringer Kardinalität als die Menge aller Sprachen. Es muss also Sprachen geben, zu denen es keine Grammatik gibt.
Es kann also **nicht** gelten, dass es für jede Sprache $L \subseteq \Sigma^*$ eine Grammatik G exisiert, für die gilt $L(G) = L$.

q.e.d.

Aufgabe 4 *Weihnachts-Bonus-Aufgabe*

Definition

Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ eine 5-Band Touringmaschine:
$Z = \{z_0, z_1, z_2, z_3, z_4\}$
$\Sigma = \{0, ..., 255, <, >, +, -, ,, ., [,]\}$
$\Gamma = \{0, ..., 255, <, >, +, -, ,, ., [,], \square\}$
$E = \{z_e\}$

und δ:

wobei jeweils gelte: $\gamma_i \in \Gamma$ mit $i \in \{1, ..., 5\}$

1.) Pointer dekrementieren: $<$
$\delta(z_0, <, \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_0, <, \gamma_2, \gamma_3, \gamma_4, \gamma_5, R, L, N, N, N)\}$

2.) Pointer inkrementieren: $>$
$\delta(z_0, >, \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_0, >, \gamma_2, \gamma_3, \gamma_4, \gamma_5, R, R, N, N, N)\}$

3.) Aktuelle Zelle inkrementieren: $+$
$\delta(z_0, +, \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_0, +, \gamma_2 + 1, \gamma_3, \gamma_4, \gamma_5, R, N, N, N, N)\}$

4.) Aktuelle Zelle dekrementieren: $-$
$\delta(z_0, -, \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_0, -, \gamma_2 - 1, \gamma_3, \gamma_4, \gamma_5, R, N, N, N, N)\}$

5.) Gib den Wert an der aktuellen Position aus: .
$\delta(z_0, ., \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_0, ., \gamma_2, \gamma_3, \gamma_2, \gamma_5, R, N, N, R, N)\}$
$\gamma_2 \neq \square$

$\delta(z_0, ., \square, \gamma_3, \gamma_4, \gamma_5) = \{(z_0, ., \square, \gamma_3, 0, \gamma_5, R, N, N, R, N)\}$

6.) Lese Eingabe und speicher sie an aktueller Position: ,
$\delta(z_0, ,, \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_0, ,, \gamma_3, \gamma_3, \gamma_4, \gamma_5, R, N, R, N, N)\}$

7.) Springe hinter die zugehörige]: [
$\delta(z_0, [, \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_0, [, \gamma_2, \gamma_3, \gamma_4, \gamma_5, R, N, N, N, N)\}$
$\gamma_2 \neq 0$

$\delta(z_0, [, 0, \gamma_3, \gamma_4, \gamma_5) = \{(z_1, [, 0, \gamma_3, \gamma_4, \gamma_5, R, N, N, N, N)\}$

$\delta(z_1, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_1, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, R, N, N, N, N)\}$
$\gamma_1 \notin \{[,]\}$

$\delta(z_1, [, \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_1, [, \gamma_2, \gamma_3, \gamma_4, [, R, N, N, N, R)\}$

$\delta(z_1,], \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_2,], \gamma_2, \gamma_3, \gamma_4, \gamma_5, N, N, N, N, L)\}$

$\delta(z_2, \gamma_1, \gamma_2, \gamma_3, \gamma_4, [) = \{(z_1, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \square, R, N, N, N, N)\}$

$\delta(z_2, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \square) = \{(z_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \square, R, N, N, N, N)\}$

8.) Springe zur zugehörigen [:]
$\delta(z_0,], \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_3,], \gamma_2, \gamma_3, \gamma_4, \gamma_5, L, N, N, N, N)\}$

$\delta(z_3, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_3, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, L, N, N, N, N)\}$
$\gamma_1 \notin \{[,]\}$

$\delta(z_3, [, \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_3, [, \gamma_2, \gamma_3, \gamma_4,], L, N, N, N, R)\}$

$\delta(z_3,], \gamma_2, \gamma_3, \gamma_4, \gamma_5) = \{(z_4,], \gamma_2, \gamma_3, \gamma_4, \gamma_5, N, N, N, N, L)\}$

$\delta(z_4, \gamma_1, \gamma_2, \gamma_3, \gamma_4,]) = \{(z_3, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \square, L, N, N, N, N)\}$

$\delta(z_4, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \square) = \{(z_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \square, N, N, N, N, N)\}$

$\delta(...) = \emptyset$ sonst.

Bei allen Berechnungen, sowie der Eingabe wird □ als 0 betrachtet. Auf dem Ausgabeband entspricht □ nicht 0, sondern keiner Ausgabe.

zu 1.) Wenn auf dem ersten Band das Zeichen < gelesen wird, dann wird der Lesekopf auf dem zweiten Band nach links verschoben.

zu 2.) Wenn auf dem ersten Band das Zeichen > gelesen wird, dann wird der Lesekopf auf dem zweiten Band nach rechts verschoben.

zu 3.) Wenn auf dem ersten Band das Zeichen + gelesen wird, dann wird der Wert an der aktuellen Stelle auf Band zwei um 1 inkrementiert.

zu 4.) Wenn auf dem ersten Band das Zeichen − gelesen wird, dann wird der Wert an der aktuellen Stelle auf Band zwei um 1 dekrementiert.

zu 5.) Der Wert im an der aktuellen Position auf Band 2 wird auf das Ausgabeband 4 gespeichert.
Besondere Übergangsrelation für □ um bei der Ausgabe eine Unterscheidung von 0 und keiner Ausgabe zu ermöglichen.

zu 6.) Der aktuelle Wert auf Band 3 wird im Array auf Band 2 abgespeichert.

zu 7.) Wird auf Band 1 eine [gelesen, überprüft die Maschine ob auf Band 3 eine 0 steht. Steht dort keine Null, bleibt die Maschine in Zustand z_0 und luft normal weiter. Steht dort eine Null, so wandert die Maschine bis hinter die passende nchste] auf Band 1. Dieses tut sie, in dem sie in Zustand z_1 wechselt. Sie luft denn Zeichen für Zeichen auf Band 1 weiter so lange sie kein [oder] findet. Findet sie ein [so schreibt sie auf das letzte Band das selbe Symbol um sich zu merken dass eine innere Klammerung existiert. Sollte noch eine weitere [gelesen werden, so schreibt sie wieder das selbe Symbol auf das letzte Band an einer Stelle rechts neben dem vorherigen Symbol. Liest die Maschine ein] auf dem ersten Band so wechselt sie in Zustand z_2 und schiebt den letzten Lesekopf einen nach links. Sollte an dieser Stelle nun eine [stehen, dann wird dieses Symbol gelöscht, der erste Lesekopf wird nach rechts verschoben und die Maschine wechselt wieder in Zustand z_1. Sollte jedoch keine [gefunden werden so wechselt die Maschine in Zustand z_0 und rückt den ersten Lesekopf weiter nach rechts.

zu 8.) Dieser Befehl läuft ähnlich wie der 7te ab. Wird eine] gelesen, so wechselt die Maschine in Zustand z_3 und wandert bis zur entsprechenden [nach links auf dem ersten Band. Sollte innerhalb dieser Klammern eine weitere Klammerung existieren, so wird für jede weitere Klammerung ein Symbol] auf dem letzten Band angelegt. Dieses Symbol wird jeweils durch die passende Gegenklammer [wieder geloescht. Sollte auf dem untersten Band kein Symbol] existieren und auf dem ersten Band ein [gelesen werden, so wechselt die Maschine zurück in Zustand z_0 und bewegt den ersten Lesekopf nicht weiter, damit aufgrund der Klammerung die Bedingung der Schleife überprüft werden kann.

Die Implementierung entspricht den Bedingungen des auf http://www.muppetlabs.com/ breadbox/bf/standards.html definierten Sprachstandards.