Automatentheorie

Grammatiken Typ-1, Typ-0 und Turingmaschinen

Prof. Dr. Franz-Karl Schmatzer
schmatzf@dhbw-loerrach.de

Literatur

- C.Wagenknecht, M.Hielscher; Formale Sprachen, abstrakte Automaten und Compiler; 3.Aufl. Springer Vieweg 2022;
- Sipser M.; Introduction to the Theory of Computation; 2.Aufl.; Thomson Course Technology 2006
- Hopecroft, T. et al; Introduction to Automata Theory, Language, and Computation; 3. Aufl. Pearson Verlag 2006
- Vossen,G. Witt K.; Grundkurs Theoretische Informatik; 4.Aufl.; Vieweg Verlag 2006
- Cohen, D; Introduction to Computer Theory; John Wiley 1990

Agenda

- kontextsensitive Grammatiken
- Rekursiv-aufzählbare Sprachen
- Turingmaschinen
- Übersicht über die Sprachklassen

Kontextsensitive Sprache
Einführung

- Betrachte die Sprache L_{KS}

  $L_{KS} = \{a^n b^n c^n \mid n \geq 0\}$

  Dies Sprache gehört nicht zu den kontextfreien Sprachen.
- Um diese Sprache abzuleiten, braucht man kontextabhängige Regeln.
- Aufheben dieser Beschränkung

Kontextsensitive Grammatik
Beispiel

- Beispiel für L_{KS}
  - $S \rightarrow aSBC \mid aBC \mid \varepsilon$
  - d.h. $S \Rightarrow^* a^n(BC)^n$
  - Nun hat man die richtige Anzahl der a‘s, b‘s und c‘s.
  - Um die richtige Reihenfolge B^nC^n zu erhalten braucht man die Regel
    - $CB \rightarrow BC$
  - und um die richtige Reihenfolge der B‘c und C‘s zu gewähren die kontextsensitiven Regeln
    - $aB \rightarrow ab$,
    - $bB \rightarrow bb$,
    - $bC \rightarrow bc$,
    - $cC \rightarrow cc$
  - Damit $G_{KS} = (\{S,B,C\}, \{a,b,c\}, P,S)$ erzeugt die Sprache L_{KS} mit

    $P = \{S \rightarrow aSBC \mid aBC , CB \rightarrow BC, aB \rightarrow ab, bB \rightarrow bb, bC \rightarrow bc, cC \rightarrow cc\}$

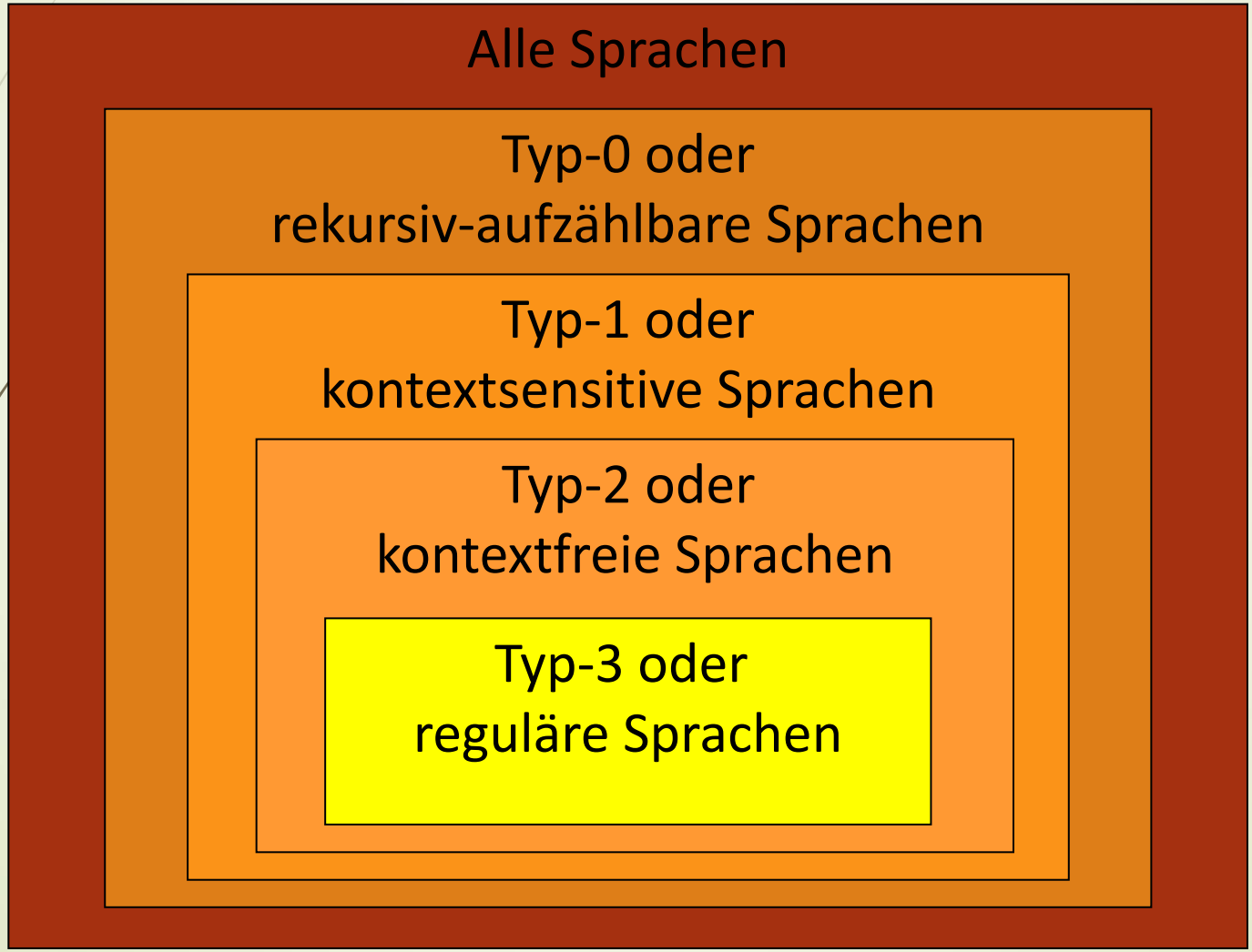
Kontextsensitive Grammatik
Definition

- Eine Grammatik G = (N,Σ,P,S) mit N, Σ und S definiert wie zuvor, heißt kontextsensitive Grammatik oder Typ-1-Grammatik, wenn
  - $P \subseteq ((\Sigma \cup N)^* - \Sigma^*) \times (\Sigma \cup N)^*$ und $|P| < \infty$
  - und mit $l \rightarrow r \in P \Rightarrow |l| \leq |r|$
  - d.h. die Anzahl der Symbole links ist kleiner gleich der Anzahl der Symbole rechts. (Monotonie der Sprache)
- Eine Sprache L heißt kontextsensitiv über Σ, falls es eine kontextsensitive Grammatik G über Σ gibt mit L = L(G)

Rekursiv-aufzählbare Sprachen (Typ-0-Grammatiken)

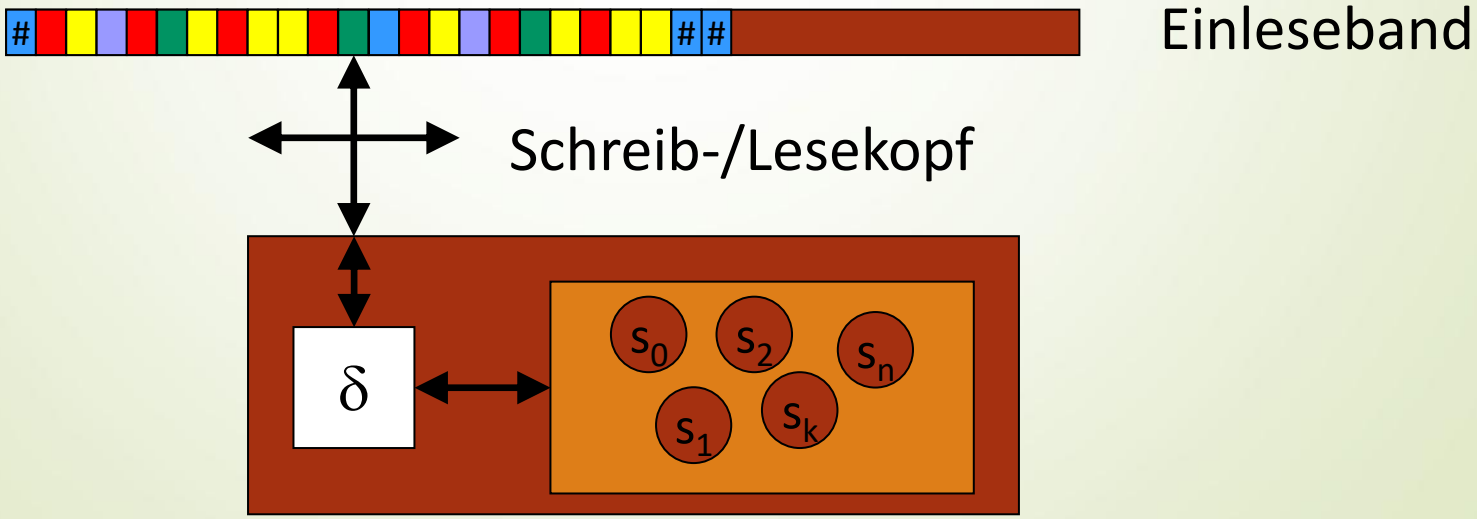
- Wenn man die Monotonie der Ableitungsregeln $|l| \leq |r|$ aufgibt, d.h. die linke Seite kann auch mehr Symbole als die rechte Seite haben, kommt man zu einer weiteren Klasse von Sprachen.
- Die rekursiv-aufzählbare Sprachen G_R
- Eine Grammatik $G = (N,\Sigma,P,S)$ mit N, Σ und S definiert wie zuvor, heißt Typ-0-Grammatik bzw. rekursiv-aufzählbar, wenn
  - $P \subseteq ((\Sigma \cup N)^* - \Sigma^*) \times (\Sigma \cup N)^*$ und $|P| < \infty$

Chomsky-Hierarchie



Turing-Automat

- Erweitert man den Kellerautomaten dahingehend, dass man wahlfrei auf den Speicher sowohl lesend als auch schreibend zugreifen kann, gelangt man zu den Turing-Automaten bzw. Turingmaschinen.
- Statt des Speichers nimmt man das Eingabeband selbst auf das man schreibend und lesend zugreifen kann.
- Man benötigt noch Steuerungszeichen für den Schreib-/Lesekopf
- Modell Turingmaschine:



Turing-Automat

Einführung formal

- Sei $T = (Q, \Sigma, \Gamma, \delta, s_0, \#, F)$ eine (deterministische) Turingmaschine.

  $Q = \{s_0, s_1, \ , s_n\}$ eine nicht leere Menge von Zuständen.

  Σ = eine nicht leere Menge von Zeichen, das Eingabealphabet.

  Γ = eine nicht leere Menge von Zeichen mit $\Sigma \subseteq \Gamma$, das Bandalphabet.

  δ : eine partielle Funktion $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R,N\}$, die Übergangsfunktion

  $s_0 \in Q$ der Anfangszustand

  $\#$ ein Blanksymbol mit $\# \in \Gamma \setminus \Sigma$

  $F \subseteq Q$ die nicht leere Menge von Endzustände

Turing-Automat

Arbeitsweise

- Das zu lesende Wort steht auf dem Band. Begrenzt durch das Blanksymbol #
- Die Maschine kann in jedem Schritt das gelesene Zeichen durch ein eigenes Zeichen überschreiben, dann den Zustand wechseln und den Kopf still halten N, oder ein Zeichen nach links L oder nach rechts R bewegen.
- Die Überführungsfunktion δ kann man so interpretieren:

  δ(q, o) = (q‘, a, L) : Im Zustand q wird das Zeichen o gelesen. Von dort wird in den Zustand q‘ übergegangen, o mit a überschrieben und der Kopf nach links (L) bewegt.

  δ(q, o) = (q‘, a, R) : Der Kopf wird nach rechts bewegt.

  δ(q, o) = (q‘, a, N) : Der Kopf wird nicht bewegt.
- Sobald ein Endzustand erreicht wird, hält die Maschine.
- Neben der Terminierung durch Endzustände hält eine deterministische Turingmaschine auch bei $\delta(q,a) = \perp$ für den aktuellen Zustand q und das gelesene Zeichen a. Gehört q zu den Endzuständen, wird das Wort akzeptiert ansonsten verworfen.

Turing-Automat

Beispiel 1

- Es soll eine Turingmaschine konstruiert werden, die ein Wort w aus dem Alphabet $\sum = \{0,1\}$ in sein Komplement umwandelt.
- Überlegung:
  - Umwandlung wird Bitweise durchgeführt.
  - Algorithmus?
  - Eingabealphabet $\sum = \{0,1\}$.
  - Bandalphabet $\sum$ und #.
  - Zustandsmenge $Q = \{s_0, s_1, s_2\}$ mit s_0 der Startzustand und s_2 der Endzustand.
  - Algorithmus festlegen und Überführungsfunktion bestimmen

Turing-Automat

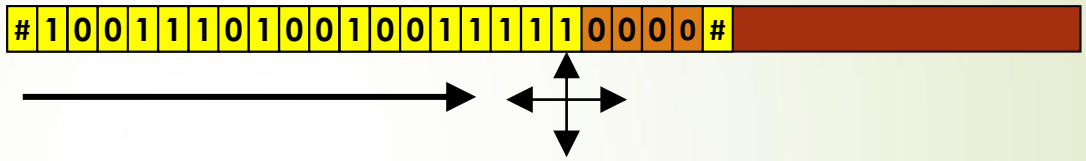
Beispiel 1:Algorithmus

- Umwandlung wird Bitweise durchgeführt.
  - 1. Schritt (Zustand 1)
    - Maschine liest das Bit von Band
    - Schreibt das Komplement auf das Band
    - Geht ein Schritt nach rechts
    - Wenn es an das Ende das Ende-Zeichen # liest, dreht sie um und geht in den nächsten Zustand
  - 2. Schritt (Zustand 2)
    - Maschine liest das Bit von Band
    - Schreibt das Bit auf das Band
    - Geht ein Schritt nach links
    - Wenn es an das Ende das Ende-Zeichen # liest, dreht sie um und geht in den Endzustand
  - 3. Schritt (Zustand 2)
    - Maschine bleibt am Anfang des Wortes stehen.

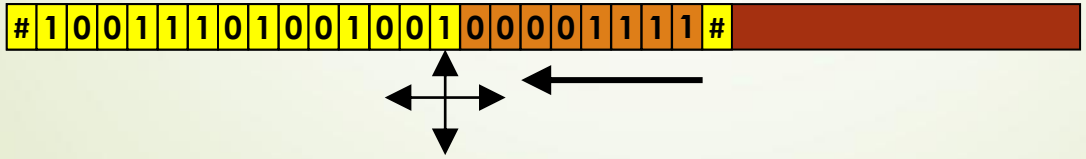
Turing-Automat

Beispiel 1 : δ-Funktion

- Mit s_0 wandert wir an das rechte Ende und wandeln 0 in 1 und 1 in 0 um. Sobald wir das # lesen, wechseln wir in den Zustand s_1.
  - $\delta(s_0,0) = (s_0,1,R)$
  - $\delta(s_0,1) = (s_0,0,R)$
  - $\delta(s_0,\#) = (s_1,\#,L)$



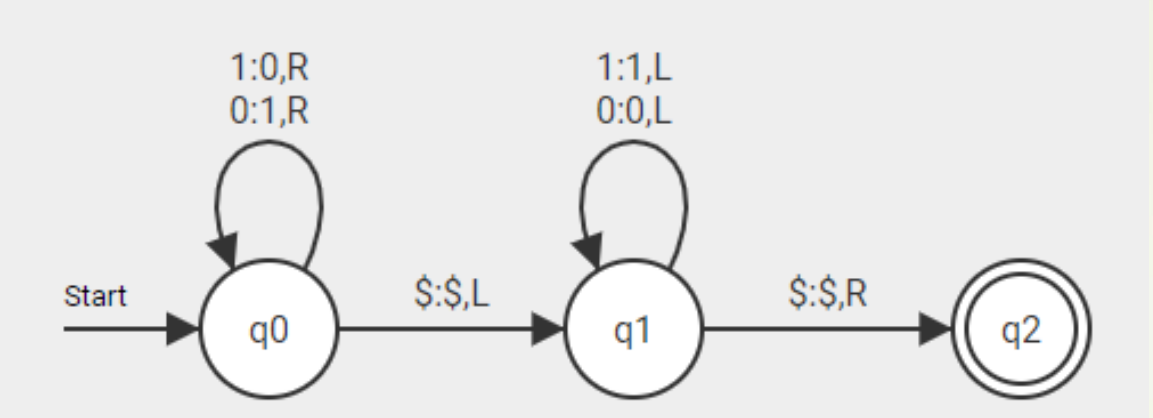
- Zustand s_1 wird für den Rücklauf genommen.
  - $\delta(s_1,1) = (s_1,1,L)$
  - $\delta(s_1,0) = (s_2,0,L)$
  - $\delta(s_1,\#) = (s_2,\#,N)$



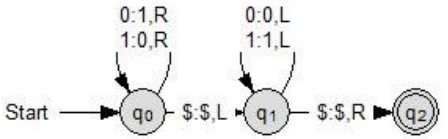
Turing-Automat

Beispiel 1

- TM für Komplementbildung in FLACI konstruiert



Automaton

Type:	TM
Transition Graph:	
Definition:	$M = (\{q_0,q_1,q_2\},\{0,1\},\{\$,0,1\},\delta,q_0,\$,\{q_2\})$

Transitions:

δ	$	0	1
q_0	$(q_1,\$,L)$	$(q_0,1,R)$	$(q_0,0,R)$
q_1	$(q_2,\$,R)$	$(q_1,0,L)$	$(q_1,1,L)$
q_2	-	-	-

Turing-Automat

Aufgabe 1

- Es soll ein Turingmaschine konstruiert werden, die eine 1 auf eine Zahl in binärschreibweise addiert.
- Das Wort $w = a_{r-1} \ldots a_1 \ldots a_0$ mit $a_i \in \{0,1\}$ und $0 \leq i < r$ ist die Binärdarstellung der Zahl $n = \sum a_i 2^i$ (Summe über $0 \leq i < r$)
- Überlegung:
  - Addition wird bitweise ausgeführt.
  - Eingabealphabet $\sum = \{0,1\}$.
  - Bandalphabet $\sum$ und #.
  - Zustandsmenge $Q = \{s_0, s_1, s_2, s_3\}$ mit s_0 der Startzustand und s_3 der Endzustand.
  - Geben Sie den Algorithmus an und dann die Überführungsfunktion. Testen Sie den Automaten mit FLACI

Turing-Automat

Aufgabe 1: Algorithmus 1

- 1. Schritt (Zustand 1)
  - Maschine liest das Bit von Band
  - Schreibt das Bit auf das Band
  - Geht ein Schritt nach rechts
  - Wenn es an das Ende das Ende-Zeichen # liest, dreht sie um und geht in den nächsten Zustand
- 2. Schritt (Zustand 2)
  - Maschine liest das Bit von Band
  - Wenn es eine 1 ist
    - Wird eine 0 auf das Band geschrieben
    - Sie geht ein Schritt nach links
  - Wenn es eine 0 ist
    - Wird eine 1 auf das Band geschrieben
    - Sie geht ein Schritt nach links
    - Sie wechselt in den Zustand 3

Turing-Automat

Aufgabe 1: Algorithmus 2

- 3. Schritt (Zustand 3)
  - Maschine liest das Bit von Band
  - Schreibt das Komplement auf das Band
  - Geht ein Schritt nach rechts
  - Wenn es an das Ende das Ende-Zeichen # liest, dreht sie um und geht in den nächsten Zustand
- 3. Schritt (Zustand 2)
  - Maschine bleibt am Anfang des Wortes stehen.

Turing-Automat

Aufgabe 2

- Es soll eine Turingmaschine konstruiert werden, die nur Worte akzeptiert, die genau so viele 1 wie 0-Zeichen enthalten.
- $L(A) = \{w \in \{0,1\}^* \mid \text{ mit } |w|_0 = |w|_1\}$.
- Überlegung:
  - Maschine läuft das Wort entlang und markiert in jedem Schritt eine 0 und eine 1. Am Schluss dürfen keine 0 oder 1 Zeichen übrigbleiben.
  - Eingabealphabet $\Sigma = \{0,1\}$.
  - Bandalphabet Σ und # und * zur Markierung.
  - Zustandsmenge $Q = \{s_0,s_1,s_2,s_3,s_4,s_5,s_6,s_7\}$ mit s_0 der Startzustand und s_7 der Endzustand.
  - Die Überführungsfunktion δ definieren wir wie folgt.

Turing-Automat

Aufgabe 2: Algorithmus 1

- 1. Schritt (Zustand 0)
  - Maschine liest das erste Bit von Band
  - Wenn es eine 1 war wird das Bit markiert, die Maschine geht einen Schritt nach rechts und in den Zustand 1
  - Wenn es eine 0 war wird das Bit markiert, die Maschine geht einen Schritt nach rechts und in den Zustand 2
  - Wenn es das Ende-Zeichen # liest, dreht sie um und geht in den Zustand 5
- 2. Schritt (Zustand 1) das zweite Zeichen muss eine 0 sein
  - Maschine liest das Bit von Band
  - Wenn es eine 1 ist
    - Wird eine 1 auf das Band geschrieben
    - Sie geht ein Schritt nach rechts
  - Wenn es ein * ist
    - Wird ein * auf das Band geschrieben
    - Sie geht ein Schritt nach rechts
  - Wenn es eine 0 ist
    - Wird ein * auf das Band geschrieben
    - Sie geht ein Schritt nach rechts
    - Sie wechselt in den Zustand 3

Turing-Automat

Aufgabe 2: Algorithmus 2

- 2. Schritt (Zustand 2) das zweite Zeichen muss eine 1 sein
  - Maschine liest das Bit von Band
  - Wenn es eine 0 ist
    - Wird eine 0 auf das Band geschrieben
    - Sie geht ein Schritt nach rechts
  - Wenn es ein * ist
    - Wird ein * auf das Band geschrieben
    - Sie geht ein Schritt nach rechts
  - Wenn es eine 1 ist
    - Wird ein * auf das Band geschrieben
    - Sie geht ein Schritt nach rechts
    - Sie wechselt in den Zustand 3

Turing-Automat

Aufgabe 2: Algorithmus 3

- 3. Schritt (Zustand 3) // Ende des Wortes abarbeiten
  - Maschine liest die restlichen Bits von Band
  - Wenn es an das Ende das Ende-Zeichen # liest, dreht sie um und geht in den Zustand 4
- 4. Schritt (Zustand 4) // An den Anfang gehen
  - Maschine geht an den Anfang des Wortes und dreht dort um und geht in den Zustand 0.
- 5. Schritt (Zustand 5)
  - Wenn das Zeichen ein * ist geht sie ein Schritt nach links
  - Wenn es ein # ist dreht die Maschine um und geht in den Zustand 6
- 6. Schritt (Zustand 7)
  - Das Wort wird akzeptiert

Turing-Automat

Aufgabe 3

- Es soll eine Turingmaschine konstruiert werden, die nur Worte akzeptiert, die genau so viele 0-, 1- und 2-Zeichen enthalten.
- $L(A) = \{0^n 1^n 2^n \mid \text{mit } n \geq 0\}$.
- Überlegung:
  - Maschine läuft das Wort entlang und markiert in jedem Schritt eine 0 und eine 1 und eine 2. Am Schluss dürfen keine 0, 1 oder 2 Zeichen übrigbleiben.
  - Eingabealphabet $\sum = \{0,1,2\}$.
  - Bandalphabet $\sum$ und # und * zur Markierung.
  - Zustandsmenge $Q = \{s_0,s_1,s_2,s_3,s_4,s_5,s_6,s_7\}$ mit s_0 der Startzustand und s_7 der Endzustand.
  - Die Überführungsfunktion δ definieren wir wie folgt.

Turing-Automat

Aufgabe 3: Algorithmus 1

- 1. Schritt (Zustand 0)
  - Wenn die Maschine ein * liest, geht sie nach rechts und das Zeichen * wird geschrieben.
  - Wenn die Maschine eine 0 liest, geht sie nach rechts und das Zeichen * wird geschrieben sowie in den Zustand 1 gesprungen.
  - Wenn es das Ende-Zeichen # liest, dreht sie um und geht in den Zustand 5
- 2. Schritt (Zustand 1) das zweite Zeichen muss eine 1 sein
  - Wenn die Maschine ein * liest, geht sie nach rechts und das Zeichen * wird geschrieben.
  - Wenn die Maschine eine 0 liest, schreibt sie die 0 und geht nach rechts.
  - Wenn die Maschine eine 1 liest, schreibt sie ein * geht nach rechts und wechselt in den Zustand 2.
- 3. Schritt (Zustand 2) das dritte Zeichen muss eine 2 sein
  - Wenn die Maschine ein * liest, geht sie nach rechts und das Zeichen * wird geschrieben.
  - Wenn die Maschine eine 1 liest, schreibt sie die 1 und geht nach rechts.
  - Wenn die Maschine eine 2 liest, schreibt sie ein * geht nach rechts und wechselt in den Zustand 3.

Turing-Automat

Aufgabe 3: Algorithmus

- 4. Schritt (Zustand 2)
  - Wenn die Maschine eine 2 liest, geht sie nach rechts und das Zeichen 2 wird geschrieben.
  - Wenn die Maschine eine # liest, schreibt sie ein # geht nach links und wechselt in den Zustand 4.
- 5. Schritt (Zustand 4)
  - Rücklauf zum Anfang
  - Dann zu Schritt 1
- 6. Schritt (Zustand 5)
  - Wenn das Zeichen ein * ist geht sie ein Schritt nach links
  - Wenn es ein # ist dreht die Maschine um und geht in den Zustand 6
- 6. Schritt (Zustand 7)
  - Das Wort wird akzeptiert

Turing-Automat

Sprache

- Sprache L(M) einer Turingmaschine M
  - M akzeptiert das Eingabewort w genau dann, wenn es ein $s_f \in F$ (Menge der Endzustände) sowie Worte v und u gibt mit
  - $(\#,s_0,w) \rightarrow^* (v,s_f,u)$
- Die von M akzeptierte Sprache ist:
  - $L(M) = \{ w \in \Sigma^* \mid M \text{ akzeptiert } w\}$
- Eine Sprache L heißt rekursiv aufzählbar, wenn es einen Turingmaschine M gibt, die L akzeptiert.
- Eine Sprache L heißt rekursiv, wenn es einen Turingmaschine M gibt, die L akzeptiert und die für jede Eingabe terminiert.

Turing-Automat

Äquivalenzen

- Deterministische ⇔ nicht deterministische Turingmaschine.
- Rekursive aufzählbar ⇔ Typ 0 Sprachen.
- Modifikationen von Turingmaschinen.
  - Turingmaschinen mit einseitigem Band
  - Turingmaschinen mit mehreren Bänder

Turing-Automat

linear beschränkte Automaten LBA

- Anstatt eines unendlichen Bandes, steht der Turingmaschine nur der Speicherplatz zur Eingabe zur Verfügung.
- Dies kann man erreichen, in dem das Eingabewort durch ein extra Zeichen z.B. $ begrenzt wird und die Turingmaschine nur innerhalb dieser Zeichen agieren darf.
- LBA sind äquivalent zu Typ 1 Sprachen

Übersicht über die Sprachklassen

Sprachklasse	Name	akzeptierender Automat	Erzeugende Grammatik
Typ 3	regulär	Endliche Automaten	Reguläre Grammatik
DKF	deterministisch kontextfrei	Deterministischer Kellerautomat	LR(k) Grammatiken
Typ 2	kontextfrei	Nichtdeterministische Kellerautomaten	kontextfreie Grammatik
Typ 1	kontextsensitiv	Linear beschränkte Turingmaschinen	kontextsensitive Grammatik
Typ 0	rekursiv aufzählbar	Turingmaschinen	beliebige Grammatik

Abgeschlossenheitseigenschaften der Sprachklassen

Sprachklasse	Durchschnitt	Vereinigung	Komplement	Konkatenation	Kleene-Stern
Typ 3	ja	ja	ja	ja	ja
DKF	nein	nein	ja	nein	nein
Typ 2	nein	ja	nein	ja	ja
Typ 1	ja	ja	ja	ja	ja
Typ 0	ja	ja	nein	ja	ja

Entscheidbarkeit für die Sprachklassen

Sprachklasse	Wortproblem [Komplexität]	Leerheit	Endlichkeit	Äquivalenz
Typ 3	ja [O(n)]	ja	ja	ja
DKF	ja [O(n)]	ja	ja	?
Typ 2	ja [$O(n^3)$]	ja	ja	nein
Typ 1	ja[$2^{O(n)}$]	nein	nein	nein
Typ 0	nein	nein	nein	nein