Theoretische Informatik

Teil 6
Turing-Maschinen (TM)

Frühlingssemester 2019

L. Di Caro
D. Flumini
O. Stern

zh
aw

Lernziele

- Die Studierenden können **Turing-Maschinen** formal und an Beispielen erklären.
- Die Studierenden kennen den Zusammenhang zwischen **Turing-Maschinen** und **rekursiven Sprachen**.
- Sie kennen mögliche Erweiterungen und Einschränkungen für **Turing-Maschinen**, und können diese anhand von Beispielen erklären.
- Den Studierenden ist die **Äquivalenz** der behandelten **Turing-Maschinen** (eingeschränkte und erweiterte) bekannt.

Lerninhalte

- Einführung und Definition einer Turing-Maschine
- Konfiguration und Berechnung einer Turing-Maschine
- Sprache einer Turing-Maschine
- Erweiterungen von Turing-Maschinen
  - (Turing-Maschine mit Speichern)
  - Turing-Maschine mit Spuren
  - Turing-Maschine mit mehreren Bändern
  - Nichtdeterministische Turing-Maschinen
- Einschränkungen von Turing-Maschinen
  - Turing-Maschine mit semiunendlichem Band
  - K-Stack-Maschinen
  - (Zähler-Maschinen)

Motivation

Bereits gezeigt wurde:

- $L_1 = \{0^m 1^m \mid m > 0\}$ ist *keine reguläre Sprache*
  (kann durch keinen endlichen Automaten erkannt werden)
- $L_2 = \{0^m 1^m 2^m \mid m > 0\}$ ist *keine kontextfreie Sprache*
  (kann weder durch einen endlichen Automaten noch einen nichtdeterministischen Kellerautomaten erkannt werden)

Fragen:

- Gibt es einen Automatentypen, der die Sprache L_2 erkennen kann?
- Gibt es einen Automatentypen, der **alle** Sprachen erkennen kann?
- Wie muss ein solcher Automatentyp aufgebaut sein?

Ziel: Ein Berechnungsmodell entwickeln, das so allgemein wie möglich ist, aber trotzdem einfach aufgebaut ist (unabhängig von konkreter Hardware).

Geschichte

Im **17. Jahrhundert** hat **Gottfried Leibniz** die erste mechanische Rechenmaschine gebaut, und er träumte davon, eine Maschine zu bauen, die allen mathematischen Aussagen einen Wahrheitswert zuordnen kann.

1926 formulierte **David Hilbert** das **Entscheidungsproblem**:

Gibt es einen Algorithmus, der als als Input eine logische Aussage (in einer formalen Sprache formuliert) bekommt und ausgibt, ob die Aussage wahr oder falsch ist?

Das heisst: Kann man die Wahrheit oder Falschheit jeder mathematischen Aussage automatisch bestimmen?[1]

[1] In der theoretischen Informatik ist das Entscheidungsproblem die Frage, ob ein gegebenes Wort zu einer Sprache gehört oder nicht.

Geschichte

1936 publizierten **Alonzo Church** und **Alan Turing** unabhängig voneinander zwei Papiere, die gezeigt haben, dass das Entscheidungsproblem nicht allgemein lösbar ist.

Ziele:

- Zeigen, dass ein Problem automatisch lösbar ist: Algorithmus zur Lösung dieses Problems entwickeln.
- Zeigen, dass ein Problem nicht automatisch lösbar ist: Nichtexistenz in einem genug einfachen Modell zeigen, das nur wenige elementare Operationen erlaubt, das aber die volle Berechnungsstärke (wie ein Computer) besitzt.

Einführung

In der Folge untersuchten verschiedene Mathematiker die Frage der ***„Berechnung“*** mit verschiedenen Ansätzen und Notationen.

1936 stellte A. M. Turing mit der Turing-Maschine ein Modell für *„jede mögliche Berechnung“* vor, das bis heute die Modellierung (und Entwicklung) von Computern prägte . . .

. . . und erstmalig den Begriff *„Algorithmus“* wie *„Berechenbarkeit“* versuchte formal zu erfassen.

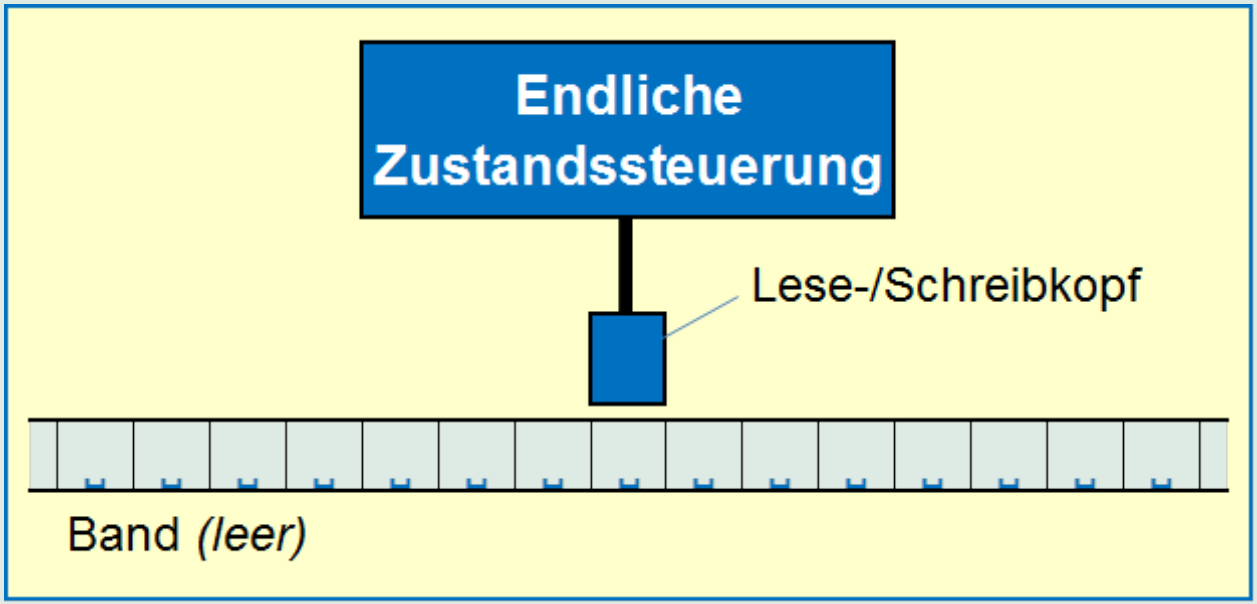
Einführung

Eine **Turing-Maschine (TM)** ist informell

- ein **endlicher Automat**, ergänzt um
- einen **Schreib-Kopf** und
- ein **unendliches Band von Zellen**, die mit dem Lese-/Schreibkopf **gelesen** und **beschrieben** werden können.

Die Eingabe steht am Anfang auf dem Band. Alle Zellen rechts und links von der Eingabe enthalten das Leerzeichen.

Beispiel (Schematische Darstellung einer TM)



Definition (Turing-Maschine)

Eine **(deterministische) Turing-Maschine** (TM) ist ein 7-Tupel

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$$

mit einer bzw. einem:

- *endlichen* Menge von **Zuständen** $Q = \{q_0, q_1, \ldots, q_n\}$ $(n \in \mathbb{N})$,
- **Eingabealphabet** $\Sigma = \{a_1, a_2, \ldots, a_m\}$ $(m \in \mathbb{N})$,
- **Übergangsfunktion** $\delta \colon Q \times \Gamma \to Q \times \Gamma \times D$, $D = \{L, R\}$,
- **Startzustand** $q_0 \in Q$,
- Menge der **akzeptierenden Zustände** $F \subseteq Q$,
- **Bandalphabet** Γ (endliche Menge von Symbolen) und $\Sigma \subset \Gamma$ und
- **Leerzeichen** $\sqcup$, mit $\sqcup \in \Gamma$ und $\sqcup \notin \Sigma$.

Definition (Turing-Maschine – *Fortsetzung*)

Die **Übergangsfunktion** δ ist eine partielle Funktion

$$\delta \colon Q \times \Gamma \to Q \times \Gamma \times D.$$

Sie bildet das **2-Tupel** (q, X) auf das **Tripel** (p, Y, D) ab:

- $q, p \in Q$ und $X, Y \in \Gamma$
- D beschreibt die Bewegung des Lese-/Schreibkopfes über dem Band. D kann die Werte L für links und R für rechts annehmen.

Definition (Turing-Maschine – *Fortsetzung*)

Das **Band**

- ist in **einzelne Zellen** unterteilt, die jeweils ein **beliebiges Symbol** aus Γ enthalten können, und
- beinhaltet zu Beginn die ***Eingabe***, d. h. ein **endliches Wort** aus Σ^*; alle anderen Zellen enthalten das besondere Symbol $\sqcup$ (in der Lit. auch als ***Leerzeichen*** oder *Blank* bezeichnet).

Der **Lese-/Schreibkopf** kann jeweils **genau eine Zelle** des Bandes **lesen** und **beschreiben**.

Konfiguration und Berechnung einer TM

Die **Konfiguration** einer **Turing-Maschine** M ist durch die Angabe

- des **Zustandes der Zustandssteuerung** von M,
- der **Position des Lese-/Schreibkopfes** von M und
- dem **Bandinhalt** von M

eindeutig spezifiziert.

Anmerkung:
Bei der Darstellung der Konfiguration vom M werden Leerzeichen nicht angegeben. Davon ausgenommen sind die Leerzeichen, die sich zwischen dem Lese-/Schreibkopf und einem vom Leerzeichen verschiedenen Zeichen befinden.

Konfiguration und Berechnung einer TM

Definition (Konfiguration einer Turing-Maschine M)

Gegeben sei $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$. Eine Zeichenreihe

$$K = X_1 X_2 \dots X_{i-1} q X_i X_{i+1} \dots X_n$$

heisst **Konfiguration von** M. Dabei

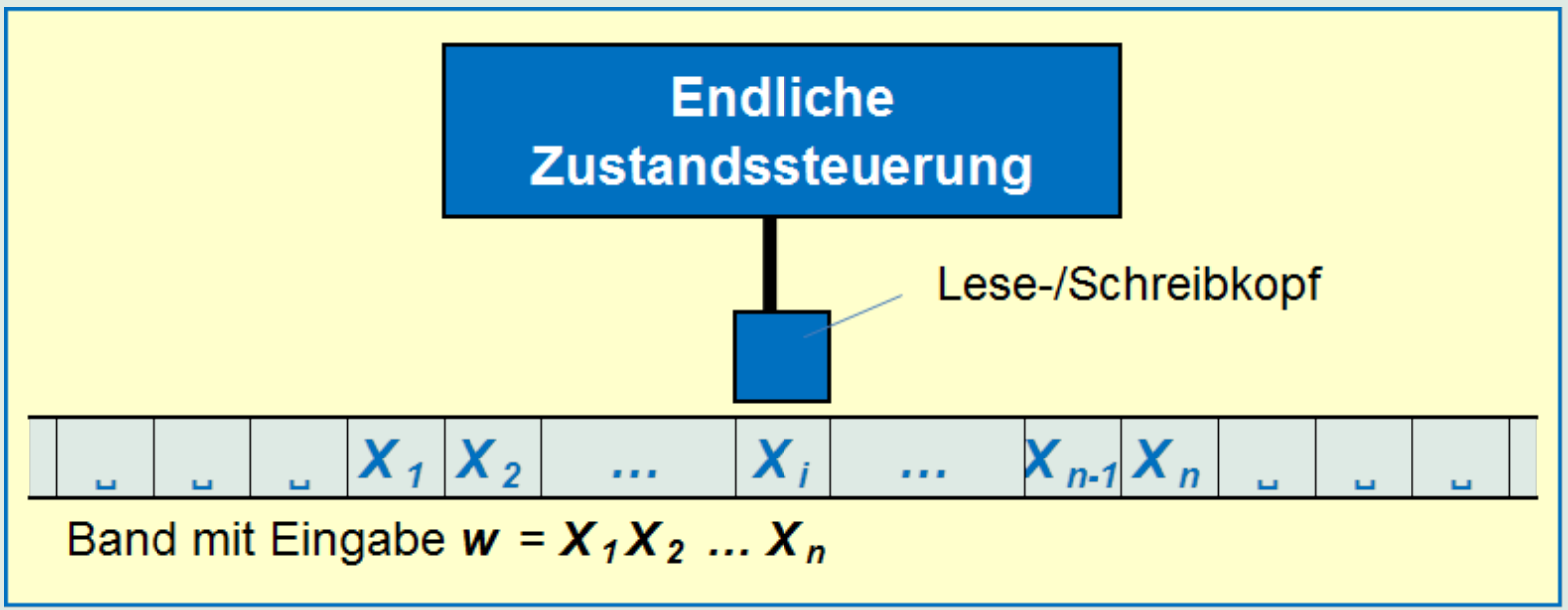
- ist q der **Zustand der Zustandssteuerung** von M ($q \in Q$),
- befindet sich der Lese-/Schreibkopf über der Zelle für X_i und
- $X_1, \dots, X_n \in \Gamma$.

Anmerkung:

Das **Präfix** $X_1 \dots X_{i-1}$ kann wie das **Suffix** $X_i \dots X_n$ **leer** sein, wenn $i = 1$ bzw. $i = n$ ist.

Konfiguration und Berechnung einer TM

Beispiel (Darstellung der Konfiguration $X_1X_2\ldots X_{i-1}qX_iX_{i+1}\ldots X_n$ einer Turing-Maschine M)



Konfiguration und Berechnung einer TM

Definition (Startkonfiguration einer Turing-Maschine M)

Gegeben sei $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ und ein Wort $X_1 \dots X_n \in \Sigma^*$.

Dann stellt die Zeichenreihe

$$K = q_0 X_1 X_2 \dots X_n$$

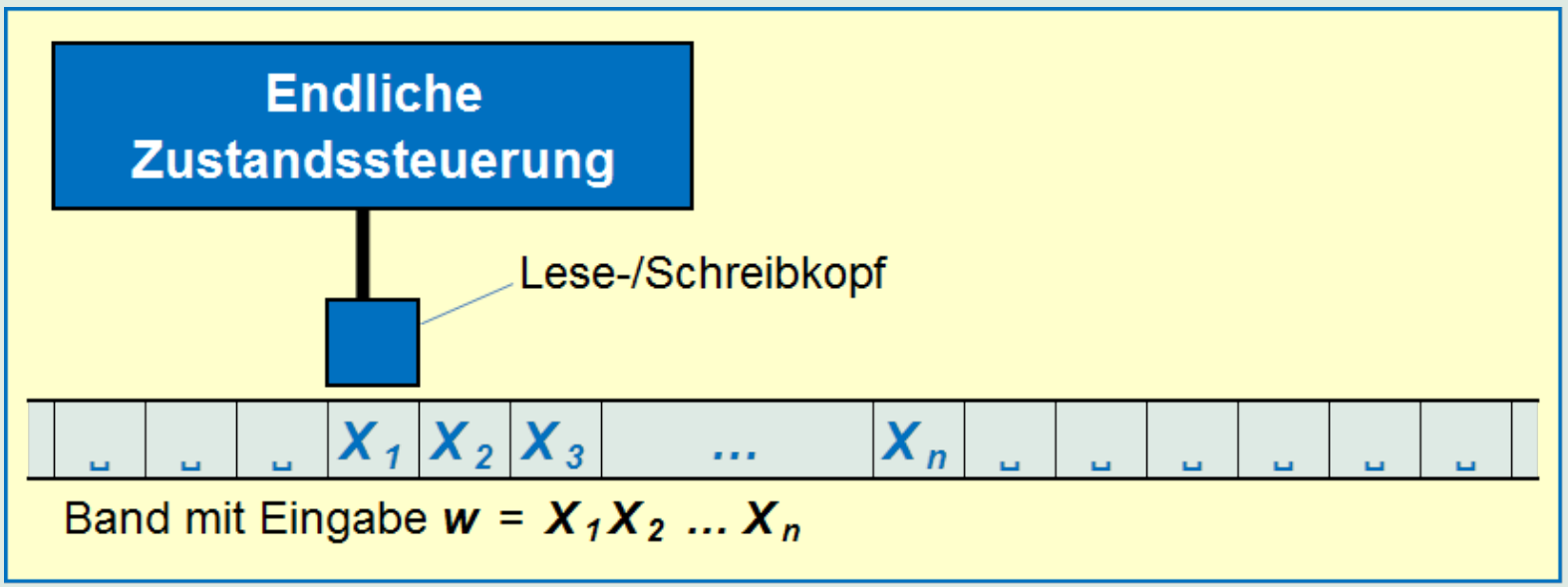
die **Startkonfiguration von** M dar.

Anmerkungen:

- Zu Beginn der Bearbeitung steht der Lese-/Schreibkopf über der Zelle mit dem ersten Zeichen der Eingabe, die von links nach rechts auf das Band geschrieben wird.
- Alle Zellen vor und nach den Zellen, die die Eingabe beinhalten, sind leer, d. h. enthalten das besondere Symbol $\sqcup$ (*Blank*).

Konfiguration und Berechnung einer TM

Beispiel (Darstellung der **Startkonfiguration** $K_0 = q_0 X_1 \dots X_n$ der **Turing-Maschine** M)



Konfiguration und Berechnung einer TM

Definition (Bewegung / Berechnungsschritt einer Turing-Maschine M)

Gegeben sei $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ und $\delta(q, X_i)$ sei definiert, $q \in Q$ und $X_i \in \Gamma$.

Der Übergang $\delta(q, X_i) = (p, Y, R)$ spezifiziert einen **Berechnungsschritt** mit einer Bewegung nach **rechts**:

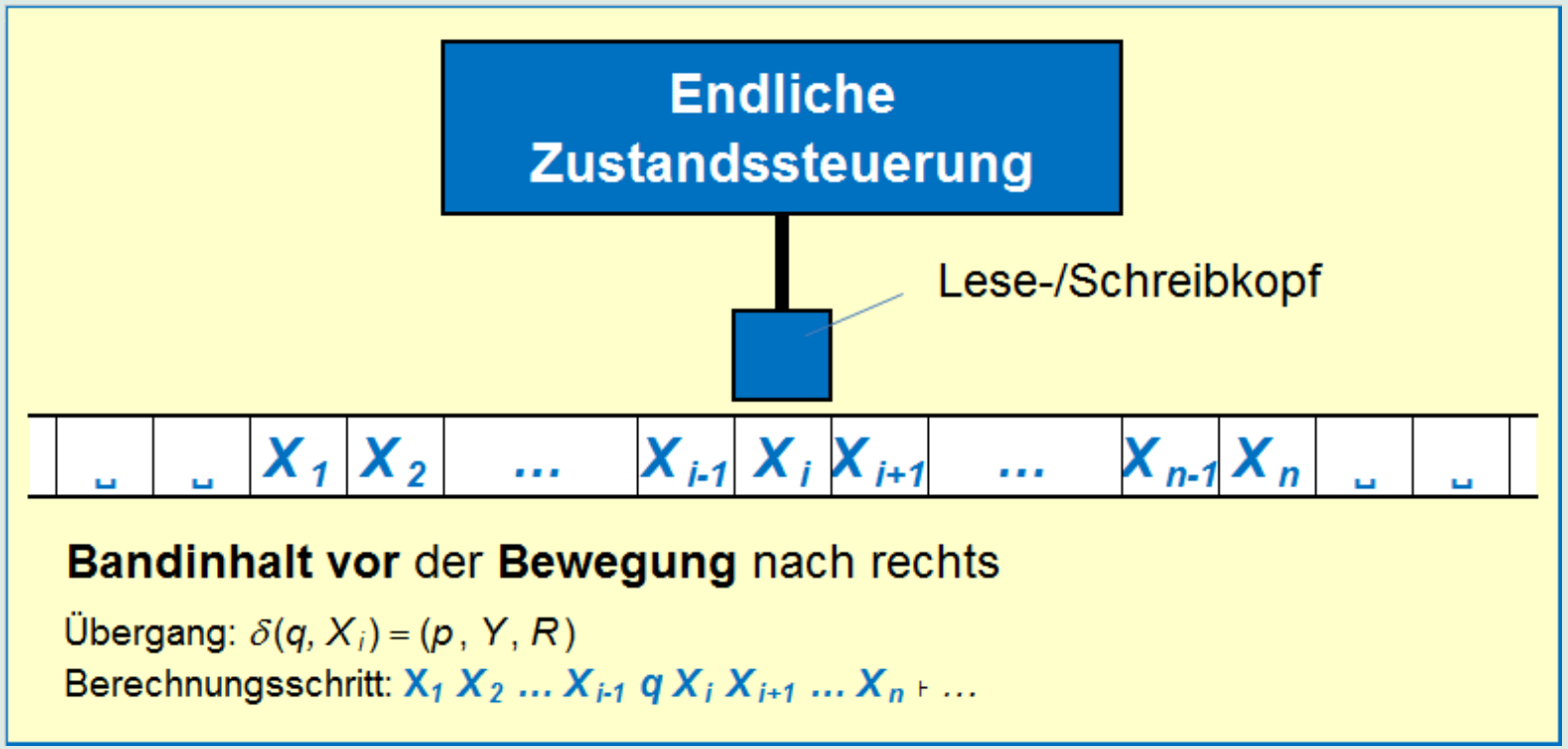
$$X_1 \dots X_{i-1} q X_i X_{i+1} \dots X_n \vdash X_1 \dots X_{i-1} Y p X_{i+1} \dots X_n$$

Der Übergang $\delta(q, X_i) = (p, Y, L)$ spezifiziert einen **Berechnungsschritt** mit einer Bewegung nach **links**:

$$X_1 \dots X_{i-1} q X_i X_{i+1} \dots X_n \vdash X_1 \dots X_{i-2} p X_{i-1} Y X_{i+1} \dots X_n$$

Konfiguration und Berechnung einer TM

Beispiel (**Bewegung nach rechts** einer **Turing-Maschine**)



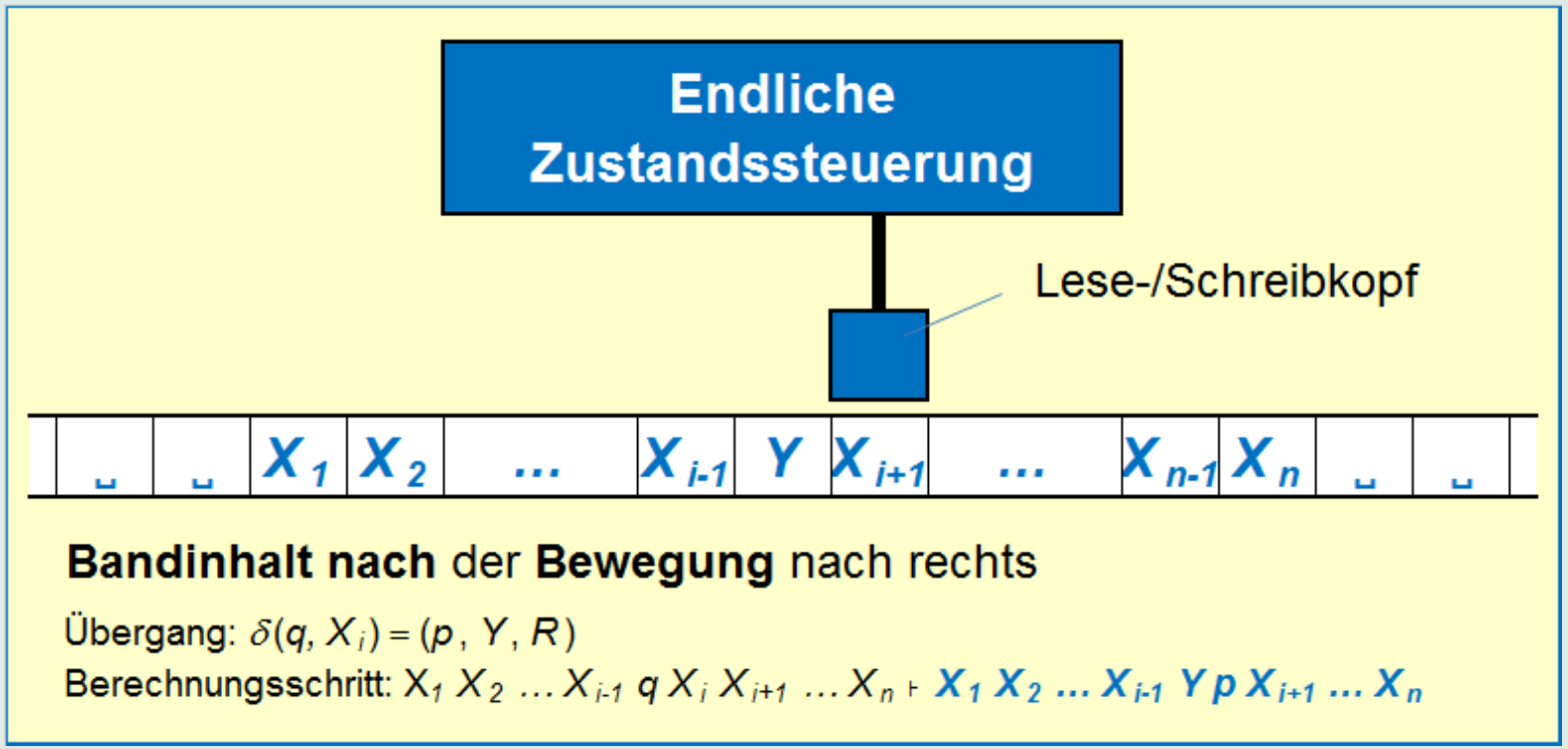
Bandinhalt vor der **Bewegung** nach rechts

Übergang: $\delta(q, X_i) = (p, Y, R)$

Berechnungsschritt: $X_1\ X_2\ \ldots\ X_{i-1}\ q\ X_i\ X_{i+1}\ \ldots\ X_n \vdash \ldots$

Konfiguration und Berechnung einer TM

Beispiel (**Bewegung nach rechts** einer **Turing-Maschine**)



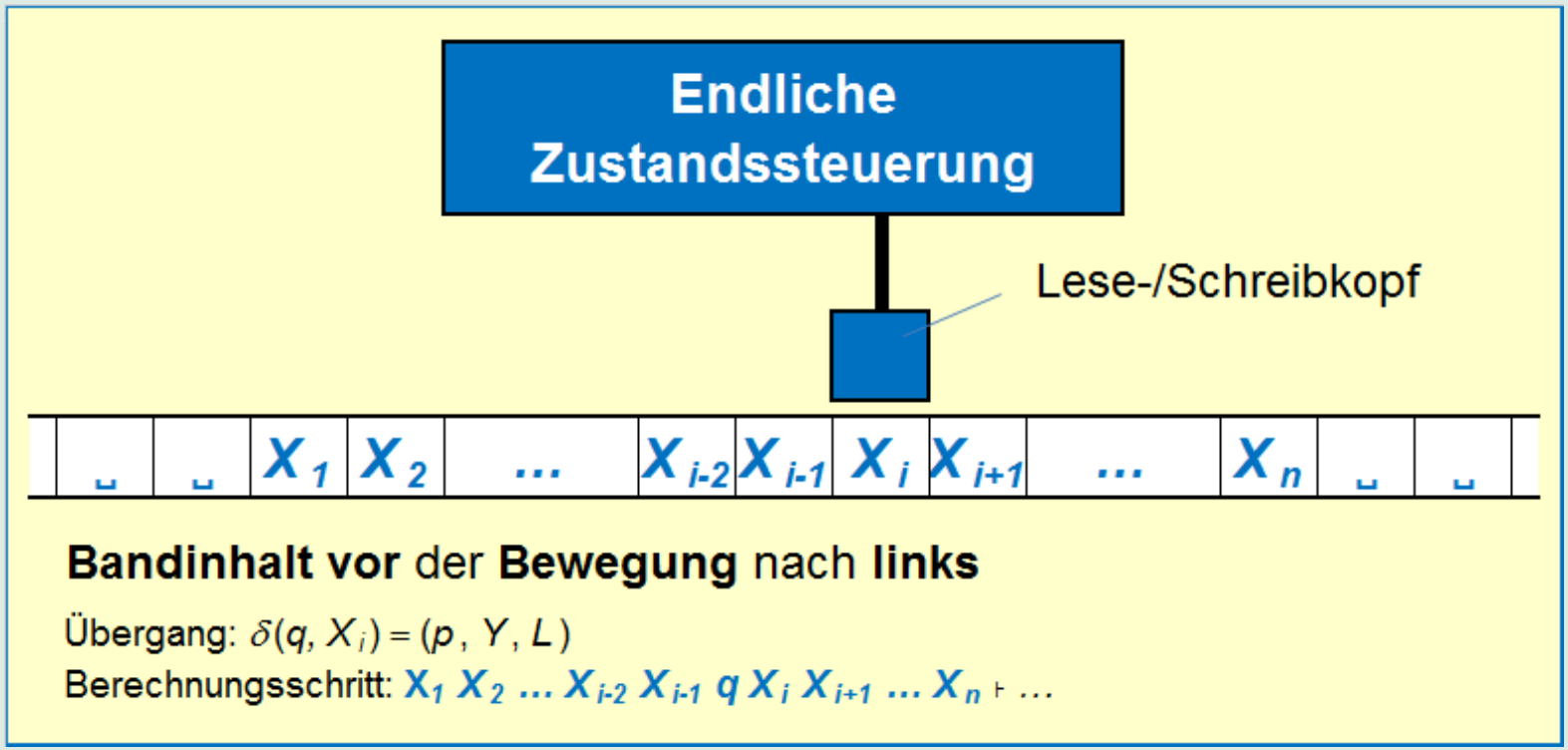
Bandinhalt nach der **Bewegung** nach rechts

Übergang: $\delta(q, X_i) = (p, Y, R)$

Berechnungsschritt: $X_1\ X_2\ \ldots X_{i-1}\ q\ X_i\ X_{i+1}\ \ldots X_n \vdash X_1\ X_2\ \ldots X_{i-1}\ Y\ p\ X_{i+1}\ \ldots X_n$

Konfiguration und Berechnung einer TM

Beispiel (**Bewegung nach links** einer **Turing-Maschine**)



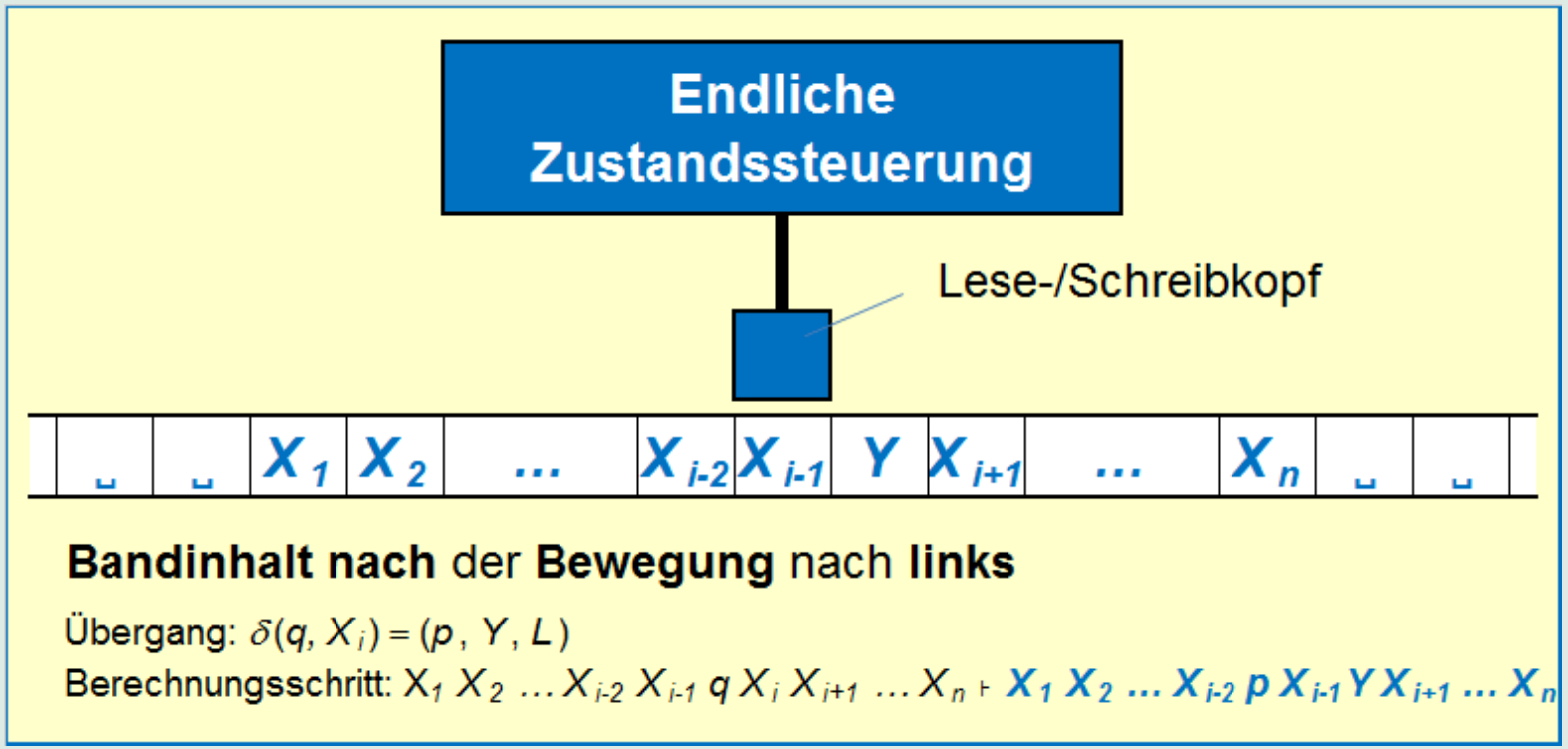
Bandinhalt vor der **Bewegung** nach **links**

Übergang: $\delta(q, X_i) = (p, Y, L)$

Berechnungsschritt: $X_1\ X_2\ \ldots\ X_{i-2}\ X_{i-1}\ q\ X_i\ X_{i+1}\ \ldots\ X_n \vdash \ldots$

Konfiguration und Berechnung einer TM

Beispiel (**Bewegung nach links** einer **Turing-Maschine**)



Bandinhalt nach der **Bewegung** nach **links**

Übergang: $\delta(q, X_i) = (p, Y, L)$

Berechnungsschritt: $X_1 X_2 \ldots X_{i-2} X_{i-1} q X_i X_{i+1} \ldots X_n \vdash X_1 X_2 \ldots X_{i-2} p X_{i-1} Y X_{i+1} \ldots X_n$

Konfiguration und Berechnung einer TM

Bewegung einer TM:**Sonderfälle**

- **Rechts-Bewegung** einer Turing-Maschine mit $\delta(q, X_i) = (p, Y, R)$
  - Für $i = n$ gilt: $X_1 \dots X_{n-1} q X_n \vdash X_1 \dots X_{n-1} Y p$
  - Für $i = 1$ und $Y = \sqcup$ gilt: $q X_1 \dots X_n \vdash p X_2 \dots X_n$
- **Links-Bewegung** einer Turing-Maschine mit $\delta(q, X_i) = (p, Y, L)$
  - Für $i = 1$ gilt: $q X_1 \dots X_n \vdash p \sqcup Y X_2 \dots X_n$
  - Für $i = n$ und $Y = \sqcup$ gilt: $X_1 \dots X_{n-1} q X_n \vdash X_1 \dots X_{n-2} p X_{n-1}$

Anmerkungen:

Wie bereits erwähnt wird das Leerzeichen nur dann angegeben, wenn es für die Eindeutigkeit erforderlich ist. In den oben erwähnten Sonderfällen ist dieses z. B. für eine Links-Bewegung mit $i = 1$ der Fall.

Allgemein werden die (unendlich vielen) Leerzeichen weggelassen, die sich rechts und links von der Konfigurationsbeschreibung befinden.

Konfiguration und Berechnung einer TM

Definition (Berechnung einer Turing-Maschine M)

Gegeben seien $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$ und eine Eingabe $w \in \Sigma$.

Eine **Berechnung** auf der Eingabe w ist eine endliche Folge von Berechnungsschritten

$$K_0 \vdash K_1 \vdash \ldots \vdash K_n,$$

so dass

- K_0 die Startkonfiguration ist und
- aus der Konfiguration K_n keine Bewegung mehr möglich ist.

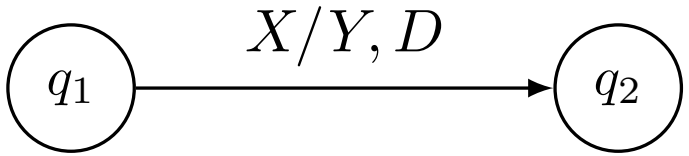
Anmerkung:

Vereinfachenderweise wird für $K_0 \vdash K_1 \vdash \ldots \vdash K_n$ auch $K_0 \vdash^* K_n$ geschrieben.

Erstes Beispiel

Analog zu den endlichen Automaten und den Kellerautomaten können wir Turing-Maschinen graphisch darstellen:

- Die **Zustände** werden wie bisher über Kreise symbolisiert.
- Ein Übergang $\delta(q, X) = (p, Y, D)$ wird dargestellt durch



  für ein $D \in \{L, R\}$ und bedeutet:

  Wenn die **TM** (im Zustand q) ein X vom Band liest, schreibt der Lese-/Schreibkopf das Symbol Y auf das Band und bewegt sich um ein Feld nach recht (falls $D = R$) oder nach links (falls $D = L$).
- Der **Anfangszustand** q_0 wird wie bisher durch einen eingehenden Pfeil gekennzeichnet und die **akzeptierenden Zustände** durch eine doppelte Konturlinie.

Erstes Beispiel

Beispiel ($L = \{0^n 1^n \mid n \geq 1\}$)

Gegeben sei die Turing-Maschine $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, \{q_4\})$ mit:

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$ und
- $\Gamma = \{0, 1, X, Y, \sqcup\}$.

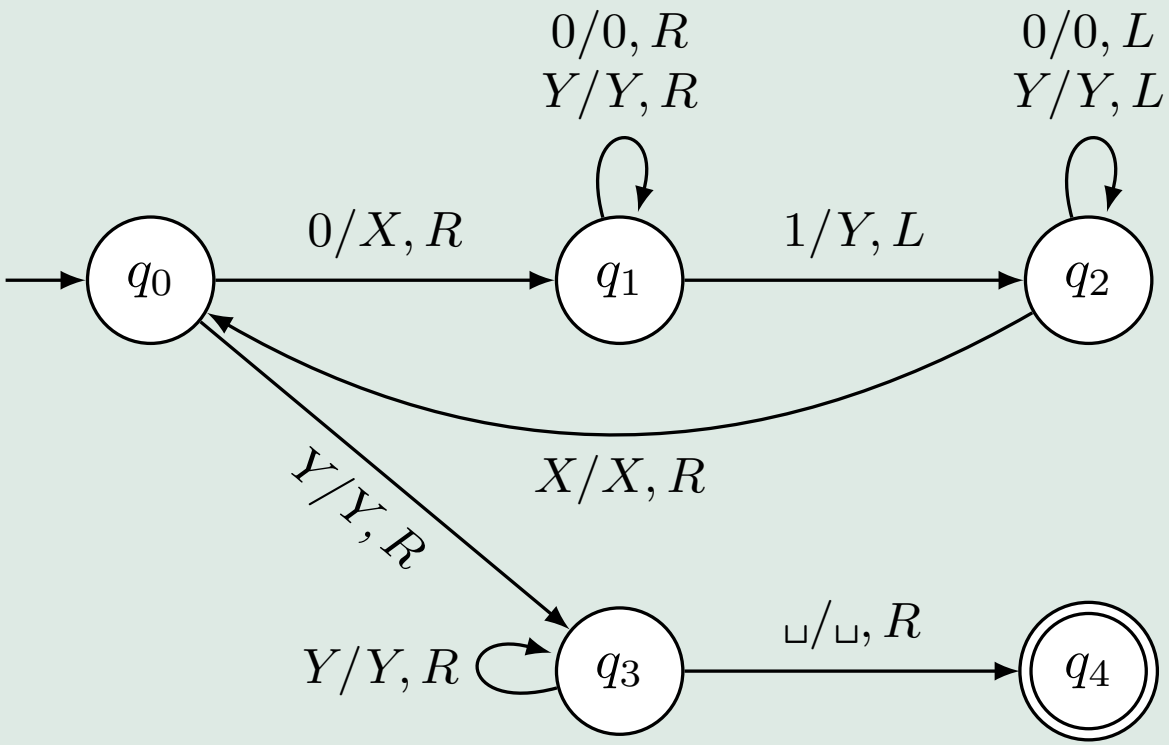
Die Übergangsfunktion δ ist gegeben durch:

$$\delta(q_0, 0) = (q_1, X, R), \delta(q_0, Y) = (q_3, Y, R),$$
$$\delta(q_1, 0) = (q_1, 0, R), \delta(q_1, 1) = (q_2, Y, L), \delta(q_1, Y) = (q_1, Y, R),$$
$$\delta(q_2, 0) = (q_2, 0, L), \delta(q_2, X) = (q_0, X, R), \delta(q_2, Y) = (q_2, Y, L),$$
$$\delta(q_3, Y) = (q_3, Y, R) \text{ und } \delta(q_3, \sqcup) = (q_4, \sqcup, R).$$

Erstes Beispiel

Beispiel (Fortsetzung)

Übergangsdiagramm für M:



Erstes Beispiel

Beispiel (Fortsetzung)

Berechnungen mit M für a) $w = 01$ und b) $w = 0011$:

a) $q_0 01 \vdash X q_1 1 \vdash q_2 XY \vdash X q_0 Y \vdash XY q_3 \vdash XY \sqcup q_4$

b) $q_0 0011 \vdash X q_1 011 \vdash X 0 q_1 11 \vdash X q_2 0 Y 1 \vdash q_2 X 0 Y 1 \vdash X q_0 0 Y 1 \vdash$
$XX q_1 Y 1 \vdash XXY q_1 1 \vdash XX q_2 YY \vdash X q_2 XYY \vdash XX q_0 YY \vdash$
$XXY q_3 Y \vdash XXYY q_3 \vdash XXYY \sqcup q_4$

Beide Berechnungen enden in q_4, einem **akzeptierenden Zustand** ($q_4 \in F$).

Erstes Beispiel

Beispiel (Fortsetzung)

Berechnungen mit M für c) $w = 010$ und d) $w = 0010$:

c) $q010 \vdash Xq_1 10 \vdash q_2 XY0 \vdash Xq_0 Y0 \vdash XYq_3 0$

Im Zustand q_3 ist für das gelesene Zeichen 0 **kein** Berechnungsschritt mehr möglich und endet hier. q_3 ist **kein akzeptierender Zustand**.

d) $q0010 \vdash Xq_1 010 \vdash X0q_1 10 \vdash Xq_2 0Y0 \vdash q_2 X0Y0 \vdash Xq_0 0Y0 \vdash XXq_1 Y0 \vdash XXYq_1 0 \vdash XXY0q_1$

Im Zustand q_1 ist für das gelesene Zeichen $\sqcup$ **kein** Berechnungsschritt mehr möglich und endet hier. q_1 ist **kein akzeptierender Zustand**.

Sprache einer TM

Definition (Sprache einer TM)

Gegeben sei $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$. Die Menge der Wörter, für die M nach einer Berechnung in einen akzeptierenden Endzustand übergeht, ist die durch M **akzeptierte Sprache** $L(M)$

$$L(M) = \{w \mid q_0 w \vdash^* \alpha p \beta, p \in F \text{ und } \alpha, \beta \in \Gamma^*\}.$$

Definition (Rekursiv aufzählbar)

Eine Sprache, die von einer TM akzeptiert wird, wird als **rekursiv aufzählbar** bezeichnet. Man spricht auch von den **rekursiv aufzählbaren Sprachen**.

Anmerkung: Die Klasse der **rekursiv aufzählbaren Sprachen** ist sehr mächtig. Sie beinhaltet u. a. alle kontextfreien Sprachen.

Sprache einer TM

- **Es gilt:**

  Wenn eine TM irgendwann anhält, kann die Zugehörigkeit von w zu $L(M)$ "mechanisch" entschieden werden.

- **Aber:**

  Solange eine TM eine Eingabe bearbeitet (und nicht anhält), kann $w \in L(M)$ nicht entschieden werden.

Rechnen mit einer TM

Anmerkungen:

- In der vorgängigen Definition wurde die TM als "Spracherkenner" gleich "Problemlöser" interpretiert.[2]
- Bei der Entwicklung des Modells der Turing-Maschinen ging es A. M. Turing jedoch um die **Modellierung eines Computers für ganzzahlige Funktionen** und der Frage, was sich mit einem Computer **überhaupt berechnen** lässt.

[2] Zur Erinnerung: Jedes Entscheidungsproblem kann als die Frage formuliert werden, ob eine Zeichenreihe w zu einer Sprache L gehört oder nicht.

Rechnen mit einer TM

Mögliche Modellierung eines Computers für ganzzahlige Funktionen:

- Ganze Zahlen werden als Blöcke unär auf das Band geschrieben (eine Zahl n wird als 0^n dargestellt).
- Die TM führt Berechnungen durch, indem sie Blöcke verändert (Zellen löscht, neue erstellt ...).
- Die Berechnung ist abgeschlossen, wenn die TM in einem Zustand p und einem gelesenen Bandsymbol X anhält, d. h. sie kann in dieser Konfiguration keine Bewegung mehr durchführen: $\delta(p, X)$ ist undefiniert (äquivalent zur Definition einer Berechnung von M).
- Das Resultat der Berechnung steht dann als Folge von Nullen auf dem Band.

Anmerkung:

Ohne die akzeptierte Sprache einer TM zu verändern, kann $\delta(p, X)$ für alle $p \in F$, als undefiniert angenommen werden.

Rechnen mit einer TM

Mögliche Modellierung eines Computers für ganzzahlige Funktionen:

Sehr viele mathematische Funktionen, wie z. B. die Addition, Subtraktion, Multiplikation, Potenzierung, Modulo-Funktion usw., können so mit einer TM *berechnet* werden.

Beispiel (Aufgabe – Addition von zwei natürlichen Zahlen)

a) Spezifizieren Sie eine TM (grafisch und formal als 7-Tupel), welche die Addition von zwei natürlichen Zahlen realisiert.

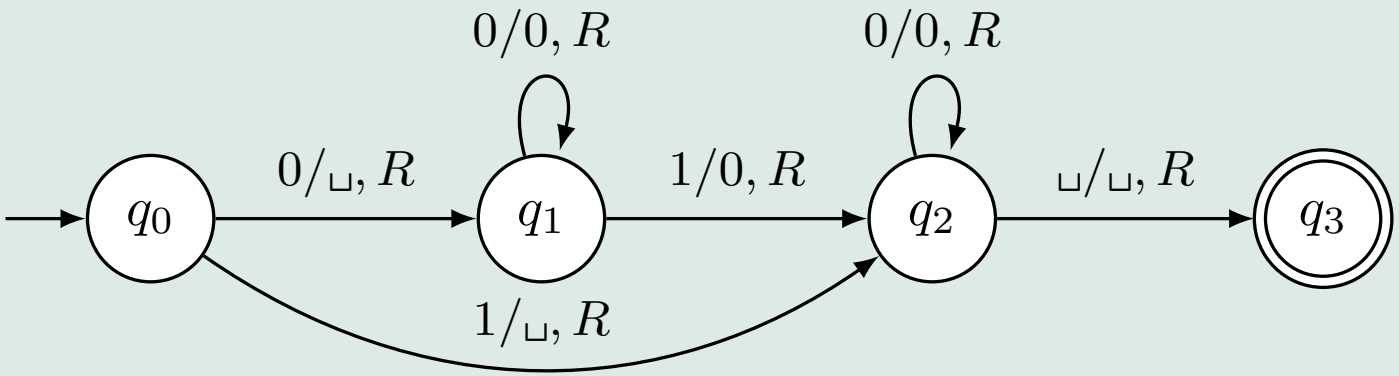
b) Berechnen Sie mit der TM: $3+1$, $3+4$, $1+0$, $0+1$ und $0+0$.

Anmerkung:

Die Eingabe mehrerer Argumente für eine Funktion erfolgt über Blöcke von Nullen, die durch eine Eins getrennt auf das Band geschrieben werden, z. B. 00010000 (für eine Funktion mit den Argumenten 3 und 4).

Beispiel (Aufgabe – Addition von zwei natürlichen Zahlen)

a) Grafische Spezifikation von M:



Spezifikation von M als 7-Tupel:

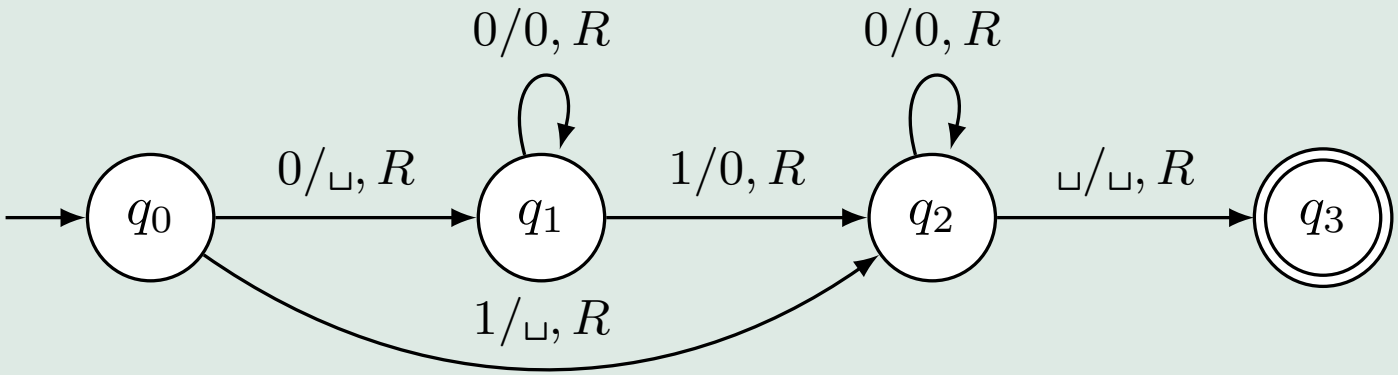
$M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, ␣\}, \delta, q_0, ␣, \{q_3\})$

mit: $\delta(q_0, 0) = (q_1, ␣, R), \delta(q_0, 1) = (q_2, ␣, R), \delta(q_1, 0) = (q_1, 0, R),$
$\delta(q_1, 1) = (q_2, 0, R), \delta(q_2, 0) = (q_2, 0, R)$ und $\delta(q_2, ␣) = (q_3, ␣, R).$

Rechnen mit einer TM

Beispiel (Aufgabe – Addition von zwei natürlichen Zahlen)

b) Berechnungen von M:

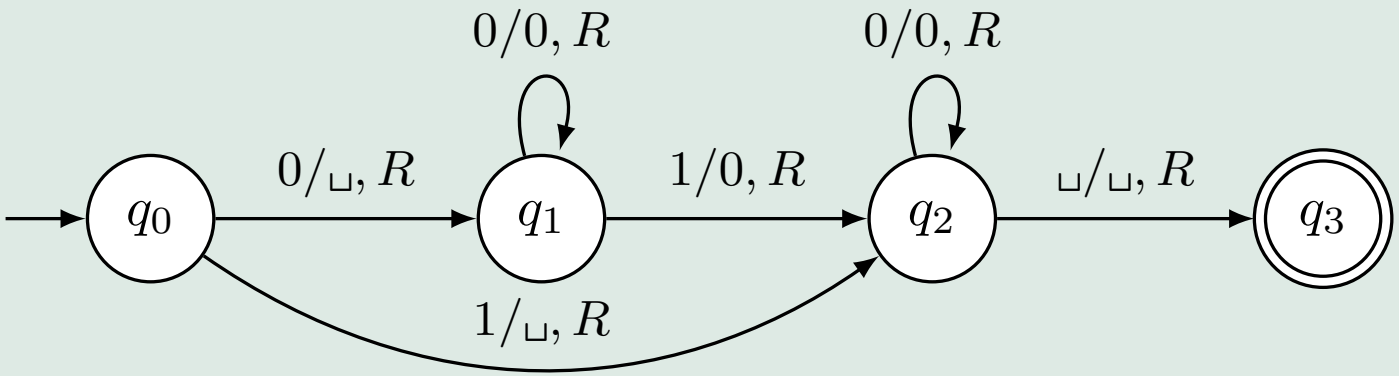


$3+4$: $q_0 00010000 \vdash q_1 0010000 \vdash 0q_1 010000 \vdash 00q_1 10000 \vdash 000q_2 0000$
$\vdash 0000q_2 000 \vdash 00000q_2 00 \vdash 000000q_2 0 \vdash 0000000q_2 \vdash 0000000\sqcup q_3$

$3+1$: $q_0 00010 \vdash q_1 0010 \vdash 0q_1 010 \vdash 00q_1 10 \vdash 000q_2 0 \vdash 0000q_2 \vdash 0000\sqcup q_3$

Beispiel (Aufgabe – Addition von zwei natürlichen Zahlen)

b) Berechnungen von M:



$1+0$: $q_0 01 \vdash q_1 1 \vdash 0 q_2 \vdash 0 \sqcup q_3$

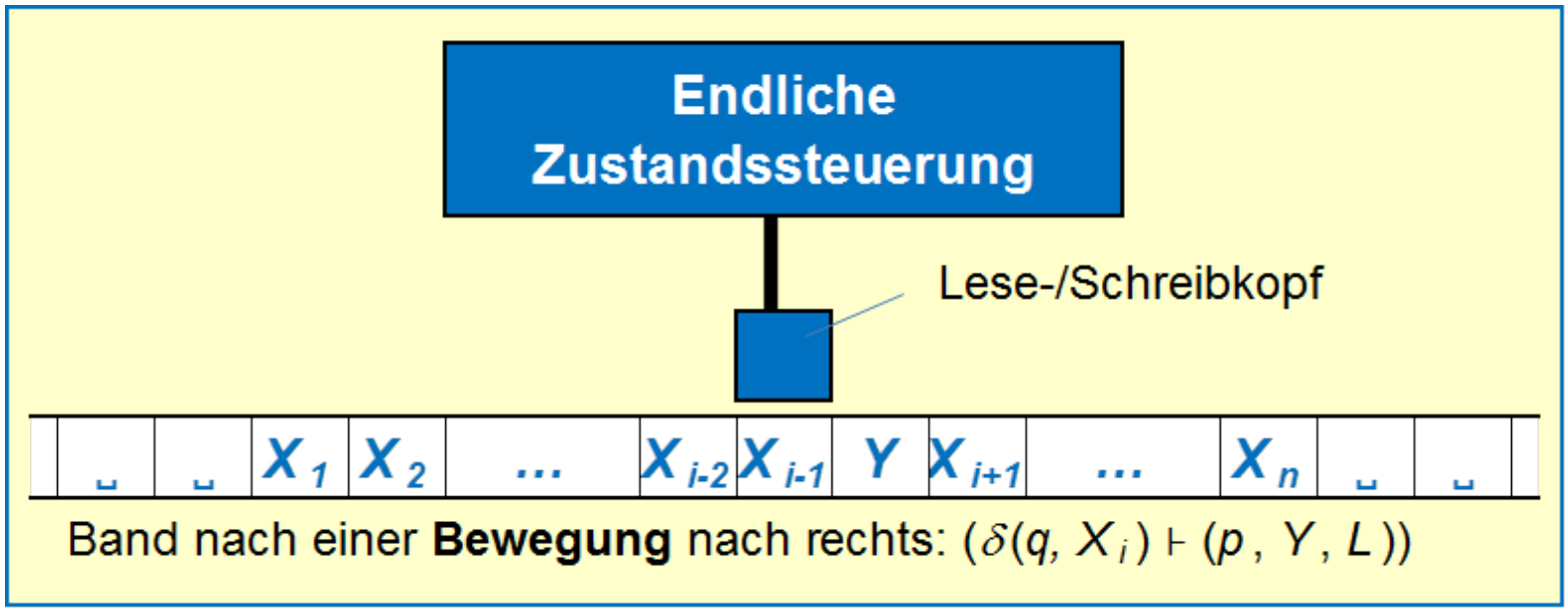
$0+0$: $q_0 1 \vdash q_2 \vdash q_3$

$0+1$: $q_0 10 \vdash q_2 0 \vdash 0 q_2 \vdash 0 \sqcup q_3$

Backup – Erweiterungen von TM

TM mit **Speichern**

- In der endlichen Zustandssteuerung einer TM können ausser dem (Steuer-)Zustand zusätzlich endlich viele (Daten-)Zustände gespeichert werden.



Backup – Erweiterungen von TM

TM mit **Speichern**

- Ist lediglich eine Vereinfachung der Darstellung, da die Steuerzustände und die endlichen Symbole des Bandalphabets, die jeder der endlich vielen „Speicher“ aufnehmen kann, auch als kartesisches Produkt für die Menge der Zustände einer TM aufgefasst werden können.

Beispiel

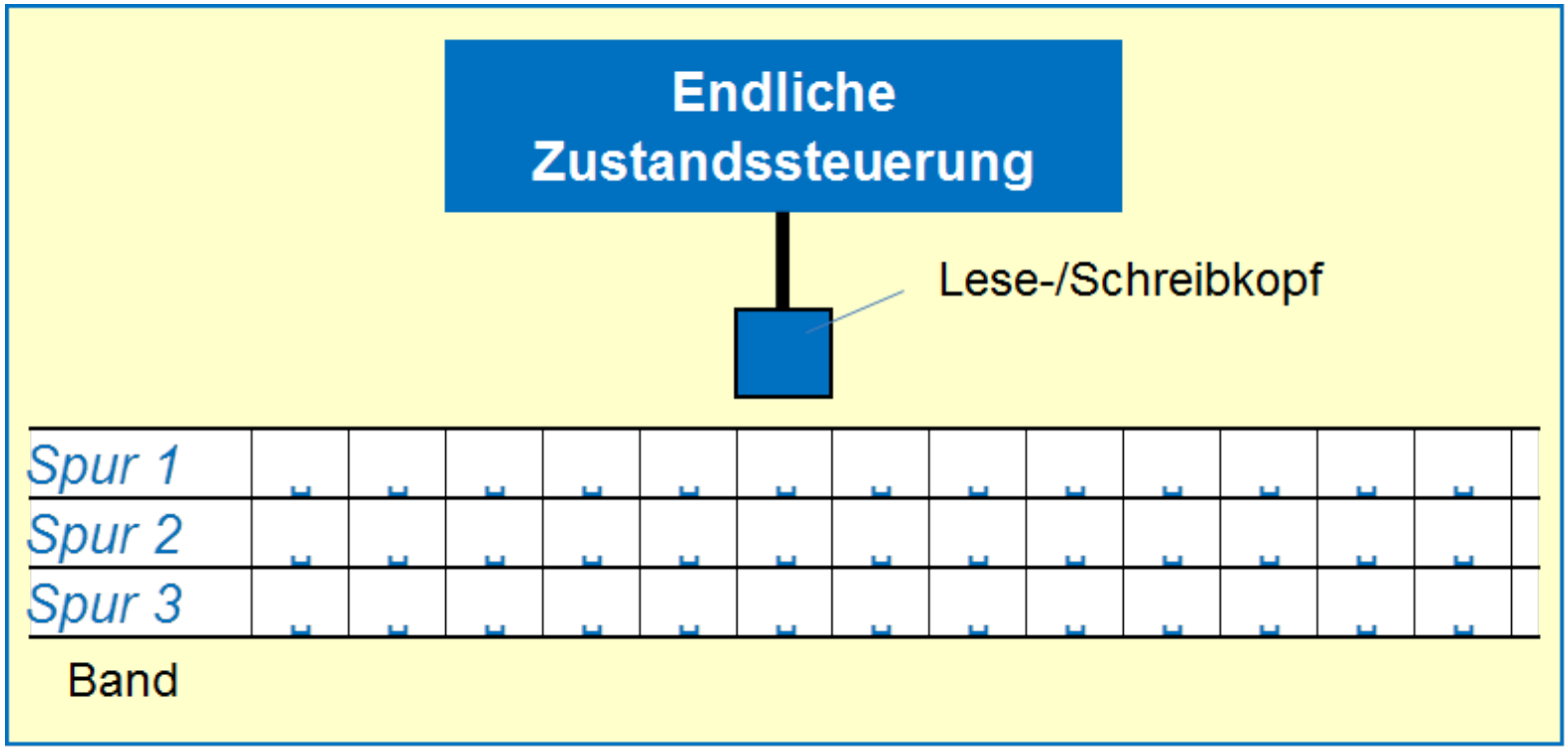
Eine TM mit den zwei Zuständen q_0 und q_1 und einem Speicher, der die Bandsymbole 0,1 und B speichern kann, weist folgende Zustände Q auf:

$$Q = \{q_0, q_1\} \times \{0, 1, \sqcup\} \text{ [6 mögliche Zustände]}$$

Erweiterungen von TM

TM mit **mehreren Spuren**

- Das Band der TM setzt sich aus mehreren „Spuren“ zusammen.
- Jede Spur kann ein Symbol des Bandalphabets speichern.



Erweiterungen von TM

TM mit **mehreren Spuren**

- Ist ebenfalls lediglich eine Vereinfachung der Darstellung.

  Eine TM M_M mit mehreren Spuren kann einfach über eine normale TM M_1 mit einer Spur simuliert werden, indem das Bandalphabet von M_1 ein n-Tupel (n ist die Anzahl der Spuren) ist und die Steuerung entsprechend angepasst wird.

Beispiel

Eine TM T_M hat drei Spuren und die Bandsymbole $\Gamma = \{0, 1, \sqcup\}$ für alle drei Spuren. T_M kann mit Hilfe einer normalen TM T_1 mit dem Bandalphabet

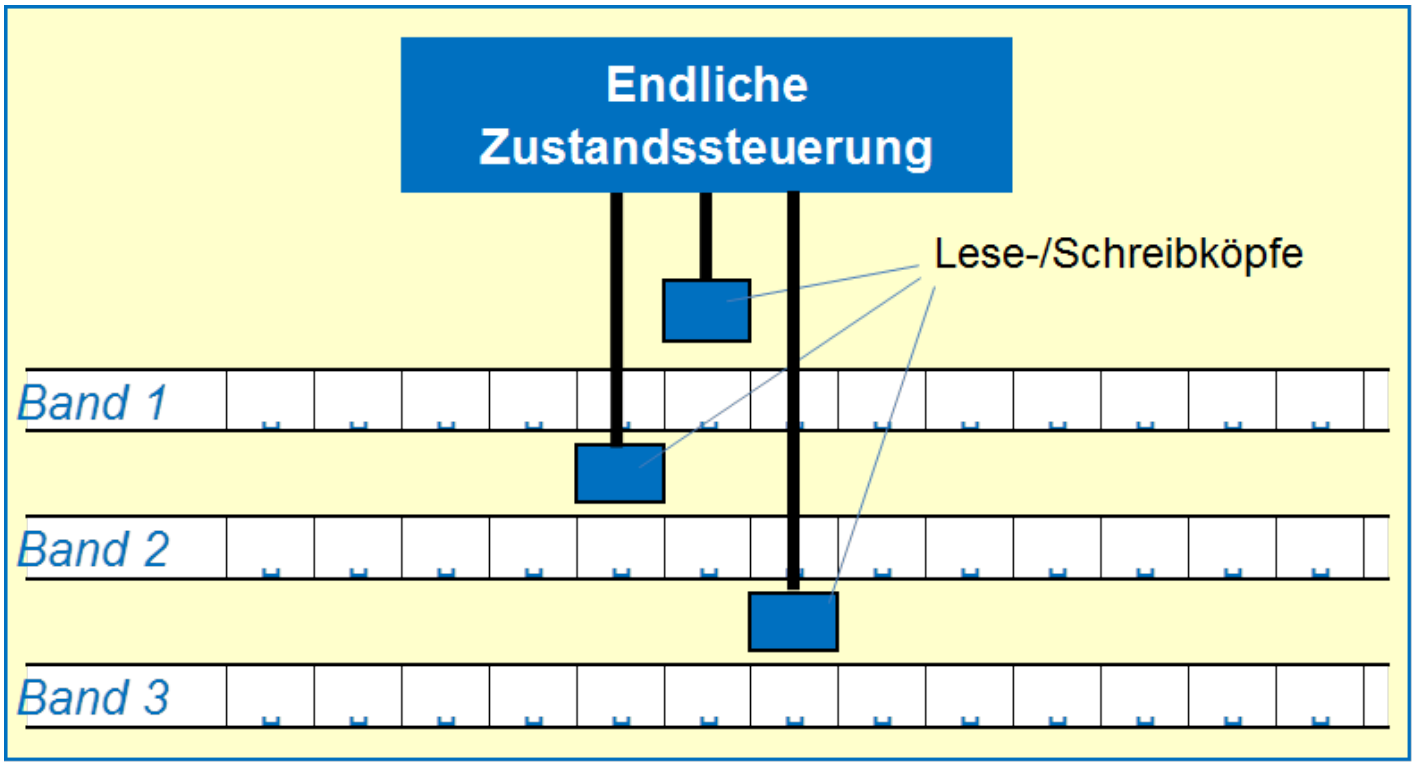
$$\Gamma_{T_1} = \{0, 1, \sqcup\} \times \{0, 1, \sqcup\} \times \{0, 1, \sqcup\} \text{ [27 Bandsymbole]}$$

simuliert werden.

Erweiterungen von TM

TM mit **mehreren Bändern** (und Lese-/Schreibköpfen)

- Die Turing-Maschine besitzt statt einem Band eine endliche Anzahl von Bändern und Lese-/Schreibköpfen.
- Jeder Lese-/Schreibkopf kann unabhängig auf ein Band zugreifen.



Erweiterungen von TM

TM mit **mehreren Bändern** (und Lese-/Schreibköpfen)

A) Funktionsprinzip – Initialisierung:

- Die Eingabe steht auf dem ersten Band.
- Alle anderen Bänder sind leer (in allen Zellen steht das Symbol $\sqcup$).
- Der Lese-/Schreibkopf des ersten Bandes befindet sich über der ersten Zelle mit der Eingabe (von links).
- Die Lese-Schreibköpfe der übrigen Bänder sind beliebig platziert (nicht relevant, da alle Zellen leer sind).
- Jedes Band kann ein beliebiges Symbol des Bandalphabets Γ in jeder Zelle aufnehmen.

Erweiterungen von TM

TM mit **mehreren Bändern** (und Lese-/Schreibköpfen)

B) Funktionsprinzip – Bewegung:

- Die Zustandssteuerung hängt von den gelesenen Symbolen auf **allen Bändern** ab.
- In die Zellen von jedem Band kann unabhängig ein Symbol geschrieben werden.
- Alle Lese-Schreibköpfe führen voneinander unabhängig eine Bewegung aus.
- Die Lese-Schreibköpfe können sich nach rechts (R) oder links (L) bewegen oder in der gleichen Position verbleiben (S), $D = \{R, L, S\}$.
- Die Übergangsfunktion lautet (für eine TM mit k Bändern) neu:

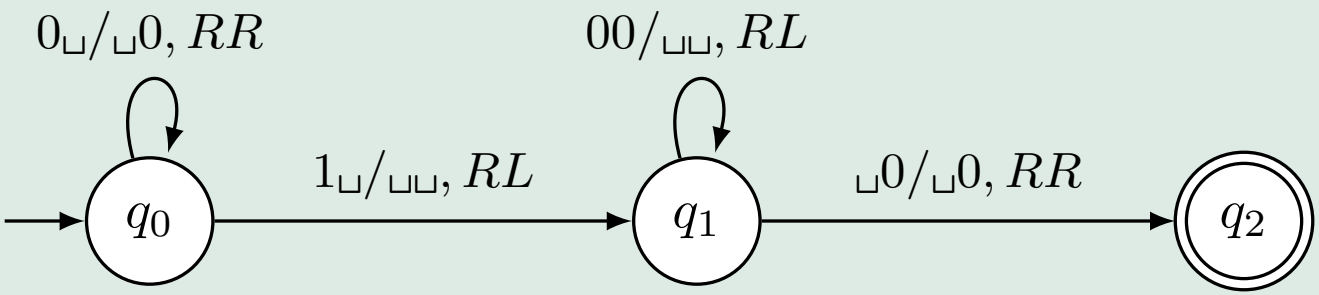
$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{R, L, S\}^k$$

Erweiterungen von TM

TM mit **mehreren Bändern**

Beispiel (Subtraktion zweier ganzer positiver Zahlen a und b, $(a > b)$)

a) Grafische Spezifikation von M:



Konvention:

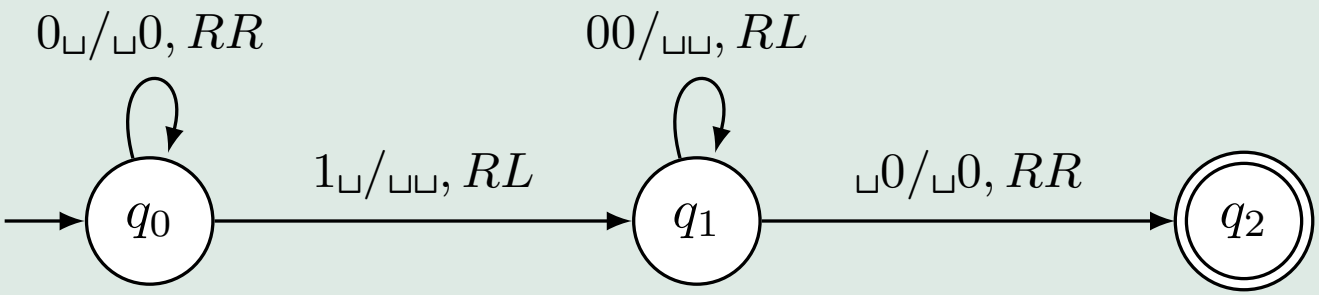
Die Beschriftung eines Übergangs mit $0\sqcup/\sqcup 1, RS$ bedeutet:

Wenn die TM auf dem ersten Band eine 0 und auf dem zweiten Band ein $\sqcup$ liest, schreibt sie auf das erste Band ein $\sqcup$ und auf das zweite Band eine 1. Der Lese-/Schreibkopf für das erste Band bewegt sich nach rechts und für das zweite Band verbleibt in seiner Position.

TM mit **mehreren Bändern**

Beispiel (Subtraktion zweier ganzer positiver Zahlen a und b, $(a > b)$)

b) Berechnung von M (für $4 - 2$):



Band 1:	$q_0 0000100$	$\vdash q_0 000100$	$\vdash q_0 00100$	$\vdash q_0 0100$	$\vdash q_0 100$
Band 2:	q_0	$\vdash 0q_0$	$\vdash 00q_0$	$\vdash 000q_0$	$\vdash 0000q_0$
		$\vdash q_1 00$	$\vdash q_1 0$	$\vdash q_1$	$\vdash q_2$
		$\vdash 000q_1 0$	$\vdash 00q_1 0$	$\vdash 0q_1 0$	$\vdash 00q_2$

Erweiterungen von TM

TM mit **mehreren Bändern** (und Lese-/Schreibköpfen)

Satz

Jede Sprache L, die von einer mehrbändigen Turing-Maschine akzeptiert wird, wird auch von einer Turing-Maschine mit nur einem Band akzeptiert.

Beweiskonstruktion[3]:

- Arbeitsalphabet so erweitern, dass alle Bänder und Lese-/Schreibköpfe gespeichert werden.

[3] Siehe auch Hopcroft et al. S. 347 ff.

Backup – Erweiterungen von TM

TM mit **mehreren Bändern** (und Lese-/Schreibköpfen)

Anmerkung zur Beweiskonstruktion:

- Simulation der k-Band-TM T_1 mit einer 1-Band-TM T_2 mit $2k$-Spuren:
  a) Jede $(2k-1)$-ste Spur von T_2 beinhaltet den Inhalt des k-ten Bandes von T_1.
  b) Jede $(2k)$-ste Spur von T_2 beinhaltet die Position des Lese-/Schreibkopfes des k-ten Bandes von T_1.
- Simulation der Bewegungen von T_1 durch T_2.

Erweiterungen von TM

Anmerkungen:

- Anwendungen der zuvor vorgestellten Erweiterungen von TM vereinfachen die Konzeption von „Programmen“ für Turing-Maschinen, die komplexere Funktionen berechnen:
  - Die Speicherung von Zuständen wird z. B. eingesetzt, wenn sich ein besonderer Wert gemerkt werden muss oder eine Fallunterscheidung behandelt werden muss.
  - Mehrere Spuren können für Markierungen genutzt werden oder Zwischenergebnisse aufnehmen.
  - Mehrere Bänder können für Zwischenergebnisse verwendet werden und vereinfachen eine TM sehr (i. d. R. viel weniger Zustände und Schritte).
- Eine weitere Technik für die Vereinfachung (und Strukturierung) ist die Nutzung von Unterprogrammen (einfach z. B. mit mehreren Bändern realisierbar).

Erweiterungen von TM

Definition (nichtdeterministische Turing-Maschine)

Eine **nichtdeterministische Turing-Maschine** (NTM) ist ein 7-Tupel

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$$

Mit Ausnahme der Übergangsfunktion δ entsprechen Q, Σ, Γ, q_0, $\sqcup$ und F der Definition der deterministischen TM.

Die Übergangsfunktion δ bildet neu auf eine Menge von Tripeln ab:

$$\delta \colon Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times D)\,.$$

Anmerkung: Wie der NEA führt die NTM bei jedem Berechnungsschritt gleichzeitig alle Möglichkeiten durch – "*Sie errät die richtige Lösung*".

Erweiterungen von TM

NTM – **Nichtdeterministische Turing-Maschine**

Satz

Jede Sprache L, die von einer NTM akzeptiert wird, wird auch von einer deterministischen TM akzeptiert.

Beweiskonstruktion[4]:

Simulation der NTM T_1 mit einer mehrbändigen deterministischen TM T_2:

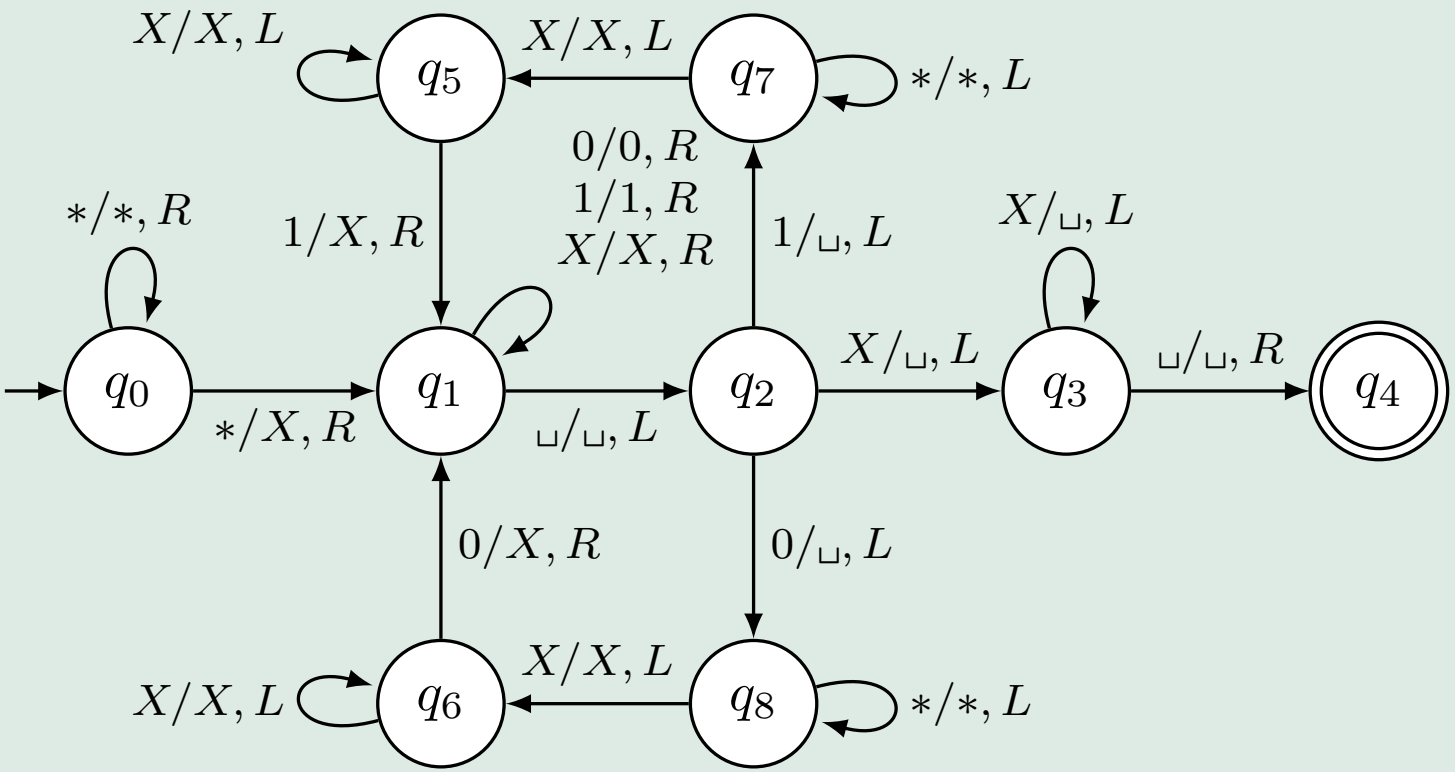
- In einem Band speichert T_2 die Folge aller möglichen Konfigurationen von T_1 in einer Warteschlange (am Ende des ersten Bandes).
- T_2 simuliert alle möglichen Konfigurationen von T_1 und prüft, ob T_1 eine akzeptierende Konfiguration erreicht. Wenn ja, akzeptiert auch T_2.
- Dabei untersucht T_2 in jedem Schritt, ob es mehrere Möglichkeiten gibt und speichert diese weiteren Konfigurationen jeweils am Ende der ersten Bandes.

[4] Siehe auch Hopcroft et al. S. 350/351.

NTM – **Nichtdeterministische Turing-Maschine**

Beispiel (für $L = \{waw \mid w \in \Sigma^*, a \in \Sigma\}$ mit $\Sigma = \{0, 1\}$)

b) Grafische Spezifikation von M: ($*$ steht für ein beliebiges Zeichen aus Σ)

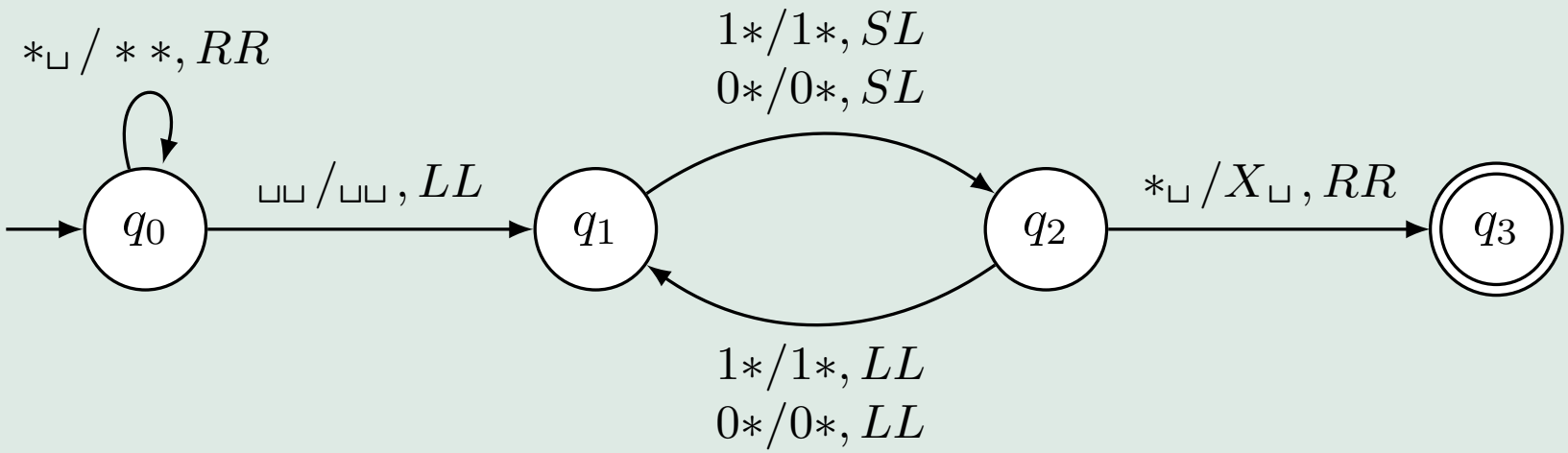


Backup – Erweiterungen von TM

TM mit **mehreren Bändern**

Beispiel (Suche der Mitte eines Wortes waw' mit: $w, w' \in \Sigma^*$, $|w| = |w'|$, $a \in \Sigma$, $\Sigma = \{0, 1\}$)

a) Grafische Spezifikation: (mit zwei Bändern)

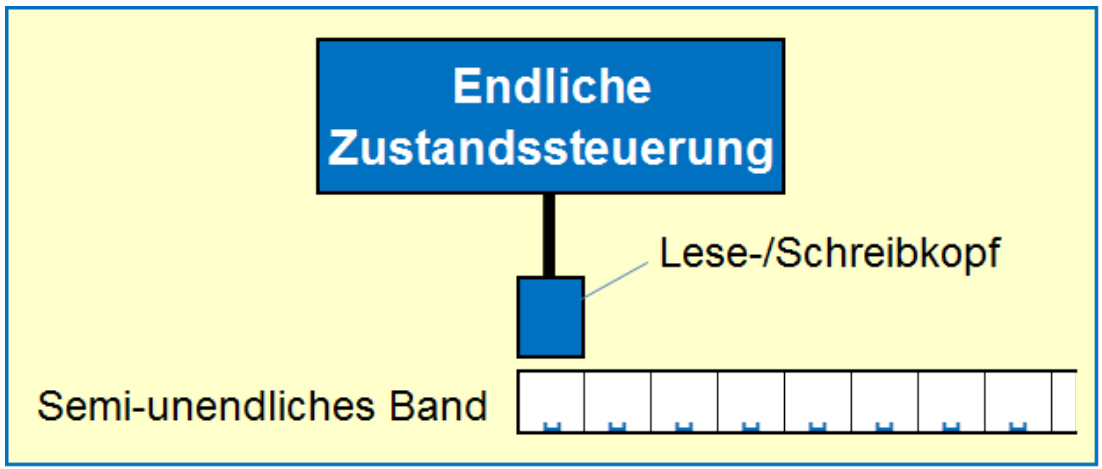


Anmerkung: Beschriftung eines Übergangs der 2-Band-TM wie bisher.

Beschränkungen von TM

TM mit **semi-unendlichem Band (beschränktem Band)**

- Das Band der Turing-Maschine ist semi-unendlich (nur in eine Richtung unendlich, z. B. nach rechts).
- Links von der Anfangsposition des Lese-/Schreibkopfes befinden sich keine Zellen.
- Der Lese-/Schreibkopf kann keine Positionen links von der Anfangsposition einnehmen.



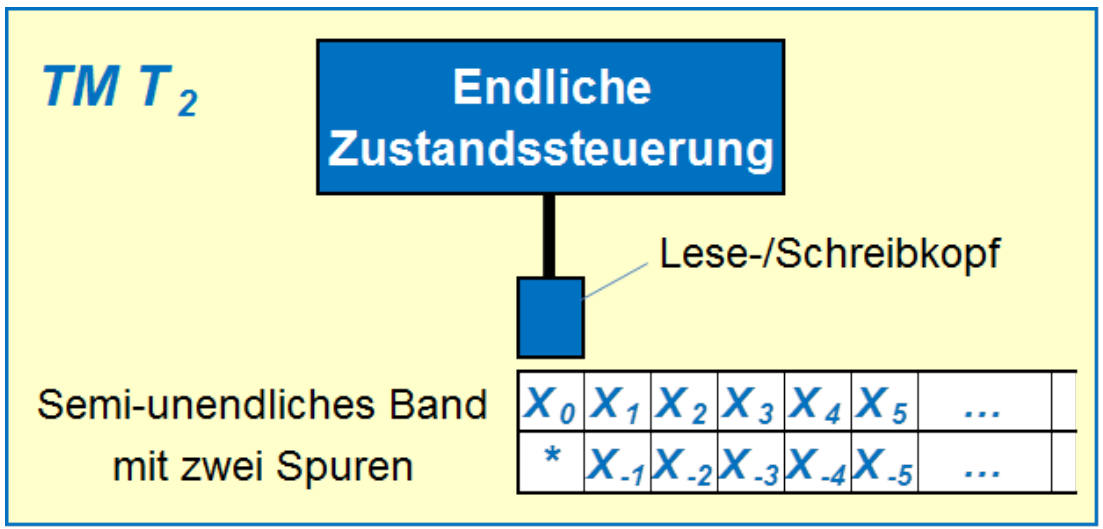
Beschränkungen von TM

TM mit **semi-unendlichem Band (beschränktem Band)**

Satz

Jede Sprache L, die von einer TM akzeptiert wird, wird auch von einer TM mit semi-unendlichem Band akzeptiert.

Beweiskonstruktion: Simulation der TM T_1 über eine TM T_2 mit semi-unendlichem Band, das zwei Spuren beinhaltet:



Beschränkungen von TM

TM mit **semi-unendlichem Band (beschränktem Band)**

Beweiskonstruktion (Fortsetzung)[5]: Simulation der TM T_1 über eine TM T_2 mit semi-unendlichem Band, das zwei Spuren beinhaltet:

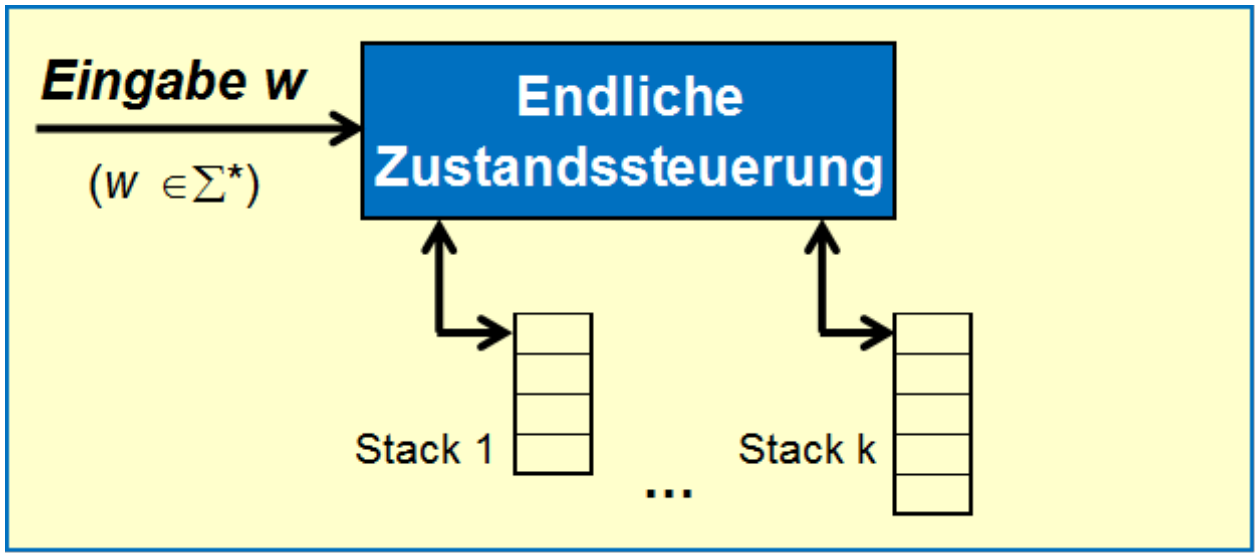
- Im oberen Band von T_2 stehen die Symbole, die auf dem Band von T_1 rechts von der Anfangsposition des Lese-/Schreibkopfes stehen.
- Im unteren Band stehen in umgekehrter Reihenfolge alle Symbole, die in T_1 links von der Anfangsposition des Lese-/Schreibkopfes stehen.
- Im unteren Band von T_1 steht in der ersten Zelle zur Markierung ein Sonderzeichen.
- Alle Übergänge in T_1 müssen entsprechend angepasst und ergänzt werden (z. B. auch die Richtung D).
- T_1 bewegt sich in der oberen oder unteren Spur.

[5] Siehe auch Hopcroft et al. S. 356 ff.

Beschränkungen von TM

Maschine mit mehreren Stacks (k-Stack-Maschine)

- Eine k-Stack-Maschine ist ein DKA mit k statt nur einem Stack.



- Die Übergangsfunktion δ hängt vom Zustand der endlichen Zustandssteuerung und den obersten Symbolen der k Stacks ab:

$$\delta \colon Q \times \Gamma_1 \times \ldots \times \Gamma_n \to Q \times \Gamma_1^* \times \ldots \times \Gamma_k^*$$

Maschine mit mehreren Stacks (k-Stack-Maschine)

Anmerkungen:

- In einer Bewegung kann eine k-Stack-Maschine in einen neuen Zustand wechseln und das oberste Symbol auf jedem der k Stacks unabhängig voneinander durch ein Symbol oder eine (evtl. leere) Zeichenreihe des Bandalphabets Γ ersetzen.
- $\delta(A \ldots, K)$ ist immer höchstens 1-elementig (deterministisch).

Beschränkungen von TM

Maschine mit mehreren Stacks (k-Stack-Maschine)

Satz

Jede Sprache L, die von einer TM T akzeptiert wird, wird auch von einer 2-Stack-Maschine S akzeptiert.

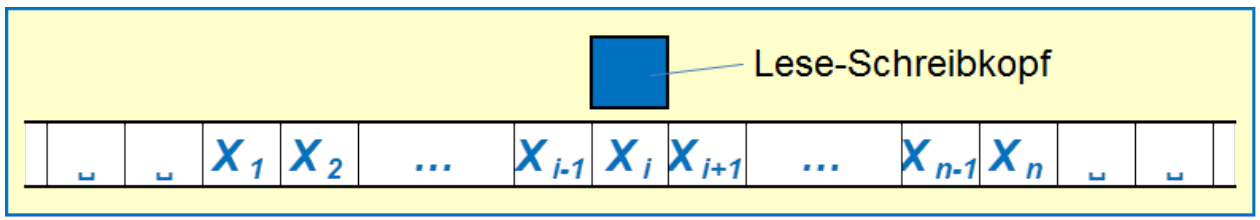
Beweiskonstruktion: Simulation der TM T mit S. Grundidee ist die Nachbildung des Bandes mit zwei Stacks:

- S liest die Eingabe zunächst in den 1. Stack.
- Nun simuliert S die Bewegung von T: Abhängig von der Bewegungsrichtung von T wird ein Symbol vom 1. oder 2. Stack von S gelesen (*pop*) und neu ggf. modifiziert in den jeweils anderen Stack geschrieben.

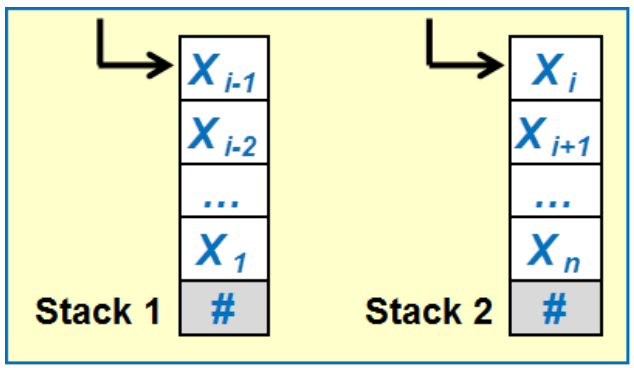
Maschine mit mehreren Stacks (k-Stack-Maschine)

Beweiskonstruktion (Fortsetzung): Simulation der TM T mit S. Grundidee ist die Nachbildung des Bandes mit zwei Stacks:

Band von T:



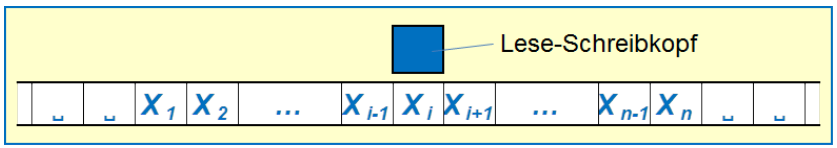
Stacks von S:

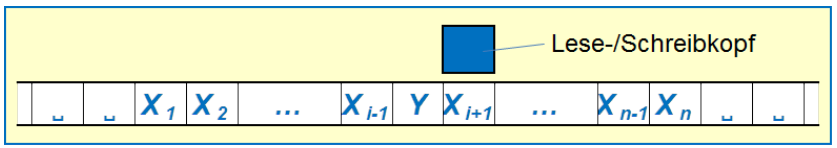


Maschine mit mehreren Stacks (k-Stack-Maschine)

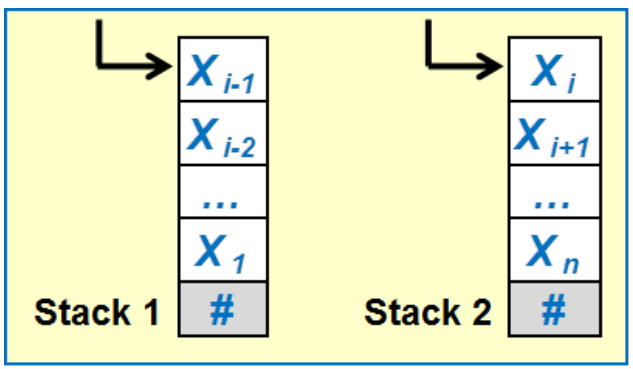
Beweiskonstruktion (Fortsetzung): Simulation der TM T mit S. Grundidee ist die Nachbildung des Bandes mit zwei Stacks:

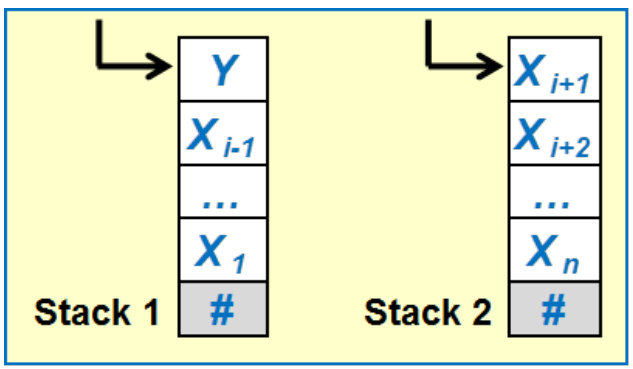
- In T: $\delta(q, X_i) \to (p, Y, R)$:





- In S: $\delta(q, X_{i-1}, X_i) \to (p, Y X_{i-1}, \epsilon)$:





Backup – Beschränkungen von TM

Frage: Geht noch „weniger“?

Kann die Funktion einer Maschine / eines Automaten vordergründig noch weiter eingeschränkt werden und dennoch wird dieselbe Sprachklasse wie die der TM akzeptiert (äquivalent zu "gleich mächtig"')?

Backup – Beschränkungen von TM

Zähler-Maschinen

Eine Zähler-Maschine mit k Zählern entspricht einer k-Stack-Maschine (Eingabeband und endliche Kontrolle) mit dem Unterschied, dass die Stacks durch einfache Zähler ersetzt werden:

- Die Zähler können eine natürliche Zahl speichern.
- Die Zähler können unabhängig voneinander um 1 erhöht oder um 1 verringert werden (der Wert kann aber nicht kleiner 0 werden).
- Es kann nur zwischen Zählern, die gleich 0 sind, und Zählern, die ungleich 0 sind, unterschieden werden.

Backup – Beschränkungen von TM

Zähler-Maschinen

Satz

Jede Sprache L, die von einer TM T akzeptiert wird, wird auch von einer Zähler-Maschine Z mit 2 Zählern akzeptiert.

Beweiskonstruktion:

a) Ein 2-Stack-Maschine kann eine TM simulieren.

b) Eine Zählermaschine mit 3 Zählern kann eine 2-Stack-Maschine simulieren.

c) Eine Zählermaschine mit 2 Zählern kann eine Zählermaschine mit 3 Zählern simulieren.

Backup – Beschränkungen von TM

Zähler-Maschinen

Beweiskonstruktion (Fortsetzung):

a) Ein 2-Stack-Maschine kann eine TM simulieren. *Bereits gezeigt* ✓

Zähler-Maschinen

Beweiskonstruktion (Fortsetzung):

b) Eine Zählermaschine mit 3 Zählern kann eine 2-Stack-Maschine simulieren.

- Zeigen, wie ein Stack mit einer Zahl dargestellt werden kann!

  Annahme: Es gibt k-1 Stacksymbole und n Symbole auf einem Stack s.
  Die Stacksymbole werden über die Zahlen 1 bis k-1 repräsentiert.
  Der Zustand eines Stacks $s = X_1 X_2 \ldots X_n$ kann dann als ganze Zahl zur Basis k: $s = X_n k^{n-1} + X_{n-1} k^{n-2} + \ldots + X_2 k + X_1$ abgebildet werden.

Beispiel

$\Gamma = \{A, B, C\} => A \to 1, B \to 2$ und $C \to 3 (k = 4)$.

Der Stack s mit dem Inhalt $ACBAC$ ergibt:

$1 + 3 \cdot 4^1 + 2 \cdot 4^2 + 1 \cdot 4^3 + 3 \cdot 4^4 = 1 + 12 + 32 + 64 + 768 = 877$

Zähler-Maschinen

Beweiskonstruktion (Fortsetzung):

b) Eine Zählermaschine mit 3 Zählern kann eine 2-Stack-Maschine simulieren (Fortsetzung).

- Zeigen, wie ein Stack mit einer Zahl dargestellt werden kann! ✓
- Zeigen wie ein Symbol vom Stack entfernt und zugefügt werden kann (als Veränderung der Zahl).

  Symbol entfernen (pop): $\rightarrow$ Zahl durch k teilen.
  Symbol X_i hinzufügen (push): $\rightarrow$ Zahl mit k multiplizieren und Zahl für Symbol X_i addieren.
- Zähler 1 und 2 stellen die Inhalte der beiden Stacks dar, mit Zähler 3 wird "gerechnet".

Backup – Beschränkungen von TM

Zähler-Maschinen

Beweiskonstruktion (Fortsetzung):

a) Ein 2-Stack-Maschine kann eine TM simulieren. *Gezeigt* ✓

b) Eine Zählermaschine mit 3 Zählern kann eine 2-Stack-Maschine simulieren. *Gezeigt* ✓

c) Eine Zählermaschine mit 2 Zählern kann eine Zählermaschine mit 3 Zählern simulieren.

   Grundansatz:

   - Die drei Zähler i, j und k werden als eine Zahl 2^i, 3^j und 5^k eindeutig dargestellt und in einem Zähler gespeichert.
   - Der 2. Zähler wird zum Rechnen genutzt.

Übersicht: Chomsky-Hierarchie

Grammatik	Regeln	Sprachen	Entscheidbarkeit	Automaten
Typ-0 Beliebige formale Grammatik	$\alpha \rightarrow \beta$ $\alpha \in V^* \setminus \Sigma^*, \beta \in V^*$	rekursiv aufzählbar	–	Turingmaschine
Typ-1 Kontextsensitive Grammatik	$\alpha A \beta \rightarrow \alpha \gamma \beta$ $A \in N, \alpha, \beta \in V^*, \gamma \in V^+$ $S \rightarrow \varepsilon$ ist erlaubt, wenn es keine Regel $\alpha \rightarrow \beta S \gamma$ in P gibt.	kontextsensitiv	Wortproblem	linear platzbeschränkte nichtdeterministische Turingmaschine
Typ-2 Kontextfreie Grammatik	$A \rightarrow \gamma$ $A \in N, \gamma \in V^*$	kontextfrei	Wortproblem, Leerheitsproblem, Endlichkeitsproblem	nichtdeterministischer Kellerautomat
Typ-3 Reguläre Grammatik	$S \rightarrow \varepsilon$ $A \rightarrow aB$ (rechtsregulär) oder $A \rightarrow Ba$ (linksregulär) $A \rightarrow a$ $A \rightarrow \varepsilon$ $A, B \in N, a \in \Sigma$ Nur links- oder nur rechtsreguläre Produktionen	regulär	Wortproblem, Leerheitsproblem, Äquivalenzproblem, Mehrdeutigkeitsproblem, Endlichkeitsproblem	Endlicher Automat