

# Theoretische Informatik I

## Einheit 4



## Allgemeine und kontextsensitive Sprachen



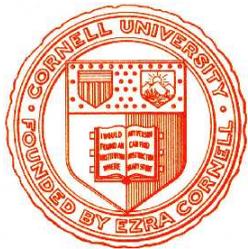
1. Turingmaschinen
2. Maschinenmodelle für  $\mathcal{L}_0$  und  $\mathcal{L}_1$
3. Eigenschaften von  $\mathcal{L}_0/\mathcal{L}_1$ -Sprachen

- **Viele wichtige Konzepte sind nicht kontextfrei**
  - Sind **Bezeichner** im Programmkörper deklariert?
  - $\{ ww \mid w \in \{0, 1\}^* \}$ : erscheint Programmcode doppelt?
  - $\{ 0^n 1^n 2^n \mid n \in \mathbb{N} \}$ : kommen mehrere Bestandteile gleich oft vor?
  - Zählen jenseits von Addition und Multiplikation
- **Wie verarbeitet man Typ-1 / Typ-0 Sprachen?**
  - Welches **Maschinenmodell** ist zur Beschreibung geeignet?
  - Wie **analysiert** man Wörter der Sprache
  - Wie kann man Sprachen **aus Bausteinen zusammensetzen**?
  - Welche **Spracheigenschaften** kann man testen?

# Theoretische Informatik I

## Einheit 4.1

### Turingmaschinen



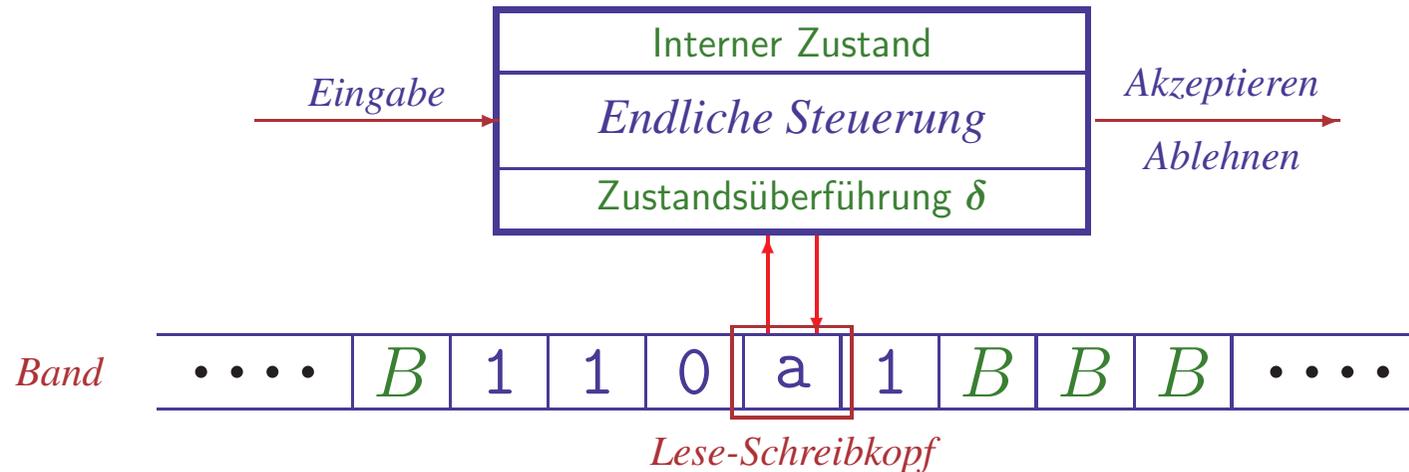
1. Das Maschinenmodell
2. Arbeitsweise & erkannte Sprache
3. Programmiertechniken
4. Ausdruckskraft

## Maschinenmodell für Typ-0 Sprachen

- **Erweiterung des Konzepts endlicher Automaten**
  - Verarbeitung interner Zustände abhängig von gelesenen Daten
  - Lese- und Schreibzugriff auf externen Speicher
  - Minimal mögliche Erweiterung
- **Maximal mögliche Ausdruckskraft**
  - Speicher muß Fähigkeiten von Typ-0 Grammatiken widerspiegeln
    - Keine Einschränkung an Ersetzungsregeln
    - Auch Terminalsymbole und ganze Wörter dürfen ersetzt werden
  - Automat muß Eingabe an jeder Stelle verarbeiten können
    - Gesamte Eingabe muß gespeichert werden
    - Speicher muß Veränderungen an jeder Stelle zulassen
    - Speicher muß beliebig erweiterbar sein

**Wähle unendliches, bewegliches Band als Speicher**

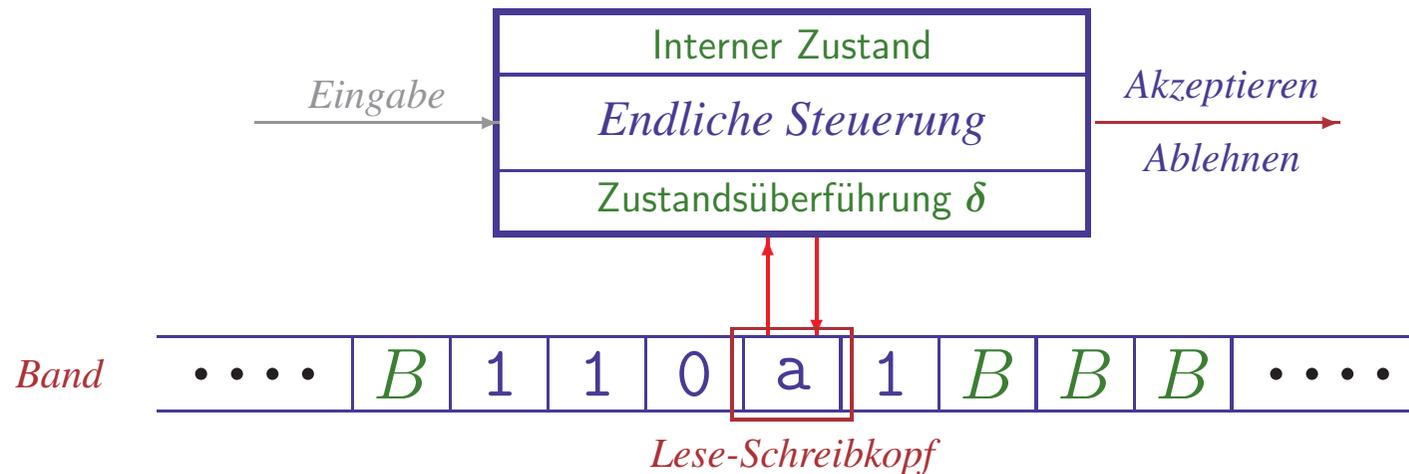
# TURINGMASCHINEN INTUITIV



- **Endlicher Automat + lineares Band**

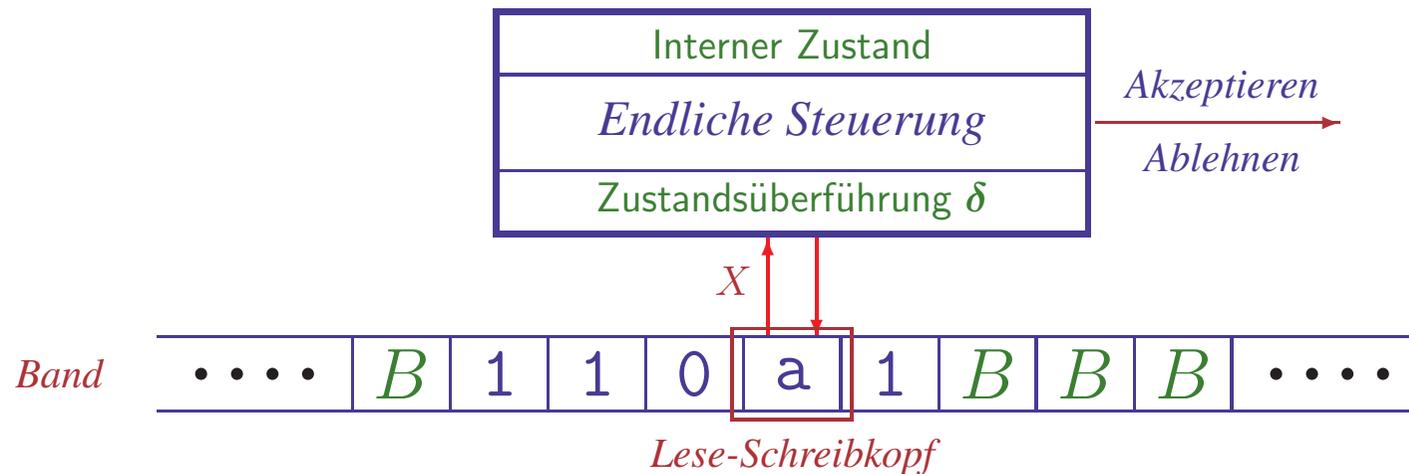
- Endliche Steuerung liest Eingabesymbole
- Gleichzeitig wird Bandsymbol unter **Lese-Schreibkopf** gelesen

# TURINGMASCHINEN INTUITIV



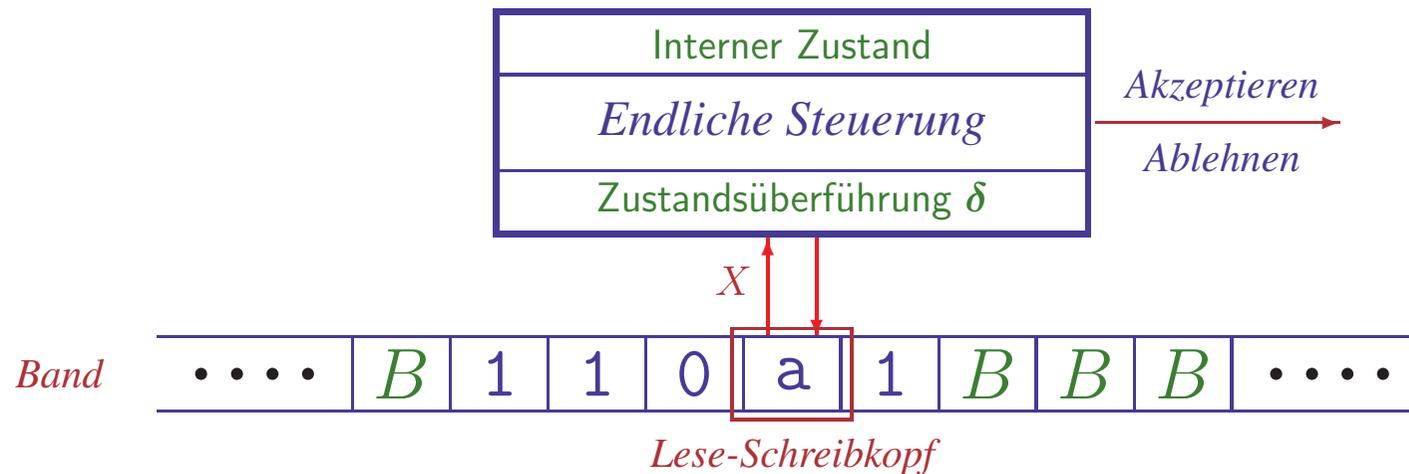
- **Endlicher Automat + lineares Band**
  - Endliche Steuerung liest Eingabesymbole
  - Gleichzeitig wird Bandsymbol unter **Lese-Schreibkopf** gelesen
- **Vereinfachung: keine separate Eingabe**
  - Eingabewort steht zu Anfang bereits auf dem Band

# TURINGMASCHINEN INTUITIV



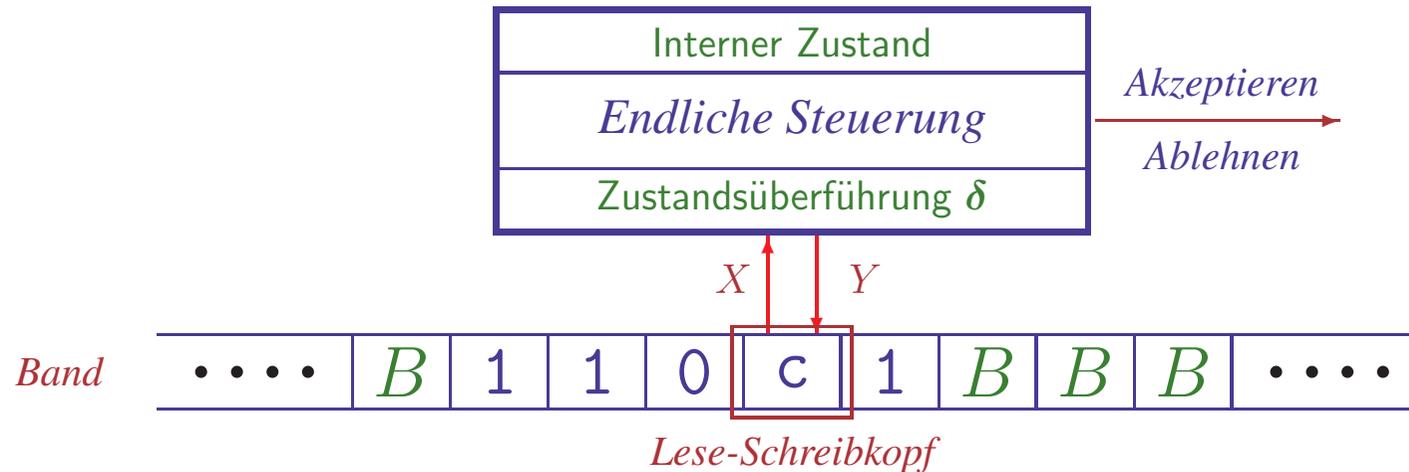
- **Endlicher Automat + lineares Band**
  - Endliche Steuerung liest Eingabesymbole
  - Gleichzeitig wird Bandsymbol unter **Lese-Schreibkopf** gelesen
- **Vereinfachung: keine separate Eingabe**
  - Eingabewort steht zu Anfang bereits auf dem Band
- **Einfacher Verarbeitungsmechanismus**
  - Bandsymbol  $X$  wird gelesen

# TURINGMASCHINEN INTUITIV



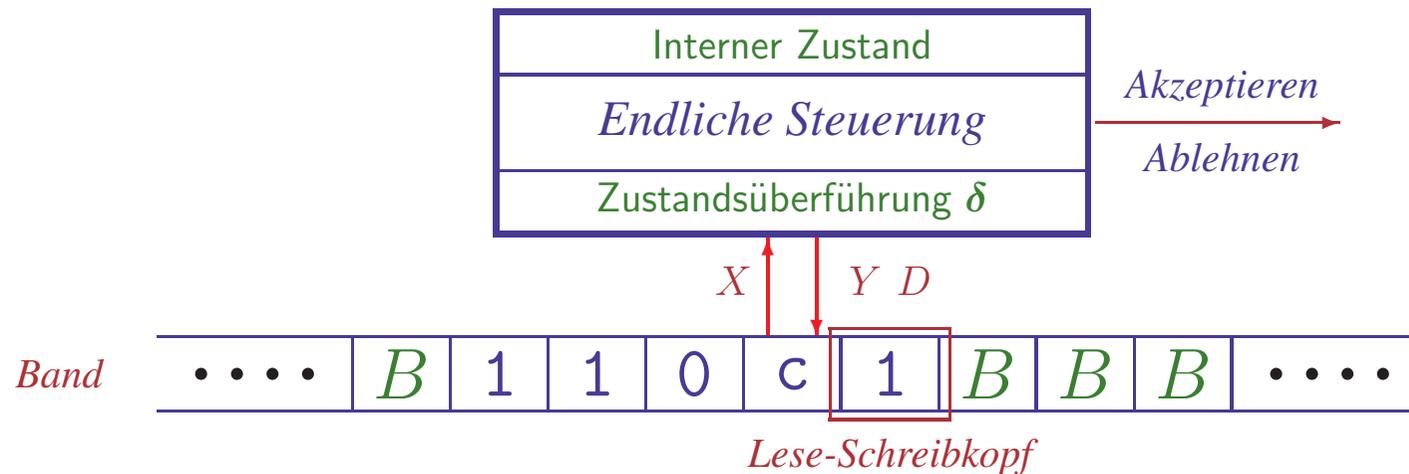
- **Endlicher Automat + lineares Band**
  - Endliche Steuerung liest Eingabesymbole
  - Gleichzeitig wird Bandsymbol unter **Lese-Schreibkopf** gelesen
- **Vereinfachung: keine separate Eingabe**
  - Eingabewort steht zu Anfang bereits auf dem Band
- **Einfacher Verarbeitungsmechanismus**
  - Bandsymbol  $X$  wird gelesen
  - Interner Zustand  $q$  wird zu  $q'$  verändert

# TURINGMASCHINEN INTUITIV



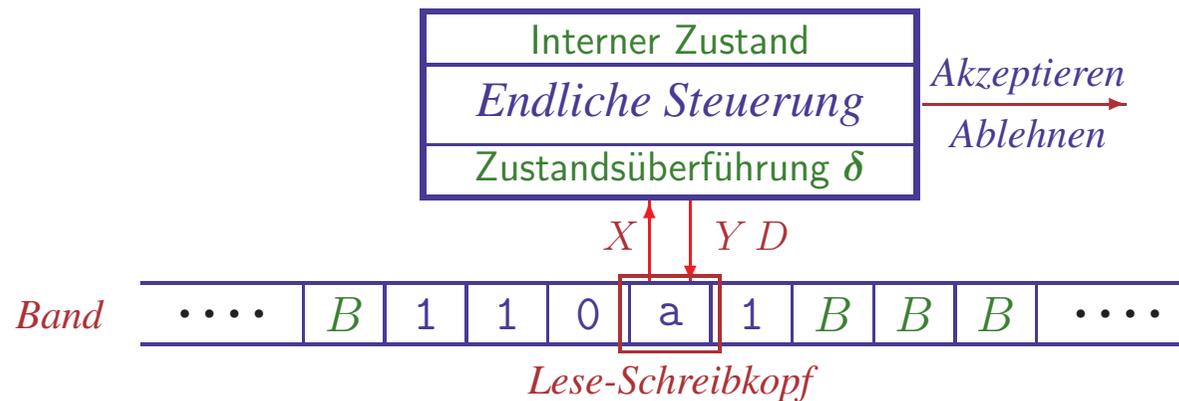
- **Endlicher Automat + lineares Band**
  - Endliche Steuerung liest Eingabesymbole
  - Gleichzeitig wird Bandsymbol unter **Lese-Schreibkopf** gelesen
- **Vereinfachung: keine separate Eingabe**
  - Eingabewort steht zu Anfang bereits auf dem Band
- **Einfacher Verarbeitungsmechanismus**
  - Bandsymbol  $X$  wird gelesen
  - Interner Zustand  $q$  wird zu  $q'$  verändert
  - Neues Symbol  $Y$  wird auf das Band geschrieben

# TURINGMASCHINEN INTUITIV



- **Endlicher Automat + lineares Band**
  - Endliche Steuerung liest Eingabesymbole
  - Gleichzeitig wird Bandsymbol unter **Lese-Schreibkopf** gelesen
- **Vereinfachung: keine separate Eingabe**
  - Eingabewort steht zu Anfang bereits auf dem Band
- **Einfacher Verarbeitungsmechanismus**
  - Bandsymbol  $X$  wird gelesen
  - Interner Zustand  $q$  wird zu  $q'$  verändert
  - Neues Symbol  $Y$  wird auf das Band geschrieben
  - Kopf wird in eine Richtung  $D$  (rechts oder links) bewegt

# TURINGMASCHINEN – MATHEMATISCH PRÄZISIERT



Eine **Turingmaschine (TM)** ist ein 7-Tupel

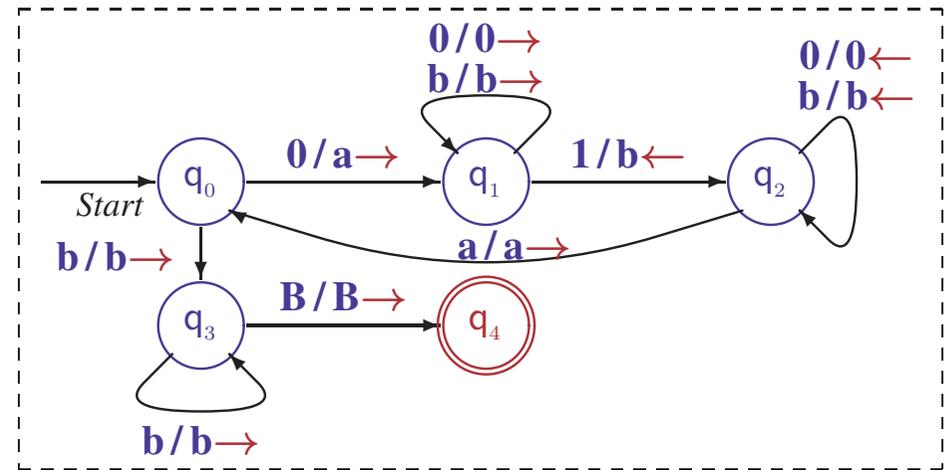
$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  mit

- $Q$  nichtleere endliche **Zustandsmenge**
- $\Sigma$  endliches **Eingabealphabet**
- $\Gamma \supseteq \Sigma$  endliches **Bandalphabet**
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$  (partielle) **Überföhrungsfunktion**
- $q_0 \in Q$  **Startzustand**
- $B \in \Gamma \setminus \Sigma$  **Leersymbol des Bands** (“blank”)
- $F \subseteq Q$  Menge von **akzeptierenden (End-)Zuständen**

# BESCHREIBUNG VON TURINGMASCHINEN

## • Übergangsdiagramme

- Zustände durch **Knoten** dargestellt
- $q_0$  markiert durch *Start*-Pfeil, Endzustände durch **doppelte Kreise**
- Für  $\delta(q, X) = (p, Y, D)$  hat das Diagramm eine **Kante**  $q \xrightarrow{X/YD} p$
- $\Sigma$  und  $\Gamma$  oft implizit durch Diagramm bestimmt, **Leersymbol** heißt  $B$



## • Übergangstabellen

- Funktionstabelle für  $\delta$   
— heißt “ $\delta$  nicht definiert”
- Pfeil  $\rightarrow$  kennzeichnet  $q_0$
- Stern  $*$  kennzeichnet  $F$
- $\Sigma$ ,  $\Gamma$  (und  $B$ ) oft implizit bestimmt

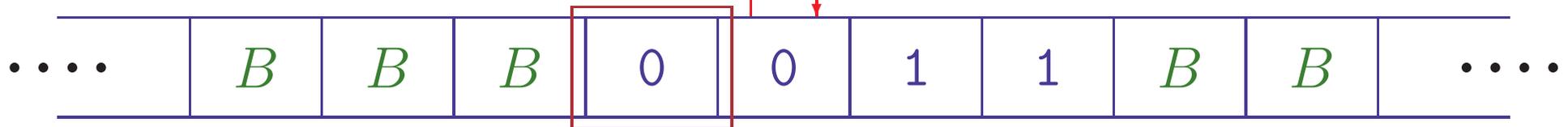
$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	$(q_1, a, R)$	—	—	$(q_3, b, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, b, L)$	—	$(q_1, b, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, a, R)$	$(q_2, b, L)$	—
$q_3$	—	—	—	$(q_3, b, R)$	$(q_4, B, R)$
$* q_4$	—	—	—	—	—

# ABARBEITUNG VON TURING-PROGRAMMEN

$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	$(q_1, a, R)$	—	—	$(q_3, b, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, b, L)$	—	$(q_1, b, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, a, R)$	$(q_2, b, L)$	—
$q_3$	—	—	—	$(q_3, b, R)$	$(q_4, B, R)$
$* q_4$	—	—	—	—	—

*Akzeptieren*

*Ablehnen*



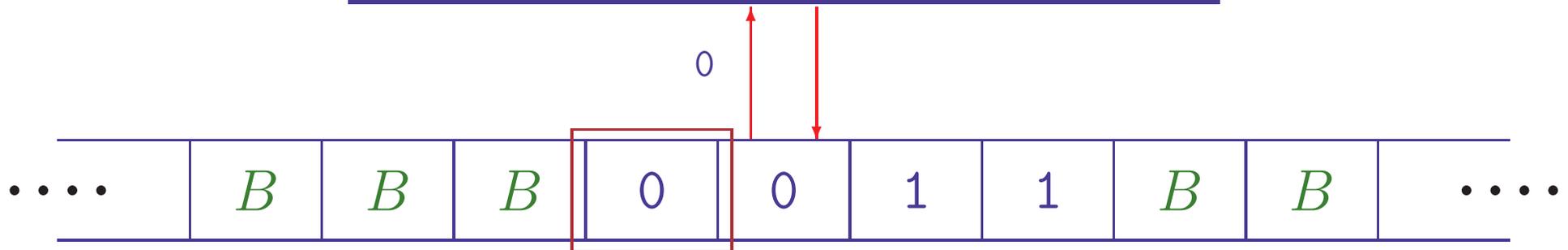
Startzustand  $q_0$

# ABARBEITUNG VON TURING-PROGRAMMEN

$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	$(q_1, a, R)$	—	—	$(q_3, b, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, b, L)$	—	$(q_1, b, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, a, R)$	$(q_2, b, L)$	—
$q_3$	—	—	—	$(q_3, b, R)$	$(q_4, B, R)$
$* q_4$	—	—	—	—	—

*Akzeptieren*

*Ablehnen*



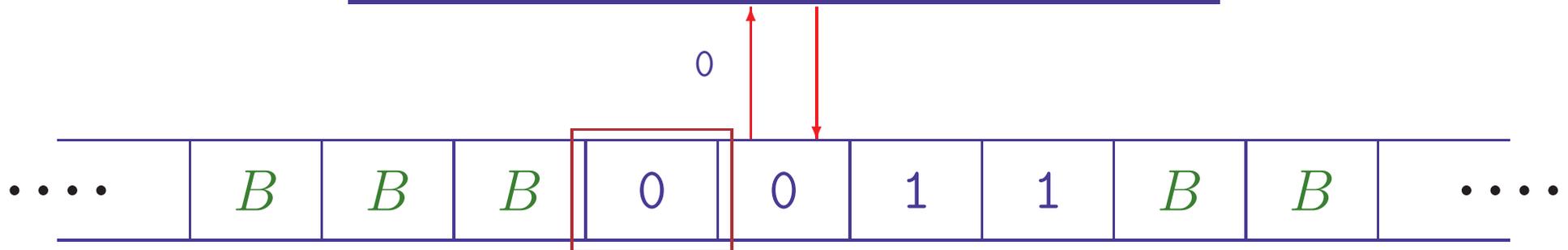
Lesen der Eingabe 0

# ABARBEITUNG VON TURING-PROGRAMMEN

$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	$(q_1, a, R)$	—	—	$(q_3, b, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, b, L)$	—	$(q_1, b, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, a, R)$	$(q_2, b, L)$	—
$q_3$	—	—	—	$(q_3, b, R)$	$(q_4, B, R)$
$* q_4$	—	—	—	—	—

Akzeptieren

Ablehnen



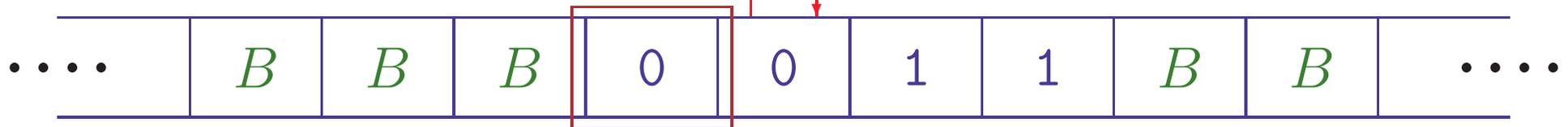
Tabelleneintrag für  $q_0/0$

# ABARBEITUNG VON TURING-PROGRAMMEN

$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	$(q_1, a, R)$	—	—	$(q_3, b, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, b, L)$	—	$(q_1, b, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, a, R)$	$(q_2, b, L)$	—
$q_3$	—	—	—	$(q_3, b, R)$	$(q_4, B, R)$
$* q_4$	—	—	—	—	—

*Akzeptieren*

*Ablehnen*



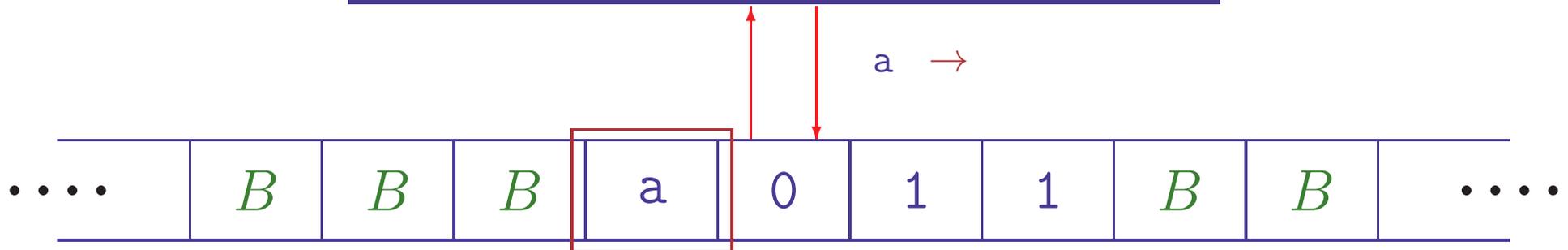
Neuer Zustand  $q_1$

# ABARBEITUNG VON TURING-PROGRAMMEN

$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	<span style="border: 1px solid red;">(q<sub>1</sub>, a, R)</span>	—	—	(q <sub>3</sub> , b, R)	—
<span style="border: 1px solid red;">q<sub>1</sub></span>	(q <sub>1</sub> , 0, R)	(q <sub>2</sub> , b, L)	—	(q <sub>1</sub> , b, R)	—
q <sub>2</sub>	(q <sub>2</sub> , 0, L)	—	(q <sub>0</sub> , a, R)	(q <sub>2</sub> , b, L)	—
q <sub>3</sub>	—	—	—	(q <sub>3</sub> , b, R)	(q <sub>4</sub> , B, R)
* q <sub>4</sub>	—	—	—	—	—

*Akzeptieren*

*Ablehnen*



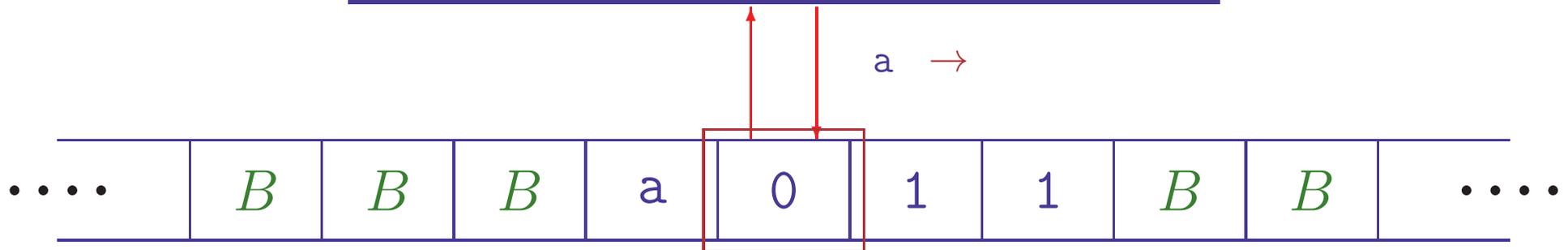
Neuer Wert ist a

# ABARBEITUNG VON TURING-PROGRAMMEN

$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	<span style="border: 1px solid red;">(q<sub>1</sub>, a, R)</span>	—	—	(q <sub>3</sub> , b, R)	—
<span style="border: 1px solid red;">q<sub>1</sub></span>	(q <sub>1</sub> , 0, R)	(q <sub>2</sub> , b, L)	—	(q <sub>1</sub> , b, R)	—
q <sub>2</sub>	(q <sub>2</sub> , 0, L)	—	(q <sub>0</sub> , a, R)	(q <sub>2</sub> , b, L)	—
q <sub>3</sub>	—	—	—	(q <sub>3</sub> , b, R)	(q <sub>4</sub> , B, R)
* q <sub>4</sub>	—	—	—	—	—

*Akzeptieren*

*Ablehnen*



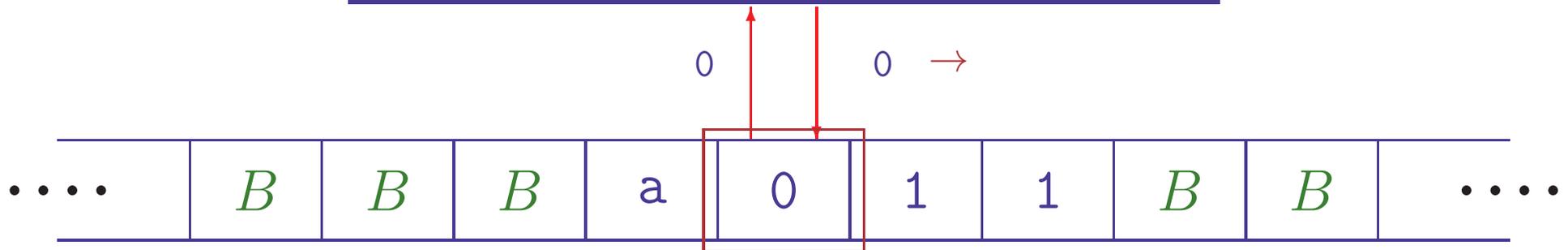
Kopf geht nach rechts

# ABARBEITUNG VON TURING-PROGRAMMEN

$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	$(q_1, a, R)$	—	—	$(q_3, b, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, b, L)$	—	$(q_1, b, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, a, R)$	$(q_2, b, L)$	—
$q_3$	—	—	—	$(q_3, b, R)$	$(q_4, B, R)$
$* q_4$	—	—	—	—	—

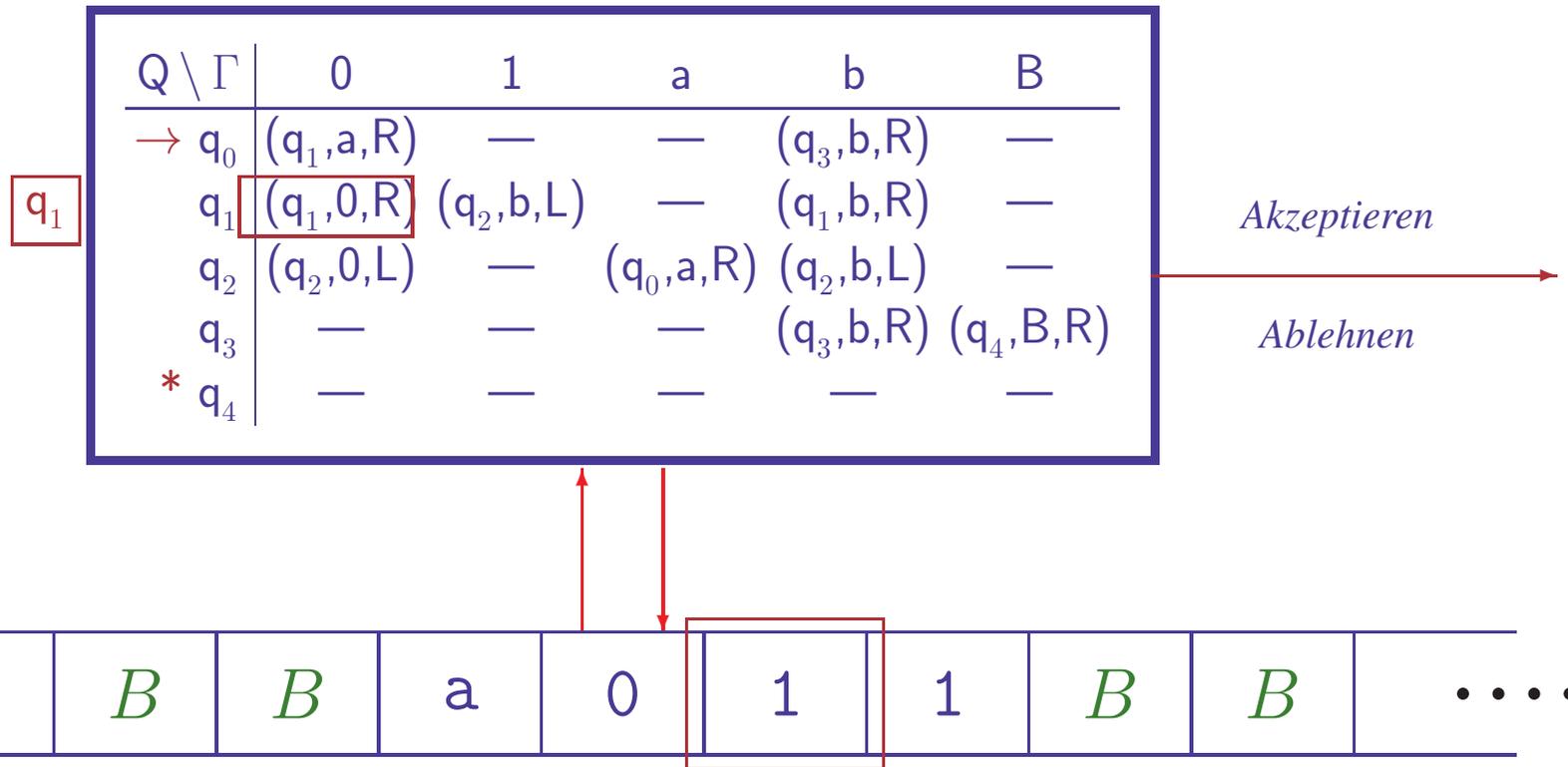
*Akzeptieren*

*Ablehnen*



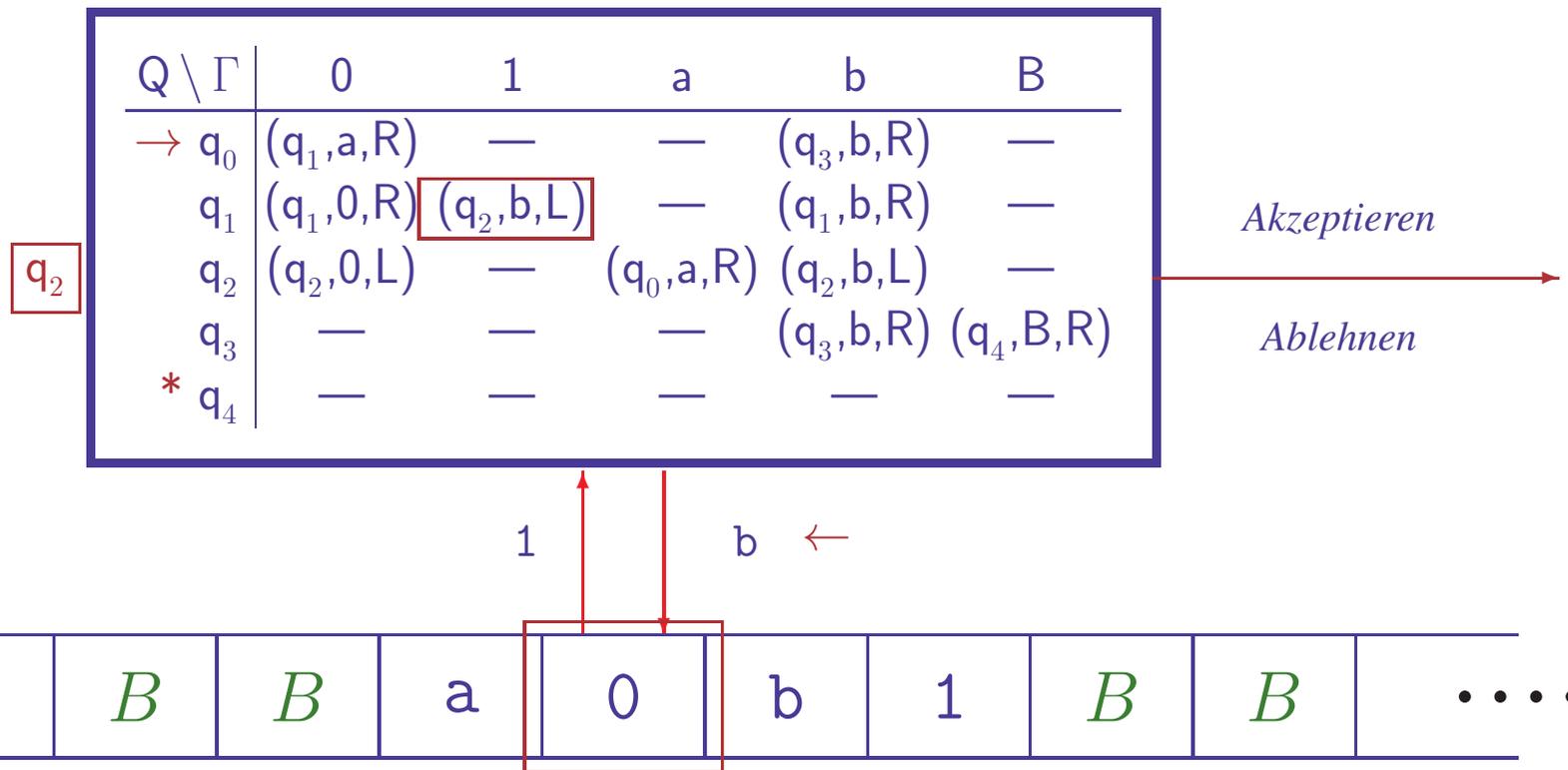
Lesen der Eingabe 0

# ABARBEITUNG VON TURING-PROGRAMMEN



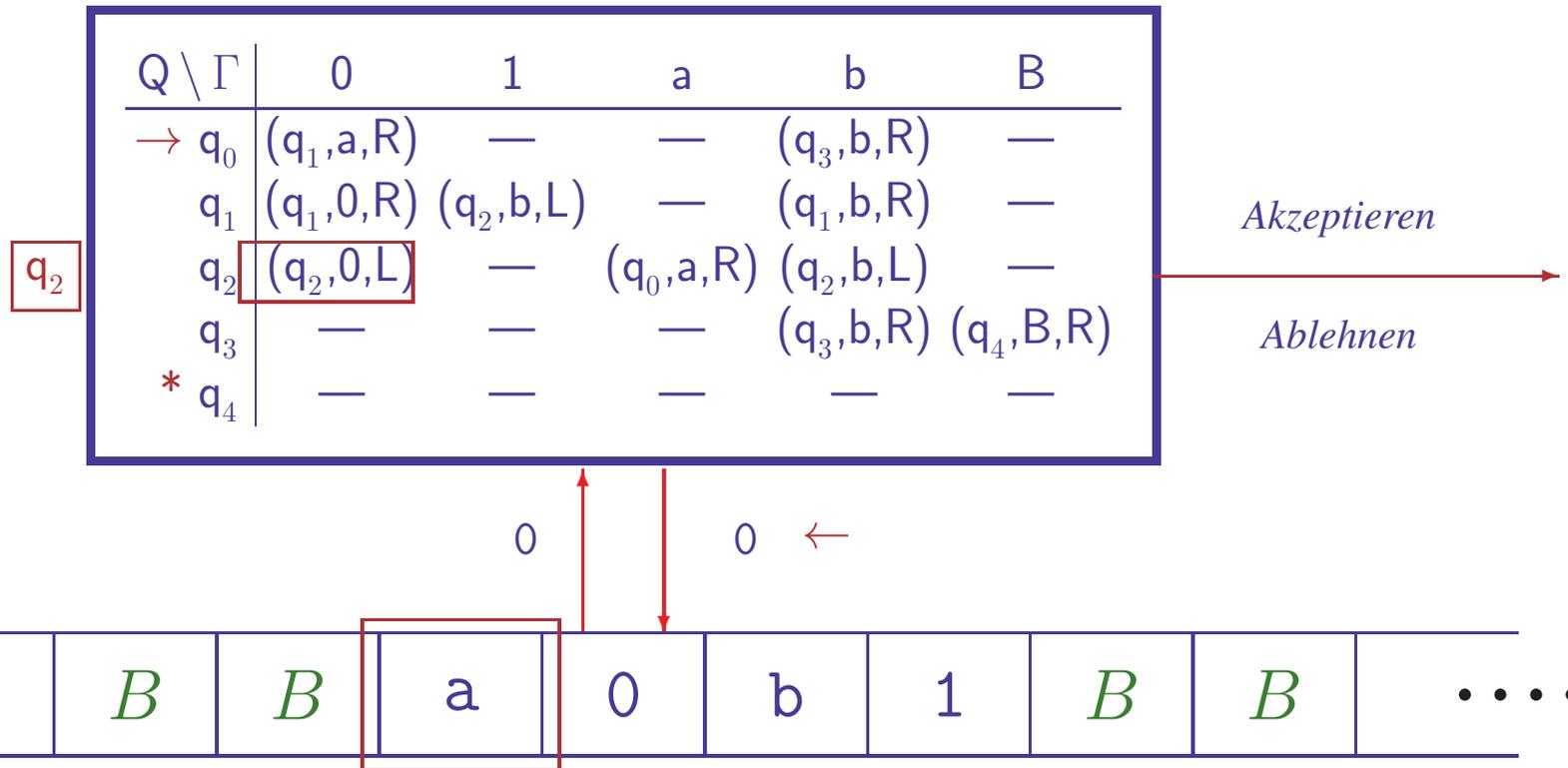
neuer Zustand  $q_1$ , Wert 0, Kopf nach rechts

# ABARBEITUNG VON TURING-PROGRAMMEN



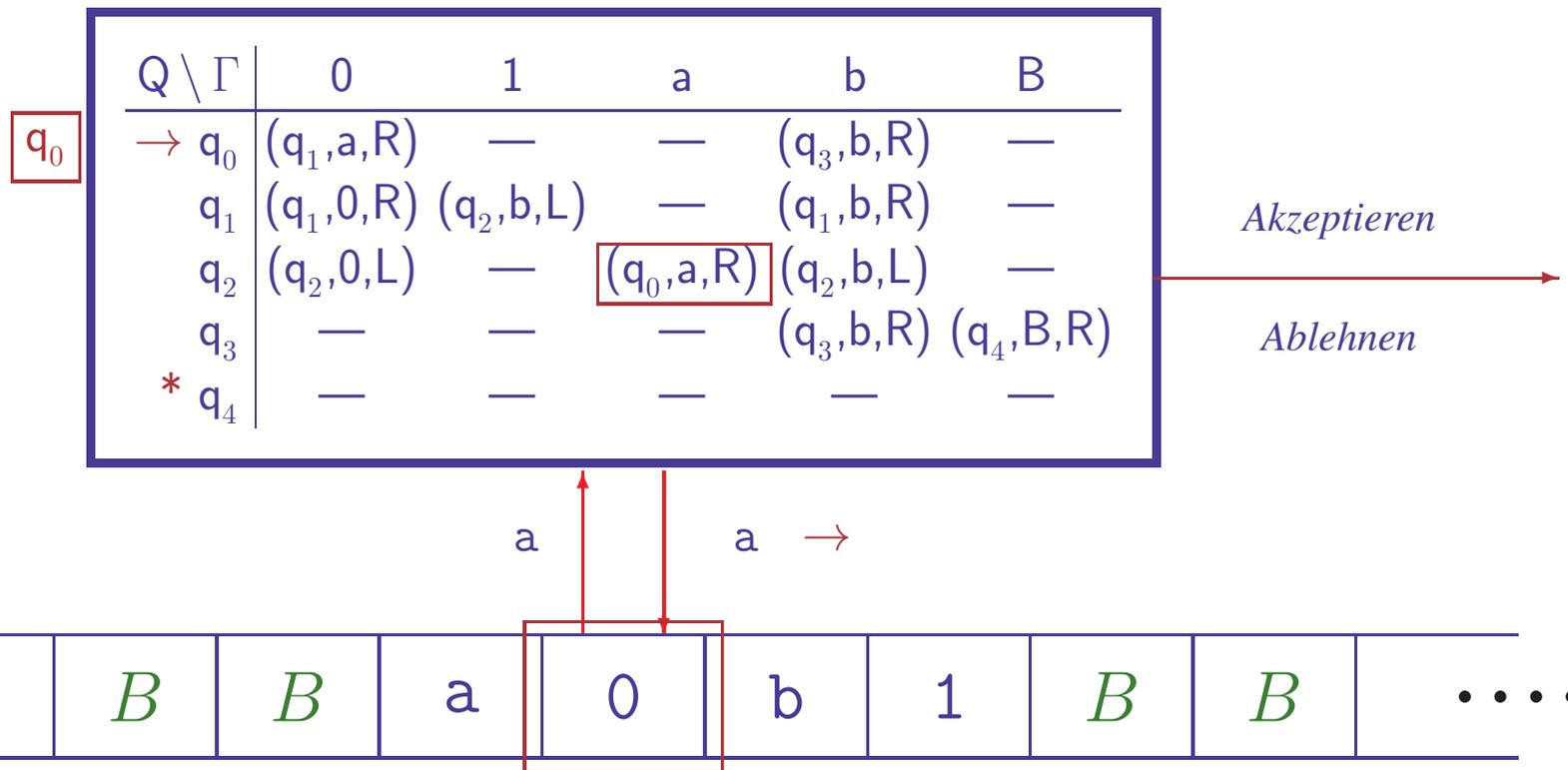
Eingabe 1, neuer Zustand  $q_2$ , Wert b, Kopf nach links

# ABARBEITUNG VON TURING-PROGRAMMEN



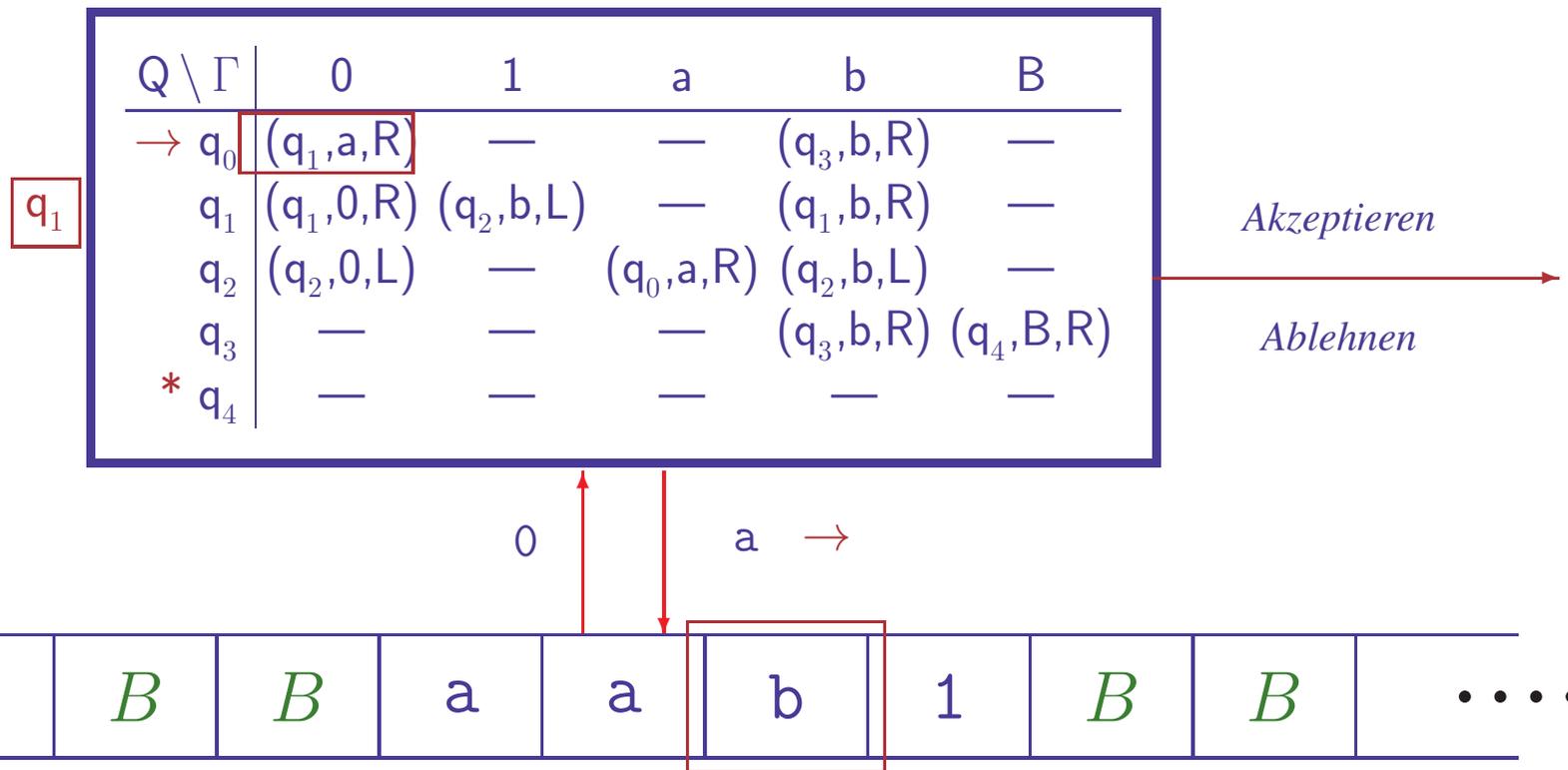
Eingabe 0, neuer Zustand  $q_2$ , Wert 0, Kopf nach links

# ABARBEITUNG VON TURING-PROGRAMMEN



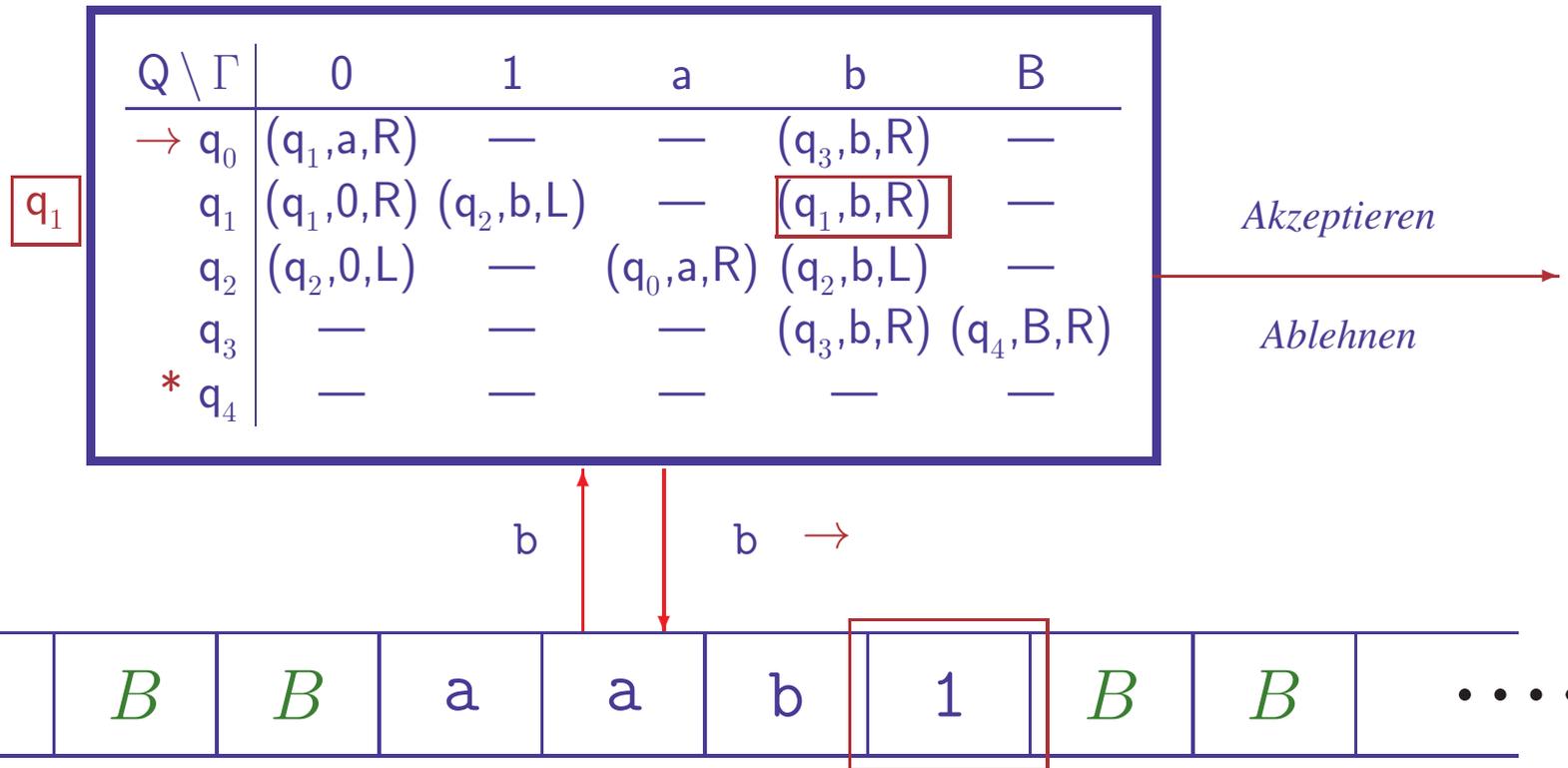
Eingabe a, neuer Zustand  $q_0$ , Wert a, Kopf nach rechts

# ABARBEITUNG VON TURING-PROGRAMMEN



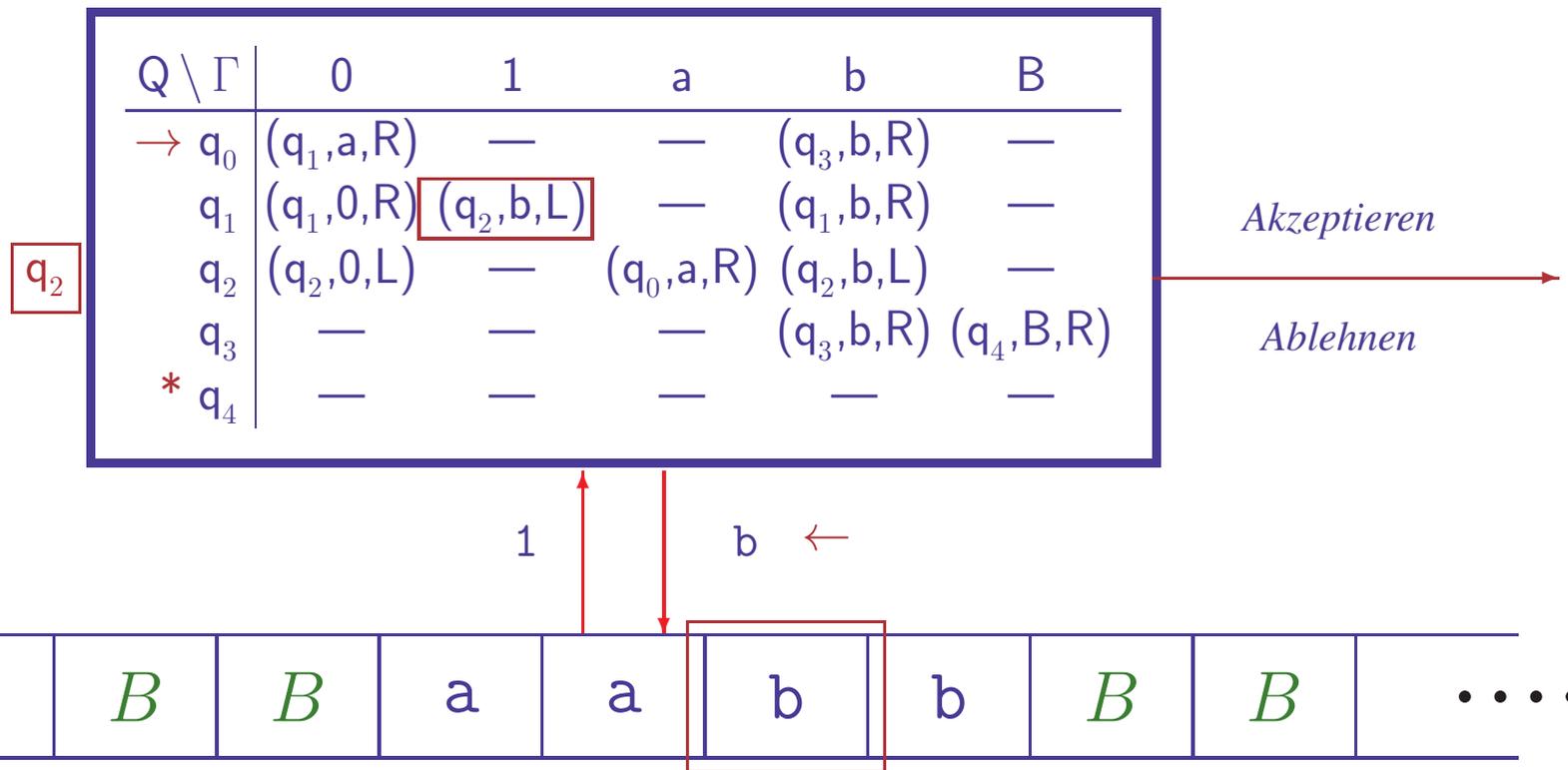
Eingabe 0, neuer Zustand  $q_1$ , Wert a, Kopf nach rechts

# ABARBEITUNG VON TURING-PROGRAMMEN



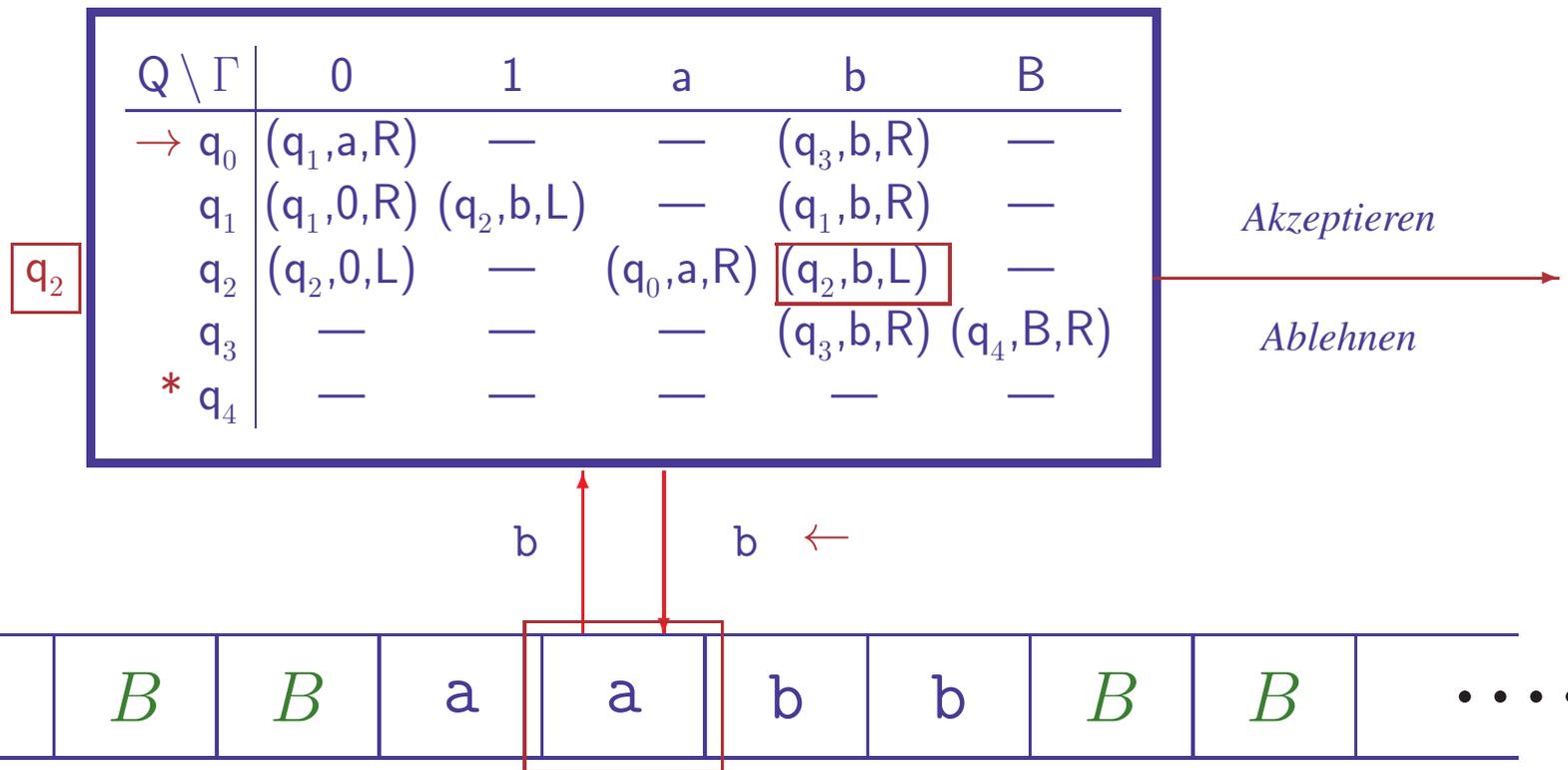
Eingabe b, neuer Zustand  $q_1$ , Wert b, Kopf nach rechts

# ABARBEITUNG VON TURING-PROGRAMMEN



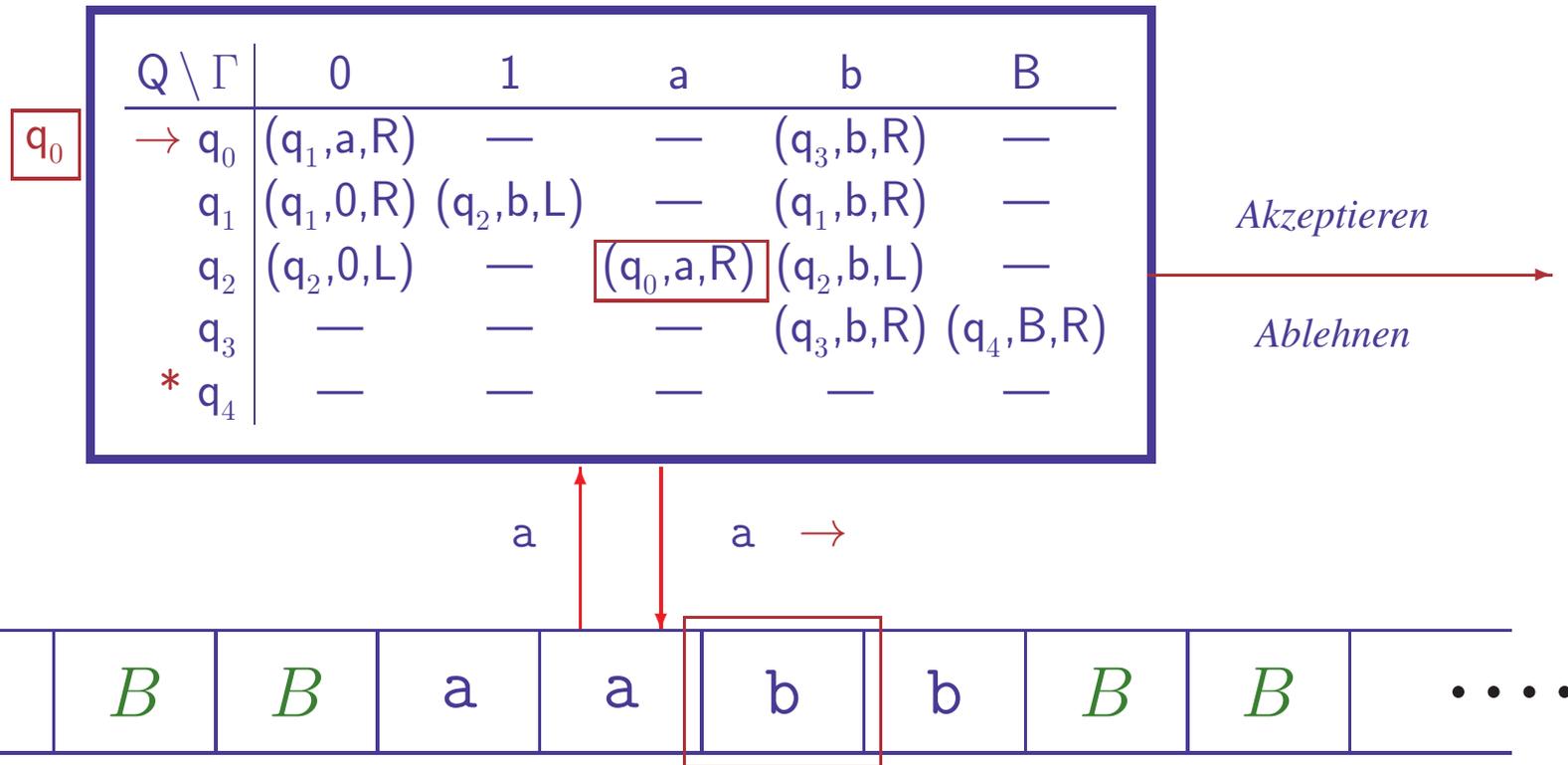
Eingabe 1, neuer Zustand  $q_2$ , Wert b, Kopf nach links

# ABARBEITUNG VON TURING-PROGRAMMEN



Eingabe b, neuer Zustand  $q_2$ , Wert b, Kopf nach links

# ABARBEITUNG VON TURING-PROGRAMMEN



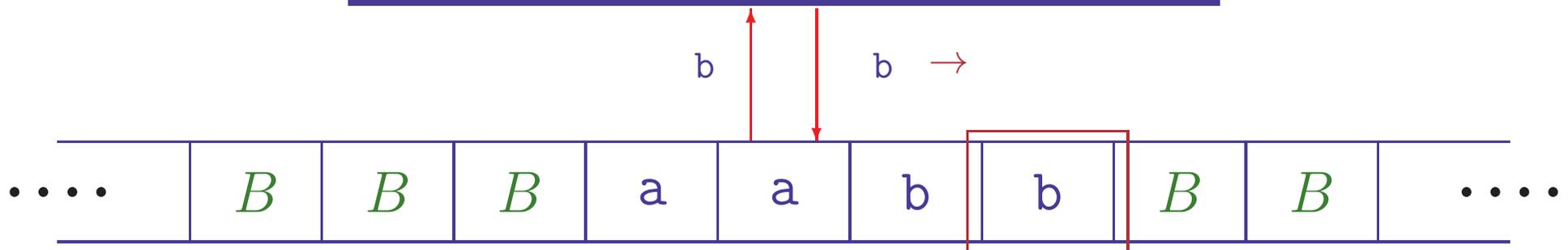
Eingabe a, neuer Zustand  $q_0$ , Wert a, Kopf nach rechts

# ABARBEITUNG VON TURING-PROGRAMMEN

$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	$(q_1, a, R)$	—	—	$(q_3, b, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, b, L)$	—	$(q_1, b, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, a, R)$	$(q_2, b, L)$	—
$q_3$	—	—	—	$(q_3, b, R)$	$(q_4, B, R)$
$* q_4$	—	—	—	—	—

Akzeptieren

Ablehnen



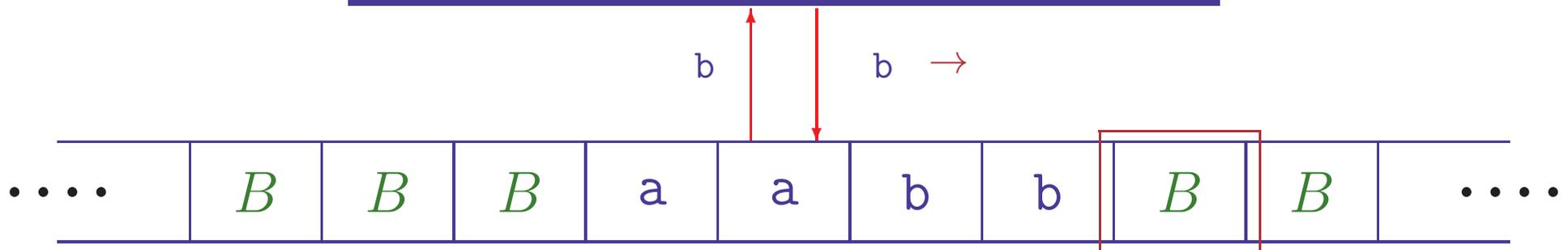
Eingabe  $b$ , neuer Zustand  $q_3$ , Wert  $b$ , Kopf nach rechts

# ABARBEITUNG VON TURING-PROGRAMMEN

$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	$(q_1, a, R)$	—	—	$(q_3, b, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, b, L)$	—	$(q_1, b, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, a, R)$	$(q_2, b, L)$	—
$q_3$	—	—	—	$(q_3, b, R)$	$(q_4, B, R)$
$* q_4$	—	—	—	—	—

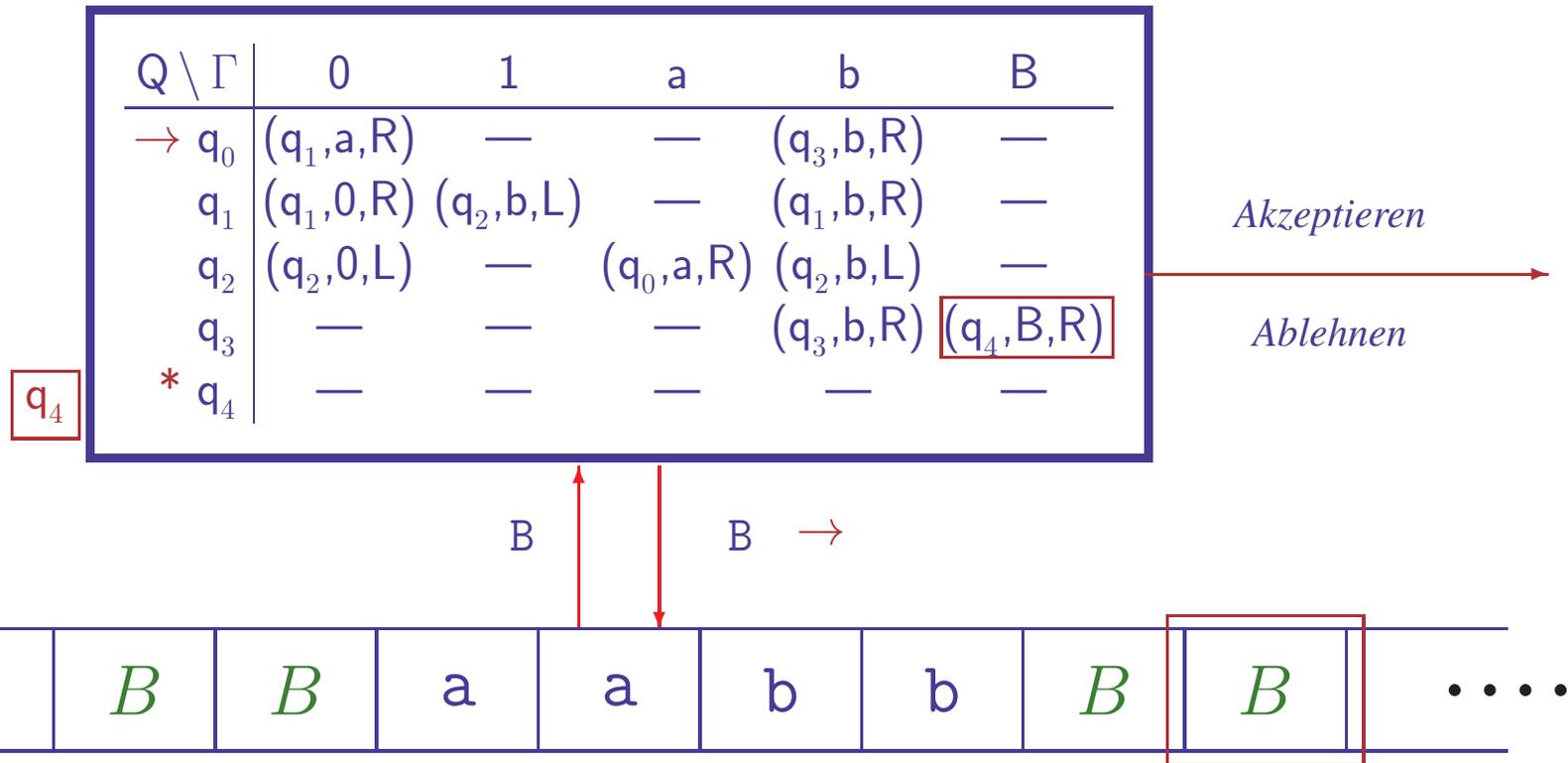
Akzeptieren

Ablehnen



Eingabe b, neuer Zustand  $q_3$ , Wert b, Kopf nach rechts

# ABARBEITUNG VON TURING-PROGRAMMEN



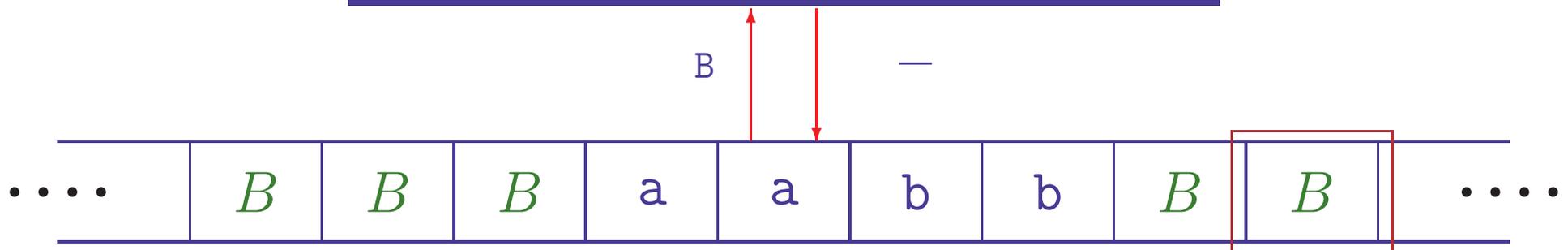
Eingabe B, neuer Zustand  $q_4$ , Wert B, Kopf nach rechts

# ABARBEITUNG VON TURING-PROGRAMMEN

$Q \setminus \Gamma$	0	1	a	b	B
$\rightarrow q_0$	$(q_1, a, R)$	—	—	$(q_3, b, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, b, L)$	—	$(q_1, b, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, a, R)$	$(q_2, b, L)$	—
$q_3$	—	—	—	$(q_3, b, R)$	$(q_4, B, R)$
$* q_4$	—	—	—	—	—

Akzeptieren

Ablehnen



**Maschine hält im Endzustand  $q_4$  an**

# ARBEITSWEISE VON TURINGMASCHINEN INTUITIV

- **Anfangssituation**

- Eingabewort  $w$  steht auf dem Band, umgeben von Leerzeichen
- Kopf ist über erstem Symbol, Startzustand ist  $q_0$

- **Arbeitsschritt**

- Im Zustand  $q$  lese Bandsymbol  $X$  und bestimme  $\delta(q, X) = (p, Y, D)$
- Wechsle in Zustand  $p$ , schreibe  $Y$  aufs Band, bewege Kopf gemäß  $D$

- **Terminierung, wenn  $\delta(q, X)$  nicht definiert**

- Alternativ: Maschine hält bei Erreichen eines Endzustands
- **Konvention:  $\delta(q, X)$  undefiniert für Endzustände  $q \in F$**

- **Ergebnis**

- Eingabewort  $w$  wird **akzeptiert**, wenn Maschine im Endzustand anhält

- **Hilfsmittel zur Präzisierung: Konfigurationen**

- Verallgemeinere bekanntes Konzept der Konfigurationsübergänge

**Details in Literatur sehr unterschiedlich!!**

- **Erweitere Begriff der Konfiguration**

- Zustand  $q$ , Inhalt des Bandes und Kopfposition

- Formal dargestellt als Tripel  $K = (u, q, v) \in \Gamma^* \times Q \times \Gamma^+$

- $u, v$ : String links/rechts vom Kopf

Achtung: im Buch wird das Tripel als ein (!) String  $uqv$  geschrieben

- Nur der bereits ‘besuchten’ Teil des Bandes wird betrachtet

- Blanks am Anfang von  $u$  oder am Ende von  $v$  entfallen, wo möglich

## • Erweitere Begriff der Konfiguration

- Zustand  $q$ , Inhalt des Bandes und Kopfposition
- Formal dargestellt als Tripel  $K = (u, q, v) \in \Gamma^* \times Q \times \Gamma^+$ 
  - $u, v$ : String links/rechts vom Kopf

Achtung: im Buch wird das Tripel als ein (!) String  $uqv$  geschrieben

- Nur der bereits ‘besuchten’ Teil des Bandes wird betrachtet

Blanks am Anfang von  $u$  oder am Ende von  $v$  entfallen, wo möglich

## • Modifiziere Konfigurationsübergangsrelation $\vdash^*$

- $(uZ, q, Xv) \vdash (u, p, ZYv)$ , falls  $\delta(q, X) = (p, Y, L)$
- $(u, q, Xv) \vdash (uY, p, v)$ , falls  $\delta(q, X) = (p, Y, R)$

Sonderfälle für Verhalten am Bandende

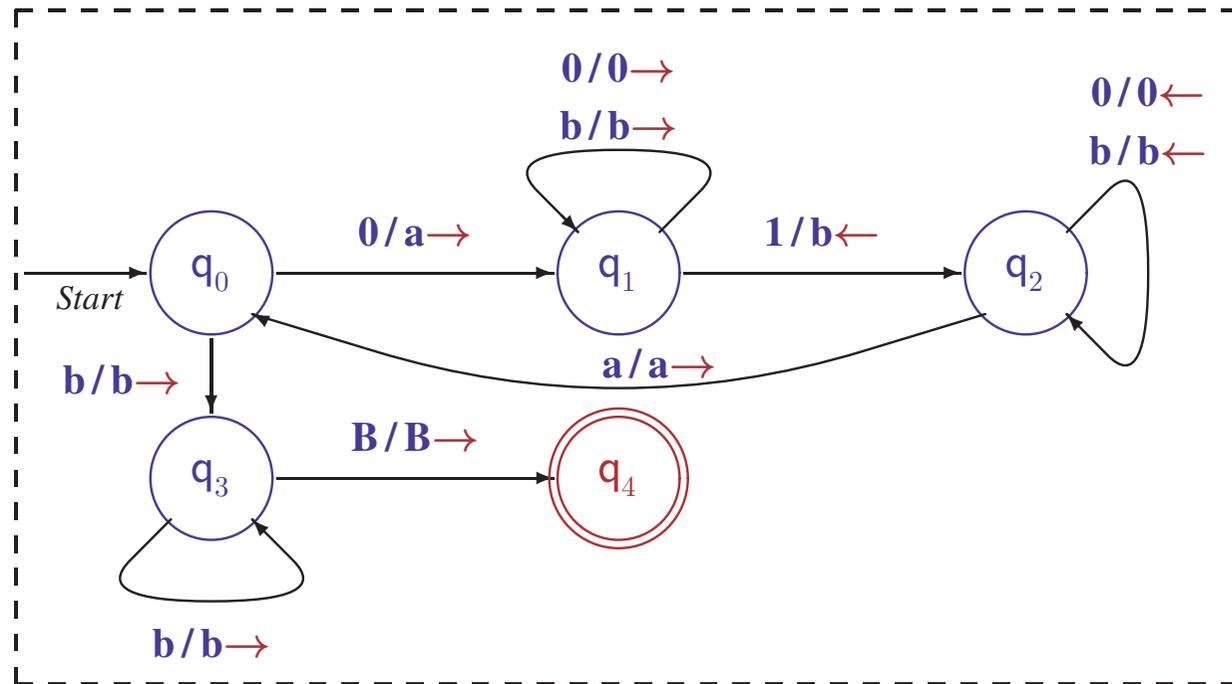
- $(\epsilon, q, Xv) \vdash (\epsilon, p, BYv)$ , falls  $\delta(q, X) = (p, Y, L)$
- $(uZ, q, X) \vdash (u, p, Z)$ , falls  $\delta(q, X) = (p, B, L)$
- $(u, q, X) \vdash (uY, p, B)$ , falls  $\delta(q, X) = (p, Y, R)$
- $(\epsilon, q, Xv) \vdash (\epsilon, p, v)$ , falls  $\delta(q, X) = (p, B, R)$

$K_1 \vdash^* K_2$ , falls  $K_1 = K_2$  oder es gibt ein  $K$  mit  $K_1 \vdash K$  und  $K \vdash^* K_2$

# VERARBEITUNG EINES EINGABEWORTES

**Eingabewort 0011 ergibt Anfangskonfiguration  $(\epsilon, q_0, 0011)$**

- $(\epsilon, q_0, 0011)$
- $\vdash (a, q_1, 011)$
- $\vdash (a0, q_1, 11)$
- $\vdash (a, q_2, 0b1)$
- $\vdash (\epsilon, q_2, a0b1)$
- $\vdash (a, q_0, 0b1)$
- $\vdash (aa, q_1, b1)$
- $\vdash (aab, q_1, 1)$
- $\vdash (aa, q_2, bb)$
- $\vdash (a, q_2, abb)$
- $\vdash (aa, q_0, bb)$
- $\vdash (aab, q_3, b)$
- $\vdash (aabb, q_3, B)$
- $\vdash (aabbB, q_4, B)$



Maschine terminiert,  
Endzustand erreicht,  
Eingabe wird akzeptiert

- **Akzeptierte Sprache**

- Menge der Eingaben, für die  $\vdash^*$  zu akzeptierendem Zustand führt

$$L(M) = \{w \in \Sigma^* \mid \exists p \in F. \exists u, v \in \Gamma^*. (\epsilon, q_0, w) \vdash^* (u, p, v)\}$$

Bei Einhalten der Konvention hält  $M$  im akzeptierenden Zustand an

# DIE SPRACHE EINER TURINGMASCHINE

- **Akzeptierte Sprache**

- Menge der Eingaben, für die  $\vdash^*$  zu akzeptierendem Zustand führt

$$L(M) = \{w \in \Sigma^* \mid \exists p \in F. \exists u, v \in \Gamma^*. (\epsilon, q_0, w) \vdash^* (u, p, v)\}$$

Bei Einhalten der Konvention hält  $M$  im akzeptierenden Zustand an

- **Semi-entscheidbare Sprache**

- Sprache, die von einer Turingmaschine  $M$  akzeptiert wird

- Alternative Bezeichnungen: **(rekursiv) aufzählbare Sprache**

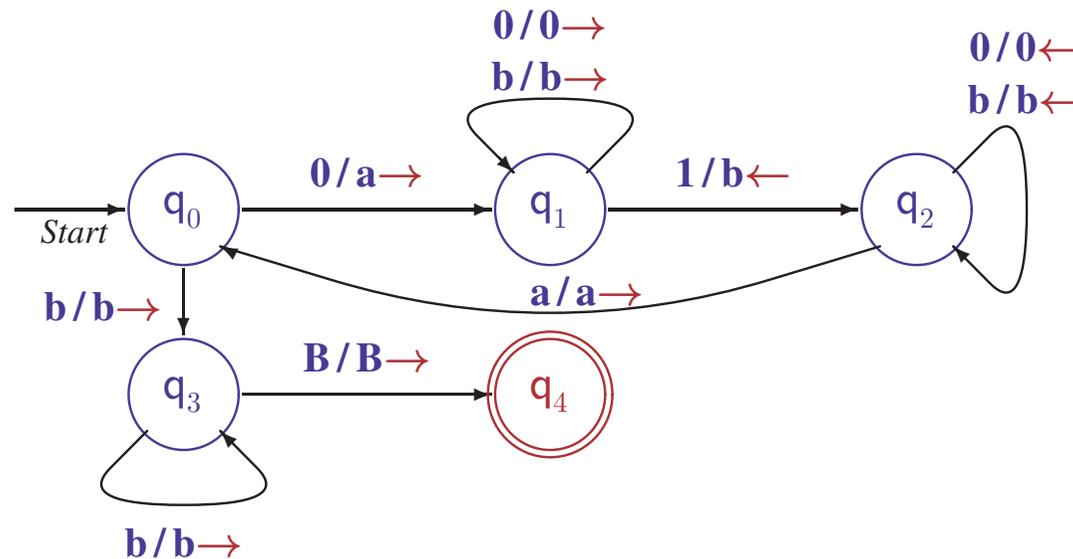
**Turing-akzeptierbare Sprache**

- **Entscheidbare Sprache**

- Sprache, die von einer Turingmaschine  $M$  akzeptiert wird,  
die bei jeder Eingabe terminiert

- Alternative Bezeichnung: **rekursive Sprache**

# ERKANNTE SPRACHE EINER TURINGMASCHINE



- **Analyse:  $M$  zählt Nullen und Einsen gleichzeitig**

- Umwandeln einer 0 in  $a$  triggert Umwandeln einer 1 in  $b$
- Maschine stoppt in  $q_0$ , wenn keine Nullen oder Einsen vorhanden sind
- Maschine stoppt in  $q_1$ , wenn zuwenig Einsen vorhanden sind
- Maschine stoppt in  $q_3$ , wenn zuwenig Nullen vorhanden sind
- Maschine akzeptiert in  $q_4$ , wenn Anzahl der Nullen und Einsen gleich

- **Es gilt  $L(M) = \{0^n 1^n \mid n \geq 1\}$**

- Beweis:  $(\epsilon, q_0, w) \vdash^* (u, q_4, v)$  genau dann, wenn  $w = 0^n 1^n$  für ein  $n \geq 1$

↪ Anhang, Folie 1

## Genauso leistungsfähig wie konventionelle Computer

- **Reale Computer bieten viele Freiheiten**

- Programme als Daten im Speicher
- Datenregister und Programmzähler
- “Simultaner” direkter Zugriff auf mehrere Speicherzellen
- Unterprogramme

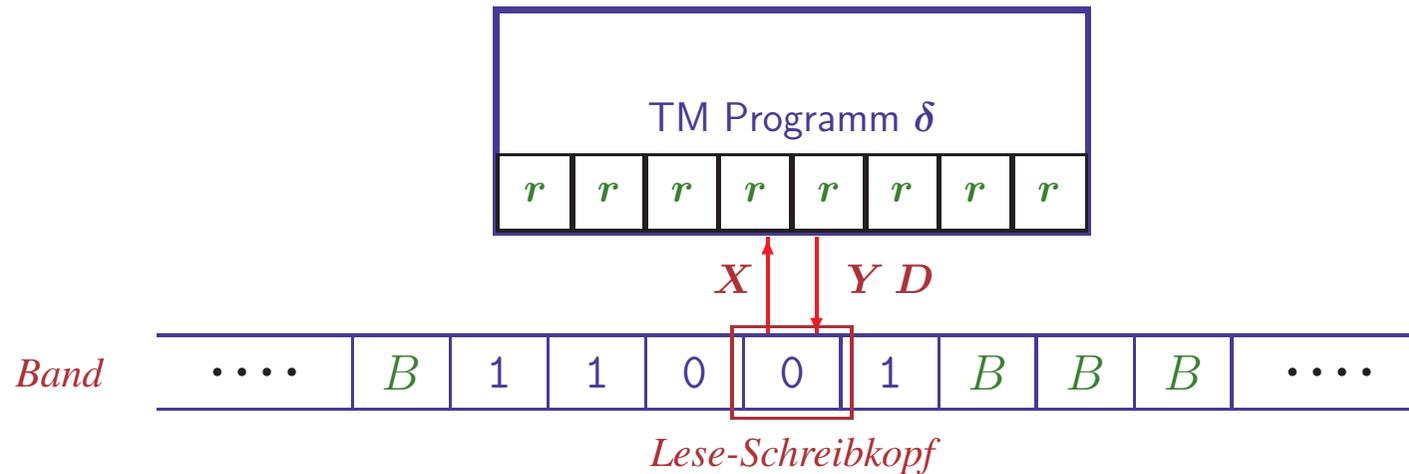
- **Turingmaschinen sind unbeschränkt**

- Beliebige große Alphabete (statt binären Daten)
- Unendliches Speicherband

- **Gegenseitige Simulation ist möglich**

- **Zusätzliche Freiheiten** als Programmier-Techniken einer TM simulierbar
- **Beschränkungen** des TM Modells verringern die Ausdruckskraft nicht

# PROGRAMMIERTECHNIK: DATENREGISTER



- **TM hat zusätzlich endliche Menge von Registern**
  - Jedes Register kann einen Wert aus einer endlichen Menge  $\Delta$  enthalten
  - Maschine kann jeweils eine Bandzelle und alle Register bearbeiten
  - Verwendung: Speichern einer Menge von Daten separat vom Band
- **Simulation durch erweiterte Zustandsmenge**
  - Bei  $k$  Registern wähle Zustandsmenge  $Q' := Q \times \Delta^k$
  - Simuliere Zustandsübergang in  $Q$  und Änderung der Register durch entsprechenden Zustandsübergang in  $Q'$

## Beschreibe Maschine, die $L((01^*)+(10^*))$ erkennt

- **Einfache Lösung mit Registern**

- Speichere erstes Bandsymbol im Register
- $q_0$ : Prüfe ob das gespeicherte Symbol im restlichen Wort vorkommt
- $q_1$ : Akzeptiere, wenn gesamtes Wort erfolgreich überprüft

# SIMULATION EINER MASCHINE MIT REGISTERN

## Beschreibe Maschine, die $L((01^*)+(10^*))$ erkennt

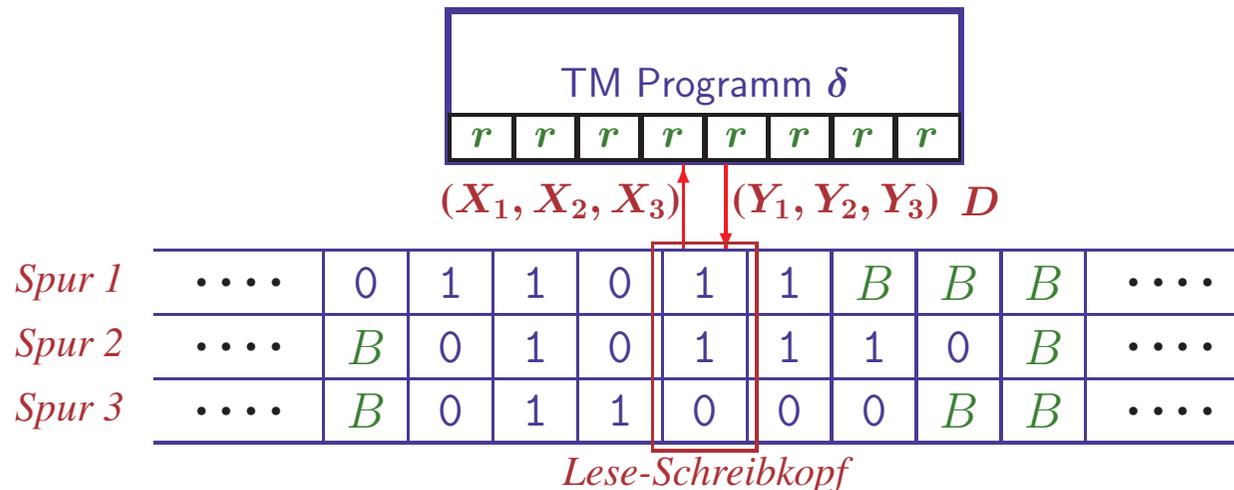
- **Einfache Lösung mit Registern**

- Speichere erstes Bandsymbol im Register
- $q_0$ : Prüfe ob das gespeicherte Symbol im restlichen Wort vorkommt
- $q_1$ : Akzeptiere, wenn gesamtes Wort erfolgreich überprüft

- **Simulation mit  $Q' := \{q_0, q_1\} \times \{0,1,B\}$**

	0	1	B	
$\rightarrow (q_0, B)$	$((q_0, 0), 0, R)$	$((q_0, 1), 1, R)$	—	<i>Erstes Symbol speichern</i>
$(q_0, 0)$	—	$((q_0, 0), 1, R)$	$((q_1, B), B, R)$	<i>Mit 0 vergleichen</i>
$(q_0, 1)$	$((q_0, 1), 0, R)$	—	$((q_1, B), B, R)$	<i>Mit 1 vergleichen</i>
$* (q_1, B)$	—	—	—	<i>Vergleich war erfolgreich</i>
$(q_1, 0)$	—	—	—	<i>(Nicht erreichbar)</i>
$(q_1, 1)$	—	—	—	<i>(Nicht erreichbar)</i>

# PROGRAMMIERTECHNIK: MEHRERE SPUREN



- **Band hat mehrere Datenspuren**

- Jede Spur enthält ein Symbol des Bandalphabets  $\Gamma$
- Alle Symbole werden simultan gelesen und geschrieben
- Kopf wird “synchron” über das Band bewegt
- Verwendung: **Simultane Verarbeitung von Teilen der Eingabe**

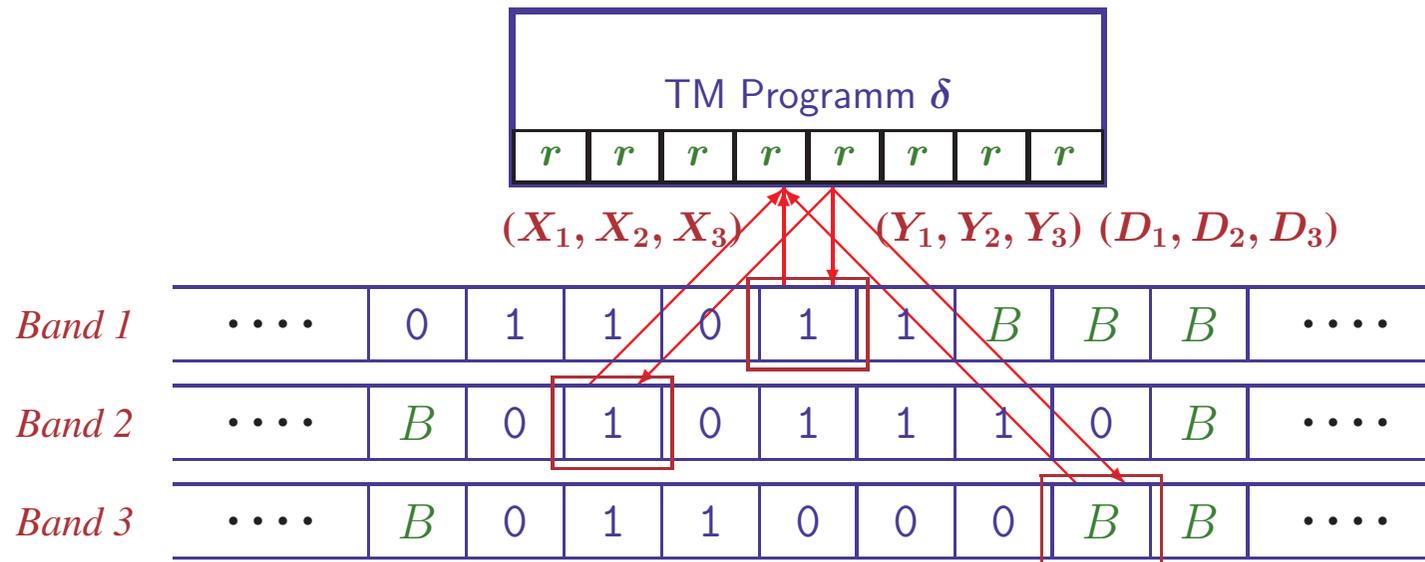
z.B. zur Erkennung von  $\{w\#w \mid w \in \{0, 1\}^*\}$

↪ HMU, §8.3.2

- **Simulation durch erweitertes Bandalphabet**

- Bei  $k$  Spuren wähle Tupelalphabet  $\Gamma' := \Gamma^k$
- In jedem Schritt wird ‘ein’ Symbol  $X := (x_1, \dots, x_k)$  verarbeitet, wobei  $x_i$  dem Symbol auf Spur  $i$  entspricht

# PROGRAMMIERTECHNIK: MEHRERE BÄNDER



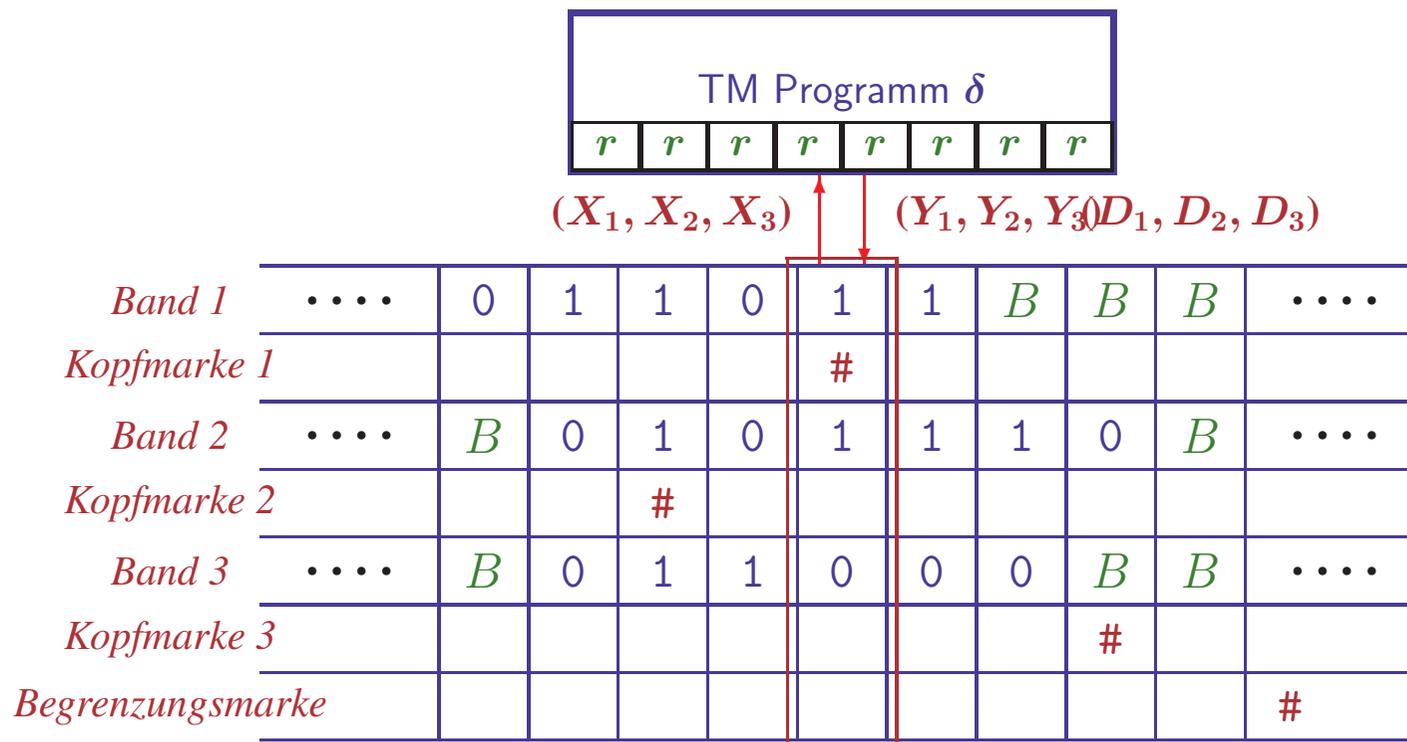
- **Maschine verwaltet mehrere Bänder**

- Jedes Band enthält ein Symbol des Bandalphabets  $\Gamma$
- Alle Symbole werden simultan gelesen und geschrieben
- Köpfe werden **unabhängig** über die Bänder bewegt
- Erheblich größere Freiheiten bei der Programmierung

- **Simulation aufwendiger**

- Mehrspurband + Verwaltung der Kopfpositionen auf separaten Spuren
- Spuren werden “einzeln aufgesucht” und modifiziert

# SIMULATION EINER MEHRBANDMASCHINE



## • Sequentielle Verarbeitung der einzelnen Bänder

- **Lesen:** Suche Begrenzungsmarke, laufe rückwärts zu Kopfmarken, sammle zu lesende Symbole in Registern
- **Schreiben + Kopfbewegungen:** lege Symbole und Richtungen in Register suche Kopfmarken und überschreibe Teilzelle entsprechend

**Simulation benötigt quadratischen Zeitaufwand**

→ HMU, §8.4.3

## Ausführung einer anderen TM als Zwischenschritt

- **Aufruf von  $M'$  in Überföhrungsfunktion von  $M$** 
  - $M'$  erhalt Eingabewort von  $M$  und gibt Resultat an  $M$  zuröck
  - $M$  wechselt nach Ausföhrung von  $M'$  in festen Folgezustand
  - Anwendungsbeispiel: Multiplikation als wiederholte Addition
- **Simulation wie bei Assembler-Unterprogrammen**
  - Umbenennung aller Zustande von  $M'$  zur Konfliktvermeidung
  - Erganze Zustand  $q_r$  f ur R ucksprung ins aufrufende Programm
  - Erganze separates Arbeitsband f ur Unterprogramm
  - **Aufruf:** Speichere R ucksprungadresse (Zustand von  $M$ ) in Register
  - Kopiere Eingabe f ur Unterprogramme auf Arbeitsband f ur  $M'$
  - Nach Abarbeitung kopiere Resultate auf Arbeitsband von  $M$
  - Wechsele in Zustand, der im Register gespeichert ist

## Restriktionen vereinfachen Analysen von TM

Einfachere Annahmen und weniger Alternativen in Beweisen

**Kein Verlust der Ausdruckskraft:** Simulation normaler TMs möglich

### 1. Halbseitig unendliches Band

↪ HMU, §8.5.1

- Beidseitig unendliches Band durch **Tupelalphabet**  $\Gamma^2$  simulierbar
- Im Paar  $(X_l, X_r)$  repräsentiert  $X_l$  die linke,  $X_r$  die rechte Bandhälfte
- Register (simulierbar im Zustand) gibt an, **welche Hälfte aktiv ist**

### 2. Binäres Bandalphabet $\Gamma = \{1, B\}$

- Symbole beliebiger Alphabete als Strings über  $\{1B, 11\}$  simulierbar

### 3. Zwei Stacks statt Turingband

↪ HMU, §8.5.2

- 2 Stacks + Zustand können jede Konfiguration  $(u, q, v)$  beschreiben

### 4. Zählermaschinen

↪ HMU, §8.5.3/4

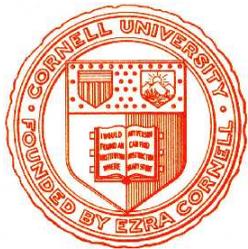
- Endliche Zahl von Registern kann **beliebig große Zahlen** verarbeiten
- Operationen: Test auf Null, Addition oder Subtraktion von Eins
- Zähler können Stacks simulieren (aufwendige Codierung von Wörtern als Zahl)

- **Computer können Turingmaschinen simulieren**
  - Repräsentiere binäres Bandalphabet und halbseitig unendliches Band
  - (Endliche) reale Speicher können nach Bedarf beliebig erweitert werden
- **Turingmaschinen können Computer simulieren**
  - Speicher wird durch einseitiges Band mit binärem Alphabet repräsentiert
  - Register enthalten Programmzähler, Speicheradressregister, etc.
  - **Aufsuchen einer Speicherzelle** vom Bandanfang durch Zählen
  - Gesuchter Speicherinhalt wird im Register abgelegt und analysiert
  - Identifizierte **Anweisungen** werden durch Unterprogramme ausgeführt
  - Nach Ausführung wird **Anweisungszähler** angepaßt und die nächste Anweisung aus dem Speicher geholt
- **Simulationsaufwand ist polynomiell** ↪ HMU, §8.6.3
  - $n$  Schritte des realen Computers benötigen maximal  $n^6$  Schritte
  - Optimierungen möglich

# Theoretische Informatik I

## Einheit 4.2

### Modelle für Typ-0 & Typ-1 Sprachen



1. Nichtdeterministische Turingmaschinen
2. Äquivalenz zu Typ-0 Sprachen
3. Linear beschränkte Automaten  
und Typ-1 Sprachen
4. Eigenschaften von  $\mathcal{L}_0/\mathcal{L}_1$ -Sprachen

- **Ableitbarkeit  $w \longrightarrow_G z$  ist nichtdeterministisch**
  - In  $w$  können verschiedene Teilworte ersetzt werden
  - Auf ein Teilwort können verschiedene Regeln angewandt werden
  - Simulation erfordert nichtdeterministisches Maschinenmodell
- **Maschinenmodelle sind i.a. deterministisch**
  - Nichtdeterministische Modelle sind “unrealistisch” und nur für elegantere Modellierung geeignet
  - Nichtdeterministische Modelle sind evtl. deterministisch simulierbar aber nur mit exponentiellem Aufwand
- **Verwende nichtdeterministische Turingmaschinen**
  - “Simultane” Behandlung vieler alternativer Konfigurationen
  - Zeige Äquivalenz zu deterministischen Turingmaschinen
  - Zeige Äquivalenz zu Typ-0 Grammatiken
  - Zeige Äquivalenz zu Typ-1 Grammatiken für eingeschränktes Modell

- Eine nichtdeterministische **Turingmaschine (NTM)** ist ein 7-Tupel  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  mit
  - $Q$  nichtleere endliche **Zustandsmenge**
  - $\Sigma$  endliches **Eingabealphabet**
  - $\Gamma \supseteq \Sigma$  endliches **Bandalphabet**
  - $\delta: Q \times \Gamma \rightarrow \mathcal{P}_e(Q \times \Gamma \times \{L, R\})$  endliche **Überföhrungsfunktion**
  - $q_0 \in Q$  **Startzustand**
  - $B \in \Gamma \setminus \Sigma$  **Leersymbol des Bands**
  - $F \subseteq Q$  Menge von **akzeptierenden (End-)Zuständen**
- **Definition von  $\vdash^*$  und  $L(M)$  analog zu DTM**
  - $(uZ, q, Xv) \vdash (u, p, ZYv), \quad \text{falls } (p, Y, L) \in \delta(q, X)$
  - $(u, q, Xv) \vdash (uY, p, v), \quad \text{falls } (p, Y, R) \in \delta(q, X)$
  - ⋮
  - $L(M) = \{w \in \Sigma^* \mid \exists p \in F. \exists u, v \in \Gamma^*. (\epsilon, q_0, w) \vdash^* (u, p, v)\}$

# JEDE NTM IST DURCH EINE DTM SIMULIERBAR

- **Verarbeite alle Alternativen sequentiell**

↳ Anhang, Folie 4

- Speichere Konfigurationen der NTM auf einem Arbeitsband
- Beginne mit Anfangskonfiguration  $\kappa_0$  der NTM
- In jedem Schritt der Simulation berechne alle Konfigurationen der NTM, die aus der aktuellen Konfiguration entstehen würden
- Verarbeite Konfigurationen in der Reihenfolge ihrer Erzeugung
- ⇒ Jede mögliche Konfiguration der NTM wird von der DTM erreicht
- DTM akzeptiert genau dann, wenn NTM akzeptiert

- **Größe der DTM wächst linear mit Größe der NTM**

- Zustandsüberführungstabelle wird durch Unterprogramme codiert
- Berechnung der Nachfolgekongfigurationen auf Hilfsband

- **Rechenzeit wächst exponentiell**

- Rechenzeit der NTM ist Länge des kürzesten akzeptierenden Pfades
- Bei  $k$  Alternativen pro Schritt muß die Simulation für  $n$  Schritte der NTM im schlimmsten Fall bis zu  $k^n$  Konfigurationen erzeugen

## Typ-0 Grammatiken und Turingmaschinen beschreiben dieselbe Klasse von Sprachen

### ● **Grammatik** $\longrightarrow$ **Turingmaschine**

$\mapsto$  Anhang, Folie 6

Turingmaschine simuliert Anwendung der Produktionsregeln

- Ableitbare Wörter werden schrittweise auf Hilfsband geschrieben
- Wörter auf dem Hilfsband werden mit der Eingabe verglichen

Maschine akzeptiert, wenn  $w \in L(G)$ , und terminiert sonst nicht

### ● **Turingmaschine** $\longrightarrow$ **Grammatik**

$\mapsto$  Anhang, Folie 8

Grammatik simuliert Konfigurationsübergänge der Turingmaschine

- Erzeuge alle möglichen Eingabewörter und Anfangskonfigurationen
- Codiere Konfigurationsübergänge von  $M$  als Regeln
- Simuliere Akzeptieren durch Löschen von Nonterminalsymbolen

Grammatik generiert genau alle Wörter, die  $M$  akzeptiert

# LINEAR BESCHRÄNKTE AUTOMATEN

**Welches Modell paßt zu Typ-1 Sprachen?**

## Welches Modell paßt zu Typ-1 Sprachen?

- **Typ-1 Sprachen werden “expansiv” erzeugt**
  - In jeder Ableitung  $S \longrightarrow w_1 \longrightarrow w_2 \dots \longrightarrow w$  eines Wortes  $w \in L(G)$  ist keines der  $w_i$  länger als  $w$  (Ausnahme  $w = \epsilon$ )
  - Turingmaschine braucht maximal  $|w|$  Bandzellen zur Simulation

## Welches Modell paßt zu Typ-1 Sprachen?

- **Typ-1 Sprachen werden “expansiv” erzeugt**
  - In jeder Ableitung  $S \longrightarrow w_1 \longrightarrow w_2 \dots \longrightarrow w$  eines Wortes  $w \in L(G)$  ist keines der  $w_i$  länger als  $w$  (Ausnahme  $w = \epsilon$ )
  - Turingmaschine braucht maximal  $|w|$  Bandzellen zur Simulation
- **Beschränke NTMs auf linearen Bandverbrauch**
  - Das Arbeitsband ist nur halbseitig unendlich
  - Anfangskonfigurationen haben die Form  $(\epsilon, q_0, w\#)$
  - $\#$  ist ein spezielles Bandende-Symbol, das niemals überlaufen oder überschrieben werden darf

## Welches Modell paßt zu Typ-1 Sprachen?

- **Typ-1 Sprachen werden “expansiv” erzeugt**
  - In jeder Ableitung  $S \longrightarrow w_1 \longrightarrow w_2 \dots \longrightarrow w$  eines Wortes  $w \in L(G)$  ist keines der  $w_i$  länger als  $w$  (Ausnahme  $w = \epsilon$ )
  - Turingmaschine braucht maximal  $|w|$  Bandzellen zur Simulation
- **Beschränke NTMs auf linearen Bandverbrauch**
  - Das Arbeitsband ist nur halbseitig unendlich
  - Anfangskonfigurationen haben die Form  $(\epsilon, q_0, w\#)$
  - $\#$  ist ein spezielles Bandende-Symbol, das niemals überlaufen oder überschrieben werden darf
- **Formal: linear beschränkter Automat (LBA)**
  - NTM  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  mit halbseitig unendlichem Band und ausgezeichnetem Symbol  $\# \in \Gamma \setminus (\Sigma \cup \{B\})$  und der Einschränkung  $\delta(q, \#) \subseteq \{(p, \#, L) \mid p \in Q\}$  für alle  $q \in Q$
  - Es gilt:  $\mathcal{L}_1 = \{L \mid L = L(M) \text{ für einen LBA } M\} \mapsto$  Anhang, Folie 10

# LINEAR BESCHRÄNKTER AUTOMAT FÜR $\{0^n 1^n 2^n \mid n \in \mathbb{N}\}$

- 1. Anfangskonfiguration ist  $(\epsilon, q_0, w\#)$**
- 2. Wenn das Band leer ist, akzeptiere die Eingabe**
- 3. Ansonsten ersetze die erste 0 durch  $B$** 
  - Wenn keine 0 unter dem Kopf steht, halte an ohne zu akzeptieren
- 4. Gehe rechts zur ersten 1; ersetze diese durch  $B$** 
  - Vor der 1 dürfen nur Nullen oder Blanks kommen (!)
  - Wenn keine 1 vorkommt, halte an ohne zu akzeptieren
- 5. Gehe rechts zur ersten 2; ersetze diese durch  $B$** 
  - Vor der 2 dürfen nur noch Einsen oder Blanks kommen (!)
  - Wenn keine 2 am Ende steht, halte an ohne zu akzeptieren
- 6. Laufe zurück zum Anfang des restlichen Wortes**
  - Fahre fort mit Schritt 2

---

## **Optimierung: Schließe Lücken durch Verschieben**

- Verfahren funktioniert analog auch für  $\{0^n 1^n 2^n 3^n 4^n \mid n \in \mathbb{N}\}$

# ZUSAMMENHÄNGE ZWISCHEN SPRACHKLASSEN

**Semi-entscheidbare Sprache:** Sprache, die von Turingmaschinen akzeptiert wird

**Entscheidbare Sprache:** Sprache, die von terminierenden Turingmaschine akzeptiert wird

**Kontextsensitive Sprache:** Sprache, die von linear beschränkten Automaten akzeptiert wird

- **Entscheidbare Sprachen sind auch semi-entscheidbar**
  - Offensichtlich, da engere Bedingung
- **Kontextsensitive Sprachen sind entscheidbar**
  - Ein LBA hat bei Eingabe  $w$  maximal  $(|\Gamma| + |Q|)^{|w|+1}$  Konfigurationen
- **Jede endliche Sprache  $L$  ist entscheidbar**
  - Bei Eingabe von  $w$  testet  $M$  durch Suchen, ob  $w \in \{w_1, \dots, w_n\} = L$  gilt
- **$L$  entscheidbar  $\Leftrightarrow L$  und  $\overline{L}$  semi-entscheidbar**
  - “ $\Rightarrow$ ”: Entscheidbare Sprachen sind abgeschlossen unter Komplement
  - “ $\Leftarrow$ ”: Simuliere Maschinen für  $L$  und  $\overline{L}$  simultan, übernehme Ergebnis  
Eine der beiden Maschinen muß terminieren und akzeptieren
- **$L$  semi-entscheidbar  $\Leftrightarrow$  es gibt ein entscheidbares  $L' \subseteq \Sigma^* \times \Sigma^*$  mit  $L = \{w \mid \exists v. (w, v) \in L'\}$  (Projektionssatz)**
  - Aufwendiger Beweis, benötigt schrittweise Simulation von Maschinen

# ABSCHLUSSEIGENSCHAFTEN SUMMARISCH

- **Alle drei Sprachklassen sind abgeschlossen unter**

- Vereinigung

$$L_1 \cup L_2$$

- Durchschnitt

$$L_1 \cap L_2$$

- Spiegelung

$$L^R$$

- Verkettung

$$L_1 \circ L_2$$

- Hüllenbildung

$$L^*$$

- Homomorphismen

$$h(L)$$

- Inverse Homomorphismen

$$h^{-1}(L)$$

- Urbild berechenbarer Funktionen

$$f^{-1}(L)$$

- **Typ-1 und entscheidbare Sprachen zusätzlich**

- Komplement

$$\overline{L}$$

- Differenz

$$L_1 - L_2$$

- Aufzählbare Sprachen: Bild berechenbarer Funktionen

$$f(L)$$

Beweise im Anhang, ab Folie 11

# PRÜFEN VON EIGENSCHAFTEN SUMMARISCH

- **“ $x \in L$ ” kann automatisch geprüft werden für**
  - Kontextsensitive und entscheidbare Sprachen  
(Folgt unmittelbar aus der Definition von Entscheidbarkeit)
  - Aber **nicht für aufzählbare Sprachen**  
(Folgt aus Existenz einer aufzählbaren, aber unentscheidbaren Sprache)
- **Für keine Sprachklasse kann getestet werden ob**
  - eine Sprache  $L$  der Klasse **leer** ist
  - zwei Sprachen  $L_1$  und  $L_2$  der Klasse **gleich** sind
  - zwei Sprachen  $L_1$  und  $L_2$  der Klasse **ineinander enthalten** sind
  - der **Durchschnitt** zweier Sprachen der Klasse **leer** ist

Beweise benötigen Beispiele für Sprachen, die nicht zur Klasse gehören

- **Entscheidbare, nicht kontextsensitive Sprache**
  - Menge aller **äquivalenten regulären Ausdrücke** (gelesen als Text), wenn diese eine Iteration  $E^k = \underbrace{E \circ E \dots \circ E}_{k\text{-mal}}$  enthalten dürfen
  - Äquivalenztest benötigt exponentiell großen Speicherplatz

- **Entscheidbare, nicht kontextsensitive Sprache**

- Menge aller **äquivalenten regulären Ausdrücke** (gelesen als Text), wenn diese eine Iteration  $E^k = \underbrace{E \circ E \dots \circ E}_{k\text{-mal}}$  enthalten dürfen
- Äquivalenztest benötigt exponentiell großen Speicherplatz

- **Aufzählbare, nicht entscheidbare Sprache**

- **Selbstanwendbarkeitsproblem:** Menge aller Programme von Turingmaschinen, die bei Eingabe des eigenen Programms als Text terminieren

- **Entscheidbare, nicht kontextsensitive Sprache**

- Menge aller **äquivalenten regulären Ausdrücke** (gelesen als Text), wenn diese eine Iteration  $E^k = \underbrace{E \circ E \dots \circ E}_{k\text{-mal}}$  enthalten dürfen
- Äquivalenztest benötigt exponentiell großen Speicherplatz

- **Aufzählbare, nicht entscheidbare Sprache**

- **Selbstanwendbarkeitsproblem**: Menge aller Programme von Turingmaschinen, die bei Eingabe des eigenen Programms als Text terminieren

- **Nicht aufzählbare Sprache**

- **Totale Berechenbarkeit**: Menge aller Programme von Turingmaschinen, die bei jeder Eingabe terminieren

# GRENZEN DER SPRACHKLASSEN

- **Entscheidbare, nicht kontextsensitive Sprache**

- Menge aller **äquivalenten regulären Ausdrücke** (gelesen als Text), wenn diese eine Iteration  $E^k = \underbrace{E \circ E \dots \circ E}_{k\text{-mal}}$  enthalten dürfen
- Äquivalenztest benötigt exponentiell großen Speicherplatz

- **Aufzählbare, nicht entscheidbare Sprache**

- **Selbstanwendbarkeitsproblem**: Menge aller Programme von Turingmaschinen, die bei Eingabe des eigenen Programms als Text terminieren

- **Nicht aufzählbare Sprache**

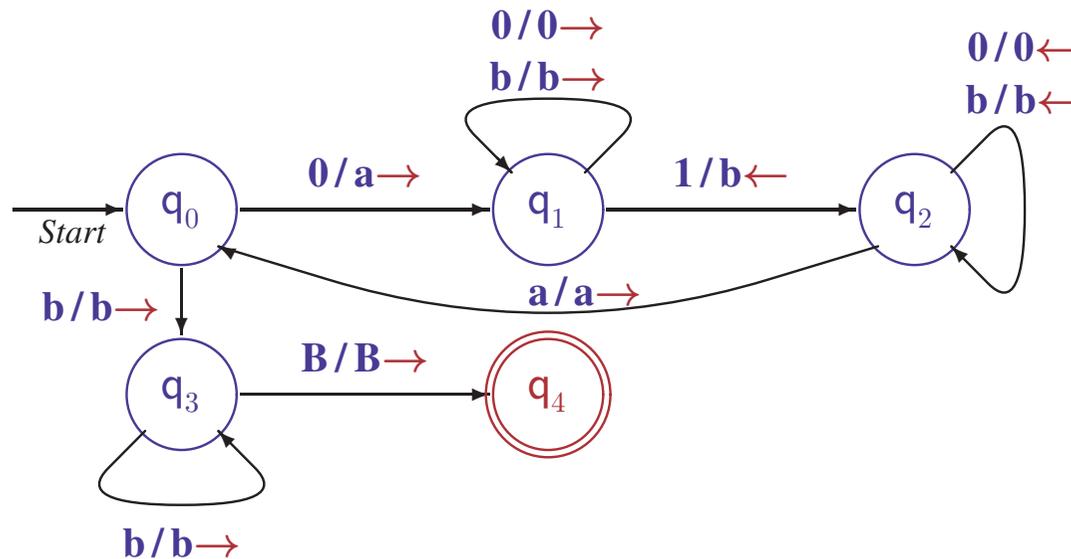
- **Totale Berechenbarkeit**: Menge aller Programme von Turingmaschinen, die bei jeder Eingabe terminieren

**Mehr dazu in Theoretischer Informatik II**

- **Turingmaschine als allgemeinstes Maschinenmodell**
  - Deterministischer endlicher Automat mit unendlichem Speicherband
  - Gleiche Ausdruckskraft wie reale Computer (aber einfacher strukturiert)
  - Nichtdeterministische Variante mit exponentiellem Aufwand simulierbar
  - Äquivalent zu Typ-0 Grammatiken
  - Bei linearer Bandbeschränkung äquivalent zu Typ-1 Grammatiken
  - **Entscheidbare Sprachen** stehen zwischen  $\mathcal{L}_0$  und  $\mathcal{L}_1$
- **Wichtige Eigenschaften der Sprachklassen**
  - Abgeschlossen unter  $\cup, \cap, R, \circ, *, h, h^{-1}$
  - $\mathcal{L}_1$  und entscheidbare Sprachen zusätzlich unter  $\bar{\phantom{x}}, -$
  - **Viele Eigenschaften können nicht automatisch getestet werden**
    - Fast alle nichtrivialen Eigenschaften sind für keine Klasse entscheidbar
    - Für  $\mathcal{L}_0$  ist selbst das Wortproblem nicht mehr entscheidbar

ANHANG

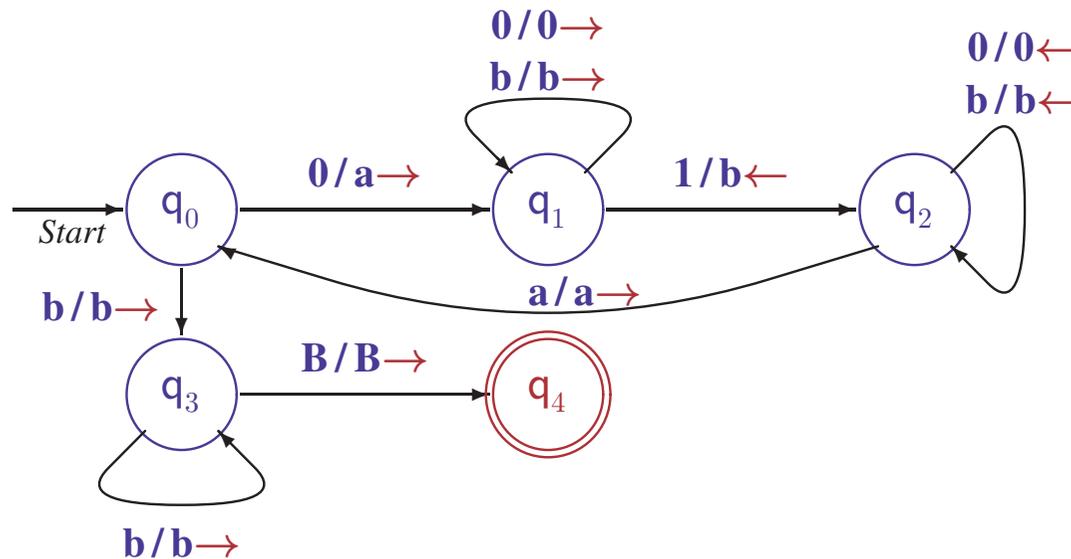
# ERKANNTE SPRACHE EINER TURINGMASCHINE



- **Analyse:  $M$  zählt Nullen und Einsen gleichzeitig**

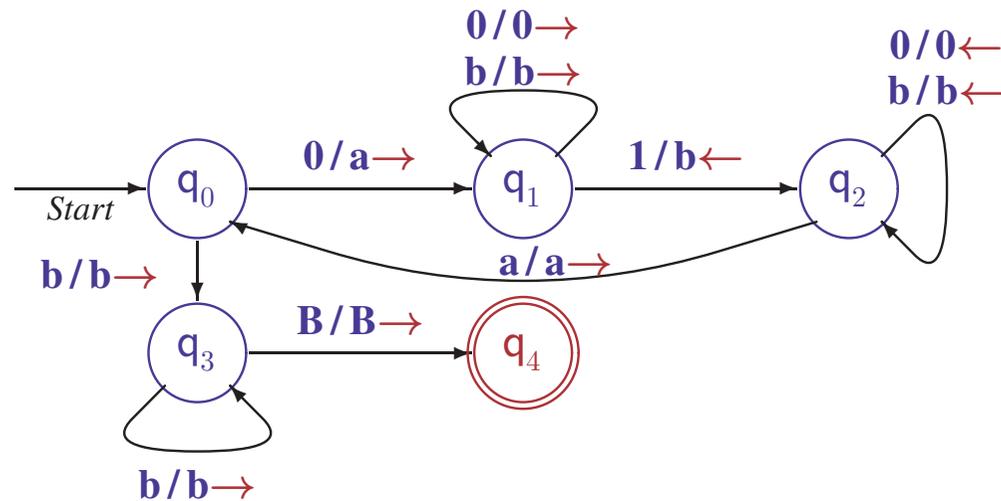
- Umwandeln einer 0 in  $a$  triggert Umwandeln einer 1 in  $b$
- Maschine stoppt in  $q_0$ , wenn keine Nullen oder Einsen vorhanden sind
- Maschine stoppt in  $q_1$ , wenn zuwenig Einsen vorhanden sind
- Maschine stoppt in  $q_3$ , wenn zuwenig Nullen vorhanden sind
- Maschine akzeptiert in  $q_4$ , wenn Anzahl der Nullen und Einsen gleich

# ERKANNTE SPRACHE EINER TURINGMASCHINE



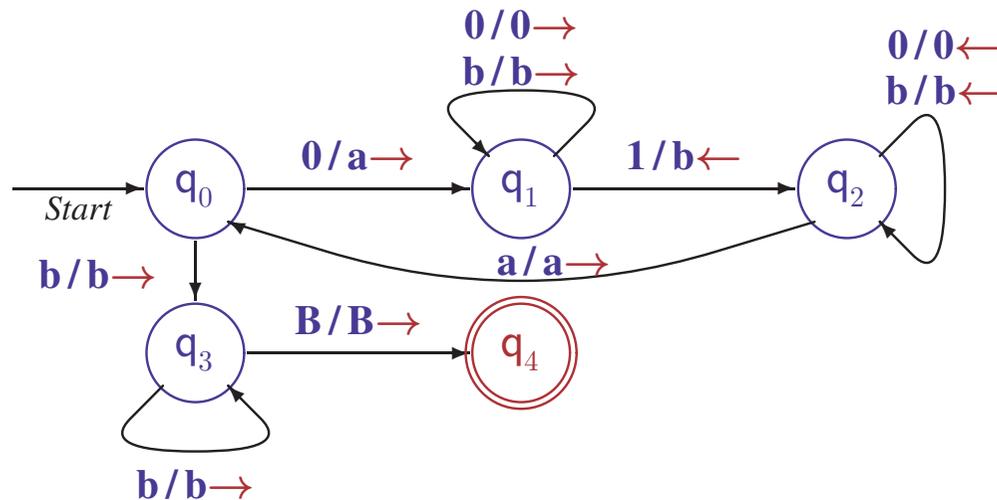
- **Analyse:**  $M$  zählt Nullen und Einsen gleichzeitig
  - Umwandeln einer 0 in  $a$  triggert Umwandeln einer 1 in  $b$
  - Maschine stoppt in  $q_0$ , wenn keine Nullen oder Einsen vorhanden sind
  - Maschine stoppt in  $q_1$ , wenn zuwenig Einsen vorhanden sind
  - Maschine stoppt in  $q_3$ , wenn zuwenig Nullen vorhanden sind
  - Maschine akzeptiert in  $q_4$ , wenn Anzahl der Nullen und Einsen gleich
- **Zeige:**  $L(M) = \{0^n 1^n \mid n \geq 1\}$ 
  - $(\epsilon, q_0, w) \vdash^* (u, q_4, v)$  genau dann, wenn  $w = 0^n 1^n$  für ein  $n \geq 1$

# NACHWEIS DER ERKANNTEN SPRACHE I



$(\epsilon, q_0, w) \vdash^* (u, q_4, v)$  wenn  $w = 0^n 1^n$  für ein  $n \geq 1$

# NACHWEIS DER ERKANNTEN SPRACHE I

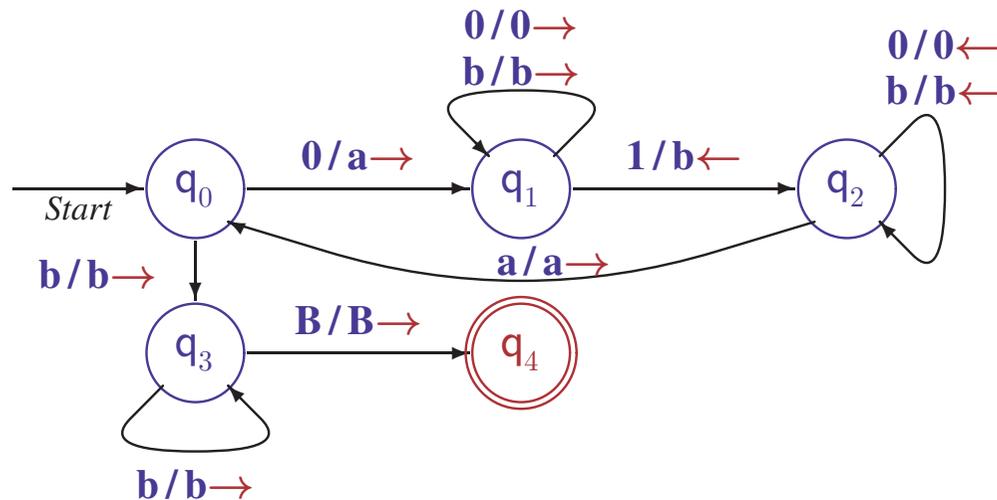


$(\epsilon, q_0, w) \vdash^* (u, q_4, v)$  wenn  $w = 0^n 1^n$  für ein  $n \geq 1$

Für alle  $u, v \in \Gamma^*$ ,  $w \in \{0, b\}^*$ ,  $k \in \mathbb{N}$ ,  $n \geq 1$  gilt

1.  $(u, q_0, 0v) \vdash (ua, q_1, v)$  *(direkt aus Diagramm ersichtlich)*
2.  $(u, q_0, 0wv) \vdash^* (uaw, q_1, v)$  *(Induktion über  $w$ )*

# NACHWEIS DER ERKANNTEN SPRACHE I

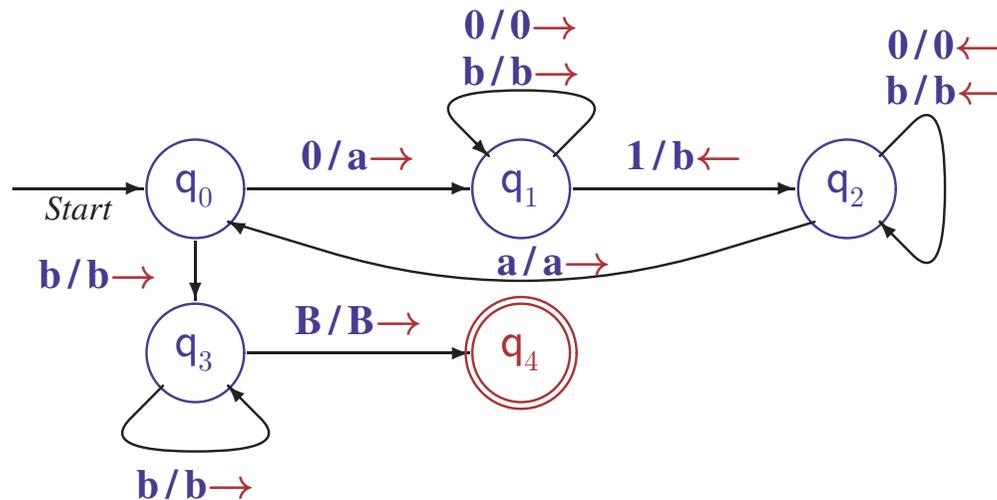


$(\epsilon, q_0, w) \vdash^* (u, q_4, v)$  wenn  $w = 0^n 1^n$  für ein  $n \geq 1$

Für alle  $u, v \in \Gamma^*$ ,  $w \in \{0, b\}^*$ ,  $k \in \mathbb{N}$ ,  $n \geq 1$  gilt

1.  $(u, q_0, 0v) \vdash (ua, q_1, v)$  *(direkt aus Diagramm ersichtlich)*
2.  $(u, q_0, 0wv) \vdash^* (uaw, q_1, v)$  *(Induktion über w)*
3.  $(u, q_0, 0w1v) \vdash^* (u, q_2, awbv)$  *(folgt aus 2. und Diagramm)*
4.  $(u, q_0, 0w1v) \vdash^* (ua, q_0, wbv)$  *(folgt aus 3. und Diagramm)*

# NACHWEIS DER ERKANNTEN SPRACHE I

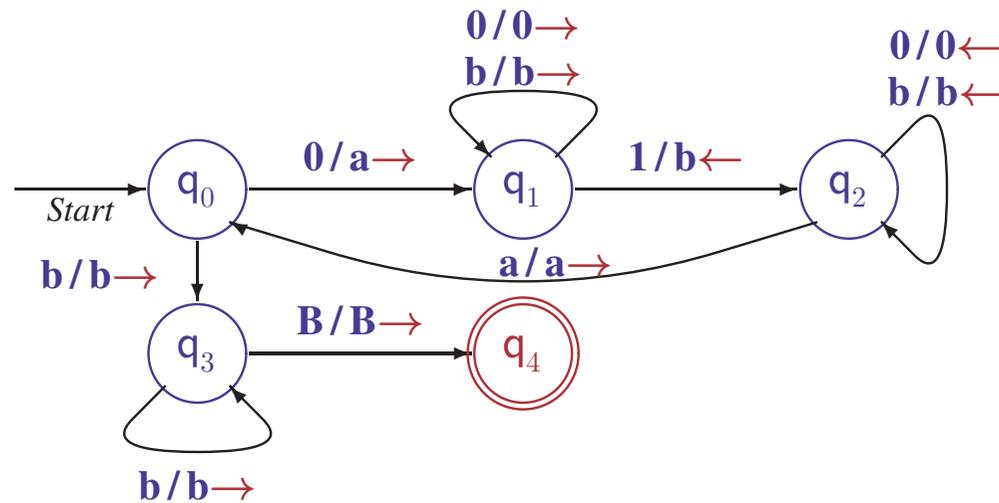


$(\epsilon, q_0, w) \vdash^* (u, q_4, v)$  wenn  $w = 0^n 1^n$  für ein  $n \geq 1$

Für alle  $u, v \in \Gamma^*$ ,  $w \in \{0, b\}^*$ ,  $k \in \mathbb{N}$ ,  $n \geq 1$  gilt

1.  $(u, q_0, 0v) \vdash (ua, q_1, v)$  *(direkt aus Diagramm ersichtlich)*
2.  $(u, q_0, 0wv) \vdash^* (uaw, q_1, v)$  *(Induktion über  $w$ )*
3.  $(u, q_0, 0w1v) \vdash^* (u, q_2, awbv)$  *(folgt aus 2. und Diagramm)*
4.  $(u, q_0, 0w1v) \vdash^* (ua, q_0, wbv)$  *(folgt aus 3. und Diagramm)*
5.  $(\epsilon, q_0, 0^k w 1^k v) \vdash^* (a^k, q_0, w b^k v)$  *(Induktion über  $k$  unter Verwendung von 4.)*
6.  $(\epsilon, q_0, 0^n 1^n v) \vdash^* (a^n b^n, q_3, v)$  *(5. mit  $w = \epsilon$ , Diagramm, Induktion in  $q_3$ )*
7.  $(\epsilon, q_0, 0^n 1^n) \vdash^* (a^n b^n, q_3, B)$  *(6. mit  $v = \epsilon$ )*
8.  $(\epsilon, q_0, 0^n 1^n) \vdash^* (a^n b^n B, q_4, B)$  *(7. und Diagramm)*

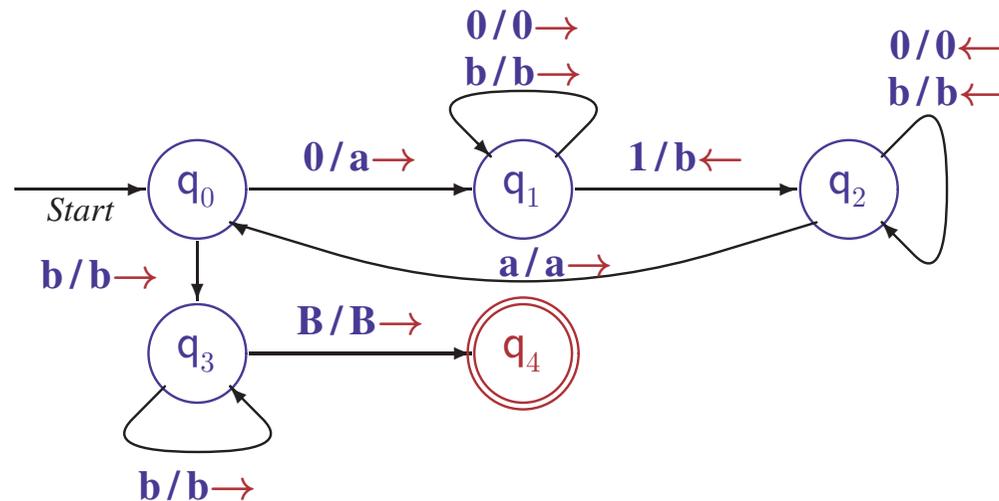
# NACHWEIS DER ERKANNTEN SPRACHE II



$(\epsilon, q_0, w) \vdash^* (u, q_4, v)$  **nur wenn**  $w = 0^n 1^n$  **für ein**  $n \geq 1$

Informales Argument, da detaillierter formaler Beweis zu aufwendig

# NACHWEIS DER ERKANNTEN SPRACHE II

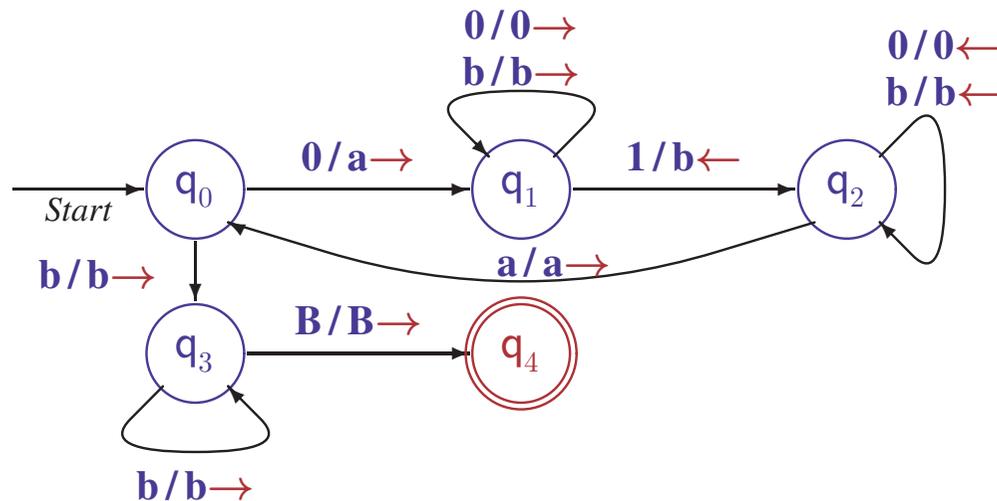


$(\epsilon, q_0, w) \vdash^* (u, q_4, v)$  **nur wenn**  $w = 0^n 1^n$  **für ein**  $n \geq 1$

Informales Argument, da detaillierter formaler Beweis zu aufwendig

1. Die Schleife  $q_0 - q_1 - q_2$  wandelt je eine 0 in ein  $a$  und eine 1 in  $b$  um
2. Am Ende der Schleife stehen alle 0 und 1 rechts vom Kopf und alle  $a$  links davon
3. Da das Eingabewort zu  $\{0, 1\}^*$  gehört, stehen in  $q_0$  gleich viele  $a$  und  $b$  auf dem Band

# NACHWEIS DER ERKANNTEN SPRACHE II



$(\epsilon, q_0, w) \vdash^* (u, q_4, v)$  **nur wenn**  $w = 0^n 1^n$  **für ein**  $n \geq 1$

Informales Argument, da detaillierter formaler Beweis zu aufwendig

1. Die Schleife  $q_0 - q_1 - q_2$  wandelt je eine 0 in ein  $a$  und eine 1 in  $b$  um
2. Am Ende der Schleife stehen alle 0 und 1 rechts vom Kopf und alle  $a$  links davon
3. Da das Eingabewort zu  $\{0, 1\}^*$  gehört, stehen in  $q_0$  gleich viele  $a$  und  $b$  auf dem Band
4. Um  $q_4$  von  $q_0$  zu erreichen, muß das Wort rechts vom Kopf die Form  $b^n$  ( $n \geq 1$ ) haben
5. Wegen 2. und 3. hat das Wort links vom Kopf die Form  $a^n$
6. Wegen 1. muß das Eingabewort die Form  $0^n 1^n$  haben, um akzeptiert zu werden.

JEDE NTM IST DURCH EINE DTM SIMULIERBAR

**Verarbeite alle Alternativen sequentiell**

# JEDE NTM IST DURCH EINE DTM SIMULIERBAR

## Verarbeite alle Alternativen sequentiell

- Für  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  definiere Mengen  $K_i^w$ 
  - Menge der in  $i$  Schritten erzeugbaren Konfigurationen bei Eingabe  $w$   
 $K_0^w := \{(\epsilon, q_0, w)\}$ ,  $K_{i+1}^w := \{\kappa' \mid \exists \kappa \in K_i^w. \kappa \vdash \kappa'\}$
  - Es gilt  $w \in L(M) \Leftrightarrow \exists i. \exists p \in F. \exists u, v \in \Gamma^*. (u, p, v) \in K_i^w$

# JEDE NTM IST DURCH EINE DTM SIMULIERBAR

## Verarbeite alle Alternativen sequentiell

- Für  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  definiere Mengen  $K_i^w$ 
  - Menge der in  $i$  Schritten erzeugbaren Konfigurationen bei Eingabe  $w$   
 $K_0^w := \{(\epsilon, q_0, w)\}$ ,  $K_{i+1}^w := \{\kappa' \mid \exists \kappa \in K_i^w. \kappa \vdash \kappa'\}$
  - Es gilt  $w \in L(M) \Leftrightarrow \exists i. \exists p \in F. \exists u, v \in \Gamma^*. (u, p, v) \in K_i^w$
- Beschreibe DTM zur Erzeugung der  $K_i^w$   
 $K_0^w$

$q_0$	$w_1 \dots w_n$	\$																	
-------	-----------------	----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

- Arbeitsband beschreibt alle bisher erzeugten Konfigurationen der NTM  
Die aktuell betrachtete Konfiguration  $\kappa$  wird markiert

# JEDE NTM IST DURCH EINE DTM SIMULIERBAR

## Verarbeite alle Alternativen sequentiell

- Für  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  definiere Mengen  $K_i^w$ 
  - Menge der in  $i$  Schritten erzeugbaren Konfigurationen bei Eingabe  $w$

$$K_0^w := \{(\epsilon, q_0, w)\}, \quad K_{i+1}^w := \{\kappa' \mid \exists \kappa \in K_i^w. \kappa \vdash \kappa'\}$$

- Es gilt  $w \in L(M) \Leftrightarrow \exists i. \exists p \in F. \exists u, v \in \Gamma^*. (u, p, v) \in K_i^w$

- Beschreibe DTM zur Erzeugung der  $K_i^w$

$K_0^w$

$K_1^w$

$q_0$	$w_1 \dots w_n$	\$	#	$u_1^1   q_1^1   v_1^1$	#	$u_1^2   q_1^2   v_1^2$	#	$\dots$	#	$u_1^{j_1}   q_1^{j_1}   v_1^{j_1}$	\$					
-------	-----------------	----	---	-------------------------	---	-------------------------	---	---------	---	-------------------------------------	----	--	--	--	--	--

- Arbeitsband beschreibt alle bisher erzeugten Konfigurationen der NTM
- Die aktuell betrachtete Konfiguration  $\kappa$  wird markiert
- **Lesen:** Extrahiere aus  $\kappa$  das gelesene Symbol  $X$  und Zustand  $q$  der NTM
- **Verarbeiten:** Erzeuge aus  $\kappa$  und  $\delta(q, X)$  alle Nachfolgekongfigurationen
- Lösche Markierung von  $\kappa$  und markiere nächste Konfiguration auf Band

# JEDE NTM IST DURCH EINE DTM SIMULIERBAR

## Verarbeite alle Alternativen sequentiell

- Für  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  definiere Mengen  $K_i^w$ 
  - Menge der in  $i$  Schritten erzeugbaren Konfigurationen bei Eingabe  $w$ 

$$K_0^w := \{(\epsilon, q_0, w)\}, \quad K_{i+1}^w := \{\kappa' \mid \exists \kappa \in K_i^w. \kappa \vdash \kappa'\}$$
  - Es gilt  $w \in L(M) \Leftrightarrow \exists i. \exists p \in F. \exists u, v \in \Gamma^*. (u, p, v) \in K_i^w$

- Beschreibe DTM zur Erzeugung der  $K_i^w$

$K_0^w$			$K_1^w$					$K_2^w$											
$q_0$	$w_1 \dots w_n$	\$	#	$u_1^1   q_1^1   v_1^1$	#	$u_1^2   q_1^2   v_1^2$	#	...	#	$u_1^{j_1}   q_1^{j_1}   v_1^{j_1}$	\$	#	$u_2^1   q_1^1   v_2^1$	#	...	#	$u_2^{j_2}   q_2^{j_2}   v_2^{j_2}$	\$	...

- Arbeitsband beschreibt alle bisher erzeugten Konfigurationen der NTM
- Die aktuell betrachtete Konfiguration  $\kappa$  wird markiert
- **Lesen:** Extrahiere aus  $\kappa$  das gelesene Symbol  $X$  und Zustand  $q$  der NTM
- **Verarbeiten:** Erzeuge aus  $\kappa$  und  $\delta(q, X)$  alle Nachfolgekonfigurationen
- Lösche Markierung von  $\kappa$  und markiere nächste Konfiguration auf Band
- Jede mögliche Konfiguration der NTM wird von der DTM aufgesucht

# SIMULATION VON NTMS IST EXPONENTIELL

- **Größe der DTM wächst linear mit Größe der NTM**
  - Zustandsüberführungstabelle wird durch Unterprogramme codiert
  - Berechnung der Nachfolgekonfigurationen auf einem Hilfsband

- **Größe der DTM wächst linear mit Größe der NTM**
  - Zustandsüberführungstabelle wird durch Unterprogramme codiert
  - Berechnung der Nachfolgekonfigurationen auf einem Hilfsband
- **Rechenzeit wächst exponentiell**
  - Einzelschritte linear in Größe einer NTM Konfiguration simulierbar
    - Bestimmen einer Nachfolgekonfiguration ist “konstant”
    - Schreiben der Nachfolgekonfiguration linear (zwei Arbeitsbänder)

# SIMULATION VON NTMS IST EXPONENTIELL

- **Größe der DTM wächst linear mit Größe der NTM**
  - Zustandsüberführungstabelle wird durch Unterprogramme codiert
  - Berechnung der Nachfolgekonfigurationen auf einem Hilfsband
- **Rechenzeit wächst exponentiell**
  - Einzelschritte linear in Größe einer NTM Konfiguration simulierbar
    - Bestimmen einer Nachfolgekonfiguration ist “konstant”
    - Schreiben der Nachfolgekonfiguration linear (zwei Arbeitsbänder)

Aber ...

- **Rechenzeit** der NTM ist **Länge des kürzesten akzeptierenden Pfades**
- Bei  $k$  Alternativen pro Schritt muß die Simulation für  $n$  Schritte der NTM im schlimmsten Fall **bis zu  $k^n$  Konfigurationen** erzeugen

SATZ:  $L \in \mathcal{L}_0 \Rightarrow L$  SEMI-ENTSCHEIDBAR

Zu jeder Grammatik  $G = (V, T, P, S)$  kann eine NTM  $M$  konstruiert werden mit  $L(G) = L(M)$

1. **Schreibe die Eingabe  $w$  auf Hilfsband 1**
2. **Schreibe das Startsymbol  $S$  auf Hilfsband 2**
3. **Simuliere eine Regelanwendung in  $G$** 
  - Wähle nichtdeterministisch ein Teilwort  $u$  des Wortes auf Band 2
  - Wähle nichtdeterministisch eine Regel der Form  $u \rightarrow v$  aus  $P$
  - Verschiebe Symbole, die rechts von  $u$  stehen, um  $|v| - |u|$  Stellen
  - Ersetze  $u$  durch  $v$
4. **Vergleiche  $w$  mit dem Wort auf Hilfsband 2**
  - Akzeptiere  $w$ , wenn die Worte gleich sind
  - Ansonsten fahre fort mit 3.

- **$M$  simuliert Ableitbarkeit in  $G$**

- Nach  $i$  Schritten steht auf Band 2 ein Wort  $w_i$  mit  $S \xrightarrow{i}_G w_i$
- Wenn  $M$  das Wort  $w$  nach  $i$  Schritten akzeptiert, dann gilt  $w = w_i$

- **$M$  simuliert Ableitbarkeit in  $G$**

- Nach  $i$  Schritten steht auf Band 2 ein Wort  $w_i$  mit  $S \xrightarrow{i}_G w_i$
- Wenn  $M$  das Wort  $w$  nach  $i$  Schritten akzeptiert, dann gilt  $w = w_i$

- **$M$  akzeptiert  $L(G)$**

- Es gilt  $w \in L(G) \Leftrightarrow \exists i. S \xrightarrow{i}_G w$
- Wenn  $M$  das Wort  $w$  nach  $i$  Schritten akzeptiert, dann gilt  $S \xrightarrow{i}_G w$ , also  $w \in L(G)$
- Wenn  $S \xrightarrow{i}_G w$  gilt, dann kann  $M$  in  $i$  Schritten das Wort  $w$  auf Band 2 erzeugen und akzeptieren, also  $w \in L(M)$

- **$M$  simuliert Ableitbarkeit in  $G$**

- Nach  $i$  Schritten steht auf Band 2 ein Wort  $w_i$  mit  $S \xrightarrow{i}_G w_i$
- Wenn  $M$  das Wort  $w$  nach  $i$  Schritten akzeptiert, dann gilt  $w = w_i$

- **$M$  akzeptiert  $L(G)$**

- Es gilt  $w \in L(G) \Leftrightarrow \exists i. S \xrightarrow{i}_G w$
- Wenn  $M$  das Wort  $w$  nach  $i$  Schritten akzeptiert, dann gilt  $S \xrightarrow{i}_G w$ , also  $w \in L(G)$
- Wenn  $S \xrightarrow{i}_G w$  gilt, dann kann  $M$  in  $i$  Schritten das Wort  $w$  auf Band 2 erzeugen und akzeptieren, also  $w \in L(M)$

- **$M$  terminiert nicht immer**

- Für  $w \notin L(G)$  gilt  $w \neq w_i$  für alle  $i$

SATZ:  $L$  SEMI-ENTSCHEIDBAR  $\Rightarrow L \in \mathcal{L}_0$

## Simuliere Abarbeitung der Turingmaschine

- **Idee: Generiere alle Konfigurationen von  $M$** 
  - Konfigurationen  $(u, q, v)$  werden als Wörter  $uqv$  codiert
  - Begrenzer  $\#$  trennt Eingabe  $w$  von Konfigurationen
  - Verarbeitung von  $w$  simuliert durch Wörter der Form  $w \# uqv \#$
- **Beschreibe Konfigurationsübergänge durch Regeln**
  - Regeln simulieren Vorschriften für Erzeugung von  $\vdash$  aus  $\delta$
- **Lege  $w$  frei, wenn  $M$  akzeptiert hat**
  - Entferne Wort nach  $\#$ , wenn  $M$  einen Endzustand erreicht
- **Grammatik erzeugt von  $M$  akzeptierte Sprache**
  - $L(G) = \{w \in \Sigma^* \mid \exists p \in F. \exists u, v \in \Gamma^*. (\epsilon, q_0, w) \vdash^* (u, p, v)\} = L(M)$

- **Erzeugung von Anfangskonfigurationen**
  - Regeln zur Erzeugung aller Wörter der Form  $w \# q_0 w \#$

# REGELN DER GRAMMATIK $G$

- **Erzeugung von Anfangskonfigurationen**

- Regeln zur Erzeugung aller Wörter der Form  $w \# q_0 w \#$

- **Simulation der Konfigurationsübergänge**

- Regeln der Form  $q X V \mapsto Y p V$  für  $V \in \Gamma$ ,  $\delta(q, X) = (p, Y, R)$

- Regeln der Form  $q X \# \mapsto Y p B \#$  für  $\delta(q, X) = (p, Y, R)$

- Regeln der Form  $Z q X \mapsto p Z Y$  für  $Z \in \Gamma$ ,  $\delta(q, X) = (p, Y, L)$

- Regeln der Form  $\# q X \mapsto \# p B Y$  für  $\delta(q, X) = (p, Y, L)$

# REGELN DER GRAMMATIK $G$

- **Erzeugung von Anfangskonfigurationen**

- Regeln zur Erzeugung aller Wörter der Form  $w \# q_0 w \#$

- **Simulation der Konfigurationsübergänge**

- Regeln der Form  $q X V \mapsto Y p V$  für  $V \in \Gamma$ ,  $\delta(q, X) = (p, Y, R)$

- Regeln der Form  $q X \# \mapsto Y p B \#$  für  $\delta(q, X) = (p, Y, R)$

- Regeln der Form  $Z q X \mapsto p Z Y$  für  $Z \in \Gamma$ ,  $\delta(q, X) = (p, Y, L)$

- Regeln der Form  $\# q X \mapsto \# p B Y$  für  $\delta(q, X) = (p, Y, L)$

- **Schlußregeln für Endzustände**

- Regeln der Form  $Z q \mapsto q$  für  $Z \in \Gamma$ ,  $q \in F$

- Regeln der Form  $q Z \mapsto q$  für  $Z \in \Gamma$ ,  $q \in F$

- Regeln der Form  $\# q \# \mapsto \epsilon$  für  $q \in F$

# REGELN DER GRAMMATIK $G$

- **Erzeugung von Anfangskonfigurationen**

- Regeln zur Erzeugung aller Wörter der Form  $w \# q_0 w \#$

- **Simulation der Konfigurationsübergänge**

- Regeln der Form  $q X V \mapsto Y p V$  für  $V \in \Gamma$ ,  $\delta(q, X) = (p, Y, R)$

- Regeln der Form  $q X \# \mapsto Y p B \#$  für  $\delta(q, X) = (p, Y, R)$

- Regeln der Form  $Z q X \mapsto p Z Y$  für  $Z \in \Gamma$ ,  $\delta(q, X) = (p, Y, L)$

- Regeln der Form  $\# q X \mapsto \# p B Y$  für  $\delta(q, X) = (p, Y, L)$

- **Schlußregeln für Endzustände**

- Regeln der Form  $Z q \mapsto q$  für  $Z \in \Gamma$ ,  $q \in F$

- Regeln der Form  $q Z \mapsto q$  für  $Z \in \Gamma$ ,  $q \in F$

- Regeln der Form  $\# q \# \mapsto \epsilon$  für  $q \in F$

**Detailbeweise z.B. in Erk-Priese, Seite 199–201**

# LBAs SIND MASCHINENMODELL FÜR TYP-1 SPRACHEN

$$\mathcal{L}_1 = \{L \mid L = L(M) \text{ für einen LBA } M\}$$

- **Beweise für  $\mathcal{L}_0$  können modifiziert werden**

# LBAs SIND MASCHINENMODELL FÜR TYP-1 SPRACHEN

$$\mathcal{L}_1 = \{ L \mid L = L(M) \text{ für einen LBA } M \}$$

- **Beweise für  $\mathcal{L}_0$  können modifiziert werden**

- **Typ-1 Grammatik  $\longrightarrow$  LBA**

Turingmaschine simuliert Anwendung der Produktionsregeln

- Ableitbare Wörter werden schrittweise erzeugt und mit  $w$  verglichen
- Beschränkung der Simulation auf Regelanwendungen, die Wörter mit maximaler Länge  $|w|$  erzeugen

**Maschine ist linear beschränkter Automat**

- Linkes und rechtes Ende des Bandes wird niemals überschritten
- Lineare Simulation der Hilfsbänder mit größerem Bandalphabet

# LBAs SIND MASCHINENMODELL FÜR TYP-1 SPRACHEN

$$\mathcal{L}_1 = \{L \mid L = L(M) \text{ für einen LBA } M \}$$

- **Beweise für  $\mathcal{L}_0$  können modifiziert werden**

- **Typ-1 Grammatik  $\longrightarrow$  LBA**

Turingmaschine simuliert Anwendung der Produktionsregeln

- Ableitbare Wörter werden schrittweise erzeugt und mit  $w$  verglichen
- Beschränkung der Simulation auf Regelanwendungen, die Wörter mit maximaler Länge  $|w|$  erzeugen

**Maschine ist linear beschränkter Automat**

- Linkes und rechtes Ende des Bandes wird niemals überschritten
- Lineare Simulation der Hilfsbänder mit größerem Bandalphabet

- **LBA  $\longrightarrow$  Typ-1 Grammatik**

- LBA muß bei Eingabe  $w$  das Band nicht mehr erweitern
- Simuliere Verarbeitung der Eingabe  $w$  mit Wörtern der Form  $(w_1, u_1)..(w_i, u_i)(w_{i+1}, q)(w_{i+1}, v_1)..(w_n, v_j)$  statt  $w \# uqv \#$
- Kürzende **Grammatikregeln können jetzt expansiv formuliert** werden

## Beweisführung mit Grammatiken

- **Vereinigung  $L_1 \cup L_2$** 
  - Sei  $L_i = L(G_i)$ , wobei  $G_i = (V_i, T_i, P_i, S_i)$  disjunkt
  - Wähle  $G = (V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}, S)$
  - Die Eigenschaften der  $G_i$  bleiben erhalten und es gilt  $L(G) = L_1 \cup L_2$

## Beweisführung mit Grammatiken

- **Vereinigung  $L_1 \cup L_2$**

- Sei  $L_i = L(G_i)$ , wobei  $G_i = (V_i, T_i, P_i, S_i)$  disjunkt
- Wähle  $G = (V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}, S)$
- Die Eigenschaften der  $G_i$  bleiben erhalten und es gilt  $L(G) = L_1 \cup L_2$

- **Spiegelung  $L^R$**

- Bilde **Spiegelgrammatik** zu  $G = (V, T, P, S)$  mit  $L = L(G)$ 
  - Setze  $G_R = (V, T, P_R, S)$  mit  $P_R = \{l^R \rightarrow \alpha^R \mid l \rightarrow \alpha \in P\}$
- Die Eigenschaften von  $G$  bleiben erhalten und es gilt  $L(G_R) = L^R$

## Beweisführung mit Grammatiken

- **Vereinigung  $L_1 \cup L_2$**

- Sei  $L_i = L(G_i)$ , wobei  $G_i = (V_i, T_i, P_i, S_i)$  disjunkt
- Wähle  $G = (V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}, S)$
- Die Eigenschaften der  $G_i$  bleiben erhalten und es gilt  $L(G) = L_1 \cup L_2$

- **Spiegelung  $L^R$**

- Bilde **Spiegelgrammatik** zu  $G = (V, T, P, S)$  mit  $L = L(G)$ 
  - Setze  $G_R = (V, T, P_R, S)$  mit  $P_R = \{l^R \rightarrow \alpha^R \mid l \rightarrow \alpha \in P\}$
- Die Eigenschaften von  $G$  bleiben erhalten und es gilt  $L(G_R) = L^R$

- **Analoge Beweise für  $L_1 \circ L_2, L^*, h(L)$**

- Verzweige aus Startsymbol oder modifiziere rechte Seite der Regeln

## Beweisführung mit Grammatiken

- **Vereinigung  $L_1 \cup L_2$**

- Sei  $L_i = L(G_i)$ , wobei  $G_i = (V_i, T_i, P_i, S_i)$  disjunkt
- Wähle  $G = (V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}, S)$
- Die Eigenschaften der  $G_i$  bleiben erhalten und es gilt  $L(G) = L_1 \cup L_2$

- **Spiegelung  $L^R$**

- Bilde **Spiegelgrammatik** zu  $G = (V, T, P, S)$  mit  $L = L(G)$ 
  - Setze  $G_R = (V, T, P_R, S)$  mit  $P_R = \{l^R \rightarrow \alpha^R \mid l \rightarrow \alpha \in P\}$
- Die Eigenschaften von  $G$  bleiben erhalten und es gilt  $L(G_R) = L^R$

- **Analoge Beweise für  $L_1 \circ L_2, L^*, h(L)$**

- Verzweige aus Startsymbol oder modifiziere rechte Seite der Regeln

- **Grammatiken helfen wenig bei Entscheidbarkeit**

- Beweisführung mit Turingmaschinen ist sinnvoller

# NACHWEIS DER ABSCHLUSSEIGENSCHAFTEN

- **Vereinigung  $L_1 \cup L_2$**

- Sei  $L_i = L(M_i)$ , wobei  $M_i = (Q_i, \Sigma_i, \Gamma_i, \delta_i, q_{0,i}, B, F_i)$  disjunkt
- Bei Eingabe eines Wortes  $w$  kopiert  $M$  das Wort auf zwei Hilfsbänder und simuliert  $M_1$  und  $M_2$  auf den beiden Bändern
- $M$  akzeptiert genau dann, wenn  $M_1$  oder  $M_2$  akzeptieren
- Die Eigenschaften der  $M_i$  bleiben erhalten und es gilt  $L(M) = L_1 \cup L_2$

# NACHWEIS DER ABSCHLUSSEIGENSCHAFTEN

## ● Vereinigung $L_1 \cup L_2$

- Sei  $L_i = L(M_i)$ , wobei  $M_i = (Q_i, \Sigma_i, \Gamma_i, \delta_i, q_{0,i}, B, F_i)$  disjunkt
- Bei Eingabe eines Wortes  $w$  kopiert  $M$  das Wort auf zwei Hilfsbänder und simuliert  $M_1$  und  $M_2$  auf den beiden Bändern
- $M$  akzeptiert genau dann, wenn  $M_1$  oder  $M_2$  akzeptieren
- Die Eigenschaften der  $M_i$  bleiben erhalten und es gilt  $L(M) = L_1 \cup L_2$

## ● Durchschnitt $L_1 \cap L_2$

- Bei Eingabe eines Wortes  $w$  kopiert  $M$  das Wort auf zwei Hilfsbänder und simuliert  $M_1$  und  $M_2$  auf den beiden Bändern
- $M$  akzeptiert genau dann, wenn  $M_1$  und  $M_2$  akzeptieren
- Die Eigenschaften der  $M_i$  bleiben erhalten und es gilt  $L(M) = L_1 \cap L_2$

# NACHWEIS DER ABSCHLUSSEIGENSCHAFTEN

## ● Vereinigung $L_1 \cup L_2$

- Sei  $L_i = L(M_i)$ , wobei  $M_i = (Q_i, \Sigma_i, \Gamma_i, \delta_i, q_{0,i}, B, F_i)$  disjunkt
- Bei Eingabe eines Wortes  $w$  kopiert  $M$  das Wort auf zwei Hilfsbänder und simuliert  $M_1$  und  $M_2$  auf den beiden Bändern
- $M$  akzeptiert genau dann, wenn  $M_1$  oder  $M_2$  akzeptieren
- Die Eigenschaften der  $M_i$  bleiben erhalten und es gilt  $L(M) = L_1 \cup L_2$

## ● Durchschnitt $L_1 \cap L_2$

- Bei Eingabe eines Wortes  $w$  kopiert  $M$  das Wort auf zwei Hilfsbänder und simuliert  $M_1$  und  $M_2$  auf den beiden Bändern
- $M$  akzeptiert genau dann, wenn  $M_1$  und  $M_2$  akzeptieren
- Die Eigenschaften der  $M_i$  bleiben erhalten und es gilt  $L(M) = L_1 \cap L_2$

## ● Spiegelung $L_1^R$

- Bei Eingabe eines Wortes  $w$  kopiert  $M$  das Wort umgedreht auf ein Hilfsband und simuliert  $M_1$  auf diesem Band
- $M$  akzeptiert genau dann, wenn  $M_1$  akzeptiert
- Die Eigenschaften von  $M_1$  bleiben erhalten und es gilt  $L(M) = L_1^R$

## NACHWEIS DER ABSCHLUSSEIGENSCHAFTEN II

- **Verkettung  $L_1 \circ L_2$**

- Bei Eingabe eines Wortes  $w$  wählt  $M$  nichtdeterministisch eine Zerlegung des Wort  $w = w_1 \circ w_2$ , kopiert die  $w_i$  auf zwei Hilfsbänder und simuliert  $M_1$  und  $M_2$  entsprechend
- $M$  akzeptiert genau dann, wenn  $M_1$  und  $M_2$  akzeptieren
- Die Eigenschaften der  $M_i$  bleiben erhalten und es gilt  $L(M) = L_1 \circ L_2$

## NACHWEIS DER ABSCHLUSSEIGENSCHAFTEN II

- **Verkettung  $L_1 \circ L_2$**

- Bei Eingabe eines Wortes  $w$  wählt  $M$  nichtdeterministisch eine Zerlegung des Wort  $w = w_1 \circ w_2$ , kopiert die  $w_i$  auf zwei Hilfsbänder und simuliert  $M_1$  und  $M_2$  entsprechend
- $M$  akzeptiert genau dann, wenn  $M_1$  und  $M_2$  akzeptieren
- Die Eigenschaften der  $M_i$  bleiben erhalten und es gilt  $L(M) = L_1 \circ L_2$

- **Hülle  $L_1^*$**

- Bei Eingabe eines Wortes  $w$  wählt  $M$  nichtdeterministisch eine Zerlegung des Wortes  $w = w_1 \circ \dots \circ w_n$ , kopiert die  $w_i$  der Reihe nach auf ein Hilfsband und simuliert  $M_1$  entsprechend
- $M$  akzeptiert genau dann, wenn  $M_1$  alle  $w_i$  akzeptiert
- Die Eigenschaften von  $M_1$  bleiben erhalten und es gilt  $L(M) = L_1^*$

# NACHWEIS DER ABSCHLUSSEIGENSCHAFTEN II

## ● Verkettung $L_1 \circ L_2$

- Bei Eingabe eines Wortes  $w$  wählt  $M$  nichtdeterministisch eine Zerlegung des Wort  $w = w_1 \circ w_2$ , kopiert die  $w_i$  auf zwei Hilfsbänder und simuliert  $M_1$  und  $M_2$  entsprechend
- $M$  akzeptiert genau dann, wenn  $M_1$  und  $M_2$  akzeptieren
- Die Eigenschaften der  $M_i$  bleiben erhalten und es gilt  $L(M) = L_1 \circ L_2$

## ● Hülle $L_1^*$

- Bei Eingabe eines Wortes  $w$  wählt  $M$  nichtdeterministisch eine Zerlegung des Wortes  $w = w_1 \circ \dots \circ w_n$ , kopiert die  $w_i$  der Reihe nach auf ein Hilfsband und simuliert  $M_1$  entsprechend
- $M$  akzeptiert genau dann, wenn  $M_1$  alle  $w_i$  akzeptiert
- Die Eigenschaften von  $M_1$  bleiben erhalten und es gilt  $L(M) = L_1^*$

## ● Homomorphismen $h(L_1)$

- Bei Eingabe eines Wortes  $w$  wählt  $M$  nichtdeterministisch eine Zerlegung des Wort  $w = w_1 \circ \dots \circ w_n$  mit  $w_i = h(a_i)$ , kopiert  $v = a_1 \dots a_n$  auf ein Hilfsband und simuliert  $M_1$  entsprechend
- $M$  akzeptiert genau dann, wenn  $M_1$  das Wort  $v$  akzeptiert
- Die Eigenschaften von  $M_1$  bleiben erhalten und es gilt  $L(M) = h(L_1)$

- **Inverse Homomorphismen  $h^{-1}(L_1)$** 
  - Bei Eingabe eines Wortes  $w = a_1..a_n$  bestimmt  $M$  das Wort  $v = h(a_1)..h(a_n)$ , kopiert es auf ein Hilfsband und simuliert  $M_1$
  - $M$  akzeptiert genau dann, wenn  $M_1$  das Wort  $v$  akzeptiert
  - Es gilt  $L(M) = h^{-1}(L_1)$
  - Beweis gilt in dieser Form nur für (semi-)entscheidbare Sprachen  
Für LBA's ist Simulation eines Bandes  $k$ -facher Länge erforderlich

## ● Inverse Homomorphismen $h^{-1}(L_1)$

- Bei Eingabe eines Wortes  $w = a_1..a_n$  bestimmt  $M$  das Wort  $v = h(a_1)..h(a_n)$ , kopiert es auf ein Hilfsband und simuliert  $M_1$
- $M$  akzeptiert genau dann, wenn  $M_1$  das Wort  $v$  akzeptiert
- Es gilt  $L(M)=h^{-1}(L_1)$
- Beweis gilt in dieser Form nur für (semi-)entscheidbare Sprachen  
Für LBA's ist Simulation eines Bandes  $k$ -facher Länge erforderlich

## ● Komplement $\overline{L_1}$

- Bei Eingabe eines Wortes  $w$  simuliert  $M$  die Berechnung von  $M_1$  und akzeptiert genau dann, wenn  $M_1$  nicht akzeptiert
- Es gilt  $L(M)=\overline{L_1}$
- Die Eigenschaften von  $M_1$  bleiben nur erhalten, wenn  $M_1$  terminiert  
Bei semi-entscheidbaren Sprachen terminiert die Berechnung für  $w \in \overline{L_1}$  nicht

## ● Inverse Homomorphismen $h^{-1}(L_1)$

- Bei Eingabe eines Wortes  $w = a_1..a_n$  bestimmt  $M$  das Wort  $v = h(a_1)..h(a_n)$ , kopiert es auf ein Hilfsband und simuliert  $M_1$
- $M$  akzeptiert genau dann, wenn  $M_1$  das Wort  $v$  akzeptiert
- Es gilt  $L(M)=h^{-1}(L_1)$
- Beweis gilt in dieser Form nur für (semi-)entscheidbare Sprachen  
Für LBA's ist Simulation eines Bandes  $k$ -facher Länge erforderlich

## ● Komplement $\overline{L_1}$

- Bei Eingabe eines Wortes  $w$  simuliert  $M$  die Berechnung von  $M_1$  und akzeptiert genau dann, wenn  $M_1$  nicht akzeptiert
- Es gilt  $L(M)=\overline{L_1}$
- Die Eigenschaften von  $M_1$  bleiben nur erhalten, wenn  $M_1$  terminiert  
Bei semi-entscheidbaren Sprachen terminiert die Berechnung für  $w \in \overline{L_1}$  nicht

## ● Differenz $L_1-L_2$

- Mathematische Begründung:  $L_1-L_2 = L_1 \cap \overline{L_2}$
- Abgeschlossenheit unter Differenz gilt für abzählbare Sprachen nicht!

- **$L$  entscheidbar  $\Leftrightarrow L$  und  $\bar{L}$  semi-entscheidbar**

“ $\Rightarrow$ ”: Entscheidbare Sprachen sind abgeschlossen unter Komplement

“ $\Leftarrow$ ”: Sei  $L=L(M_1)$  und  $\bar{L}=L(M_2)$ .

- Bei Eingabe eines Wortes  $w$  kopiert  $M$  das Wort auf zwei Hilfsbänder und simuliert  $M_1$  und  $M_2$  auf den beiden Bändern
- $M$  akzeptiert genau dann, wenn  $M_1$  akzeptiert und terminiert ohne zu akzeptieren, wenn  $M_2$  akzeptiert
- Da eine der beiden Maschinen das Wort  $w$  akzeptieren muß, terminiert  $M$  und es gilt  $L(M)=L$

# AUFZÄHLBARKEIT VS. ENTSCHIEDBARKEIT

- **$L$  entscheidbar  $\Leftrightarrow L$  und  $\bar{L}$  semi-entscheidbar**

“ $\Rightarrow$ ”: Entscheidbare Sprachen sind abgeschlossen unter Komplement

“ $\Leftarrow$ ”: Sei  $L=L(M_1)$  und  $\bar{L}=L(M_2)$ .

- Bei Eingabe eines Wortes  $w$  kopiert  $M$  das Wort auf zwei Hilfsbänder und simuliert  $M_1$  und  $M_2$  auf den beiden Bändern
- $M$  akzeptiert genau dann, wenn  $M_1$  akzeptiert und terminiert ohne zu akzeptieren, wenn  $M_2$  akzeptiert
- Da eine der beiden Maschinen das Wort  $w$  akzeptieren muß, terminiert  $M$  und es gilt  $L(M)=L$

- **Jede endliche Sprache  $L$  ist entscheidbar**

- Jede endliche Sprache ist als Liste von Wörtern  $[w_1; \dots; w_n]$  darstellbar
- Bei Eingabe eines Wortes  $w$  vergleicht  $M$  das Wort mit dieser Liste

# AUFZÄHLBARKEIT VS. ENTSCHIEDBARKEIT

- **$L$  entscheidbar  $\Leftrightarrow L$  und  $\bar{L}$  semi-entscheidbar**

“ $\Rightarrow$ ”: Entscheidbare Sprachen sind abgeschlossen unter Komplement

“ $\Leftarrow$ ”: Sei  $L=L(M_1)$  und  $\bar{L}=L(M_2)$ .

- Bei Eingabe eines Wortes  $w$  kopiert  $M$  das Wort auf zwei Hilfsbänder und simuliert  $M_1$  und  $M_2$  auf den beiden Bändern
- $M$  akzeptiert genau dann, wenn  $M_1$  akzeptiert und terminiert ohne zu akzeptieren, wenn  $M_2$  akzeptiert
- Da eine der beiden Maschinen das Wort  $w$  akzeptieren muß, terminiert  $M$  und es gilt  $L(M)=L$

- **Jede endliche Sprache  $L$  ist entscheidbar**

- Jede endliche Sprache ist als Liste von Wörtern  $[w_1; \dots; w_n]$  darstellbar
- Bei Eingabe eines Wortes  $w$  vergleicht  $M$  das Wort mit dieser Liste

- **$L$  semi-entscheidbar  $\Leftrightarrow$  es gibt ein entscheidbares**

**$L' \subseteq \Sigma^* \times \Sigma^*$  mit  $L = \{w \mid \exists v. (w, v) \in L'\}$**  (Projektionssatz)

- Aufwendiger Beweis, benötigt schrittweise Simulation von Maschinen