

# 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**

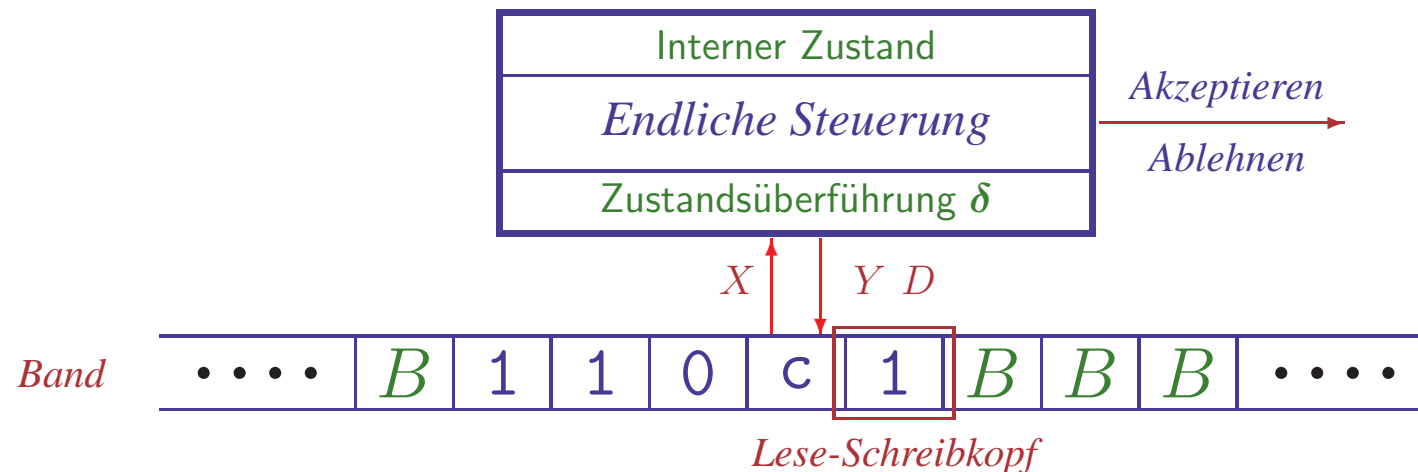
- Verarbeitung interner **Zustände** abhängig von gelesenen Daten
- Lese- und Schreibzugriff auf **externen Speicher**
- Minimal mögliche Erweiterung

- **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

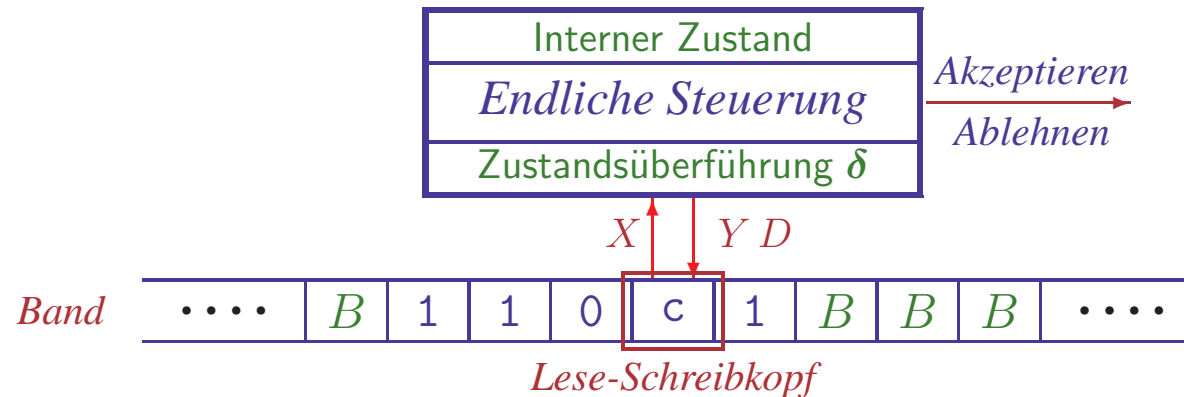
**Wähle unendliches, bewegliches Band als Speicher**

# TURINGMASCHINEN INTUITIV



- **Endlicher Automat + lineares Band**
  - Endliche Steuerung liest Eingabesymbole
  - Gleichzeitig wird Bandsymbol unter **Lese-Schreibkopf** gelesen
- **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

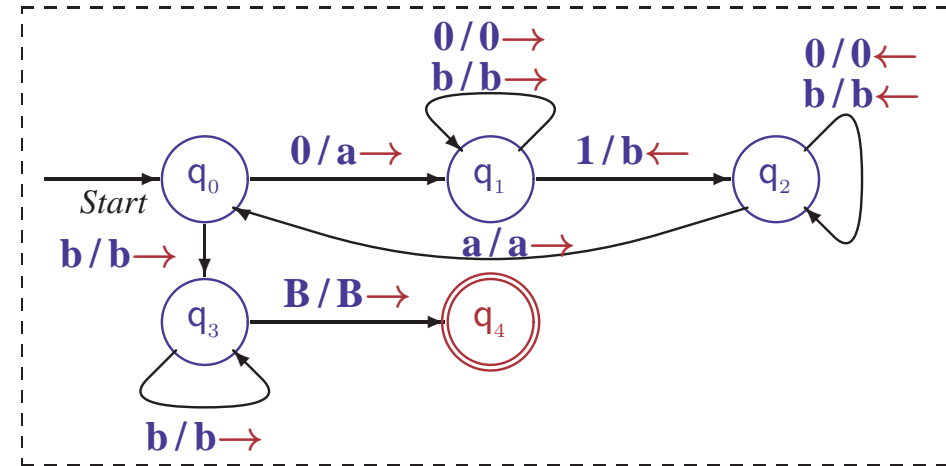
$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  mit

- $Q$  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  $\Gamma$  implizit durch Diagramm bestimmt, **Leersymbol** heißt  $B$



## • Übergangstabellen

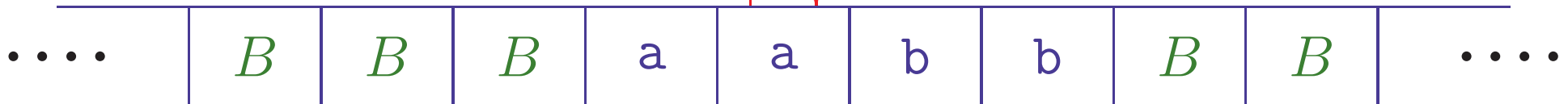
- Funktionstabelle für  $\delta$   
— heißt “ $\delta$  nicht definiert”
- Pfeil  $\rightarrow$  kennzeichnet  $q_0$
- Stern  $*$  kennzeichnet  $F$
- $\Sigma$ ,  $\Gamma$  und  $B$  implizit bestimmt

$Q \setminus \Gamma$	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**



**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  $\mathbf{K} = (\mathbf{u}, \mathbf{q}, \mathbf{v}) \in \Gamma^* \times Q \times \Gamma^+$ 
  - $u, v$ : String links/rechts vom Kopf

Achtung: im Buch wird das Tripel als ein (!) String  $uq v$  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

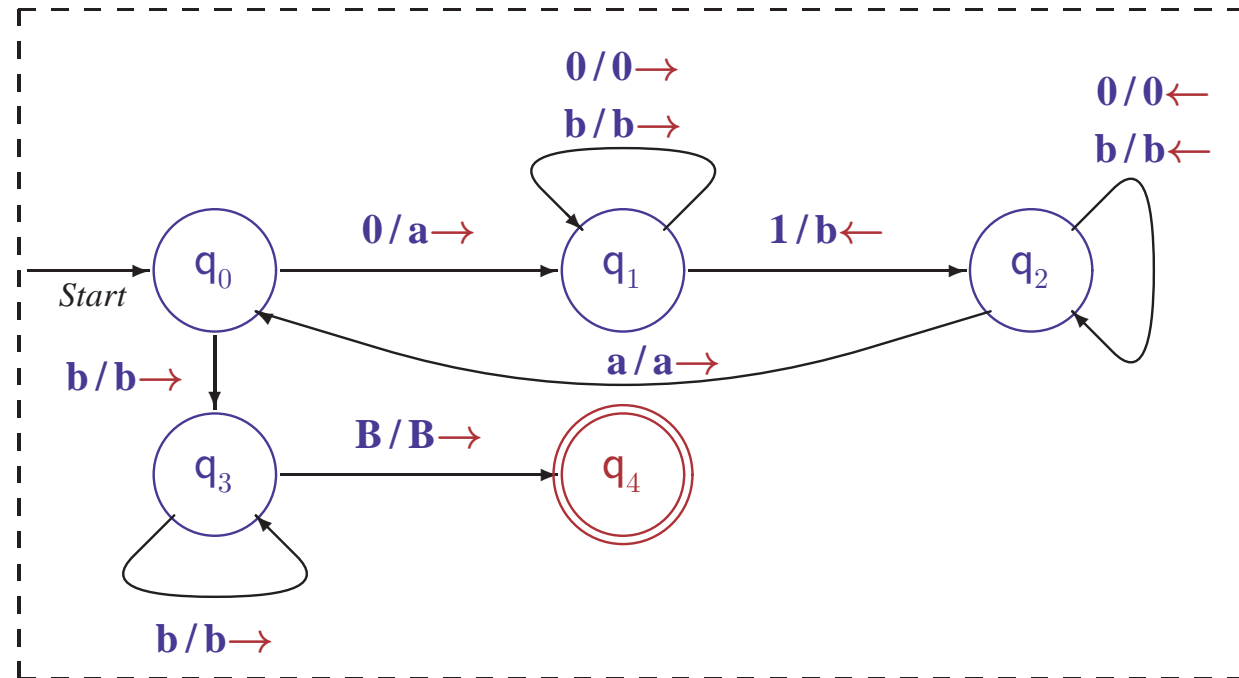
- $(\epsilon, q, Xv) \vdash (\epsilon, p, BYv),$  falls  $\delta(q, X) = (p, Y, L)$
- $(uZ, q, X) \vdash (u, p, Z),$  falls  $\delta(q, X) = (p, B, L)$
- $(u, q, X) \vdash (uY, p, B),$  falls  $\delta(q, X) = (p, Y, R)$
- $(\epsilon, q, Xv) \vdash (\epsilon, p, v),$  falls  $\delta(q, X) = (p, B, R)$

$\mathbf{K}_1 \vdash^* \mathbf{K}_2$ , 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

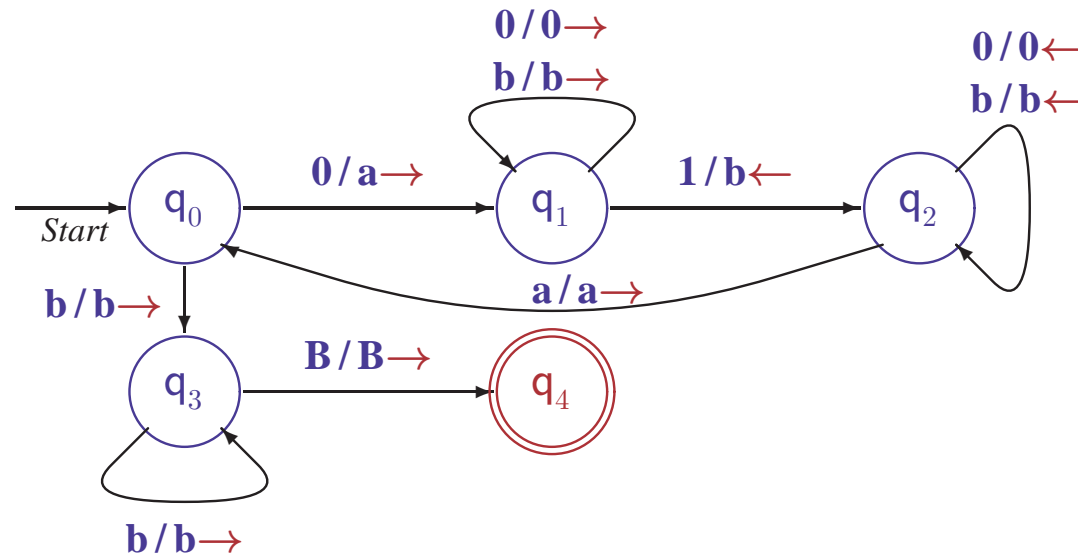
- **Semi-entscheidbare Sprache**

- Sprache, die von einer Turingmaschine  $M$  akzeptiert wird
- Alternative Bezeichnungen: **(rekursiv) aufzählbare Sprache**  
**Turing-akzeptierbare Sprache**

- **Entscheidbare Sprache**

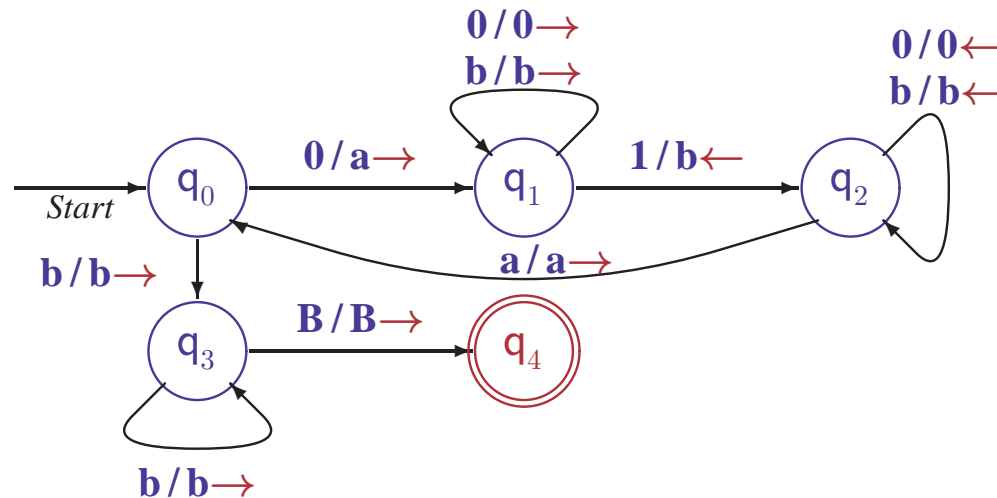
- Sprache, die von einer Turingmaschine  $M$  akzeptiert wird, die bei jeder Eingabe terminiert
- Alternative Bezeichnung: **rekursive Sprache**

# 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  $q_1$ , 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

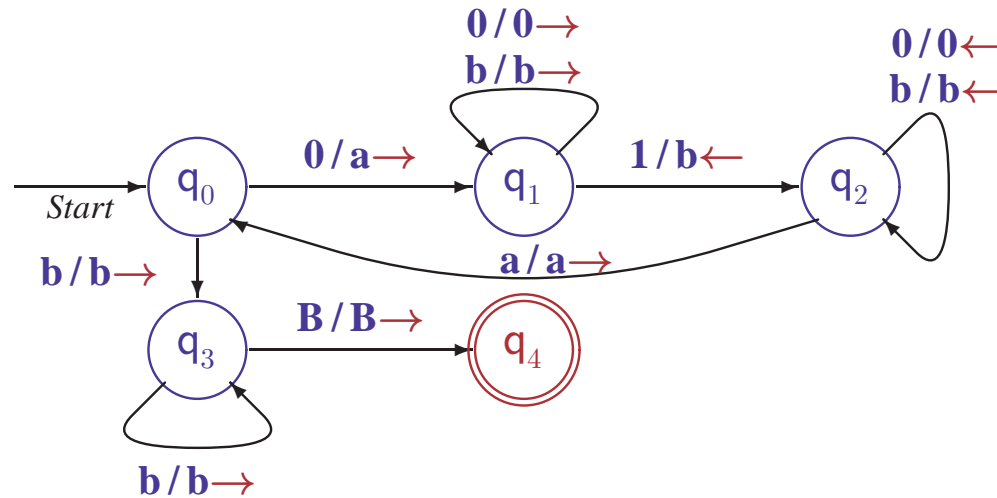


$(\epsilon, q_0, w) \vdash^* (u, q_4, v)$  wenn  $w = 0^n 1^n$  für ein  $n \geq 1$

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

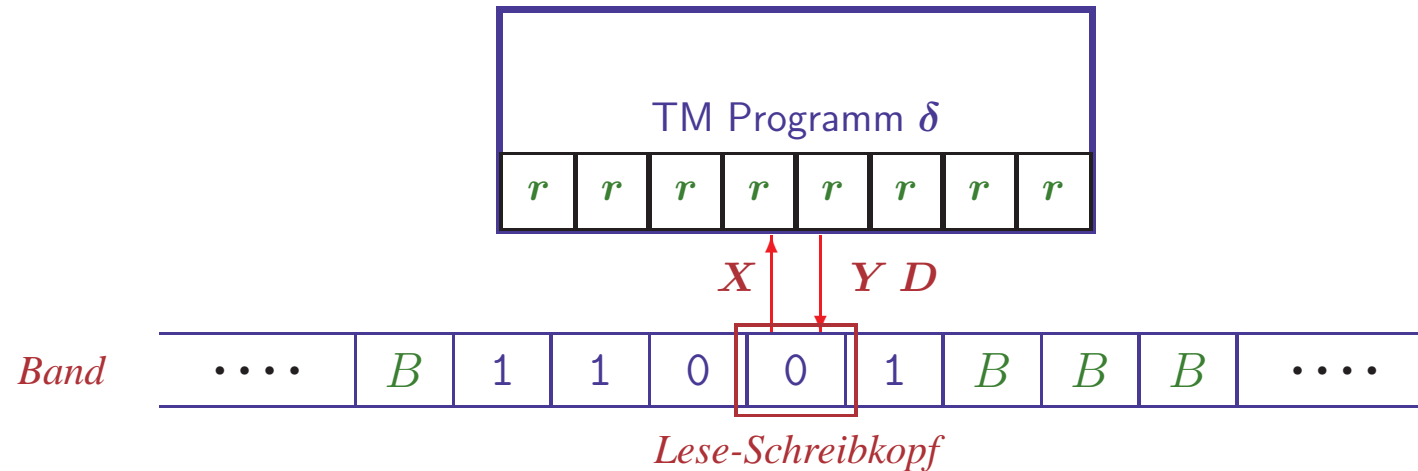
1. Die Schleife  $q_0 - q_1 - q_2$  wandelt je eine 0 in ein  $a$  und eine 1 in  $b$  um
2. Am Ende der Schleife stehen alle 0 und 1 rechts vom Kopf und alle  $a$  links davon
3. Da das Eingabewort zu  $\{0, 1\}^*$  gehört, stehen in  $q_0$  gleich viele  $a$  und  $b$  auf dem Band
4. Um  $q_4$  von  $q_0$  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.

## 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

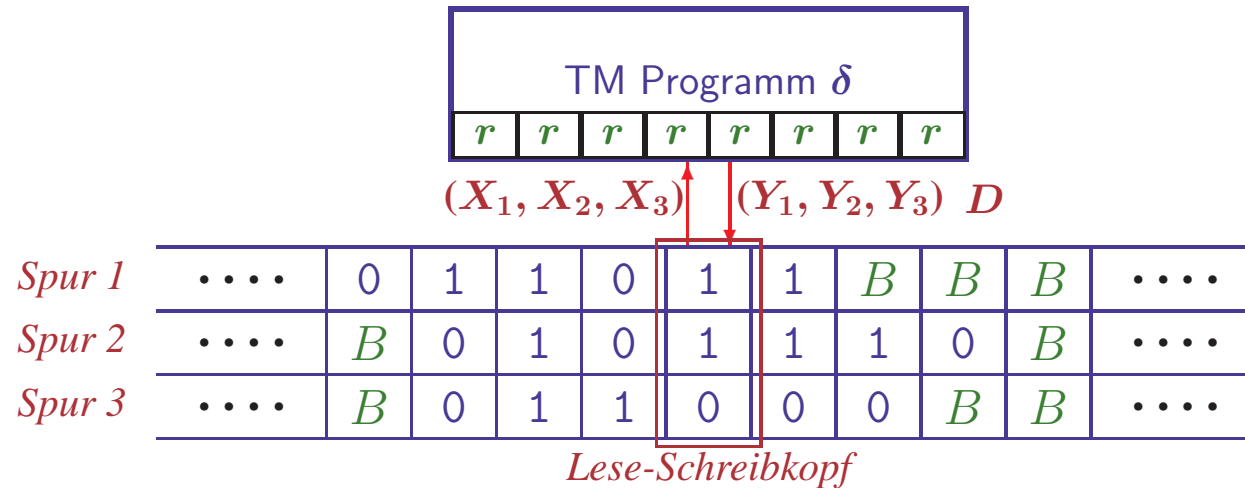
### • Einfache Lösung mit Registern

- 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 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

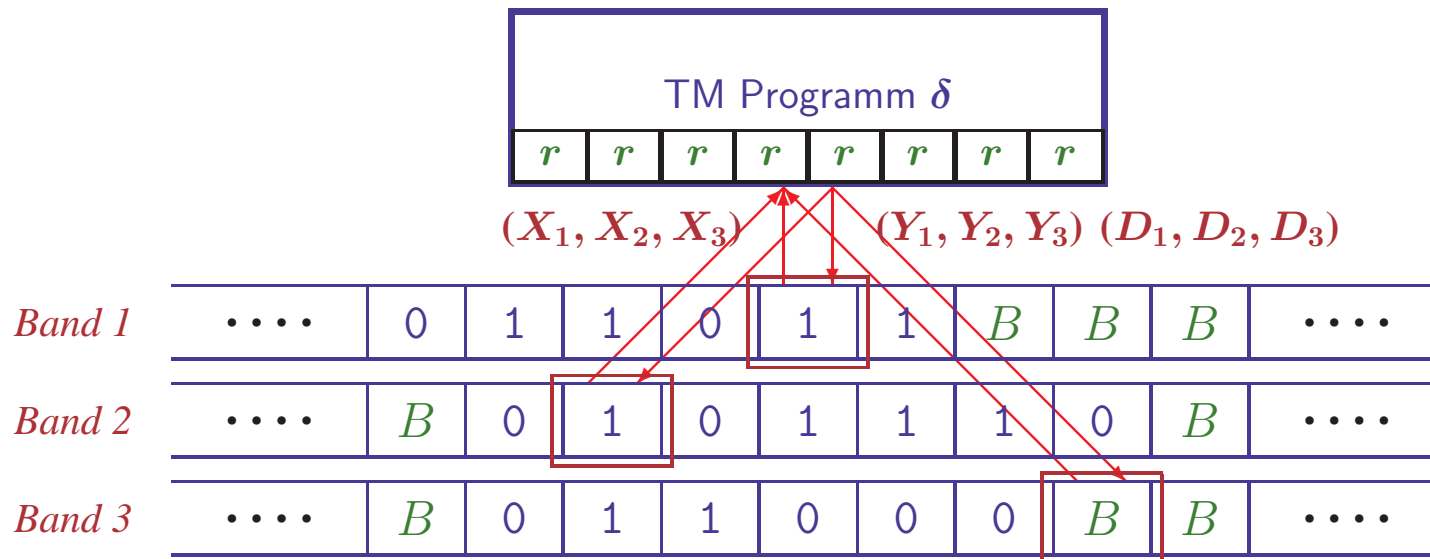
z.B. zur Erkennung von  $\{w\#w \mid w \in \{0, 1\}^*\}$

↪ 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



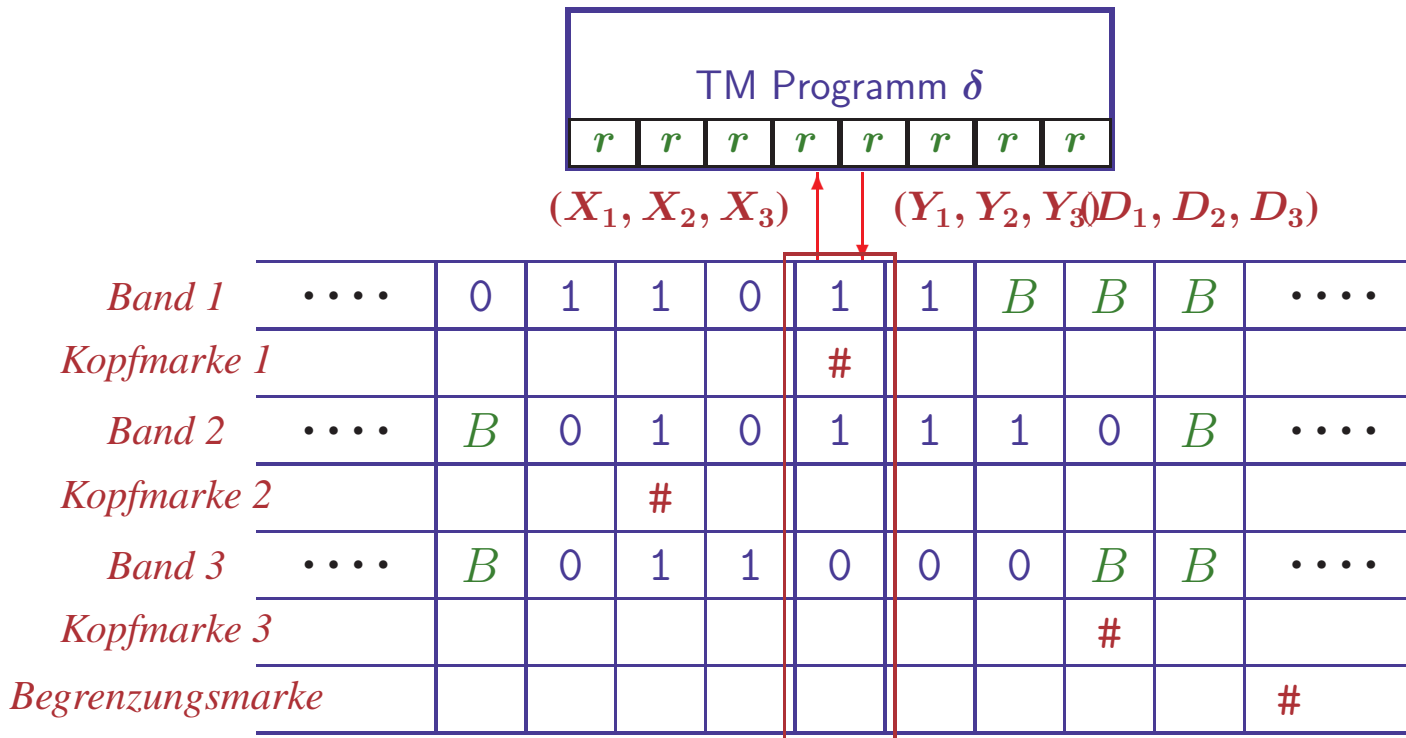
- **Maschine verwaltet mehrere Bänder**

- 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  $M$** 
  - $M'$  erhält Eingabewort von  $M$  und gibt Resultat an  $M$  zurück
  - $M$  wechselt nach Ausführung von  $M'$  in festen Folgezustand
  - Anwendungsbeispiel: Multiplikation als wiederholte Addition
- **Simulation wie bei Assembler-Unterprogrammen**
  - Umbenennung aller Zustände von  $M'$  zur Konfliktvermeidung
  - Ergänze Zustand  $q_r$  für Rücksprung ins aufrufende Programm
  - Ergänze separates Arbeitsband für Unterprogramm
  - **Aufruf**: Speichere Rücksprungadresse (Zustand von  $M$ ) in Register
  - Kopiere Eingabe für Unterprogramme auf Arbeitsband für  $M'$
  - 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  $(u, \mathbf{q}, v)$  beschreiben

### 4. Zählmaschinen

→ 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



- **Allgemeinstes Maschinenmodell**
  - Deterministischer endlicher Automat mit unendlichem Speicherband
  - “Beliebiger” Zugriff auf Speicherzellen
  - Erkennung von Wörtern durch Endzustand
- **Verhaltensanalyse durch Konfigurationsübergänge**
  - Konfigurationen beschreiben Zustand, Bandinhalt & Kopfposition
- **Äquivalent zu realen Computern**
  - Register, mehrere Bänder, Unterprogramme, etc. simulierbar
  - Beschränkte Maschinenmodelle sind ebenfalls gleich mächtig