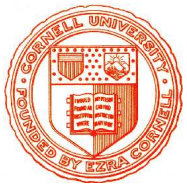
Theoretische Informatik II



Einheit 4

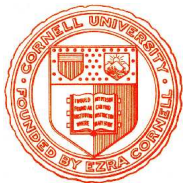
Allgemeine und kontextsensitive Sprachen

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

Jenseits von Kontextfreiheit

- **Viele wichtige Konzepte sind nicht kontextfrei**
  - Sind Bezeichner im Programmkörper deklariert?
  - $\{ ww \mid w \in \{0,1\}^* \}$: erscheint Programmcode doppelt?
  - $\{0^n 1^n 2^n | n \in \mathbb{N}\}$: 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 II



Einheit 4.1

Turingmaschinen

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

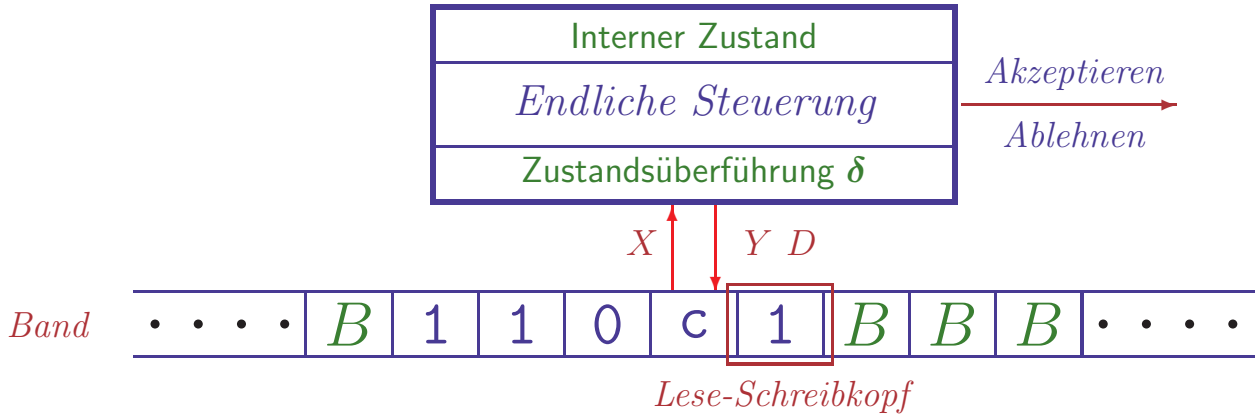
Turingmaschinen (Alan Turing, 1936)

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

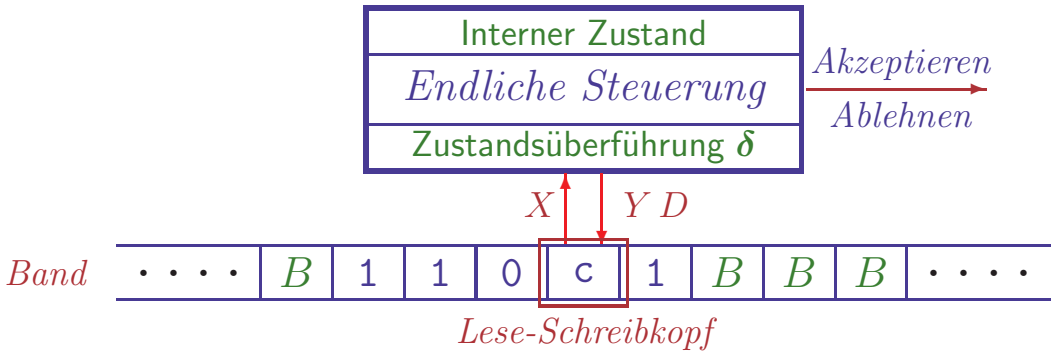
Wähle unendliches, bewegliches Band als Speicher

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



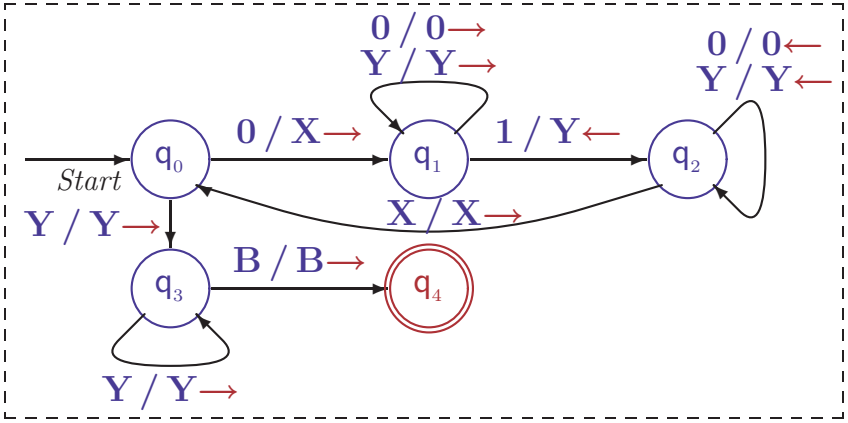
Eine **Turingmaschine (TM)** ist ein 7-Tupel

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

- Q nichtleere endliche **Zustandsmenge**
- Σ endliches **Eingabealphabet**
- $\Gamma \supseteq \Sigma$ endliches **Bandalphabet**
- $\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ (partielle) **Überführungsfunktion**
- $q_0 \in Q$ **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 $q \xrightarrow{X/Y\,D} p$



  - Σ und Γ implizit durch Diagramm bestimmt, Leersymbol heißt B

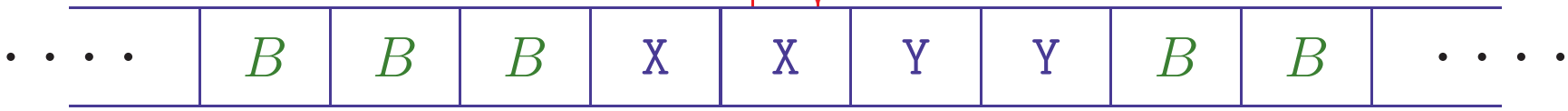
- **Übergangstabellen**
  - Funktionstabelle für δ
    — heißt "δ nicht definiert"
  - Pfeil $\rightarrow$ kennzeichnet q_0
  - Stern * kennzeichnet F
  - Σ, Γ und B implizit bestimmt

Q \ Γ	0	1	X	Y	B
→ q_0	(q_1,X,R)	—	—	(q_3,Y,R)	—
q_1	(q_1,0,R)	(q_2,Y,L)	—	(q_1,Y,R)	—
q_2	(q_2,0,L)	—	(q_0,X,R)	(q_2,Y,L)	—
q_3	—	—	—	(q_3,Y,R)	(q_4,B,R)
* q_4	—	—	—	—	—

Abarbeitung von Turing-Programmen

$Q \setminus \Gamma$	0	1	X	Y	B
$\rightarrow q_0$	(q_1,X,R)	—	—	(q_3,Y,R)	—
q_1	$(q_1,0,R)$	(q_2,Y,L)	—	(q_1,Y,R)	—
q_2	$(q_2,0,L)$	—	(q_0,X,R)	(q_2,Y,L)	—
q_3	—	—	—	(q_3,Y,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!!

Arbeitsweise von Turingmaschinen präzisiert

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

    Achtung: im Buch wird das Tripel als ein (!) String $\boldsymbol{uqv}$ 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 $\vdash^*$**
  - $(\boldsymbol{uZ}, \boldsymbol{q}, \boldsymbol{Xv}) \vdash (\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{ZYv})$, falls $\delta(q, X) = (p, Y, L)$
  - $(\boldsymbol{u}, \boldsymbol{q}, \boldsymbol{Xv}) \vdash (\boldsymbol{uY}, \boldsymbol{p}, \boldsymbol{v})$, falls $\delta(q, X) = (p, Y, R)$

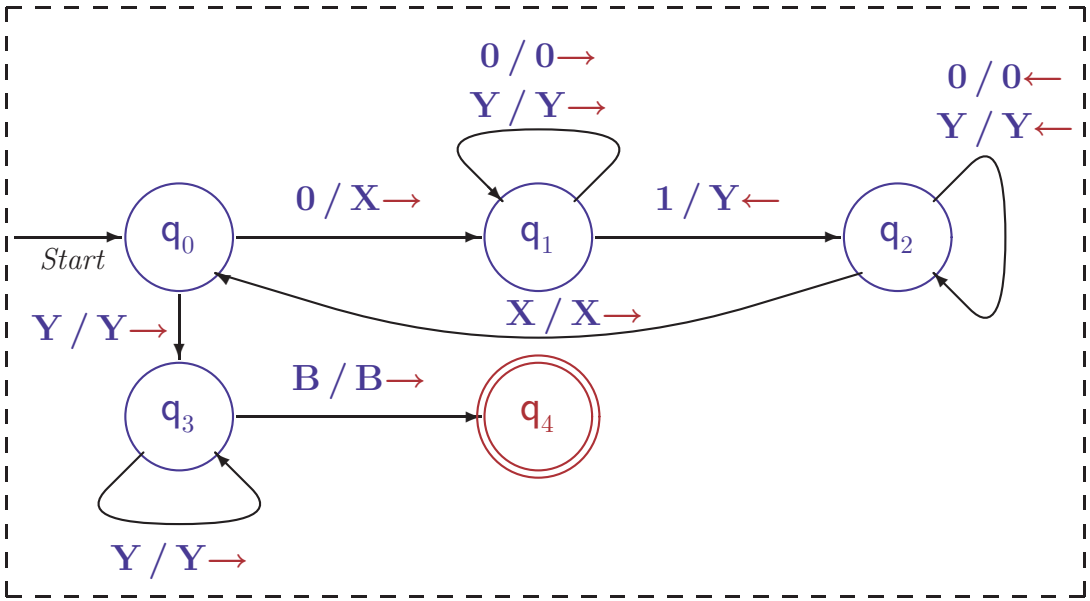
  Sonderfälle
  - $(\boldsymbol{\epsilon}, \boldsymbol{q}, \boldsymbol{Xv}) \vdash (\boldsymbol{\epsilon}, \boldsymbol{p}, \boldsymbol{BYv})$, falls $\delta(q, X) = (p, Y, L)$
  - $(\boldsymbol{uZ}, \boldsymbol{q}, \boldsymbol{X}) \vdash (\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{Z})$, falls $\delta(q,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

  $\boldsymbol{K_1} \vdash^* \boldsymbol{K_2}$, 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 $(\epsilon, q_0, 0011)$

$(\epsilon, q_0, 0011)$
$\vdash (X, q_1, 011)$
$\vdash (X0, q_1, 11)$
$\vdash (X, q_2, 0Y1)$
$\vdash (\epsilon, q_2, X0Y1)$
$\vdash (X, q_0, 0Y1)$
$\vdash (XX, q_1, Y1)$
$\vdash (XXY, q_1, 1)$
$\vdash (XX, q_2, YY)$
$\vdash (X, q_2, XYY)$
$\vdash (XX, q_0, YY)$
$\vdash (XXY, q_3, Y)$
$\vdash (XXYY, q_3, B)$
$\vdash \boldsymbol{(XXYYB, q_4, B)}$



Maschine terminiert, Endzustand erreicht, Eingabe wird akzeptiert

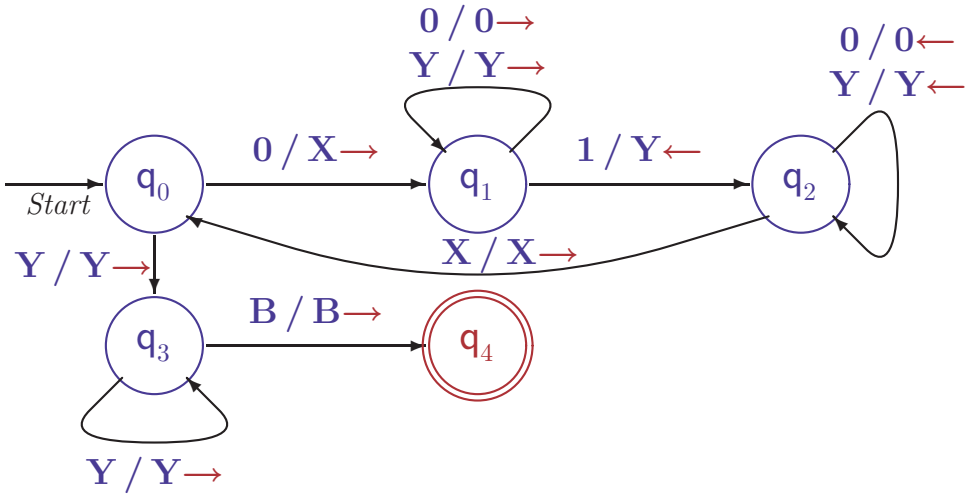
Die Sprache einer Turingmaschine

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

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

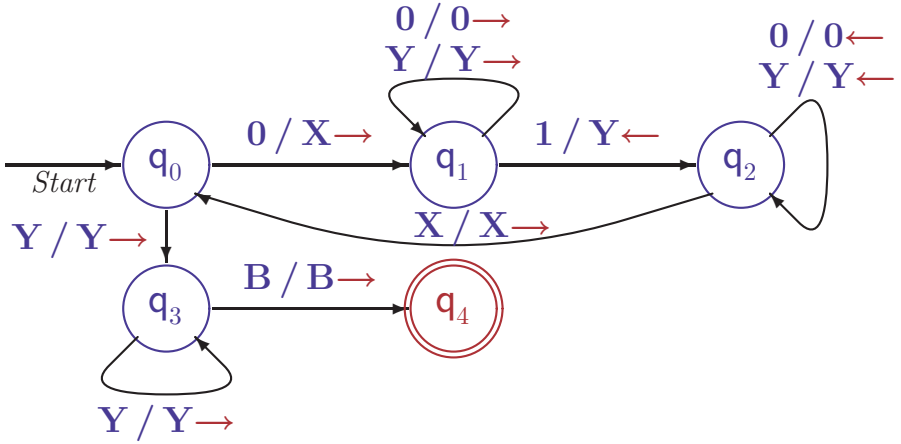
  - Bei Einhalten der Konvention hält M im akzeptierenden Zustand an
- **Semi-entscheidbare Sprache**
  - Sprache, die von einer Turingmaschine M akzeptiert wird
  - Alternative Bezeichnung: **(rekursiv) aufzählbare Sprache**
- **Entscheidbare Sprache**
  - Sprache, die von einer Turingmaschine M akzeptiert 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 X triggert Umwandeln einer 1 in Y
  - 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: $L(M) = \{0^n 1^n | n \geq 1\}$**
  - $(\epsilon, q_0, w) \vdash^* (u, q_4, v)$ genau dann, wenn $w = 0^n 1^n$ für ein $n \geq 1$

Nachweis der erkannten Sprache



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

- $(u, q_0, 0v) \vdash (uX, q_1, v)$ für alle $u, v \in \Gamma^*$
- $(u, q_0, 0wv) \vdash^* (uXw, q_1, v)$ für alle $u, v \in \Gamma^*, w \in \{0, Y\}^*$
- $(u, q_0, 0w1v) \vdash^* (u, q_2, XwYv)$ für alle $u, v \in \Gamma^*, w \in \{0, Y\}^*$
- $(u, q_0, 0w1v) \vdash^* (uX, q_0, wYv)$ für alle $u, v \in \Gamma^*, w \in \{0, Y\}^*$
- $(\epsilon, q_0, 0^k w 1^k v) \vdash^* (X^k, q_0, wY^k v)$ für alle $v \in \Gamma^*, w \in \{0, Y\}^*, k \in \mathbb{N}$
- $(\epsilon, q_0, 0^k 1^k v) \vdash^* (X^k Y^k, q_3, v)$ für alle $v \in \Gamma^*, k \geq 1$
- $(\epsilon, q_0, 0^k 1^k) \vdash^* (X^k Y^k, q_3, B)$ für alle $v \in \Gamma^*, k \geq 1$
- $\mathbf{(\epsilon, q_0, 0^k 1^k) \vdash^* (X^k Y^k B, q_4, B)}$ für alle $v \in \Gamma^*, k \geq 1$

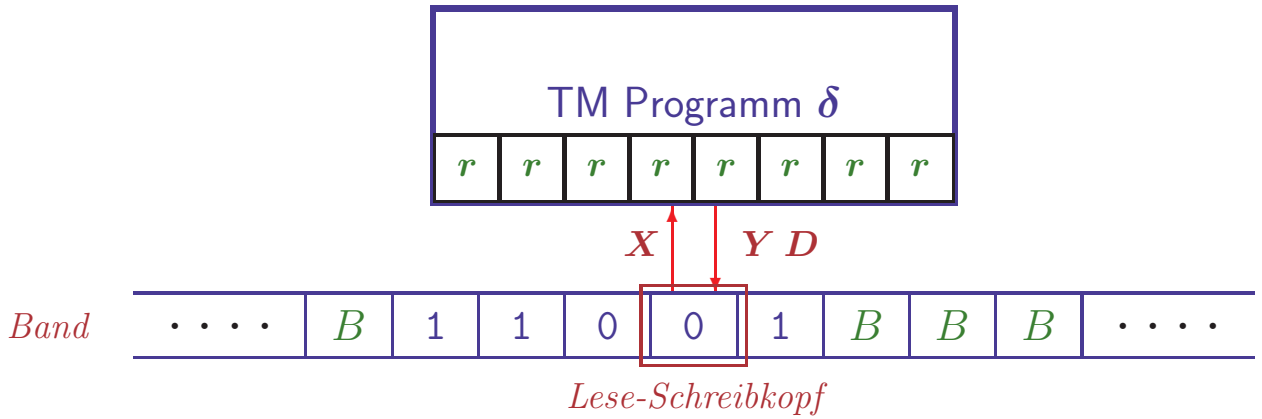
Argument, warum andere Wörter nicht akzeptiert werden, ist aufwendiger

Ausdruckskraft von Turingmaschinen

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**
  - Beliebig große Alphabete (statt binären Daten)
  - Unendliches Speicherband
- **Gegenseitige Simulation ist möglich**
  - Zusätzliche Freiheiten als Programmiertechniken 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 Δ 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'

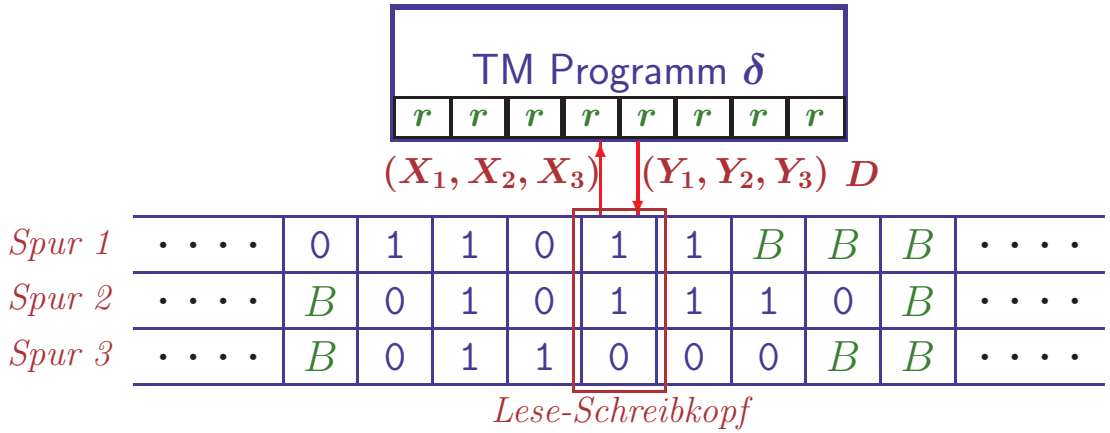
Simulation einer Maschine mit Registern

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

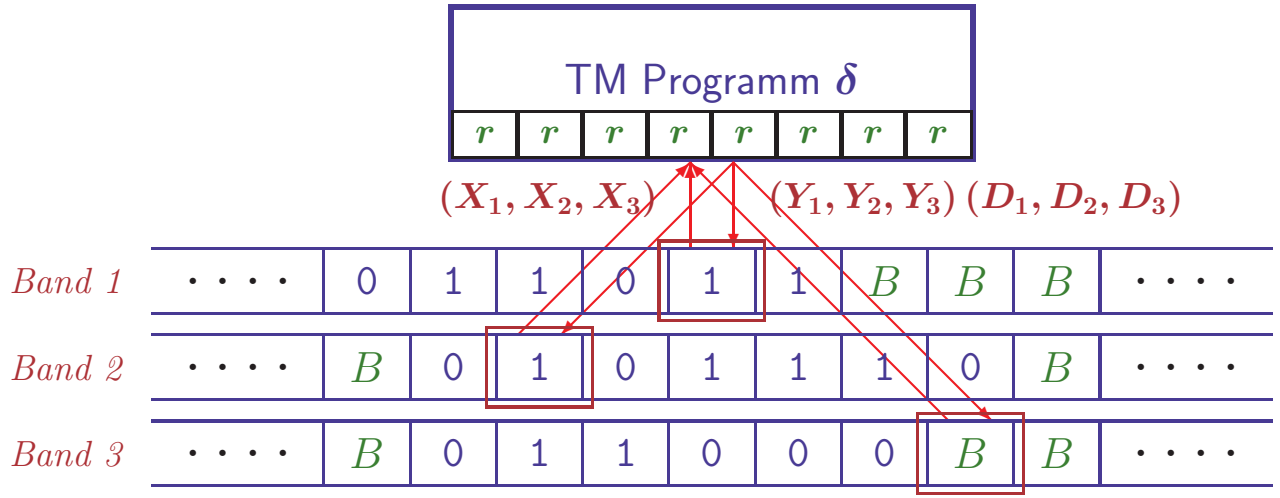
	0	1	B	
$\rightarrow (q_0,B)$	$((q_0,0),0,R)$	$((q_0,1),1,R)$	—	*Erstes Symbol speichern*
$(q_0,0)$	—	$((q_0,0),1,R)$	$((q_1,B),B,R)$	*Mit 0 vergleichen*
$(q_0,1)$	$((q_0,1),0,R)$	—	$((q_1,B),B,R)$	*Mit 1 vergleichen*
$*\ (q_1,B)$	—	—	—	*Vergleich war erfolgreich*
$(q_1,0)$	—	—	—	*(Nicht erreichbar)*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

Programmiertechnik: Mehrere Spuren



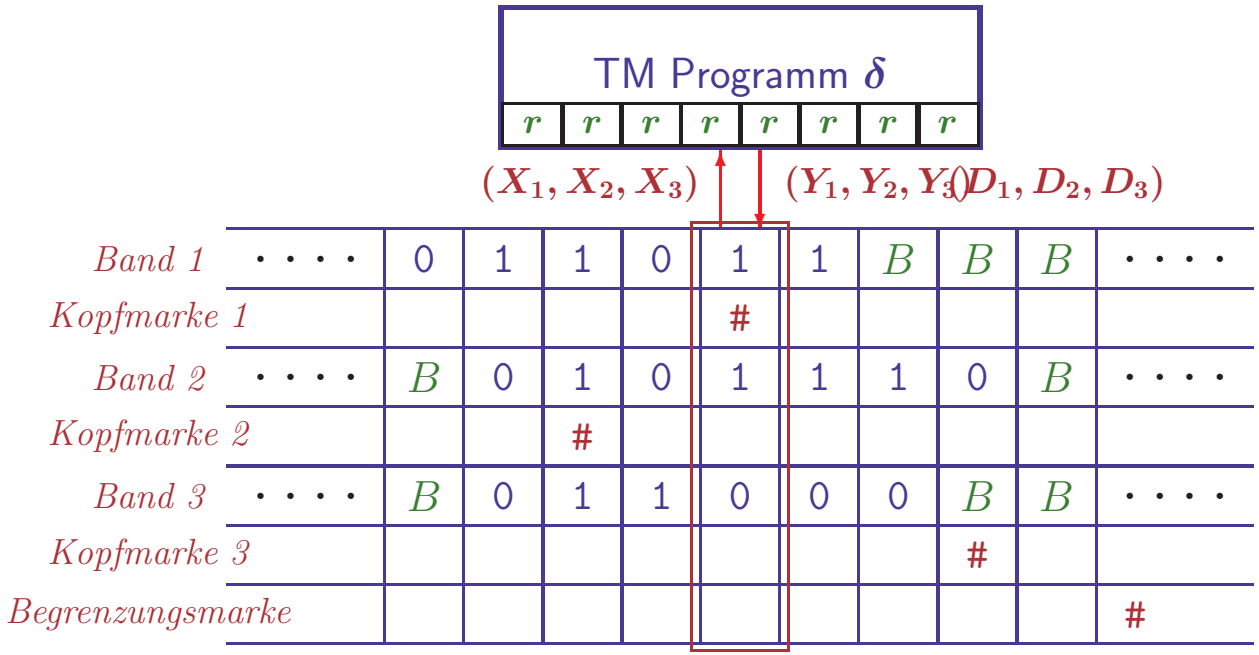
- **Band hat mehrere Datenspuren**
  - Jede Spur enthält ein Symbol des Bandalphabets Γ
  - Alle Symbole werden simultan gelesen und geschrieben
  - Kopf wird "synchron" über das Band bewegt
  - Verwendung: Simultane Verarbeitung von Teilen der Eingabe
    z.B. zur Erkennung von $\{w\#w | w \in \{0,1\}^*\}$ ↦ HMU, §8.3.2
- **Simulation durch erweitertes Bandalphabet**
  - Bei k Spuren wähle Tupelalphabet $\Sigma' := \Sigma^k$
  - In jedem Schritt wird 'ein' Symbol $X := (x_1, .., x_k)$ verarbeitet, wobei x_i dem Symbol auf Spur i entspricht

Programmiertechnik: Mehrere Bänder



- **Maschine verwaltet mehrere Bänder**
  - Jedes Band enthält ein Symbol des Bandalphabets Γ
  - 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 $\mapsto$ HMU, §8.4.3

Ausführung einer anderen TM als Zwischenschritt

- **Aufruf von M' in Überführungsfunktion 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

Beschränkte Turingmaschinenmodelle

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 Γ^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 $(\boldsymbol{u},\boldsymbol{q},\boldsymbol{v})$ beschreiben

4. Zählermaschinen

↦ HMU, §8.5.3/4

– Endliche Anzahl 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)

Der Vergleich mit realen Computern

- **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

Turingmaschinen im Rückblick

- **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