

# Theoretische Informatik HS24

---

Nicolas Wehrli

Übungsstunde 05

22. Oktober 2024

ETH Zürich

*nwehrli@ethz.ch*

- ① Feedback zur Serie
- ② Nichtdeterministische Endliche Automaten  
Mindestanzahl Zustände
- ③ Turing Maschinen
- ④ Midterm Prep

## Feedback zur Serie

---

- Pumping Lemma war mid. Heute Repetition.
- Common Mistakes:
  - I. Ihr zeigt für eine Teilmenge aller möglichen Aufteilungen, dass (i), (ii) und (iii) nicht alle gelten. Es muss für **alle** gezeigt werden.
  - II. Vorsicht bei Case Distinctions Aufgabe 12.b.
- Widerspruch jeweils zu Ende führen.

## Beispielaufgabe Pumpinglemma

Wir zeigen per Pumping Lemma, dass die Sprache

$$L = \{w \in \{a, b, c\}^* \mid w \text{ enthält das Teilwort } ab \text{ gleich oft wie das Teilwort } ba\}$$

nicht regulär ist.

### Pumping Lemma

Sei  $L$  regulär. Dann existiert eine Konstante  $n_0 \in \mathbb{N}$ , so dass jedes Wort  $w \in \Sigma^*$  mit  $|w| \geq n_0$  in drei Teile  $x, y$  und  $z$  zerlegen lässt, das heisst  $w = yxz$ , wobei

- (i)  $|yx| \leq n_0$
- (ii)  $|x| \geq 1$
- (iii) entweder  $\{yx^kz \mid k \in \mathbb{N}\} \subseteq L$  oder  $\{yx^kz \mid k \in \mathbb{N}\} \cap L = \emptyset$ .

# Beispielaufgabe Pumpinglemma

## Lösung

Sei  $L$  regulär.

Nach dem Pumping Lemma existiert eine Konstante  $n_0 \in \mathbb{N}$ , so dass jedes Wort  $w$  mit  $|w| \geq n_0$  die Bedingung des PL erfüllt.

Sei  $w = (abc)^{n_0}(bac)^{n_0}$ . Offensichtlich gilt  $|w| \geq n_0$ . Nach dem PL existiert eine Zerlegung  $w = yxz$ , die (i), (ii) und (iii) erfüllt.

Da  $yxz$  die Bedingung (i) erfüllt, gilt  $|yx| \leq n_0$ . Insbesondere folgt daraus, dass  $x$  komplett in der ersten Hälfte (i.e.  $(abc)^{n_0}$ ) enthalten ist.

Aus (ii) folgt weiter, dass  $x$  mindestens ein Buchstaben enthält.

## Case Distinction

### I. Case $x = c$

In diesem Fall enthält  $yx^0z = yz$  das Teilwort  $ba$  einmal mehr als  $ab$ .

Somit gilt in diesem Fall  $yx^0z \notin L$ .

### II. Case $x$ enthält mindestens ein $a$ oder $b$

Wir betrachten  $yx^0z = yz$ . In diesem Fall bleibt die Anzahl der Teilwörter  $ba$  gleich oder erhöht sich. Da aber die Anzahl der Teilwörter  $ab$  um mindestens 1 kleiner wird, gilt  $yx^0z \notin L$ .

Da die Case Distinction alle Fälle abdeckt folgt für die Zerlegung  $yx^0z \notin L$ . Aus  $yxz \in L$  ergibt sich somit ein Widerspruch.

Demnach ist die Annahme falsch und  $L$  nicht regulär.

# Nichtdeterministische Endliche Automaten

---



Ein **nichtdeterministischer endlicher Automat (NEA)** ist ein Quintupel  $M = (Q, \Sigma, \delta, q_0, F)$ . Dabei ist

- (i)  $Q$  eine endliche Menge, **Zustandsmenge** genannt,
- (ii)  $\Sigma$  ein Alphabet, **Eingabealphabet** genannt,
- (iii)  $q_0 \in Q$  der **Anfangszustand**,
- (iv)  $F \subseteq Q$  die Menge der **akzeptierenden Zustände** und
- (v)  $\delta$  eine Funktion von  $Q \times \Sigma$  nach  $\mathcal{P}(Q)$ , **Übergangsfunktion** genannt.

Ein NEA kann zu einem Zustand  $q$  und einem gelesenen Zeichen  $a$  mehrere oder gar keinen Nachfolgezustand haben.

Eine **Konfiguration** von  $M$  ist ein Tupel  $(q, w) \in Q \times \Sigma^*$ .

- "M befindet sich in einer Konfiguration  $(q, w) \in Q \times \Sigma^*$ , wenn  $M$  im Zustand  $q$  ist und noch das Suffix  $w$  eines Eingabewortes lesen soll."

Ein **Schritt** von  $M$  ist eine Relation (auf Konfigurationen)  $\mid_M \subseteq (Q \times \Sigma^*) \times (Q \times \Sigma^*)$ , definiert durch

$$(q, w) \mid_M (p, x) \iff w = ax, a \in \Sigma \text{ und } p \in \delta(q, a)$$

Eine **Berechnung von M** ist eine endliche Folge  $C_1, \dots, C_k$  von Konfigurationen, so dass

$$C_i \xrightarrow{M} C_{i+1} \text{ für alle } 1 \leq i \leq k.$$

Eine **Berechnung von M auf x** ist eine Berechnung  $C = C_0, \dots, C_m$ , wobei  $C_0 = (q_0, x)$  und **entweder**  $C_m \in Q \times \{\lambda\}$  so dass  $\delta(q, a) = \emptyset$ .

Falls  $C_m \in F \times \{\lambda\}$ , sagen wir, dass  $C$  eine **akzeptierende Berechnung** von  $M$  auf  $x$  ist, und dass  $M$  **das Wort  $x$  akzeptiert**.

## Weitere Definitionen

Die Relation  $\mid_M^*$  ist die reflexive und transitive Hülle von  $\mid_M$ , genau wie bei einem EA.

Wir definieren

$$\mathbf{L}(\mathbf{M}) = \{w \in \Sigma^* \mid (q_0, w) \mid_M^* (p, \lambda) \text{ für ein } p \in F\}$$

als die **von M akzeptierte Sprache**.

Zu der Übergangsfunktion  $\delta$  definieren wir die Funktion  $\hat{\delta} : (Q \times \Sigma^*) \rightarrow \mathcal{P}(Q)$  wie folgt:

- (i)  $\hat{\delta}(q, \lambda) = \{q\}$  für alle  $q \in Q$
- (ii)  $\hat{\delta}(q, wa) = \bigcup_{r \in \hat{\delta}(q, w)} \delta(r, a)$  für alle  $q \in Q, a \in \Sigma, w \in \Sigma^*$ .

## Beispiel aus der Vorlesung

Wir betrachten folgenden NEA  $M = (\{q_0, q_1, q_2\}, \Sigma_{\text{bool}}, \delta, q_0, \{q_2\})$

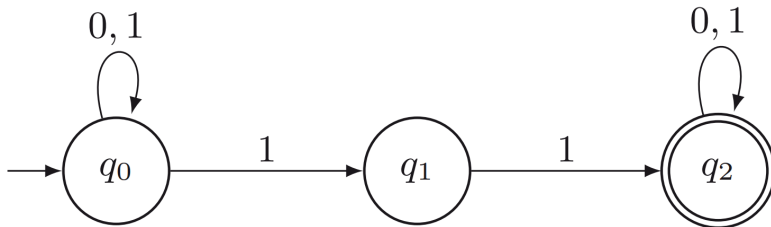


Abbildung 1: Abb. 3.15 aus dem Buch

# Berechnungsbaum

Für ein Wort  $x \in (\Sigma_{\text{bool}})^*$  ist ein Berechnungsbaum  $\mathcal{B}_M(x)$  nützlich, um zu erkennen, ob  $x \in L(M)$ .

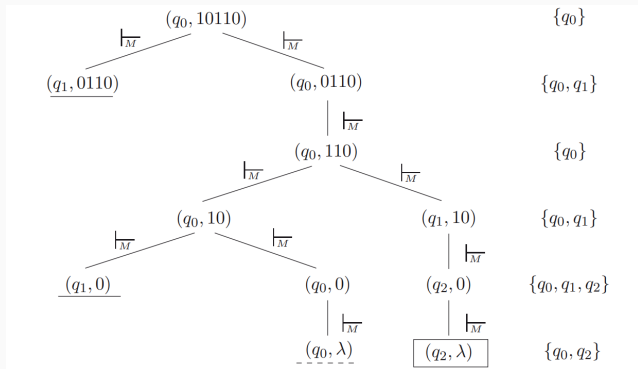


Abbildung 2: Abb. 3.16 aus dem Buch

## Sprache des NEA - Lemma 3.5

Wir können die Sprache des NEA bestimmen.

$$L(M) = \{x11y \mid x, y \in (\Sigma_{\text{bool}})^*\}$$

### Beweisidee

Beide Inklusionen zeigen und fertig. (Siehe Buch)

Wir definieren die Klasse  $\mathcal{L}_{\text{NEA}}$ .

$$\mathcal{L}_{\text{NEA}} = \{L(M) \mid M \text{ ist ein NEA}\}$$

# Äquivalenz von NEA und EA

Beweis von  $\mathcal{L}_{\text{NEA}} = \mathcal{L}_{\text{EA}}$  per **Potenzmengenkonstruktion**.

## Satz 3.2

Zu jedem NEA  $M$  existiert ein EA  $A$ , so dass

$$L(M) = L(A)$$

## Beweisidee

Potenzmengenkonstruktion und dann Induktion auf der Länge von einem Input i.e.  $|x|$ . (Siehe Buch)



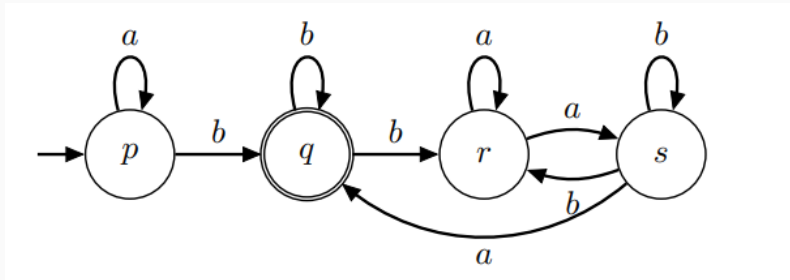
Sei  $M = (Q, \Sigma, \delta_M, q_0, F)$  ein NEA. Wir konstruieren einen äquivalenten Endlichen Automaten  $A = (Q_A, \Sigma_A, \delta_A, q_{0A}, F_A)$ .

- (i)  $Q_A = \{\langle P \rangle \mid P \subseteq Q\}$
- (ii)  $\Sigma_A = \Sigma$
- (iii)  $q_{0A} = \langle \{q_0\} \rangle$
- (iv)  $F_A = \{\langle P \rangle \mid P \subseteq Q \text{ und } P \cap F \neq \emptyset\}$
- (v)  $\delta_A : (Q_A \times \Sigma_A) \rightarrow Q_A$  ist eine Funktion, definiert wie folgt. Für jedes  $\langle P \rangle \in Q_A$  und jedes  $a \in \Sigma_A$  ist

$$\begin{aligned}\delta_A(\langle P \rangle, a) &= \left\langle \bigcup_{p \in P} \delta_M(p, a) \right\rangle \\ &= \langle \{q \in Q \mid \exists p \in P, \text{ so dass } q \in \delta_M(p, a)\} \rangle\end{aligned}$$

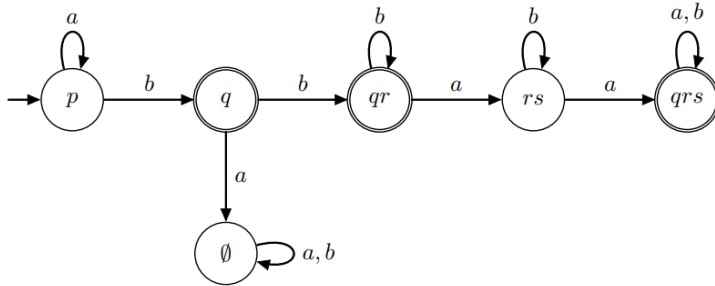
# Potenzmengenkonstruktion mit Beispiel NEA

Wenden wir Potenzmengenkonstruktion an:



# Potenzmengenkonstruktion mit Beispiel NEA

Lösung:



# Exponentiell mehr Zustände - manchmal

Sei

$$L_k = \{x1y \mid x \in (\Sigma_{\text{bool}})^*, y \in (\Sigma_{\text{bool}})^{k-1}\}$$

Folgender NEA  $A_k$  mit  $k + 1$  Zuständen akzeptiert  $L_k$ .

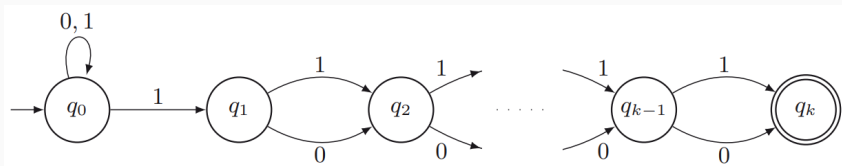


Abbildung 3: Abb. 3.19 im Buch

### Lemma 3.6

Für alle  $k \in \mathbb{N} \setminus \{0\}$  muss jeder EA, der  $L_k$  akzeptiert, mindestens  $2^k$  Zustände haben.

### Beweis

Sei  $B_k = (Q_k, \Sigma_{bool}, \delta_k, q_{0k}, F_k)$  ein EA mit  $L(B_k) = L_k$ .

Nach **Lemma 3.3** gilt für  $x, y \in (\Sigma_{bool})^*$ :

Wenn  $\hat{\delta}_k(q_{0k}, x) = \hat{\delta}_k(q_{0k}, y)$ , dann gilt für alle  $z \in (\Sigma_{bool})^*$ :

$$xz \in L(B_k) \iff yz \in L(B_k)$$

## Exponentiell mehr Zustände - manchmal

Die Idee des Beweises ist es, eine Menge  $S_k$  von Wörtern zu finden, so dass für keine zwei unterschiedlichen Wörter  $x, y \in S_k$  die Gleichung  $\hat{\delta}_k(q_{0k}, x) = \hat{\delta}_k(q_{0k}, y)$  gelten darf. Dann müsste  $B_k$  mindestens  $|S_k|$  viele Zustände haben.

Wir wählen  $S_k = (\Sigma_{bool})^k$  und zeigen, dass  $\hat{\delta}_k(q_{0k}, u)$  paarweise unterschiedliche Zustände für alle  $u \in S_k$  sind.

Wir beweisen dies per Widerspruch.

## Exponentiell mehr Zustände - manchmal

Seien  $x = x_1x_2\dots x_k$  und  $y = y_1y_2\dots y_k$  für  $x_i, y_i \in \Sigma_{bool}, i \in \{1, \dots, k\}$  zwei unterschiedliche Wörter aus  $S_k$ .

Nehmen wir zum Widerspruch an, dass  $\hat{\delta}_k(q_{0k}, x) = \hat{\delta}_k(q_{0k}, y)$ .

Weil  $x \neq y$ , existiert ein  $j \in \{1, \dots, k\}$ , so dass  $x_j \neq y_j$ . O.B.d.A. setzen wir  $x_j = 1$  und  $y_j = 0$ . Betrachten wir nun  $z = 0^{j-1}$ . Dann ist

$$xz = x_1\dots x_{j-1}1x_{j+1}\dots x_k0^{j-1} \text{ und } yz = y_1\dots y_{j-1}0y_{j+1}\dots y_k0^{j-1}$$

und daher  $xz \in L_k$  und  $yz \notin L_k$ .

Dies ist ein Widerspruch! Folglich gilt  $\hat{\delta}_k(q_{0k}, x) \neq \hat{\delta}_k(q_{0k}, y)$  für alle paarweise unterschiedliche  $x, y \in S_k = (\Sigma_{bool})^k$ .

Daher hat  $B_k$  mindestens  $|S_k| = 2^k$  viele Zustände.

## Mindestanzahl Zustände $n$ - Beweisschema

Die Grundidee ist es  $n$  Wörter anzugeben und zu beweisen, dass jedes von diesen  $n$  Wörtern in einem eigenen Zustand enden muss.

Seien  $w_1, \dots, w_n$  diese Wörter. Dann geben wir für jedes Paar von Wörtern  $w_i \neq w_j$  einen Suffix  $z_{i,j}$  an, so dass folgendes gilt:

$$w_i z_{i,j} \in L \not\leftrightarrow w_j z_{i,j} \in L$$

Dann folgt aus Lemma 3.3

$$\hat{\delta}(q_0, w_i) \neq \hat{\delta}(q_0, w_j)$$

Es eignet sich die Suffixe als Tabelle anzugeben.

Um die Wörter und Suffixe zu finden, kann es sich als nützlich erweisen, den Endlichen Automaten zu konstruieren.



## Mindestanzahl Zustände $n$ - Beweisschema

Wir nehmen zum Widerspruch an, dass es einen EA für  $L$  gibt mit weniger als  $n$  Zuständen.

Betrachten wir  $w_1, \dots, w_n$ . Per Pigeonhole-Principle existiert  $i < j$ , so dass

$$\hat{\delta}(q_0, w_i) = \hat{\delta}(q_0, w_j)$$

Per Lemma 3.3 folgt daraus, dass

$$\forall z \in \Sigma^* : w_i z \in L \iff w_j z \in L$$

Für  $z = z_{i,j}$  gilt aber per Tabelle

$$w_i z_{i,j} \in L \not\iff w_j z_{i,j} \in L \quad (1)$$

für alle  $i < j$ .

Da keines der  $n$  Wörter im gleichen Zustand enden kann: Widerspruch.

# Mindestanzahl Zustände $n$ - Beweisschema

Dann noch Angabe der Tabelle für (1)

	$w_2$	$\dots$	$w_n$
$w_1$	$z_{1,2}$	$\dots$	$z_{1,n}$
$\dots$		$\dots$	$\dots$
$w_{n-1}$			$z_{n-1,n}$

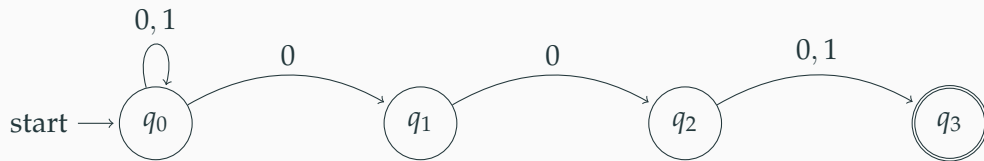
- Wenn es offensichtlich ist, muss (1) nicht bei jedem Suffix begründet werden.
- Ein minimaler endlicher Automat ist nicht notwendig für den Beweis. Hilft aber fürs
  - Finden der  $w_i$
  - Finden der  $z_{i,j}$
  - Beweis von  $w_i z_{i,j} \in L \not\iff w_j z_{i,j} \in L$  (Leicht überprüfbar)

## Klassische Aufgabe - HS19 Aufgabe 3.a

Wir betrachten die Sprache

$$L = \{x00y \mid x \in \{0,1\}^* \text{ und } y \in \{0,1\}\}$$

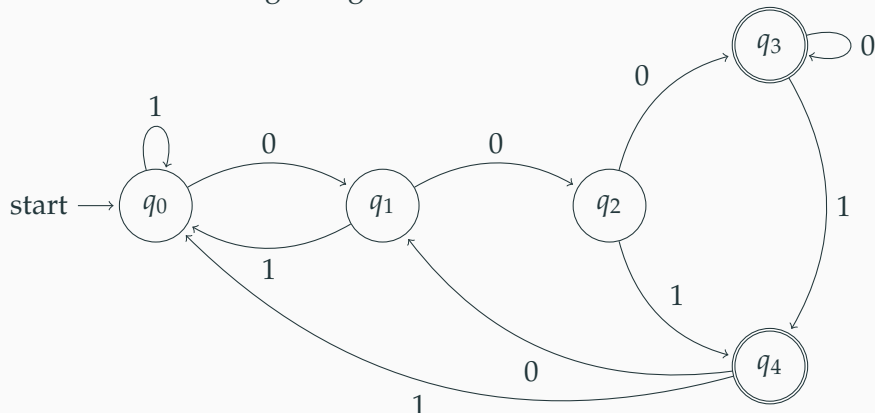
Konstruieren Sie einen nichtdeterministischen endlichen Automaten mit höchstens 4 Zuständen, der  $L$  akzeptiert.



## Klassische Aufgabe - HS19 Aufgabe 3.b

Zeigen Sie, dass jeder deterministische endliche Automat, der  $L$  akzeptiert, mindestens 5 Zustände braucht.

Wir zeichnen den zugehörigen EA zuerst.



## Klassische Aufgabe - HS19 Aufgabe 3.b

Nehmen wir zum Widerspruch an, dass es einen endlichen Automaten gibt, der  $L$  akzeptiert und weniger als 5 Zustände hat.

Wir wählen die Wörter  $B = \{\lambda, 0, 00, 000, 001\}$ .

Nach dem Pigeonhole-Principle existieren zwei Wörter  $w_i, w_j \in B, w_i \neq w_j$ , so dass

$$\hat{\delta}(q_0, w_i) = \hat{\delta}(q_0, w_j)$$

Per Lemma 3.3 folgt daraus, dass

$$\forall z \in \Sigma^* : w_i z \in L \iff w_j z \in L$$

## Klassische Aufgabe - HS19 Aufgabe 3.b

Wir betrachten folgende Tabelle mit Suffixen.

	0	00	000	001
$\lambda$	01	1	$\lambda$	$\lambda$
0		1	$\lambda$	$\lambda$
00			$\lambda$	$\lambda$
000				1

Der zeigt für jedes Wortpaar  $x, y \in B, x \neq y$  die Existenz eines Suffixes  $z$ , so dass

$$(xz \in L \wedge yz \notin L) \vee (xz \notin L \wedge yz \in L)$$

Dies kann man mit den angegebenen Suffixen und dem angegebenen EA einfach überprüfen.

Dies widerspricht der vorigen Aussage, dass ein Wortpaar  $w_i, w_j \in B, w_i \neq w_j$  existiert, so dass

$$\forall z \in \Sigma^* : w_i z \in L \iff w_j z \in L$$

Somit ist unsere Annahme falsch und es existiert kein EA mit  $< 5$  Zuständen für  $L$ .



Manchmal ist es zu schwierig einen minimalen EA zu finden und es funktioniert einfacher die Wörter durch Trial and Error zu finden. (Siehe Midterm HS22)



# Turing Maschinen

---

Formalisierung notwendig, um mathematisch über die automatische Unlösbarkeit zu argumentieren.

Jede vernünftige Programmiersprache ist eine zulässige Formalisierung.

Aber nicht geeignet (meistens komplexe Operationen).

Die Turingmaschine erlaubt ein paar **elementare Operationen** und besitzt trotzdem die **volle Berechnungsstärke** beliebiger Programmiersprachen.

Ziel dieses Kapitels ist, dass ihr ein gewisse Gespür dafür bekommt, was eine Turingmaschine kann und was nicht.

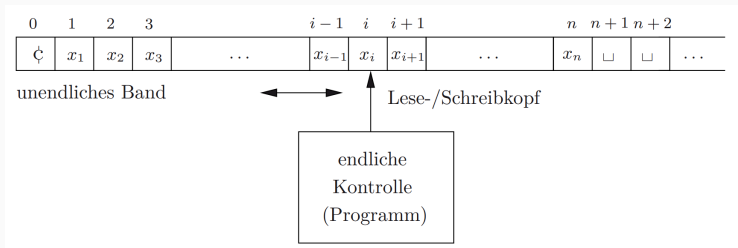
## Informell

Eine Turingmaschine besteht aus

- (i) einer endlichen Kontrolle, die das Programm enthält,

Für formale Beschreibung siehe Buch.

# Turing Maschinen - Formalisierung von Algorithmen



**Abbildung 4:** Abb. 4.1 vom Buch

## Elementare Operation einer TM - Informell

### Input

- Zustand der Maschine (der Kontrolle)
- Symbol auf dem Feld unter dem Lese-/Schreibkopf

### Aktion

- (i) ändert Zustand
- (ii) schreibt auf das Feld unter dem Lese-/Schreibkopf
- (iii) bewegt den Lese-/Schreibkopf nach links, rechts oder gar nicht. Ausser wenn  $\phi$ , dann ist links nicht möglich.

Eine **Konfiguration**  $C$  von  $M$  ist ein Element aus

$$\mathbf{Konf}(\mathbf{M}) = \{\mathfrak{c}\} \cdot \Gamma^* \cdot Q \cdot \Gamma^+ \cup Q \cdot \{\mathfrak{c}\} \cdot \Gamma^*$$

- Eine Konfiguration  $\clubsuit w_1 q a w_2$  mit  $w_1, w_2 \in \Gamma^*$ ,  $a \in \Gamma$  und  $q \in Q$  sagt uns:  
 $M$  im Zustand  $q$ , Inhalt des Bandes  $\clubsuit w_1 a w_2 \sqcup \dots$ , Kopf an Position  $|w_1| + 1$  und liest gerade  $a$ .

# Turing Maschinen - Formalisierung von Algorithmen

Es gibt wieder eine Schrittrelation  $\mid_M \subseteq \text{Konf}(M) \times \text{Konf}(M)$ .

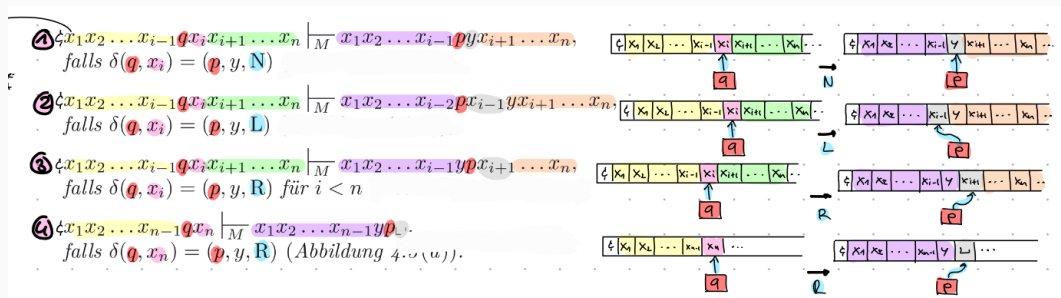


Abbildung 5: Diagramm von Adeline

Berechnung von  $M$ , Berechnung von  $M$  auf einer Eingabe  $x$  etc. durch  $\left| \overline{M} \right.$  definiert.

Die Berechnung von  $M$  auf  $x$  heisst

- **akzeptierend**, falls sie in einer akzeptierenden Konfiguration  $w_1 q_{\text{accept}} w_2$  endet (wobei  $\$$  in  $w_1$  enthalten ist).
- **verwerfend**, wenn sie in in einer verwerfenden Konfiguration  $w_1 q_{\text{reject}} w_2$  endet.
- **nicht-akzeptierend**, wenn sie entweder eine **verwerfende** oder unendliche Berechnung ist.



Die von der Turingmaschine **M** akzeptierte Sprache ist

$$\mathbf{L(M)} = \{w \in \Sigma^* \mid q_0 \dot{\vdash} w \mid_M^* y q_{\text{accept}} z, \text{ für irgendwelche } y, z \in \Gamma^*\}$$

## Reguläre Sprachen

$$\mathcal{L}_{\text{EA}} = \{L(A) \mid A \text{ ist ein EA}\} = \mathcal{L}_{\text{NEA}}$$

## Rekursiv aufzählbare Sprachen

Eine Sprache  $L \subseteq \Sigma^*$  heisst **rekursiv aufzählbar**, falls eine TM  $M$  existiert, so dass  $L = L(M)$ .

$$\mathcal{L}_{\text{RE}} = \{L(M) \mid M \text{ ist eine TM}\}$$

ist die **Klasse aller rekursiv aufzählbaren Sprachen**.

## Halten

Wir sagen das  $M$  **immer hält**, wenn für alle Eingaben  $x \in \Sigma^*$

- (i)  $q_0 \dot{\vdash} x \mid_M^* y q_{\text{accept}} z, y, z \in \Gamma^*$ , falls  $x \in L$  und
- (ii)  $q_0 \dot{\vdash} x \mid_M^* u q_{\text{reject}} v, u, v \in \Gamma^*$ , falls  $x \notin L$ .

## Rekursive Sprachen

Eine Sprache  $L \subseteq \Sigma^*$  heisst **rekursiv (entscheidbar)**, falls  $L = L(M)$  für eine TM  $M$ , die **immer hält**.

$$\mathcal{L}_R = \{L(M) \mid M \text{ ist eine TM, die immer hält}\}$$

ist die **Klasse der rekursiven (algorithmisch erkennbaren) Sprachen**.

## Mehrband-TM - Informelle Beschreibung

Für  $k \in \mathbb{N} \setminus \{0\}$  hat eine  $k$ -Band Turingmaschine

- eine endliche Kontrolle
- ein endliches Band mit einem Lesekopf (Eingabeband)
- $k$  Arbeitsbänder, jedes mit eigenem Lese-/Schreibkopf (nach rechts unendlich)

**Insbesondere gilt 1-Band TM  $\neq$  "normale" TM**

Am Anfang der Berechnung einer MTM  $M$  auf  $w$

- Arbeitsbänder "leer" und die  $k$  Lese-/Schreibköpfe auf Position 0.
- Inhalt des Eingabebands  $\$w\$$  und Lesekopf auf Position 0.
- Endliche Kontrolle im Zustand  $q_0$ .

# Äquivalenz von Maschinen (TM, MTM)

Seien  $A$  und  $B$  zwei Maschinen mit **gleichem**  $\Sigma$ .

Wir sagen, dass  **$A$  äquivalent zu  $B$  ist**, wenn für jede Eingabe  $x \in \Sigma^*$

- (i)  $A$  akzeptiert  $x \iff B$  akzeptiert  $x$
- (ii)  $A$  verwirft  $x \iff B$  verwirft  $x$
- (iii)  $A$  arbeitet unendlich lange auf  $x \iff B$  arbeitet unendlich lange auf  $x$

Wir haben

$$A \text{ und } B \text{ äquivalent} \implies L(A) = L(B)$$

aber

$$L(A) = L(B) \not\Rightarrow A \text{ und } B \text{ äquivalent}$$

da  $A$  auf  $x$  unendlich lange arbeiten könnte, während  $B$   $x$  verwirft.

# Äquivalenz von 1-Band TM zu TM

## Lemma 4.1

Zu jeder TM  $A$  existiert eine zu  $A$  äquivalente 1-Band-TM  $B$

## Beweisidee

$B$  kopiert die Eingabe zuerst aufs Arbeitsband und simuliert dann  $A$ .

## Lemma 4.2

Zu jeder Mehrband-TM  $A$  existiert eine zu  $A$  äquivalente TM  $B$

**Beweis**

postponed

# Äquivalenz Folgerung

Aus Lemma 4.1 und 4.2 folgt direkt

## Satz 4.1

Die Maschinenmodelle von Turingmaschinen und Mehrband-Turingmaschinen sind äquivalent.

Note:

“Äquivalenz” für Maschinenmodelle wird in Definition 4.2 definiert.

Maschinenmodelle sind Klassen von Maschinen (i.e. Mengen von Maschinen mit gewissen Eigenschaften).



# Midterm Prep

---