# Theoretische Informatik

Lucien Perret, Jil Zerndt
May 2024

## Alphabete, Wörter, Sprachen

**Alphabete** sind endliche, nichtleere Mengen von Symbolen.
- $\Sigma = \{a, b, c\}$ Mengen von drei Symbolen
- $\Sigma_{\text{Bool}} = \{0, 1\}$ Boolsches Alphabet

**Keine Alphabete**
- $\mathbb{N}, \mathbb{R}, \mathbb{Z}$ usw. (unendliche Mächtigkeit)

**Wort** ist eine endliche Folge von Symbolen eines bestimmten Alphabets.
- $abc$ Wort über dem Alphabet $\Sigma_{\text{lat}}$ (oder über $\Sigma = \{a, b, c\}$ )
- 100111 Wort über dem Alphabet $\{0, 1\}$
- $\varepsilon$ Leeres Wort (über jedem Alphabet)

**Schreibweisen** $|\omega|$ = **Länge eines Wortes**
- $|100111| = 6$
- $|\varepsilon| = 0$

$|\omega|_x$ = **Häufigkeit eines Symbols** $x$ in einem Wort
- $|100111|_1 = 4$
- $|\varepsilon|_0 = 0$
- $|\varepsilon|_\varepsilon = 1$

$\omega^R$ = **Spiegelwort/Reflection** zu $\omega$
- $(abc)^R = cba$
- $(100111)^R = 111001$
- $\varepsilon^R = \varepsilon$

**Teilwort (Infix)** $v$ ist ein Teilwort (Infix) von $\omega$ ist, wenn man $\omega$ als $\omega = xvy$.

$$\omega \neq v \rightarrow \text{ Echtes Teilwort}$$

- Teilwörter von $abba$ $\varepsilon, a, b, ab, abb, bb, bba, abba, ba$
- Präfixe von $abba$ $\varepsilon, a, ab, abb, abba$
- Suffixe von $abba$ $\varepsilon, a, ba, bba, abba$

**Mengen von Wörtern** $\Sigma^k$ = Die Menge aller Wörter der Länge $\boldsymbol{k}$ über einem Alphabet $\Sigma$
- $\Sigma = \{a, b, c\}$ $\Sigma^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$
- $\Sigma = \{0, 1\}$ $\{0, 1\}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$
- $\Sigma^0 = \{\varepsilon\}$
- $\Sigma^* = \underbrace{\Sigma^0}_{1} \cup \underbrace{\Sigma^1} \cup \underbrace{\Sigma^2} \cup \underbrace{\Sigma^3} \cdots$ Kleensche Hülle
- $\Sigma^+ = \underbrace{\Sigma^1}_{2} \cup \underbrace{\sum^2}_{4} \cup \underbrace{\Sigma^3}_{8} \ldots = \Sigma^* \backslash \{\varepsilon\}$ Positive Hülle

**Konkatenation** = Verkettung von zwei beliebigen Wörtern $x$ und $y$

$$x \circ y = xy := (x_1, x_2 \ldots x_n, y_1, y_2 \ldots y_m)$$

**Wortpotenzen** Sei $x$ ein Wort über einem Alphabet $\Sigma$.
- $x^0 = \varepsilon$
- $x^{n+1} = x^n \circ x = x^n x$
- bbabababababbaaaabab $= b^2(ab)^4ba^3(ab)^2$

**Sprache** über einem Alphabet $\Sigma$ = Eine Teilmenge $L \subseteq \Sigma^*$ von Wörtern.
- $\Sigma_1 \subseteq \Sigma_2 \wedge L$ Sprache über $\Sigma_1 \rightarrow L$ Sprache über $\Sigma_2$
- $\Sigma^*$ Sprache über jedem Alphabet $\Sigma$
- $\{\} = \emptyset$ ist die leere Sprache

**Konkatenation** von zwei Sprachen $A \subset \Sigma^*$ und $B \subset \Gamma^*$

$$AB = \{uv \mid u \in A \text{ und } v \in B\}$$

Die **Kleenesche Hülle** $A^*$ einer Sprache $A = \{\varepsilon\} \cup A \cup AA \cup AAA \cup \ldots$

## Reguläre Ausdrücke

**Reguläre Ausdrücke** sind Wörter, die Sprachen beschreiben.
Die Sprache $RA_\Sigma$ der Regulären Ausdrücke über einem Alphabet $\Sigma$ ist wie folgt definiert:
- $\emptyset, \epsilon \in RA_\Sigma$
- $\Sigma \subset RA_\Sigma$
- $R \in RA_\Sigma \Rightarrow (R^*) \in RA_\Sigma$
- $R, S \in RA_\Sigma \Rightarrow (RS) \in RA_\Sigma$
- $R, S \in RA_\Sigma \Rightarrow (R \mid S) \in RA_\Sigma$

Für jeden regulären Ausdruck $R \in RA_\Sigma$ definieren wir die Sprache $L(R)$ von $R$ wie folgt:
- Leere Sprache: $L(\emptyset) = \emptyset$
- Sprache, die nur das leere Wort enthält: $L(\varepsilon) = \{\varepsilon\}$
- Beschreibt die Sprache $\{a\}$: $L(a) = \{a\}$ $\forall a \in \Sigma$
- Kombiniert die Wörter von R: $L(R^*) = L(R)^*$
- Verkettung von Wörtern (R = prefix): $L(RS) = L(R) \circ L(S)$
- Wörter die in R oder S beschrieben werden: $L(R \mid S) = L(R) \cup L(S)$

**Reguläre Sprache**
Eine Sprache $A$ über dem Alphabet $\Sigma$ heisst regulär, falls
- $A = L(R)$ für einen regulären Ausdruck $R \in RA_\Sigma$ gilt.

Beispiele
- $R_1 = a^*b$ $L(R_1) = \{b, ab, aab, aaab, \ldots\}$
- $R_2 = (aa)^*b^*aba$ $L(R_2) = \{aba, baba, aaaba, aababa, \ldots\}$
- $R_3 = (a \mid ab)^*$ $L(R_3) = \{\varepsilon, a, ab, aa, abab, \ldots\}$

$L(R_1)$ : Menge der ganzen Zahlen in Dezimaldarstellung
- $((- \mid \varepsilon)(1, 2, 3, 4, 5, 6, 7, 8, 9)(0, 1, 2, 3, 4, 5, 6, 7, 8, 9) \mid 0).0$

**Eigenschaften und Konventionen** Die Menge $RA_\Sigma$ über dem Alphabet $\Sigma$ ist eine Sprache über dem Alphabet

$$\{\emptyset, \epsilon, *, (), , |\} \cup \Sigma$$

**Priorisierung von Operatoren**
- (1) $*$ = Wiederholung $\rightarrow$ (2) Konkatenation $\rightarrow$ (3) $|$= Oder

**Beispiele**
- $(aa)^*b^*aba = (aa)^*b^*aba$
- $(ab)|(ba) = ab|ba$
- $a(b(ba))|b = abba|b$

**Erweiterte Syntax**
- $R^+ = R(R^*)$
- $R? = (R \mid \epsilon)$
- $[R_1, \ldots, R_k] = R_1 \mid R_2 \mid \ldots \mid R_k$

## Endliche Automaten

**Endliche Automaten** entsprechen Maschinen, die Entscheidungsprobleme lösen.
- Links nach rechts
- Keinen Speicher
- Keine Variablen
- Speichert aktuellen Zustand
- Ausgabe über akzeptierende Zustände

**DEA** Ein deterministischer endlicher Automat (DEA) ist ein 5-Tupel $M = (Q, \Sigma, \delta, q_0, F)$
- $Q$ endliche Menge von Zuständen
- $\Sigma$ endliches Eingabealphabet
- $\delta : Q \times \Sigma \rightarrow Q$ Übergangsfunktion
- $q_0 \in Q$ Startzustand
- $F \subseteq Q$ Menge der akzeptierenden Zustände

**DEA Funktionen**
$M = (Q, \Sigma, \delta, q_0, F)$ ein EA. **Konfiguration** von $M$ auf $\omega$ ist ein Element aus $Q \times \Sigma^*$.
- Startkonfiguration von $M$ auf $\omega$ $\{q_0, \omega\} \in \{q_0\} \times \Sigma^*$
- Endkonfiguration $(q_n, \varepsilon)$

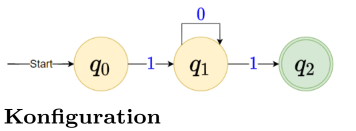
**Berechnungsschritt** $\vdash_M$ von $M$

$$(q, \omega) \vdash_M (p, x)$$

**Berechnung** ist eine endliche Folge von Berechnungsschritten

$$(q_a, \omega_1\omega_2 \ldots \omega_n) \vdash_M \ldots \vdash_M \big(q_e, \omega_j \ldots \omega_n\big) \rightarrow (q_a, \omega_1\omega_2 \ldots \omega_n) \vdash_M^* \big(q_e, \omega$$

**Beispiel DEA (eindeutig)**
- Sprache: $L(M) = \big\{1x1 \mid x \in \{0\}^*\big\}$

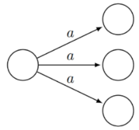



**Konfiguration**
- Startkonfiguration auf $\omega = 101 \rightarrow (q_0, 101)$
- Endkonfiguration auf $\omega = 101 \rightarrow (q_2, \varepsilon)$

**Berechnung**
- $\omega = 101 \rightarrow (q_0, 101) \vdash_M (q_1, 01) \vdash_M (q_1, 1) \vdash_M (q_2, \varepsilon) \rightarrow$ akzeptierend
- $\omega = 10 \rightarrow (q_0, 10) \vdash_M (q_1, 0) \vdash_M (q_1, \varepsilon) \rightarrow$ verwerfend

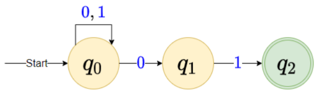
**Nichtdeterministischer endlicher Automat (NEA)**
Der einzige Unterschied zum DEA besteht in der Übergangsfunktion $\delta$
- Übergangsfunktion $\delta : Q \times \Sigma \rightarrow P(Q)$

Ein $\varepsilon$-NEA erlaubt zusätzlich noch $\varepsilon$-Übergänge.

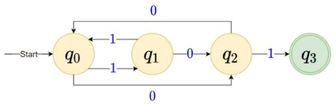


### NEA (nicht eindeutig)

- Sprache: $L(M) = \left\{x01 \mid x \in \{0,1\}^*\right\}$




- Äquivalenter NEA

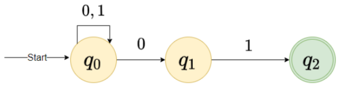


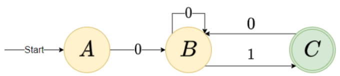

### Teilmengenkonstruktion

Jeder NEA kann in einen DEA umgewandelt werden. (gleichmächtig)

1. $Q_{NEA} \to P\left(Q_{NEA}\right) = Q_{DEA}$ (Potenzmenge)
2. Verbinden mit Vereinigung aller möglichen Zielzustände
3. Nicht erreichbare Zustände eliminieren
4. Enthält akzeptierenden Zustand $= F_{NEA} \to$ akzeptierend

| ↓ | $q$ | $\delta(q,0)$ | $\delta(q,1)$ |
|---|---|---|---|
| 0 | ~~$\emptyset$~~ | ~~$\emptyset$~~ | ~~$\emptyset$~~ |
| | $A = \{q_0\}$ | $\{q_0, q_1\}$ | $\{q_0\}$ |
| 1 | ~~$\{q_1\}$~~ | ~~$\emptyset$~~ | ~~$\{q_2\}$~~ |
| 4 | ~~$\{q_2\}$~~ | ~~$\emptyset$~~ | ~~$\emptyset$~~ |
| | $B = \{q_0, q_1\}$ | $\{q_0, q_1\}$ | $\{q_0, q_2\}$ |
| | $C = \{q_0, q_2\}$ | $\{q_0, q_1\}$ | $\{q_0\}$ |
| 2 | ~~$\{q_1, q_2\}$~~ | ~~$\emptyset$~~ | ~~$\{q_2\}$~~ |
| 3 | ~~$\{q_0, q_1, q_2\}$~~ | ~~$\{q_0, q_1\}$~~ | ~~$\{q_0, q_2\}$~~ |

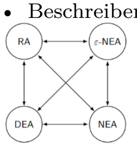



## Reguläre Sprachen

### Äquivalente Mechanismen

- Akzeptierender Mechanismus DEA, NEA, $\varepsilon$-NEA
- Beschreibender Mechanismus RA




**Äquivalenz DEA und RA**

- Es gibt einen DEA, der die Sprache $L$ akzeptiert
- Es gibt einen RA, der die Sprache $L$ akzeptiert.

### Abschlusseigenschaften regulärer Sprachen

Seien $L_1$ und $L_2$ zwei reguläre Sprachen über $\Sigma$. Dann ist die Vereinigung ... regulär.

$$L_1 \cup L_2 = \left\{\omega \mid \omega \in L_1 \vee \omega \in L_2\right\}$$

Sei $L$ eine reguläre Sprache über $\Sigma$. Dann ist auch das Komplement ... regulär.

$$\bar{L} = \Sigma^* - L = \left\{\omega \in \Sigma^* \mid \omega \notin L\right\}$$

- Schnitt: $L_1 \cap L_2 = \left\{\omega \mid \omega \in L_1 \wedge \omega \in L_2\right\}$
- Differenz: $L_1 - L_2 = \left\{\omega \mid \omega \in L_1 \wedge \omega \notin L_2\right\}$
- Konkatenation: $L_1 \cdot L_2 = L_1 L_2 = \left\{\omega = \omega_1 \omega_2 \mid \omega_1 \in L_1 \wedge \omega_1 \in L_2\right\}$
- Kleenesche Hülle: $L^* = \left\{\omega = \omega_1 \omega_2 \ldots \omega_n \mid \omega_i \in L \text{ für alle } i \in \{1, 2, \ldots, n\}\right\}$

### Zustandsklasse

Jedes Wort landet in einem Zustand

$$\Sigma^* = \bigcup_{p \in Q} [p]$$

Kein Wort landet nach dem Lesen in zwei Zuständen

$$[p] \cap [q] = \emptyset, \text{ für alle } p \neq q, p, q \in Q$$

Nach dem Lesen von $\omega$ landet man im Zustand $p$.

$$\text{Klasse } [q_0] = \left\{\omega \in \{0,1\}^* \middle| |\omega|_0 \bmod (3) = 1\right\}$$

Von $M$ akzeptierte Sprache

$$L(M) = \bigcup_{p \in F} [p]$$

$$L(M) = \left\{\forall \omega \in \{0,1\}^* \middle| |\omega|_0 \bmod (3) = 1\right\}$$

## Kontextfreie Grammatiken

### Kontextfreie Grammatik

Eine KontextFreie Grammatik $G(KFG)$ ist ein 4-Tupel $(N, \Sigma, P, A)$ mit

- $N$ ist das Alphabet der Nichtterminale (Variablen)
- $\Sigma$ ist das Alphabet der Terminale
- $P$ ist eine endliche Menge von Produktionen mit der Form $X \to \beta$

Mit Kopf $X \in N$ und Rumpf $\beta \in (N \cup \Sigma)^*$

- $A$ ist das Startsymbol, wobei $A \in N$

Ein Wort $\beta \in (N \cup \Sigma)^*$ nennen wir Satzform.

Seien $\alpha, \beta$ und $\gamma$ Satzformen und $A \to \gamma$ eine Produktion.

- Ableitungsschritt mit Produktion $A \to \gamma \quad \alpha A \beta \to \alpha \gamma \beta$
- Ableitung Folge von Ableitungsschritten $\alpha \to \cdots \to \omega$

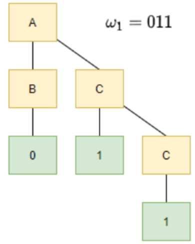
### Ableitungsbaum

Eine Ableitung kann als Ableitungsbaum / Parsebaum dargestellt werden. KGF $G_1$ für die Sprache $\left\{0^n 1^m \mid n, m \in N\right\}$

- $G_1 = \{\{A, B, C\}, \{0, 1\}, P, A\}$
- $P = \{A \to BC, B \to 0B|0|\varepsilon, C \to 1C|1|\varepsilon\}$

Ableitung von $\omega_1 = 011$

- $A \to BC \to 0AA \to 01C \to 011 \to \ldots \to 011$




### Mehrdeutigkeit

Eine KFG nennen wir mehrdeutig, wenn es ein Wort gibt, das mehrere Ableitungsbäume besitzt.

**Mehrdeutigkeiten eliminieren:**

- Korrekte Klammerung vom Benutzer erzwingen
- Grammatik anpassen
- Den Produktionen einen Vorrang vergeben

### KFG für Sprache L

Jede reguläre Sprache kann durch eine kontextfreie Grammatik beschrieben werden. Sei $L$ eine reguläre Sprache. Dann gibt es einen DEA $M = (Q, \Sigma, \delta, q_0, F)$ mit $L(M) = L$

Dann können wir einen KFG für $L$ wie folgt bauen:

- Für jeden Zustand $q_i$ gibt es ein Nichtterminal $Q_i$
- Für jede Transition $\delta(q_i, a) = q_j$ erstellen wir die Produktion $Q_i \to aQ_j$
- Für jeden akzeptierenden Zustand $q_i \in F$ erstellen wir die Produktion $Q_i \to \varepsilon$
- Das Nichtterminal $Q_0$ wird zum Startsymbol $A$.

## Kellerautomaten

**Kellerautomaten** haben einen «Speicher». PDA = Push Down Automat.

Ein deterministischer Kellerautomat KA ist ein 7-Tupel

$$M = \left(Q, \Sigma, \boldsymbol{\Gamma}, \boldsymbol{\delta}, q_0, \$, F\right)$$

- Menge von Zuständen: $Q$
- Alphabet der Eingabe: $\Sigma$
- Alphabet des Kellers: $\Gamma$
- Übergangsfunktion: $\boldsymbol{\delta} : \boldsymbol{Q} \times (\boldsymbol{\Sigma} \cup \boldsymbol{\varepsilon}) \times \boldsymbol{\Gamma} \to \boldsymbol{Q} \times \Gamma^*$
- Anfangszustand: $q_0 \in Q$
- Symbol vom Alphabet des Kellers: $\$ \in \Gamma$
- Akzeptierende Zustände: $F \subseteq Q$

## Zusätzliche Einschränkungen für DKAs

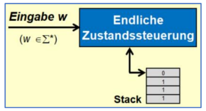
Für jeden Zustand $q$ und alle Symbole $x, b$ gilt, wenn $\delta(q, b, c)$ definiert ist, dann ist $\delta(q, \varepsilon, x)$ undefiniert.
Ein Übergang $\delta(q, b, c) = (p, \omega)$ wird graphisch dargestellt

$$q - b, c/\omega \longrightarrow p$$

## Berechnungsschritte

Ein Berechnungsschritt $\delta(q, b, c) = (p, \omega)$ wird wie folgt interpretiert

- $q$ = Aktueller Zustand
- $b$ = Symbol der Eingabe
- c = Symbol wird entfernt
- $\omega$ = Wort auf Stack geschrieben
- $p$ = Neuer Zustand




## Sprache eines Kellerautomaten

Die Sprache $L(M)$ des Kellerautomaten $M$ ist definiert durch

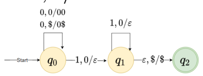
$$L(M) = \left\{\omega \in \Sigma^* \mid \left(q_0, \omega, \$\right) \vdash^* (q, \varepsilon, \gamma) \text{ für ein } q \in F \text{ und ein } \gamma \in \Gamma^*\right\}$$

Elemente von $L(M)$ werden von $M$ akzeptierte Wörter genannt.

## Kellerautomat für eine Sprache erstellen

Ein Kellerautomat für die kontextfreie Sprache $\left\{0^n 1^n \mid n > 0\right\}$

- $0, 0/00$ Read 0 Add 0 $(00 - 0) = 0$
- $0, \$/0\$$ Read 0 Add 0 $(\$0 - \$) = 0$
- $1, 0/\varepsilon$ Read 1 Remove 0 Read $(\varepsilon - 0) = -0$
- $\varepsilon, \$/\$$ Read $\varepsilon$ - $(\$ - \$) = \varepsilon$



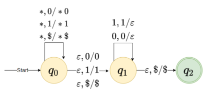

- $\omega_1 = 011 : \left(q_0, 011, \$\right) \vdash \left(q_1, 11, 0\$\right) \vdash \left(q_1, 1, \$\right) \rightarrow \omega_1$ verwerfend

Das Zeichen \$ zeigt an, dass der «Stack» leer ist.

## NKA: Übergangsfunktion

- $\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \rightarrow P(Q \times \Gamma *)$

Kellerautomat für die Sprache $\left\{\omega\omega^R \mid \omega \in \{0, 1\}^*\right\}$




# Turingmaschinen

## Turingmaschinen (TM)

- Einen Lese- / Schreib-Kopf
- Ein unendliches Band von Zellen

Eine deterministischer Turing-Maschine TM ist ein 7-Tupel

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

- Menge von Zuständen: $Q$
- Alphabet der Eingabe: $\Sigma$
- Bandalphabet: $\Gamma$ und $\Sigma \subset \Gamma$
- Übergangsfunktion: $\boldsymbol{\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times D, D = \{L, R\}}$
- Anfangszustand: $q_0 \in Q$
- Akzeptierende Zustände: $F \subseteq Q$
- Leerzeichen $\sqcup$, mit $_{\mu} \in \Gamma$ und $_{\mu} \notin \Sigma$

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

- $q, p \in Q$ und $X, Y \in \Gamma$
- $D$ = Direction
- $X$ = Read
- $Y$ = Overwrite

$$q - X/Y, D \rightarrow p$$

## Band

- Unterteilt in einzelne Zellen mit jeweils einem beliebigen Symbol
- Beinhaltet zu Beginn die Eingabe, d.h. ein endliches Wort aus $\Sigma^*$. Alle anderen Zellen enthalten das besondere Symbol 4 .

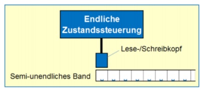
## Konfiguration

einer Turing-Maschine $M$ ist durch die folgenden Angaben eindeutig spezifiziert

- Zustand der Zustandssteuerung
- Position des Lese- / Schreibkopfes
- Bandinhalt

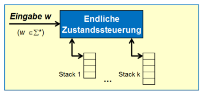
## Semi-Unendliches Band

Das Band der Turingmaschine ist nur in eine Richtung unendlich.
Jede Sprache $L$ die von einer TM $T$ akzeptiert wird, wird auch von einer TM mit semi-unendlichem Band akzeptiert.



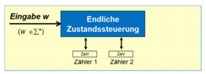

## Mehrere Stacks

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



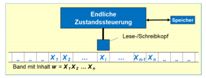

## Zähler-Maschinen

Eine Zähler-Maschine (Counter Machine) mit $k$ Zählern entspricht einer $k$ Stack-Maschine mit dem Unterschied, dass die Stacks durch einfache Zähler ersetzt werden.
Jede Sprache $L$ die von einer TM $T$ akzeptiert wird, wird auch von einer 2Zähler-Maschine $Z$ mit 2 Zählern akzeptiert.



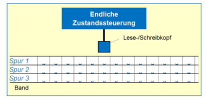

## TM mit Speicher

In der endlichen Zustandssteuerung einer TM können ausser dem SteuerZustand zusätzlich endlich viele Daten-Zustände gespeichert werden.



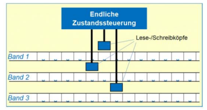

## Mehrere Spuren

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



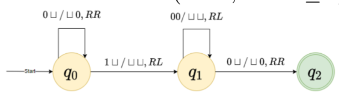

## Mehrere Bänder

- TM mit endlich vielen Bändern und Lese- / Schreibköpfen
- Jeder Lese- / Schreibkopf kann unabhängig auf ein Band zugreifen




## Mehrband-Maschine

Spezifizieren Sie eine TM $M_4$, welche die Subtraktion von zwei natürlichen Zahlen ($a - b$, mit $a \geq b$) realisiert.




Beispiel: $4 - 2 = 2$

| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $q_0 0000100 \vdash$ | $0\sqcup/\sqcup 0, RR$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | | |
| 2 | $q_0 \sqcup \vdash$ | | | | | | | | | | |
| 1 | $\sqcup q_0 000100 \vdash$ | $0\sqcup/\sqcup 0, RR$ | | 0 | 0 | 0 | 1 | 0 | 0 | | |
| 2 | $0q_0 \sqcup \vdash$ | | 0 | | | | | | | | |
| 1 | $\sqcup\sqcup q_0 00100 \vdash$ | $0\sqcup/\sqcup 0, RR$ | | | 0 | 0 | 1 | 0 | 0 | | |
| 2 | $00q_0 \sqcup \vdash$ | | 0 | 0 | | | | | | | |
| 1 | $\sqcup\sqcup\sqcup q_0 0100 \vdash$ | $0\sqcup/\sqcup 0, RR$ | | | | 0 | 1 | 0 | 0 | | |
| 2 | $000q_0 \sqcup \vdash$ | | 0 | 0 | 0 | | | | | | |
| 1 | $\sqcup\sqcup\sqcup\sqcup q_0 100 \vdash$ | $1\sqcup/\sqcup\sqcup, RL$ | | | | | 1 | 0 | 0 | | |
| 2 | $0000q_0 \sqcup \vdash$ | | 0 | 0 | 0 | 0 | | | | | |
| 1 | $\sqcup\sqcup\sqcup\sqcup\sqcup q_1 00 \vdash$ | $00/\sqcup\sqcup, RL$ | | | | | | 0 | 0 | | |
| 2 | $000q_1 0 \vdash$ | | 0 | 0 | 0 | 0 | | | | | |
| 1 | $\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup q_1 0 \vdash$ | $00/\sqcup\sqcup, RL$ | | | | | | | 0 | | |
| 2 | $00q_1 0 \vdash$ | | 0 | 0 | 0 | | | | | | |
| 1 | $\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup q_1$ | $\sqcup 0/\sqcup 0, RR$ | | | | | | | | | |
| 2 | $0q_1 0 \vdash$ | | 0 | 0 | | | | | | | |
| 1 | $\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup q_2 \sqcup$ | | | | | | | | | | |
| 2 | $00q_2 \sqcup \vdash$ | | 0 | 0 | | | | | | | |

## Berechnungsmodelle

### Turing-berechenbar

Jedes algorithmisch lösbare Berechnungsproblem kann von einer Turing-Maschine gelöst werden.

- Computer und Turing-Maschinen sind äquivalent.

Turing-berechenbare Funktion: Turing-Maschine $T = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$

$$T : \Sigma^* \to \delta^*$$

$$T(\omega) = \begin{cases} u & \text{falls T auf } \omega \in \Sigma^* \text{ angesetzt, nach endlich vielen} \\ & \text{Schritten mit u auf dem Band anhält} \\ \uparrow & \text{falls T bei Input } \omega \in \Sigma^* \text{ nicht hält} \end{cases}$$

### Primitiv rekursive Grundfunktionen

Für jedes $n \in \mathbb{N}$ und jede Konstante $k \in \mathbb{N}$ die n-stellige konstante Funktion:

$$c_k^n = \mathbb{N}^n \to \mathbb{N} \text{ mit } c_k^n(x_1, ..., x_n) = k$$

Nachfolgerfunktion:

$$\eta : \mathbb{N} \to \mathbb{N} \text{ mit } \eta(x) = x + 1$$

Für jedes $n \in \mathbb{N}$ und jedes $1 < k < n$ die n-stellige Projektion auf die k-te Komponente:

$$\pi_k^n : \mathbb{N}^n \to \mathbb{N} \text{ mit } \pi_k^n(x_1, ..., x_k, ..., x_n) = k$$

n = Anzahl der Argumente, k = Position des Arguments

### Loop (primitiv-rekursiv)

- Zuweisungen: $x = y + c$ und $x = y - c$
- Sequenzen: $P$ und $Q \to P; Q$
- Schleifen: $P \to$ Loop $x$ do $P$ until End

### Addition von natürlichen Zahlen

Add(x, y) = x + y

```
LOOP x1 DO
    x2 = x2 + 1
END
x0 = x2 + 0
```

### While (Turing vollständig)

Erweiterung deer Sprache Loop

- While $x_i > 0$ do ... until End

### Multiplikation von natürlichen Zahlen

Mul(x, y) = x * y

```
WHILE x1 > 0 DO
    x1 = x1 - 1
    LOOP x2 DO
        x0 = x0 + 1
    END
END
```

### GoTo (Turing vollständig)

- Zuweisungen: $x_i = x_j + c$ und $x_i = x_j - c$
- Sprunganweisung: IF $x_i = c$ THEN GOTO $L_k$ ELSE GOTO $L_t$
  - or simple: GOTO $L_k$
- Schleifen: WHILE $x_i > 0$ DO ... HALT

### Case distinction

```
M1: x0 = x3 + 0
M2: IF x1 = 0 THEN GOTO M4
M3: x0 = x2 +0
M4: HALT
```

## Entscheidbarkeit

### Entscheidbarkeit

- Ein Problem ist entscheidbar, wenn es einen Algorithmus gibt, der für jede Eingabe eine Antwort liefert.
- Ein Problem ist semi-entscheidbar, wenn es einen Algorithmus gibt, der für jede Eingabe eine Antwort liefert, falls die Antwort ja ist.

Eine Sprache $A \subset \Sigma^*$ ist genau dann entscheidbar, wenn sowohl $A$ als auch $\bar{A}$ semi-entscheidbar ist.

- $\bar{A}$ steht für das Komplement von $A$ in $\Sigma^*$ : $\bar{A} = \Sigma^* \backslash A = \{\omega \in \Sigma^* \mid \omega \notin A\}$

### Entscheidbarkeit und Turingmaschinen

Eine Sprache $A \subset \Sigma^*$ heisst entscheidbar, wenn eine TM $T$ existiert, die sich wie folgt verhält:

- Bandinhalt $x \in A$ $T$ hält mit Bandinhalt «1» (Ja) an
- Bandinhalt $x \in \Sigma^* \backslash A$ $T$ hält mit Bandinhalt «0» (Nein) an

Äquivalente Aussagen:

- $A \subset \Sigma^*$ ist entscheidbar
- Es existiert eine $TM$, die das Entscheidungsproblem $T(\Sigma, A)$ löst
- Es existiert ein WHILE-Programm, dass bei einem zu $A$ gehörenden Wort stets terminiert $\to$ Entscheidungsverfahren für $A$

### Semi-Entscheidbarkeit Turingmaschinen

Eine Sprache $A \subset \Sigma^*$ heisst semi-entscheidbar, wenn eine TM $T$ existiert, die sich wie folgt verhält:

- Bandinhalt $x \in A$ $T$ hält mit Bandinhalt «1» (Ja) an
- Bandinhalt $x \in \Sigma^* \backslash A$ $T$ hält nie an

Äquivalente Aussagen

- $A \subset \Sigma^*$ ist semi-entscheidbar
- $A \subset \Sigma^*$ ist rekursiv aufzählbar
- Es gibt eine TM, die zum Entscheidungsproblem $T(\Sigma, A)$ nur die positiven («Ja») Antworten liefert und sonst gar keine Antwort
- Es gibt ein WHILE-Programm, dass bei einem zu $A$ gehörenden Wort stets terminiert und bei Eingabe von Wörtern die nicht zu $A$ gehören nicht terminiert

### Reduzierbarkeit

Eine Sprache $A \subset \Sigma^*$ heisst auf eine Sprache $B \subset \Gamma^*$ reduzierbar, wenn es eine totale, Turing-berechenbare Funktion $F : \Sigma^* \to \Gamma^*$ gibt, so dass für alle $\omega \in \Sigma^*$

$$\omega \in A \Leftrightarrow F(\omega) \in B$$

- $A \preccurlyeq B$ $A$ ist reduzierbar auf $B$
- $A \preccurlyeq B$ und $B \preccurlyeq C \to A \preccurlyeq C$

### Halteproblem

Das allgemeine Halteproblem $H$ ist die Sprache ($\# =$ Delimiter)

- $H := \{\omega \# x \in \{0, 1, \#\}^* \mid T_\omega \text{ angesetzt auf } x \text{ hält }\}$

Sprachen der Halteprobleme (HP): leeres $HPH_0$ und spezielles HP $H_S$

- $H_0 := \{\omega \in \{0, 1\}^* \mid T_\omega \text{ angesetzt auf das leere Band hält }\}$
- $H_S := \{\omega \in \{0, 1\}^* \mid T_\omega \text{ angesetzt auf } \omega \text{ hält }\}$

$H_0, H_S$ und $H$ sind semi-entscheidbar.

## Komplexitätstheorie

### Quantitative Gesetze und Grenzen

der algorithmischen Informationsverarbeitung

- Zeitkomplexität: Laufzeit des besten Programms, welches das Problem löst
- Platzkomplexität: Speicherplatz des besten Programms
- Beschreibungskomplexität: Länge des kürzesten Programms

### Zeitbedarf

Der Zeitbedarf von $M$ auf Eingaben der Länge $n \in \mathbb{N}$ im schlechtesten Fall definiert als

$$\text{Time}_M(n) = \max\{\text{Time}_M(\omega) || \omega |= n\}$$

Sei $M$ eine TM, die immer hält und sei $\omega \in \Sigma^*$. Der Zeitbedarf von $M$ auf der Eingabe $\omega$ ist

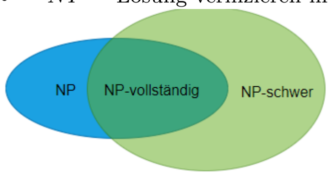
- Time $_M(\omega) =$ Anzahl von Konfigurationsübergängen in der Berechnung von $M$ auf $\omega$

### P vs NP

Klassifizierung von Problemen

Ein Problem $U$ heisst in Polynomzeit lösbar, wenn es eine obere Schranke $O(n^c)$ gibt für eine Konstante $c \geq 1$.

- $P \doteq$ Lösung finden in Polynomzeit
- $NP \doteq$ Lösung verifizieren in Polynomzeit




### NP-schwer

Eine Sprache $L$ heisst $NP$-schwer, falls für alle Sprachen

$$L' \in NP \text{ gilt, dass } L' \preccurlyeq_p L$$

Eine Sprache $L$ heisst NP-vollständig, falls $L \in NP$ und $L$ ist NP-schwer.

**Verifikation**
Polynomzeit-Verifizierer: Überprüft die einzelnen Eingaben in einem Problem
Zeuge: Informationen einer gültigen Eingabe

**Asymptotische Komplexitätsmessung** $O$-Notation (Landau Symbole)

- $f \in O(g)$: Es existiert ein $n_0 \in \mathbb{N}$ und ein $c \in \mathbb{N}$, so dass für alle $n \geq n_0$ gilt
  - $f(n) \leq c \cdot g(n) f$ wächst asymptotisch nicht schneller als $g$
- $f \in \Omega(g)$: Es existiert ein $n_0 \in \mathbb{N}$ und ein $d \in \mathbb{N}$, so dass für alle $n \geq n_0$ gilt
  - $f(n) \geq \frac{1}{d} \cdot g(n) f$ wächst asymptotisch mindestens so schnell wie $g$
- $f \in \Theta(g)$: Es gilt $f(n) \in O(g(n))$ und $f(n) \in \Omega(g(n))$
  - $f$ und $g$ sind asymptotisch gleich

**Schranken für die Zeitkomplexität von U**

- $O(f(n))$ ist eine obere Schranke, falls

Eine TM existiert, die $U$ löst und eine Zeitkomplexität in $O(f(n))$ hat.

- $\Omega(g(n))$ ist eine untere Schranke, falls

Für alle TM $M$, die $U$ lösen, gilt dass $\mathrm{Time_M}(n) \in \Omega(g(n))$

**Rechenregeln**

- Konstante Vorfaktoren $c$ ignorieren ($c \in O(1)$).
- Bei Polynomen ist nur die höchste Potenz entscheidend:

$$a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 \in O\left(n^k\right)$$

- Die $O$-Notation ist transitiv.
- $f(n) \in O(g(n)) \wedge g(n) \in O(h(n)) \rightarrow f(n) \in O(h(n))$

- $O(n)$ $\quad 7n + 4$
- $O(n^3)$ $\quad 25n^2 + n^3 + 100n$
- $O(n^2 \cdot \log(n))$ $\quad n^2 + n \cdot n \cdot (\log(n)) + 20n^2 + 50n \cdot 100$
- $O(2^n)$ $\quad 10^{20} + 3n^3 + 2^n + 2^{10} \cdot 2^{30}$

**Übersicht wichtigste Laufzeiten** **TODO:** Tabelle mit Laufzeiten

## Übersicht