Alphabete, Wörter und 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)

Sprache über einem Alphabet Σ = Eine Teilmenge $L \subseteq \Sigma^*$ von Wörtern.

- $\Sigma_1 \subseteq \Sigma_2 \land L$ Sprache über $\Sigma_1 \rightarrow L$ Sprache über Σ_2
- Σ^* Sprache über jedem Alphabet Σ
- $\{\} = \emptyset$ ist die *leere Sprache*

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

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

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

Menge von Wörter

- $\Sigma^* = \underbrace{\Sigma^0}_{1} \cup \underbrace{\Sigma^1}_{2} \cup \underbrace{\Sigma^2}_{4} \cup \underbrace{\Sigma^3}_{8} \ldots$ Kleensche Hülle
- $\Sigma^+ = \underbrace{\Sigma^1}_{2} \cup \underbrace{\Sigma^2}_{4} \cup \underbrace{\Sigma^3}_{8} \ldots = \Sigma^* \backslash \{\varepsilon\}$ Positive Hülle

Konkatenation = Verkettung von zwei beliebigen Wörtern x und y

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

Wortpotenzen, Sei x ein Wort über einem Alphabet Σ.

- $x^0 = \varepsilon$
- $x^{n+1} = x^n \circ x = x^n x$
- $bbabababab baaaabab = b^2(ab)^4 ba^3 (ab)^2$

Ein **Wort** ist eine endliche Folge von Symbolen eines bestimmten Alphabets.

- abc Wort über dem Alphabet Σ_{lat} (oder über $\Sigma = \{a, b, c\}$)
- 100111 Wort über dem Alphabet $\{0,1\}$
- ε Leeres Wort (über jedem Alphabet)

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

$$\omega \neq v \rightarrow Echtes\ Teilwort$$

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

Σ^k = Die **Menge aller Wörter der Länge k** über einem Alphabet Σ

- $\Sigma = \{a, b, c\}$ $\Sigma^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$
- $\Sigma = \{0,1\}$ $\{0,1\}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$
- $\Sigma^0 = \{\varepsilon\}$

$|\omega|$ = **Länge eines Wortes**

- $|100111| = 6$
- $|\varepsilon| = 0$

$|\omega|_x$ = **Häufigkeit eines Symbols** x in einem Wort

- $|100111|_1 = 4$
- $|\varepsilon|_0 = 0$
- $|\varepsilon|_\varepsilon = 1$

ω^R = **Spiegelwort** / **R**eflection zu ω

- $(abc)^R = cba$
- $(100111)^R = 111001$
- $\varepsilon^R = \varepsilon$

Reguläre Ausdrücke

Reguläre Ausdrücke sind Wörter, die Sprachen beschreiben.

Die Sprache RA_Σ der **<u>R</u>egulären <u>A</u>usdrücke** über einem Alphabet Σ ist wie folgt definiert:

- $\emptyset, \epsilon \in RA_\Sigma$
- $\Sigma \subset RA_\Sigma$
- $R \in RA_\Sigma \quad \Rightarrow (R^*) \in RA_\Sigma$
- $R, S \in RA_\Sigma \quad \Rightarrow (RS) \in RA_\Sigma$
- $R, S \in RA_\Sigma \quad \Rightarrow (R|S) \in RA_\Sigma$

Für jeden regulären Ausdruck $R \in RA_\Sigma$ definieren wir die Sprache $L(R)$ von R wie folgt:

- $L(\emptyset) \quad = \emptyset$ Leere Sprache
- $L(\epsilon) \quad = \{\varepsilon\}$ Sprache , die nur das leere Wort enthält
- $L(a) \quad = \{a\}$ für $a \in \Sigma$ Beschreibt die Sprache $\{a\}$
- $L(R^*) \quad = L(R)^*$ Kombinierten Wörter von R
- $L(R|S) \quad = L(R) \cup L(S)$ Wörter die von R oder S beschrieben werden
- $L(RS) \quad = L(R)L(S)$ Verkettungen von Wörtern ($R = prefix$)

Reguläre Sprache

Eine Sprache A über dem Alphabet Σ heisst *regulär*, falls

- $A = L(R)$ für einen regulären Ausdruck $R \in RA_\Sigma$ gilt.

<u>Beispiele</u>

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

$L(R_1)$: Menge der ganzen Zahlen in Dezimaldarstellung

- $\big((-|\varepsilon)(1,2,3,4,5,6,7,8,9)(0,1,2,3,4,5,6,7,8,9)\big|0\big).0$

Eigenschaften und Konventionen

Die Menge RA_Σ über dem Alphabet Σ ist eine

- Sprache über dem Alphabet $\{\emptyset, \epsilon, *, (,), |\} \cup \Sigma.$

Priorisierung von Operatoren

- *(1) * = Wiederholung → (2) Konkatenation → (3) | = Oder*

<u>Beispiele</u>

- $(aa)^*b^*aba \quad = (aa)^*b^*aba$
- $(ab)|(ba) \quad = ab|ba$
- $a\big(b(ba)\big)|b \quad = abba|b$

Erweiterte Syntax

- $R^+ = R(R^*)$
- $R? = (R|\epsilon)$
- $[R_1, \dots, R_k] = R_1|R_2| \dots |R_k$

Endliche Automaten

Endliche Automaten entsprechen Maschinen, die Entscheidungsprobleme lösen.

- Links nach rechts
- Keinen Speicher
- Keine Variablen
- Speichert aktuellen Zustand
- Ausgabe über akzeptierende Zustände

$M = (Q, \Sigma, \delta, q_0, F)$ ein EA. **Konfiguration** von M auf ω ist ein Element aus $Q \times \Sigma^*$.

- Startkonfiguration von M auf ω $\quad \{q_0, \omega\} \in \{q_0\} \times \Sigma^*$
- Endkonfiguration $\quad (q_n, \varepsilon)$

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

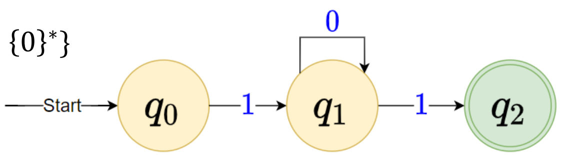
$$(q, \omega) \vdash_M (p, x)$$

Eine **Berechnung** ist eine endliche Folge von Berechnungsschritten

$$(q_a, \omega_1\omega_2 \dots \omega_n) \vdash_M \dots \vdash_M \left(q_e, \omega_j \dots \omega_n\right) \rightarrow (q_a, \omega_1\omega_2 \dots \omega_n) \vdash_M^* \left(q_e, \omega_j \dots \omega_n\right)$$

Beispiel DEA (eindeutig)

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



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*

Ein (deterministischer) endlicher Automat (**DEA**) ist ein Quintupel

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

- Zustände $\quad Q = \{q_0, q_1, \dots, q_n\} \quad (n \in \mathbb{N})$
- Eingabealphabet $\quad \Sigma = \{a_1, a_2, \dots, a_m\} \quad (m \in \mathbb{N})$
- Übergangsfunktion $\quad \delta\colon Q \times \Sigma \rightarrow Q$
- Startzustand $\quad q_0 \in Q$
- Akzeptierende Zustände $\quad F \subseteq Q$

Ein (nichtdeterministischer) endlicher Automat (**NEA**) ist ein Quintupel

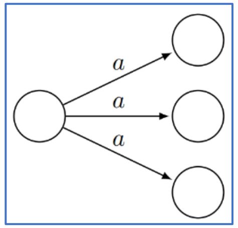
$$M = (Q, \Sigma, \delta, q_0, F)$$

Der einzige Unterschied zum DEA besteht in der Übergangsfunktion δ

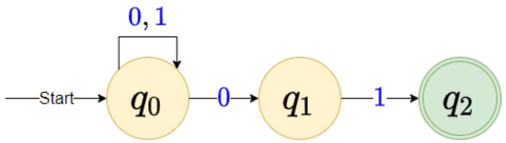
- Übergangsfunktion $\quad \delta\colon Q \times \Sigma \rightarrow P(Q)$

Ein ε-NEA erlaubt zusätzlich noch ε-Übergänge.

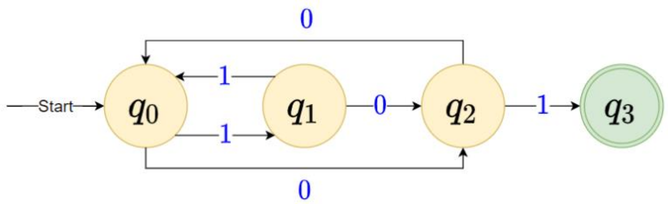
Beispiel NEA (nicht eindeutig)



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



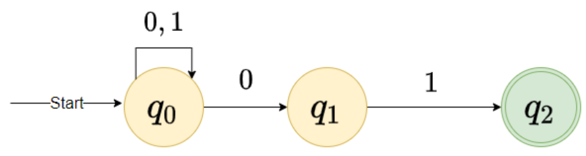
Äquivalenter DEA

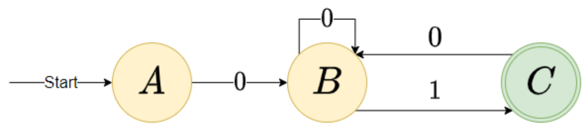


Teilmengenkonstruktion (DEA's und NEA's sind gleich Mächtig)

1. $Q_{NEA} \rightarrow P(Q_{NEA}) = Q_{DEA}$ (*Potenzmenge*)
2. Verbinden mit Vereinigung aller möglichen Zielzustände
3. *Nicht erreichbare Zustände* eliminieren
4. Enthält akzeptierenden Zustand $= F_{NEA} \rightarrow$ 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	~~$\{q_0, q_1, q_2\}$~~	~~$\{q_0, q_1\}$~~	~~$\{q_0, q_2\}$~~

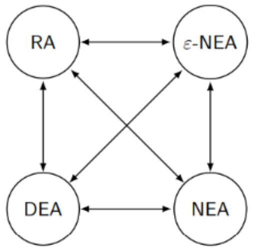




Reguläre Sprachen

Reguläre Sprachen sind durch verschiedene äquivalente Mechanismen darstellbar

- Akzeptierender Mechanismus DEA, NEA, ε-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 Σ. Dann ist die *Vereinigung* … regulär.

$$L_1 \cup L_2 = \{\omega | \omega \in L_1 \vee \omega \in L_2\}$$

Sei L eine reguläre Sprache über Σ. Dann ist auch das *Komplement* … regulär.

$$\bar{L} = \Sigma^* - L = \{\omega \in \Sigma^* | \omega \notin L\}$$

- Schnitt $L_1 \cap L_2 = \{\omega | \omega \in L_1 \wedge \omega \in L_2$
- Differenz $L_1 - L_2 = \{\omega | \omega \in L_1 \wedge \omega \notin L_2\}$
- Konkatenation $L_1 \cdot L_2 = L_1 L_2 = \{\omega = \omega_1 \omega_2 | \omega_1 \in L_1 \wedge \omega_1 \in L_2\}$
- Kleenesche Hülle $L^* = \{\omega = \omega_1 \omega_2 \dots \omega_n | \omega_i \in L \; f\ddot{u}r \; alle \; i \in \{1,2,\dots,n\}\}$

Zustandsklasse

Jedes Wort landet in einem Zustand

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

Kein Wort landet nach dem Lesen in zwei Zuständen

$$[p] \cap [q] = \emptyset, f\ddot{u}r \; alle \; p \neq q, p, q \in Q$$

<u>Beispiel</u>

Nach dem Lesen von ω landet man im Zustand p.

$$Klasse[q_0] = \{\omega \in \{0,1\}^* | |\omega|_0 \; mod(3) = 1\}$$

Von M akzeptierte Sprache

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

<u>Beispiel</u>

$$L(M) = \{\forall \omega \in \{0,1\}^* | |\omega|_0 \; mod(3) = 1\}$$

Kontextfreie Grammatiken

Eine **K**ontext**F**reie **G**rammatik $G(KFG)$ ist ein 4-Tupel (N, Σ, P, A) mit

- N ist das Alphabet der **Nichtterminale** (Variablen)
- Σ 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 α, β und γ *Satzformen* und $A \to \gamma$ eine Produktion.

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

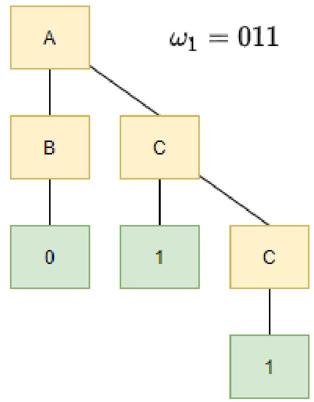
Eine Ableitung kann als ***Ableitungsbaum*** / *Parsebaum* dargestellt werden.

KGF G_1 für die Sprache $\{0^n 1^m \mid n, m \in N\}$

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

Ableitung von $\omega_1 = 011$

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



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

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

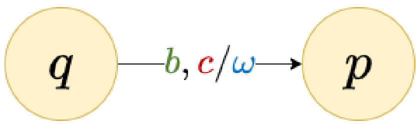
$$M = (Q, \Sigma, \boldsymbol{\Gamma}, \boldsymbol{\delta}, q_0, \$, F)$$

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

Zusätzliche Einschränkung für DKA's

Für jeden Zustand q und alle Symbole x, b gilt, wenn $\delta(q, b, c)$ definiert ist, dann ist $\delta(q, \varepsilon, x)$ undefiniert.

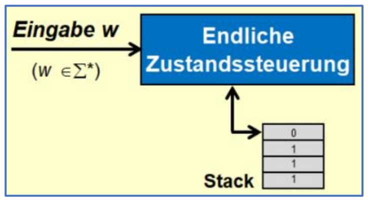
Ein Übergang $\delta(q, b, c) = (p, \omega)$ wird graphisch dargestellt



Berechnungsschritte

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

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



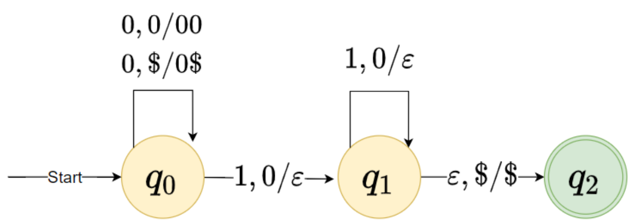
Die Sprache $L(M)$ des Kellerautomaten M ist definiert durch

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

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

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

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



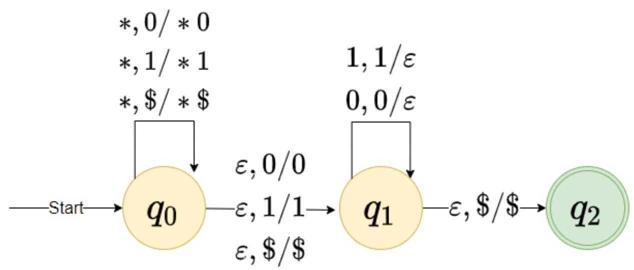
- $\omega_1 = 011: (q_0, 011, \$) \vdash (q_1, 11, 0\$) \vdash (q_1, 1, \$) \rightarrow \omega_1$ *verwerfend*

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 $\{\omega\omega^R \mid \omega \in \{0,1\}^*\}$



Turingmaschinen

Eine Turing-Maschine TM hat …

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

Eine deterministischer Turing-Maschine TM ist ein 7-Tupel

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

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

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

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

Das **Band**

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

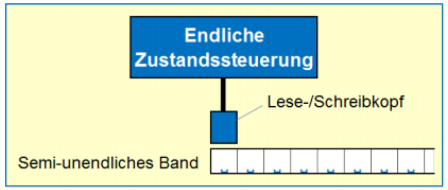
Die **Konfiguration** einer Turing-Maschine M ist durch die folgenden Angaben eindeutig spezifiziert

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

Semi-Unendliches Band

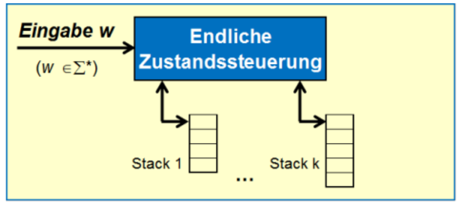
Das Band der Turingmaschine ist nur in eine Richtung unendlich.

Jede Sprache L die von einer TM T akzeptiert wird, wird auch von einer *TM mit semi-unendlichem Band* akzeptiert.



Maschinen mit mehreren Stacks

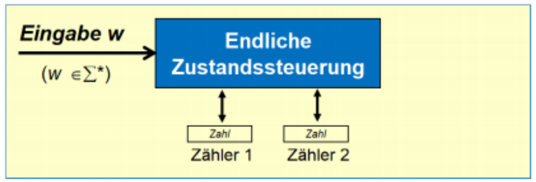
Jede Sprache L die von einer TM T akzeptiert wird, wird auch von einer *2-Stack-Maschine* S akzeptiert.



Zähler-Maschinen

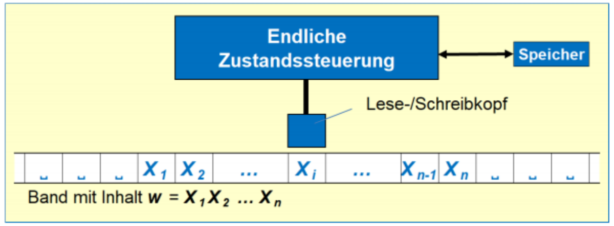
Eine Zähler-Maschine (Counter Machine) mit k Zählern entspricht einer *k-Stack-Maschine* mit dem Unterschied, dass die Stacks durch einfache Zähler ersetzt werden.

Jede Sprache L die von einer TM T akzeptiert wird, wird auch von einer *2-Zähler-Maschine* Z mit 2 Zählern akzeptiert.



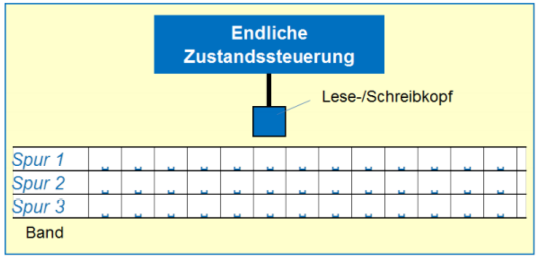
Erweiterung: **TM mit Speicher**

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



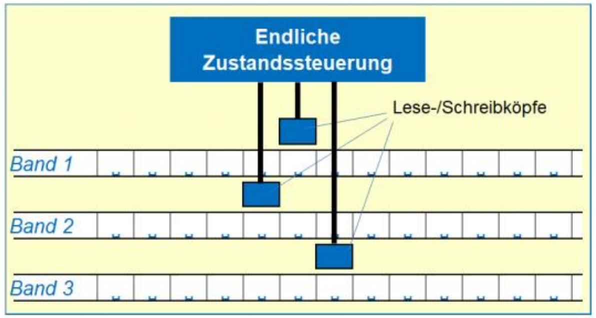
Erweiterung: **Mehrere Spuren**

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



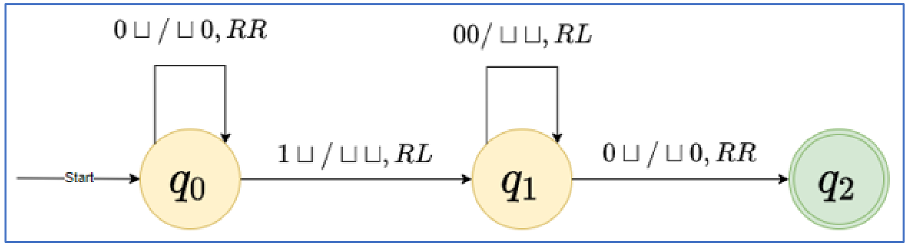
Erweiterung: **Mehreren Bändern**

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



Beispiel einer 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	$0 q_0 \sqcup \vdash$		0								
1	$\sqcup\sqcup q_0 00100 \vdash$	$0\sqcup/\sqcup 0, RR$			0	0	1	0	0		
2	$00 q_0 \sqcup \vdash$		0	0							
1	$\sqcup\sqcup\sqcup q_0 0100 \vdash$	$0\sqcup/\sqcup 0, RR$				0	1	0	0		
2	$000 q_0 \sqcup \vdash$		0	0	0						
1	$\sqcup\sqcup\sqcup\sqcup q_0 100 \vdash$	$1\sqcup/\sqcup\sqcup, RL$					1	0	0		
2	$0000 q_0 \sqcup \vdash$		0	0	0	0					
1	$\sqcup\sqcup\sqcup\sqcup\sqcup q_1 00 \vdash$	$00/\sqcup\sqcup, RL$						0	0		
2	$000 q_1 0 \vdash$		0	0	0	0					
1	$\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup q_1 0 \vdash$	$00/\sqcup\sqcup, RL$							0		
2	$00 q_1 0 \vdash$		0	0	0						
1	$\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup q_1$	$\sqcup 0/\sqcup 0, RR$									
2	$0 q_1 0 \vdash$		0	0							
1	$\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup q_2 \sqcup$										
2	$00 q_2 \sqcup \vdash$		0	0							

Berechnungsmodelle

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

- Computer und Turing-Maschinen sind *äquivalent.*

Turing-berechenbare Funktion: Turing-Maschine $T = (Q, \Sigma, \Gamma, \dots)$

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

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

Die primitiv rekursiven *Grundfunktionen* sind

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

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

- Die *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.

$$\pi_k^n = \mathbb{N}^n \to \mathbb{N} \text{ mit } \pi_k^n(\underbrace{x_1, \dots, x_k \dots, x_n}_{n = Anz.\ Argumente}) = k$$

Loop (Primitiv-Rekursiv)

- *Zuweisungen* $x = y + c$ und $x = y - c$
- *Sequenzen* $P \text{ und } Q \to P; Q$
- *Schleifen* $P \to Loop\ x\ Do\ P\ End$

Beispiel

Addition von natürlichen Zahlen. $Add(x, y) = x + y$

```
Loop x1 Do
        x2 = x2 + 1
End;
x0 = x2 + 0
```

While (Turing vollständig)

Erweiterung der Sprache LOOP

- *While* $x_i > 0$ *Do … End*

Beispiel

Multiplikation. $Mul(x, y) = x \cdot y$

```
While x1 > 0 Do
        x1 = x1 - 1;
        Loop x2 Do
                x0 = x0 + 1
        End
End
```

GoTo (Turing vollständig)

- *Zuweisungen* Mk: $x_i = x_j + c$ und $x_i = x_j - c$
- *Sprunganweisung* Mk: IF $x_i = c$ THEN GOTO Mr
  Mk: GOTO Mr
- *Schleifen* Mk: HALT

Beispiel

$$F(x, y, z) = \begin{cases} y & falls\ x \neq 0 \\ z & sonst \end{cases}$$

M1: $x_0 = x_3 + 0;$
M2: $\text{IF } x_1 = 0 \text{ THEN GOTO } M4$
M3: $x_0 = x_2 + 0;$
M4: HALT

Entscheidbarkeit

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

- Bandinhalt $x \in A$ $\quad$ T hält mit Bandinhalt «1» (Ja) an
- Bandinhalt $x \in \Sigma^* \setminus A$ $\quad$ 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 → *Entscheidungsverfahren* für A

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

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

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 Σ^*: $\quad \bar{A} = \Sigma^* \setminus A = \{\omega \in \Sigma^* \mid \omega \notin A\}$

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 $\omega \in \Sigma^*$

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

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

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

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

Sprachen der Halteprobleme (HP): *leeres HP* H_0 und *spezielles HP* H_S

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

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

Komplexitätstheorie

Theorie der quantitativen 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

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

- $Time_M(\omega) =$ Anzahl von Konfigurationsübergängen in der Berechnung von M auf ω

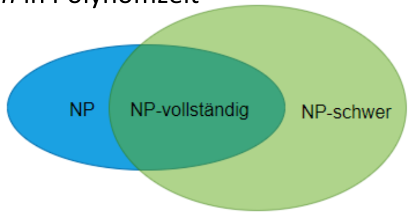
Der **Zeitbedarf** von M auf *Eingaben der Länge* $n \in \mathbb{N}$ im schlechtesten Fall definiert als

$$Time_M(n) = \max\,\{Time_M(\omega) \mid |\omega| = n\}$$

Klassifizierung von Problemen

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

- $P \triangleq$ Lösung *finden* in Polynomzeit
- $NP \triangleq$ Lösung *verifizieren* in Polynomzeit



Polynomzeit-Verifizierer: Überprüft die eine einzelne Eingabe in einem Problem.

Zeuge: Informationen einer gültigen? Eingabe.

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.

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 $f(n) \in O\big(g(n)\big)$ und $f(n) \in \Omega\big(g(n)\big)$ f und g sind asymptotisch gleich

Schranken für die Zeitkomplexität von U

- $O\big(f(n)\big)$ ist eine *obere Schranke*, falls
  Eine TM existiert, die U löst und eine Zeitkomplexität in $O\big(f(n)\big)$ hat.
- $\Omega\big(g(n)\big)$ ist eine *untere Schranke*, falls
  Für alle TM M, die U lösen, gilt dass $\mathrm{Time}_M(n) \in \Omega\big(g(n)\big)$

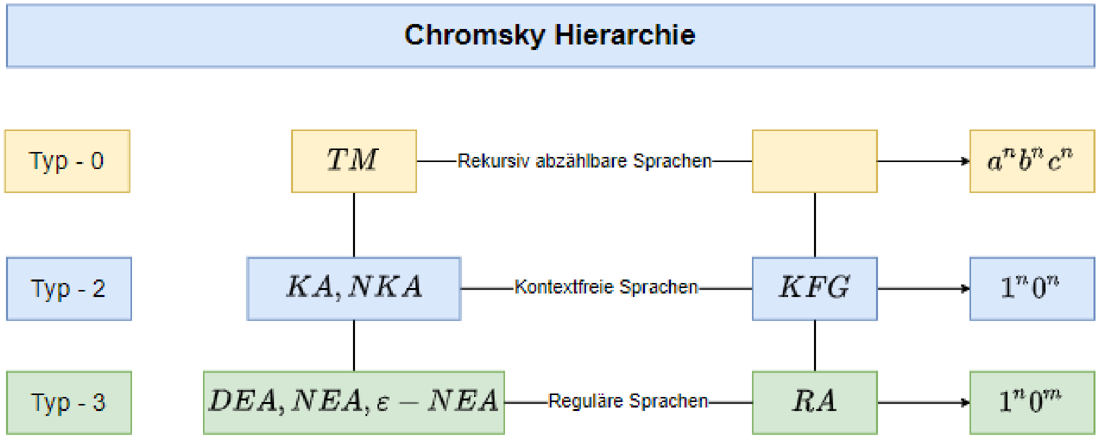
Rechenregeln

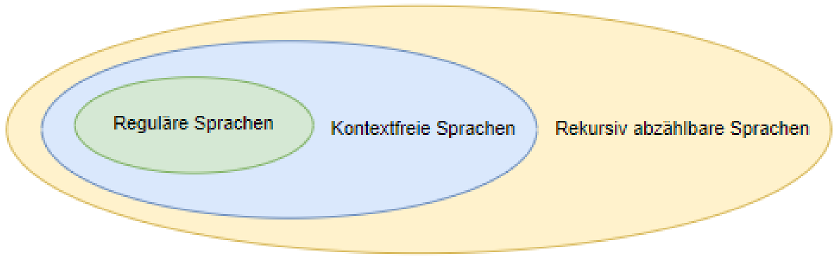
- Konstante Vorfaktoren c ignorieren $\big(c \in O(1)\big)$.
- Bei Polynomen ist nur die höchste Potenz entscheidend:
  $$a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 \in O(n^k)$$
- Die O-Notation ist transitiv.
- $f(n) \in O\big(g(n)\big) \wedge g(n) \in O\big(h(n)\big) \rightarrow f(n) \in O\big(h(n)\big)$

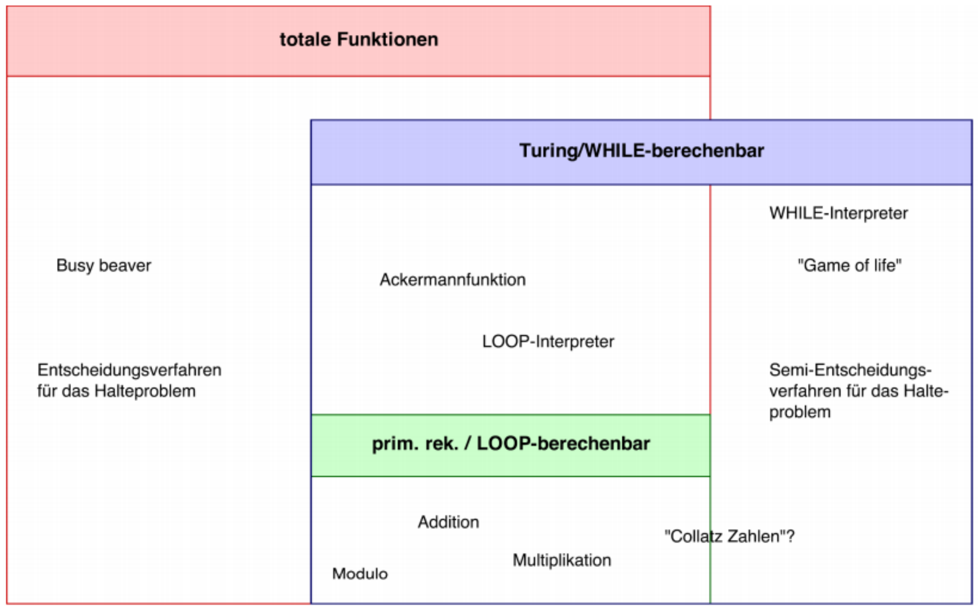
<u>Beispiel</u>

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

Übersicht


Chromsky Hierarchie
Typ - 0
Typ - 2
Typ - 3
TM
Rekursiv abzählbare Sprachen
$a^n b^n c^n$
KA, NKA
Kontextfreie Sprachen
KFG
$1^n 0^n$
$DEA, NEA, \varepsilon - NEA$
Reguläre Sprachen
RA
$1^n 0^m$


Reguläre Sprachen
Kontextfreie Sprachen
Rekursiv abzählbare Sprachen


totale Funktionen
Turing/WHILE-berechenbar
WHILE-Interpreter
"Game of life"
Busy beaver
Ackermannfunktion
LOOP-Interpreter
Entscheidungsverfahren für das Halteproblem
Semi-Entscheidungs-verfahren für das Halte-problem
prim. rek. / LOOP-berechenbar
Addition
"Collatz Zahlen"?
Multiplikation
Modulo