

## Alphabete, Wörter, Sprachen

**Alphabete** endliche, nichtleere Mengen von Symbolen.

- $\Sigma_{\text{Bool}} = \{0, 1\}$  Boolesches Alphabet

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

**Wort** endliche Folge von Symbolen eines bestimmten Alphabets.

**Schreibweisen**  $|\omega| = \text{Länge eines Wortes}$

$|\omega|_x = \text{Häufigkeit eines Symbols } x \text{ in einem Wort}$

$\omega^R = \text{Spiegelwort/Reflection zu } \omega$

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

$\omega \neq v \rightarrow$  Echtes Teilwort, Präfix = Anfang, Suffix = Ende

**Mengen von Wörtern**  $\Sigma^k =$  Wörter der Länge  $k$  über Alphabet  $\Sigma$

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

$\varepsilon$  Leeres Wort (über jedem Alphabet)  $\Sigma^0 = \{\varepsilon\}$

**Konkatenation** = Verkettung von zwei beliebigen Wörtern  $x$  und  $y$   
 $x \circ y = xy := (x_1, x_2 \dots x_n, y_1, y_2 \dots y_m)$

**Wortpotenzen** Sei  $x$  ein Wort über einem Alphabet  $\Sigma$

- $x^0 = \varepsilon$
- $x^{n+1} = x^n \circ x = x^n x$

**Sprache** über Alphabet  $\Sigma =$  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  $A$  und  $B$ :  $AB = \{uv \mid u \in A \text{ und } v \in B\}$

**Kleenesche Hülle**  $A^*$  von  $A$ :  $\{\varepsilon\} \cup A \cup AA \cup AAA \cup \dots$

**Reguläre Ausdrücke** Wörter, die Sprachen beschreiben

**$RA_\Sigma$**  Sprache der Regulären Ausdrücke über  $\{\emptyset, \epsilon, *, (), \cdot, | \} \cup \Sigma$

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

**Priorisierung von Operatoren**

(1)  $*$  = Wiederholung  $\rightarrow$  (2) Konkatenation  $\rightarrow$  (3)  $\mid$  Oder

**Erweiterter Syntax**

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

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

**$\forall R \in RA_\Sigma$**  definieren wir die Sprache  $L(R)$  von  $R$  wie folgt:

- Leere Sprache:  $L(\emptyset) = \emptyset$
- Sprache, die nur das leere Wort enthält:  $L(\epsilon) = \{\epsilon\}$
- Beschreibt die Sprache  $\{a\}$ :  $L(a) = \{a\} \quad \forall a \in \Sigma$
- Kombiniert die Wörter von  $R$ :  $L(R^*) = L(R)^*$
- Verkettung von Wörtern ( $R = \text{prefix}$ ):  $L(RS) = L(R) \circ L(S)$
- Wörter in  $R$  oder  $S$ :  $L(R \mid S) = L(R) \cup L(S)$

## Endliche Automaten

**Endliche Automaten** Maschinen, die Entscheidungsprobleme lösen

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

**DEA** deterministischer endlicher Automat:  $M = (Q, \Sigma, \delta, q_0, F)$

$Q$ : endliche Menge von Zuständen  $q_0 \in Q$  Startzustand

$\Sigma$ : endliches Eingabealphabet  $F \subseteq Q$  Menge der

$\delta : Q \times \Sigma \rightarrow Q$  Übergangsfunktion akzeptierenden Zustände

**DEA Funktionen**  $M = (Q, \Sigma, \delta, q_0, F) : \text{EA}$ .

**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  $(q_n, \epsilon)$

**Berechnungsschritt**  $\vdash_M$  von  $M$   $(q, \omega) \vdash_M (p, x)$

**Berechnung** ist eine endliche Folge von Berechnungsschritten

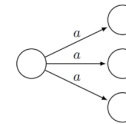
$(q_a, \omega_1 \omega_2 \dots \omega_n) \vdash_M \dots \vdash_M (q_e, \omega_j \dots \omega_n) \rightarrow (q_a, \omega_1 \omega_2 \dots \omega_n) \vdash_M^* (q_e, \omega_j \dots \omega_n)$

**Nichtdeterministischer endlicher Automat (NEA)**

Unterschied zum DEA: Übergangsfunktion  $\delta$

Übergangsfunktion  $\delta : Q \times \Sigma \rightarrow P(Q)$

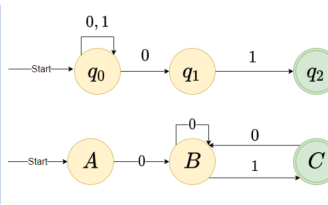
Ein  $\varepsilon$ -NEA erlaubt zusätzlich noch  $\varepsilon$ -Übergänge



**Teilmengekonstruktion**  $\forall$  NEA kann in DEA umgewandelt werden

- $Q_{NEA} \rightarrow P(Q_{NEA}) = Q_{DEA}$  (Potenzmenge)
- Verbinden mit Vereinigung aller möglichen Zielzustände
- Nicht erreichbare Zustände eliminieren
- Enthält akzeptierenden Zustand  $= F_{NEA} \rightarrow$  akzeptierend

$\downarrow$	$q$	$\delta(q, 0)$	$\delta(q, 1)$
0	$\emptyset$	$\emptyset$	$\emptyset$
1	$A = \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
2	$\{q_1\}$	$\emptyset$	$\{q_2\}$
3	$\{q_2\}$	$\emptyset$	$\emptyset$
4	$B = \{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
5	$C = \{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
6	$\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
7	$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



Reguläre Sprachen und endliche Automaten

**Reguläre Sprachen** durch äquivalente Mechanismen beschreibbar

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

**Zustandsklasse**  $\Sigma^* = \bigcup_{p \in Q} [p]$   $[p] \cap [q] = \emptyset, \forall p \neq q, p, q \in Q$

Jedes Wort landet in einem Zustand, aber kein Wort landet nach dem Lesen in zwei Zuständen!

**Eigenschaften** Seien  $L, L_1$  und  $L_2$  reguläre Sprachen über  $\Sigma$

- Vereinigung:  $L_1 \cup L_2 = \{\omega \mid \omega \in L_1 \vee \omega \in L_2\}$
- Schnitt:  $L_1 \cap L_2 = \{\omega \mid \omega \in L_1 \wedge \omega \in L_2\}$
- Differenz:  $L_1 - L_2 = \{\omega \mid \omega \in L_1 \wedge \omega \notin L_2\}$
- Komplement:  $\bar{L} = \Sigma^* - L = \{\omega \in \Sigma^* \mid \omega \notin L\}$
- Konkatenation:  $L_1 \cdot L_2 = L_1 L_2 = \{\omega = \omega_1 \omega_2 \mid \omega_1 \in L_1 \wedge \omega_2 \in L_2\}$
- Kleenesche Hülle:  $L^* = \{\omega = \omega_1 \omega_2 \dots \omega_n \mid \omega_i \in L \text{ für alle } i \in \{1, 2, \dots, n\}\}$

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

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

## Kontextfreie Grammatiken

**Kontextfreie Grammatik** ( $KFG$ ) ist ein 4-Tupel  $(N, \Sigma, P, A)$  mit

- $N$ : Alphabet der Nichtterminale (Variablen)
- $\Sigma$ : Alphabet der Terminale
- $P$ : endliche Menge von Produktionen mit der Form  $X \rightarrow \beta$   
Mit Kopf  $X \in N$  und Rumpf  $\beta \in (N \cup \Sigma)^*$
- $A$ : Startsymbol, wobei  $A \in N$

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

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

- Ableitungsschritt mit Produktion  $A \rightarrow \gamma$   $\alpha A \beta \rightarrow \alpha \gamma \beta$
- Ableitung Folge von Ableitungsschritten  $\alpha \rightarrow \dots \rightarrow \omega$

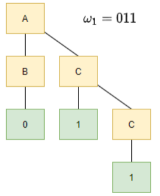
**Ableitungsbaum (Parsebaum)**

mögliche Darstellung einer Ableitung

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

Ableitung von  $\omega_1 = 011$

- $A \rightarrow BC \rightarrow 0AA \rightarrow 01C \rightarrow 011 \rightarrow \dots$   
 $\rightarrow 011$



**Mehrdeutige KFG**  $\exists$  Wort, das mehrere Ableitungsbäume besitzt.

**Mehrdeutigkeiten eliminieren:**

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

**Reguläre Sprache durch KFG beschreiben**  $\forall L \exists KFG$

$L$  reguläre Sprache  $\Rightarrow \exists$  DEA  $M = (Q, \Sigma, \delta, q_0, F)$  mit  $L(M) = L$

KFG für  $L$  bauen:

- $\forall$  Zustand  $q_i$  gibt es ein Nichtterminal  $Q_i$
- $\forall$  Transition  $\delta(q_i, a) = q_j$  erstelle Produktion  $Q_i \rightarrow aQ_j$
- $\forall$  akzeptierenden Zustand  $q_i \in F$  erstelle Produktion  $Q_i \rightarrow \varepsilon$
- Nichtterminal  $Q_0$  wird zum Startsymbol  $A$ .

## Kellerautomaten

**Kellerautomaten** (KA) besitzt «Speicher»

Deterministischer KA (DKA):  $M = (Q, \Sigma, \Gamma, \delta, q_0, \$, F)$

$Q$ : Menge von Zuständen  $q_0 \in Q$ : Anfangszustand

$\Sigma$ : Alphabet der Eingabe  $\$ \in \Gamma$ : Symbol vom Alphabet des Kellers

$\Gamma$ : Alphabet des Kellers  $F \subseteq Q$ : Akzeptierende Zustände

Übergangsfunktion:  $\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \rightarrow Q \times \Gamma^*$

**NKA:**  $\delta : Q \times (\Sigma \cup \varepsilon) \times \Gamma \rightarrow P(Q \times \Gamma^*)$  (Nichtdeterministischer KA)

**Zusätzliche Einschränkungen**  $\forall$  Zustand  $q$  und  $\forall$  Symbole  $x, b$  gilt: wenn  $\delta(q, b, c)$  definiert ist, dann ist  $\delta(q, \varepsilon, x)$  undefiniert.

Darstellung Übergang  $\delta(q, b, c) = (p, \omega) : q - b, c/\omega \rightarrow p$

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

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

**Sprache  $L(M)$**  eines Kellerautomaten  $M$  ist definiert durch

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

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

## Turingmaschinen

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

$Q$ : Menge von Zuständen  $q_0 \in Q$ : Anfangszustand

$\Sigma$ : Alphabet der Eingabe  $F \subseteq Q$ : Akzeptierende Zustände

$\Gamma$  und  $\Sigma \subset \Gamma$ : Bandalphabet  $\sqcup$ : Leerzeichen mit  $\mu \in \Gamma$  und  $\mu \notin \Sigma$

Übergangsfunktion:  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times D, D = \{L, R\}$

Sie bestehen aus einem Lese-/Schreibkopf und einem unendlichen Band von Zellen.

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

$D$  = Direction

$X$  = Read  $q, p \in Q$  und  $X, Y \in \Gamma$

$Y$  = Overwrite

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

**Band** Unterteilt in 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  $\sqcup$

**Konfiguration** einer TM  $M$  wird eindeutig spezifiziert durch:  
Zustand der Zustandssteuerung, Position des Lese-/Schreibkopfes  
und Bandinhalt

**Arten von TMs**  $\forall$  Sprachen  $L$  gleich akzeptierend wie normale TM

- semi-unendliches Band
- mehrere Stacks
- mit Speicher
- mehrere Spuren
- mit Zählern
- mehrere Bändern

### Mehrband-Maschine

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



Beispiel:  $4 - 2 = 2$

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

## Berechnungsmodelle

**Turing-berechenbar** = kann von Turing-Maschine gelöst werden

Turing-berechenbare Funktion für TM  $T : \Sigma^* \rightarrow \delta^*$

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

Jedes algorithmisch lösbares Berechnungsproblem ist turing-berechenbar  
 $\Rightarrow$  Computer  $\equiv$  TM

### Primitiv rekursive Grundfunktionen

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

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

Nachfolgerfunktion:  $\eta : \mathbb{N} \rightarrow \mathbb{N}$  mit  $\eta(x) = x + 1$

$\forall n \in \mathbb{N}, 1 < k < n$  die n-stellige Projektion auf die k-te Komponente:

$$\pi_k^n : \mathbb{N}^n \rightarrow \mathbb{N} \text{ mit } \pi_k^n(x_1, \dots, x_k, \dots, x_n) = x_k$$

n = Anzahl der Argumente, k = Position des Arguments

### Loop (primitiv-rekursiv)

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

### Addition von natürlichen Zahlen

Add(x, y) =  $x + y$

```
1  LOOP x1 DO
2     x2 = x2 + 1
3  END
4  x0 = x2 + 0
```

### While (Turing vollständig)

Erweiterung der Sprache Loop

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

**Multiplikation von natürlichen Zahlen**  
Mul(x, y) =  $x * y$

```
1  WHILE x1 > 0 DO
2     x1 = x1 - 1
3  LOOP x2 DO
4     x0 = x0 + 1
5  END
6  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

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

## Entscheidbarkeit

### Entscheidbarkeit

**Entscheidbar:**  $\exists$  Algorithmus, der  $\forall$  Eingabe eine Antwort liefert  
**Semi-entscheidbar:**  $\exists$  Algorithmus, der  $\forall$  Eingabe eine Antwort liefert, falls Antwort die Antwort «Ja» ist

**Entscheidbarkeit** einer Sprache  $A \subset \Sigma^*$

$A \subset \Sigma^*$  ist entscheidbar  $\Leftrightarrow A$  und  $\bar{A}$  sind semi-entscheidbar

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

### Entscheidbarkeit mit Turingmaschinen

$A \subset \Sigma^*$  heisst entscheidbar, wenn TM  $T$  existiert, sodass:

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

### Semi-Entscheidbarkeit mit Turingmaschinen

$A \subset \Sigma^*$  heisst semi-entscheidbar, wenn TM  $T$  existiert, sodass:

- Bandinhalt  $x \in A$   $T$  hält mit Bandinhalt «1» (Ja) an
- Bandinhalt  $x \in \Sigma^* \setminus A$   $T$  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**  $A \preceq B$   $A$  ist reduzierbar auf  $B$

$A \subset \Sigma^*$  heisst auf  $B \subset \Gamma^*$  reduzierbar, wenn es eine totale, Turing-berechenbare Funktion  $F : \Sigma^* \rightarrow \Gamma^*$  gibt, so dass:

$$\forall \omega \in \Sigma^* \quad \omega \in A \Leftrightarrow F(\omega) \in B$$

$A \preceq B$  und  $B \preceq C \rightarrow A \preceq C$

**Allgemeines Halteproblem H** ist die Sprache ( $\#$  = Delimiter):

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

Weitere Halteprobleme (HP): leeres  $HPH_0$  und spezielles HP  $HS$

- $H_0 := \{ \omega \in \{0, 1\}^* \mid T_\omega \text{ angesetzt auf das leere Band hält} \}$
  - $HS := \{ \omega \in \{0, 1\}^* \mid T_\omega \text{ angesetzt auf } \omega \text{ hält} \}$
- $H_0, HS$  und  $H$  sind semi-entscheidbar.

# Komplexitätstheorie

## Quantitative Gesetze und Grenzen

der algorithmischen Informationsverarbeitung

- Zeitkomplexität: Laufzeit des besten Programms
- Platzkomplexität: Speicherplatz des besten Programms
- Beschreibungskomplexität: Länge des kürzesten Programms

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

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

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

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

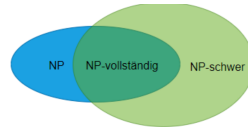
## P vs NP Klassifizierung von Problemen

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 und NP-vollständig

Eine Sprache  $L$  heisst  $NP$ -schwer, falls für alle Sprachen  $L' \in NP$  gilt, dass  $L' \preceq_p L$ .  
 $L$  heisst  $NP$ -vollständig, falls  $L \in NP$  und  $L$  ist  $NP$ -schwer.



## Polynomzeit-Verifizierer

Überprüft die einzelnen Eingaben in einem Problem  
Zeuge: Informationen einer gültigen Eingabe

## Asymptotische Komplexitätsmessung $O$ -Notation

- $f \in O(g)$ : 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  $\text{Time}_M(n) \in \Omega(g(n))$

## Rechenregeln

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

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

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

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

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