

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 ω 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 Σ

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

ε 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 Σ

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

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 Σ_2
- Σ^* Sprache über jedem Alphabet Σ
- $\{\} = \emptyset$ ist die leere Sprache

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

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

Reguläre Ausdrücke Wörter, die Sprachen beschreiben

RA_Σ Sprache der Regulären Ausdrücke über $\{\emptyset, \epsilon, *, (), , \mid\} \cup \Sigma$

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

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 Σ 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

Σ : 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 ω ist ein Element aus $Q \times \Sigma^*$

- Startkonfiguration von M auf ω $\{q_0, \omega\} \in \{q_0\} \times \Sigma^*$

- Endkonfiguration (q_n, ϵ)

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

Berechnung ist eine endliche Folge von Berechnungsschritten

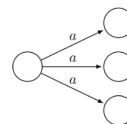
$(q_a, \omega_1 \omega_2 \dots \omega_n) \vdash_M \dots \vdash_M (q_e, \omega_j \dots \omega_n) \rightarrow (q_a, \omega_1 \omega_2 \dots \omega_n) \vdash_M^* (q_e, \omega_j \dots \omega_n)$

Nichtdeterministischer endlicher Automat (NEA)

Unterschied zum DEA: Übergangsfunktion δ

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

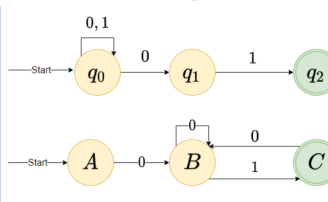
Ein ε -NEA erlaubt zusätzlich noch ε -Übergänge



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, ε -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 Σ

- 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: $L_1 \cdot L_2 = L_1 L_2 = \{\omega = \omega_1 \omega_2 \mid \omega_1 \in L_1 \wedge \omega_2 \in L_2\}$
- Kleensche 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, Σ, P, A) mit

- N : Alphabet der Nichtterminale (Variablen)
- Σ : 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 α, β und γ Satzformen und $A \rightarrow \gamma$ eine Produktion.

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

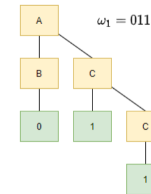
Ableitungsbaum (Parsebaum)

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 $\forall L \exists \text{KFG}$

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

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

Γ : 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 ω = 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

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

Γ 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 Σ^* .
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

Berechnungsmodelle

Turing-berechenbar = kann von Turing-Maschine gelöst werden
Turing-berechenbare Funktion $T : \Sigma^* \rightarrow \delta^*$

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

\forall algorithmisch lsbare Problem ist turing-berechenbar \Rightarrow Computer \equiv TM

Primitiv rekursive Grundfunktionen $\forall n \in \mathbb{N}, \forall k \in \mathbb{N}$ (k = Konstante)

n -stellige konstante Funktion: $c_k^n = \mathbb{N}^n \rightarrow \mathbb{N}$ mit $c_k^n(x_1, \dots, x_n) = k$

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

n -stellige Projektion auf die k -te Komponente:

$\pi_k^n : \mathbb{N}^n \rightarrow \mathbb{N}$ mit $\pi_k^n(x_1, \dots, x_k, \dots, x_n) = k$ ($1 < k < n$)

n = Anzahl der Argumente, k = Position des Arguments

LOOP, WHILE, GOTO

Variablen: x_0, x_1, x_2, \dots
Konstanten: $0, 1, 2, \dots$
Operationszeichen: $+, -$
Trennzeichen: ;

Loop (primitiv-rekursiv)

Schlsselwrter: Loop, Do, End
Sequenzen: P und $Q \rightarrow P; Q$
Schleifen: Loop x do P ... End

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

While (Turing vollstndig)

Erweiterung der Sprache Loop
While $xi > 0$ Do ... End

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

GoTo (Turing vollstndig)

Schlsselwrter: Goto, If, Else
Marker Mk: M1, M2, ...
Sprunganweisung:
If $xi = c$ Then Goto Mk

```
M1: x0 = x3 + 0
M2: If x1 = 0 Then Goto M4
M3: x0 = x2 + 0
M4: Halt
```

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 $A \subseteq \Sigma^*$

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

$\bar{A} = \Sigma^* \setminus A = \{\omega \in \Sigma^* \mid \omega \notin A\}$ (Komplement von A)

Entscheidbarkeit mit Turingmaschinen

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

- Bandinhalt $x \in A$ T hlt mit Bandinhalt «1» (Ja) an
- Bandinhalt $x \in \Sigma^* \setminus A$ T hlt mit Bandinhalt «0» (Nein) an

quivalente Aussagen: $A \subseteq \Sigma^*$ ist entscheidbar

- Es existiert eine TM, die das Entscheidungsproblem $T(\Sigma, A)$ lst
- Es existiert ein WHILE-Programm, dass bei einem zu A gehrenden Wort stets terminiert \rightarrow Entscheidungsverfahren fr A

Semi-Entscheidbarkeit mit Turingmaschinen

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

- Bandinhalt $x \in A$ T hlt mit Bandinhalt «1» (Ja) an
- Bandinhalt $x \in \Sigma^* \setminus A$ T hlt nie an

quivalente Aussagen: $A \subseteq \Sigma^*$ ist semi-entscheidbar

- $A \subseteq \Sigma^*$ ist rekursiv aufzhlbar
- 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 gehrenden Wort stets terminiert und bei Eingabe von Wrtern die nicht zu A gehren nicht terminiert

Reduktion $A \preceq B \Rightarrow A \subseteq \Sigma^*$ reduzierbar auf $B \subseteq \Gamma^*$

Gilt wenn $\exists T : \Sigma^* \rightarrow \Gamma^*$ so dass: $\forall \omega \in \Sigma^* \quad \omega \in A \Leftrightarrow F(\omega) \in B$

T := total Turing-berechenbare Funktion

Eigenschaften:

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

$A \preceq B \Rightarrow$ Entscheidbarkeit von B gleich wie von A

Halteproblem Ist es mglich einen Algorithmus zu schreiben, der fr jede TM entscheiden kann, ob sie hlt oder nicht? (Nein!)

Halteprobleme (HP) definiert als Sprachen: ($\#$ = Delimiter)

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

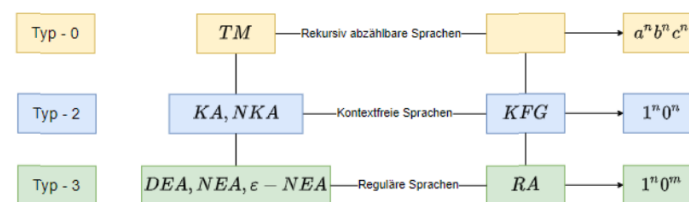
Leeres $H_0 := \{\omega \in \{0, 1\}^* \mid T_\omega \text{ angesetzt auf das leere Band hlt}\}$

Spezielles $H_S := \{\omega \in \{0, 1\}^* \mid T_\omega \text{ angesetzt auf } \omega \text{ hlt}\}$

$H_0 \preceq H_S \preceq H$

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

Chromsky Hierarchie



Komplexittstheorie

Quantitative Gesetze und Grenzen

- Zeitkomplexitt: Laufzeit des besten Programms
- Platzkomplexitt: Speicherplatz des besten Programms
- Beschreibungskomplexitt: Lnge des krzesten Programms

Zeitbedarf von M auf Eingaben der Lnge $n \in \mathbb{N}$

im schlechtesten Fall: $\text{Time}_M(n) = \max \{ \text{Time}_M(\omega) \mid |\omega| = n \}$

Sei M eine TM, die immer hlt und sei die Eingabe $\omega \in \Sigma^*$

Zeitbedarf von M auf ω : $\text{Time}_M(\omega) = \text{Anzahl Konfigurationsbergnge in der Berechnung von } M \text{ auf } \omega$

P vs NP Klassifizierung von Problemen

Ein Problem U heisst in Polynomzeit lsbar, wenn es eine obere Schranke $O(n^c)$ gibt fr eine Konstante $c \geq 1$.

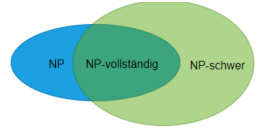
- $P \doteq$ Lsung finden in Polynomzeit
- $NP \doteq$ Lsung verifizieren in Polynomzeit

ACHTUNG: NP heisst nicht «nicht-polynomial» sondern «nicht-deterministisch polynomial»

NP-schwer und NP-vollstndig

Eine Sprache L heisst NP-schwer, falls fr alle Sprachen $L' \in NP$ gilt, dass $L' \preceq_p L$

Eine Sprache L heisst NP-vollstndig, falls $L \in NP$ und L ist NP-schwer.



$P \subset NP$ gilt, aber $P \neq NP$ noch nicht bewiesen

Polynomzeit-Verifizierer berprft die Eingaben in einem Problem
Zeuge: Informationen einer gltigen Eingabe

Asymptotische Komplexittsmessung O -Notation

- $f \in O(g)$: $f(n) \leq c \cdot g(n)$
 - f wchst asymptotisch nicht schneller als g
- $f \in \Omega(g)$: $f(n) \geq \frac{1}{c} \cdot g(n)$
 - f wchst 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 fr die Zeitkomplexitt von U

- $O(f(n))$ ist eine obere Schranke, falls eine TM existiert, die U lst und eine Zeitkomplexitt in $O(f(n))$ hat.
- $\Omega(g(n))$ ist eine untere Schranke, falls fr alle TM M , die U lsen, gilt dass $\text{Time}_M(n) \in \Omega(g(n))$

Rechenregeln

- Konstante Vorfaktoren c ignorieren ($c \in O(1)$)
- Bei Polynomen ist nur die hchste Potenz entscheidend
- Transitiv: $f(n) \in O(g(n)) \wedge g(n) \in O(h(n)) \rightarrow f(n) \in O(h(n))$

bersicht wichtigste Laufzeiten

$O(1) < O(\log \log n) < O(\sqrt{\log n}) < O(\log \sqrt{n}) < O(\log n)$
 $< O(\sqrt{n}) < O(n) < O(n \log n) < O(n^2) < O(n^3)$
 $< O(n^c) < O(2^n) < O(n!) < O(n^n)$

good luck <3