Theoretische Informatik

Jil Zerndt
FS 2024

Alphabete, Wörter, Sprachen

Alphabete endliche, nichtleere Mengen von Symbolen.
- $\Sigma_{\text{Bool}} = \{0,1\}$ Boolsches Alphabet

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

Wort endliche Folge von Symbolen eines bestimmten Alphabets.

Schreibweisen $|\omega|$ = **Länge eines Wortes**
$|\omega|_x$ = **Häufigkeit eines Symbols** x in einem Wort
ω^R = **Spiegelwort/Reflection** zu ω

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 Σ^k = Wörter der Länge $\boldsymbol{k}$ über Alphabet Σ
- $\Sigma^* = \underbrace{\Sigma^0}_{1} \cup \underbrace{\Sigma^1} \cup \underbrace{\Sigma^2} \cup \underbrace{\Sigma^3} \cdots$ Kleensche Hülle
- $\Sigma^+ = \underbrace{\Sigma^1}_{2} \cup \underbrace{\Sigma^2}_{4} \cup \underbrace{\Sigma^3}_{8} \ldots = \Sigma^* \backslash \{\varepsilon\}$ Positive Hülle

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

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

Wortpotenzen Sei x ein Wort über einem Alphabet Σ
- $x^0 = \varepsilon$
- $x^{n+1} = x^n \circ x = x^n x$

Sprache über Alphabet Σ = 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: $AB = \{uv \mid u \in A \text{ und } v \in B\}$
Kleenesche Hülle A^* von A: $\{\varepsilon\} \cup A \cup AA \cup AAA \cup \ldots$

Reguläre Ausdrücke Wörter, die Sprachen beschreiben

$\boldsymbol{RA_\Sigma}$ Sprache der Regulären Ausdrücke über $\{\emptyset, \epsilon, *, (), , |\} \cup \Sigma$
- $R \in RA_\Sigma \Rightarrow (R^*) \in RA_\Sigma$
- $R, S \in RA_\Sigma \Rightarrow (RS) \in RA_\Sigma$
- $R, S \in RA_\Sigma \Rightarrow (R \mid S) \in RA_\Sigma$
- $\emptyset, \epsilon \in RA_\Sigma$
- $\Sigma \subset RA_\Sigma$

Priorisierung von Operatoren
(1) $*$ = Wiederholung $\rightarrow$ (2) Konkatenation $\rightarrow$ (3) $|$= Oder
Erweiterter Syntax
$R^+ = R(R^*)$ $R? = (R \mid \epsilon)$ $[R_1, \ldots, R_k] = R_1 \mid R_2 \mid \ldots \mid R_k$

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: $L(\varepsilon) = \{\varepsilon\}$
- 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 in R oder S: $L(R \mid S) = L(R) \cup L(S)$

Endliche Automaten

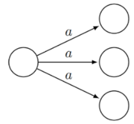
Endliche Automaten Maschinen, die Entscheidungsprobleme lösen
- Links nach rechts
- Keinen Speicher
- Keine Variablen
- Speichert aktuellen Zustand
- Ausgabe über akzeptierende Zustände

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

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

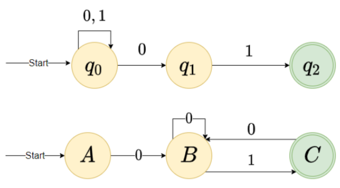
Berechnungsschritt $\vdash_M$ von M $(q, \omega) \vdash_M (p, x)$
Berechnung ist eine endliche Folge von Berechnungsschritten
$(q_a, \omega_1 \omega_2 \ldots \omega_n) \vdash_M \ldots \vdash_M (q_e, \omega_j \ldots \omega_n) \rightarrow (q_a, \omega_1 \omega_2 \ldots \omega_n) \vdash_M^* (q_e, \omega_j \ldots \omega_n)$

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



Teilmengenkonstruktion $\forall$ NEA kann in DEA umgewandelt werden
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

$\downarrow$	$\boldsymbol{q}$	$\boldsymbol{\delta(q,0)}$	$\boldsymbol{\delta(q,1)}$
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 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:
  $L_1 \cdot L_2 = L_1 L_2 = \{\omega = \omega_1 \omega_2 \mid \omega_1 \in L_1 \wedge \omega_1 \in L_2\}$
- Kleenesche Hülle:
  $L^* = \{\omega = \omega_1 \omega_2 \ldots \omega_n \mid \omega_i \in L \text{ für alle } i \in \{1, 2, \ldots, n\}\}$

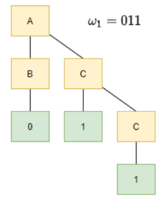
$L(R_1)$: Menge der ganzen Zahlen in Dezimaldarstellung
- $((- \mid \varepsilon)(1,2,3,4,5,6,7,8,9)(0,1,2,3,4,5,6,7,8,9) \mid 0).0$

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 $\alpha \rightarrow \cdots \rightarrow \omega$

Ableitungsbaum (Parsebaum)
mögliche Darstellung einer Ableitung
- $G_1 = \{\{A, B, C\}, \{0, 1\}, P, A\}$
- $P = \{A \rightarrow BC, B \rightarrow 0B|0|\varepsilon, C \rightarrow 1C|1|\varepsilon\}$



Ableitung von $\omega_1 = 011$
- $A \rightarrow BC \rightarrow 0AA \rightarrow 01C \rightarrow 011 \rightarrow \ldots$
  $\rightarrow 011$

Mehrdeutige KFG $\exists$ Wort, das mehrere Ableitungsbäume besitzt.
Mehrdeutigkeiten eliminieren:
- Korrekte Klammerung vom Benutzer erzwingen
- Grammatik anpassen
- Den Produktionen einen Vorrang vergeben

Reguläre Srache 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): $M = \left(Q, \Sigma, \boldsymbol{\Gamma}, \boldsymbol{\delta}, q_0, \$, F\right)$
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: $\boldsymbol{\delta} : \boldsymbol{Q} \times (\boldsymbol{\Sigma} \cup \varepsilon) \times \boldsymbol{\Gamma} \rightarrow \boldsymbol{Q} \times \Gamma^*$
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 $\delta(q, b, c) = (p, \omega)$: $\boldsymbol{q - b, c/\omega \longrightarrow p}$

Berechnungsschritt $\delta(q, b, c) = (p, \omega)$ wird wie folgt interpretiert:
q = Aktueller Zustand
b = Symbol der Eingabe
c = Symbol wird entfernt
ω = Wort auf Stack geschrieben
p = Neuer Zustand

Sprache $L(M)$ eines 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.

Turingmaschinen

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

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

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
Y = Overwrite
$q, p \in Q$ und $X, Y \in \Gamma$
$\boldsymbol{q - X/Y, D \to p}$

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

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

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

Berechnungsmodelle

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

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

Jedes algorithmisch lösbare Berechnungsproblem 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: $c_k^n = \mathbb{N}^n \to \mathbb{N}$ mit $c_k^n(x_1, ..., x_n) = k$

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

n-stellige Projektion auf die k-te Komponente: $(1 < k < n)$

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

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

While (Turing vollständig)
Erweiterung deer Sprache Loop
- While $x_i > 0$ do ... until 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

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

Reduktion $A \preccurlyeq B$ $\Rightarrow A \subset \Sigma^*$ reduzierbar auf $B \subset \Gamma^*$
Gilt wenn $\exists\, T : \Sigma^* \to \Gamma^*$ so dass: $\forall \omega \in \Sigma^*$ $\omega \in A \Leftrightarrow F(\omega) \in B$
T := total Turing-berechenbare Funktion

Eigenschaften:
Transitiv: $A \preccurlyeq B$ und $B \preccurlyeq C \to A \preccurlyeq C$
$A \preccurlyeq B \Rightarrow$ Entscheidbarkeit von B gleich wie von A

Halteproblem Ist es möglich einen Algorithmus zu schreiben, der für jede TM entscheiden kann, ob sie hält oder nicht? (Nein!)

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

Allgemeines $H := \{\omega \# x \in \{0, 1, \#\}^* \mid T_\omega$ angesetzt auf x hält$\}$
Leeres $H_0 := \{\omega \in \{0, 1\}^* \mid T_\omega$ angesetzt auf das leere Band hält $\}$
Spezielles $H_S := \{\omega \in \{0, 1\}^* \mid T_\omega$ angesetzt auf ω hält $\}$
$H_0 \preccurlyeq H_S \preccurlyeq H$
H_0, H_S und H sind semi-entscheidbar und nicht entscheidbar.

Komplexitätstheorie

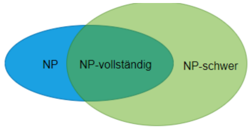
Quantitative Gesetze und Grenzen
- Zeitkomplexität: Laufzeit des besten Programms
- Platzkomplexität: Speicherplatz des besten Programms
- Beschreibungskomplexität: Länge des kürzesten Programms

Zeitbedarf von M auf Eingaben der Länge $n \in \mathbb{N}$
im schlechtesten Fall: $\text{Time}_M(n) = \max\left\{\text{Time}_M(\omega) || \omega |= n\right\}$

Sei M eine TM, die immer hält und sei die Eingabe $\omega \in \Sigma^*$
Zeitbedarf von M auf ω: Time $_M(\omega)$ = Anzahl Konfigurationsübergänge in der Berechnung von M auf ω

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
- $\mathbb{N}P \doteq$ Lösung verifizieren in Polynomzeit

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



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

Asymptotische Komplexitätsmessung O-Notation
- $f \in O(g)$: $f(n) \leq c \cdot g(n)$
  – f wächst asymptotisch nicht schneller als g
- $f \in \Omega(g)$: $f(n) \geq \frac{1}{d} \cdot g(n)$
  – 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 $\text{Time}_\text{M}(n) \in \Omega(g(n))$

Rechenregeln
- Konstante Vorfaktoren c ignorieren ($c \in O(1)$)
- Bei Polynomen ist nur die höchste Potenz entscheidend
- Transitiv: $f(n) \in O(g(n)) \land g(n) \in O(h(n)) \to f(n) \in O(h(n))$

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