Zusammenfassung

01 March 2024 08:26

Alphabet

- endliche, nicht leere Menge von Symbolen
- ε (leeres Wort) über jedem Alphabet
- z.B. $\Sigma = \{a, b, c\}$, $|\Sigma| = 3$

Wort

- endliche Folge von Symbolen
- $|wort|$ => länge eines Wortes (4)
  - ↳ $|wort|_o$ => Häufigkeit von o (1)
- Σ^n => Menge aller Wörter mit Länge n
  - ↳ $\Sigma^2 = \{aa, ab, \ldots cc\}$ (Σ von oben)
  - ↳ $|\Sigma^n| = 3^n$
  - ↳ $\Sigma^0 = \{\varepsilon\}$
- Kleenesche Hülle => Menge aller Wörter über Σ
  - ↳ $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \ldots \cup \Sigma^n$
  - ↳ $\Sigma^+ = \Sigma^* \setminus \Sigma^0 = \Sigma^1 \cup \ldots \cup \Sigma^n$
- $wort^R$ => Spiegelwort (Polindrom) (trow)
- Infix (Teilwort) => $xvy = w$
  - ↳ ε ist auch Teilwort
  - ↳ echtes Teilwort wenn x, y nicht leer
- Präfix => $vx = w$
  - ↳ ε ist auch Präfix
  - ↳ echtes Präfix wenn x nicht leer
- Suffix => $xv = w$
  - ↳ ε ist auch Suffix
  - ↳ echtes Suffix wenn x nicht leer

Sprache

- Teilmenge von Wörtern über einem Alphabet
  - ↳ $L \subseteq \Sigma^*$, Sprache über Σ
- $\varepsilon \notin$ jeder Sprache
- $\emptyset = \{\} =$ leere Sprache über jedem Σ
- können aus unendlich vielen Wörtern bestehen
- z.B. $L = \{w \mid w$ enthält n Einsen gefolgt von n Nullen für $n \in \mathbb{N}\}$
  - $\Rightarrow L = \{1^n 0^n \mid n \in \mathbb{N}\}$
- A und B sei Sprache -> $AB = \{uv \mid u \in A$ und $v \in B\}$
- Kleenesche Hülle von Sprache A
  - $A^* = \{\varepsilon\} \cup A \cup AA \cup AAA \ldots$ => Beliebig häufig

Regulärer Ausdruck

- RA_Σ = Menge Regulärer Ausdrücke über Σ
- $a \in \Sigma$ = Wort in Sprache
- $a, b \in RA \}$ => $a|b \in RA$ (a oder b)
  - => $ab \in RA$ (a und b)
- $a \in RA$ => $a^* \in RA$ (beliebig oft)
- $* >$ "konkatination" $> |$
- $\varepsilon, \emptyset \in RA$
- $R^+ = R(R^*)$, $R? = R|\varepsilon$
  - $R_1 = (0|1)^* 00 (0|1)^*$ = "Regulärer Ausdruck"

=> $L(R_1) = \{xwv \mid w = 00, x, v \in \{0,1\}^*\}$ = "Reguläre Sprache"

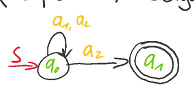
- Reguläre Sprache: Durch RA beschrieben, Durch EA akzeptiert
- Ist $a^* b$ RA_Σ? $a \in RA_\Sigma$ => $a^* \in RA_\Sigma$, $b \in RA_\Sigma$ => $a^* b \in RA_\Sigma$

Endliche Automaten

- Moore: Ausgabe Abhängig vom Zustand (Output im State)
- Mealy: Ausgabe Abhängig von Input und Zustand (Output auf Übergang)

$M = (\{q_0, q_1\}, \{a_1, a_2\}, \delta, q_0, \{q_1\})$

$\delta(q_0, a_1) = q_0$, $\delta(q_0, a_2) = q_0$, $\delta(q_0, a_2) = q_1$



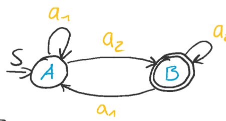
$(q_0, a_1 a_1 a_2) \vdash_M (q_0, a_1 a_2) \vdash_M (q_0, a_2) \vdash_M (q_1, \varepsilon)$ ✓

$\vdash_M (q_0, \varepsilon)$ ✗

- Endzustand => Wort ist leer
- DEA <=> NEA <=> ε-NEA <=> RA

1. Teilmengen konstruktion (ε-eliminieren)
2. Unerreichbare Zustandmengen Rausstreichen
3. Neue Zustände definieren

	Zustand	a_1	a_2	
	~~$\emptyset$~~	~~$\emptyset$~~	~~$\emptyset$~~	II
A ⇐	$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$	
	~~$\{q_1\}$~~	~~$\emptyset$~~	~~$\emptyset$~~	I
B ⇐	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0, q_1\}$	



- Zustandsklassen für EA M

Klasse $[P] = \{w \in \Sigma^* \mid M$ endet nach lesen von w in $P\}$

- Beweis für mind. 3 Zustände $L = \{x11y \mid x, y \in \{0,1\}^*\}$

1. Zustands klassen finden

Initial	Erstes 1	Zweites 11
$X_1 = \varepsilon$	$X_2 = 1$	$X_3 = 11$

2. Widerspruchbeweis -> Wort kann nach lesen nicht in 2 Zuständen sein

Z_{ab} => Zustand $a \neq$ Zustand b

$Z_{12} = 1$ $X_1 Z_{12} = 1 \notin L$, $X_2 Z_{12} = 11 \in L$

$Z_{13} = \varepsilon$ $X_1 Z_{13} = \varepsilon \notin L$, $X_3 Z_{13} = 11 \in L$

$Z_{23} = \varepsilon$ $X_2 Z_{23} = 1 \notin L$, $X_3 Z_{23} = 11 \in L$

3. Beweis formulieren: Jeder EA für L hat mind. 3 Zustände

Kontextfreie Grammatik

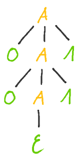
$G = (N, \Sigma, P, A)$

Nichtterminale (Variablen) — Alphabet der Terminale
Produktionen — Startsymbol

$G = (\{A\}, \{0,1\}, P, A)$ für $L = \{ w \in \Sigma^* \mid A \overset{*}{\Rightarrow} w \} = \{ 0^n 1^n \mid n \in N \}$

$P = \{ A \to 0A1,\ A \to \varepsilon \}$

- Ableitung $w = 0011 \Rightarrow A \to 0A1 \to 00A11 \to 00\varepsilon 11 \to 0011$



- w ist ableitbar wenn G existiert: $A \overset{*}{\Rightarrow} w$
  w von A erzeugt/generiert
- mehrdeutig: w mit mehreren Ableitungsbäumen

- Reguläre Sprache $\Rightarrow$ KFG
  1. Pro Zustand q_n ein Nichtterminal
  2. Pro Transition δ eine Produktion
  3. Pro akzeptiertem Zustand eine Produktion $Q_i \to \varepsilon$
  4. Q_0 zu Startsymbol A
- ! Grammatik darf keine Wörter generieren die nicht in der Sprache sind

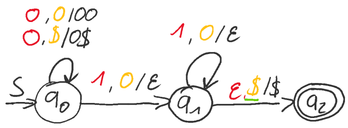
Kellerautomaten

- DKA < NKA (Palindrome $\Rightarrow$ unbekannte Mitte)
- Sprache kontextfrei: NKA existiert für die Sprache
- Endzustand $\Rightarrow$ Wort ist leer, Stack egal
- Achtung: ε - Übergänge bei DKA

$L = \{ 0^n 1^n \mid n > 0 \}$

$M = (\{q_0, q_1, q_2\}, \Sigma, \Gamma, \delta, q_0, \$, \{q_2\})$

Σ Inputalphabet
Γ Kelleralphabet ($\$ \in \Gamma$)



$\delta(q_0, 0, 0) = (q_0, 0\$)$, $\delta(q_0, 0, 0) = (q_0, 00)$, $\delta(q_0, 1, 0) = (q_1, \varepsilon)$ ...

$w = 001 \Rightarrow (q_0, 001, \$) \vdash (q_0, 01, 0\$) \vdash (q_0, 1, 00\$)$ ...
$\ldots \vdash (q_1, \varepsilon, 0\$)$ ✗ $\Rightarrow$ keine mögliche Übergänge

Turingmaschine

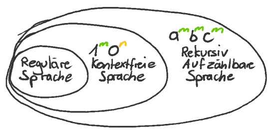
- TM mit Speicher: Zustände Q als kartesisches Produkt $Q = \{q_0 \ldots q_n\} \times \Gamma$
- Mehrspurige (n) TM:
  - Ein oder n Köpfe
  - Einspurige TM mit Band als n-Tupel $\Gamma = \{0,1,\sqcup\} \times \ldots \times \{0,1,\sqcup\}$ (x^n Zustände)
  - Berechnung für alle Bänder
- DTM $\Leftrightarrow$ NTM $\Leftrightarrow$ Semi-Unendliche TM $\Leftrightarrow$ Zähler Maschine

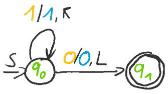
$M = \{Q, \Sigma, \Gamma, \delta, q_0, \sqcup, \{q_1\}\}$

$Q = \{q_0, q_1\}$, $\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, \sqcup\}$

$\delta(q_0, 1) = (q_0, 1, R)$, $\delta(q_0, 0) = (q_1, 0, L)$





$w = 110$

$q_0 110 \vdash 1 q_0 10 \vdash 11 q_0 0 \vdash 1 q_1 10$

- Semi-Unendlich mit 2 Spuren $\Rightarrow$ TM
- k-Stack: 2 Stacks für Link/Rechts vom Band $\Rightarrow$ TM
- Zähler Maschine: Als 2-Stack Maschine $\Rightarrow$ TM
  - Stack Codieren $\Gamma = \{A, B, C\}$ $A \to 1$, $B \to 2$, $C \to 3$, $\$ = 0$
    $\Rightarrow k = 4$

Stack: A C $ — Push: C: $0 \cdot 4 + 3 = 3$; A: $3 \cdot 4 + 1 = 13$ $\Leftrightarrow$ Pop: $13 / 4 = 3$ Rest 1 (A); $3 / 4 = 0$ Rest 3 (C)

Universelle TM:

Kodierungsmöglichkeit:

1. Schritt: Die Zustände Q einer TM werden codiert als:
- q_1: für den Startzustand, 0
- q_2: für den akzeptierenden Zustand[1] und 00
- $q_3, \ldots, q_i$: für alle weiteren Zustände. $00\ldots0^i$

2. Schritt: Die Bandsymbole Γ einer TM werden codiert als:
- X_1: für das Symbol 0, 0
- X_2: für das Symbol 1, 00
- X_3: für das Symbol $\sqcup$ (*Blank*) und 000
- $X_4, \ldots, X_j$: für alle weiteren Symbole. 0^j

3. Schritt: Codierung der Richtung des Lese-Schreibkopfes D einer TM:
- D_1: für die Richtung L und 0 — S: 000
- D_2: für die Richtung R. 00

Trennung Zustände: 11
Trennung Codierung: 111
$\delta(q_1, 1) = (q_3, 0, R)$
0100100010100100011 ...

Berechenbarkeit

- intuitiv berechenbar: algorithmisch berechenbar
- $f(x) = \uparrow$ => Input x nicht definiert (kein Wert)
- Turing-berechenbar: $f(x)$ von TM berechnet
- Loop Programme
  - $x_x = x_n +/- c$
  - Semicolon
- While Programme (While $x_i > 0$ Do ... End)
  - müssen nicht Terminieren
- Goto
  - If, Then, Halt
  - M_k : If $x_i = c$ Then Goto M_r
- Primitiv rekursive Funktion
  - Konstant $c_k^n : N^n \to N$, $c_k^n(x_1, \dots, x_n) = k$
  - Nachfolgerfunktion $\eta : N \to N$, $\eta(x) = x+1$
  - Projektion $(1 \le k \le n)$, $\pi_k^n : N^n \to N$, $\pi_k^n(x_1, \dots x_k \dots x_n) = x_k$

Bsp Addition:

$$Add(0, y) = y = \pi_1^1(y)$$

$$Add(x+1, y) = Add(x, y) + 1 = \eta(\pi_1^3(Add(x,y), x, y))$$

=> $Add(2,3) = Add(1,3) + 1 = Add(0,3) + 1 + 1 = 3+1+1 = 5$

- Turing-Vollständig $f : N^* \to N$
  - f ist Turing berechenbar
  - f ist While berechenbar
  - f ist Goto berechenbar

Nicht: Loop, primitiv Rekursiv

```
IF x1 = 0 Then P Else Q

x2 = x2 + 1;
x3 = x3 + 1;
Loop x1 Do
   x2 = x2 - 1;
   x4 = x3 + 0
End;
Loop x2 Do
   P
End;
Loop x4 Do
   Q
End
```

Mod(X, Y)

```
x3 = x1 + 1;
Loop x3 Do
      Loop x3 Do
            x0 = x3 + 0
      End;                  } If x3 > 0 (x0 = x3)
      Loop x2 Do
            x3 = x3 - 1     } (x3 - x2)
      End
End;
x0 = x0 - 1
```

Entscheidbarkeit

↳ Sprache $A \subset \Sigma^*$ von Turingmaschine lösbar
- $w \in A$: "Ja"
- $w \notin A$: "Nein"
- muss nach n-Schritten halten
- While-Programm lösbar

- wenn A und $\bar{A}$ semi-Entscheidbar ($\bar{A} = \Sigma^* \setminus A$)
- Semi-Entscheidbar: nur Ja, kein Nein (Endlosloop)
- A ist Entscheidbar => $\bar{A}$ Entscheidbar
- A, B (semi-)Entscheidbar => $A \cup B$, $A \cap B$ (semi-) Entscheidbar

Äquivalente Aussagen:

A ist:
- Turing berechenbar
- rekursiv aufzählbar
- semi-entscheidbar
- Wertebereich total berechenbarer Funktion
- Definitionsbereich berechenbarer Funktion

Reduktion

"Problem mit Lösung von anderem Problem lösen"

- $A \subset \Sigma^*$ auf $B \subset T^*$ reduzierbar wenn
  $F : \Sigma^* \to T^*$ => $A \preceq B$
- Wenn $A \preceq B$ und B (semi-) entscheidbar -> A (semi-)entscheidbar

Bsp: A: n durch 3 Teilbar, B: n durch 6 Teilbar

=> $F(n) = B(n \cdot 2)$ => $A \preceq B$ (A mit B lösbar)

Halteproblem

- Hält Turing-Maschine auf w an? (endlos oder nur sehr lange?)
- allgemeines Halteproblem
- leeres Halteproblem => leeres Band
- spezielle Halteproblem => Input = Definition TM

=> Semi-Entscheidbar

$H_S \preceq H \preceq H_0$

=> nicht Entscheidbar (T entscheided Halteproblem, P prüft T => Widerspruch)

Satz von Rice

Nicht möglich mechanisch zu prüfen:

→ Programm bei jedem Input terminiert

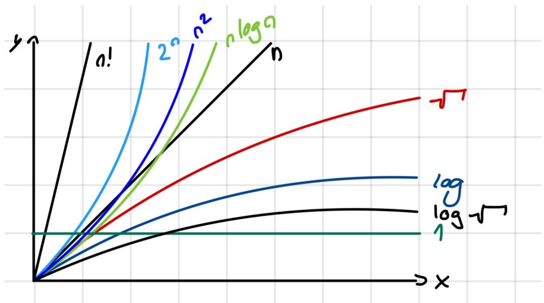
→ zwei Programme gleiche Funktionalität haben

Komplexität

- obere Schranke : $f \in O(g)$ ⇒ f wächst nicht schneller als g

  zB. $f(x) = 3n^3 + 2n - 5$ ⇒ $O(n^3)$

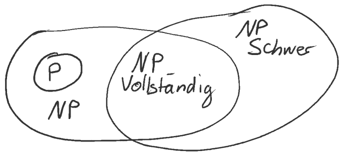
- untere Schranke : $f \in \Omega(g)$ ⇒ f wächst mind. so schnell wie g
- $f \in \Theta(g)$ ⇒ f und g gleiches Wachstum
- Konstante Faktoren ignorieren ⇒ $c = O(1)$



- P ⇒ det. Tm löst A in $O(p)$ (p = Polynom)
- NP ⇒ nicht det. Tm 〃

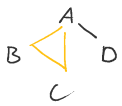
  ⇒ p-Verifizierer ⇒ Alle möglichen Lösungen ermitteln

  ⇒ Lösung verifizieren in Polynomzeit



- P-Reduktion ($\preceq_p$): Funktion um in p-Zeit von A nach B
- NP-Schwer : Jedes Problem von NP auf dieses Reduzierbar
- NP-Vollständig : NP-Schwer und in NP

  ↳ Clique : Enthält ungerichteter Graph Clique von k, k = 3 :



  SAT : Ist Logik-Formel erfüllbar? DNF : $(a \wedge b) \vee (c \wedge d)$

  ↳ SAT $\preceq_p$ Clique ⇒ Verbinden wenn keine Negation

  $(x_1 \vee x_2) \wedge (\overline{x_1}) \wedge (\overline{x_2})$

  ⇒ keine Clique ⇔ nicht erfüllbar