Einführung in die Theoretische Informatik
Zusammenfassung

Ali, Mihir, Noah

May 12, 2022

Contents

1 Formale Sprachen

1.1 Grundbegriffe

- Alphabet Σ (endliche Menge) z.B. $\{1, 0\}$
- Wort/String über Σ ist eine endliche Folge von Zeichen aus Σ
- $|w|$ länge des Wortes w
- Leeres Wort ϵ
- uv konkatenation der Wörter u und w
- Ist w ein Wort so ist $w^0 = \epsilon$ und $w^{n+1} = ww^n$
- Σ^* Menge aller Wörter über Σ
- (formale) Sprache $L \subseteq \Sigma^*$

1.1.1 Operationen auf Sprachen

Seien $A, B \subseteq \Sigma^*$

- Konkatenation:

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

- Konkatenation mit sich selbst:

$A^n = \{w_1...w_n|w_1, ..., w_n \in A\} = A...A$

- $A^* = \{w_1...w_n|n \geq 0 \wedge w_1, ..., w_n \in A\} = \bigcup_{n\in\mathbb{N}} A^n$
- $A^+ = AA^* = \bigcup_{n\geq 1} A^n$

1. Sonderfälle:
   - $\forall A : \epsilon \in A^*$
   - $\emptyset^* = \{\epsilon\}$
   - $\emptyset A = \emptyset$
   - $\{\epsilon\}A = A$
   - $A^*A^* = A^* = (A^*)^*$

1.1.2 Grammatiken

4-Tupel $G = (V, \Sigma, P, S)$

- V ist endliche Menge von Nichtterminalzeichen
- Σ ist endliche Menga von Terminalzeichen (= Alphabet)
- $P \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ ist Menge von Produktionen
- $S \in V$ ist das Startsymbol

Die Sprache von G ist die Menge aller Wörter, die von G erzeugt werden. Sie wird mit $L(G)$ bezeichnet. Also jedes Wort, dass die Grammatik erzeugt muss in der Sprache erhalten sein und jedes Wort in der Sprache muss von der Grammatik erzeugt werden.

1. Reflexve transitive Hülle

   - $\alpha \rightarrow_G^0 \alpha$
   - $\alpha \rightarrow_G^{n+1} \gamma : \exists \beta . \alpha \rightarrow_G^n \rightarrow_G \gamma$
   - $\alpha \rightarrow_G^* \beta : \exists n . \alpha \rightarrow_G^n \beta$
   - $\alpha \rightarrow_G^+ \beta : \exists n > 0 . \alpha \rightarrow_G^n \beta$

1.1.3 Chomsky Hierarchie

Eine Grammatik G ist vom

- Typ 0 immer
- Typ 1 falls fpr jede Produktion $\alpha \rightarrow \beta$ außer $S \rightarrow \epsilon$ gilt $|\alpha| \leq |\beta|$
- Typ 2 Falls G vom typ 1 ist und für jede Produktion $\alpha\beta$ gilt $\alpha \in V$
- Typ 3 falls G vom Typ 2 ist und für jede Produktion $\alpha \rightarrow \beta$ außer $S \rightarrow \epsilon$ gilt $\beta \in \Sigma \cup \Sigma V$

1. Grmmatiken und Sprachklassen:

Typ 3	Rechtslineare Grammatiken	Reguläre Sprachen
Typ 2	Kontextfreie Grammatik	Kontextfreie Sprachen
Typ 1	Kontextsensitive Grammatik	Kontextsens. Sprachen
Typ 0	Phrasenstrukturgrammatik	Rekursiv aufzählbare Sprachen

2. Satz 2.13 $L(Typ3) \subset L(Typ2) \subset L(Typ1) \subset L(Typ0)$

1.1.4 Wortproblem

Gegeben: eine Grammatik G, ein Wort $w \in \Sigma^*$ Frage: Ist das Wort in w enthalten ($w \in L(G)$)?

2 Reguläre Sprachen

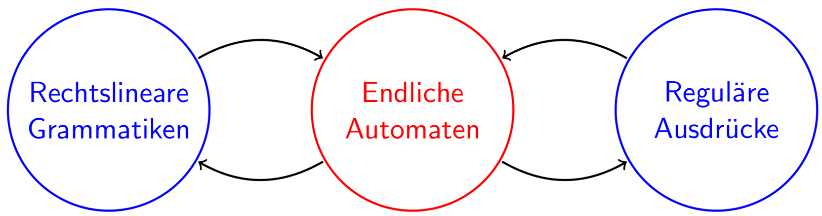

Figure 1: Reguläre Sprachen Schema

2.1 Deterministische endliche Automaten

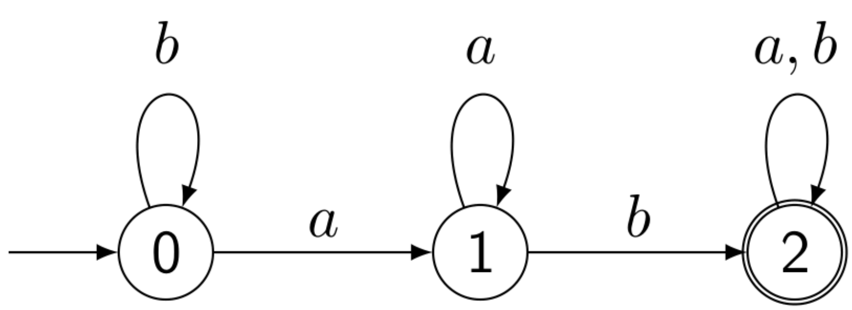

Figure 2: Beispiel Automat

- Beispiel:
  - Eingabewort *baba* $\rightarrow$ Zustandsfolge 0,0,1,2,2
- "Bei dieser Grammatik muss mindestens nach einem a ein b kommen"
- Die Sprache des DFA ist die Menge aller Wörter über $\{a, b\}$, die ab enthalten

Erkannte Sprache: Menge der Wörter, die vom Startzustand in einen Endzustand führen. Recognizer, die nur einmal das Wort durchläuft und in linearer Zeit es akzeptiert oder ablehnt.

2.1.1 Definition

Ein deterministischer endlicher Automat $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- endliche Menge von Zuständen Q
- endlichem Eingabealphabet Σ
- einer totalen Übergangsfunktion $\delta : Q \times \Sigma \to Q$
- eienm Startzustand $q_0 \in Q$
- einer Menge $F \subset Q$ von Endzuständen

1. Akzeptierte Sprachen (Definition 3.2) Von M akzeptierte Sprache $L(M) := \{w \in \Sigma^* | \hat{\delta}(q_0, w) \in F\}$ wobei
$\hat{\delta} : Q \times \Sigma^* \to Q$ induktiv definiert ist:
$\hat{\delta}(q, \epsilon) = q$
$\hat{\delta}(q, aw) = \hat{\delta}(\delta(q, a), w)$, für $a \in \Sigma, w \in \Sigma^*$
($\hat{\delta}(q, w)$ bezeichnet den Zustand, den man aus q mit w erreicht.)
Eine Sprache ist regulär **gdw** sie von einem DFA akzeptiert wird.
2. Beispiel Automat der Sprache akzeptiert Induktiv beweisen pro Zustand.

2.2 Von rechtslinearen Grammatiken zu DFA

- Für jede rechtslineare Grammatik G gibt es einen DFA M mit $L(M) = L(G)$
- Für jeden DFA M gibt es eine rechtslineare Grammatik G mit $L(G) = L(M)$

2.2.1 Nichtdeterministischer endlicher Automat

Ein deterministischer endlicher Automat $M = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- Q, Σ, q_0, F sind wie DFA

- $\delta : Q \times \Sigma \to P(Q)$
  $P(Q)$ = Menge aller Teilmengen von $Q = 2^Q$
  Alternative: Relation $\delta \subseteq Q \times \Sigma \times Q$

$\bar{\delta}(S, a) := \bigcup_{q \in S} \delta(q, a)$
Es folgt: $\hat{\bar{\delta}} : P(Q) \times \Sigma^* \to P(Q)$

1. Intuition: $\hat{\bar{\delta}}(S, w)$ ist Menge aller Zustände, die sich von einem Zustand in S aus w erreichen lassen.

2. Von nichtdeterminitsichen Automaten N akzeptierte Sprache $L(N) := \{w \in \Sigma^* | \hat{\bar{\delta}}(\{q_0\}, w) \cap F \neq \emptyset\}$

2.2.2 Satz 3.9

Für jede rechtslineare Grammatik G gibt es einen NFA M mit $L(G) = L(M)$

2.2.3 Satz 3.13

Für jeden DFA M gibt es eine rechtslineare Grammatik G mit $L(M) = L(G)$

2.3 3.3 NFAs mit ϵ -Übergängen

Grammatiken von Programmiersprachen enthalten viele Produktionen der Gestalt $A \to B$.

Ein NFA mit ϵ-Übergängen (auch ϵ -NFA) ist ein NFA mit einem speziellen Symbol $\epsilon \notin \Sigma$ und mit $\delta : Q \times (\Sigma \cup \epsilon) \to P(Q)$. Ein ϵ übergang darf ausgef¨uhrt werden, ohne dass ein

2.3.1 Lemma 3.16

Für jeden ϵ -NFA N gibt es einen NFA N' mit $L(N) = L(N')$.

2.4 3.4 Regex

- $\emptyset$ ist ein regex
- ϵ ist ein regex
- Für jedes $a \in \Sigma$ ist a ein regulärer Audruck
- Wenn α und β regex dann auch

1. $\alpha\beta$
2. $\alpha|\beta$
3. α^*

- Sonst NIX!

2.4.1 Definition 3.20

Zu einem regulären Ausdruck γ ist die zugehörige Sprache $L(\gamma)$ rekursiv definiert:

- $L(\emptyset) = \emptyset$
- $L(\epsilon) = \{\epsilon\}$
- $L(a) = a$
- $L(\alpha\beta) = L(\alpha)L(\beta))$
- $L(\alpha|\beta) = L(\alpha) \cup L(\beta)$
- $L(\alpha^*) = L(\alpha)^*$

2.4.2 Satz 3.23 (Kleene 1956)

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann durch einen regulären Ausdruck darstellbar, wenn sie regulär ist.

2.4.3 Wie teuer sind unsere Konversionen?

$$\begin{array}{ccccc} \mathsf{DFA} & \leftarrow & \mathsf{NFA} & \leftarrow & \epsilon\text{-}\mathsf{NFA} \\ & \searrow & \downarrow & \nearrow & \\ & & \mathsf{RE} & & \end{array}$$

- RE $\rightarrow \epsilon$ -NFA: RE der Länge n, $O(n)$ Zustände
- ϵ -NFA $\rightarrow$ NFA: Q
- NFA $\rightarrow$ DFA: $O(2^n)$
- FA $\rightarrow$ RE: $O(n4^n)$

2.5 Abschlusseigenschaften regulärer Sprachen

2.5.1 Satz 3.24

Seien $R, R_1, R_2 \subseteq \Sigma^*$ reguläre Sprachen. Dann sind auch

- R_1R_2
- $R_1 \cup R_2$
- R^*
- $\bar{R}(:= \Sigma^* \backslash R)$
- $R_1 \cup R_2$
- $R_1 \backslash R_2$

reguläre Sprachen

1. Produkt-Konstruktion Für den Schnitt ist die De-Morgan regel zu teuer also kann man auch eine Produkt Konstruktion ohne Umweg über De-Morgen benutzen.

   Das funktioniert über Parallelismus also beide DFAs laufen synchron parallel (kreuzprodukt der Zustandsräume).

2.6 Rechnen mit Regulären Ausdrücken

2.6.1 Definition 3.26

Zwei reguläre Ausdrücke sind **äquivalent gdw** sie die gleiche Sprache darstellen:
$\alpha \equiv \beta :\Leftrightarrow L(\alpha) = L(\beta)$

(by the way $\equiv$ steht für Bedeutungsäquivalenz und $=$ für syntaktische gleichheit)

2.7 Pumping Lemma

Wie zeigt man, dass eine Sprache nicht regulär ist?

2.7.1 Satz 3.32 (Pumping Lemma für Reguläre Sprachen)

Sei $R \subseteq \Sigma^*$ regulär. Dann gibt es ein $n > 0$, so dass sich jedes $z \in R$ mit $|z| \geq n$ so in $z = uvw$ zerlegen lässt, dass

- $v \neq \epsilon$,
- $|uv| \leq n$
- $\forall i \geq 0. uv^i w \in R$.

Es gibt nicht-reguläre Sprachen, für die das Pumping-Lemma gilt! $\Rightarrow$ Pumping-Lemma hinreichend aber nicht notwendig um Nicht-Regularität zu zeigen.

regulär $\subset$ Pumping-Lemma gilt $\subset$ alle Sprachen

2.8 Entscheidungsverfahren

Eingabe: Ein oder mehrere Objekte, die Reguläre Sprachen beschreiben (DFA, NFA, RE Typ3 Gram, ...) **Frage:** Haben die Sprachen die Eigenschaft X? Ein (Entscheidungs-)Problem ist entscheidbar, wenn es einen Algorithmus gibt, der bei jeder Eingabe in endlicher Zeit die richtige Antwort auf die Frage feststellt.

Welche Entscheidungsprobleme sind für rechtslineare Grammatiken entscheidbar und wie hängt die Laufzeit mit der Beschreibung zusammen.

2.8.1 Definition 3.37

Sei D ein DFA, NFA, RE, rechtslineare Grammatik ...

- **Wortproblem**: Gegeben w und D: gilt $w \in L(D)$
- **Leerheitsproblem**: Gegeben D: gilt $\emptyset = L(D)$
- **Endlichkeitsproblem**: Gegeben D: isz $L(D)$ endlich
- **Äquivalenzproblem**: Gegeben D_1, D_2, gilt $L(D_1) = L(D_2)$