

Automaten und Komplexität

SS 20

Prof. Dr. Dietrich Kuske

1 Einführung

Gegenstand der Vorlesung I

Klassifikation der „Entscheidungsprobleme“ bzgl. ihrer „Komplexität“

Teil 1 „Komplexität“ = Art der Beschreibung

- verschiedene Formalismen, um „Probleme“ zu beschreiben
- Vergleich der Formalismen bzgl. Ausdrucksstärke
- Algorithmen für die Manipulation dieser Formalismen

Teil 2 „Komplexität“ = algorithmischer Aufwand zur Lösung des „Problems“

- welche Probleme sind durch einen Algorithmus schnell / nur langsam / gar nicht lösbar?
- führt zur wesentlichen Frage $P \stackrel{?}{=} NP$

Definition 1.1. Ein *Alphabet* ist eine endliche, nichtleere Menge (Elemente: „Symbole“ oder „Buchstaben“)

Beispiele $\{0, 1\}$, $\{0, 1, \dots, 9\}$, $\{a, b, \dots, z\}$, der ASCII-Zeichensatz

Konvention Alphabete: Σ, Γ
Buchstaben: $a, b, c \dots$

Definition 1.2. Sei Σ ein Alphabet.

Ein *Wort über Σ* ist ein Tupel $w = (a_1, a_2, \dots, a_n)$ beliebiger Länge mit Einträgen in Σ .

Länge von w : $|w| = n$

Σ^* ist die Menge aller Wörter über Σ

Beispiele $w = (0, 1, 1, 0, 1, 0, 0) \in \{0, 1\}^*$ mit $|w| = 7$

$w = (a, b, b, a) \in \{a, b, \dots, z\}^*$ mit $|w| = 4$

$\varepsilon = () \in \Sigma^*$ mit $|\varepsilon| = 0$

Konvention „Wort“ = „Wort über Σ “ wenn Σ aus Kontext klar oder unwesentlich
übliche Bezeichnungen: u, v, w

für $w = (a_1, a_2, \dots, a_n)$ schreiben wir auch $a_1 a_2 \cdots a_n$
 $\varepsilon = ()$ heißt *leeres Wort*

Definition 1.3. Sei Σ ein Alphabet.

Eine *formale Sprache* über Σ ist eine Teilmenge von Σ^* .

L ist eine *formale Sprache*, wenn es ein Alphabet Σ gibt, so daß $L \subseteq \Sigma^*$.

Beispiele

- Menge der syntaktisch korrekten C-Programme (Alphabet: ASCII-Zeichensatz)
- Σ^* , \emptyset , $\{\varepsilon\}$
- $\{w \in \{0, 1\}^* \mid |w|_0 = |w|_1\}$, wobei $|w|_a$ die Anzahl der Vorkommen des Buchstaben a im Wort w bezeichnet

Konvention formale Sprachen: K , L
„Sprache“ für „formale Sprache“

Gegenstand der Vorlesung II

Klassifikation der Sprachen bzgl. ihrer „Komplexität“

Teil 1 „Komplexität“ = Art der Beschreibung

- verschiedene Formalismen, um Sprachen zu beschreiben
- Vergleich der Formalismen bzgl. Ausdrucksstärke
- Algorithmen für die Manipulation dieser Formalismen

Teil 2 „Komplexität“ = Aufwand zur Lösung des „Wortproblems“

Definition 1.4. Sei L eine Sprache über dem Alphabet Σ . Mit *Wortproblem für L* bezeichnen wir das folgende algorithmische Problem:

Eingabe: $w \in \Sigma^*$

Frage: Gehört w zu L ?

- für welche Sprachen ist das Wortproblem durch einen Algorithmus schnell / nur langsam / gar nicht lösbar?
- führt zur wesentlichen Frage $P \stackrel{?}{=} NP$

2 Reguläre Sprachen

Idee/Ziel: Beschreibung eines Systems,

- das sich in endlich vielen Zuständen befinden kann,
- das beobachtbare Aktionen ausführen kann
- und dabei Wörter (über den beobachtbaren Aktionen) akzeptiert bzw. ablehnt.

2.1 Deterministische Endliche Automaten (DFAs)

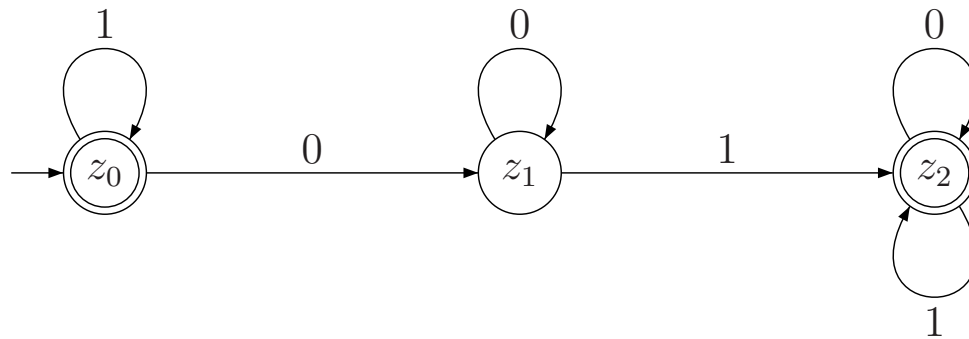
Definition 2.1. Ein *DFA* ist ein Tupel $M = (Z, \Sigma, \iota, \delta, F)$ mit folgenden Komponenten:

- Z ist eine endliche Menge (Elemente: „Zustände“)
- Σ ist ein Alphabet
- $\iota \in Z$ („Initialzustand“)
- $\delta: Z \times \Sigma \rightarrow Z$ („Überföhrungsfunktion“)
- $F \subseteq Z$ („akzeptierende“ oder „Finalzustände“)

Beispiel 2.2. $Z = \{z_0, z_1, z_2\}$, $\Sigma = \{0, 1\}$, $\iota = z_0$, $F = \{z_0, z_2\}$

δ	0	1
z_0	z_1	z_0
z_1	z_1	z_2
z_2	z_2	z_2

anschaulich:



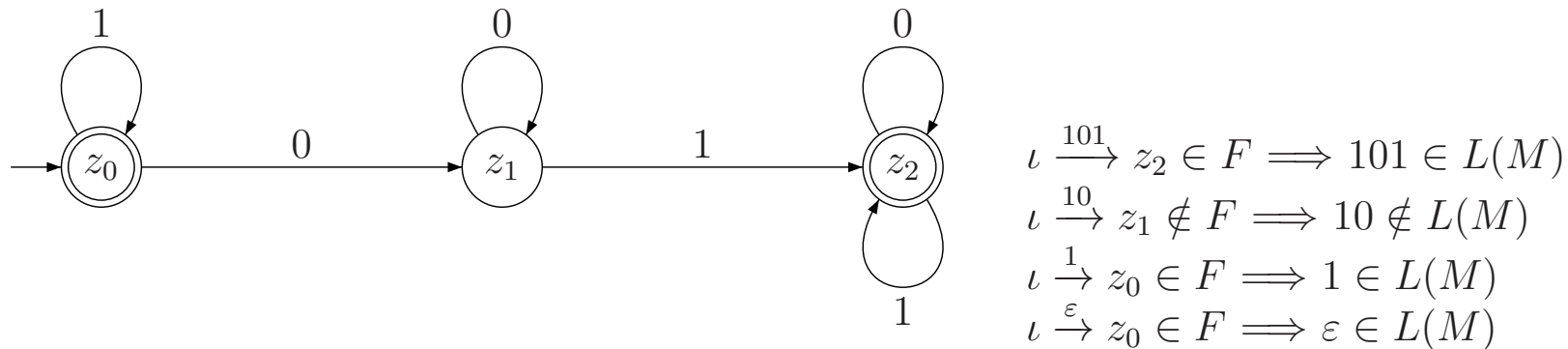
Achtung:

- von *jedem* Zustand muß es für *jeden* Buchstaben *genau eine* ausgehende Kante geben
- es muß *genau einen* Initialzustand geben

Definition 2.3. Sei $M = (Z, \Sigma, \iota, \delta, F)$ ein DFA.

1. Seien $z, z' \in Z$ und $w \in \Sigma^*$. Wir schreiben $z \xrightarrow{w} z'$, wenn der in z startende w -beschriftete Pfad in z' endet
2. Die von M akzeptierte Sprache ist $L(M) = \{w \in \Sigma^* \mid \exists f \in F: \iota \xrightarrow{w} f\}$.
3. Eine Sprache $L \subseteq \Sigma^*$ heißt *regulär*, wenn es einen DFA M gibt mit $L(M) = L$.

Beispiel 2.2 (Fortsetzung)



allgemein: $w \in L(M)$ gdw. $\iota \xrightarrow{w} z_0$ oder $\iota \xrightarrow{w} z_2$ gdw. $w \in \{1\}^*$ oder w enthält 01

Damit: die Menge der Wörter über $\{0, 1\}$, die keine 0 oder aber 01 enthalten, ist regulär.

Lemma 2.4. *Sei Σ ein Alphabet und $L \subseteq \Sigma^*$.
Ist L regulär, so auch $\Sigma^* \setminus L$.*

BewIdee:

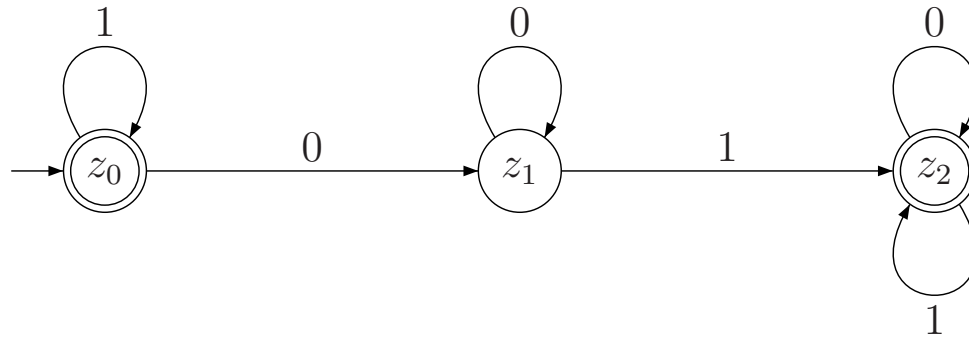
L regulär \implies es gibt DFA $M = (Z, \Sigma, \iota, \delta, F)$ mit $L = L(M)$

betrachte den DFA $M' = (Z, \Sigma, \iota, \delta, Z \setminus F)$. Dann gilt für ein beliebiges Wort $w \in \Sigma^*$:

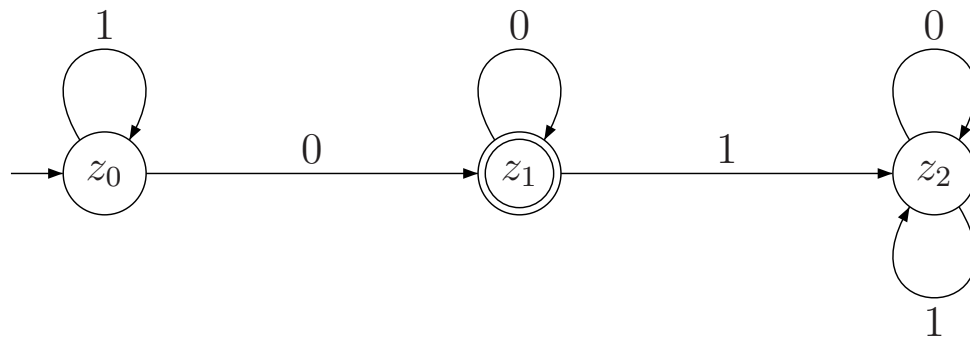
$$\begin{aligned}
 w \in L(M') &\iff \exists z' \in Z \setminus F : \iota \xrightarrow{w} z' \\
 &\iff \exists z' \notin F : \iota \xrightarrow{w} z' \\
 &\iff \text{nicht: } \exists z \in F : \iota \xrightarrow{w} z \\
 &\iff w \notin L(M) \\
 &\iff w \in \Sigma^* \setminus L(M)
 \end{aligned}$$

Also gilt $L(M') = \Sigma^* \setminus L(M) = \Sigma^* \setminus L$. □

Beispiel 2.2 (Fortsetzung)



Der DFA M' sieht wie folgt aus:



Lemma 2.5. *Seien Σ ein Alphabet und $L_1, L_2 \subseteq \Sigma^*$.
Sind L_1 und L_2 regulär, so auch $L_1 \cup L_2$.*

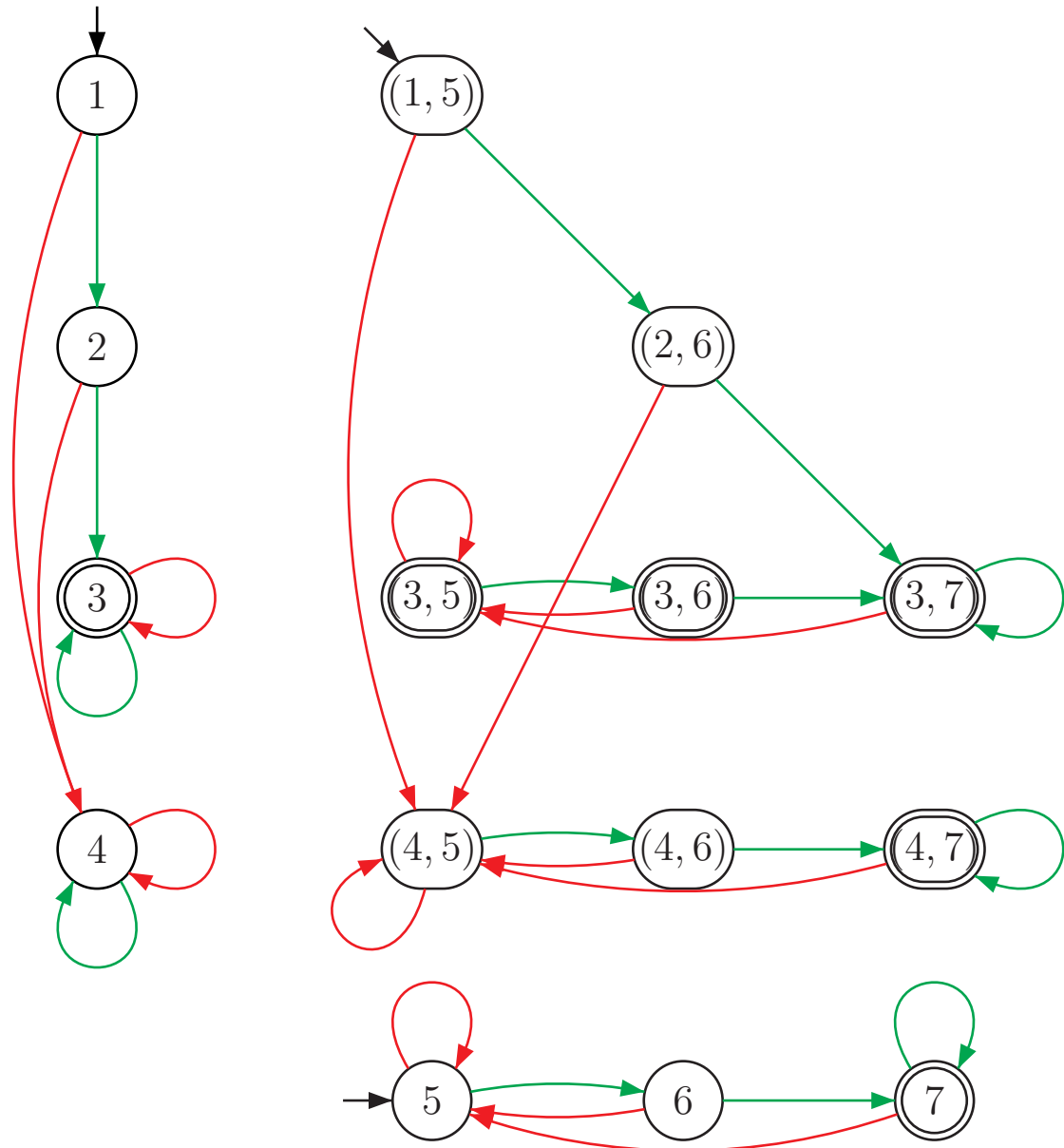
Beweis durch Beispiel:

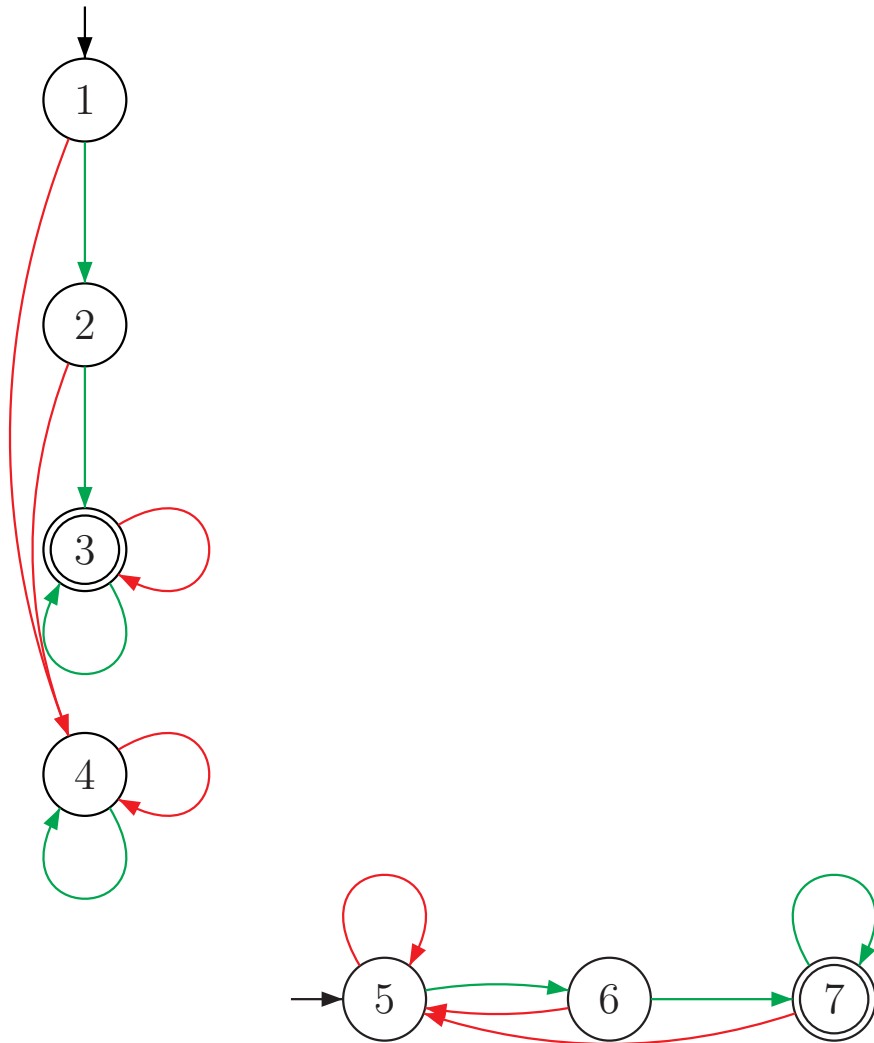
Seien M_1 der linke, M_2 der untere und M_\cup der mittlere DFA im Bild auf der folgenden Seite. Es gelten

$$L(M_1) = \{w \mid w \text{ beginnt mit } gg\}$$

$$L(M_2) = \{w \mid w \text{ endet mit } gg\}$$

$$L(M_\cup) = \{w \mid w \text{ beginnt oder endet mit } gg\} = L(M_1) \cup L(M_2)$$





Wir haben M_{\cup} aus M_1 und M_2 durch den folgenden Algorithmus erhalten:

setze $\iota' = (\iota_1, \iota_2)$, $Z' = \{\iota'\}$ und $F' = \begin{cases} \{\iota'\} & \text{falls } \iota_1 \in F_1 \text{ oder } \iota_2 \in F_2 \\ \emptyset & \text{sonst} \end{cases}$

solange es $(p_1, p_2) \in Z'$, $a \in \Sigma$ gibt mit $(\delta_1(p_1, a), \delta_2(p_2, a)) \notin Z'$

do $Z' := Z' \cup \{(\delta_1(p_1, a), \delta_2(p_2, a))\}$

 if $\delta_1(p_1, a) \in F_1$ oder $\delta_2(p_2, a) \in F_2$ then $F' = F' \cup \{(\delta_1(p_1, a), \delta_2(p_2, a))\}$

enddo

für $(p_1, p_2) \in Z'$, $a \in \Sigma$ setze $\delta'((p_1, p_2), a) = (\delta_1(p_1, a), \delta_2(p_2, a))$ □

Folgerung 2.6. Seien Σ ein Alphabet und $L_1, L_2 \subseteq \Sigma^*$.
Sind L_1 und L_2 regulär, so auch $L_1 \cap L_2$.

Beweis:

L_1, L_2 regulär

$\xrightarrow{\text{Lemma 2.4}} \Sigma^* \setminus L_1, \Sigma^* \setminus L_2$ regulär

$\xrightarrow{\text{Lemma 2.5}} \Sigma^* \setminus L_1 \cup \Sigma^* \setminus L_2$ regulär

$\xrightarrow{\text{Lemma 2.4}} \Sigma^* \setminus (\Sigma^* \setminus L_1 \cup \Sigma^* \setminus L_2)$ regulär

Dies ist aber nach den de Morganschen Regeln gleich $L_1 \cap L_2$. □