Algorithmen, Automaten und Komplexität

(Sommersemester 2023)

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 $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$

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

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

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

Definition 1.2. Sei Σ ein Alphabet.
Ein *Wort über* Σ ist ein Tupel $w = (a_1, a_2, \ldots, a_n)$ 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 $|w| = 7$
$w = (a,b,b,a) \in \{a,b,\ldots,z\}^*$ 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 $w = (a_1, a_2, \ldots, a_n)$ 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^*, $\emptyset$, $\{\varepsilon\}$
- $\{w \in \{0,1\}^* \colon |w|_0 = |w|_1\}$, 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 $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$

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)

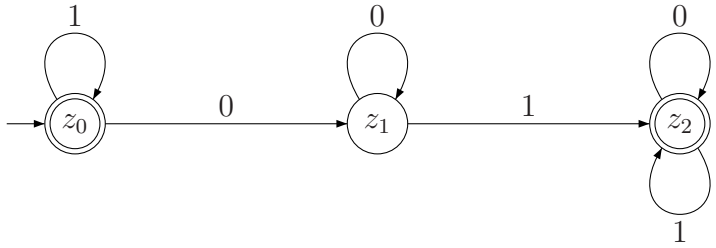
Definition 2.1. Ein *DFA* ist ein Tupel $M = (Z, \Sigma, \iota, \delta, F)$ mit folgenden Komponenten:

- Z ist eine endliche Menge (Elemente: „Zustände“)
- Σ ist ein Alphabet
- $\iota \in Z$ („Initialzustand“)
- $\delta\colon Z \times \Sigma \to Z$ („Überführungsfunktion“)
- $F \subseteq Z$ („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:



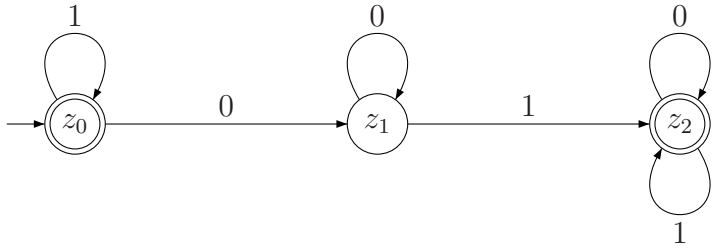
Achtung:

- 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 $L(M) = \{w \in \Sigma^* \mid \exists f \in F \colon \iota \xrightarrow{w} f\}$.
3. Eine Sprache $L \subseteq \Sigma^*$ heißt *regulär*, wenn es einen DFA M gibt mit $L(M) = L$.

Beispiel 2.2 (Fortsetzung)



$\iota \xrightarrow{101} z_2 \in F \Longrightarrow 101 \in L(M)$
$\iota \xrightarrow{10} z_1 \notin F \Longrightarrow 10 \notin L(M)$
$\iota \xrightarrow{1} z_0 \in F \Longrightarrow 1 \in L(M)$
$\iota \xrightarrow{\varepsilon} z_0 \in F \Longrightarrow \varepsilon \in L(M)$

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.

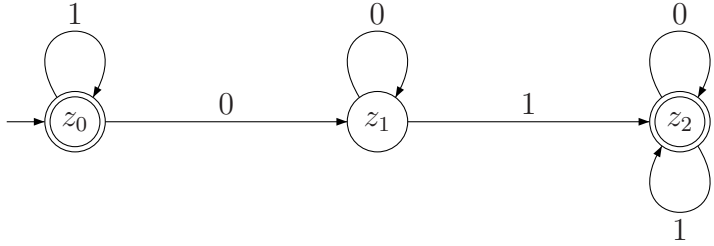
Lemma 2.4. *Sei Σ ein Alphabet und $L \subseteq \Sigma^*$.*
Ist L regulär, so auch $\Sigma^* \setminus L$.

BewIdee:
L regulär $\Longrightarrow$ 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 $w \in \Sigma^*$:

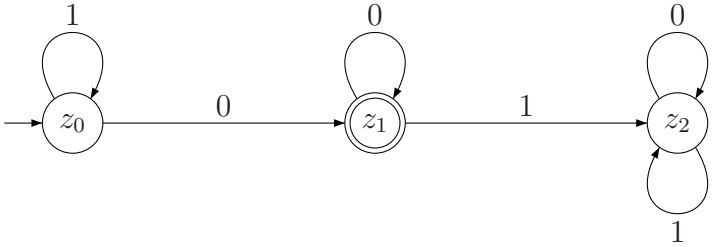
$$\begin{aligned} w \in L(M') &\iff \exists z' \in Z \setminus F \colon \iota \xrightarrow{w} z' \\ &\iff \exists z' \notin F \colon \iota \xrightarrow{w} z' \\ &\iff \text{nicht: } \exists z \in F \colon \iota \xrightarrow{w} z \\ &\iff w \notin L(M) \\ &\iff w \in \Sigma^* \setminus L(M) \end{aligned}$$

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

Beispiel 2.2 (Fortsetzung)



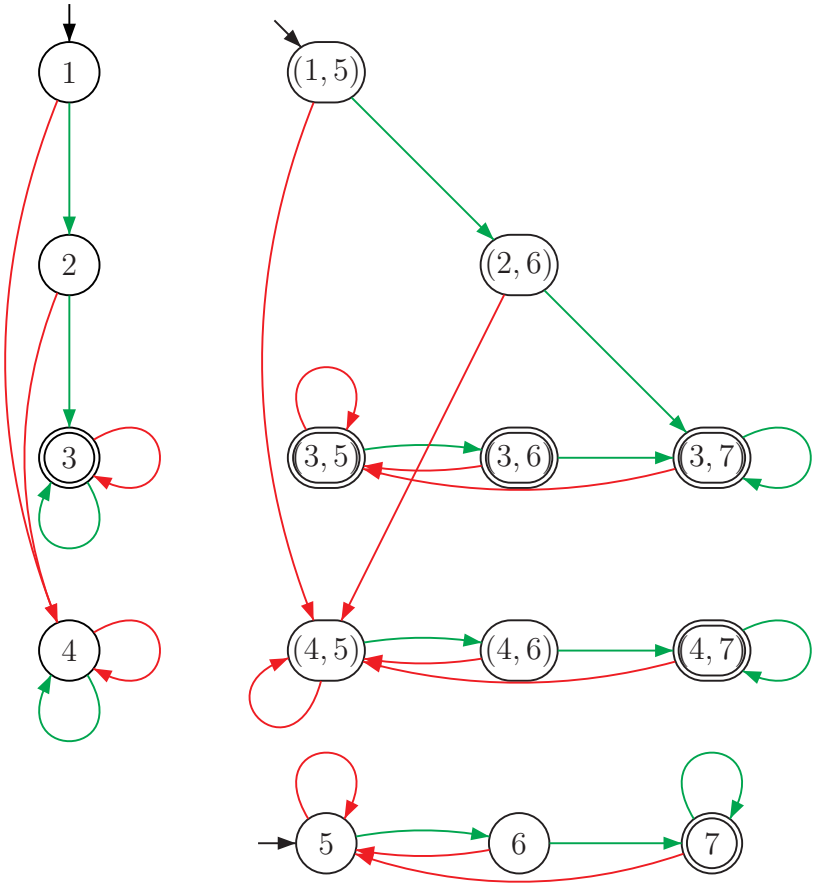
Der DFA M' sieht wie folgt aus:



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

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


1
2
3
4
(1, 5)
(2, 6)
(3, 5)
(3, 6)
(3, 7)
(4, 5)
(4, 6)
(4, 7)
5
6
7

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 $\big(\delta_1(p_1, a), \delta_2(p_2, a)\big) \notin Z'$

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

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

enddo

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

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 $L_1 \cap L_2$. □

2.2 Nichtdeterministische Endliche Automaten (NFA)

Motivation:

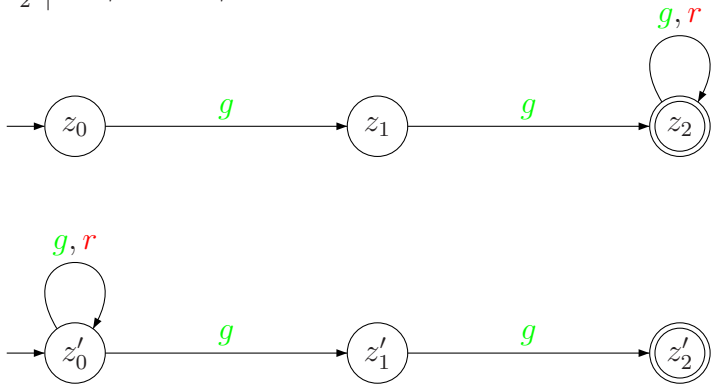
- Aktionen u.U. nicht vollständig beobachtbar bzw. unterscheidbar
- Vereinfachung der Konstruktionen

Definition 2.7. Ein *NFA* ist ein Tupel $M = (Z, \Sigma, I, \delta, F)$ mit folgenden Komponenten:

- Z ist eine endliche Menge (Elemente: „Zustände“)
- Σ ist ein Alphabet
- $I \subseteq Z$ („Initialzustände“)
- $\delta\colon Z \times \Sigma \to \mathcal{P}(Z)$ („Überführungsfunktion“)
- $F \subseteq Z$ („akzeptierende“ oder „Finalzustände“)

Beispiel 2.8. $Z = \{z_0, z_1, z_2, z_0', z_1', z_2'\}$, $\Sigma = \{g, r\}$, $I = \{z_0, z_0'\}$, $F = \{z_2, z_2'\}$

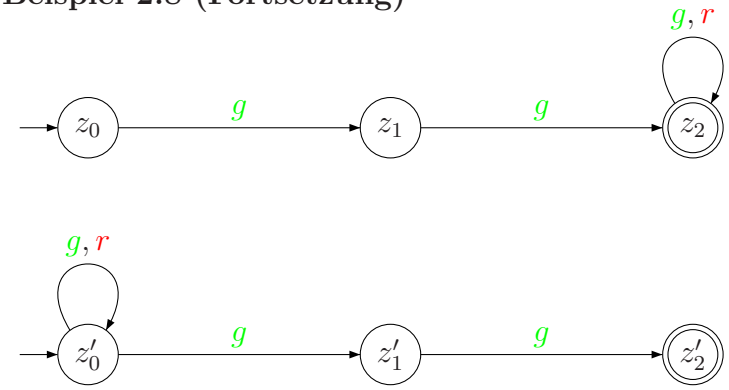
δ	g	r
z_0	$\{z_1\}$	$\emptyset$
z_1	$\{z_2\}$	$\emptyset$
z_2	$\{z_2\}$	$\{z_2\}$
z_0'	$\{z_0', z_1'\}$	$\{z_0'\}$
z_1'	$\{z_2'\}$	$\emptyset$
z_2'	$\emptyset$	$\emptyset$



Definition 2.9. Sei $M = (Z, \Sigma, I, \delta, F)$ ein NFA.

1. Seien $z, z' \in Z$ und $w \in \Sigma^*$. Wir schreiben $z \xrightarrow{w} z'$, wenn es einen in z startenden w-beschrifteten Pfad gibt, der in z' endet
2. Die *von M akzeptierte Sprache* ist $L(M) = \{w \in \Sigma^* \mid \exists \iota \in I, f \in F \colon \iota \xrightarrow{w} f\}$.

Beispiel 2.8 (Fortsetzung)



$z'_0 \xrightarrow{r} z'_0$
$z'_0 \xrightarrow{g} z'_0$ und $z'_0 \xrightarrow{g} z'_1$
$z_0 \xrightarrow{w} z_2$ gdw. w beginnt mit gg
$z'_0 \xrightarrow{w} z'_2$ gdw. w endet auf gg

Daher ist $L(M)$ die Menge der Wörter, die mit gg beginnen oder auf gg enden