02_03_Übungsaufgaben_EndlicheAutomaten__YanboZhu

1

Es ist ein deterministischer endlicher Automat E anzugeben, der genau diejenigen W¨orter ¨uber
dem Alphabet {0, 1, 2} akzeptiert, in denen die Anzahl der Zeichen 0 und 1 jeweils gerade und
die Anzahl des Zeichens 2 ungerade ist. Spezifizieren Sie E als Zustandsgraph.

1.1 Zustandsnamen aufbauen

die Parität (gerade GGG / ungerade UUU) der Anzahl von 2,1,0 in dieser Reihenfolge:
Zustand s_2 s_1 s_0
z.B. UGG = „Anzahl(2) ungerade, Anzahl(1) gerade, Anzahl(0) gerade".

Startzustand: GGG
Akzeptierend EndZustände mit s_2 = U und s_1 = G und s_0= G, also **nur** UGG

1.2 Formale Spezifikation

Alphabet: $\Sigma = \{0, 1, 2\}$

Zustände: Q=$Q = \{sss|s\}$ (insgesamt 2^3 = 8 Zustände)
Konkret: {GGG,GGU,GUG,GUU,UGG,UGU,UUG,UUU}

Startzustand: $q_0 = GGG$

Akzeptierende Zustände: F={UGG}

Übergangsfunktion $\delta : Q \times \Sigma \to Q$

$$\delta(s_2 s_1 s_0, 0) = s_2 s_1 \text{toggle}(s_0)$$
$$\delta(s_2 s_1 s_0, 1) = s_2 \text{toggle}(s_1) s_0$$
$$\delta(s_2 s_1 s_0, 3) = \text{toggle}(s_2) s_1 s_0$$

toggle(G)=U, toggle(U)=G.

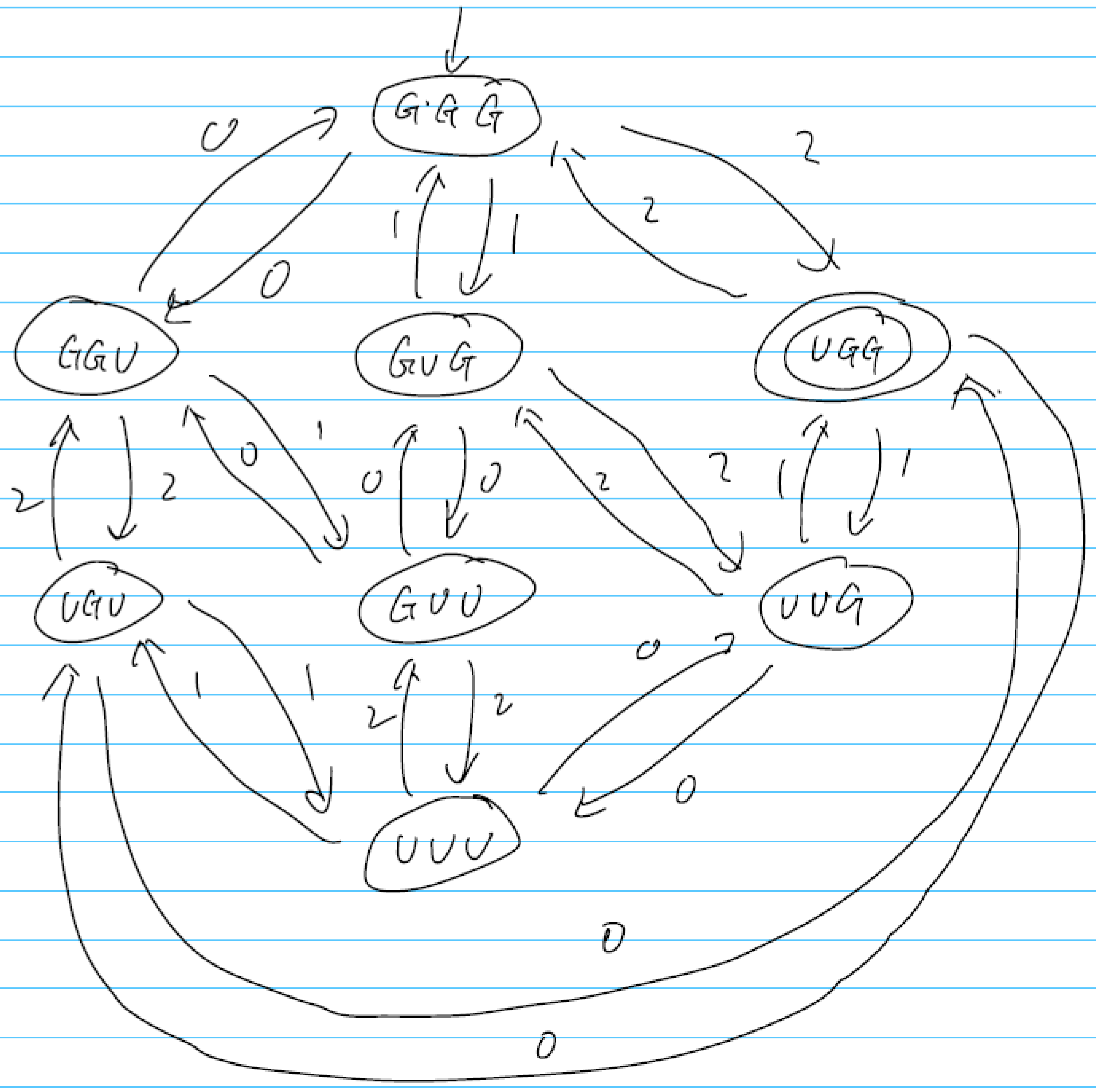
1.3 Übergangstabelle (explizit)

Zustand	δ(0)	δ(1)	δ(2)
GGG	GGU	GUG	UGG
GGU	GGG	GUU	UGU

Zustand	δ(0)	δ(1)	δ(2)
GUG	GUU	GGG	UUG
GUU	GUG	GGU	UUU
UGG	UGU	UUG	GGG
UGU	UGG	UUU	GGU
UUG	UUU	UGG	GUG
UUU	UUG	UGU	GUU

1.4 Zustandsgraph

Man kann die 8 Zustände als Knoten zeichnen; von jedem Knoten gehen drei Kanten aus (für 0,1,2) zu den Zuständen, die die jeweilige Parität umschalten. Start: `GGG` (Pfeil von „Start“). Akzeptierend (doppelt gekennzeichnet): `UGG`.



2

Es ist ein deterministischer endlicher Automat E zu konstruieren, der Wörter über dem Alphabet Σ = {A, B, C, . . . , Z} verarbeitet und dabei genau die erkennt, in denen MEALY oder MOORE Teilwörter sind. Spezifizieren Sie E als Zustandsgraph.

Der Automat erkennt, ob das gelesene Wort eines der beiden Muster `MEALY` oder `MOORE` **als Teilwort (Substring)** enthält.
Sobald eines der beiden Muster vollständig erkannt wurde, geht der Automat in einen **akzeptierenden Endzustand**, in dem er alle weiteren Eingaben akzeptiert.

2.1 Zustand

Wir verwenden folgende Zustände, die jeweils den bereits erkannten Präfix des Musters darstellen:

Zustand	Bedeutung (bisher erkannter Präfix)
`q0`	Startzustand, kein Präfix erkannt
`q1`	`M`
`q2`	`ME`
`q3`	`MEA`
`q4`	`MEAL`
`q6`	`MO`
`q7`	`MOO`
`q8`	`MOOR`
q_{end}	vollständiges Wort `MEALY` oder `MOORE` erkannt (akzeptierender Zustand)

Startzustand: `q0`
Akzeptierender Zustand: q_{end}

2.2 Übergangsregeln (intuitive Beschreibung)

- `q0` :
  - bei `M` → `q1`
  - sonst → `q0`
- `q1` :
  - bei `E` → `q2`
  - bei `O` → `q6`
  - bei `M` → `q1`
  - sonst → `q0`
- `q2` :

    - bei `A` → `q3`
    - bei `M` → `q1`
    - sonst → `q0`
- `q3`:
    - bei `L` → `q4`
    - bei `M` → `q1`
    - sonst → `q0`
- `q4`:
    - bei `Y` → `q_acc` (Wort `MEALY` vollständig erkannt)
    - bei `M` → `q1`
    - sonst → `q0`
- `q6`:
    - bei `O` → `q7`
    - bei `M` → `q1`
    - sonst → `q0`
- `q7`:
    - bei `R` → `q8`
    - bei `M` → `q1`
    - sonst → `q0`
- `q8`:
    - bei `E` → `q_acc` (Wort `MOORE` vollständig erkannt)
    - bei `M` → `q1`
    - sonst → `q0`
- `q_acc`:
    - bei jedem Zeichen → `q_end` (bleibt im akzeptierenden Zustand)

Die Bedeutung von **„sonst → q0"** ist folgende:
Das aktuelle Zeichen kann **keinen gültigen Präfix** eines der beiden Muster fortsetzen. Falls dieses Zeichen **ein „M"** ist, wechselt der Automat (wie bereits explizit angegeben) in den Zustand **q1**; anderenfalls kehrt er in den **Startzustand q0** zurück.

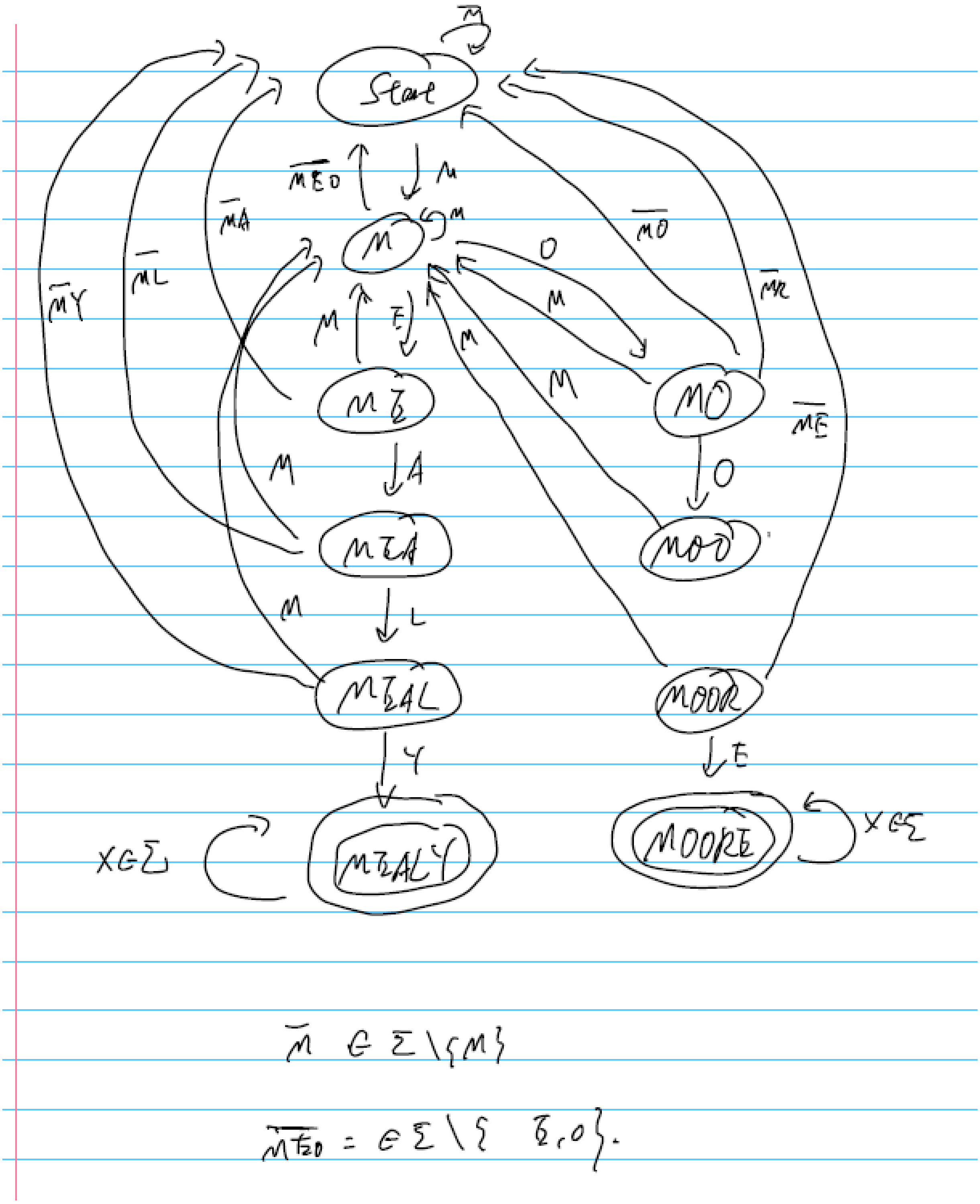
Dies stellt eine **einfache und korrekte DFA-Approximation** dar, die stets das „längste Suffix, das gleichzeitig ein Präfix eines der Muster sein kann" beibehält und sich somit gut für **manuelle Erläuterungen und Implementierungen** eignet.

2.3 Zusammenfassung

Der Automat verfolgt zu jedem Zeitpunkt das längste Präfix der Wörter `MEALY` oder `MOORE`, das dem aktuellen Suffix des Eingabewortes entspricht.
Sobald ein vollständiges Wort erkannt wurde, wird der akzeptierende Zustand `q_acc` erreicht.

Ab diesem Moment bleibt der Automat in `q_end` und akzeptiert jede weitere Eingabe, da das Wort bereits eines der Teilwörter enthält.

2.4 Zustandsgraph



3

L = {w | w ∈ {a, b} ∗ , das vorletzte Zeichen in w ist ein b

3.1 a

(a) Konstruieren Sie einen nichtdeterministischen endlichen erkennenden Automaten N , der die Sprache L erkennt.

Der beschriebene NFA funktioniert folgendermaßen:

1. **„Raten" der Position**:
   Der NFA „rät" an einer beliebigen Position des Eingabeworts, dass das nächste gelesene Zeichen das **vorletzte Zeichen** sein soll.
2. **Beliebiges Durchlaufen**:
   Vor diesem Raten kann der NFA beliebig viele Zeichen lesen, also in einer Schleife (`loop`) über das Wort laufen, ohne in die Prüfphase zu wechseln.
3. **ε-Übergang in die Prüfphase**:
   Sobald er rät, dass das nächste Zeichen das vorletzte ist, wechselt er per **ε-Übergang** (d.h. ohne Eingabe zu konsumieren) in eine Prüfphase.
4. **Prüfung der vorletzten und letzten Zeichen**:
   - In der Prüfphase erwartet der NFA zuerst ein `b` (das vorletzte Zeichen).
   - Danach liest er **genau noch ein beliebiges Zeichen** (das letzte Zeichen).
5. **Akzeptanzbedingung**:
   - Gelingt dies genau in einem Pfad und ist danach das Eingabewort zu Ende, akzeptiert der NFA das Wort.
   - Es reicht, wenn **mindestens ein Pfad** diese Bedingung erfüllt.

3.1.1 Formale Komponenten (kurz):

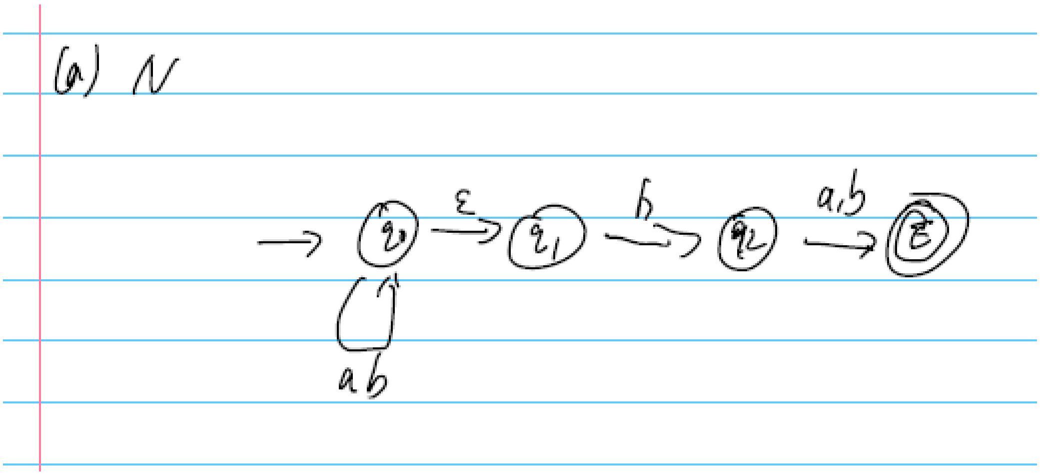
Alphabet: Σ={a,b}
Zustände: Q={q0,q1,q2,qf}

- q0: Startzustand, scan-Phase
- q1: wir haben den vermuteten vorletzten `b` gelesen (wartet auf das letzte Zeichen)
- q2: wir haben das letzte Zeichen gelesen → akzeptierend (aber nur, wenn Eingabe zu Ende)
- qf ist hier nicht extra nötig; wir benutzen q2 als Endzustand

Startzustand: q0
Akzeptierende Zustände: F={q2}

Übergänge (informell / in Worten):

- q_0 -a-> q_0 und q_0 -b-> q_0 (weiter scannen)
- $q_0 \ -\epsilon- > \ q_1$ (an beliebiger Position „raten", diese Stelle könnte vorletzte sein)
- q_1 -b-> q_2 (vorletztes Zeichen muss `b` sein)
- q_2 -a-> ϕ / q_2 -b-> ϕ **keine** Übergänge mehr, d.h. q2 akzeptiert **nur** wenn am Ende der Eingabe (sonst würde weiteres Lesen das Ablehnen erzwingen)



3.2 b

(b) Konstruieren Sie, ohne das Verfahren Teilmengen-Konstruktion anzuwenden, einen deterministischen endlichen erkennenden Automaten E, der die Sprache L erkennt.

Kurze Korrektheitsbegründung (DFA).

- Die Zustände `aa, ab, ba, bb` repräsentieren jeweils die letzten beiden gelesenen Zeichen.
- Wenn das Wort bereits mindestens zwei Zeichen lang ist, trägt der aktuelle Zustand genau die Information, welches Zeichen vorletztes ist (das erste Zeichen des Zustands).
- Daher sind genau die Zustände `ba` und `bb` akzeptierend (vorletztes Zeichen = `b`).
- Wörter mit Länge < 2 landen in `s0` oder `s1_*` und werden nicht akzeptiert (wie gefordert).

3.2.1 Zustandsaufbau

Wir unterscheiden Zustände nach bisher gelesener Länge und nach den letzten ein / zwei Zeichen:

- `s0` : keine Zeichen gelesen (Start)
- `s1_a` : genau 1 Zeichen gelesen, dieses ist `a`
- `s1_b` : genau 1 Zeichen gelesen, dieses ist `b`
- `aa` : zuletzt gelesen waren `a` dann `a` (letzte zwei = `aa`)
- `ab` : zuletzt `a` dann `b`
- `ba` : zuletzt `b` dann `a`
- `bb` : zuletzt `b` dann `b`

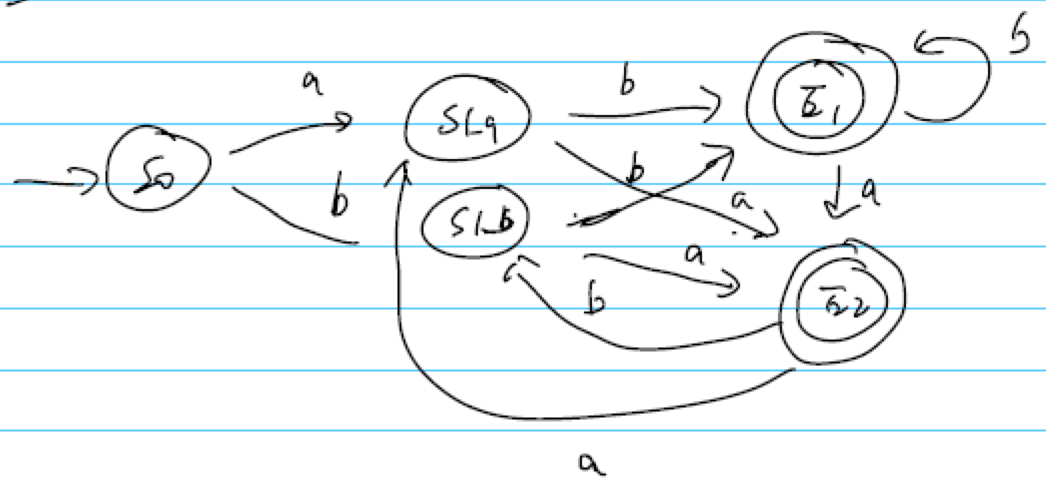
Akzeptierende Zustände: genau diejenigen, bei denen **vorletztes Zeichen** `b` ist, also `ba` und `bb` .

Startzustand: `s0` .

3.2.2 Übergangsregeln

- Von `s0`:
  - `a` → `s1_a`
  - `b` → `s1_b`
- Von `s1_a` (wir haben bislang 1 Zeichen `a`):
  - `a` → `aa`
  - `b` → `ab`
- Von `s1_b`:
  - `a` → `ba`
  - `b` → `bb`
- Von einem Zwei-Buchstaben-Zustand `XY` (X,Y ∈ {a,b}):
  - Bei Eingabe `z` → neuer Zustand ist `Yz` (also verschiebe Fenster um eins)
    - z.B. `aa` mit `b` → `ab`; `ba` mit `b` → `bb`; `bb` mit `a` → `ba` usw.

(b) E



4

Gegeben sei der nichtdeterministische endliche Automat $N = (\{1,2,3\}, \{a,b\}, \delta, 1, \{2\})$ mit

δ:

		a	b	ϵ
$\rightarrow$	1	$\{3\}$	$\emptyset$	$\{2\}$
$*$	2	$\{1\}$	$\emptyset$	$\emptyset$
	3	$\{2\}$	$\{2,3\}$	$\emptyset$

Konstruieren Sie nach dem Verfahren *Teilmengen-Konstruktion* einen zu N äquivalenten deterministischen endlichen Automaten E. Spezifizieren Sie E als Zustandsgraph.

4.1 Ohne Berücksichtigung von Überführungen

4.1.1 Formale Spezifikation

Zustandmenge: $Q' = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$

Eingabealphabet: $\Sigma' := \{a, b\}$

Startzustand: $q_0'' := \{A\}$

Endzustände: $F'' := \{2\}\ \{1,2\}\ \{2,3\}\ \{1,2,3\}$

Überführungsfunktion $\delta'': Q' \times \Sigma' \rightarrow Q$ mit.

$$\delta''(R,x) = \bigcup_{r \in R} \delta(r,x)$$

4.1.2 Übergangstabelle

R	a	b
$\emptyset$	$\emptyset$	$\emptyset$
$\{1\}$	$\{3\}$	$\emptyset$
$\{2\}$	$\{1\}$	$\emptyset$
$\{3\}$	$\{2\}$	$\{2,3\}$
$\{1,2\}$	$\{1,3\}$	$\emptyset$
$\{1,3\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	$\{1,2\}$	$\{2,3\}$
$\{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$

4.2 Unter Berücksichtigung von Überführungen

Die Zustandsmengen E(R) und die Überführungsfunktion $\delta(R, x)$ sind für jedes $R \in Q'$ und $x \in \{a, b\}$ zu construieren

E(R) := {q | q ist von R über keine oder mehrere ϵ -Überführungen erreichbar }

R	$E(R)$
$\emptyset$	$\emptyset$
{1}	{1,2}
{2}	{2}
{3}	{3}
{1,2}	{1,2}
{1,3}	{1,2,3}
{2,3}	{2,3}
{1,2,3}	{1,2,3}

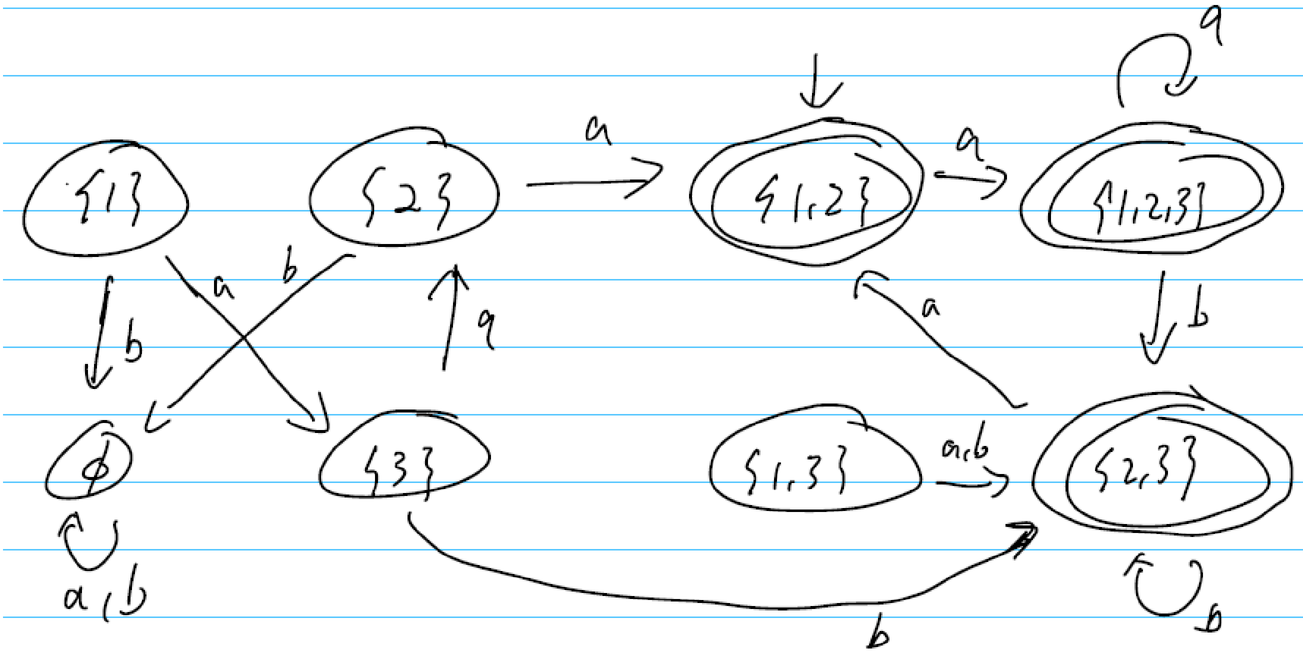
4.3 $\delta'(R, x)$

R	a	$E(\,)$	b	$E(\,)$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
{1}	{3}	{3}	$\emptyset$	$\emptyset$
{2}	{1}	{1,2}	$\emptyset$	$\emptyset$
{3}	{2}	{2}	{2,3}	{2,3}
{1,2}	{1,3}	{1,2,3}	$\emptyset$	$\emptyset$
{1,3}	{2,3}	{2,3}	{2,3}	{2,3}
{2,3}	{1,2}	{1,2}	{2,3}	{2,3}
{1,2,3}	{1,2,3	{1,2,3}	{2,3}	{2,3}

4.4 Zustandsgraph

Startzustand von NEA: {1} => Startzustand von Äquivalenten DEA : {1,2}

Endzustand von NEA: {2} => Startzustand von Äquivalenten DEA : {1,2}, {2,3}, {1,2,3}


{1}
{2}
{1,2}
{1,2,3}
∅
{3}
{1,3}
{2,3}
a
b
a
a
a
b
a,b
a,b
b
b
a