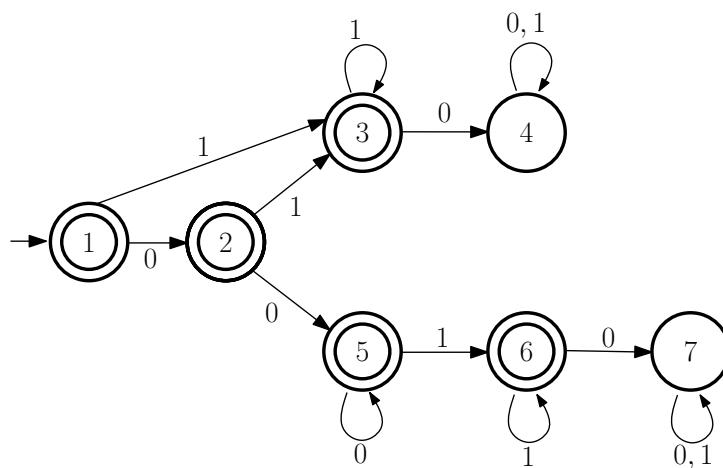


Abgabe bis zum 02. Juni 2020, 10 Uhr, im Whiteboard

Aufgabe 1 Minimalautomat

10 Punkte

Minimieren Sie den folgenden deterministischen endlichen Automaten mit dem Tabellenausfüllalgorithmus.



Welche Sprache wird akzeptiert?

Aufgabe 2 Verständnisfragen

10 Punkte

Welche der folgenden Aussagen treffen zu? Begründen Sie Ihre Antwort jeweils in einem Satz.

- (a) Jede endliche Teilmenge von Σ^* wird von einem endlichen Automaten akzeptiert.
- (b) Ein endlicher Automat kann nur endlich viele Wörter akzeptieren.
- (c) Jede Teilmenge von Σ^* wird von einem endlichen Automaten akzeptiert.
- (d) Zu jedem deterministischen endlichen Automaten A gibt es einen äquivalenten nichtdeterministischen endlichen Automaten, der nicht mehr Zustände als A hat. (Zwei Automaten sind *äquivalent*, wenn sie die gleiche Sprache akzeptieren.)
- (e) Die leere Sprache wird von jedem endlichen Automaten akzeptiert.

Aufgabe 3 Turing-Maschinenprogrammierung

10 Punkte

Geben Sie eine Turing-Maschine an, welche die Sprache

$$L = \{w\#w \mid w \in \{0, 1\}^*\}$$

über dem Eingabealphabet $\Sigma = \{0, 1, \#\}$ erkennt. Beschreiben Sie Ihre Maschine in Worten und formal durch Angabe der Überführungsfunktion.

Hinweis: Führen Sie zusätzliche Zeichen im Bandalphabet ein, um Markierungen auf dem Band darzustellen.

Aufgabe 4 JFlex

freiwillig, 10 Zusatzpunkte

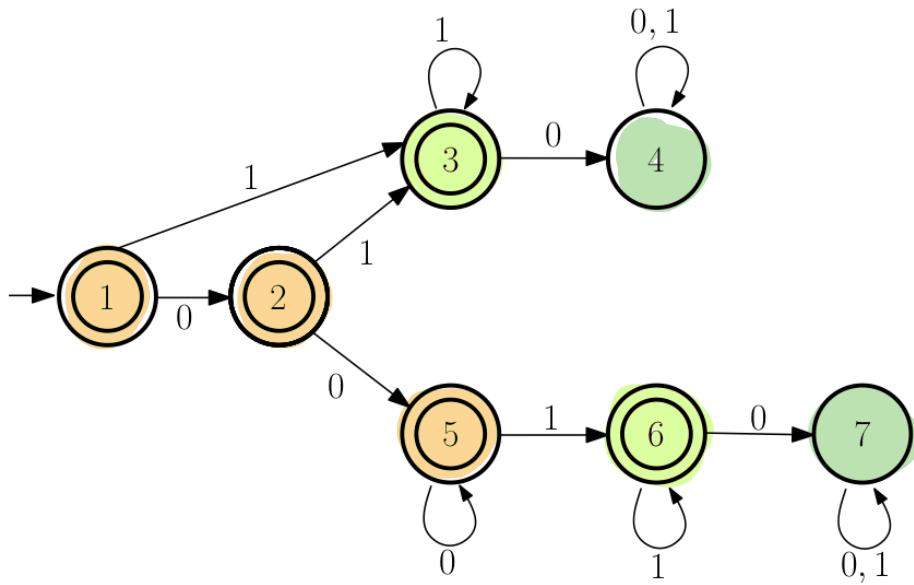
In der Vorlesung wurden Scanner-Generatoren erwähnt, die beim Übersetzerbau zum Einsatz kommen. Ein solcher Scanner-Generator heißt *JFlex*.

- (a) Auf der Veranstaltungsseite befindet sich das Beispielprogramm **summe.flex**. Laden Sie das Programm herunter und finden Sie mit Hilfe der Dokumentation von JFlex heraus, wie man es ausführt. Führen Sie einige Testläufe durch und dokumentieren Sie diese. Gehen Sie dann den Quelltext von **summe.flex** durch und erklären Sie, was die einzelnen Anweisungen bedeuten.
- (b) Bearbeiten Sie eine der beiden folgenden Aufgaben:
 - (i) Ermitteln Sie, wie der Algorithmus funktioniert, der Scanner-Generatoren zu Grunde liegt (z.B. aus dem Buch von I. Wegener). Erläutern Sie diesen Algorithmus kurz in zwei Absätzen. Erklären Sie auch den Zusammenhang zum Vorlesungsstoff.
 - (ii) Modifizieren Sie **summe.flex** so, dass bei Eingabe des Zeichens * die nächste Zahl nicht addiert, sondern multipliziert wird. Bei Eingabe von = soll das bisherige Ergebnis ausgedruckt werden. Außerdem sollen Kommentare im Stil von Haskell unterstützt werden: alles zwischen -- und dem Ende der Zeile wird ignoriert.

Aufgabe 1 Minimalautomat

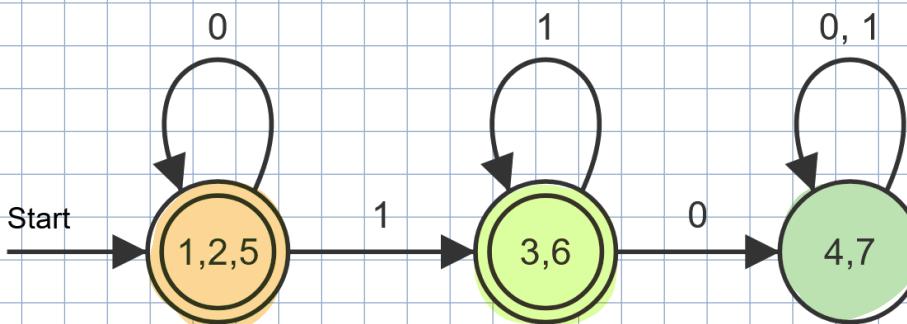
10 Punkte

Minimieren Sie den folgenden deterministischen endlichen Automaten mit dem Tabelleausfüllalgorithmus.



2	33 25				
3	24✓/5 4✓				
4	✓ ✓ ✓				
5	36 25	36 55	45✓ ✓	✓	
6	2✓/5 ✓/1	36 47	✓	57✓	
7	✓ ✓ ✓	47	47	✓ ✓	
	1	2	3	4	5 6

- Die Einträge mit "✓" oder "✗" sind nicht Äquivalent. ✓ steht für Endzustand
- Die farbigen Einträge sind Äquivalent



Aufgabe 2 Verständnisfragen

10 Punkte

Welche der folgenden Aussagen treffen zu? Begründen Sie Ihre Antwort jeweils in einem Satz.

- (a) Jede endliche Teilmenge von Σ^* wird von einem endlichen Automaten akzeptiert.
- (b) Ein endlicher Automat kann nur endlich viele Wörter akzeptieren.
- (c) Jede Teilmenge von Σ^* wird von einem endlichen Automaten akzeptiert.
- (d) Zu jedem deterministischen endlichen Automaten A gibt es einen äquivalenten nichtdeterministischen endlichen Automaten, der nicht mehr Zustände als A hat. (Zwei Automaten sind äquivalent, wenn sie die gleiche Sprache akzeptieren.)
- (e) Die leere Sprache wird von jedem endlichen Automaten akzeptiert.

a) Ja, denn wenn die Teilmenge endlich ist gilt es auch nur endlich viele Wörter, diese Sprache könnte man also in folgender Art darstellen $w_1 \cdot w_2 \dots w_n$ und da es ein regulärer Ausdruck ist, lässt er sich von einem DET akzeptieren

b) Nein, da der $DEA \Rightarrow \textcircled{a}$ z.B. unendlich viele Wörter erzeugen kann

c) Nein, da unter den Teilmengen von Σ^* sich nicht reguläre Sprachen befinden können. Bsp. $\Sigma = \{0, 1\}$ $\{0^n 1^n \mid n \geq 0\} \subseteq \Sigma^*$

d) Richtig, da 1. aus der Vorlesung wissen wir, dass NFA und DFA äquivalent sind

und 2. Der NFA hat maximal soviele Zustände wie der DFA, da die DFAs für jeden Zustand soviele Übergänge wie Elemente in Σ brauchen, bei NFA's ist dies nicht der Fall

e) Stimmt, da wir die leere Sprache als leere Menge definiert haben und jede Menge hat die leere Menge als Teilmenge

Aufgabe 3 Turing-Maschinenprogrammierung

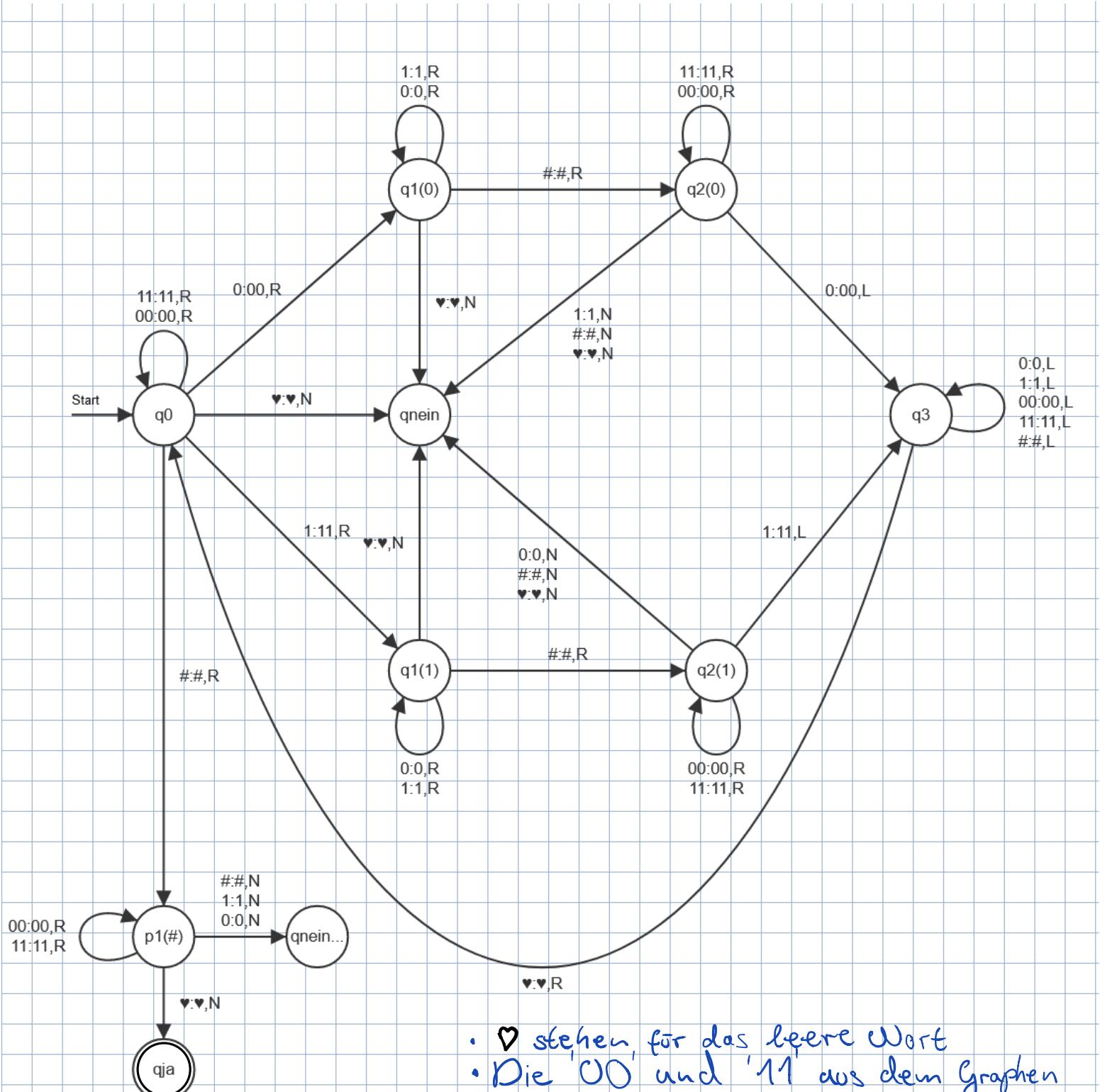
10 Punkte

Geben Sie eine Turing-Maschine an, welche die Sprache

$$L = \{w\#w \mid w \in \{0, 1\}^*\}$$

über dem Eingabealphabet $\Sigma = \{0, 1, \#\}$ erkennt. Beschreiben Sie Ihre Maschine in Worten und formal durch Angabe der Überführungsfunktion.

Hinweis: Führen Sie zusätzliche Zeichen im Bandalphabet ein, um Markierungen auf dem Band darzustellen.



- \heartsuit stehen für das leere Wort
- Die '00' und '11' aus dem Graphen ist mit der 0 und 1 aus der Tabelle gleich zu stellen und dient lediglich als Markierung.
- '00' und '11' wird von der Turing Maschine als 1 char gelesen

$\Sigma = \{0, 1, \#, \text{q}, \text{qnein}\}$ $\Gamma = \{0, 1, \#, \text{q}, \text{qnein}\}$ $Q = \{q_0, q_1(0), q_1(1), q_2(0), q_2(1), q_3, q_{ja}, q_{nein}\}$

δ	0	1	#	ω	0	1
q_0	$(q_1(0), 1, R)$	$(q_1(1), 1, R)$	$(q_1(\#), \#, N)$	(q_{nein}, ω, N)	$(q_0, 0, R)$	$(q_0, 1, R)$
$q_1(0)$	$(q_1(0), 0, R)$	$(q_1(0), 1, R)$	$(q_2(0), \#, R)$	(q_{nein}, ω, N)	—	—
$q_1(1)$	$(q_1(1), 0, R)$	$(q_1(1), 1, R)$	$(q_2(1), \#, R)$	(q_{nein}, ω, N)	—	—
$q_2(0)$	$(q_3, 0, L)$	$(q_{nein}, 1, N)$	$(q_{nein}, \#, N)$	(q_{nein}, ω, N)	$(q_2(0), 0, R)$	$(q_2(0), 1, R)$
$q_2(1)$	$(q_{nein}, 0, N)$	$(q_3, 1, L)$	$(q_{nein}, \#, N)$	(q_{nein}, ω, N)	$(q_2(1), 0, R)$	$(q_2(1), 1, R)$
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	$(q_3, \#, L)$	(q_0, ω, R)	$(q_3, 0, R)$	$(q_3, 1, R)$
$q_{ja}(\#)$	$(q_{nein}, 0, N)$	$(q_{nein}, 1, N)$	$(q_{nein}, \#, N)$	(q_{ja}, ω, N)	$(q_{ja}(\#), 0, R)$	$(q_{ja}(\#), 1, R)$

Idée: Der erste Buchstabe des linken Wortes, welches noch nicht markiert wurde, wird von q_0 markiert und gemerkt.
Das gemerkte Zeichen wird an q_1 übergeben.

Dann geht q_1 solange nach rechts bis das Zeichen # gefunden wurde. Das gemerkte Zeichen wird an q_2 übergeben.

Darauf übernimmt q_3 und sucht das erste unmarkierte Zeichen, falls es mit dem Speicher übereinstimmt, wird das neue Zeichen auch markiert.

Danach geht q_3 solange nach rechts bis es ω findet. Mit dem ω wird q_0 wieder aufgerufen und der Ablauf beginnt erneut.

Die Abbruchbedingung ist, falls q_0 solange nach rechts laufen könnte bis das # gefunden wurde. Denn darauf übernimmt $q_{ja}(\#)$. Wenn $q_{ja}(\#)$ weiter nach rechts laufen kann, d.h. ohne das unmarkierte oder # Zeichen kommen, bis ein ω erscheint, ist es ein legales Wort.

Aufgabe 4 JFlex

freiwillig, 10 Zusatzpunkte

In der Vorlesung wurden Scanner-Generatoren erwähnt, die beim Übersetzerbau zum Einsatz kommen. Ein solcher Scanner-Generator heißt *JFlex*.

- (a) Auf der Veranstaltungsseite befindet sich das Beispielprogramm `summe.flex`. Laden Sie das Programm herunter und finden Sie mit Hilfe der Dokumentation von JFlex heraus, wie man es ausführt. Führen Sie einige Testläufe durch und dokumentieren Sie diese. Gehen Sie dann den Quelltext von `summe.flex` durch und erklären Sie, was die einzelnen Anweisungen bedeuten.
- (b) Bearbeiten Sie eine der beiden folgenden Aufgaben:
 - (i) Ermitteln Sie, wie der Algorithmus funktioniert, der Scanner-Generatoren zu Grunde liegt (z.B. aus dem Buch von I. Wegener). Erläutern Sie diesen Algorithmus kurz in zwei Absätzen. Erklären Sie auch den Zusammenhang zum Vorlesungsstoff.
 - (ii) Modifizieren Sie `summe.flex` so, dass bei Eingabe des Zeichens * die nächste Zahl nicht addiert, sondern multipliziert wird. Bei Eingabe von = soll das bisherige Ergebnis ausgedruckt werden. Außerdem sollen Kommentare im Stil von Haskell unterstützt werden: alles zwischen -- und dem Ende der Zeile wird ignoriert.

a) Testläufe: Bei Eingaben von Zahlen im "Double Style" (Bsp.: - 0,7 ; 1; + 55,00) wird die Zahl auf der Konsole mit dem Text "Zahl" ausgegeben.

- Bei einer Eingabe anderer Wörter wird ein Fehler geworfen und das erste illegale Zeichen gezeigt
 - Wenn man das Programm beendet, wird die Summe ausgegeben
- Weitere Erklärung im Quelltext

```
thore@ubuntu:~/Desktop$ java Summe
-2.4
Zahl -2.4
+4
Zahl +4
2.4
Zahl 2.4
Summe = 4.0
thore@ubuntu:~/Desktop$ java Summe
-
Exception in thread "main" java.lang.RuntimeException: Unerlaubtes Zeichen -".
        at Summe.yylex(Summe.java:567)
        at Summe.main(Summe.java:245)
thore@ubuntu:~/Desktop$
```

b)

ii) im Quelltext