1. Aufgabenblatt zur Vorlesung

Grundlagen der Theoretischen Informatik, SoSe 2008

(Dr. Frank Hoffmann)

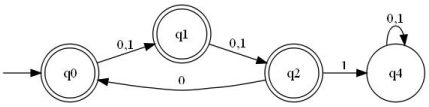
Lösung von Manuel Jain und Benjamin Bortfeldt

Aufgabe 2 Zustandsdiagramme (6 Punkte, wird korrigiert)

Geben Sie dfa's durch ihre Zustandsdiagramme an, die die folgenden Sprachen L über dem Alphabet $\Sigma = \{0, 1\}$ entscheiden.

(a) L enthält alle Wörter, für die jeder dritte Buchstabe eine 0 ist.

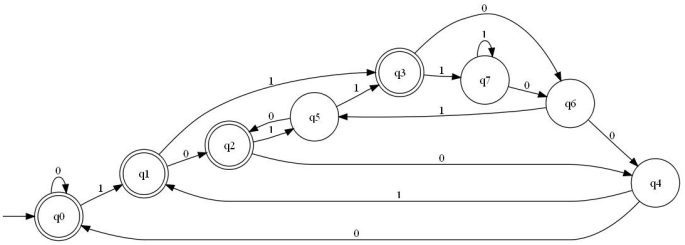
Lösung: So kann der Automat aussehen, der die gegebene Sprache akzeptiert:



Nur jedes dritte Zeichen ist von Bedeutung, falls dieses eine 1 ist, kann die Forderung nicht mehr erfüllt werden und der Automat landet in einem „toten" Zustand q4. Falls es eine 0 ist, geht der Automat in eine Schleife und wieder an den Anfang zurück. $q_0, q_1, q_2 \in F$ damit der Automat Wörter der Länge < 3 akzeptiert und auch Wörter, deren Länge kein Vielfaches von 3 ist (z.B. 11011).

(b) L enthält alle Wörter, deren drittletzter Buchstabe eine 0 ist. Der Automat sollte nicht mehr als 8 Zustände haben.

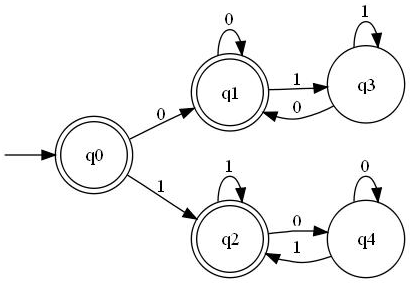
Lösung: So könnte der Automat aussehen:



Es gibt genau vier akzeptierende Zustände. Jeder Zustand kodiert die letzten drei gelesenen Zeichen. Diese drei gelesenen Zeichen als Binärzahl ergeben den Zustand, z.b. $010_2 = 2_{10}$, also ist in Zustand q_2 kodiert, dass die letzte gelesene Dreiersequenz „010" war. Falls bisher erst ein oder zwei Zeichen gelesen wurden, werden die fehlenden Zeichen einfach als 0 angenommen. So erreicht man, dass der Automat auch für Wörter der Länge < 3 akzeptiert und man trotzdem nicht mehr als acht Zustände benötigt. Falls nur Wörter der Länge ≥ 3 akzeptiert werden sollen, setze als Startzustrand q_7. So wird erreicht, dass bei Wörtern von geringerer Länge die fehlenden Zeichen als 1 angenommen werden und der Automat somit nicht akzeptiert.

(c) $L = \{w \in \Sigma^* | \text{w enthält gleichviele Teilwörter 01 und 10}\}$

Lösung: So könnte dieser Automat aussehen:



Es kann immer nur höchstens ein 01-Übergang mehr als ein 10-Übergang auftreten oder umgekehrt. Des weiteren kann man schnell feststellen, dass das Wort mit dem Zeichen aufhören muss, mit dem es angefangen hat. Daher kommt die anfängliche Fallentscheidung und dann in den jeweiligen Teilen die Schleife.

Aufgabe 3 Reguläre Ausdrücke (4 Punkte, wird korrigiert)

Geben Sie reguläre Ausdrücke für die folgenden Sprachen L über dem Alphabet $\Sigma = \{0, 1\}$ an. Hinweis: Beachten Sie, für Wörter mit weniger als drei Buchstaben sind beide Forderungen trivial erfüllt.

(a) L enthält alle Wörter, für die jeder dritte Buchstabe eine 0 ist.
Lösung:

$$\underbrace{(0 \cup 1)}_{|w|=1} \cup \underbrace{((0 \cup 1) \circ (0 \cup 1))}_{|w|=2} \cup \underbrace{(((0 \cup 1) \circ (0 \cup 1) \circ 0)^* \circ (0 \cup 1 \cup \epsilon) \circ (0 \cup 1 \cup \epsilon))}_{|w|\geq 3, |w|=0}$$

(b) L enthält alle Wörter, deren drittletzter Buchstabe eine 0 ist.
Lösung:

$$\epsilon \cup \underbrace{(0 \cup 1)}_{|w|=1} \cup \underbrace{((0 \cup 1) \circ (0 \cup 1))}_{|w|=2} \cup \underbrace{((0 \cup 1)^* \circ 0 \circ (0 \cup 1)^2)}_{|w|\geq 3}$$

Hier wurden noch die Fälle berücksichtigt, in denen die Wörtlänge < 3 ist. Falls man diese Fälle nicht berücksichtigen möchte, nimmt man einfach nur den regulären Ausdruck, der sich über der Klammer $|w| \geq 3$ befindet.

Aufgabe 4 RoboMaus$^{\text{TM}}$

Wir wollen mit Hilfe eines endlichen Automaten einen Roboter simulieren, der endliche Irrgärten durchläuft. Solche Automaten werden "Mäuse" genannt, in Anlehnung an eine der ersten kybernetischen Maschinen, die *Maus* von Claude Shannon.
Wir betrachten die Maschen des ganzzahligen Gitters $\mathbb{Z}^2$. Zwei Maschen heißen benachbart, wenn sie eine gemeinsame Kante haben. Eine Menge von Maschen heißt zusammenhängend, wenn sie bezüglich dieses Nachbarschaftsbegriffes zusammenhängend ist.
Wir färben zunächst alle Maschen des Gitters schwarz/weiß. Die Menge L der weißen Maschen einer Färbung heißt Irrgarten, wenn L mindestens zwei Zellen enthält und endlich ist und sowohl L wie auch die Menge der schwarzen Maschen zusammenhängend ist.

Entwerfen Sie eine durch einen dfa gesteuerte Maus M, die folgendes leistet: M soll beim Start auf einer beliebigen Masche eines beliebigen Irrgartens L mindestens alle Randmaschen von L besuchen, d.h. alle weißen Maschen, die eine benachbarte schwarze Masche haben.

Formalisierung M befindet sich zum Zeitpunkt t in Masche c_t. Er erkennt die freien Richtungen $A \subseteq D = \{n, e, s, w\}$, in denen sich Zellen des Irrgartens befinden (M kann nur benachbarte Zellen sehen). M berechnet ein $r \in A$ und geht einen Schritt in Richtung r zur benachbarten Zelle c_{t+1} u.s.w.

Lösungsvorschlag M findet zuerst irgendeine Randzelle und geht dann "immer an der Wand entlang". Versuchen Sie nicht, eine Maus zu konstruieren, die nach getaner Arbeit anhält. Eine solche gibt es nicht, was aber recht schwierig zu beweisen ist.

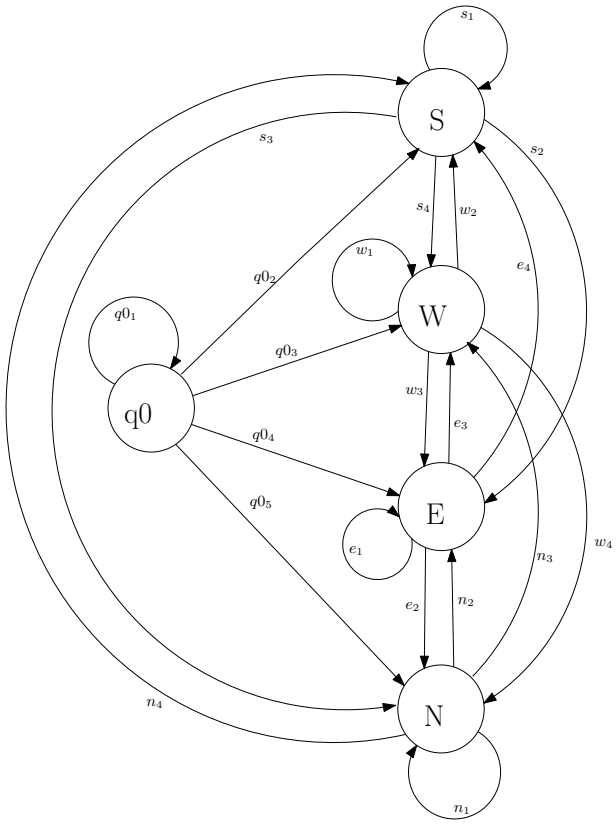
Lösung :

Dieser Lösungsvorschlag kann durch einen dfa folgendermassen umgesetzt werden:

Als erstes brauchen wir einen Startzustand $q0$. Dieser wird dafür zuständig sein eine erste Zelle am Rand zu finden. Dies geht recht einfach indem man, solange noch alle Richtungen frei sind, immer in eine bestimmte Richtung geht. In unserem Fall ist dies norden. Sobald eine Richtung nicht mehr frei ist haben wir eine Randzelle gefunden.

Um alle Randzellen abzulaufen können wir uns zuerst folgende Beobachtung machen: Wenn wir, wann immer es möglich ist, nach links laufen, erreichen wir jede Zelle des Randes. Dies lässt sich leicht realisieren: Wenn wir aus einer Richtung kommen, wird in der nächsten Zelle ersteinmal geschaut ob wir nach links können. Wenn dies nicht möglich ist testen wir ob es geradeaus weitergeht. Wenn auch dies nicht geht sind wir an einer Ecke angekommmen und müssen testen ob wir nach rechts laufen können. Ist all das nicht möglich sind wir in einer Sackgasse und müssen zurücklaufen.

In unserem dfa setzten wir das ganze genau so um. Wir haben neben $q0$ vier weitere Zustände. Diese symboliesieren die Richtung in die wir gerade gegangen sind. In jedem Zustand ist demnach klar was links, geradeaus und rechts bedeuten und wir können dementsprechend handeln. Das ganze sieht dann folgendermassen aus:



Da es zu unübersichtlich wird die komplette Kantenbeschriftung in den Automaten zu schreiben, haben wir die Kanten benannt und geben die Überführungsfunktion für jede Kante an. Hierbei entspricht jede Menge einem $A \subseteq D = \{n, e, s, w\}$ und der Buchstabe nach dem / gibt an in welche Richtung sich die Maus bewegt.

$q0_1 : \{n, e, s, w\} \to n$
$q0_2 : \{n, s, w\}, \{s, w\}, \{s\}, \{n, s\} \to s$
$q0_3 : \{n, e, w\}, \{n, w\}, \{w\} \to w$
$q0_4 : \{e, s, w\}, \{e, s\}, \{e\}, \{e, w\} \to e$
$q0_5 : \{n, e, s\}, \{n, e\}, \{n\} \to n$
$s_1 : \{n, s, w\}, \{n, s\} \to s$
$s_2 : \{n, e, s, w\}, \{n, e, s\}, \{n, e\}, \{n, e, w\} \to e$
$s_3 : \{n\} \to n$
$s_4 : \{n, w\} \to w$
$e_1 : \{e, s, w\}, \{e, w\} \to e$
$e_2 : \{n, e, s, w\}, \{n, e, w\}, \{n, w, s\}, \{n, w\} \to n$
$e_3 : \{w\} \to w$
$e_4 : \{s, w\} \to s$
$w_1 : \{n, e, w\}, \{e, w\} \to w$
$w_2 : \{n, e, s, w\}, \{n, e, s\}, \{e, s, w\}, \{e, s\} \to s$
$w_3 : \{e\} \to e$
$w_4 : \{n, e\} \to n$
$n_1 : \{n, e, s\}, \{n, s\} \to n$
$n_2 : \{e, s\} \to e$
$n_3 : \{n, e, w, s\}, \{n, w, s\}, \{e, w, s\}, \{w, s\} \to w$
$n_4 : \{s\} \to s$

Aufgabe 5 RoboMaus^{TM} v2.0

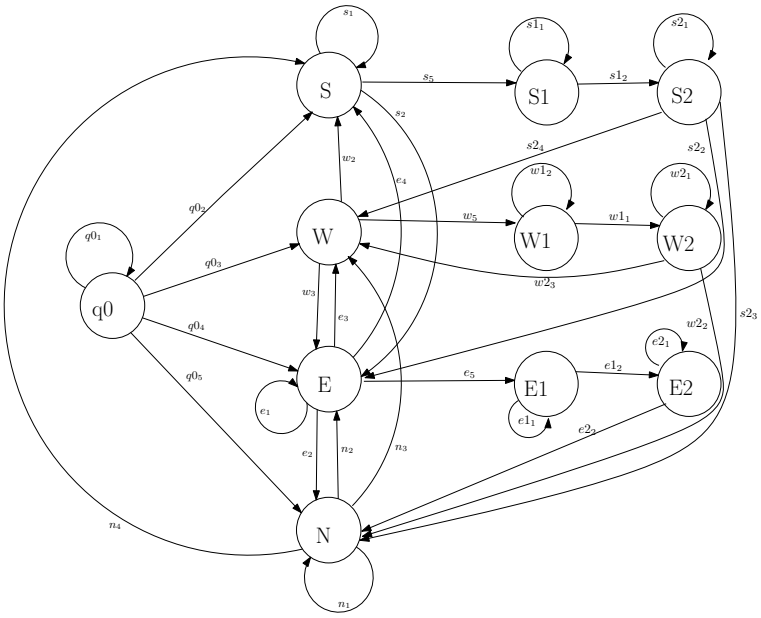
Modifizieren Sie Ihre Lösung aus Aufgabe 4 so, dass die Maus alle freien Maschen aus L besucht.

Lösung: Um zu erreichen, dass die Maus alle freien Maschen besucht, benutzen wir folgende Beobachtung:
Sei c eine beliebige freie Masche. Dann gibt es eine Masche $b = b(c)$ mit den Eigenschaften:

- (i) in Masche b ist Richtung s nicht frei.
- (ii) Masche c kann von b aus durch eine endliche Folge von Schritten in Richtung n erreicht werden.

Aus dieser Beobachtung ergibt sich folgende algorithmische Umsetzung:
Die Maus läuft den Rand des Irrgartens wie in Aufgabe 4 ab. Sie unterbricht dies in jeder Zelle c, in der Richtung s nicht frei ist, läuft so weit wie möglich nach Norden und wieder zurück nach c und geht dann erst zur nächsten Randzelle.

Um eine dfa-Steurung der Maus mit diesem Algorithmus zu bauen, muss nur der Automat aus Aufgabe 4 angepasst werden. Der Grundaufbau, in unserem Fall das Ablaufen des Randes, bleibt gleich. Allerdings springen wir sobald Richtung s nicht frei ist in eine Art Subroutine, die nur dafür sorgt, dass wir solange nach norden laufen bis eine Wand kommt und dann wieder zurück nach süden. Diese Subroutine lässt sich mit zwei Zuständen realisieren. Der erste der dazu da ist nach norden zu laufen und der zweite der für das Zurücklaufen zusändig ist. Am Ende der Subroutine gehen wir zurück in den schon bekannten Teil des Automats, der das Randablaufen realisiert. In welche Richtung wir laufen, wenn wir aus der Subroutine zurückkommen, hängt davon ab aus welcher Richtung wir vor dem Eintritt in sie kamen. Deshalb gibt es mehrere Zustandspaare die unsere Subroutine realisieren. Dabei ist $w1$ der für das Laufen nach norden und $w2$ der für das Laufen nach süden verantwortliche Knoten, wenn wir im letzten Schritt nach westen gelaufen sind. $n1$, $n2$, $s1$ und $s2$ sind die enstsprechenden Knoten für die anderen Richtungen.
Als fertiger Automat sieht es dann so aus:



Auch hier werden die Zustansübergänge aufgeschrieben wie in Aufgabe 4:
$q0_1 : \{n, e, s, w\} \rightarrow n$
$q0_2 : \{n, s, w\}, \{s, w\}, \{s\}, \{n, s\} \rightarrow s$
$q0_3 : \{n, e, w\}, \{n, w\}, \{w\} \rightarrow w$
$q0_4 : \{e, s, w\}, \{e, s\}, \{e\}, \{e, w\} \rightarrow e$
$q0_5 : \{n, e, s\}, \{n, e\}, \{n\} \rightarrow n$
$s_1 : \{n, s, w\}, \{n, s\} \rightarrow s$
$s_2 : \{n, e, s, w\}, \{n, e, s\}, \rightarrow e$
$s_5 : \{n, e\}, \{n, e, w\}, \{n, w\}, \{n\} \rightarrow n$
$e_1 : \{e, s, w\}, \{e, w\} \rightarrow e$
$e_2 : \{n, e, s, w\}, \{n, w, s\} \rightarrow n$
$e_3 : \{w\} \rightarrow w$
$e_4 : \{s, w\} \rightarrow s$
$e_5 : \{n, e, w\}, \{n, w\} \rightarrow n$
$w_2 : \{n, e, s, w\}, \{n, e, s\}, \{e, s, w\}, \{e, s\} \rightarrow s$
$w_3 : \{e\} \rightarrow e$
$w_5 : \{n, e, w\}, \{e, w\}, \{n, e\} \rightarrow n$
$n_1 : \{n, e, s\}, \{n, s\} \rightarrow n$
$n_2 : \{e, s\} \rightarrow e$
$n_3 : \{n, e, w, s\}, \{n, w, s\}, \{e, w, s\}, \{w, s\} \rightarrow w$
$n_4 : \{s\} \rightarrow s$
$s1_1 : \{n, e, s, w\}, \{n, e, s\}, \{n, w, s\}, \{n, s\} \rightarrow n$
$s1_2 : \{e, s, w\}, \{e, s\}, \{w, s\}, \{s\} \rightarrow s$
$s2_1 : \{n, e, s, w\}, \{n, e, s\}, \{n, w, s\}, \{n, s\} \rightarrow s$
$s2_2 : \{n, e\}, \{n, e, w\} \rightarrow e$
$s2_3 : \{n\} \rightarrow n$
$s2_4 : \{n, w\} \rightarrow w$
$w1_1 : \{n, e, s, w\}, \{n, e, s\}, \{n, w, s\}, \{n, s\} \rightarrow n$
$w2_2 : \{e, s, w\}, \{e, s\}, \{w, s\}, \{s\} \rightarrow s$
$w2_1 : \{n, e, s, w\}, \{n, e, s\}, \{n, w, s\}, \{n, s\} \rightarrow s$
$w2_2 : \{n, e\} \rightarrow n$
$w2_3 : \{n, e, w\}, \{e, w\} \rightarrow w$

$e1_1 : \{n, e, s, w\}, \{n, e, s\}, \{n, w, s\}, \{n, s\} \rightarrow n$
$e1_2 : \{e, s, w\}, \{e, s\}, \{w, s\}, \{s\} \rightarrow s$
$e2_1 : \{n, e, s, w\}, \{n, e, s\}, \{n, w, s\}, \{n, s\} \rightarrow s$
$e2_2 : \{n, e, w\}, \{n, w\}, \rightarrow n$