Universität des Saarlandes
FR 6.2 Informatik

Prof. Dr. Kurt Mehlhorn, Dr. Konstantinos Panagiotou **WiSe 2011/2012**

Übungen zu Computational Thinking

http://www.mpi-inf.mpg.de/departments/d1/teaching/ws11/ct/

Bonus-Blatt 3

Regeln: Bis zum Semesterende müssen mindestens 42% der maximal erreichbaren Punkte aller Übungszettel erworben werden.
Programmcode ist elektronisch per E-Mail abzugeben. Zusätzlich müssen die Ausgaben einer exemplarischen Programmausführung mitgeliefert werden.

Punkte können Sie erhalten, wenn Sie die Aufgaben bis zum 9.1.12 bearbeiten. Wenn Sie Fragen haben, die Erläuterungen Ihnen nicht ausführlich genug sind, oder Sie Tipps für die Lösung brauchen, wenden Sie sich an Bernhard, er hilft Ihnen gern.

Im Laufe dieses Blattes lernen sie *Endliche Automaten* kennen, ein wichtigen Konzept aus der Theoretischen Informatik, das Anwendungen findet in den unterschiedlichsten Gebieten, vom Erkennen eines Wortes innerhalb eines Textes bis hin zur Erstellung der Kontrolle eines Flugzeuges. Selbst innerhalb ihres Python Interpreters ist ein Endlicher Automat zu finden.
Informell gesprochen ist ein Endlicher Automat ein Mechanismus der ein Eingabe Buchstabe für Buchstabe liest und erkennt, ob das entstehende Wort zu einer *Sprache*, also einer Menge von Wörtern, wie zum Beispiel alle Palindrome, gehört. Anders als die Turingmaschinen, die Sie bereits kennen, hat ein Endlicher Automat kein Speicherband für seine Berechnungen zur Verfügung, er muss mit einer konstanten Menge Speicher, seinen Zuständen, auskommen. Für Informatiker ist ein Endlicher Automat M durch fünf Werte bestimmt:

$$M = (Z, \Sigma, \Delta, S, E).$$

Hierbei ist Z die Menge aller Zustände des Automaten und Σ das Alphabet aus dem die Buchstaben der Sprache stammen. S ist der Startzustand, in dem der Automat sich initial befindet, $E \subseteq Z$ sind die akzeptierenden Zustände und $\Delta \subseteq Z \times \Sigma \times Z$ ist die sogenannte Übergangsrelation oder auch Transitionsrelation genannt, die angibt wie sich der derzeitige Zustand ändert, wenn man ein weiteres Zeichen des Alphabets liest. Ein Automat akzeptiert ein Wort, wenn er sich nach dem Lesen in einem der akzeptierenden Zustände befindet.
Eine anschaulichere Darstellung für die Übergangsrelation ist der sogenannte Zustandsgraph. In dem Zustandsgraphen ist jeder Zustand ein Knoten und wir fügen eine (gerichtete) Kante zwischen zwei Zuständen u und v mit dem Buchstaben a als Beschriftung ein, genau dann wenn $(u, a, v) \in \Delta$ ist, also wenn der Automat von u nach v wechselt, wenn er a liest.

Zum Beispiel ist der Zustandsgraph zu folgendem Automaten:

$$
\begin{aligned}
Z =& \{S, A, B\} \\
S =& S \\
\Sigma =& \{a, b, c\} \\
\Delta =& \{(S, a, A), (S, b, Err), (S, c, Err) \\
& (A, a, Err), (A, b, S), (A, c, Err) \\
& (Err, a, Err), (Err, b, Err), (Err, c, Err)\} \\
E =& \{A\}
\end{aligned}
$$

gegeben durch:

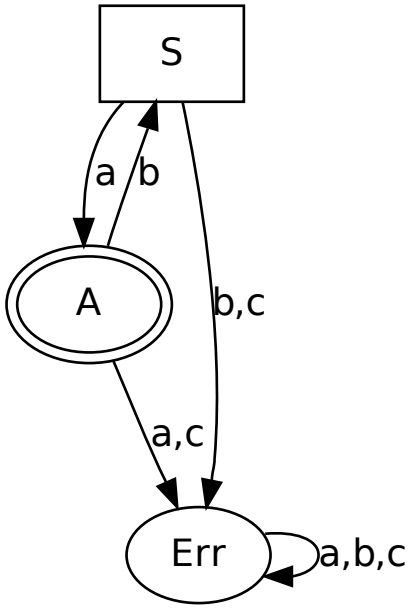

Abbildung 1: Der Übergangsgraph

Der Startzustand wird hierbei durch die eckige Box makiert und akzeptierenden Zustände sind durch umkreiste Knoten markiert.
Eine Folge von Zuständen $z_0, z_1, z_2, \ldots, z_n$ eines Automatens auf einem Wort $a_0, a_1, a_2, \ldots, a_{n-1}$ nennt man *Berechnung* wenn für zwei aufeinander folgende Zustände z_i, z_{i+1} gilt $(z_i, a_i, z_{i+1}) \in \Delta$.
Ein Automat akzeptiert nun ein Wort w, wenn es eine Berechnung des Automaten auf w gibt, die folgenden Eigenschaften erfüllt:

- z_0 ist der Startzustand
- z_n ist ein akzeptierender Zustand, also $z_n \in E$

Falls es keine solche Berechnung gibt, akzeptiert der Automat das Wort nicht. Insbesondere akzeptiert der Automat nicht, wenn er „steckenbleibt“, also es vom derzeitigen Zustand keine Kante gibt, die mit dem derzeitigen Zeichen beschriftet ist. Der Automat aus Abbildung 1 akzeptiert zum Beispiel das Wort *ababa*, nicht jedoch *abbaa* oder *ccc*.
Die Sprache, die der Automat akzeptiert, ist dann die Menge aller Wörter, die von dem Automaten akzeptiert werden. Natürlich gibt es sehr viele verschiedene Automaten, die die selbe Sprache akzeptieren.

Aufgabe 1 (*10 Punkte*)

a) ✍ *5 Punkte.* Zeichnen Sie einen Übergangsgraphen zu einem Automaten der Zahlen in Dezimaldarstellung (mit Komma) oder binäre Zahlen akzeptiert.

b) ✍ *5 Punkte.* Geben Sie eine Berechnung Ihres Automaten auf dem Eingabewort

$$12345,6789$$

an.

Aufgabe 2 (*10 Punkte*) In dieser Aufgabe sollen Sie eine Klasse implementieren, die einen Endlichen Automaten als Übergangsgraph repräsentiert.

a) *3 Punkte.* Schreiben Sie eine Initialisierungsfunktion für ihre Klasse, die als Argument ein Eingabealphabet, eine Menge von Zuständen, einen Startzustand und eine Menge von Endzuständen erhält.

b) *3 Punkte.* Schreiben Sie eine Funktion die zwei Zustände u und v sowie einen Buchstaben des Eingabealphabets a erhält und die Transition (u, a, v) zu ihrem Automaten hinzufügt

c) *4 Punkte.* Schreiben Sie eine Funktion die als Argument einen Dateinamen erhält und den Übergangsgraphen Ihres Automaten im Dot Format in diese Textdatei schreibt.

Aufgabe 3 (*10 Punkte*) Implementieren Sie eine Funktion die einen Automaten aus einer textuellen Representation in Datei konstruiert. Das Format ist wie folgt definiert:

```
Start:<id>
States:<id>;<id>;...
Final:<id>;<id>;<id>;...
Alphabet:<c>;<c>;<c>;...
Transition:(<id>;<c>;<id>;),(<id>;<c>;<id>),...
```

Wobei `<id>` für einen beliebigen String aus den Buchstaben a,...,z,A,...,Z steht und <c> für einen einzelnen Buchstabe aus a,...,z,A,...,Z,0,...,9.
Der Automat aus dem ersten Beispiel, Abbildung 1, hätte zum Beispiel folgende textuelle representation:

```
Start:S
States:S;A;Err
```

```
Final:A
Alphabet:a;b;c
Transition:(S,a,A);(S,b,Err);(S,c,Err);
(A,a,Err);(A,b,S);(A,c,Err);
(Err,a,Err);(Err,b,Err);(Err,c,Err)
```

Testen Sie ihre Implementierung auf den in Aufgabe 1 konstruierten Automaten.

Aufgabe 4 (*15 Punkte*) Man nennt einen Endlichen Automaten deterministisch, wenn es für alle Zustände z und alle Buchstaben a des Alphabets Σ genau ein y gibt so, dass $(z, a, y) \in \Delta$. Ansonsten spricht man von einem nichtdeterministischen Endlichen Automaten. Bei solchen Automaten gibt es also mehrere mögliche Berechnungen für ein Wort, der Automat akzeptiert, wenn eine dieser Berechnungen akzeptierend ist. Nichtdeterministische Endliche Automaten haben den Nachteil, dass bei ihren Berechnungen der Automat die richtige Transition erraten muss. Der folgende Algorithmus konstruiert aus einem nichtdeterministischen Endlichen Automaten einen deterministischen Endlichen Automat:

```
w = list()
S0 = set()
S0.add(S)
states = S0
w = [S0]
T = set()
T' = set()
trans = set()
while len(w)!= 0:
    T = w.pop()
    for x in Alphabet:
        T' = nextState(T,x)
        trans.add((T,x,T'))
        if not T' in states:
            states.add(T')
            w.append(T')
```

Wobei die Funktion `nextState` so definiert ist:

```
function nextState(T,x):
  T'=set()
  for (a,b,c) in Delta:
  if a in T and b=x:
  T'.add(c)
```

Der entstehende Automat hat als Zustandsmenge $states$, den Startzustand S, die Übergangsrelation $trans$ und als Endzustand die Menge aller Zustände in $states$, die einen Endzustand des Ursprünglichen Automaten enthalten.

a) ✍ *3 Punkte.* Beschreiben Sie die Funktionsweise des Algorithmus.

b) ✍ *3 Punkte.* Wenden Sie den Algortihmus auf dem folgenden Automaten an:

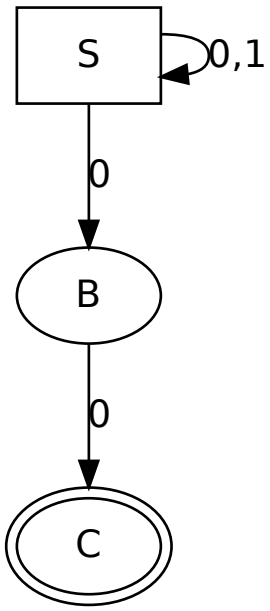

Abbildung 2: Der Übergangsgraph

c) *3 Punkte.* Ergänzen Sie zu ihrer Klasse eine Funktion, die feststellt, ob ihr Automat deterministisch ist.

d) *6 Punkte.* Implementieren Sie den Algorithmus für ihre ihre Automaten-Klasse aus Aufgabe 2.

Aufgabe 5 (*10 Punkte*) Die deterministischen Endlichen Automaten, die mit dem Algorithmus aus der letzten Aufgabe konstruiert werden, können sehr groß werden. Der folgende Algorithmus minimiert einen geben Automaten:

```
T = Tabelle aller Zustandspaare
Makiere alle Paare (s,s') bei denen s nicht in E und s' in E
Für alle unmarkierten Paare (s,s') und alle Buchstaben a tue
    Falls für markiertes (t,t'), (s,a,t) und (s',a,t') in Delta
        Makiere (s,s')
Wiederhole den letzen Schritt bis sich die Tabelle nicht mehr ändert
Vereinige alle unmarkierten Paare in einen Zustand
```

a) ✍ *5 Punkte.* Wenden Sie den Algorithmus auf den Deterministischen Automaten, den Sie in der letzten Aufgabe konstruiert haben, an.

b) *5 Punkte.* Implementieren Sie den Algorithmus für ihre Klasse.

Aufgabe 6 (*5 Punkte*) Implementieren Sie einen Algorithmus der ein Eingabewort und einen Automaten erhält und testet ob der Automat dieses Wort akzeptiert.
Hinweis: Machen konstruieren Sie erst den deterministische Automat und mininieren Sie ihn anschließend.