Stefan Waidele
Ensisheimer Straße 2
79395 Neuenburg am Rhein
Stefan@Waidele.info

AKAD University
Immatrikulationsnummer: 102 81 71

Lösungsvorschläge

22. August 2014

AKAD University

Inhaltsverzeichnis

1 FMI11 — Kellerautomaten

1.1 Definition von Automaten

In der Klausur ist es hilfreich, wenn man die formale Definition von Automaten in natürliche Sprache Übersetzen kann. Denn eigentlich steckt da schon alles drin, was man baucht:

Für jeden Automaten brauchen wir Zustände (Q, Menge der Zustände), ein Eingabealphabet (Σ, Groß–Sigma) und Vorschriften darüber, was beim Lesen von Eingaben getan werden soll (σ, Klein–Sigma). Des weiteren benötigen wir einen Startzustand (q_0) und einen bzw. mehrere Endzustände (F).

Somit ist ein Automat A definiert werden als $A = (Q, \Sigma, \sigma, q_0, F)$.

Bei Kellerautomaten benötigen wir zusätzlich das Kelleralphabet (Γ, Groß–Gamma, die Menge der Zeichen, die in den Keller geschrieben werden dürfen). Da schon im ersten Ausführungsschritt als erstes ein Zeichen vom Keller gelesen wird, ist ein Zeichennotwendig, das bereits zum Start des Automaten im Keller ist (k_0). Daher lautet die formale Definition eines Kellerautomaten $A = (Q, \Sigma, \Gamma, \sigma, q_0, k_0, F)$.

1.2 Definition von σ

In Klausuren sind Eingabealphabet Σ meist angegeben. In Q sind alle für die Lösung notwendigen Zustände und die Endzustände in F ergeben sich wieder meistens aus der Aufgabenstellung. Die eigentliche Arbeit liegt also in der Definition der Übergänge σ zwischen den Zuständen.

Bei σ handelt es sich um eine Funktion. Eine genau definierte Eingabe wird auf eine Ausgabe (deterministische Automaten) oder mehrere (nicht deterministische Automaten) Ausgaben abgebildet. Dies geschieht entweder über ein Diagramm oder eine Automatentafel.

In der Automatentafel sind somit alle Komponenten anzugeben, die man für eine solche Zuordnung benötigt. Für einen Kellerautomat sind dies:

Parameter			Funktionswert	
Zustand	Eingabe	Keller	Zustand	Keller
q_0	a	#	q_0	X
q_0	a	X	q_1	ε
„Wenn. . . “			„. . . dann“	

Tabelle 1: Automatentafel: Kellerautomat (Beispiel)

Wird der Automat gezeichnet, so müssen diese Informationen ebenfalls im Diagramm zu erkennen sein:

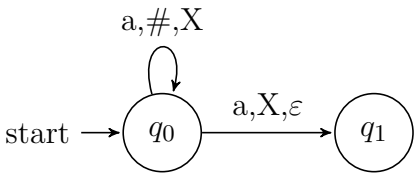

Abbildung 1: Diagramm: Kellerautomat (Beispiel)

1.3 Musterklausur 2, Komplexaufgabe 2.2

Gegeben ist $L = \{a^i b^i c^k | i, k \in N\} \subseteq \{a, b, c\}*$. Somit sollen alle Worte akzeptiert werden, die mit einem oder mehr a beginnen, dann gleich viele b enthalten und zuletzt auf ein oder mehr c enden.

Durch die Dreiteilung des Wortes bietet sich ein Automat mit mindestens drei Zuständen an. Jeweils einer zum Lesen jedes Buchstabens.

Der Automat muss speichern, wie viele a von der Eingabe gelesen wurden, um anschließend die Zahl der b kontrollieren zu können. Hierfür schreibt der Automat in Zustand q_a für jedes gelesene a ein X in den Keller. Hierbei ist zu beachten, dass in jedem Schritt zunächt ein Zeichen aus dem Keller geholt wird, welches wieder zurückgelegt wird. Dies ist beim ersten Durchlauf (unterer Pfeil) das Zeichen für den leeren Keller # und bei allen weiteren Durchläufen (oberer Pfeil) das zuvor geschriebene X.

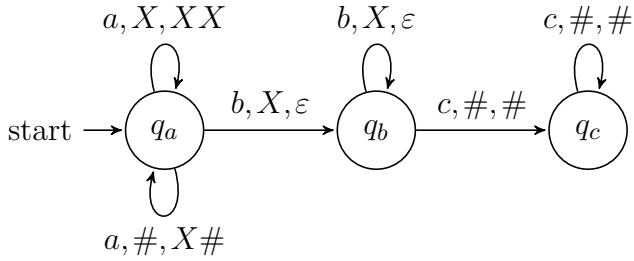

Abbildung 2: Diagramm: Kellerautomat ($a^i b^i c^k$)

Sobald das erste b gelesen wird, wechselt der Automat in den Zustand q_b. Hier werden dann die X aus dem Keller entnommen und nichts (also das leere Element ε) zurückgeschrieben. Bis zu dem Punkt, an dem der Keller leer ist und als Eingabe ein c gelesen wird.

In Zustand q_c werden dann die verbleibenden c gelesen, der Keller bleibt leer. Da bei leerem Keller und leerer Eingabe der Automat das Wort akzeptiert, ist kein weiterer Zustand notwendig.

Parameter			Funktionswert	
Zustand	Eingabe	Keller	Zustand	Keller
q_a	a	#	q_a	X#
q_a	a	X	q_a	XX
q_a	b	X	q_b	ε
q_b	b	X	q_b	ε
q_b	c	#	q_c	#
q_c	c	#	q_c	#

Tabelle 2: Automatentafel: Kellerautomat ($a^i b^i c^k$)

2 CPP01 — Mehrdimensionale Arrays

2.1 Aufgabenstellung

Klausuraufgabe aus CPP01:

> Das Programm enthält die Klasse Artikel mit den Attributen int Preis, int Artikelnummer und eine zweidimensionale Rabatttabelle mit 2 Zeilen und 5 Spalten, in der in der ersten Zeile angegeben wird um welchen Kundentyp es sich handelt (1,2,3,4,5) und in der zweiten, wie viel Rabatt der jeweilige Kundentyp bekommt (1%, 3%, 5%, 10%, 20%). Das Programm soll den Endpreis eines Artikels für den jew. Kundentyp ausgeben.

2.2 Eindimensionales Array

Ich würde das so machen, wie in Listing **??** gezeigt: Mit einem eindimensionalen Array für die Rabatte. Dann kann man den Prozentsatz direkt lesen und in Rechnungen verwenden. Der Kundentyp dient hierbei als Index für das Array.

```
1  #include <iostream>
2  using namespace std;
3
4  class Artikel {
5  private:
6     int artno;
7     int preis;
8     int staffel[5] = {1, 3, 5, 10, 20};
9
10 public:
11    // Konstruktoren
12    Artikel () {}
13    Artikel (int n, int p) {
14       artno = n;
15       preis = p;
16    }
```

```
17
18      // Getter
19      int getPreis() {
20          // Ohne Rabatt
21          return preis;
22      }
23      int getPreis(int kt) {
24          // Mit Rabatt
25          return preis * (100 - staffel[kt-1]) / 100;
26      }
27  };
28
29  int main(int argc, char* argv[])
30  {
31      Artikel artikel(1, 100);
32
33      cout << artikel.getPreis() << "\n";
34      cout << artikel.getPreis(1) << "\n";
35      cout << artikel.getPreis(2) << "\n";
36      cout << artikel.getPreis(3) << "\n";
37      cout << artikel.getPreis(4) << "\n";
38      cout << artikel.getPreis(5) << "\n";
39
40      return 0;
41  }
```

2.3 Zweidimensionales Array

Als Alternative mit zweidimensionalem Array könnte man in die erste Zeile die Kundentypen schreiben. Diese können dann auch anders nummeriert werden. In der zweiten Zeile steht dann der Rabattsatz. Um diesen zu ermitteln muss man durch das Array durchlaufen, bis man den Kundentyp gefunden hat. Hier mein Vorschlag dazu steht in Listing ??.

```
1  #include <iostream>
2  using namespace std;
3
4  class Artikel {
5  private:
6      int artno;
7      int preis;
```

```
8     int staffel[5] = {1, 3, 5, 10, 20};
9
10  public:
11    // Konstruktoren
12    Artikel () {}
13    Artikel (int n, int p) {
14       artno = n;
15       preis = p;
16    }
17
18    // Getter
19    int getPreis() {
20       // Ohne Rabatt
21       return preis;
22    }
23    int getPreis(int kt) {
24       // Mit Rabatt
25       return preis * (100 - staffel[kt-1]) / 100;
26    }
27  };
28
29  int main(int argc, char* argv[])
30  {
31    Artikel artikel(1, 100);
32
33    cout << artikel.getPreis() << "\n";
34    cout << artikel.getPreis(1) << "\n";
35    cout << artikel.getPreis(2) << "\n";
36    cout << artikel.getPreis(3) << "\n";
37    cout << artikel.getPreis(4) << "\n";
38    cout << artikel.getPreis(5) << "\n";
39
40    return 0;
41  }
```