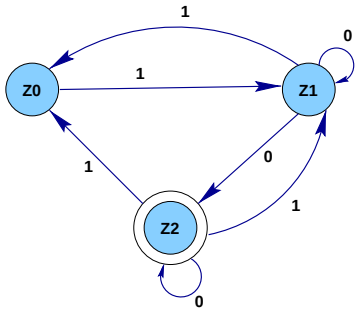
Universität Mannheim
Lehrstuhl für Praktische Informatik IV
Prof. Dr. W. Effelsberg
Christoph Kuhmünch, Gerald Kühne

Praktische Informatik I
WS 1999/2000

Übungsblatt 12 **Ausgabe: Mi, 26.01.00** **Abgabe: Di, 01.02.00, 18 Uhr**

Aufgabe 1: Endliche Automaten [5 Punkte]

Untenstehende Abbildung zeigt einen nichtdeterministischen endlichen Automaten $A = (Z, E, \delta, z_0, F)$. Konstruieren Sie einen äquivalenten deterministischen endlichen Automaten $A' = (Z', E', \delta', z_0', F')$. Zeichnen Sie A' und geben Sie zusätzlich die Belegung des Fünftupels $(Z', E', \delta', z_0', F')$ an.



Lösung:

Konstruieren Sie einen äquivalenten deterministischen Automaten $A' = (Z', E', \delta', z_0', F')$.

Zeichnen Sie A' und geben Sie zusätzlich die Belegung des Fünftupels $(Z', E', \delta', z_0', F')$ an.

Nach der Konstruktionsvorschrift auf Seite 4-93 im Skript ergibt sich die Menge Z' der möglicherweise notwendigen Zustände von A' aus der Potenzmenge von Z, indem man fuer jede Teilmenge $\{Z0, Z1, ...\}$ in 2^Z einen Zustand $[Z0Z1...]$ in Z' aufnimmt.

Es gilt also $Z' = \{[], [Z0], [Z1], [Z2], [Z0Z1], [Z0Z2], [Z1Z2], [Z0Z1Z2]\}$.

Die Übergangsfunktion δ des gegebenen NEA läßt sich unmittelbar aus dem gegebenen Automaten ablesen:

$\delta(\{Z0\}, 0) = \{\}$ $\delta(\{Z0\}, 1) = \{Z1\}$

$\delta(\{Z1\}, 0) = \{Z1, Z2\}$ $\delta(\{Z1\}, 1) = \{Z0\}$

$\delta(\{Z2\}, 0) = \{Z2\}$ $\delta(\{Z2\}, 1) = \{Z0, Z1\}$

Hieraus folgt zusätzlich:

$\delta(\{Z0, Z1\}, 0) = \{Z1, Z2\}$ $\delta(\{Z0, Z1\}, 1) = \{Z0, Z1\}$

$\delta(\{Z1, Z2\}, 0) = \{Z1, Z2\}$ $\delta(\{Z1, Z2\}, 1) = \{Z0, Z1\}$

$\delta(\{\}, 0) = \{\}$ $\delta(\{\}, 1) = \{\}$

Mit diesem δ gilt laut Skript für δ':

$\delta'([Z0], 0) = []$ $\delta'([Z0], 1) = [Z1]$

$\delta'([Z1], 0) = [Z1Z2]$ $\delta'([Z1], 1) = [Z0]$

$\delta'([Z2], 0) = [Z2]$ $\delta'([Z2], 1) = [Z0Z1]$

$\delta'([Z0Z1], 0) = [Z1Z2]$ $\delta'([Z0Z1], 1) = [Z0Z1]$

$\delta'([Z1Z2],0) = [Z1Z2] \quad \delta'([Z1Z2],1) = [Z0Z1]$

$\delta'([],0) = [] \quad \delta'([],1) = []$

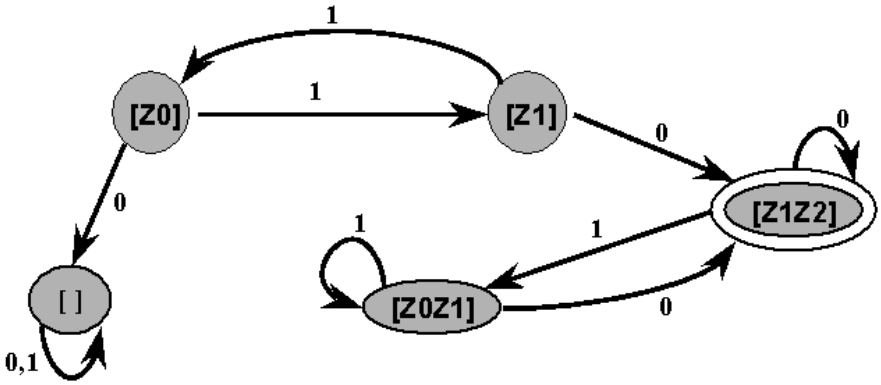
Die für A' relevanten Zustände sind also $Z' = \{[],[Z0],[Z1],[Z0Z1],[Z1Z2]\}$.

(Der Zustand [Z2] ist nicht relevant, da er von keinem Zustand außer sich selbst aus erreichbar ist, insbesondere also nicht vom Startzustand aus.)

Die Menge der möglichen Endzustände von A' ist mit der Definition im Skript $F' = \{[Z2],[Z0Z2],[Z1Z2],[Z0Z1Z2]\}$. Ohne die nicht relevanten Zustände gilt somit $F' = \{[Z1Z2]\}$.

Laut Skript gilt weiterhin $E' = E$ und $z_0' = [z_0]$ ($hier = [Z0]$).

Mit diesen Informationen können wir den zum gegebenen NEA äquivalenten DEA zeichnen.



Wie man aus dem DEA noch besser als aus dem originalen NEA erkennen kann, akzeptieren die Automaten Zeichenfolgen der folgenden Form:

Die Zeichenfolge beginnt mit einer ungeraden Anzahl von Einsen, darauf folgt eine Null und daraufhin eine beliebige Folge von Nullen und Einsen. Außerdem endet die Zeichenkette mit einer Null.

Andere Zeichenketten werden nicht akzeptiert.

Der Zustand [Z0] kann nicht eliminiert werden, da ansonsten (mit [Z0Z1] als Startzustand) z.B. die Zeichenkette “0” akzeptiert würde.

Der Zustand [Z1] kann ebenfalls nicht zugunsten von [Z0Z1] eliminiert werden, da ansonsten z.B. die Zeichenkette “110” akzeptiert würde.

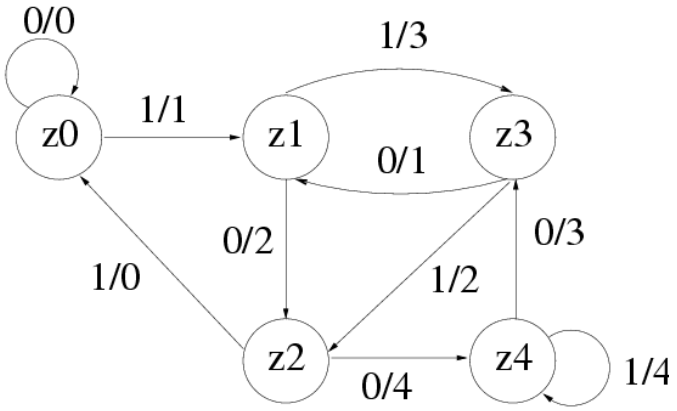
Der Zustand [] ist notwendig, da ohne ihn das Verhalten des Automaten im Zustand [Z0] bei Eingabe einer Null nicht definiert wäre, was bei einem DEA nicht sein darf.

Aufgabe 2: Mealy-Automaten (Abgabe per abox) [9 Punkte]

Im Vorlesungsskript ist auf Seite 4-105 ein Moore-Automat zur Berechnung von Restklassen modulo 5 angegeben.

(a) [3 Punkte] Geben Sie einen dazu äquivalenten Mealy-Automaten an!

Lösung:



Notation: e/a
e = Ereignis
a = Ausgabe

(b) [6 Punkte] Entwickeln Sie ein Java-Programm, welches Ihren Mealy-Automaten implementiert. Es soll ein als Kommandozeilenparameter angegebener Eingabe-Bitstring abgearbeitet werden und bei jeder Transition das entsprechende Ausgabezeichen ausgeben werden. Folgende Klassenstruktur ist vorgegeben:

```
public class Mod5Mealy {
  public static String compute(String inputString);
  public static void main(String[] argv);
}
```

Lösung:

```
//

/*****************************************************************************
 * Praktische Informatik I, WS 1999/2000
 * Übungsblatt 12, Aufgabe 2b
 * Dateiname : Mod5Mealy.java
 * Autor     : dmi
 * Datum     : 31. Januar 2000
 *****************************************************************************
 * Beschreibung : Das Programm simuliert einen Mealy-Automaten, der die
 *                Restklassen modulo 5 einer als Kommandozeilenparameter
 *                uebergebenen binaeren Zeichenkette berechnet.
 *****************************************************************************/


public class Mod5Mealy{

    /*************************************************************************
     * Name : compute
     * Beschreibung: Eigentliche Simulation eines Mealy-Automaten zur
     *               Berechnung der Restklassen mod 5. Als Eingabeparameter
     *               wird eine Zahl in Binärdarstellung erwartet.
     *               Als Rückgabewert wird eine Zeichenkette mit allen
     *               Ausgabezeichen der aufgetretenen Transitionen
     *               zurückgeliefert.
     *               Der Eingabestring wird dabei von links nach rechts
     *               abgearbeitet.
     *               Beispiel: Eingabestring = "1101", Dezimal = 13
```

```
     *                          Ausgabestring = "1313", d.h. 13 mod 5 = 3
     ************************************************************************/

    public static String compute(String inputString){

int zustand = 0;           // als Startzustand wird z0 angenommen
        int folgezustand = 0;      // Temporäre Variable für Folgezustand
        String outputString = ;  // Initialisierung des Rückgabestrings

char ausgabe;  // Ausgabezeichen für einzelne Transition

for(int i=0; i<inputString.length(); i++){
    ausgabe = ' ';
            switch (inputString.charAt(i)) {

    case '0': {  // Eingabestring enthält an aktueller Stelle 0

switch (zustand) {
case 0:  { ausgabe = '0'; folgezustand = 0; break; }
case 1:  { ausgabe = '2'; folgezustand = 2; break; }
case 2:  { ausgabe = '4'; folgezustand = 4; break; }
case 3:  { ausgabe = '1'; folgezustand = 1; break; }
case 4:  { ausgabe = '3'; folgezustand = 3; break; }
}
break;

    }
    case '1': {  // dto. für 1 im Eingabestring

switch (zustand) {
case 0: { ausgabe = '1'; folgezustand = 1; break; }
case 1: { ausgabe = '3'; folgezustand = 3; break; }
case 2: { ausgabe = '0'; folgezustand = 0; break; }
case 3: { ausgabe = '2'; folgezustand = 2; break; }
case 4: { ausgabe = '4'; folgezustand = 4; break; }
}
                break;
    }
    }
            zustand = folgezustand;    // gehe zum ermittelten neuen Zustand
    outputString += ausgabe;   // Hänge Ausgabezeichen dieser
    // Transition an den bisherigen Rückgabestring
}
return (outputString);
    }

    /************************************************************************
     * Name : main
     * Beschreibung: Ruft die Methode zur Simulation des Mealy-Automaten auf
     ************************************************************************/
    public static void main(String[] args){

        System.out.println(compute(args[0]));

    }
}
//
```

Aufgabe 3: Programmverifikation [6 Punkte]

Untenstehendes Programm berechnet für alle Eingabeparameter $a, b > 0$ das kleinste gemeinsame Vielfache (kgV). Geben Sie Zusicherungen an denen mit `/* <A> */` markierten Stellen an. Geben Sie weiterhin eine Schleifeninvariante an.

```
x = a;
y = b;
/* a > 0, b > 0, x = a,  y = b */
while (x != y)
  if (x < y)
    x = x+a;
  else
    y = y+b;

/* x = y, x >= a > 0, y >= b > 0 */
kgV = x;

/* eine mögliche Schleifeninvariante: x >= a> 0, y >= b > 0 */
/* eine weitere Schleifeninvariante:  es existieren n,m: x = n*a, y = m*b */
```