Prof. Dr. Franz Schweiggert   Abteilung Angewandte Informationsverarbeitung   17. Dezember 2003
Dr. Andreas Borchert, Michael Wiedemann   Blatt 10

Allgemeine Informatik I (WS 2003/2004)

Abgabetermin: 14. Januar 2004

22 Weihnachten mit zellulären Automaten – 20 Punkte

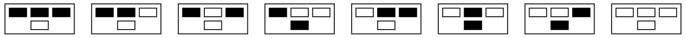
Bei zellulären Automaten hat jede Zelle einen Zustand. Im einfachen Falle ist dies 0 oder 1 oder bei komplizierteren Varianten ein Wert zwischen 1 und k (vorzustellen als k verschiedene Farben). Die Zellen ergeben sich durch die Zerlegung eines vorgegebenen Raumes (beispielsweise des $\mathbb{R}^2$) in gleichförmige und gleichgroße Teilmengen (beispielsweise Quadrate oder gleichseitige Sechsecke im $\mathbb{R}^2$). Aus dem Raum und der Zerlegung ergeben sich Nachbarschaftsbeziehungen zwischen den Zellen, so daß jede Zelle genau n Nachbarn besitzt.

In jedem Transitionsschritt eines zellulären Automaten wird der neue Zustand einer Zelle bestimmt durch den eigenen Zustand und den der n Nachbarn[1]. Alle Transitionen erfolgen zeitgleich, d.h. jede Zelle berücksichtigt nur die Zustände der Nachbarn *vor* deren Transition.

Beschrieben wird ein zellulärer Automat durch die Topologie (Ausgangsraum, Nachbarschaftsbeziehungen) und die Transitionsfunktion. Transitionsfunktionen gibt es für eine vorgegebene Topologie nur endlich viele. Bei n Nachbarn und k Farben gibt es nur k^{n+1} verschiedene Ausgangssituationen und k verschiedene Funktionswerte. Entsprechend ergeben sich daraus $k^{k^{n+1}}$ verschiedene Transitionsfunktionen und damit zelluläre Automaten.

Für dieses Übungsblatt betrachten wir den einfachsten Fall mit $n = 2$ und $k = 2$, den sogenannten elementaren zellulären Automaten. Vorstellen können wir uns dies als ein fortlaufendes eindimensionales Band, bei dem jede Zelle genau zwei Nachbarn hat:

Bei zwei Farben ($k = 2$) wird typischerweise schwarz und weiß verwendet. Gegeben sei nun beispielsweise folgende Transitionsregel: Nur wenn *genau* eine Zelle von den beiden Nachbarschaftszellen und der eigenen Zelle schwarz ist, dann wird die eigene Zelle im nächsten Schritt schwarz. Ansonsten bleibt sie weiß. Da es bei einem elementaren zellulären Automaten nur $k^{n+1} = 2^3 = 8$ verschiedene Zustände gibt, können wir Transitionsfunktionen in der folgenden übersichtlichen Form darstellen:

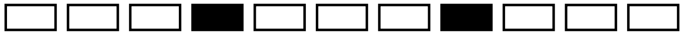


Hier wird für jede Kombination von drei Zellen (Nachbarzellen einschließlich eingeschlossener Zelle) der zugehörige Funktionswert der Transitionsfunktion darunter angegeben.

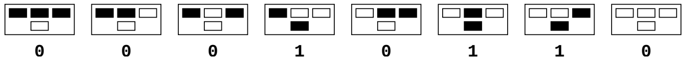
Angenommen, wir beginnen mit einem Ausgangszustand, bei dem nur eine einzige Zelle schwarz ist und alle anderen weiß sind:

Im nächsten Transitionsschritt bleibt dann die mittlere Zelle schwarz und die beiden Nachbarzellen wechseln von weiß auf schwarz:

Im darauffolgenden Schritt geht es nur am Rand des schwarzen Trios weiter, da im inneren Bereich immer mehr als eine Zelle schwarz ist:



Für elementare zelluläre Automaten gibt es insgesamt $k^{k^{n+1}} = 2^{2^3} = 256$ verschiedene Transitionsfunktionen. Um die 256 verschiedenen Automaten über die Nummer zu identifizieren, wird eine Binärkodierung verwendet. Schwarz steht dabei für 1, weiß für die 0. Diese Binärziffern können unmittelbar unter die Regel geschrieben werden und so zu einer Regelnummer zusammengefaßt werden. So ergibt sich aus



die Binärzahl 00010110, die in das dezimale System konvertiert 22 ergibt. Zu beachten ist hier auch, daß die einzelnen Regeln nicht zufällig angeordnet sind, sondern entsprechend den Binärwerten der drei Nachbarzellen von 111 bis 000. Das niedrigstwertige Bit der Regelnummer gibt also den Funktionswert von 000 an, das nächste Bit spezifiziert den Funktionswert von 001 und so weiter bis zum höchstwertigen Bit, das den Funktionswert von 111 festlegt.

Schreiben Sie ein Oberon-Programm, das eine Regelnummer aus dem Bereich 0 bis 255 einliest, mit einem Anfangszustand beginnt, bei dem nur eine einzige Zelle schwarz ist und dann eine festgelegte Zahl von Transitionen durchführt und sie jeweils ausgibt. So könnte das aussehen:

[1] Im einfachen Falle werden nur die direkten Nachbarn betrachtet. Im verallgemeinerten Falle gehen in die Transitionsfunktion alle Zellen ein, die nicht weiter als der Radius r entfernt sind.

```
doolin$ ElementaryCellularAutomaton
Rule out of 0..255: 22

                                       X
                                      XXX
                                     X   X
                                    XXX XXX
                                   X       X
                                  XXX     XXX
                                 X   X   X   X
                                XXX XXX XXX XXX
                               X               X
                              XXX             XXX
                             X   X           X   X
                            XXX XXX         XXX XXX
                           X       X       X       X
                          XXX     XXX     XXX     XXX
                         X   X   X   X   X   X   X   X
                        XXX XXX XXX XXX XXX XXX XXX XXX
                       X                               X
                      XXX                             XXX
                     X   X                           X   X
                    XXX XXX                         XXX XXX
                   X       X                       X       X
                  XXX     XXX                     XXX     XXX
                 X   X   X   X                   X   X   X   X
                XXX XXX XXX XXX                 XXX XXX XXX XXX
               X               X               X               X
              XXX             XXX             XXX             XXX
             X   X           X   X           X   X           X   X
            XXX XXX         XXX XXX         XXX XXX         XXX XXX
           X       X       X       X       X       X       X       X
          XXX     XXX     XXX     XXX     XXX     XXX     XXX     XXX
         X   X   X   X   X   X   X   X   X   X   X   X   X   X   X   X
        XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX
       X                                                               X
      XXX                                                             XXX
     X   X                                                           X   X
    XXX XXX                                                         XXX XXX
   X       X                                                       X       X
  XXX     XXX                                                     XXX     XXX
 X   X   X   X                                                   X   X   X   X
XXX XXX XXX XXX                                                 XXX XXX XXX XXX
doolin$
```

Hinweise:

Da Ihnen nicht unendlich viel Speicher und Zeit zur Verfügung stehen, können Sie nur mit einer endlichen Zahl von Zellen arbeiten. Bei den elementaren zellulären Automaten empfiehlt es sich dabei, die Zellen in einem Ring zusammenzuschließen. Wenn Sie also m Zellen haben, dann hat die Zelle 0 auch die Zelle $m-1$ als Nachbarn und umgekehrt. Das ist leicht implementiert, wenn Sie den `MOD`-Operator verwenden: `(index-1) MOD m` und `(index+1) MOD m` tun genau das, was Sie sich erhoffen.

Diese Aufgabe wird deutlich leichter, wenn Sie sich von Anfang an ausführlich Gedanken über die notwendigen Datenstrukturen machen. Es ist sinnvoll, den Zustand des Automaten als CHAR-Array zu verwalten. Dann können sie ihn bequem mit `Write.Line` ausgeben, ohne dafür eine Schleife investieren zu müssen. Ein einzelnes Zustands-Array allein genügt Ihnen nicht, da Sie bei der Transition solange noch den alten Zustand benötigen, bis die Transition abgeschlossen ist. Beachten Sie dabei, daß sie Arrays einander zuweisen können, wenn sie vom identischen Typ sind. Dies läßt sich am einfachsten durch eine `TYPE`-Deklaration sicherstellen. Wenn Sie beispielsweise

```
VAR
   x: ARRAY 3 OF INTEGER;
   y: ARRAY 3 OF INTEGER;
```

haben, dann können Sie beide Arrays nicht einander zuweisen, wohl aber unter Verwendung einer gemeinsamen `TYPE`-Deklaration:

```
TYPE
   ThreeIntegers = ARRAY 3 OF INTEGER;
VAR
   x: ThreeIntegers;
   y: ThreeIntegers;
```

Sie müssen in dieser Aufgabe die Regelnummer binär zerlegen und später bei einem Transitionsschritt den Zustand von drei Nachbarn entsprechend der binären Darstellung in eine Zahl konvertieren, um sie als Index für Ihre Regelsammlung zu verwenden. Diese Konvertierungen werden eleganter, wenn Sie die Standardfunktion `ASH` in Oberon verwenden. Der Name `ASH` steht dabei für *arithmetic shift*. Die Funktion `ASH(`n`, `k`)` liefert für ganzzahlige n und k den Wert $n * 2^k$. Entsprechend liefert `ASH(1, `k`)` den Wert von 2^k. Den Wert des i-ten Bits einer ganzen Zahl n können Sie dann bequem mit `n DIV ASH(1, i) MOD 2` ermitteln.

Mehr Hintergrund zu den elementaren zellulären Automaten gibt es unter `http://mathworld.wolfram.com/ElementaryCellularAutomaton.html`.

Frohe Weihnachten!