

Κυτταρικά αυτόματα

- [Σκοπός της εργασίας](#)
- [Περιγραφή του συστήματος](#)
- [Υλοποίηση της πλατφόρμας](#)
 - [Οδηγίες](#)
 - [Δείτε ακόμη ...](#)

Σκοπός της εργασίας

Σκοπός της εργασίας είναι η υλοποίηση και ο πειραματισμός με διάφορα μοντέλα κυτταρικών αυτομάτων σε διδιάστατο χώρο (κάνναβο).

Δίνεται μία πλατφόρμα μοντελοποίησης και προσομοίωσης η οποία περιλαμβάνει ήδη έναν αριθμό μοντέλων κυτταρικών αυτομάτων (μεταξύ των οποίων και το "παιχνίδι της ζωής") και επιτρέπει τον πειραματισμό με αυτά και την υλοποίηση νέων μοντέλων.

Η αξία ενός μοντέλου κυτταρικού αυτομάτου εξαρτάται τόσο από τα μακροσκοπικά φαινόμενα που προκύπτουν αυθόρμητα (π.χ. ένα μοντέλο όπου το σύστημα πεθαίνει αμέσως δεν είναι ενδιαφέρον), όσο και από την ικανότητα των συστημάτων να "αντιστέκονται" στις διαταραχές (π.χ. ένα ζωντανό σύστημα με πολύπλοκη δυναμική μορφή να μην καταλήγει σε σταθερό "μπλοκ" αν προσθέσουμε ή αφαιρέσουμε ένα ζωντανό κύτταρο) και να δίνουν τις ίδιες ή παρόμοιες μορφές ανεξαρτήτως αρχικής κατάστασης. Αρα, ο μελετητής πρέπει να σχεδιάσει και να μελετήσει διαφορετικά μοντέλα σε διαφορετικές αρχικές συνθήκες και για διαφορετικές τιμές των παραμέτρων του συστήματος, όπως περιγράφεται αναλυτικά παρακάτω.

Περιγραφή του συστήματος

Περιβάλλον

Το περιβάλλον είναι ο διδιάστατος χώρος μέσα στον οποίο υπάρχει ένας αριθμός από κύτταρα (απεικονίζονται με μαύρο). Η (αυτόματη) γένεση και ο θάνατος κυττάρων στις διάφορες θέσεις ανά πάσα χρονική στιγμή ορίζουν τη δυναμική συμπεριφορά του συστήματος. Το είδος των σχηματισμών, η ανάπτυξή τους στο χρόνο και η ανεκτικότητα τους σε διαταραχές αποτελούν κριτήρια αξιολόγησης για ένα μοντέλο.

Μοντέλα διδιάστατων κυτταρικών αυτομάτων

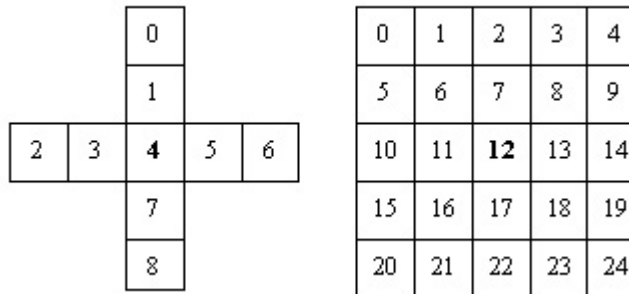
- **Αυτόματο τύπου LIFE.** Σε ένα τέτοιο αυτόματο, η επόμενη κατάσταση ενός κυττάρου (ζωντανό/νεκρό) εξαρτάται από τον αριθμό των ζωντανών κυττάρων στη γειτονιά του και από την τρέχουσα κατάστασή του. Η γειτονιά ενός κυττάρου ορίζεται από δύο παραμέτρους, το **βάθος πεδίου (depth)** (ακτίνα αντίληψης του κυττάρου γύρω του) και τη **διαγώνια συνδετικότητα (diagonal connectivity)** (εάν το κύτταρο βλέπει και σε διαγώνιες θέσεις ή μόνον οριζόντια και κάθετα, σε τέσσερις δηλαδή κατευθύνσεις). Έτσι για βάθος πεδίου 2 και όχι διαγώνια συνδετικότητα ο αριθμός των κυττάρων που γίνονται αντιληπτοί από ένα κύτταρο είναι $4 \times 2 = 8$ γείτονες, ενώ για βάθος πεδίου 2 και διαγώνια συνδετικότητα ο αριθμός των κυττάρων που γίνονται αντιληπτοί από ένα κύτταρο είναι $(2 \times 2 + 1)^2 = 25$ γείτονες, όπου σε αυτούς συμπεριλαμβάνεται κατ' αρχήν και το ίδιο το κύτταρο που εξετάζεται (μπορούμε να το εξαιρέσουμε αν θέλουμε).

Άλλη δευτερογενής παράμετρος είναι η ύπαρξη θορύβου κατά την αντίληψη ή κατά την "υλοποίηση" : στην πρώτη περίπτωση, ένα κύτταρο αντιλαμβάνεται την κατάσταση ενός γειτονικού κυττάρου με κάποια πιθανότητα λάθους, ενώ στη δεύτερη "αποφασίζει" για την επόμενη του κατάσταση με κάποια πιθανότητα λάθους. Τέλος ο αριθμός των δυνατών καταστάσεων ενός κυττάρου είναι 2, αλλά μπορούμε να ορίσουμε και αυτόματα με περισσότερες καταστάσεις (το ανάλογο των συνεχών κυτταρικών αυτομάτων αντί των διακριτών).

- **Λογικό αυτόματο.** Σε ένα τέτοιο αυτόματο, η επόμενη κατάσταση ενός κυττάρου (ζωντανό/νεκρό) εξαρτάται από τις ακριβείς "τιμές" των κυττάρων στη γειτονιά του, δηλαδή από το ποιά ακριβώς από αυτά είναι ζωντανά. Έτσι για ένα αυτόματο με βάθος πεδίου 2 και διαγώνια συνδετικότητα, ο αριθμός των κανόνων που απαιτούνται είναι 2^{25} κανόνες, όπου το 25 είναι το πλήθος των γειτόνων όπως υπολογίσθηκε προηγούμενα, οπότε κάθε κανόνας έχει 25 bits εισόδου και ένα bit εξόδου.

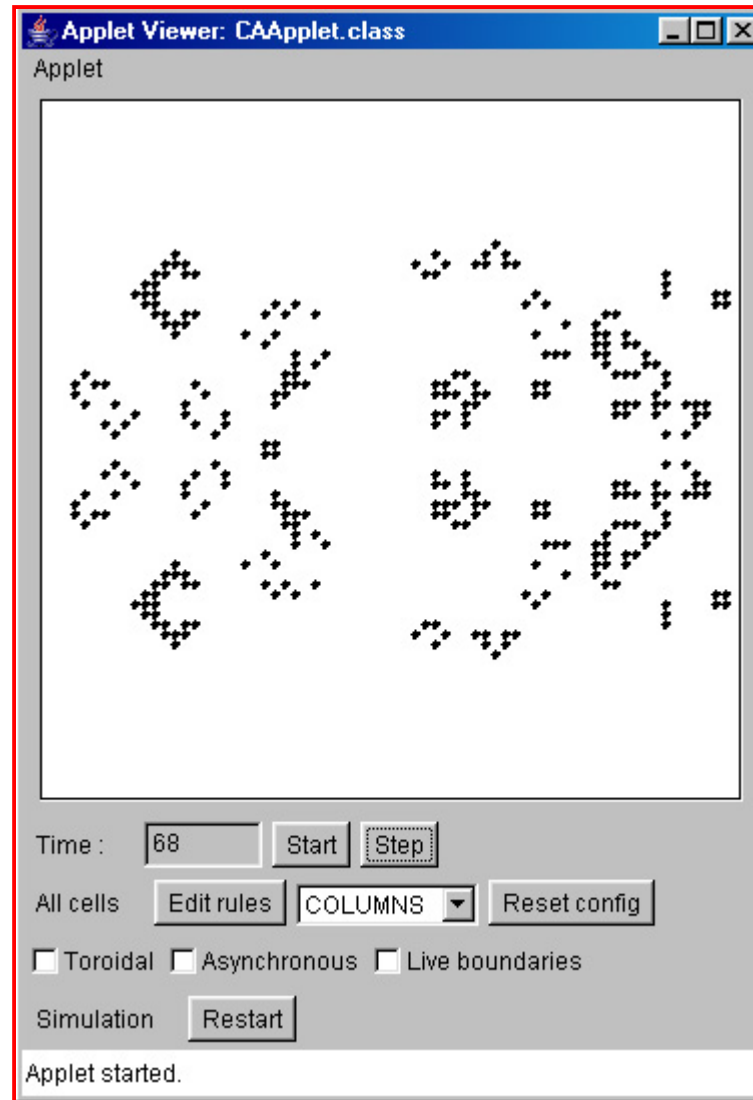
Επειδή αυτοί οι αριθμοί είναι υπέρογκοι, όπως επίσης και για λόγους που θα γίνουν προφανείς στα παραδείγματα παρακάτω, στο σύστημά μας έχουμε περιορίσει τον αριθμό κανόνων σε 100, και έχουμε ορίσει ότι κάθε κανόνας έχει έξοδο 1 και είσοδο τόσα bits όσα απαιτούνται από το βάθος πεδίου και τη διαγώνια συνδετικότητα του εκάστοτε μοντέλου. Αυτό σημαίνει ότι ορίζουμε τους (λίγους) εκείνους κανόνες για τους οποίους υφίσταται ζωντανό κύτταρο, ενώ στη γενική περίπτωση έχουμε απουσία ζωής. Επιπλέον, η τιμή ενός bit εισόδου για έναν κανόνα μπορεί να είναι 0, 1 ή *, που σημαίνει ότι το κύτταρο στην αντίστοιχη θέση θα πρέπει να είναι 0, 1 ή ο, τидήποτε, αντίστοιχα.

Σε κάθε περίπτωση, η διάσχιση των γειτόνων ενός κυττάρου γίνεται κατά γραμμές (αύξον -x) από πάνω προς τα κάτω (αύξον -y), όπως φαίνεται στο παρακάτω σχήμα. Οι παράμετροι θορύβου ορίζονται όπως και προηγούμενα.



Υλοποίηση της πλατφόρμας

Κάνετε "κλικ" μέσα στην εικόνα για να ξεκινήσει η προσομοίωση



Για την πρόσβαση στον κώδικα, απαιτείται η χρήση του username/password που έχει δοθεί στο μάθημα.

[Source Java code](#)

Εγχειρίδιο χρήσης

Η οθόνη του applet παρουσιάζει μία λευκή περιοχή στο πάνω μέρος όπου βρίσκονται τα κύτταρα (μαύρες κουκίδες).

Το κάτω μέρος περιλαμβάνει έναν αριθμό συνιστωσών ελέγχου της προσομοίωσης.

- **Το ρολόι του συστήματος.**

- **Ενα κουμπί start/stop της προσομοίωσης.**

Σε κάθε βήμα της προσομοίωσης, όλα τα κύτταρα εκτελούν σύγχρονα, δηλαδή παράλληλα (default), ή ασύγχρονα, δηλαδή με τυχαία σειρά (αν το checkbox asynchronous είναι on).

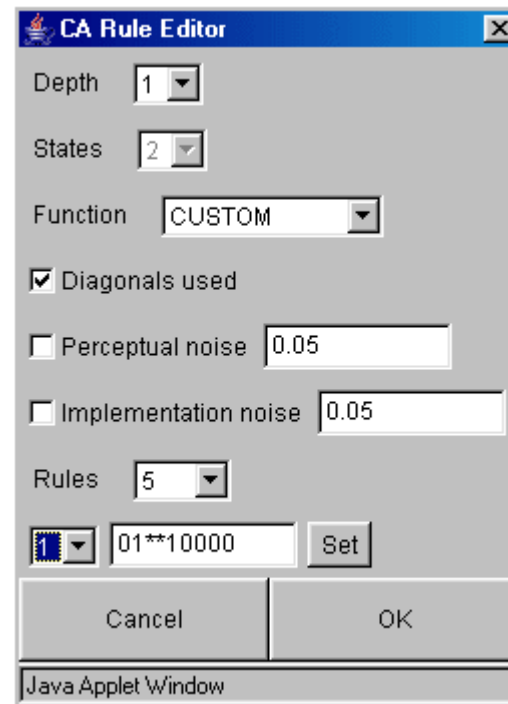
- **Ενα κουμπί step της προσομοίωσης.**

Προχωρά την προσομοίωση κατά μία μονάδα χρόνου.

- **Ενα κουμπί "Edit rules".**

Χρησιμεύει για επεξεργασία των κανόνων του αυτομάτου και ανοίγει ένα dialog που επιτρέπει το χειρισμό των παραμέτρων του κανόνα. Οι προ-προγραμματισμένοι κανόνες "life", "life2", "life3", "logical", "random", δεν επιτρέπουν το χειρισμό των παραμέτρων τους από το χρήστη, αλλά μόνον την επιλογή τους για πειραματισμό. Το ίδιο ισχύει και για τον κανόνα "custom life", που μπορεί να προγραμματισθεί από το χρήστη άμεσα (εσωτερικά στο πρόγραμμα) κατ' αναλογίαν με τους κανόνες "life", "life2" και "life3". Αρα μόνον ο προγραμματιζόμενος λογικός κανόνας "custom" επιτρέπει το χειρισμό των παραμέτρων του.

ΠΡΟΣΟΧΗ! Για άγνωστο μέχρι στιγμής λόγο, όταν επιλεγεί ο κανόνας "custom", τα controls του διαλόγου δεν φαίνονται κανονικά, οπότε ο χρήστης πρέπει να κάνει resize το παράθυρο, έστω και μόνο 1 pixel, για να εμφανισθούν στα σωστά μεγέθη και θέσεις.



- **Εναν επιλογή σχηματισμού (configuration).**

Χρησιμεύει για την αρχικοποίηση του συστήματος με έναν σχηματισμό. Δίνονται οι επιλογές "CLEAR" (τίποτε ζωντανό), "RANDOM" (τυχαία κύτταρα), "CHUNKS" (2 ζωντανά blocks), "COLUMNS" (κολώνες ζωής), "2 GLIDERS" (δύο gliders, δηλαδή σχηματισμοί που κυλίνουν και αυτο-αναπαράγονται στο παιχνίδι της ζωής) και "GRID" (κάνναβος ζωής).

- **Τρεις παράμετροι on/off : toroidal, asynchronous, live boundaries.**

- **Toroidal** : Στο αυτόματο αυτό, θεωρείται ότι ένα συνοριακό κύτταρο δε βλέπει "τοίχο", αλλά γειτονεύει με το αντίστοιχο κύτταρο από τη συμμετρική πλευρά. Π.χ. ένα κύτταρο στην τελευταία δεξιά στήλη, δε βλέπει τοίχο στα δεξιά του αλλά βλέπει το κύτταρο της αριστερότερης στήλης στην ίδια γραμμή. Δηλαδή το περιβάλλον αποτελεί ένα γεωμετρικό torus (toroidal κυτταρικό αυτόματο).
- **Asynchronous** : Στο αυτόματο αυτό, τα κύτταρα δεν μεταβαίνουν στην επόμενη κατάστασή τους παράλληλα (σύγχρονα), αλλά σειριακά (ασύγχρονα), δηλαδή ένα-ένα με τυχαία σειρά. Αυτό σημαίνει ότι ένα κύτταρο μπορεί να επηρεασθεί από την κατάσταση ενός γειτονικού κυττάρου που άλλαξε ήδη τιμή πιο γρήγορα. Δοκιμάστε τα αποτελέσματα που προκύπτουν με τους δύο αυτούς τρόπους προσομοίωσης.
- **Live boundaries** : Στο αυτόματο αυτό, θεωρείται ότι ένα συνοριακό κύτταρο δε βλέπει "τοίχο", αλλά γειτονεύει με ζωντανό κύτταρο στην αντίστοιχη θέση, π.χ. ένα κύτταρο στην τελευταία δεξιά στήλη, δε βλέπει τοίχο στα δεξιά του αλλά βλέπει ένα ζωντανό κύτταρο (που παραμένει αθάνατο!).

- **Ένα κουμπί "Restart".**

Επανεκκίνηση της προσομοίωσης. Αρχικοποιεί το ρολόι στο 0, αλλά αφήνει άθικτο τον τρέχοντα σχηματισμό.

Οι δυνατότητες αυτές της πλατφόρμας επιτρέπουν στο χρήστη να πειραματισθεί συστηματικά με διάφορα μοντέλα κυτταρικών αυτομάτων, σε προκαθορισμένους σχηματισμούς περιβάλλοντος, για διαφορετικές τιμές των επιμέρους παραμέτρων ή με άλλους περιορισμούς της επιλογής του.

Οδηγίες

Η πλατφόρμα περιλαμβάνει τρεις κλάσεις:

- **Κλάση CAApplet.** Αυτή είναι η κλάση που υλοποιεί το interface του συστήματος (applet) και την προσομοίωση του κυτταρικού αυτομάτου. Ο χρήστης δε χρειάζεται να ξέρει τα εσωτερικά της λειτουργίας αυτής της κλάσης, αλλά την χρησιμοποιεί αυτούσια.
- **Κλάση CARule.** Αυτή είναι η κλάση που υλοποιεί το κυτταρικό αυτόματο αυτό καθαυτό, δηλαδή τον κανόνα ή το μοντέλο του. Αυτή είναι και η κλάση στην οποία ο χρήστης μπορεί να παρέμβει, είτε έμμεσα χρησιμοποιώντας τον CARuleEditor, είτε άμεσα στον κώδικα αν θέλει.
- **Κλάση CARuleEditor.** Αυτή είναι η κλάση που υλοποιεί τον editor του κυτταρικού αυτομάτου, δηλαδή του αντιστοίχου κανόνα ή μοντέλου. Ο χρήστης δε χρειάζεται να ξέρει τα εσωτερικά της λειτουργίας αυτής της κλάσης, αλλά την χρησιμοποιεί αυτούσια.

Ζητούνται

1. Να υλοποιηθεί ένα μοντέλο κυτταρικού αυτομάτου που να δίνει σταθερούς τελικούς σχηματισμούς για κάποια αρχικά περιβάλλοντα. Να χρησιμοποιηθεί ο CARuleEditor και το custom λογικό μοντέλο. Τα παραδείγματα που δίνονται παρακάτω μπορούν να χρησιμεύσουν σαν αφετηρία ή σαν πηγή έμπνευσης.
2. Να υλοποιηθεί ένα δεύτερο μοντέλο που να δίνει "πολύπλοκους" σχηματισμούς για κάποιον ή κάποιους από τους αρχικούς σχηματισμούς που δίνονται (chunks, columns, grid). Ένας σχηματισμός μπορεί να θεωρηθεί "πολύπλοκος" όταν διατηρείται εν ζωή επί μακρόν αλλάζοντας συνεχώς μορφή, π.χ. ο κανόνας "life2" οδηγεί σε πολύπλοκο σχηματισμό εκκινώντας από ένα αρχικό grid, ενώ ένας λογικός κανόνας "σβήνει" πολύ γρήγορα. Επίσης μπορείτε να χρησιμοποιήσετε τα παραδείγματα που δίνονται στη συνέχεια. Μπορείτε να βρείτε μοντέλο που να δίνει πολύπλοκες μορφές εκκινώντας από τυχαίους αρχικούς σχηματισμούς; Μπορείτε να βρείτε μοντέλο που να δίνει αυτο-αναπαραγόμενες μορφές ;
3. **Προαιρετικά.** Να υλοποιηθεί προγραμματιστικά μία παραλλαγή του κανόνα "life" της επιλογής σας, που να υλοποιεί κάποια ποιοτικά διαφορετική περίπτωση μοντέλου, π.χ. life με μνήμη του προηγούμενου κύκλου. Η ποιοτική διαφορά θα έγκειται στην εμφάνιση διαφορετικών

φαινομένων ή σχηματισμών με το νέο μοντέλο, σε σχέση με τα "life", "life2", "life3".

Παραδοτέα.

1. Ένα κείμενο περιγραφής των μοντέλων σας. Ένα μοντέλο ορίζεται από τις τιμές των παραμέτρων του και τον ή τους κανόνες που το διέπουν (στην περίπτωση του λογικού μοντέλου, οι κανόνες αυτοί είναι λογικοί, ενώ στην περίπτωση του μοντέλου τύπου παραλλαγής "life" ο ή οι κανόνες είναι "ποιοτικοί"). Αν χρειάζεται για λόγους πληρότητας ή απλότητας της παρουσίασης, μπορούν να δοθούν πολλές εκδοχές ενός μοντέλου που έχουν προκύψει και αξιολογηθεί σταδιακά, καθώς και μία περιγραφή της πορείας της σκέψης που τους γέννησε. Επίσης να συμπεριληφθούν αποτελέσματα συστηματικής εφαρμογής των μοντέλων, συγκριτικά ή μη, σε οποιαδήποτε μορφή κρίνετε σκόπιμο (κατ' αρχήν εικόνες σχηματισμών ή και άλλα αποτελέσματα κατά την κρίση σας). Καλό είναι για ένα μοντέλο να έχετε ει δυνατόν διερευνήσει πλήρως όλες τις δυνατότητές του, δηλαδή σχηματισμούς ή άλλα φαινόμενα που προκύπτουν σε ένα ικανό εύρος αρχικών σχηματισμών ή/και τιμών παραμέτρων κατά την κρίση σας, π.χ. για αντιπροσωπευτικούς αρχικούς σχηματισμούς πέραν αυτών που δίνονται, σε κυκλικό ή μη κόσμο.
2. Για όσους υλοποιήσουν το προαιρετικό μοντέλο, το αρχείο CARule.java (κώδικας).
3. Επίδειξη των μοντέλων σας.

Περιγραφή της κλάσης CARule

Η κλάση CARule ορίζει τα εξής δεδομένα σύμφωνα με όσα έχουν περιγραφεί παραπάνω:

```

private int depth = DEFAULT_DEPTH; // = 2;

private int states = DEFAULT_STATES; // = 2;
final static int ANY = 500; // '*', i.e. any input state (0, 1 or other)

private boolean diagonal_connectivity = true;

private boolean hasPerceptionNoise = false;
private boolean hasImplementationNoise = false;
private double perceptionNoise = 0.05;
private double implementationNoise = 0.05;

final static int MAX_RULES = 100;
final static int MAX_RULE_LENGTH = 26;
// = 25 input states + output (if applicable)
private int lines[][] = new int[MAX_RULES][MAX_RULE_LENGTH];
private int numberOfLines = 10;

final static int LIFE = 0;
final static int LIFE2 = 1;
final static int LIFE3 = 2;
final static int CUSTOM_LIFE = 3;
final static int LOGICAL = 4;
final static int RANDOM = 5;
final static int CUSTOM = 6;
private int function = LIFE;
final static String Functions[] =
    {"LIFE", "LIFE2", "LIFE3", "CUSTOM_LIFE", "LOGICAL",
     "RANDOM", "CUSTOM"};

```

Επιπλέον, η κλάση CARule ορίζει, πλέον των βοηθητικών μεθόδων και των μεθόδων υποστήριξης, τις εξής μεθόδους στις οποίες μπορείτε να επέμβετε :

- **public int executeAt(int, int)**

- **public int[] senseAt(int, int)**

Μέθοδοι αρχικοποίησης, επαναφοράς, εκτέλεσης και αντίληψης, αντίστοιχα. Η executeAt() επιστρέφει την νέα τιμή του κυττάρου στη δοσμένη θέση, ενώ η senseAt() επιστρέφει όλον τον πίνακα των τιμών των γειτόνων όπως αυτοί γίνονται αντιληπτοί (πριν την εφαρμογή τυχόν αντιληπτικού θορύβου).

- **public int execute(int[])**

```
public int execute(int[] aState)
{
    switch (function)
    {
        case LIFE:
        case LIFE2:
        case LIFE3:
            return defaultLife(aState);
        case CUSTOM_LIFE:
            return customLife(aState);
        case LOGICAL:
        case RANDOM:
        case CUSTOM:
            return match(aState);
    }
    return 0;
}

public int defaultLife(int[] aState)
{
    int myState[] = perceive(aState);
    int n = myState.length, curr = myState[n/2], sum = 0, val;
    for (int i=0;i<n;i++) sum += (myState[i] > 0) ? 1 : 0;
    if (curr == 1) // Alive
    {
        sum--; // Subtract the on value at current position
        val = ((sum == 2) || (sum == 3)) ? 1 : 0;
    }
    else // n == 0, unborn
        val = (sum == 3) ? 1 : 0;
    return implement(val);
}

public int customLife(int[] aState)
{
    return defaultLife(aState);
}
```

```

public int match(int[] aState)
{
    int n = computeNumberOfNeighbours();
    int myState[] = perceive(aState);

    int i = 0;
    while (i<numberOfLines)
    {
        int j = 0; boolean undecided = true;
        while ((j<n) && undecided)
            if ((myState[j] != lines[i][j]) && (lines[i][j] != ANY))
                undecided = false;
            else j++;
        if (undecided)
            return implement(live_value());
        // or, equivalently, for this rule, lines[i][25]
        i++;
    }
    return implement(0);
}

```

Σύμφωνα με τα προηγούμενα, εάν τροποποιήσετε την κλάση CARule, αυτό θα γίνει τροποποιώντας κατάλληλα την *setFunction(int)* και γράφοντας τη δική σας *customLife(int[])* ή/και τη δική σας *customMatch(int[])*.

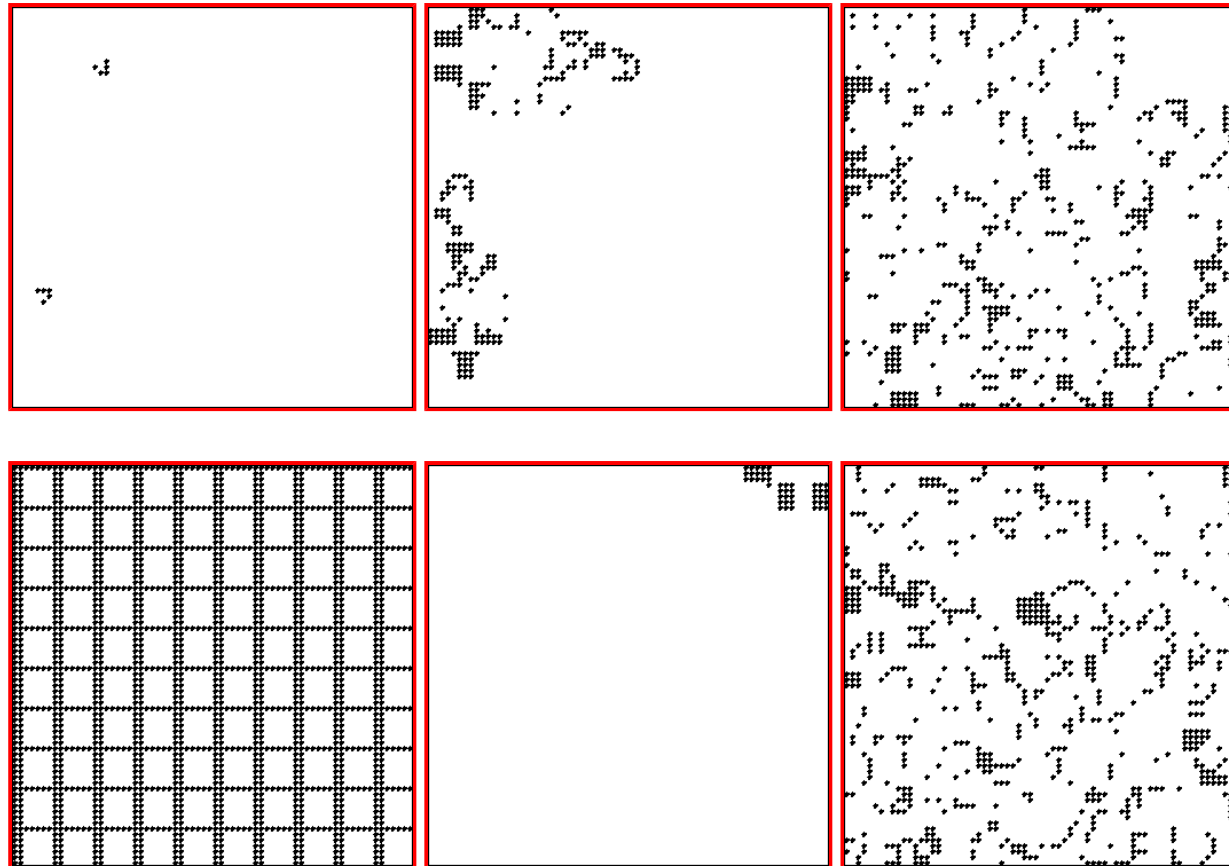
Παραδείγματα μελέτης μοντέλων

Τρία παραδείγματα κανόνων τύπου LIFE:

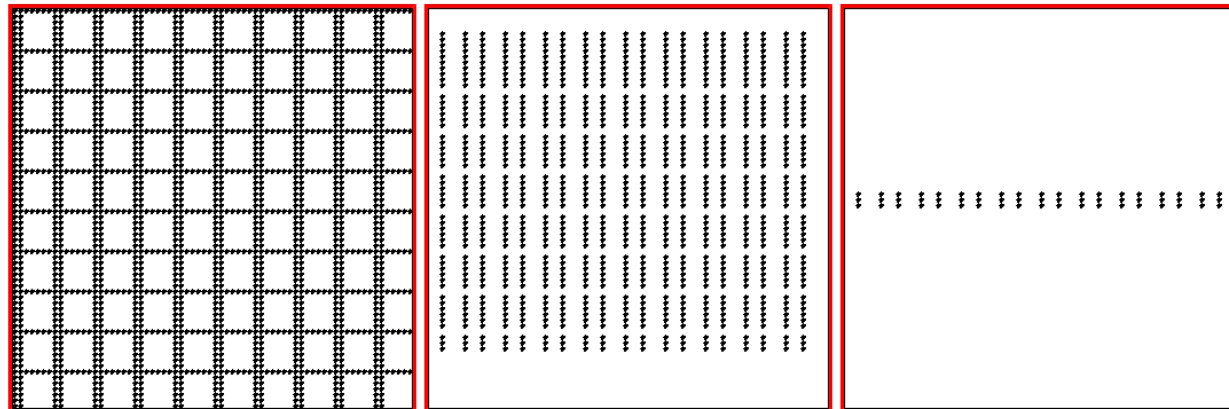
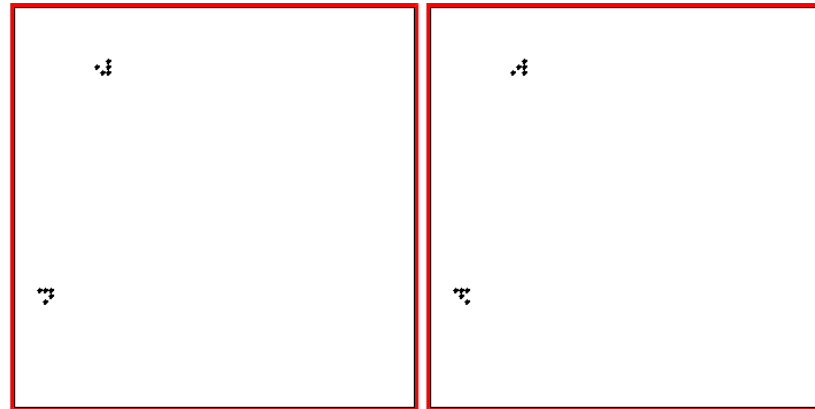
- **LIFE (original).** Βάθος = 1, diagonal connectivity = true, απουσία θορύβων.
- **LIFE 2.** Βάθος = 2, diagonal connectivity = true, απουσία θορύβων.
- **LIFE 3.** Βάθος = 2, diagonal connectivity = false, απουσία θορύβων.

Για το μοντέλο LIFE2, οι σχηματισμοί "chunks" και "columns" σβήνουν πολύ γρήγορα (σε 2 κύκλους), ενώ οι σχηματισμοί "2 gliders" και "grid" οδηγούν σε πολύπλοκη συμπεριφορά. Έτσι, οι τρεις πρώτες εικόνες παρακάτω δίνουν την κατάσταση του σχηματισμού "2 gliders" για t=0, t=10 και t=100, αντίστοιχα, ενώ οι τρεις επόμενες εικόνες δίνουν την κατάσταση του σχηματισμού "grid" για t=0, t=10 και t=100, αντίστοιχα.

Μπορούμε να παρατηρήσουμε ότι το σύστημα έχει την τάση να "εξατμίζεται" και να καταλήγει σε σχεδόν τυχαίες κατανομές με ορισμένα τοπικά chunks ή γραμμές ζωής.



Για το μοντέλο LIFE3, οι σχηματισμοί "chunks" και "columns" σβήνουν πολύ γρήγορα (σε 1 και 2 κύκλους, αντίστοιχα), ενώ οι σχηματισμοί "gliders" και "grid" οδηγούν σε περιοδική και στατική συμπεριφορά, αντίστοιχα. Έτσι, οι δύο πρώτες εικόνες παρακάτω δίνουν την κατάσταση του σχηματισμού "2 gliders" για $t=2k$ και $t=2k+1$, αντίστοιχα, ενώ οι τρεις επόμενες εικόνες δίνουν την κατάσταση του σχηματισμού "grid" για $t=0$, $t=10$ και $t=100$, αντίστοιχα. Μπορούμε να παρατηρήσουμε ότι το σύστημα έχει την τάση να καταλήγει σε στατικές ή περιοδικές κατανομές, αλλά χρήζει περαιτέρω διερεύνησης.

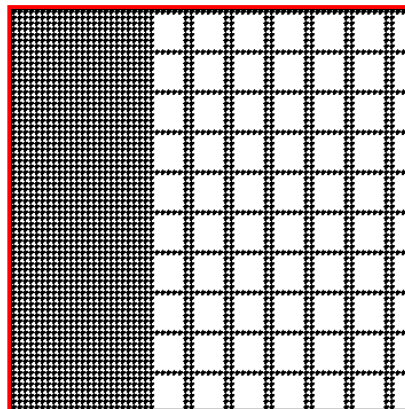


Ένα παράδειγμα λογικού κανόνα (υλοποιείται μέσα στο μοντέλο LOGICAL):

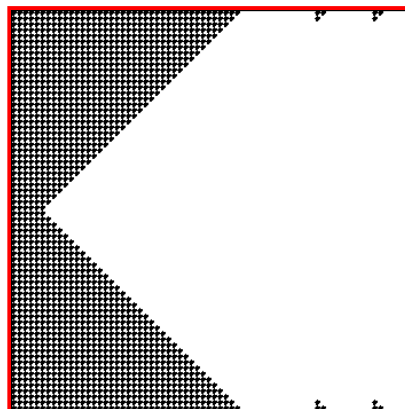
LOGICAL. Βάθος = 1, diagonal connectivity = false, απουσία θορύβων, 3 λογικοί κανόνες, οι εξής:

```
11***
01**1
01***
```

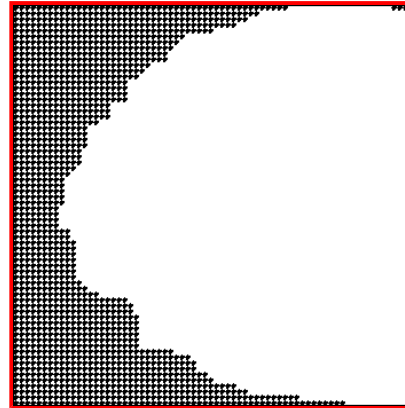
Για το μοντέλο αυτό, παρατηρούμε ότι κάθε σχηματισμός μετακινείται προς τα δεξιά και σβήνει όταν πέσει πάνω στον δεξί τοίχο (ξαναμπαίνει από τα αριστερά όμως, εάν το περιβάλλον είναι κυκλικό, δηλαδή toroidal !). Επίσης, εάν το περιβάλλον έχει "ζωντανά σύνορα", παρατηρούμε ένα συνεχές κύμα που μπαίνει από τα αριστερά και τελικά καλύπτει όλη την επιφάνεια της εικόνας (όπως στο παρακάτω σχήμα).



Εάν όμως αφαιρέσουμε τον τρίτο κανόνα, τότε όλοι οι σχηματισμοί σβήνουν σχετικά γρήγορα. Για περιβάλλον με ζωντανά σύνορα όμως προκύπτουν και πάλι κύματα που γεμίζουν την εικόνα, αλλά της παρακάτω μορφής:



Τα κύματα είναι πιο ρεαλιστικά (με πιο ακανόνιστο μέτωπο) όταν η ενεργοποίηση των κυττάρων είναι ασύγχρονη:



Συμπέρασμα

Κατ' αρχήν οι λογικοί κανόνες οδηγούν σε πολύ κανονικές (μη βιολογικές) μορφές, με μικρό βαθμό πολυπλοκότητας, όπως τον περιγράψαμε παραπάνω. Αυτό μπορεί να πιστοποιηθεί επίσης με δοκιμές του κανόνα "random" (τυχαίος λογικός κανόνας), που αρχικοποιείται με άλλους κανόνες κάθε φορά, και οδηγεί κατά κανόνα σε ταχύτατη νέκρωση του συστήματος. Τούτο είναι σε αντιδιαστολή με τα δυαδικά δίκτυα (boolean networks), όπου το σύνολο των τυχαίων λογικών κανόνων των κόμβων οδηγεί σε περιοδικές ή πιο πολύπλοκες συμπεριφορές και σπανιότατα σε νέκρωση του συστήματος. Μία αιτία είναι προφανώς ότι η οικογένεια των κυτταρικών αυτομάτων που μελετάμε εδώ είναι ομογενής ως προς το λογικό μοντέλο συμπεριφοράς (λόγω των υπολογιστικών απαιτήσεων του συστήματος), ενώ τα δυαδικά δίκτυα δεν είναι. Μη ομογενή κυτταρικά αυτόματα έχει μελετήσει ο Moshe Sipper (βλ. το άρθρο του "*Non-Uniform Cellular Automata: Evolution in Rule Space and Formation of Complex Structures*" παρακάτω).

Για τον πειραματισμό, προτείνεται λοιπόν η χρήση μεγάλου εύρους κανόνων με έξυπνη χρήση του '*' (τιμή 0 ή 1) σε επιλεγμένες θέσεις. Σαν άσκηση, δοκιμάστε να υλοποιήσετε τους κανόνες "life" και "life3" σε σύνολο λογικών κανόνων όπως παραπάνω. Δοκιμάστε στη συνέχεια την υλοποίηση λογικών κανόνων που δείχνουν προτίμηση στις συγκεντρώσεις σε κάποιες κατευθύνσεις μόνο κ.ο.κ.

Δείτε ακόμη ...

- **M.Sipper.** [Non-Uniform Cellular Automata: Evolution in Rule Space and Formation of Complex Structures](#), *Artificial Life IV, Proceedings of the 4th Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*, by R.Brooks & P.Maes (Eds.), MIT Press, 1994.
*Πρόσβαση μέσω του **secure server** του Μεταπτυχιακού Προγράμματος.*
- Stephen Wolfram's articles

- <http://www.stephenwolfram.com/publications/articles/ca/>
 - The Google Cellular Automata Page
http://directory.google.com/Top/Computers/Artificial_Life/Cellular_Automata/
 - Algorithmic Composition using Cellular Automata <http://www-106.ibm.com/developerworks/java/library/j-camusic/>
 - Paul Callahan's Game of Life Patterns Archive <http://www.radicaleye.com/lifepage/patterns/contents.html>
-

Τελευταία ενημέρωση 22 Ιουνίου 2004.

[Στείλτε μου mail \(brensham@softlab.ece.ntua.gr\)](mailto:brensham@softlab.ece.ntua.gr)