*14.06.2013*

**Paweł Troka, nr indeksu: 132334**

# Automaty komórkowe w modelowaniu i symulacji

Sprawozdanie

Sztuczna Inteligencja Projekt

1. Wstęp teoretyczny

Automaty komórkowe (ang. "cellular automaton", w liczbie mnogiej "cellular automata") zostały wynalezione w latach czterdziestych przez Stanisława Ulama. John von Neumann dostrzegł w nich uproszczony model fizyki rzeczywistego świata i poszukiwał z ich użyciem najprostszych struktur zdolnych do samopowielania się.

Jednym z głównych popularyzatorów tej dziedziny był na początku lat osiemdziesiątych Stephen Wolfram - twórca pakietu Mathematica. Wysunął on tezę, że w regułach automatu komórkowego mogą znaleźć odzwierciedlenie podstawowe prawa przyrody. Podjął też próbę ich systematycznej klasyfikacji.

Automaty komórkowe są zaliczane do szerokiej i modnej ostatnio dziedziny, jaką jest sztuczna inteligencja. W pewnym sensie są pokrewne do metody elementów skończonych i symulacji dynamiki płynów. Mają także związek z chaosem.

Z punktu widzenia programisty, na automat komórkowy składa się struktura danych i algorytm, który na niej operuje. Struktura ta ma postać tablicy komórek pewnego typu. Może to być tablica o dowolnej liczbie wymiarów - od jednowymiarowego wektora, poprzez macierz dwuwymiarową aż po tablice trój- i więcej wymiarowe. W przestawionych przykładach ograniczę się do dwóch wymiarów.

Jako podstawowe parametry opisujące automat komórkowy można wyróżnić:

1. Typ komórki. W najprostszym przypadku jest to pojedyncza wartość logiczna przyjmująca jeden z dwóch stanów - prawdy lub fałszu. Może to być wartość wyliczeniowa, liczba całkowita, zmiennoprzecinkowa, wektor czy nawet rozbudowana struktura złożona z wielu pól.
2. Stan początkowy. Może to być wypełnienie całej tablicy określoną wartością, wartościami losowymi, zapamiętanymi wcześniej w pliku czy wczytanymi z innego źródła (np. na podstawie obrazu graficznego) albo wygenerowanymi przez pewien algorytm.
3. Funkcja przejścia. Tak jak dyskretną przestrzeń w automatach komórkowych stanowi tablica komórek, tak dyskretny czas wyznaczają kolejne cykle przeliczania nazywane dalej iteracjami. Algorytm może w każdej iteracji, według pewnych określonych reguł przeliczać kolejno wszystkie komórki, tylko jedną (na pewnej pozycji pamiętanej między iteracjami jako bieżąca), jeden wiersz tablicy itp. Dla każdej przeliczanej komórki wyliczany jest jej nowy stan na podstawie pewnych parametrów, przede wszystkim jej stanu aktualnego i stanu komórek sąsiednich.

Automat określa się jako sprawiedliwy, kiedy jego działanie jest niezależne od kierunku, w jakim przeliczane są komórki w ramach każdej iteracji. Niezależność tą można uzyskać poprzez pamiętanie dwóch kopii tablicy. Odczyt odbywa się wtedy z jednej, a zapis do drugiej. Dzięki temu podczas przeliczania kolejnych komórek wartości sąsiednich odczytywane są w stanie sprzed modyfikacji.

Jako automaty komórkowe możnaby określać bardzo wiele stosowanych powszechnie algorytmów, np. wszelkie filtry graficzne. Jednak w praktyce określenia tego używa się raczej w pewnych konkretnych przypadkach i zastosowaniach.

Automaty komórkowe wydają się dziedziną mniej znaną, niż inne ciekawe grupy algorytmów (np. algorytmy fraktalne), choć mają szersze zastosowanie i w pewnym sensie są od nich prostsze.

1. Wybrane automaty komórkowe

W związku z ograniczony czasem postanowiłem zaimplementwować jedynie niektóre, wybrane przeze mnie automaty komórkowe. Wybór padł na najciekwsze i najpopularniejsze, zostały jednak zaimplementowane autorskie pomysły na automat komórkowy, takiej jak automat komórkowy do modelowania przepływu ciepła. Wymienię teraz i krótko opiszę zaimplementowane ak.

1. Gra w życie

Ten najbardziej znany z automatów komórkowych (ang. "Conway's Game of Life") działa wg następujących zasad: Komórki są pojedynczymi wartościami logicznymi. Każda z nich może być w stanie aktywnym (logiczna prawda, 1) lub nieaktywnym (logiczny fałsz, 0). W każdej iteracji przeliczane są w sposób sprawiedliwy wszystkie.

Na stan każdej komórki w chwili następnej (po danej iteracji) ma wpływ jej stan i stan komórek sąsiednich w danej chwili (przed tą iteracją). Sąsiedztwo Moore'a to 8 komórek sąsiadujących z daną pionowo, poziomo i na ukos (jak w systemowej grze Saper). Sąsiedztwo Neumanna natomiast to jedynie 4 komórki sąsiadujące z daną w pionie i w poziomie (powyżej, poniżej, na lewo i na prawo).

W grze w życie, dla każdej komórki sprawdzana jest liczba aktywnych komórek sąsiadujących z daną wg sąsiedztwa Moore'a. Może ich być od 0 do 8. Nowy stan komórki wyznacza następująca reguła:

Jeśli jest nieaktywna i dokładnie 3 sąsiednie są aktywne, staje się aktywna.

Jeśli jest aktywna i 2 lub 3 sąsiednie są aktywne, pozostaje aktywna.

W przeciwnym wypadku staje się nieaktywna.

Nazwa tej gry pochodzi od porównania tablicy do terenu, na którym każda komórka (organizm) może być żywa lub martwa. W przedstawionych regułach można doszukać się analogii do sytuacji, kiedy komórki rozmnażają się albo giną z braku towarzystwa bądź ze zbyt dużego zagęszczenia.

W oryginalnej, teoretycznej grze w życie plansza powinna być nieskończona. W implementacji komputerowej do częściowej realizacji tego wymagania można używać dynamicznych struktur danych, ale najczęściej wystarcza skończona tablica i założenie, że poza jej zakresem komórki są zawsze nieaktywne, tak właśnie uczyniłem w swojej implementacji.

Początkowo w grę tą grano bez użycia komputera - na kartce papieru, na tablicy, za pomocą żetonów itp. Dziś można ją znaleźć choćby jako dodatek do edytora Emacs. Ten znany automat komórkowy doczekał się też niezwykle dokładnych analiz, co zaowocowało m.in. zrealizowaniem za jego pomocą funkcji logicznych, a także sklasyfikowaniem i nazwaniem wielu układów komórek pozostających w stanie stabilnym, zmieniających się cyklicznie, a nawet przemieszczających się po planszy czy regularnie wyrzucających z siebie pewne układy. W roku 1982 udowodniono, że w życie jest automatem równoważnym uniwersalnej maszynie Turinga. Oznacza to, że jest zdolny do wykonania każdego algorytmu.

Usystematyzowanie możliwych efektów końcowych automatu (stanu po bardzo wielu czy nieskończenie wielu iteracjach) przysparza problemów. Ogólnie można jednak wyróżnić pewne sytuacje:

1. Stan stabilny, w którym nic się już nie zmienia.
2. Stan cyklicznie zmieniający się z pewnym niedużym okresem.
3. Stan chaotyczny, w którym trudno dopatrzeć się jakiegokolwiek uporządkowania.
4. Złożone, stabilne konfiguracje lokalne, często o długich czasach życia.

Najciekawsza jest oczywiście ta ostatnia grupa. Jak określił to Piotr Lasoń, "Prostota generuje nam urzekające piękno złożoności"

1. QuadLife

Tzw. Modyfikacja kolorystyczna Game of Life. Dla żywych komórek dostępne są aż cztery kolory – zwykle czerwony, żółty(u mnie czarny), zielony i niebieski. Dla martwych sytuacja się nie zmienia względem oryginału. Definiując warunki początkowe każdej z komórek przypisujemy jeden kolor. Na przynajmniej dwa spośród kolorów powinna być zabarwiona przynajmniej jedna komórka, przeciwnym razie uzyskamy zwykłą grę w życie. Nowo powstające komórki przyjmują taki kolor, jaki ma większość z ich 3 żywych sąsiadów; jeżeli ma po jednym sąsiedzie z każdego koloru, przyjmuje pozostały kolor. Kolory żywych komórek nie zmieniają się w trakcie gry.

1. Darwinia

Gra w życie pojawiła się jako jedno z intr do gry komputerowej Darwinia Każda komórka może pozostać żywa najwyżej przez 50 cykli, później umiera. Plansza ma skończony rozmiar, poza którym komórki nie mogą się rodzić. Poza tym stosowane są klasyczne zasady Conwaya 23/3.

Warunki początkowe zostały dobrane tak, że kończą się śmiercią wszystkich komórek. Istnieją jednak obiekty żyjące wiecznie, jak chociażby Krokodyl.

1. Immigration

Kolejna modyfikacja kolorystyczna Game of Life. Dla żywych komórek dostępne są dwa kolory – zwykle czerwony i żółty. Dla martwych sytuacja się nie zmienia względem oryginału. Definiując warunki początkowe każdej z komórek przypisujemy jeden kolor. Na każdy z kolorów powinna być zabarwiona przynajmniej jedna komórka, w przeciwnym razie uzyskamy zwykłą grę w życie. Nowo powstające komórki przyjmują taki kolor, jaki ma większość z ich 3 żywych sąsiadów.

Kolory żywych komórek nie zmieniają się w trakcie gry.

1. Modelowanie przepływu ciepła

Autorski automat komórkowy wymyślony przeze mnie w celu modelowania fizycznego zjawiska przepływu ciepła. Wymyślony i zaimplementowany bazując na prostych regułach. Istnieją komórki o wysokiej temperaturze (źródła ciepła) oraz komórki o niskiej temperaturze, pusta przestrzeń to komórki powietrza które nie jest najlepszym przewodnikime, istnieje jeszcze przewodnik (komórki o kolorze zielonym) i przegroda/izolator – komórki o kolorze czarnym. Komórki mogą tracić ciepło lub ogrzewać się w oparciu o przewodnik, który może się nagrzać. Powietrze mnie chętnie niż przewodnik odbiera ciepło a zimne i ciepłe komórki razem mogą wyrównać swoje temperatury i stać się zwykłym powietrzem. Powietrze natomiast może się ogrzać/ochłodzić, robi to mniej chętnie jednak niż przewodnik.

1. TrokaAutomaton

Prosty, autorski, automat komórkowy wymyślony w celu modelowania rozwoju komórek uwzględniając mutacje, choroby itd. Okazuje się, że stan końcowy zazwyczaj jest stanem okresowym, większość kolonii przeżywa.

1. **Wireworld**

Wireworld - dobrze znany automat komórkowy zaproponowany przez Briana Silvermana w roku 1987, jako część jego programu o nazwie Phantom Fish Tank. Stał on się popularny po publikacji poświęconego mu artykułu w dziale "Computer Recreations" czasopisma Scientific American. Wireworld jest często używany do symulacji elementów elektronicznych operujących na wartościach bitowych. Pomimo prostoty reguł, jakie nim rządzą, za pomocą Wireworld można stworzyć działający komputer. Komórka w Wireworld może znajdować się w jednym z czterech stanów:

* pusta;
* głowa elektronu;
* ogon elektronu;
* przewodnik.

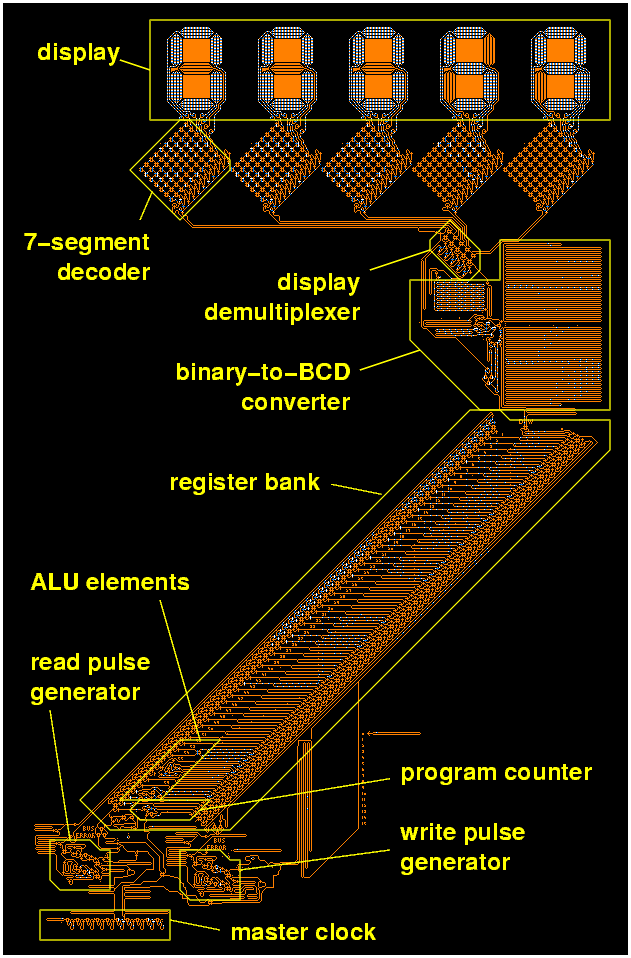
Warto zauważyć, że w oprogramowaniu obrazującym działanie Wireworld komórki są oznaczone zwykle 0-3, a nie 1-4. W przykładach znajdujących się na tej stronie, stany komórek są reprezentowane przez kolory odpowiednio: czarny, niebieski, czerwony i żółty. Nie ma jednak jakiejkolwiek powszechnej reguły na barwienie komórek.

Jak we wszystkich automatach komórkowych upływ czasu przedstawiony jest w postaci dyskretnych kroków czasowych, czyli generacji (w skrócie gens). Komórka pusta (w stanie zero) na zawsze pozostaje w swoim stanie; inne komórki zachowują się w następujący sposób:

* Głowa elektronu → ogon elektronu;
* Ogon elektronu → przewodnik;
* Przewodnik → głowa elektronu, ale tylko wtedy, gdy dokładnie 1 lub 2 komórki sąsiadujące są głowami elektronu.

Stosowane sąsiedztwo to sąsiedztwo Moore'a, czyli sąsiadem jest każda komórka, która graniczy z daną bokiem lub wierzchołkiem.

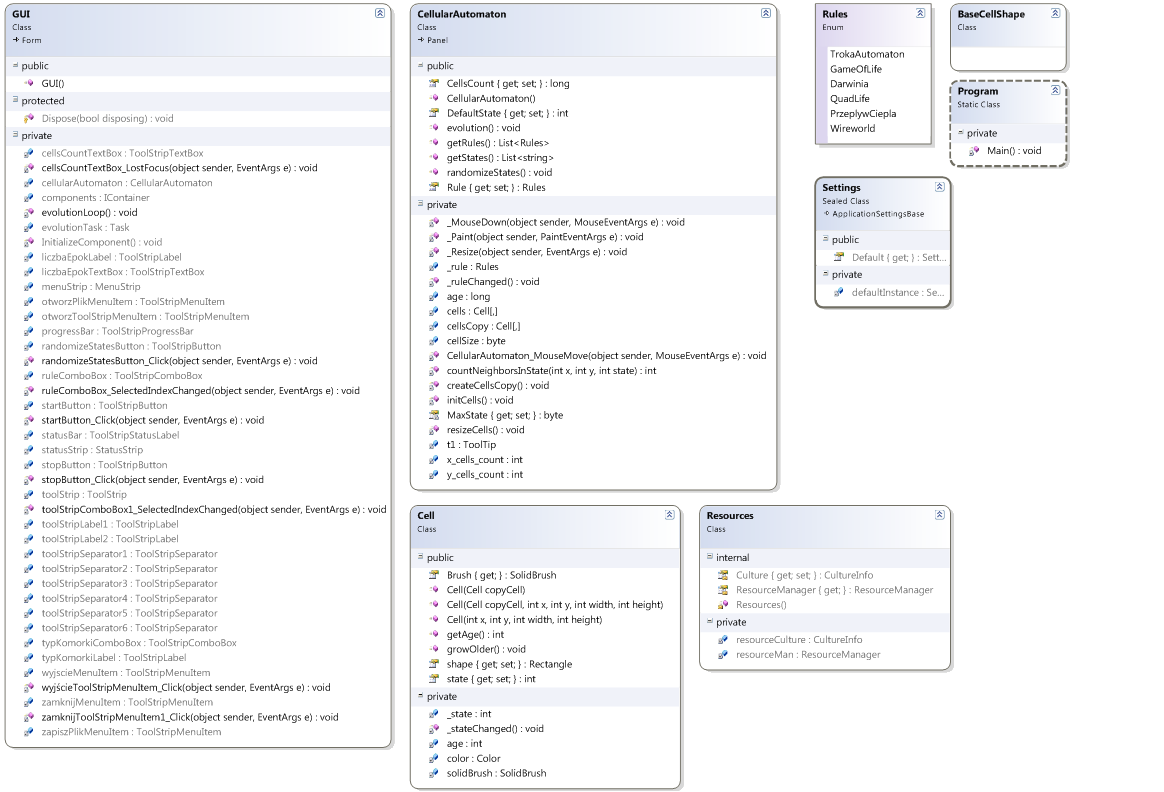
Stosowanie tych reguł umożliwia tworzenie struktur wykonujących operacje logiczne, tak prostych jak funkcja logiczna OR i tak złożonych jak komputer.



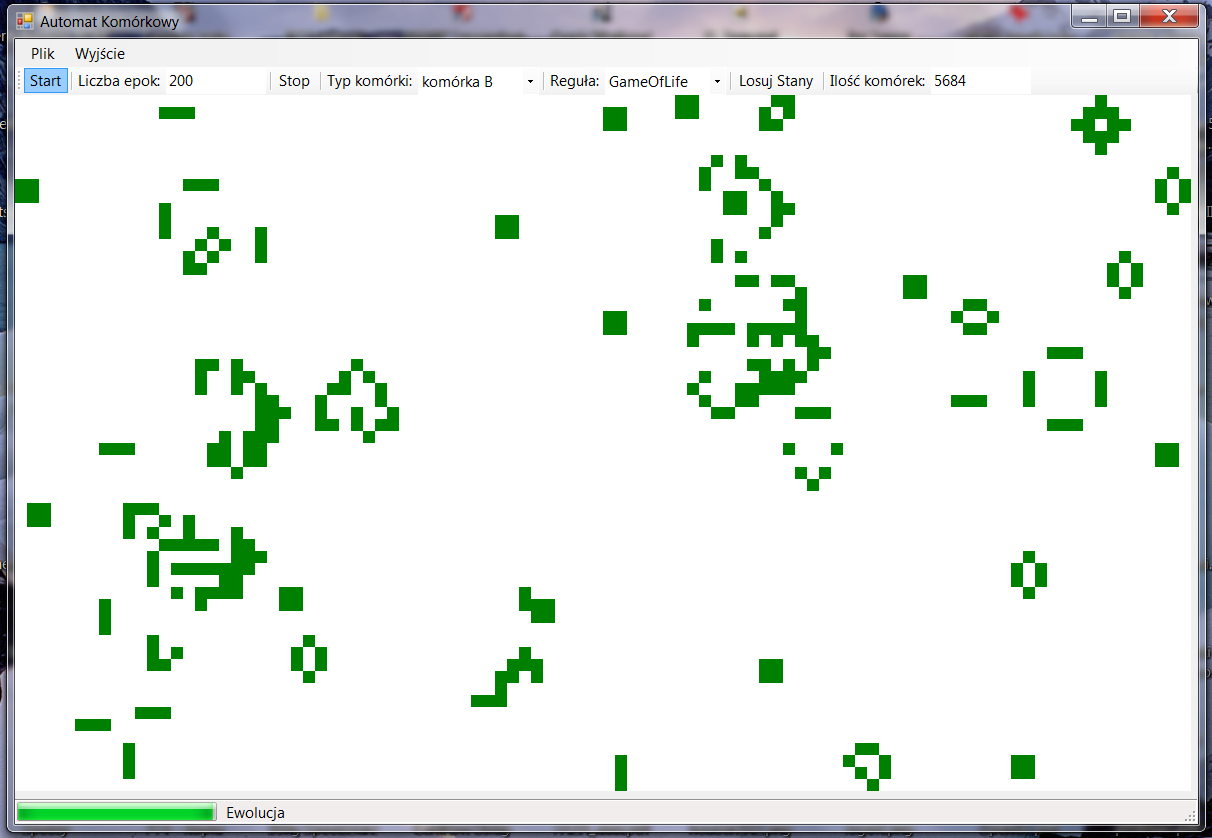
Rysunek - Komputer liczący liczby pierwsze zbudowany w oparciu o elementy automatu komórkowego Wireworld

1. Implementacja

Do implementacji wybrano język C# ze względu na prostotę i wygodę, środowisko Microsoft Visual Studio 2010 Ultimate ze względu na podobne pobudki oraz przyzwyczjenia. Cała aplikacja jest aplikacją z GUI opartą na WindowsForms. Sercem aplikacji jest graficzna reprezentacja automatu komórkowego w oparciu o klasę CellularAutomaton dziedziczącą po klasie Panel z namespace’a „System.Windows.Forms” pokrytą graficznymi reprezetacjami komórek automatu opartymi o klasę Cell, mającą m.in. pole z klasy Rectangle (System.Drawing), która jest „Sealed” (nie można było po niej dziedziczyć). Interfesj użytkownika umożliwia ustalania najróżniejszych parametrów takich jak ilość komórek, ilość epok ewolucji a nawet wybór konkretnego automatu komórkowego (reguła). Program zwiększa rozmiar komórek wraz z zwiększaniem jego rozmiaru przez użytkownik, pilnuje jednak aby komórki miały kształt kwadratowy, który zdaniem autora jest najlepszy do prezentacji danych. Liczba wspieranych komórek również jest na bardzo dobrym poziomie, z przyczyn technicznych minimalny rozmiar komórki to 1px x 1px, co daje możliwość posiadania nawet ponad miliona komórek w automacie przy współczesnych rozdzielczościach ekranu. Autor dołożył również wszelkich starań aby aplikacja działała wydajnie. Wzorce projektowe oraz dobre nawyki programowania obiektowego również grały istotną rolę w procesie programowania.



Rysunek - diagram klas użytych w programie



Rysunek - przykładowy widok aplikacji



Rysunek - bramka OR zrealizowana na automacie Wireworld

1. Podsumowanie i wnioski

Automaty komórkowe okazują się być niewyobrażalnie potężną dziedziną nauki, zarówno w symulacji zjawisk fizycznych jak i w modelowaniu różnych procesów. Niestety wydają się też one nieco niedocenione i niewykorzystane we współczesnym modelowaniu. Może o tym świadczyć sam fakt, że trudno znaleźć konkretne informacje o ich praktycznych, zastosowaniach. Większość informacji na temat ich zastosowań to jedynie „gdybania” teoretyczne bądź wymienianie *potencjalnych* (nie rzeczywistych) zastosowań[7]. Ich potencjał jednak jest niebagatelny, o czym może chociażby świadczyć fakt, że spora część z nich może wykonywać obliczenia i jest równoważna maszynie Turinga. Powstałe na komputerach automaty komórkowe, mogą być więc modelami komputera. Co więcej na automacie Wireworld zostało już tak śmiałem posunięcie zrealizowane[1].

1. Bibliografia
2. <http://en.wikipedia.org/wiki/Cellular_automaton> [dostęp: 13.06.2013]
3. <http://en.wikipedia.org/wiki/Conway's_Game_of_Life> [dostęp: 13.06.2013]
4. <http://en.wikipedia.org/wiki/Langton's_ant> [dostęp: 13.06.2013]
5. <http://en.wikipedia.org/wiki/Wireworld> [dostęp: 13.06.2013]
6. <http://www.quinapalus.com/wires11.html> [dostęp: 13.06.2013]
7. <http://en.wikipedia.org/wiki/Rule_30> [dostęp: 13.06.2013]
8. <http://warsztat.gd/wiki/Automaty+komórkowe+i+ich+zastosowania> [dostęp: 13.06.2013]