AKADEMIA GÓRNICZO – HUTNICZA
KRAKÓW

Tomasz Kąkol

Automaty komórkowe

Projekt 1

Data złożenia sprawozdania:
02.06.2018

Ocena:

Aplikacja

Aplikacja jest dostępna pod adresem:

http://orion.fis.agh.edu.pl/~4kakol/

Wstęp

Automat komórkowy jest systemem zbudowanym z pojedynczych komórek tworzących większe zbiory dzięki obecności w swoim sąsiedztwie. Każdy AK jest obiektem dynamicznym w czasie. Każda komórka może przyjąć jeden z możliwych stanów, ale liczba stanów jest skończona. Zmiany stanów komórki następują w sposób zsynchronizowany, odpowiedni do wartości wynikających z nałożonych reguł. Nowe wartości stanu w większości przypadków zależą od obecnego stanu komórki oraz stanu komórek znajdujących się w otoczeniu tej komórki.

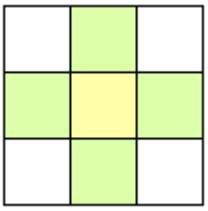
Automaty komórkowe posiadają przestrzeń, w której następuje obserwowana ewolucja. Przestrzeń budują identyczne komórki, w których każda przechowuje wartość swojego stanu. Istnieje jednoznaczność odnośnie położenia komórki w siatce (jesteśmy w stanie określić położenie każdej komórki w siatce). Czynnikami wpływającymi na strukturę siatki komórek są :

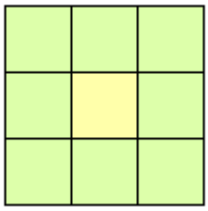
- wymiar przestrzeni
- warunek regularności
- liczba sąsiadów

Wymiar przestrzeni jest zależny od rozwiązywanego problemu i może rozpoczynać się od wymiaru ‘1D’, a może kończyć się na wymiarze ‘nD’. Warunek regularności dotyczy informacji o (ilościowym) zapełnieniu całej siatki przez jednakowe komórki (tj. w 2D komórki trójkątne, kwadratowe, a w 3D np. sześciany). Liczba sąsiadów jest zależna od wartości 2 poprzednich czynników.

Sąsiedztwo komórek

Opiszę krótko zagadnienie sąsiedztwa komórek, a mianowicie podstawowych typów, czyli sąsiedztwo von Neumanna i sąsiedztwo Moore'a.





Rysunek 1 Sąsiedztwo von Neumanna Rysunek 2 Sąsiedztwo Moor'a

W przypadku sąsiedztwa von Neumanna, gdy zaznaczymy kierunki róży wiatrów N, S, E, W oraz kierunku składowe, tj. NW, NE, SE, SW, to w efekcie sąsiedztwem von Neumana jest wymieniony pierwszy zbiór komórek. Z kolei w przypadku sąsiedztwa Moore'a, zbiorem tym są oba wymienione zbiory, czyli łącznie osiem komórek wokół komórki w centrum.

Warunki początkowe

Dotyczą stanu poszczególnych komórek na starcie, tj. zerowej iteracji. Ze sposobu początkowego ustawienia komórek wynika przebieg ewolucji automatu, zachowanie, a także stan końcowy – czyli powodzenie lub nie całej symulacji.

Warunki brzegowe

Pokrótce opiszę zagadnienia związane z warunkami brzegowymi. Istnieje kilka typów takich warunków, a mianowicie: periodyczne, zamknięte pochłaniające, zamknięte odbijające. Periodyczne (inaczej przenikające) definiują zamkniętą siatkę tak, że stan komórki poruszający się do krawędzi, po dotarciu pojawi się z drugiej strony. Warunek zamknięty pochłaniający dotyczy siatki zdefiniowanej tak, że brzegi tej siatki wypełnione są z góry ustaloną wartością. Z użyciem funkcji przejścia ustalany jest wpływ na zachowanie automatu. (praktyczny przykład: symulując cząstkę gazu, po przekroczeniu krawędzi siatki cząstka ta przestane istnieć) . Ten warunek brzegowy stosuje się w tych automatach, gdzie generowane są automatycznie komórki. W ten sposób redukowane są komórki, co zapobiega zagęszczeniu liczby komórek w automacie. Warunek zamknięte odbijające dotyczy tworzenia na krawędzi siatki bariery, od której symulowane cząstki są odbijane. Ten warunek brzegowy stosuje się w symulacjach zamkniętych przestrzeni doświadczalnych.

Reguły przejść

Określają ewolucję AK w dyskretnym czasie. W każdej kolejnej iteracji stany poszczególnych komórek są aktualizowane. Stan komórki 'x' w chwili 't' oznaczamy jako 'x_t', natomiast stan sąsiedztwa jako '$u(x_t)$' . Wobec takich założeń stan komórki 'x' w następnej iteracji opisujemy jako: $x_{t+1} = f(u(x_t), x_t)$, gdzie 'f' jest funkcją przejścia. Może być ona opisana różnego rodzaju zależnościami (np. tabela przejść). Reguły przejść muszą być zdefiniowane obok przestrzeni stanów i zdefiniowanego sąsiedztwa (inaczej nie będzie możliwa ewolucja automatu).

Ewolucja AK

W pierwszej kolejności należało ustalić i zdefiniować wszystkie elementy Składowe AK. Dopiero następnie nakładamy regułę na siatkę (czyli aktualizujemy ją).

Proces ewolucji automatu:

1. Stan początkowy – ustalenie warunków początkowych
2. aktualizacja siatki automatu - w każdej iteracji każda komórka automatu jest poddana sekwencji:
   2.1. Sprawdzenie reguł przejść – sprawdzenie stanu komórki, komórek sąsiednich, i pozostałe
   2.2. Sprawdzenie sąsiedztwa – sprawdzenie, usunięcie stanów konfliktów
   2.3. Sprawdzenie warunków brzegowych – sprawdzenie komórek na brzegu siatki (usuwane – sąsiedztwo zamknięte pochłaniające, tworzone – sąsiedztwo periodyczne)
   2.4. Sprawdzenie ilości iteracji – gdy jest określona wartość epoki (cyklu życiowego automatu), czasami jest tutaj sprawdzane czy automat nie osiągnął już stanu stabilnego
   2.5. Zwiększenie licznika iteracji i powrót do sprawdzenia reguł przejść (2.1)

Podział Wolframa

Wyróżniane są 4 klasy automatów w podziale Wolframa, a mianowicie:

- Klasa I – niezmienne (zbieżne) , ewolucja do momentu osiągnięcia identycznego stanu komórek(niezależnie od warunków początkowych)
- Klasa II – ewoluujące do stanu stabilnego lub okresowe
- Klasa III – wykazujące nieporządek lokalnie oraz globalnie, chaotyczne (brak wzorca)
- Klasa IV - bardziej złożone, długotrwałe zachowanie (automaty „żywe”)

Elementarne AK

Celem wykonanego projektu było przedstawienie działania elementarnych AK, tj. zaimplementowanie obsługi elementarnych AK , należących do podstawowych i najlepiej zbadanej rodziny automatów, tj. jednowymiarowych automatów deterministycznych cechujących dwa stany komórki (k=2) wraz z otoczeniem komórki zawierającym wyłącznie najbliższych sąsiadów (r=1). Istnieje możliwość określenia 256 elementarnych AK. Właściwość ta wynika z wartości argumentu funkcji F, określającej stan komórki w kolejnym kroku czasowym (t+1) w zależności od wartości w aktualnym kroku czasowym (t) tej komórki oraz komórek otaczających. Dla elementarnych AK wartości 3 (2r + 1=3) jest argumentem reguły F, gdzie parametr ‘r’ to wspomniane otoczenie najbliższych sąsiadów. Rezultatem podniesienia ilości możliwych stanów komórki do potęgi wartości argumentu reguły F skutkuje otrzymaniem 256 możliwych określeń.

W zaimplementowanej symulacji celowo została zablokowana możliwość ustawienia numeru automatu powyżej 255 (0-255). Dzięki temu rozwiązaniu zapobiegnięto skutkom wprowadzenia błędnych wartości.

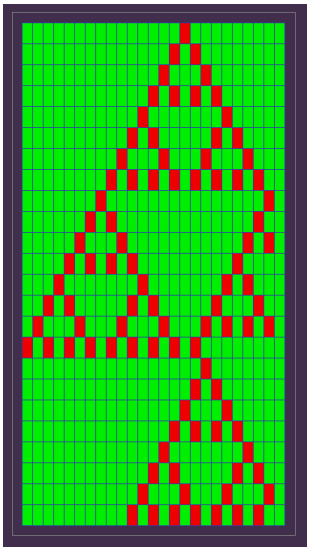
W symulacji zastosowałem następujący elementarny AK (w skutek określenia kolejności wartości funkcji F stanów otoczenia):

111	110	101	100	011	010	001	000

Zdefiniowanie reguły automatu polega na wyznaczeniu ośmiu stanów ‘0’ lub ‘1’ (‘centralne’ wartości z grup komórek) i scaleniu tego ciągu stanów. Popularnym elementarnym AK jest automat 90 (notacja dziesiętna), który jest równoważny funkcji XOR na stanach sąsiadów komórki centralnej. Poniżej zobrazowano wynik symulacji uzyskany dla tego automatu, którym jest fraktal – dywan Sierpińskiego. Automat 90 posiada regułę typu (pogrubiona czcionka):

111	110	101	100	011	010	001	000
1	**1**	**0**	**1**	**1**	**0**	**1**	**0**

W celu odtworzenia symulacji – to jest określenia stanu komórki w chwili t+1 w zależności od stanów otoczenia komórki w chwili t - należy wpisać wartość 90 w polu do tego przeznaczonym. Domyślnie typ automatu jest ustawiony na numer.



Rysunek 3 Fraktal: dywan Sierpińskiego

W symulacji wprowadzono losowość, tj. uzależnienie funkcji F od zmiennej losowej. O takim automacie mówimy, że jest probabilistycznym automatem komórkowym.

Legalne AK

W projekcie poświęcono uwagę legalnym AK, należących do rodziny elementarnych AK, jednakże muszących spełniać dodatkowe wymagania. Pierwszym wymaganiem jest wykazanie przez automat symetrię, tj. F(001) = F(100) oraz F(110) = F(011). Drugim wymaganiem jest 'zachowanie stanu próżni' (skutek: brak niepożądanych oscylacji), tj. spełnienie warunek u '000' -> '0' .

111	110	101	100	011	010	001	000
A	**β**	**γ**	**δ**	**β**	**ε**	**δ**	**0**

Na podstawie przedstawionych warunków, istnieją 32 ($2^{1+1+1+1+1+1} = 2^6$) legalne AK spełniających regułę typu przedstawionej powyżej.

Definicja została zaimplementowana jako funkcja 'actionLCA' w pliku *.js. Zgodnie z regułą, odpowiednie komórki zmieniają swoje stany w zależności od zmiany w otoczeniu. W widoku aplikacji – kliknięcie w odpowiedni przycisk stanu komórki skutkuje zmianą stanów odpowiednich komórek w otoczeniu. Przykłady wizualizacji zależności w widoku aplikacji:

- Zależność **β:** 

- Zależność **δ:** 

Głosujące AK

Kolejnym zbiorem należącym do rodziny elementarnych AK, podobnie jak Legalne AK, jest zbiór głosujących (liczących) AK. Te automaty muszą spełniać jeszcze bardziej rygorystyczne warunki. Ograniczenie reguł do głosujących AK (istotna tylko '1' w otoczeniu) jest następujące:

111	110	101	100	011	010	001	000
A	**β**	**β**	**γ**	**β**	**γ**	**γ**	**δ**

To oznacza, że kombinacja 110 musi dać taki sam wynik jak kombinacja 101 oraz 011, a kombinacja 100 taki sam rezultat jak kombinacja 010 oraz 001. Liczebność automatów spełniających powyżej przedstawioną regułę jest szesnaście ($2^{1+1+1+1}$). Poniżej przestawiłem przykłady wizualizacji zależności w widoku aplikacji:

- Zależność **β:** 

- Zależność **γ:** 

Wybór reguły automatu należy wykonać z użyciem menu rozwijanego stworzonego w tym celu. Symulacja automatu komórkowego przedstawiona jest z zastosowaniem 3 tablic, oraz wykresu typu linear-chart. Wartości komórek dla chwili t=0 dobierane są losowo. Są one

wyświetlane w pierwszym wierszu tabeli (nagłówek „Losowe wartości"). Następnie, dla każdego następnego kroku stany są iterowane, skutkiem jest przesuwanie wierszy tabel 'w dół'. Na podstawie wartości komórek w tabeli nazwanej „Losowe wartości" i „Jedno zaburzenie" wyznaczane są wartości w tabeli „Różnica wartości". Na podstawie tej tabeli oceniamy aktualny stan automatu. W 2-giej tabeli („Jedno zaburzenie") wprowadzono różnicą w porównaniu z tabelą nr.1. W chwili t=0 pojedyncza losowa komórka przyjmuje odwrotną wartość niż odpowiadająca jej komórka w tabeli 1. Tabela 3 ilustruje tą różnicę – czerwone kafelki oznaczają różnicę, a zielone zgodność pomiędzy stanami odpowiednich komórek tabel 1 i 2. Na wykresie linear-chart również przedstawiam różnice stanów obu automatów, poprzez wyświetlanie na wykresie odległości Hamminga. Ograniczyłem wartość rozmiaru szerokości automatu na 25 komórek ze względów wizualizacyjnych, aby widok aplikacji był umiarkowanie przyjazny dla użytkownika. Wzór na odległość Hamminga jest następujący:

$$d(\sigma_1, \sigma_2) = \sum_{i=1}^{N} |s_i(\sigma_1) - s_i(\sigma_2)|$$

Równanie 1 Równanie Hamminga

- si ($\sigma 1$) - wartość i-tej komórki pierwszego automatu
- $si(\sigma 2)$ jest wartością i-tej komórki drugiego automatu
- $d(\sigma 1, \sigma 2)$ liczba komórek, dla których stan w obu automatach jest różny

Panel sterowania zawiera 3 przycisk umożliwiające kontrolowanie symulacji:

- Start – uruchomienie symulacji
- Stop – wstrzymanie symulacji
- Restart – restartowanie symulacji, po zatrzymaniu i zrestartowaniu istnieje możliwość nowego skonfigurowania reguł automatu

Wartość komórki ustalana jest na podstawie jej stanu oraz stanu jej sąsiadów. Istnieje problem stanów komórek na granicy siatki, posiadających jednego sąsiada. Tutaj w zależności od dobru warunków brzegowych, można potraktować brzegową kolumnę z prawej strony jako lewostronnego sąsiada kolumny na lewym brzegu (i analogicznie w drugą stronę).

Zastosowany komponent linear-chart jest gotowym rozwiązaniem deweloperskim, szczegóły adresu do tego komponentu znajdują się w pliku źródłowym ca.js. Również usunięto ręcznie polskie znaki tekstowe, mimo zastosowania odpowiednich czcionek, po wrzuceniu paczki plików na środowisko Orion istnieje problem z obsługą 'polskich znaków', którego jeszcze nie rozwiązałem.