Matematyka z *Mathematicą*: automaty komórkowe

Galina FILIPUK[*], *Andrzej KOZŁOWSKI*[**]

Systemy algebry komputerowej. We współczesnym świecie komputery mają coraz większy wpływ zarówno na sposób uprawiania matematyki, jak i na samą matematykę. Wielu ważnych problemów, zarówno czysto teoretycznych, jak i pochodzących z zastosowań, nie można rozwiązać bez pomocy odpowiednich programów. Umiejętność posługiwania się komputerem w obliczeniach numerycznych i symbolicznych należy dziś do podstawowego wykształcenia matematyka.

Zwykle najbardziej efektywne jest użycie programów zaprojektowanych specjalnie do takich obliczeń, często nazywanych CAS (*Computer Algebra System* – system algebry komputerowej). Są to np. Macsyma, Maple, Maxima, MuPad, MatLab, Reduce, Axiom, Magma, Macaulay, Singular, i wiele innych – wśród nich znajdują się zarówno programy komercyjne, jak i darmowe oraz *open source*. Te programy mają zaimplementowane najnowsze algorytmy z różnych dziedzin matematyki, a także udostępniają języki programowania pozwalające użytkownikowi na pisanie własnych programów. Odgrywają wielką rolę w powstawaniu nowego działu nauki zwanego *matematyką eksperymentalną*.

Można wyróżnić dwa typy systemów algebry komputerowej: programy wyspecjalizowane w szczególnych działach matematyki (np. algebra przemienna, teoria grup, numeryczna algebra liniowa, równania różniczkowe, statystyka itd.), oraz programy ogólne, służące „do wszystkiego”. Chcemy tu przedstawić program *Mathematica* [1], należący do tej drugiej grupy. Jest on bardzo wszechstronny, a jednocześnie różne jego funkcje dobrze współdziałają. Spróbujemy pokazać zakres możliwości zastosowania *Mathematiki* na przykładzie automatów komórkowych. Jest to zagadnienie, w którym bez programu komputerowego byłoby bardzo trudno otrzymać ciekawe wyniki. Ponadto nie tylko leży w obrębie matematyki eksperymentalnej, ale jest także blisko związane z historią *Mathematiki* oraz z osobą jej pomysłodawcy. *Mathematica* jest, oczywiście, dziełem wielu osób – nikt nie byłby w stanie napisać samodzielnie tak dużego, a przede wszystkim wszechstronnego programu. Jednak główna idea i istotna część implementacji są dziełem jednej osoby: fizyka Stephena Wolframa.

Wolfram i jego dzieło. Wolfram, urodzony w 1959 r., jest niezwykłą i wybitną, choć także kontrowersyjną, postacią. Swój pierwszy artykuł opublikował, mając 18 lat, jako student fizyki na uniwersytecie w Oxfordzie (gdzie został przyjęty w wieku 15 lat). Opuścił uniwersytet, formalnie nie ukończywszy studiów, jako autor dziewięciu artykułów. Uzyskał stopień doktora na Uniwersytecie Caltech w Los Angeles, gdzie został zatrudniony. Był jedną z pierwszych osób, której przyznano słynny „grant dla geniuszy” (MacArthur Fellowship). Będąc w Caltech, Wolfram współtworzył system algebry komputerowej SMP, który stał się później podstawą *Mathematiki*.

W 1983 r. w Institute for Advanced Study w Princeton Wolfram rozpoczął badania automatów komórkowych, które stały się głównym obiektem jego zainteresowań naukowych. Podstawową metodę stanowiły symulacje komputerowe, a pierwsze wersje *Mathematiki* były ich głównym narzędziem. W 1987 r. Wolfram stworzył firmę Wolfram Research, zajmującą się głównie rozwojem i marketingiem *Mathematiki* i innych powiązanych z nią programów. Rok później opuścił świat akademicki, poświęcając się całkowicie prowadzeniu firmy oraz badaniom dotyczącym automatów komórkowych. W dalszej części tekstu przyjrzymy się obiektom tych badań i możliwościom, które przy pracy nad tym zagadnieniem daje *Mathematica*.

Automaty komórkowe. Pojęcie automatu komórkowego zostało wprowadzone przez Johna von Neumanna i Stanisława Ulama w celu modelowania samoreprodukcji wyidealizowanych systemów biologicznych. Nadaje się ono jednak równie dobrze do modelowania dyskretnych systemów fizycznych lub chemicznych. Takie systemy można uważać za przybliżenia układów dynamicznych, czyli systemów modelowanych za pomocą równań różniczkowych.

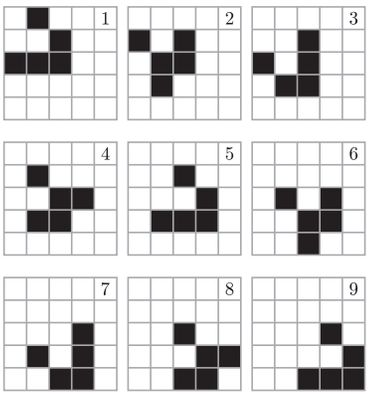
Rozwiązanie zadania F 842.
Podczas lotu pocisku z dużą prędkością siła oporu jest proporcjonalna do kwadratu prędkości pocisku. Z dobrym przybliżeniem można też założyć, że pocisk porusza się po linii prostej. Odpowiedź na pytanie postawione w zadaniu wymaga zbadania, jak zmienia się energia pocisku z przebytą drogą. Działająca na pocisk siła to $F = kv^2$, gdzie v oznacza prędkość chwilową, a k to stała zależna od kształtu pocisku; zwrot tej siły jest przeciwny do zwrotu prędkości. Ponieważ dla ruchu jednowymiarowego siła jest równa zmianie energii $\mathrm{d}E$ na jednostkę długości toru $\mathrm{d}s$ (w tym przypadku mamy do czynienia jedynie z energią kinetyczną $E = \frac{1}{2}mv^2$), możemy napisać równanie ruchu w postaci

$$\frac{\mathrm{d}E}{\mathrm{d}s} = -\frac{2kE}{m}.$$

Przy tej samej energii początkowej i przebytej drodze pocisk cięższy ma więc większą energię.

W 2002 r. ukazała się książka *A New Kind of Science*, oparta na wynikach wieloletniej pracy Wolframa – niewątpliwie najobszerniejsza i najbardziej znacząca praca poświęcona automatom komórkowym. Jak inne dzieła tego autora, książka przyciągnęła wielką uwagę nie tylko środowisk naukowych, ale także mediów, i wywołała kontrowersje. Jej główne idee mają pewien związek z *Mathematicą*.

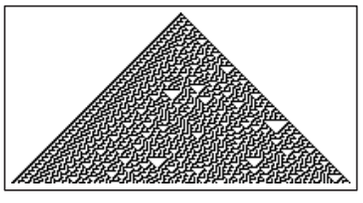
[*]Instytut Matematyki,
[**]Instytut Matematyki Stosowanej i Mechaniki,
Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski



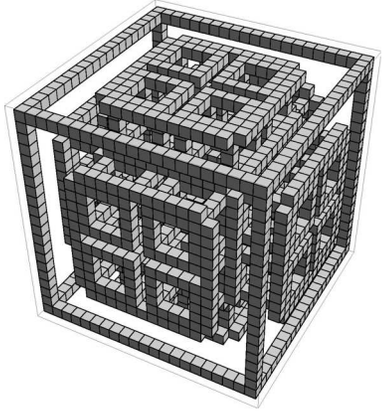
Rys. 1. Dziewięć generacji dwuwymiarowego automatu komórkowego *Gra w życie*.

Ponieważ Wolfram stworzył *Mathematicę* w dużym stopniu w celu ułatwienia badań automatów komórkowych, jest to jedno z najwygodniejszych narzędzi do tego celu. Badanie automatów komórkowych to jedna z dziedzin, w których bez specjalnego przygotowania programistycznego i głębokiej znajomości matematycznej strony tematu, można (za pomocą *Mathematiki*, ale nie tylko) dokonywać oryginalnych odkryć.

Oczywiście, w kodzie obok jedynie funkcja `CellularAutomaton` i jej argumenty mają bezpośredni związek z automatami komórkowymi – reszta kodu dotyczy tworzenia grafiki.



Rys. 2



Rys. 3. Etap ewolucji trójwymiarowego automatu nr 14.

Automat komórkowy składa się z dyskretnej siatki komórek w n-wymiarowej przestrzeni. Każda komórka może przyjmować skończoną liczbę stanów. System ewoluuje z pewnej konfiguracji początkowej, czas jest liczony dyskretnie. Nowy stan każdej komórki zależy od jej otoczenia w poprzedniej chwili. Jest kilka naturalnych definicji otoczenia, ale najczęściej przyjmuje się, że składa się ono z rozważanej komórki i wszystkich z nią sąsiadujących. Stan wszystkich komórek zmienia się równocześnie, tworząc nową generację.

Najbardziej popularny przykład, dwuwymiarowy, to wymyślona przez Johna Conwaya *Gra w życie*. W grze Conwaya komórki przyjmują dwa stany: są żywe lub martwe. Martwa komórka, która ma dokładnie trzech żywych sąsiadów, staje się żywa w następnej jednostce czasu (rodzi się), żywa komórka z dwoma albo trzema żywymi sąsiadami pozostaje nadal żywa, natomiast przy innej liczbie sąsiadów umiera (z samotności albo zatłoczenia).

Automaty w *Mathematice*. Wolfram przywiązuje wielką wagę do badań złożonych systemów za pomocą automatów komórkowych, określił to nawet jako „nowy rodzaj nauki” – część wspomnianej wcześniej matematyki eksperymentalnej. Zatem trudno się dziwić, że w *Mathematice* istnieje funkcja CellularAutomaton. Na przykład kod tworzący rysunku 1 wygląda tak:

```
GraphicsGrid[{ArrayPlot[#, ImageSize -> 80, Mesh -> True] &
  /@CellularAutomaton[{224, {2, {{2, 2, 2}, {2, 1, 2}, {2, 2, 2}}},
  {1, 1}}, {{{0, 1, 0}, {0, 0, 1}, {1, 1, 1}}, 0}, 8]}]
```

By wyjaśnić działanie tej funkcji (a także uzyskać pewne pojęcie o języku *Mathematiki*), posłużymy się bardzo prostym przykładem automatu: jednowymiarowym z tylko dwoma stanami, które będziemy oznaczali cyframi 1 i 0. Za otoczenie danej komórki będziemy przyjmowali tę komórkę wraz z dwiema sąsiednimi. Takie automaty Wolfram nazywa *elementarnymi*. Reguła ewolucji takiego automatu może być zapisana w języku *Mathematiki* na wiele różnych sposobów. Najczęściej używana jest metoda Wolframa, która przedstawia regułę ewolucji w postaci liczby.

W przypadku elementarnego automatu musimy zapisać, jak w kolejnym kroku ewolucji zmienia się środkowa cyfra w każdej z 2^3 możliwych trójek (i, j, k) złożonych z zer i jedynek. To jest równoważne podaniu jednej liczby ośmiocyfrowej zapisanej w systemie dwójkowym (lub jej dziesiętnego równoważnika). Kolejne cyfry (dwójkowe) tej liczby mówią, na co zmieni się środkowy element w kolejnych trójkach ustawionych w ciąg malejący liczb w systemie dwójkowym. Na przykład reguła 30, czyli binarnie 00011110, oznacza automat, w którym

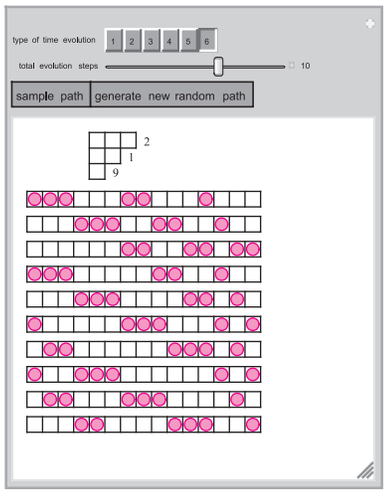
$$\{1,1,1\}\mapsto 0,\quad \{1,1,0\}\mapsto 0,\quad \{1,0,1\}\mapsto 0,\quad \{1,0,0\}\mapsto 1,$$
$$\{0,1,1\}\mapsto 1,\quad \{0,1,0\}\mapsto 1,\quad \{0,0,1\}\mapsto 1,\quad \{0,0,0\}\mapsto 0.$$

Dla automatów w wyższych wymiarach można zastosować podobny zapis. W najprostszej formie funkcja `CellularAutomaton` ma trzy argumenty: liczbę oznaczającą regułę (w przykładzie z rysunku 1 jest to 224), stan początkowy i liczbę kroków ewolucji.

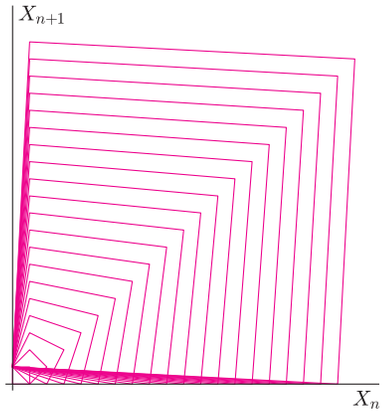
Automat – generator liczb pseudolosowych. Rysunek 2 prezentuje początkowe sto generacji rozwoju automatu komórkowego opisywanego przez regułę 30. Na tym automacie oparty jest jeden z generatorów liczb pseudolosowych używanych przez *Mathematicę*. Ewolucję tego automatu można też opisać za pomocą formuły rekurencyjnej

$$b_i = a_{i-1} + a_{i+1} + a_i + a_i a_{i+1} \bmod 2,$$

gdzie a_i oznacza 0 (biały piksel) lub 1 (czarny piksel) w kolumnie i pewnego wiersza a b_j w kolumnie j następnego. Stan początkowy określony jest w kodzie przez $\{1,\{0\}\}$, co oznacza jedynkę otoczoną nieskończonymi ciągami zer (jak widać w pierwszym wierszu). Okazuje się, że wartości w środkowej kolumnie zachowują się losowo. Otrzymujemy w ten sposób generator liczb pseudolosowych, który bardzo dobrze się sprawdza, mimo że oparty jest na prostej i oczywiście deterministycznej formule. Jak widać, nie tylko rozwój *Mathematiki* służy badaniu automatów komórkowych, ale zachodzi również zależność odwrotna.



Rys. 4. Przykład solitonowego automatu komórkowego (Reiko Sakamoto, źródło: strona internetowa Wolfram Demonstrations Project, `http://demonstrations.wolfram.com/PeriodicBoxBallSystem`).



Rys. 5. Rozwiązania równania (2) dla $K = 1, \ldots, 20$.

Oczywiście, ten automat można dosyć łatwo zaprojektować i przedstawić graficznie właściwie w dowolnym języku programowania, ale w *Mathematice* cały kod tworzący rysunek 2 to tylko jedna linijka:

```
ArrayPlot[CellularAutomaton[30, {{1}, 0}, 100]]
```

Dla bardziej złożonych automatów komórkowych wcale nie jest dużo trudniej. Ponadto, za pomocą graficznych bibliotek *Mathematiki* można stworzyć wiele spektakularnych ilustracji ewolucji automatów komórkowych – jedną z nich pokazuje rysunek 3.

Automaty i równania różniczkowe. Jak wspomnieliśmy, automaty komórkowe mogą być rozważane jako skrajne przykłady dyskretyzacji równań różniczkowych, gdzie nie tylko czas i przestrzeń są dyskretne, ale także rozwiązanie istnieje tylko w skończonym zbiorze możliwych stanów. Okazuje się, że większość zjawisk opisywanych przy użyciu teorii równań różniczkowych ma swoje odpowiedniki w teorii automatów komórkowych. Między innymi istnieją odpowiedniki tzw. *solitonów* (pewnych fal o ciekawych własnościach), które od wielu lat intrygują badaczy. Symulację solitonu, autorstwa Reiko Sakamoto, widzimy na rysunku 4. Przy okazji, przykład ten pokazuje tzw. *dynamikę*, czyli rodzaj interaktywności, będący jedną z najciekawszych cech *Mathematiki* od wersji 6 włącznie.

Automaty (i uogólnienia), które można badać (prawie) bez komputera. Wprawdzie automaty komórkowe bardzo trudno badać metodami matematycznymi – jest niewiele nietrywialnych twierdzeń – ale, jak widzieliśmy, doskonale nadają się do symulacji komputerowych. Istnieją także automaty komórkowe, nazywane *całkowalnymi*, dla których można otrzymać pewne wyniki za pomocą metod matematyczno-fizycznych. Właśnie takim automatem jest ten na rysunku 4. Ostatnio duże zainteresowanie budzą uogólnienia automatów komórkowych nazywane *równaniami ultradyskretnymi* (ang. *ultra discrete equations*), mające własność całkowalności. Powstają one z równań dyskretnych przez pewne przejście graniczne.

Rozważmy, na przykład, równanie dyskretne

$$x_{n+1}x_{n-1} = k + \frac{1}{x_n}, \tag{1}$$

gdzie k jest stałą. Ponieważ

$$\lim_{\varepsilon\to 0^+} \log(e^{A/\varepsilon} + e^{B/\varepsilon}) = \max(A, B),$$

więc wprowadzając nowe zmienne X_i zdefiniowane wzorem $x_i = e^{X_i/\varepsilon}$ i przechodząc do granicy przy $\varepsilon \to 0^+$, uzyskujemy równanie ultradyskretne

$$X_{n+1} + X_n + X_{n-1} = \max(X_n + K, 0). \tag{2}$$

Dla stałej K i początkowych wartości X_0 i X_1 będących liczbami całkowitymi wszystkie rozwiązania tego równania są całkowite. Można wykazać, że przy iteracjach równania (1) nie zmienia się następująca wielkość:

$$I = x_n + x_{n-1} + \frac{k}{x_n} + \frac{k}{x_{n-1}} + \frac{1}{x_n x_{n-1}}.$$

Odpowiadający jej niezmiennik dla równania (2) to

$$I = \max(X_n, X_n - 1, K - X_{n-1}, K - X_n, -X_n - X_{n-1}).$$

Więcej na ten temat można znaleźć w artykule [2]. Wprawdzie *Mathematica* nie jest w stanie znaleźć ogólnego rozwiązania tych równań, ale za to możemy łatwo zilustrować rozwiązania graficznie w przestrzeni X_n, X_{n+1}, np. z wartościami początkowymi $X_0 = 0$, $X_1 = 1$ dla $K = 1, \ldots, 20$ (zob. rys. 5).

Ogólnie szukanie rozwiązań równań ultradyskretnych jest bardzo trudne, ale zdarzają się wyjątki. Następujący przykład pochodzi z pracy [2]. W równaniu dyskretnym $x_{n-1}x_{n+1} = zx_n$ wprowadzamy nowe zmienne $X_i = \varepsilon \log x_i$, otrzymując ultradyskretne równanie $X_{n-1} + X_{n+1} = X_n + n$. Szczęśliwie jest to po prostu równanie liniowe, więc *Mathematica* potrafi je rozwiązać:

```
RSolve[X[n-1]+X[n+1] == X[n] + n, X[n], n] // Simplify
```

$$\{\{\texttt{X[n]} \to c_1 e^{\frac{i\pi n}{3}} + c_2 e^{\frac{-i\pi n}{3}} + n\}\}.$$

Literatura

[1] `http://www.wolfram.com`

[2] D. Takahashi, T. Tokihiro, B. Grammaticos, Y. Ohta, A. Ramani, *Constructing solutions to the ultradiscrete Painlevé equations*, J. Phys. A Math. Gen. 30 (1997), 7953–7966.

[3] N. Joshi, S. Lafortune, *How to detect integrability in cellular automata*, J. Phys. A Math. Gen. 38 (2005), L49.