Autômatos Celulares: História, Definição e Implementação em C

Lucas Pacheco[1], **Guilherme dos Santos Ortiz**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR) - Ponta Grossa, PR - Brazil

lucaspacheco@alunos.utfpr.edu.br, gui_ortiz_santos12@gmail.com

Abstract. *Cellular automata are able to represent complex systems simply and efficiently, using cells, predefined states and a set of rules, if it is possible to simulate biological evolution, model urban spaces and even generate music. This article tells a bit of its history, shows its definitions, workings, and makes a C implementation for better visualization and understanding.*

Resumo. *Autômatos Celulares são capazes de representar sistemas complexos de forma simples e eficiente. Utilizando de células, estados pré-definidos e um conjunto de regras, se é possível simular evoluções biológicas, modelar espaços urbanos e até gerar músicas. Neste artigo é contado um pouco de sua história, mostrado suas definições, funcionamentos e realizado uma implementação em C para melhor visualização e entendimento.*

1. Introdução

Os Autômatos Celulares são uma classe de sistemas matemáticos determinísticos discretos, ou seja, o espaço, tempo e propriedades do autômato possuem um número finito de estados. Seu objetivo é a simulação de sistemas complexos através de iterações de células utilizando regras. Eles também possuem uma alta capacidade de executar computações complexas de forma eficiênte e segura.

O uso de Autômatos Celulares (ACs) são amplos, estudos partem desde modeladem da dinâmica espacial urbana [2] até a representação musical da evolução de um autômato celular [3]. Por a lógica de um AC ser considerada de extrema simplicidade, há pesquisas visando seu uso em arquiteturas descentralizadas de computadores [4] utilizando de uma programação mais simplificada. Mesmo com tantas pesquisas, seu conhecimento ainda é muito vago e acredita-se que ainda há muito o que se descobrir.

Neste artigo vamos ter conhecimento sobre um pouco da história dos Autômatos Celulares, veremos sua definição e do que é composto, por final implementaremos um Auômato Celular Unidimensional utilizando a liguagem C.

2. Autômatos Celulares

2.1. História

Em 1948, motivado pela busca de um modelo reducionista de evolução biológica, John von Neumann elaborou o Autômato Celular. Através da ajuda de Stanislaw Ulam, que trabalhava com estudo em crescimento de cristais modelando-os usando grelha, Neumann concebeu um modelo matemático abstrato, o primeiro automato celular de duas dimensões onde cada célula possuía 29 estados possíveis.

O matemático John Conway, vinte anos depois, concebeu um automato celular de duas dimensões denominado "O Jogo da Vida". Nele a vida e a morte das células são definidas de acordo com sua vizinhança, surgindo assim um padrão das células ou a morte de todas. O jogo possui regras simples porém bem pensadas, e com elas se possibilitou reproduzir evolução de seres vivos.

Em 1983, Wolfram publicou um artigo de revisão contendo as propriedades de um Automato Celular, com isso legitimou efetivamente como um campo de pesquisa na matemática. Um ano após, Wolfram juntamente com Toffoli e Farmer organizaram a primeira conferência sobre Automatos Celulares, sendo sediada no MIT em Boston [10].

2.2. Definições de um Autômato Celular

2.2.1. Definição Informal

Um Autômato Celular é composto pelas seguintes propriedades:

- Um espaço de células.
- Um conjunto de estados para cada célula.
- Uma vizinhança para definir o estado da célula.
- Uma regra de transição para especificar o comportamento de uma célula de acordo com sua vizinhança.

O estado de cada célula de um Autômato Celular no tempo t+1 é definido de acordo com sua vizinhança no tempo t e de acordo com a regra de transição.

2.2.2. Definição Formal

Segundo Weimar (*1996, apud CASTRO 2015 p. 89*) [6], sejam:

- L uma grade regular (os elementos de L são chamados de células)
- S um conjunto finito de estados
- N um conjunto finito, (de tamanho n = $|N|$) de vizinhança, tal que $\forall c \in N, \forall r \in L : r + c \in L$
- f : S $\rightarrow$ S

Temos a quádrupla (L, S, N, f), um Autômato Celular.

2.3. Elementos de um Autômato Celular

2.3.1. Célula

A célula de um Autômato Celular guarda o seu estado. Em Autômatos mais simples os estados são binários, 1 ou 0, em Autômatos mais complexos podem haver mais estados para uma célula. A partir do estado inicial de uma célula é que o Autômato Celular gera seu crescimento, podendo variar de acordo com diferentes tipos de estados em seu início.

Pensando em uma forma de representação física de um Autômato Celular, suas células podem ser triângulares, quadrangulares, hexagonais ou outras formas geométricas dependendo da aplicação (Figura 1).

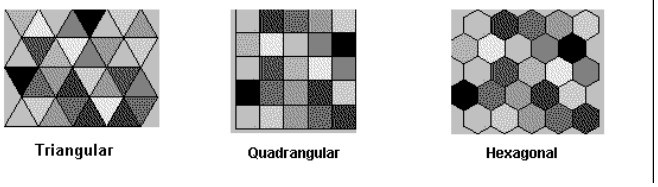

Figura 1. Formato da Célula

2.3.2. Grade ou Malha

Como visto em sua definição formal, é o conjunto de células L. Nele está contido todas as células em n-dimensões, no conjunto mais simples unidimensional as células são organizadas em uma linha. Um Autômato Celular unidimensional é mais fácil de ser visualizado e entendido, como veremos ser implementado posteriormente. Autômatios Celulares de duas dimensões podem ser representados utilizando uma forma matricial de x e y, já o de três dimensões adiciona-se a coordenada z (Figura 2).

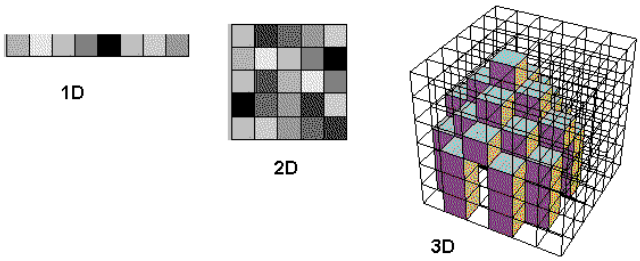

Figura 2. Dimensão da Malha

2.3.3. Vizinhança

Com os elementos anteriores temos um estado estático, para termos uma dinâmica no Autômato devemos declarar um conjunto de regras. A regras possuem como objetivo definir o estado da célula no tempo t+1 de acordo com sua vizinhança no tempo t.

Von Neumann definiu a vizinhança de uma célula com sendo um conjunto de quatro células, no caso tendo raio igual a 1 (Figura 3). Já Moore definiu vizinhança como sendo o conjunto de oito células ao redor da célula principal quando raio igual a 1 (Figura 3).

2.3.4. Regras

Em um Autômato Celular unidimensional, cada célula pode receber valores de 0 a k-1 valores. Se considerarmos uma vizinhança de três células, temos 2^3= 8 padrões, logo 2^8 = 256 regras possíveis, indo de 0 a 255. Cada valor de regra é transformada em binário de 8 bits e com esse valor conseguimos definir qual estado a célula irá receber de acordo com sua vizinhança (Figura 4).

De forma generalizada temos quatro tipos de regras:

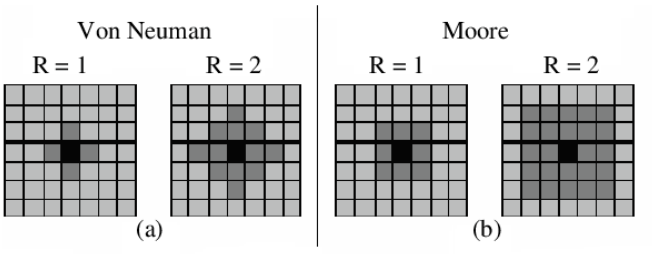

Figura 3. Alguns Tipos de Vizinhança

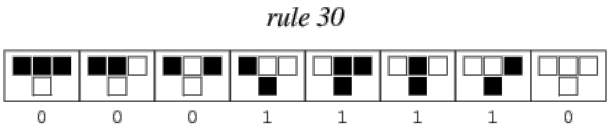

Figura 4. Regra 30

1. **Especificação Direta:** Ela consiste na tabelação de estados possíveis dos vizinhos da célula atual. Através da tabela e da regra selecionada conseguimos definir o novo estado da célula.
2. **Regra Totalísticas**: Nela não importa a ordem de estados de cada vizinho e sim na quantidade de vizinhos que estão em um estado específico. Através do valor da quantidade se define qual será o estado da célula atuala.
3. **Especificação Implícita:** O estado a ser definido na célula é calculado através de uma fórmula, nessa fórmula se encontra possíveis estados dos vizinhos e com isso o novo estado da célula.
4. **Regra Probabilística:** De acordo com a organização da vizinhança se define uma probabilidade para cada estado, a soma das probabilidades devem sempre ser igual a 1.

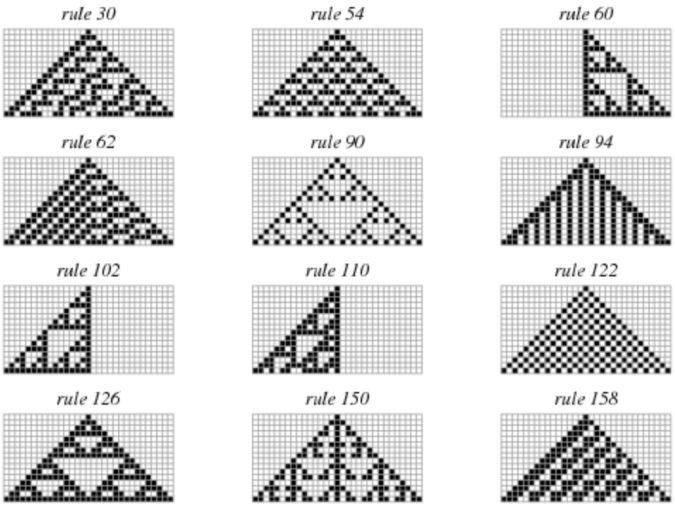

Figura 5. Exemplos da Evolução de Acordo com Regras Diferentes

3. Implementação em C

Autômatos Celulares Unidimensionais são de mais fácil compreensão e visualização, se comparados com de duas ou três dimensões, por isso iremos implementar-los utilizando

a linguagem C. Neste código será possível selecionar o tamanho da linha, ou seja, o tamanho do vetor contendo as células, quantas evoluções ocorrerá, definido pelo número de linhas, e qual regra será utilizada para definir sua evolução.

Sua ideia base é utilizar de vetores para reprensentar as células no estado atual e as células em um estado prévio. Ambos vetores são de char e alocados dinamicamente, o vetor *CAprev* é inicializado tendo todos as celulas como 0 menos a célula do meio. Neste caso a célula do meio é quem inicia a evolução, com ela se começa a gerar os padrões de acordo com a regra selecionada.

```
16    // Alocar dois vetores char, um para a linha prévia e outra para a linha atual
17    CAprev = (char*) malloc(N*sizeof(char));
18    CAatu = (char*) malloc(N*sizeof(char));
19
20    for(n=0; n<N; n++)  CAprev[n] = 0;
21
22    // Inicializa tendo apenas um estado ativo no meio da linha prévia
23    CAprev[N/2] = 1;
```

Figura 6. Alocação e Inicialização dos Vetores

Após gerado ambos vetores, começamos com o cálculo da célula atual. Como o vetor *CAprev* já foi inicializado com uma célula de estado 1 no meio, o cálculo é feito a partir de 2, ou seja, a segunda linha da imagem. Através de um laço percorremos por todas as linhas, dentro desse laço possuimos outro laço que percorre célula por célula no vetor.

Como o vetor *CAprev* temos acesso aos elementos vizinhos da célula atual num espaço de tempo anterior, com isso utilizamos operadores de shift bit a bit para criarmos um valor em binário que mostra os etados das três células vizinhas. A célula *CAprev[n-1]* é representada pelo terceiro bit, a célula *CAprev[n]* pelo segundo bit e a célula *CAprev[n+1]* pelo primeiro bit. Agora para calcularmos o valor da célula atual utilizamos do *1*<<|bit, ele realiza 2^{bit} deslocamentos de bits. O *(rule & (1 <<|bit))* faz uma verificação se dentro de *rule* a vizinhança coincide, com *!= 0* garantimos que a célula apenas receberá os valores 0 ou 1(Figura 7).

```
27    for(k = 2; k <= M; k++){
28      for(n = 1; n < N - 1; n++){
29        bit = CAprev[n-1] << 2 | CAprev[n] << 1 | CAprev[n+1];
30        CAatu[n] = (rule &(1<<bit)) != 0;
31      }
32
33      // Print do vetor CAprev
34      for(int p = 0; p < N; p++){
35        if(CAprev[p] == 0) printf("  ");
36        else  printf("O ");
37      }
38      puts("");
39
40      // Copia o Vetor
41      for(int p = 0; p < N; p++)  CAprev[p] = CAatu[p];
42      system("sleep 0.18");
```

Figura 7. Execução do Cálculo do Estado da Célula Atual

Após o cálculo do vetor, podemos imprimir ele no termincal de comando. É necessário também realizar a cópia dos valores do vetor *CAatu* para o vetor *CAprev*, já que no próximo período de tempo o atual se tornará o prévio.

Com essas funções conseguimos ter como saída todas as regras possíveis, de 0 a 255 (Figura 8 e 9).

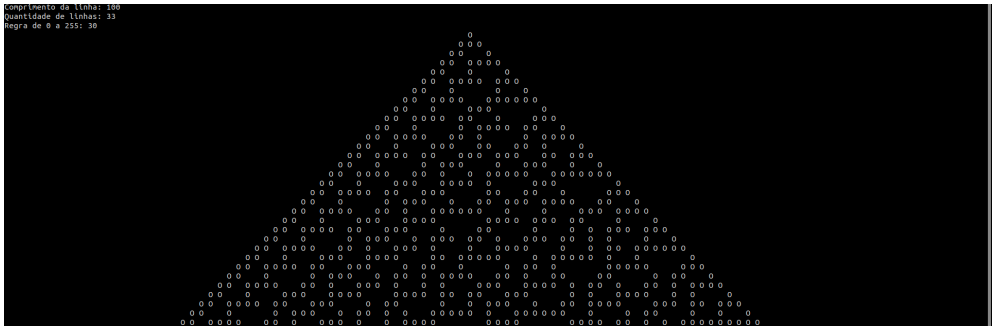

Figura 8. Autômato Celular Regra 30

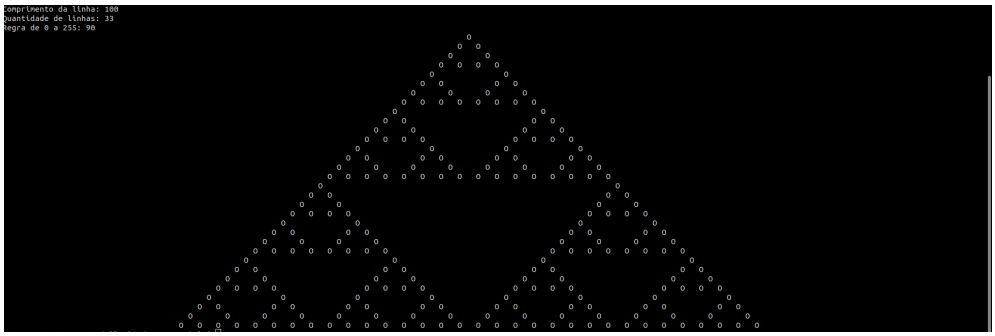

Figura 9. Autômato Celular Regra 90

4. Conclusão

O estudo dos Autômatos Celulares é uma área que está crescendo e ainda há muita coisa a ser descoberta, seu uso pode acabar simplificando muitos sistemas complexos de serem representados e isso é de grande necessidade. Pudemos ver seu funcionamento e entender mais a fundo como tudo é realizado. Em sua implementação somos capazes de perceber o tamanho de sua simplicidade e termos noção de seu modo de volução.

Referências

[1] Toffoli T. (1984), *Cellular Automata as an Alternative to (Rather than an Approximation of) Differential Equations in Modelling Physics*, Physica 10D, 117-127

[2] TRENTIN, G.; FREITAS, M. I. C. DE. *Modelagem Da Dinâmica Espacial Urbana: Modelo De Autômato Celular Na Simulação De Cenários Para o Município De Americana-SP*. Revista Brasileira de Cartografia, v. 62, 11.

[3] IBM. *Cellular Automata and Music* "https://www.ibm.com/developerworks/library/j-camusic/'

[4] Mitchell, M. (1996). *Computation in cellular automata: A selected review.* Nonstandard Computation, 95-140.

[5] Oliveira, G. M., Omar, N., de Oliveira, P. P. (2000). *Computação e evolução em autômatos celulares unidimensionais.* Revista Mackenzie de Engenharia e Computação, 1(1).

[6] Castro, M. L. A., de Oliveira Castro, R. (2015). *Autômatos celulares: implementações de von Neumann, Conway e Wolfram.* Revista de Ciências Exatas e Tecnologia, 3(3), 89-106.

[7] Sarkar, P. (2000). *A brief history of cellular automata.* Acm computing surveys (csur), 32(1), 80-107.

[8] LifeWIki. *Cellular Automata* "https://www.conwaylife.com/wiki/Cellular_automaton Common_dimensions_and_neighborhoods"

[9] WolframMathWolrd *Elementary Cellular Automata* "http://mathworld.wolfram.com/ElementaryCellularAutomaton.html"

[10] Ilachinski, A. (2001). *Cellular automata: a discrete universe.* World Scientific Publishing Company.

[11] Peitgen, H. O., Jürgens, H., Saupe, D. (2006). *Chaos and fractals: new frontiers of science.* Springer Science Business Media.

[12] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.