**Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG**
**Disciplina:** Princípio de Modelagem Matemática
**Dia/Horário:** Segunda e Quarta-feira, das 14h50min às 16h30min


# Autômato Celular - Wolfram

## RELATÓRIO – PRÁTICA 03

Michelle Hanne Soares de Andrade

Belo Horizonte
Maio, 2023.

# SUMÁRIO

# 1- INTRODUÇÃO

Os Autômatos Celulares de Wolfram são sistemas dinâmicos discretos, compostos por um conjunto de células organizadas em uma grade unidimensional, bidimensional ou multidimensional. Essas células podem assumir um número finito de estados e evoluem ao longo do tempo de acordo com um conjunto de regras determinísticas simples (Wolfram, 1986).

O cientista Stephen Wolfram fez importantes contribuições para a teoria e aplicação dos Autômatos Celulares. Wolfram explora diferentes aspectos dos Autômatos Celulares e apresenta uma seleção de trabalhos que abordam suas propriedades e aplicações em vários domínios (Wolfram, 1994).

Um aspecto importante dos Autômatos Celulares de Wolfram é o estudo de suas regras de transição. Cada célula atualiza seu estado com base no estado das células vizinhas, conforme especificado pelas regras. Essas regras podem ser definidas de maneira binária ou em outras formas, e a escolha das regras influencia a evolução e o comportamento global do Autômato Celular.

## 1.1 - Objetivo

Reproduzir o algoritmo Autômato Celular proposto pelo cientista Wolfram, com 256 regras possíveis (de 0 a 255). O algoritmo deverá solicitar o número da regra a ser calculada e iniciar os valores iniciais de $L$ (largura) e $H$ (altura). Posteriormente, será calculado o autômato correspondente à regra e exibida a imagem referente gerada pelo Autômato Celular de Wolfram.

## 2- REVISÃO BIBLIOGRÁFICA

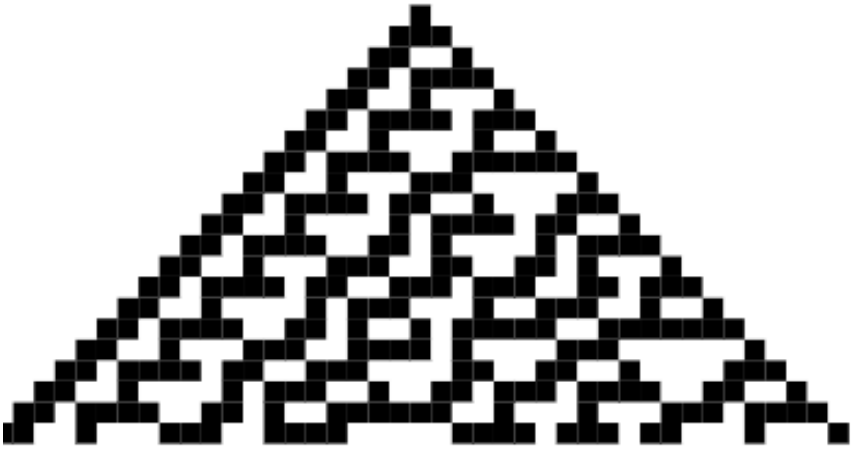
Os autômatos celulares de Wolfram são sistemas computacionais discretos que consistem em uma grade regular de células, onde cada célula pode estar em um estado específico. Essas células evoluem ao longo do tempo de acordo com regras de transição determinísticas. As regras de transição governam como o estado de cada célula é atualizado com base nos estados das células vizinhas em uma determinada geração.
(Wolfram, 1994), (Wolfram, 2002).

Em autômatos celulares de uma dimensão, cada célula tem duas células vizinhas (esquerda e direita). Em autômatos celulares bidimensionais, as células têm vizinhanças de Moore ou vizinhanças de von Neumann, onde diferentes padrões de células vizinhas são considerados.

Nas regras de Wolfram, as células vizinhas e o estado atual da célula são mapeados para um novo estado na próxima geração. Para isso usamos uma tabela de 8 bits, onde cada combinação de estados vizinhos corresponde a um novo estado para a célula central.

No autômato celular elementar de Wolfram, existem 256 regras possíveis (de 0 a 255). Cada regra é representada como um número decimal, que é convertido em uma sequência binária de 8 bits. Cada bit da sequência binária especifica o novo estado da célula central com base no estado das células vizinhas (Wolfram, 2002).

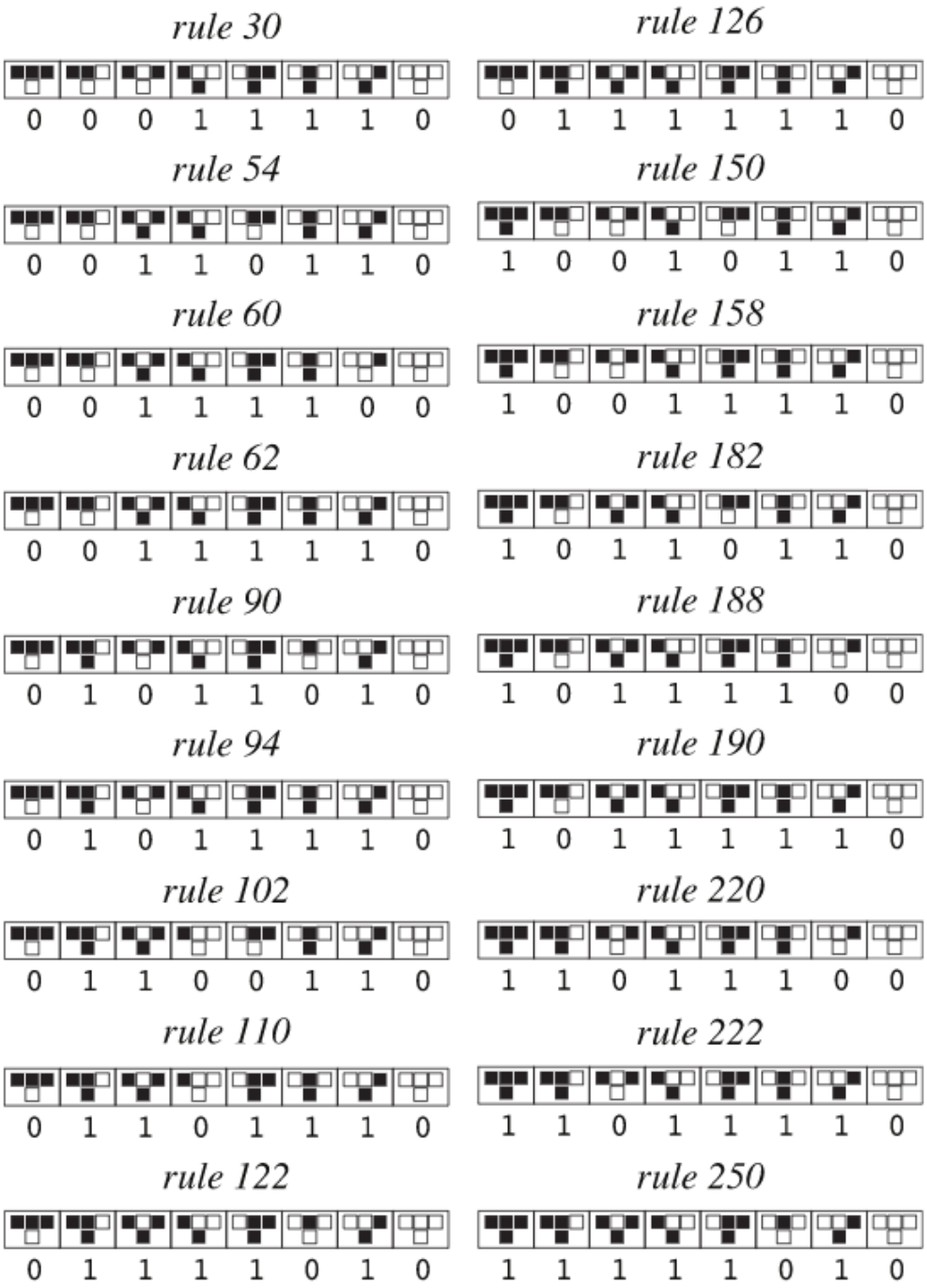
**Figura 1:** Autômato Celular da regra 30



**Fonte:** Wolfram MathWorld – Disponível em <https://mathworld.wolfram.com/ElementaryCellularAutomaton.html>. Acesso em: 26 maio 2023.

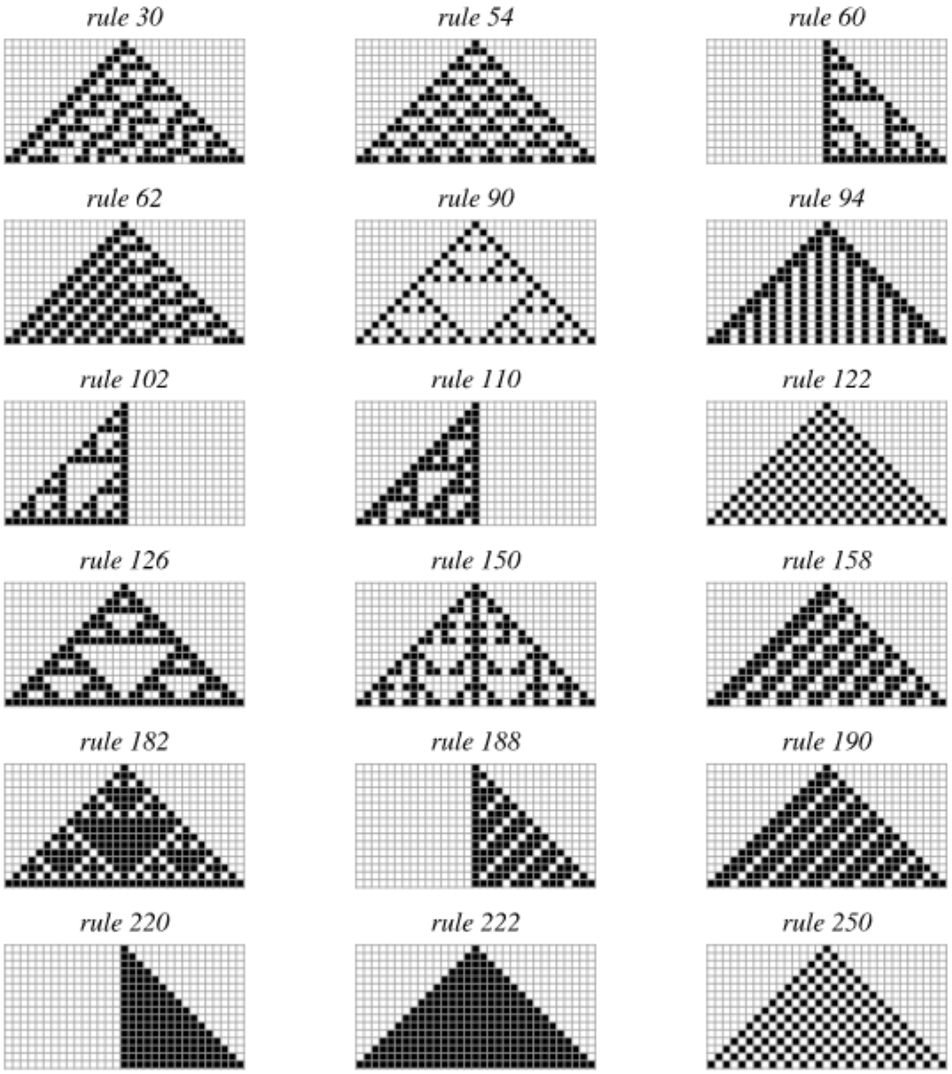
A evolução de um autômato celular unidimensional pode ser ilustrada começando com o estado inicial (geração zero) na primeira linha, a primeira geração na segunda linha e assim por diante. A Figura 1 ilustra as primeiras 20 gerações do autômato celular elementar da regra 30 começando com uma única célula preta.

**Figura 2:** Exemplos de Regras em Binário do Autômato Celular de Wolfram




**Fonte:** Wolfram MathWorld – Disponível em <https://mathworld.wolfram.com/ElementaryCellularAutomaton.html>. Acesso em: 26 maio 2023.

**Figura 3:** Exemplos de Saídas de Regras do Autômato Celular de Wolfram




**Fonte:** Wolfram MathWorld – Disponível em <https://mathworld.wolfram.com/ElementaryCellularAutomaton.html>. Acesso em: 26 maio 2023.

As ilustrações acima (Figura 2 e Figura 3) mostram alguns números de autômatos que fornecem um padrão particularmente interessante propagado por 15 gerações começando com uma única célula ocupada (preta) na iteração inicial (Wolfram, 2002).

Ao iniciar o autômato celular com uma configuração inicial de células, as regras de transição são aplicadas repetidamente para cada geração subsequente. A evolução do autômato celular pode levar a padrões complexos e comportamentos emergentes, mesmo quando as regras de transição são simples.

Assim, pode-se observar que $a_i(t)$ denota o estado $i$ da célula no tempo $t = 0, 1, \ldots$ O valor pode ser escrito explicitamente em termos das células adjacentes, sendo anterior, atual e posterior, conforme a função abaixo:

$$a_{i(t)} = f\left(a_{(i-1)(t-1)}, a_{i(t-1)}, a_{(i+1)(t-1)}\right). \quad (1)$$

Pode-se apresentar os valores $a_{i(t)}$ por expressões booleanas, conforme os exemplos de regras abaixo (Wolfram 2002, p. 869 ).

$$f_{(30)(p,q,r)} = Xor\left[p, Or[q, r]\right] \quad (2)$$

$$f_{(90)(p,q,r)} = Xor[p, r] \quad (3)$$

$$f_{(110)(p,q,r)} = Xor\left[Or[p, q], And[p, q, r]\right] \quad (4)$$

$$f_{(250)(p,q,r)} = Or[p, r] \quad (5)$$

$$f_{(254)(p,q,r)} = Or[p, q, r] \quad (6)$$

Dos $2^8 = 256$ Autômatos Celulares elementares, existem 88 regras fundamentalmente inequivalentes (Wolfram 2002, p. 57).

Os Autômatos Celulares de uma dimensão possuem características qualitativas com relação a evolução e fornecem evidências empíricas para a existência de quatro classes básicas de comportamento (Wolfram *apud* Melotti, 2009):

- **Classe I (comportamento fixo):** a evolução leva a um estado homogêneo (todas as células estão no mesmo estado). Exemplo de regras 0, 32 e 255.
- **Classe II (comportamento cíclico ou periódico):** nesta classe os autômatos geralmente criam imagens que se repetem periodicamente, com poucos períodos ou imagens estáveis. Exemplo de regras 36 e 51 e 54.
- **Classe III (comportamento caótico):** exibem comportamentos caóticos. O termo "caótico" aqui refere-se o comportamento espaço-tempo aparentemente imprevisível (Mitchell *apud* Melotti, 2009). Os autômatos da classe III são caracterizados pela forte dependência com as condições iniciais. Exemplo de regras 18, 30 e 90.
- **Classe IV (comportamento complexo):** exibem comportamento complexo, apresentam uma repetição irregular de padrões no tempo, ocorrendo em diferentes escalas e posições no espaço. Esses autômatos celulares combinam certa regularidade com alguma imprevisibilidade. Apesar da sua numeração sugerir que estão entre as Classes

II e III, eles se distinguem por evoluírem para estruturas complexas localizadas que podem crescer e contrair ao longo do tempo. Exemplo de regras 20 e 110.

## 3- ALGORITMO PROPOSTO

O algoritmo proposto foi escrito em Pyhton3, e implementa o Autômato Celular de Wolfram com a possibilidade de escolha de 256 regras, do número 0 até 255 e o estado inicial (o - ocupado, r – aleatório)

Primeiramente foram importadas bibliotecas para gerar imagem (PIL - ***pillow)*** e números randômicos.

```
from PIL import Image
import random
```

Para melhor organização foram criadas duas funções: `to_binary(value)` e `def cellular_automaton(rule, width, height, random_initial_state=False):`

Abaixo o código fonte relativo a função de conversão em um número binário:

```
def to_binary(value):
    binary = ""
    while value > 0:
        binary = str(value % 2) + binary
        value = value // 2
    return binary.zfill(8)
```

A função `to_binary(value)`é responsável por converter um número decimal para uma representação binária. A função recebe um valor decimal *value*. Em seguida tem um loop, ele divide o valor por 2 e obtém o resto da divisão, que é um dígito binário. O dígito binário é concatenado à esquerda da string *binary*. Em seguida, o valor é dividido por 2 novamente, e o processo continua até que o valor seja 0.
A função retorna a string binária preenchida com zeros à esquerda usando o método *zfill(8)* para garantir um ***comprimento de 8 caracteres***.

Abaixo a função que recebe os parâmetros do Autômato Celular e implementa a execução das regras:

```
def cellular_automaton(rule, width, height, random_initial_state=False):
    # Inicialização da matriz do autômato celular
    automaton = [[0] * width]
```

```
    # Definindo o estado inicial
    if random_initial_state:
        automaton[0] = [random.choice([0, 1]) for _ in range(width)]
    else:
        automaton[0][width // 2] = 1  # Definindo a célula central como 1

    # Definição das regras
    ruleset = to_binary(rule)

    # Geração das próximas linhas do autômato celular
    for i in range(1, height):
        row = []
        for j in range(width):
            # Obtendo os valores das células vizinhas
            left = automaton[i - 1][(j - 1) % width]
            center = automaton[i - 1][j]
            right = automaton[i - 1][(j + 1) % width]

            # Convertendo os valores das células vizinhas em uma sequência bi-
nária
            pattern = ''.join([str(left), str(center), str(right)])

            # Aplicando as regras
            index = 7 - int(pattern, 2)  # Revertendo o padrão binário para
corresponder à ordem das regras
            next_cell = int(ruleset[index])

            row.append(next_cell)

        automaton.append(row)

    # Imprimindo o autômato celular como uma figura
    for row in automaton:
        line = ''.join(['#' if cell == 1 else ' ' for cell in row])
        print(line)

    # Criando a imagem
    image = Image.new("1", (width, height))

    # Preenchendo a imagem com os valores do autômato celular
    for y, row in enumerate(automaton):
        for x, cell in enumerate(row):
            pixel = 255 if cell == 1 else 0
            image.putpixel((x, y), pixel)

    # Salvando a imagem como um arquivo PNG
    image.save("automaton.png")
```

```
    print("Imagem salva com sucesso!")
```

A função *cellular_automaton(rule, width, height)* implementa o autômato celular de Wolfram. Ela recebe quatro parâmetros: `*rule*` (número da regra), `*width*` (largura), `*height*` (altura) e `*random_initial_state*`=*False*.

Inicializa uma matriz `**automaton**` para representar o Autômato Celular, com uma linha inicial de **zeros** e a **célula central definida como 1 ou com valores aleatórios**.
Se **random_initial_state for True**, ou seja, se o usuário escolheu um estado inicial aleatório, cada célula na primeira linha do autômato será definida como um valor **aleatório entre 0 e 1.** Isso é feito usando uma a função ***random.choice([0, 1])***. Se *random_initial_state for False*, ou seja, se o usuário **escolheu um estado inicial com um único sítio ocupado**, **apenas a célula central da primeira linha do autômato será definida como 1**, enquanto as outras células serão **preenchidas com 0**. Isso é feito atribuindo diretamente **automaton[0][width // 2] = 1**, onde **width // 2** representa o índice da célula central na primeira linha.

Em seguida, converte o número da regra em uma representação **binária de 8 dígitos usando a função `to_binary`.** Posteriormente, em um loop, gera as próximas linhas do autômato celular. Para cada célula na linha atual, obtém os valores das **células vizinhas (esquerda, centro e direita)** a partir da linha anterior, considerando os casos de borda. Converte os valores das células vizinhas em uma **sequência binária de três dígitos.** Posteriormente, inverte a sequência binária para corresponder à ordem das regras de Wolfram.
Com base no valor binário invertido, determina o próximo estado da célula atual usando a regra fornecida. E armazena o próximo estado da célula na linha atual.
Após gerar todas as linhas do autômato celular, imprime-o como uma figura, onde "#" representa uma célula com conteúdo e " " representa uma célula sem conteúdo. E, por fim, a imagem é montada com os valores do autômato celular gerado anteriormente.

```
# Parâmetros iniciais
L = 80
H = 100
rule = int(input("Digite o número da regra (0-255): "))
initial_state_choice = input("Escolha o estado inicial (o - ocupado, r - alea-
tório): ")
```

```
# Convertendo a escolha do estado inicial para um valor booleano
random_initial_state = False
if initial_state_choice.lower() == "r":
    random_initial_state = True

# Executando o autômato celular
cellular_automaton(rule, L, H, random_initial_state)
```

A parte final do código possui os parâmetros iniciais **`L` (largura) e `H` (altura).** Solicita ao usuário que digite **o número da regra desejada**. Também é solicitado ao usuário a escolha entre: ***"o"*** para um estado inicial com um único **sítio ocupado** ou **"r"** para um **estado inicial aleatório**, onde os sítios são preenchidos de forma aleatória com 0s e 1s.
A função `**cellular_automaton**` é chamada com os parâmetros fornecidos para executar o autômato celular e exibir a figura resultante.

O código está disponível em: https://github.com/mihanne/Praticas_PMMC/tree/main/Pratica3-CA
**Versão Notebook: Cellular_Automata.ipynb** - Para executar o arquivo é necessário abrir um ambiente Notebook, como por exemplo o **Google Colaboratory**. Em seguida, deve-se carregar o arquivo e executar.
**Versão Python: cellular_automata.py** – Para executar o arquivo é necessário o Python3 instalado no computador, em seguida digitar no local onde o arquivo se encontra: ***python3 cellular_automata.py***
**Consideração:** Caso esteja usando alguma versão anterior do **Python3**, instalar a biblioteca de imagem usando o comando ***pip install pillow***.

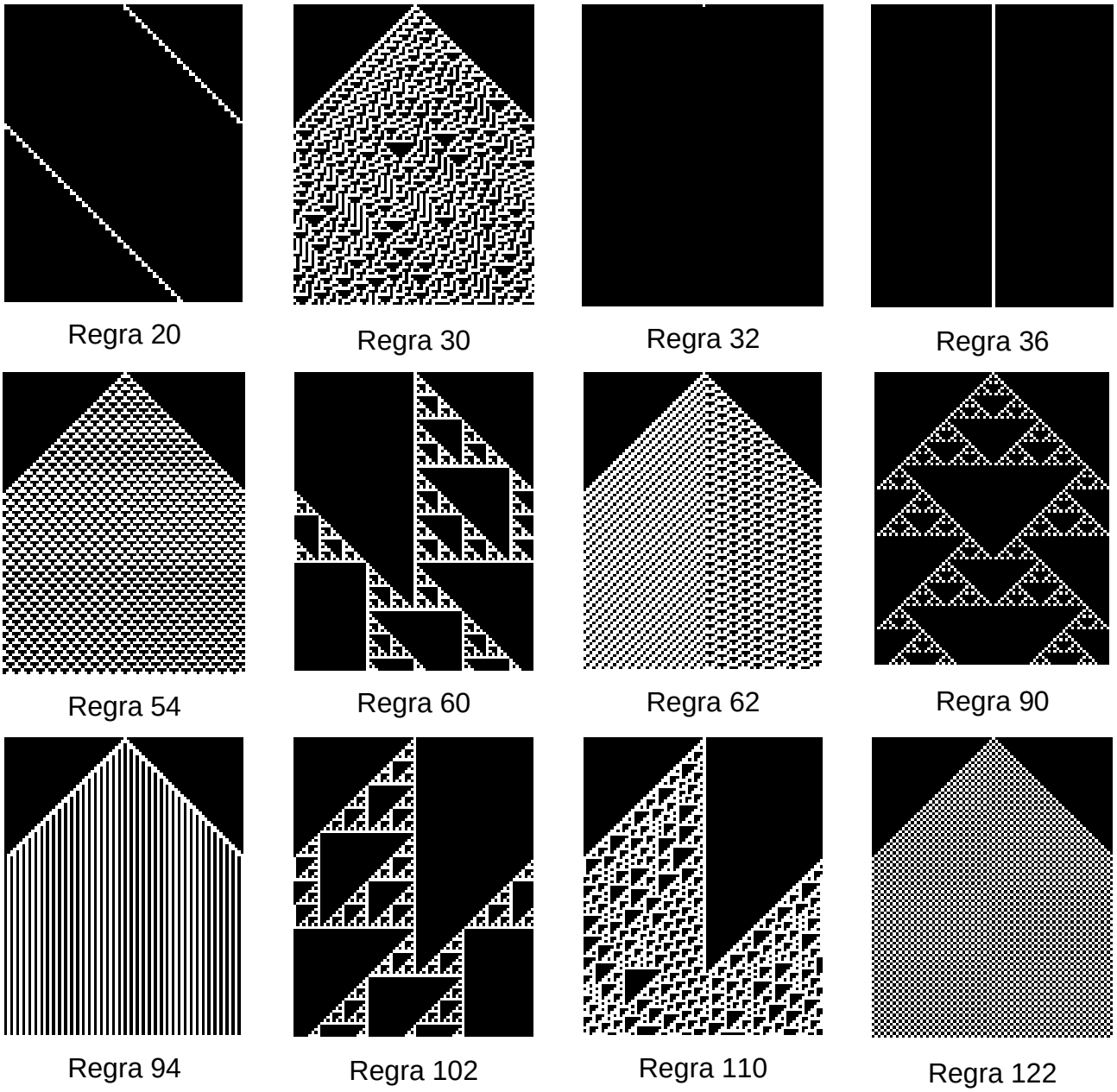
## 4- DISCUSSÃO

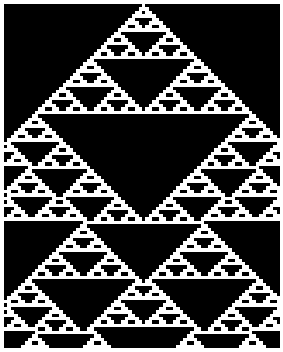
O algoritmo foi executado com todas as regras. Foram geradas imagens de **22 regras** (Figura 4), utilizando o algoritmo de Autômatos Celulares, são elas: **20, 30, 32, 36, 54, 60, 62, 90, 94, 102, 110,122, 126, 150, 158, 182, 188, 190, 220, 222, 250 e 255**. Todos os exemplos foram gerados utilizando a característica de um sítio ocupado.

Visando exemplificar cada Classe de Autômatos Celulares de Wolfram, tem-se que as seguintes regras:

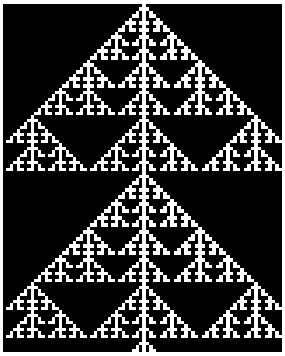
- 32 e 255 pertencem a Classe I (comportamento fixo ou homogêneo);
- 36 e 54 pertencem a Classe II (comportamento cíclico ou periódico);
- 30 e 90 pertencem a Classe III (comportamento caótico);
- 20 e 110 pertencem a Classe IV (comportamento complexo);

**Figura 4:** Exemplos de Saídas de Regras do Autômato Celular de Wolfram

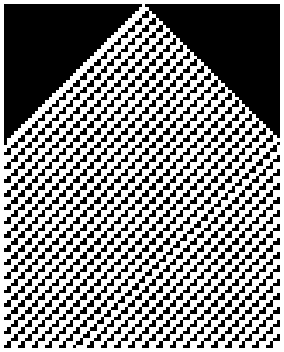




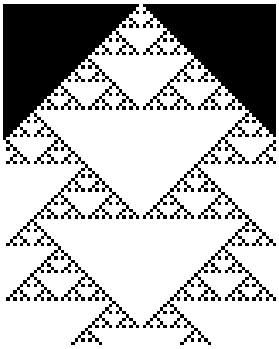
Regra 126



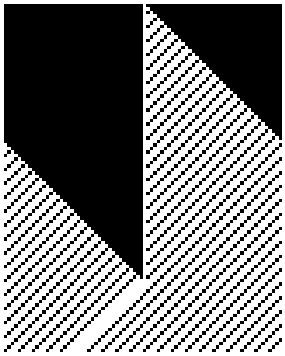
Regra 150



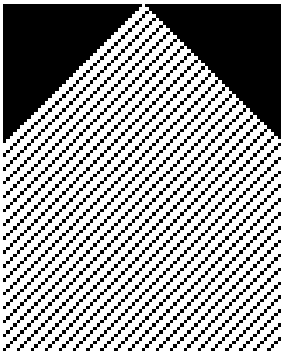
Regra 150



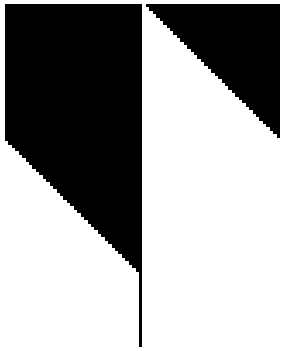
Regra 182



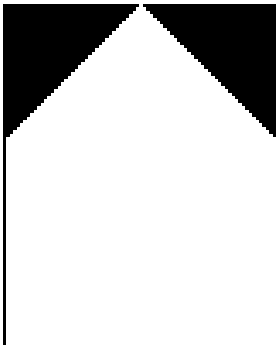
Regra 188



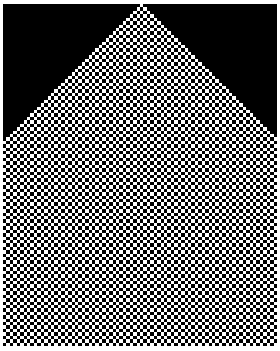
Regra 190



Regra 220



Regra 222



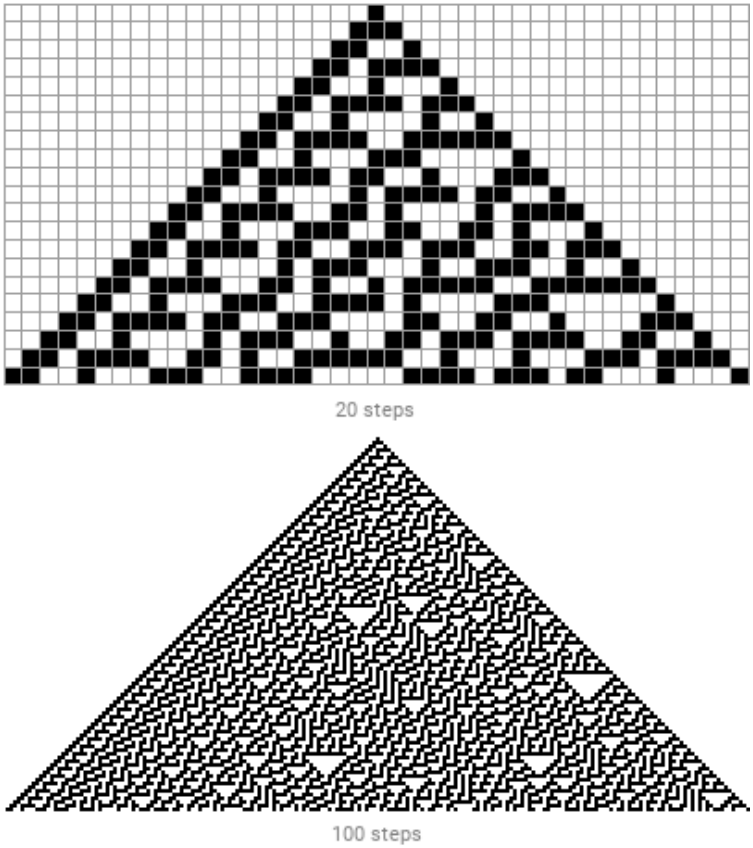
Regra 250

Regra 255

**Fonte:** o autor, gerado pelo algoritmo Autômato Celular Wolfram.

As imagens em pixels (Figura 4) geradas pelas regras acima possuem a largura (80) e altura (100) determinados na chamada da função. Comparando com as saídas mencionadas no site Wolfram Alpha sobre os Autômatos Celulares, as imagens das regras são equivalentes. Como exemplo, a regra número 30, disponível em https://www.wolframalpha.com/input?i=rule+30.

**Figura 5:** Regras 30 do Autômato Celular de Wolfram




**Fonte:** Wolfram Alpha – Disponível em <https://www.wolframalpha.com/input?i=rule+30>. Acesso em: 26 jun. 2023.

Além de gerar as imagens, conforme visto na Figura 4, o algoritmo implementado também mostra a imagem no formato de caractere ‘#’, conforme a regra 30 exibida na Figura 5.

**Figura 6:** Exemplos de Saída da Regra 30 do Autômato Celular de Wolfram


**Fonte:** o autor, gerado pelo algoritmo Autômato Celular Wolfram.

## 5- CONCLUSÃO

Os Autômatos Celulares de Wolfram podem ser aplicados em diversos campos, incluindo física, matemática, biologia e computação. Os Autômatos Celulares são usados para modelar fenômenos complexos, como padrões de crescimento, propagação de epidemias, simulações de fluidos e criptografia.

Em resumo, a teoria e as aplicações dos Autômatos Celulares de Wolfram são um campo fascinante de pesquisa que explora as propriedades emergentes de sistemas simples, mas altamente dinâmicos. Eles fornecem uma maneira de estudar e simular fenômenos complexos, revelando comportamentos sobre o objeto de estudo.

## 6- REFERÊNCIAS

MELOTTI, Gledson, Aplicação de Autômatos Celulares em Sistemas Complexos: Um Estudo de Caso em Espalhamento de Epidemias, Dissertação de Mestrado, 12 de fevereiro de 2009, UFMG, Minas Gerais. Disponível em <https://www.ppgee.ufmg.br/documentos/Defesas/802/Dissertacao_Gledson_final.pdf>. Acesso em: 26 maio de 2023.

WOLFRAM, S. (Ed.). Theory and Application of Cellular Automata. Reading, MA: Addison-Wesley, 1986.

WOLFRAM, S. Cellular Automata and Complexity: Collected Papers. Reading, MA: Addison-Wesley, 1994.

WOLFRAM, S. A New Kind of Science. Champaign, IL: Wolfram Media, pp. 23-60, 112, and 865-866, 2002.