Cellular Automata

Diego Castillo Reyes[a], **Marthon Leobardo Yañez Martinez**[a], **Aldo Escamilla Resendiz**[a] **and Muñoz González Eduardo**[a]

[a]*Researcher*

Dra. Miriam Pescador Rojas

Abstract—Cellular automata are a mathematical and computational model used to simulate dynamic systems. This work presents a review of cellular automata, their history, classification, and applications. Additionally, an example of one-dimensional and two-dimensional cellular automata is shown.

Keywords—*Automata, Cellular, Genetic, Algorithms, Simulation*

Contents

1 Introduction 1
2 Background 1
2.1 Motivation 1
2.2 Developments 1
2.3 Applications 1
3 Cellular Automata Algorithm 2
4 Equation 2
5 Adding codes 2
6 References 2
7 Appendix 2
7.1 Alternative title 2
7.2 Info environment 2
7.3 Equation skip value 2
7.4 References 2
8 Contact me 3
References 3

1. Introduction

Cellular automata (CA) are a mathematical and computational model used to simulate dynamic systems. They are composed of a grid of cells, each of which can be in a finite number of states. The state of each cell is updated based on a set of rules that define the behavior of the system. CA are used in various fields, such as physics, biology, and computer science, to model complex systems and study their behavior. This work presents a review of cellular automata, their history, classification, and applications. Additionally, an example of one-dimensional and two-dimensional cellular automata is shown.

2. Background

The concept of cellular automata was invented by Stansilaw Ulam and John Von Neumann in the 1940s while they were working at the Los Alamos National Laboratory. The work on cellular automata began in the 1940s, with significant developments occuring throughout that decade. Von Neumann's comprehensive work on self-replicating automata was published posthumously in 1966 in the book "Theory of Self-Reproducing Automata," edited by Arthur W. Burks.

2.1. Motivation

The primary motivation behind cellular automata was to understand and model complex systems using simple, local rules. This idea was rooted in the study of biological processes and the desire to create self-replicating machines.

2.2. Developments

- Conway's Game of Life (1970): British mathematician John Conway popularized cellular automata with his "Game of Life", a bidimensional binary cellular automaton. This game demonstrated how simple rules could lead to complex emergent behavior, sparking widespread interest and research in cellular automata.
- Stephen Wolfram's work (1980s): Wolfram conducted extensive research on cellular automata, classifying them into four types based on their behavior and demonstrating their potential as models of natural processes and as computational systems.

Fig. 2 An example of Conway's Game of Life.

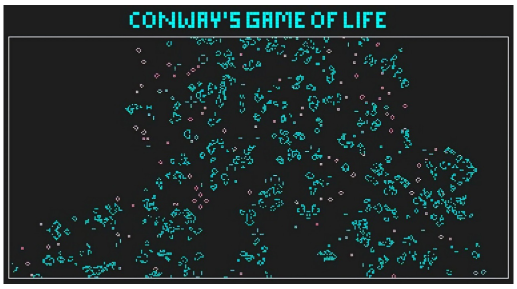

Figure 1. Conway's Game of Life.

2.3. Applications

Cellular automata have been used in various fields, including:

- Computer Science: Parallel computation, cryptography, and image processing.
- Physics: Modeling physical systems, such as fluid dynamics and crystal growth.
- Biology: Simulating biological processes, such as population dynamics and pattern formation.

Fig. 2 Application of cellular automata on Computer Science.

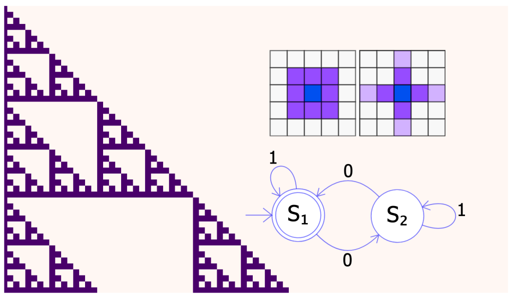

Figure 2. Example of cellular automata.

3. Cellular Automata Algorithm

Algorithm 1: Algoritmo básico de PSO

```
   Input: n: número de partículas, ω: coeficiente de inercia, c_1: factor cognitivo, c_2: factor social
   Output: La mejor solución encontrada
 1 Inicializar aleatoriamente las posiciones $\vec{x}_i$ y $\vec{v}_i$ para las $n$ partículas;
 2 Evaluar la función objetivo con las posiciones $\vec{x}$;
 3 Encontrar $\vec{p}_{Best}$;
 4 Encontrar $\vec{g}_{Best}$;
 5 while $t <$ número máximo de iteraciones do
 6     for $i \leftarrow 1$ to $n$ do
 7         Actualizar la velocidad utilizando;
 8         $\vec{v}_i(t+1) = \omega \cdot \vec{v}_i(t) + c_1 \cdot r_1 \cdot (\vec{p}_{Best} - \vec{x}_i(t)) + c_2 \cdot r_2 \cdot (\vec{g}_{Best} - \vec{x}_i(t))$;
 9         Calcular las nuevas posiciones;
10         $\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t)$;
11         Evaluar las nuevas soluciones utilizando la función objetivo;
12     Encontrar $\vec{p}_{Best}$;
13     Encontrar $\vec{g}_{Best}$;
14     $t := t + 1$;
15 Reportar la mejor solución encontrada;
```

4. Equation

Equation 1, shows the Schrödinger equation as an example.

$$\frac{\hbar^2}{2m}\nabla^2\Psi + V(\mathbf{r})\Psi = -i\hbar\frac{\partial\Psi}{\partial t} \tag{1}$$

The *amssymb* package was not necessary to include, because stix2 font incorporates mathematical symbols for writing quality equations. In case you choose another font, uncomment this package in tau-class/tau.cls/math packages.

If you want to change the values that adjust the spacing above and below the equations, play with `\setlength{\eqskip}{8pt}` value until the preferred spacing is set.

5. Adding codes

This class[1] includes the *listings* package, which offers customized features for adding codes in LaTeX documents specifically for C, C++, LaTeX and Matlab.

You can customize the format in tau-class/tau.cls/listings style.

```
function fibonacci_sequence(num_terms)
    % Initialize the first two terms of the sequence
    fib_sequence = [0, 1];

    if num_terms < 1
        disp('Number of terms should be greater than or equal to 1.');
        return;
    elseif num_terms == 1
        fprintf('Fibonacci Sequence:\n%d\n', fib_sequence(1));
        return;
    elseif num_terms == 2
        fprintf('Fibonacci Sequence:\n%d\n%d\n', fib_sequence(1), fib_sequence(2));
        return;
    end

    % Calculate and display the Fibonacci sequence
    for i = 3:num_terms
        fib_sequence(i) = fib_sequence(i-1) + fib_sequence(i-2);
    end

    fprintf('Fibonacci Sequence:\n');
    disp(fib_sequence);
end
```

Code 1. Example of Matlab code.

[1] Hello there! I am a footnote :)

If line numbering is enabled, we recommend placing the command `\nolinenumbers` at the beginning and `\linenumbers` at the end of the code.

This will temporarily remove line numbering and the code will look better as shown in this example.

6. References

The default formatting for references follows the IEEE style. You can modify the style of your references, for that, go to tau-class/tau.cls/biblatex. See appendix for more information.

7. Appendix

7.1. Alternative title

You can make the following modification in tau-class/tau.cls/title preferences section to change the position of the title.

```
\newcommand{\titlepos}{\centering}
```

Code 2. Alternative title.

This will move the title to the center.

7.2. Info environment

An example of the info environment declared in the 'tauenvs.sty' package is shown below. Remember that *info* and *note* are the only packages that translate their title (english or spanish).

Information

Small example of info environment.

7.3. Equation skip value

With the `\eqskip` command you can change the spacing for equations. The default *eqskip* value is 8pt.

```
 \newlength{\eqskip}\setlength{\eqskip}{8pt}
 \expandafter\def\expandafter\normalsize\expandafter{%
   \normalsize%
   \setlength\abovedisplayskip{\eqskip}%
   \setlength\belowdisplayskip{\eqskip}%
   \setlength\abovedisplayshortskip{\eqskip-\baselineskip}%
   \setlength\belowdisplayshortskip{\eqskip}%
 }
```

Code 3. Equation skip code.

7.4. References

In case you require another reference style, you can go to tau-class/tau.cls/biblatex and modify the following.

```
\RequirePackage[
  backend=biber,
  style=ieee,
  sorting=ynt
]{biblatex}
```

Code 4. References style.

By default, *tau class* has its own .bib for this example, if you want to name your own bib file, change the bibresource.

```
\addbibresource{tau.bib}
```

8. Contact me

Enjoy writing with tau LaTeX class

https://memonotess1.wixsite.com/memonotess

memo.notess1@gmail.com

memo.notess

Did you like this class document? Check out our new project the rho class, made for complex articles and reports.