Neiva, 28 de abril de 2021

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

Luis Angel Gonzales Esquivel, con C.C. No. 83221992,

Autor(es) de la tesis y/o trabajo de grado o TESIS

titulado Problemas P y NP en la Complejidad Computacional

presentado y aprobado en el año 2021 como requisito para optar al título de

Magister en Estudios Interdisciplinarios de la Complejidad;

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

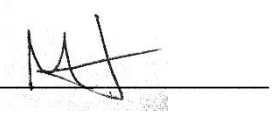
- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores”, los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE:

Firma: 

TÍTULO COMPLETO DEL TRABAJO: Problemas P y NP en la Complejidad Computacional

AUTOR O AUTORES: Luis Angel Gonzales Esquivel

Primero y Segundo Apellido	Primero y Segundo Nombre
Gonzales Esquivel	Luis Angel

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Montealegre Cárdenas	Mauro

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre
Montealegre Cárdenas	Mauro

PARA OPTAR AL TÍTULO DE: Magister en Estudios Interdisciplinarios de la Complejidad

FACULTAD: Facultad de Ciencias Exactas y Naturales

PROGRAMA O POSGRADO: Maestría en Estudios Interdisciplinarios de la Complejidad

CIUDAD: Neiva **AÑO DE PRESENTACIÓN:** 2021 **NÚMERO DE PÁGINAS:** 118

TIPO DE ILUSTRACIONES (Marcar con una **X**):

Diagramas_X__ Fotografías___ Grabaciones en discos___ Ilustraciones en general__X_ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros_X_

SOFTWARE requerido y/o especializado para la lectura del documento: PDF

MATERIAL ANEXO: Sin Anexo

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

Español	Inglés	Español	Inglés
1. Algoritmo	Algorithm	6. Lógica clásica	Classical logic
2. Máquina de Turing	Turing Machine	7. Lógica cuántica	Quantum logic
3. Complejidad computacional	Computational complexity	8. Interdisciplinariedad	Interdisciplinarity
4. Determinismo	Determinism		
5. No determinismo	Non-determinism		

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

Este documento de tesis tiene el objetivo de sustentar por qué un problema de tipo no polinomial (NP), dentro de la lógica de la computación clásica, se puede convertir en un problema de tipo polinomial (P) dentro de la lógica de la computación cuántica. Para ello, se busca relacionar la complejidad computacional con la lógica cuántica.

La complejidad computacional está diseñada dentro de la lógica clásica, pero no hay todavía un diseño de la complejidad computacional dentro de la lógica cuántica, debido a que la lógica cuántica es una ciencia en pleno desarrollo, todavía no hay computadores cuánticos capaces de superar a los computadores clásicos, la supremacía cuántica por ahora es algorítmica y teórica.

Se desarrolla la complejidad computacional desde las Máquinas de Turing, con la lógica de la computación clásica, los algoritmos computacionales clásicos, el determinismo y la secuencialidad de los computadores clásicos. Las Máquinas de Turing son máquinas secuenciales, deterministas, y en la práctica son modelos teóricos de los computadores clásicos. Aunque existen Máquinas de Turing no deterministas a nivel teórico, en la práctica los computadores clásicos son deterministas.

Se desarrolla la lógica cuántica dentro de la computación cuántica, con algoritmos cuánticos, el no determinismo y la no secuencialidad. Por último, se comparan las características de la lógica clásica frente a la lógica cuántica, encontrándose que la supremacía cuántica se da por las características no deterministas que tiene la computación cuántica, especialmente el paralelismo, la superposición y el entrelazamiento cuántico.

ABSTRACT: (Máximo 250 palabras)

This thesis document has the objective of supporting why a problem of non-polynomial type (NP), within the logic of classical computing, can be converted into a problem of polynomial type (P) within the logic of quantum computing. For this, it seeks to relate computational complexity with quantum logic.

Computational complexity is designed within classical logic, but there is not yet a computational complexity design within quantum logic, because quantum logic is a science in full development, there are still no quantum computers capable of surpassing the classical computers, quantum supremacy for now is algorithmic and theoretical.

Computational complexity is developed from Turing Machines, with the logic of classical computation, classical computational algorithms, determinism and the sequentiality of classical computers. Turing Machines are sequential, deterministic machines, and in practice they are theoretical models of classical computers. Although there are non-deterministic Turing machines at the theoretical level, in practice classical computers are deterministic.

Quantum logic is developed within quantum computing, with quantum algorithms, non-determinism and non-sequentiality. Finally, the characteristics of classical logic are compared with quantum logic, finding that quantum supremacy is given by the non-deterministic characteristics of quantum computing, especially parallelism, superposition and quantum entanglement.

APROBACION DE LA TESIS

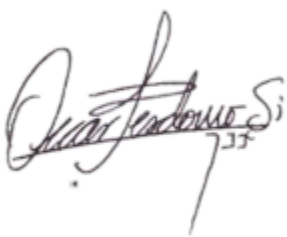
Nombre Presidente Jurado: (No hay)

Firma:

Nombre Jurado: MSc. Carlos Javier Martínez Moncaleano

Firma: 

Nombre Jurado: MSc. Oscar Iván Perdomo Sánchez

Firma: 

PROBLEMAS P Y NP EN LA COMPLEJIDAD COMPUTACIONAL

TESIS DE MAESTRÍA

LUIS ANGEL GONZALES ESQUIVEL

Asesor: Mauro Montealegre Cárdenas

UNIVERSIDAD SURCOLOMBIANA

MAESTRIA EN ESTUDIOS INTERDISCIPLINARIOS DE LA COMPLEJIDAD

NEIVA – HUILA

2021

PROBLEMAS P Y NP EN LA COMPLEJIDAD COMPUTACIONAL

TESIS DE MAESTRÍA

LUIS ANGEL GONZALES ESQUIVEL

Asesor: Mauro Montealegre Cárdenas

Disertación presentada para optar el título de

Magíster en Estudios Interdisciplinarios de la Complejidad

UNIVERSIDAD SURCOLOMBIANA

MAESTRIA EN ESTUDIOS INTERDISCIPLINARIOS DE LA COMPLEJIDAD

NEIVA – HUILA

2021

La simplicidad hace referencia a la ausencia (o casi) de complejidad. Etimológicamente, simplicidad significa «plegado una vez», mientras que complejidad significa «todo trenzado» (nótese que tanto «plic-» para pliegue como «plej-» para trenza derivan de la misma raíz indoeuropea plek).

Murray Gell-Mann, 2003.

Resumen

Este documento de tesis tiene el objetivo de sustentar por qué un problema de tipo no polinomial (NP), dentro de la lógica de la computación clásica, se puede convertir en un problema de tipo polinomial (P) dentro de la lógica de la computación cuántica. Para ello, se busca relacionar la complejidad computacional con la lógica cuántica; se sabe que la complejidad computacional trata de los problemas P vs NP, pero dentro de la lógica clásica.

Se sabe que la complejidad computacional está diseñada dentro de la lógica clásica, pero no hay todavía, un diseño de la complejidad computacional dentro de la lógica cuántica, debido a que la lógica cuántica es una ciencia en pleno desarrollo, todavía no hay computadores cuánticos capaces de superar a los computadores clásicos, la supremacía cuántica por ahora es algorítmica y teórica.

Es necesario desarrollar la complejidad computacional desde las Máquinas de Turing, con la lógica de la computación clásica, los algoritmos computacionales clásicos, el determinismo y la secuencialidad de los computadores clásicos. Se sabe que las Máquinas de Turing son máquinas secuenciales, deterministas, y en la práctica son modelos teóricos de los computadores clásicos. Aunque existen Máquinas de Turing no deterministas a nivel teórico, en la práctica los computadores clásicos son deterministas.

Se desarrolla la lógica cuántica dentro de la computación cuántica, con algoritmos cuánticos, el no determinismo y la no secuencialidad. Por último, se compara las características de la lógica clásica frente a la lógica cuántica, encontrándose que la supremacía cuántica se da por las características no deterministas que tiene la computación cuántica, especialmente el paralelismo, la superposición y el entrelazamiento cuántico. En la complejidad computacional clásica, se sabe que los problemas de tipo no polinomial se vuelven polinomiales con algoritmos no deterministas.

También es importante la interdisciplinariedad en la biología con el cerebro cuántico, la lingüística, las matemáticas, la física, las telecomunicaciones, la psicología educativa, la didáctica y la informática.

Palabras Claves

Algoritmo, Máquina de Turing, complejidad computacional, determinismo, no determinismo, lógica clásica, lógica cuántica, interdisciplinariedad.

Abstract

This thesis document has the objective of supporting why a problem of non-polynomial type (NP), within the logic of classical computing, can be converted into a problem of polynomial type (P) within the logic of quantum computing. For this, it seeks to relate computational complexity with quantum logic; computational complexity is known to deal with P vs NP problems, but within classical logic.

It is known that computational complexity is designed within classical logic, but there is not yet a design of computational complexity within quantum logic, because quantum logic is a science in full development, there are still no capable quantum computers to surpass classical computers, quantum supremacy for now is algorithmic and theoretical.

It is necessary to develop computational complexity from Turing Machines, with the logic of classical computation, classical computational algorithms, determinism and the sequentiality of classical computers. Turing Machines are known to be sequential, deterministic machines, and in practice they are theoretical models of classical computers. Although there are non-deterministic Turing Machines at the theoretical level, in practice classical computers are deterministic.

Quantum logic is developed within quantum computing, with quantum algorithms, non-determinism and non-sequentiality. Finally, the characteristics of classical logic are compared with quantum logic, finding that quantum supremacy is given by the non-deterministic characteristics of quantum computing, especially parallelism, superposition and quantum entanglement. In classical computational complexity, it is known that non-polynomial problems become polynomial with non-deterministic algorithms.

Interdisciplinarity is also important in biology with the quantum brain, linguistics, mathematics, physics, telecommunications, educational psychology, didactics, and computer science.

Keywords

Algorithm, Turing Machine, computational complexity, determinism, non-determinism, classical logic, quantum logic, interdisciplinarity.

Agradecimientos

Mis más sinceros agradecimientos al profesor y Dr. Mauro Montealegre Cárdenas, por su paciencia, constancia, insistencia y perseverancia para que yo terminara esta tesis, especialmente con el enfoque de la Teoría de la Complejidad. Y además, por haberme sugerido la bibliografía principal, que entrelazara los temas de esta tesis con la Teoría de la Complejidad.

Mis agradecimientos a mi familia y amigos, por su paciencia y porque me regalaron su apoyo y el tiempo necesario para no claudicar.

Índice general

1. Introducción **11**

2. Planteamiento del Problema de Investigación **13**
- 2.1. Descripción del Problema . . . 14
- 2.2. Sistematización del Problema . . . 19
- 2.3. Enunciación del Problema . . . 19

3. Antecedentes y Justificación **20**

4. Objetivos de la Investigación **25**
- 4.1. Objetivo General . . . 25
- 4.2. Objetivos Específicos . . . 25

5. Metodología **26**

6. Didáctica de los Problemas P vs NP **28**
- 6.1. Matrioshka (Muñecas rusas) . . . 29
- 6.2. Torres de Hanoi . . . 30

7. Máquinas de Turing **35**
- 7.1. Máquinas de Turing Deterministas . . . 35
- 7.2. Máquinas de Turing Multicintas . . . 41
- 7.3. Máquinas de Turing No Deterministas . . . 49

8. Computación Clásica **54**
- 8.1. Compuertas de la Lógica Clásica . . . 55
  - 8.1.1. Compuerta Si o Buffer . . . 55
  - 8.1.2. Compuerta No, Not, Inversor, $\neg$ . . . 56
  - 8.1.3. Compuerta AND, Y, $\wedge$ . . . 57
  - 8.1.4. Compuerta OR, O, $\vee$ . . . 58
  - 8.1.5. Compuerta XOR, Ô, $\veebar$ . . . 60
  - 8.1.6. Negación o complemento de compuertas . . . 62
  - 8.1.7. Suma de números binarios utilizando compuertas lógicas . . . 63

9. Complejidad **69**
9.1. ¿Qué es la Complejidad? . . . 69
9.2. Ciencias de la Complejidad . . . 72

10. Teoría Matemática de la Información **74**
10.1. Probabilidad frecuentista . . . 74
10.2. Probabilidad condicional . . . 75
10.3. Teorema de Bayes . . . 76
10.4. Problema de Monty Hall . . . 77
10.5. Teoría de la Información . . . 77
10.6. Entropía de Shannon . . . 80
10.6.1. Curva de Entropía Máxima de Shannon en un Evento Binario . . . 86

11. Complejidad Computacional **87**
11.1. Problemas de Clase P . . . 88
11.2. Problemas de Clase NP . . . 88
11.3. Problemas de Clase NP Completos . . . 89
11.4. Teoría Matemática de la Complejidad . . . 90
11.5. Complejidad de un Algoritmo . . . 90

12. Computación Cuántica **94**
12.1. Mecánica Cuántica . . . 95
12.2. Espacio Vectorial de Hilbert de Dimensión Finita en un Campo Escalar Complejo. . . . 95
12.3. Los Qubits . . . 97
12.4. El Qubit y sus Características . . . 99
12.5. Compuertas Cuánticas Sobre Un Qubits . . . 100
12.5.1. Compuerta Cuántica X o NOT . . . 100
12.5.2. Compuerta Cuántica U o Unidad. . . . 101
12.5.3. Compuerta Cuántica H o Hadamard. . . . 101
12.6. Compuertas Cuánticas Sobre Dos Qubits . . . 102
12.6.1. Compuerta Cuántica CNOT: Controlled Not. . . . 103
12.7. Reversibilidad de la Compuertas Cuánticas . . . 105
12.8. Paralelismo Cuántico . . . 105
12.9. Algoritmos Cuánticos . . . 105
12.9.1. Algoritmo Deutsch . . . 106

13. Análisis y Discusión de Resultados **108**
13.1. Conclusiones . . . 111

Bibliografía **113**

Índice de figuras

2.0.1. Mapa conceptual de la tesis, realizado con el software libre CmapTools. . . 13
2.1.1. Gato de Schrödinger (Europa Press, 2019) 16
2.1.2. Interdisciplinariedad en la complejidad computacional vs lógica cuántica, realizado con el software libre CmapTools. 18

3.0.1. Mapa conceptual de los Antecedentes y Justificación, realizado con el software libre CmapTools. 20

6.1.1. Muñeca rusa en situación inicial (Khan Academy, 2021). 30
6.1.2. Muñeca rusa en situación final (Khan Academy, 2021). 30
6.2.1. Torres de Hanoi de tres discos, posición inicial (Uterra.com, 2021). 31
6.2.2. Torres de Hanoi de tres discos, primer movimiento (Uterra.com, 2021). . . 31
6.2.3. Torres de Hanoi de tres discos, segundo movimiento (Uterra.com, 2021). . 31
6.2.4. Torres de Hanoi de tres discos, tercer movimiento (Uterra.com, 2021). . . 32
6.2.5. Torres de Hanoi de tres discos, cuarto movimiento (Uterra.com, 2021). . . 32
6.2.6. Torres de Hanoi de tres discos, quinto movimiento (Uterra.com, 2021). . . 32
6.2.7. Torres de Hanoi de tres discos, sexto movimiento (Uterra.com, 2021). . . . 33
6.2.8. Torres de Hanoi de tres discos, séptimo y último movimiento (Uterra.com, 2021). 33

7.1.1. Máquina de Turing que reconoce o decide si una cadena de Os es una potencia de 2 (Sipser, 2006). 36
7.1.2. Ejemplo de ingreso de cadena de la Máquina de Turing, con el programa JFLAP. 38
7.1.3. Resultado de ingresar la cadena OOOOOOOOOOOOOO, con el programa JFLAP 38
7.1.4. Representación del complemento a binario en una Máquina de Turing, bajo el programa JFLAP. 39
7.1.5. Representación del complemento a binario en una Máquina de Turing, con el programa JFLAP, para una cadena determinada. 40
7.1.6. Máquina de Turing para marcar un número binario par o impar. 41
7.2.1. Máquina de Turing en el programa JFLAP. 42
7.2.2. Máquina de Turing que recibe una cadena de unos y los cuenta. 44
7.2.3. Ingreso de la cadena de entrada. 45
7.2.4. Resultado de contar dentro de una cadena 24 unos. 45
7.2.5. Estados 0 y 1. 46

7.2.6. Estados 1, 2 y 3. 46
7.2.7. Estados 3 y 4. 47
7.2.8. Estados 9 y 0. 48
7.2.9. Estado 10. 49
7.3.1. Máquina no determinista para determinar cuándo los subconjuntos de un conjunto de tres elementos suman cero. 50
7.3.2. Cálculos deterministas y no deterministas con una rama de aceptación (Sipser, 2006) 51
7.3.3. Máquina de Turing No Determinista, para reconocer un lenguaje específico. 53

8.1.1. Circuito eléctrico abierto, para la compuerta «Si». 56
8.1.2. Circuito eléctrico cerrado, para la compuerta «Si». 56
8.1.3. Representación gráfica de la compuerta «No». 56
8.1.4. Símbolo gráfico de la compuerta «No». 57
8.1.5. Circuito eléctrico para la compuerta «Y». 57
8.1.6. Representación gráfica de la compuerta «Y». 58
8.1.7. Símbolo gráfico de la compuerta «Y». 58
8.1.8. Circuito eléctrico cerrado, para la compuerta «O». 59
8.1.9. Circuito eléctrico abierto, para la compuerta «O». 59
8.1.10. Representación gráfica para la compuerta de la «O». 60
8.1.11. Símbolo gráfico de la compuerta «O». 60
8.1.12. Circuito eléctrico abierto, para la compuerta de la «O exclusiva». 61
8.1.13. Circuito eléctrico cerrado, para la compuerta de la «O exclusiva». 61
8.1.14. Representación gráfica para la compuerta de la «O exclusiva». 62
8.1.15. Símbolo gráfico de la compuerta «O exclusiva». 62
8.1.16. Símbolo gráfico de la compuerta «NOT». 62
8.1.17. Símbolo gráfico de la compuerta «NAND». 63
8.1.18. Símbolo gráfico de la compuerta «NOR». 63
8.1.19. Símbolo gráfico de la compuerta «XNOR». 63
8.1.20. Sumador de números binarios utilizando compuertas lógicas. 64
8.1.21. Circuito para la suma de los números binarios 1 + 1, con acarreo de entrada 0. 65
8.1.22. Circuito para la suma de los números binarios 0 + 1, con acarreo de entrada 1. 66
8.1.23. Circuito para la suma de los números binarios 1 + 0, con acarreo de entrada 1. 67

10.2.1. Probabilidad Condicional. 75
10.4.1. Tres puertas para ambientar el experimento de Monty Hall 77
10.5.1. Diagrama de comunicación de Shannon en la versión de Weaver (1948 a), citado por Seising (2012). 78
10.5.2. Diagrama de Shannon con la adición de Weaver de las dos casillas "Receptor semántico" y "Ruido semántico", en la versión de Seising (2012). . 80
10.6.1. Incertidumbre mínima en un bits de información 84
10.6.2. Curva de entropía máxima para un bits de información (López, 2002). . . 86

12.4.1. Gráfica bidimensional del Qubit. (Interpretación propia). 100

13.1.1. Mapa conceptual de las conclusiones, realizado con el software libre CmapTools. 111

Índice de cuadros

8.1. Tabla de verdad para la compuerta «Si». 56
8.2. Tabla de verdad para la compuerta «No». 56
8.3. Tabla de verdad para la compuerta «Y». 57
8.4. Tabla de verdad para la compuerta «O» 60
8.5. Tabla de verdad para la compuerta de la «O exclusiva». 61

11.1. Funciones de referencia (Mañas, 1997) 92

Capítulo 1

Introducción

En el primer capítulo llamado «Didáctica de los Problemas P vs NP», se comienza con una idea muy general sobre el complejo proceso de enseñanza-aprendizaje, luego se expone de manera didáctica el concepto de recursividad, utilizando la famosa muñeca rusa conocida como Matrioshka; y por último, mediante el juego de las Torres de Hanoi, se expone el concepto intuitivo de problema Polinomial (P), problema No Polinomial (NP) y problema NP-Completo.

En el siguiente capítulo, llamado «Máquinas de Turing», la idea es conocer el concepto de Máquina de Turing, su historia y la importancia dentro del desarrollo de la computación. Se da un concepto intuitivo y una definición matemática, explicándose cada uno de sus componentes. Se exponen algunos ejemplos de Máquinas de Turing de una sola cinta y también multicintas, para algunos problemas computacionales. Por último, se expone el concepto de Máquinas de Turing no deterministas, y se explican algunos ejemplos. El concepto de computabilidad, algoritmo y no determinismo es quizá difícil de entender sino se explica dentro del marco de las Máquinas de Turing, ya que estas máquinas son el modelo computacional sobre el que se apoya toda la informática, y esto se apoya en la tesis de Church - Turing: «Si la Máquina de Turing no puede computar un problema, ningún otro ordenador puede computarlo, ya que no existe algoritmo que lo resuelva» (Cortez, 2004).

El siguiente capítulo, llamado «Computación Clásica», explica la lógica de la computación clásica, las compuertas lógicas mediante circuitos lógicos y la importancia de algunos elementos electrónicos, especialmente el transistor, en el desarrollo de estos circuitos. Luego, se explican algunos circuitos lógicos para resolver sumas de números binarios, utilizando las compuertas lógicas, de la misma manera como se ejecuta en la unidad aritmética lógica de la unidad central de procesamiento de un computador actual. Este capítulo es fundamental para poder explicar mejor el capítulo que sigue. Ya que, si no conocemos la lógica de la computación clásica, será muy complicado entender la lógica de la computación cuántica.

En el siguiente capítulo llamado «Complejidad», se exponen las dos vertientes principales que estudian el concepto de complejidad, sin haber una definición precisa hasta el momento, ya que según los autores que se estudian, la complejidad no se puede reducir a una definición. Las dos vertientes que se estudian son «El Pensamiento Complejo» y las «Ciencias de

la Complejidad», desde sus diferentes precursores llegándose a un concepto que nos brinden las características más sobresalientes de los sistemas complejos. Este capítulo es fundamental dentro de este documento, ya que los problemas P vs NP y la lógica cuántica son dos ejes principales de las Ciencias de la Complejidad (Maldonado, 2010).

Sigue el capítulo «Teoría Matemática de la Información», en este capítulo se aborda el concepto de probabilidad y algunos tipos de probabilidad más comunes, luego se exponen los sistemas de comunicación de Shannon y los de Weaver, abordando el concepto de ruido y semántica, dejándose una expectativa hacia la lógica difusa. Por último, se expone la entropía de Shannon en términos probabilísticos, con el objetivo de hacer un tratamiento matemático de la información y exponer las diferentes aplicaciones de la entropía informática. Es muy particular la explicación de cómo se obtiene la ecuación para la entropía de Shannon.

El capítulo que sigue se llama «Complejidad Computacional», aquí la complejidad se aborda desde el punto de vista del tiempo de ejecución de algoritmos, está muy relacionado con la métrica de software, pero no en el tiempo de ejecución del lenguaje de programación sino en el tiempo de ejecución del algoritmo. Se exponen los conceptos de problemas P, NP y NP-Completos, ya que son ejemplos donde se tiene en cuenta el orden de complejidad de los algoritmos y se introduce el concepto de algoritmos no deterministas. De nuevo se aborda el concepto de Máquinas de Turing deterministas y no deterministas, especialmente porque los algoritmos se ejecutan en estas máquinas teóricas.

Por último, se expone el capítulo llamado «Computación Cuántica», se comienza con un estado del arte de la computación cuántica y con algo de historia. Luego, se expone el concepto de qubits desde el punto de vista de los espacios vectoriales de Hilbert en dos dimensiones, sobre un campo escalar complejo, con el objetivo de mostrar la computación cuántica, como una teoría científica de sustento matemático. Luego se exponen los principios básicos de la física cuántica en los que se basa la computación cuántica. Sigue una presentación de las compuertas cuánticas como matrices de números complejos, que hacen transformaciones a los vectores llamados qubits, respetando los principios de la física cuántica, ya que los qubits representan estados cuánticos y geométricos probabilísticos de partículas físicas a nivel atómico. También se expone la reversibilidad de las compuertas cuánticas, propiedad matemática que no tienen todas las compuertas lógicas de la computación clásica. Luego se habla de paralelismo cuántico, es una capacidad muy común en procesos no determinísticos, ya que un solo algoritmo cuántico puede ejecutar procesos simultáneos. Por último, se habla de algoritmos cuánticos, con un poco de historia desde la lógica matemática, la computación clásica y las Máquinas de Turing. Se termina el capítulo con una de las conclusiones más importantes sobre la supremacía cuántica en cuanto a los problemas P vs NP, que es lo mismo que la supremacía de la lógica cuántica sobre la lógica clásica.

Capítulo 2

Planteamiento del Problema de Investigación

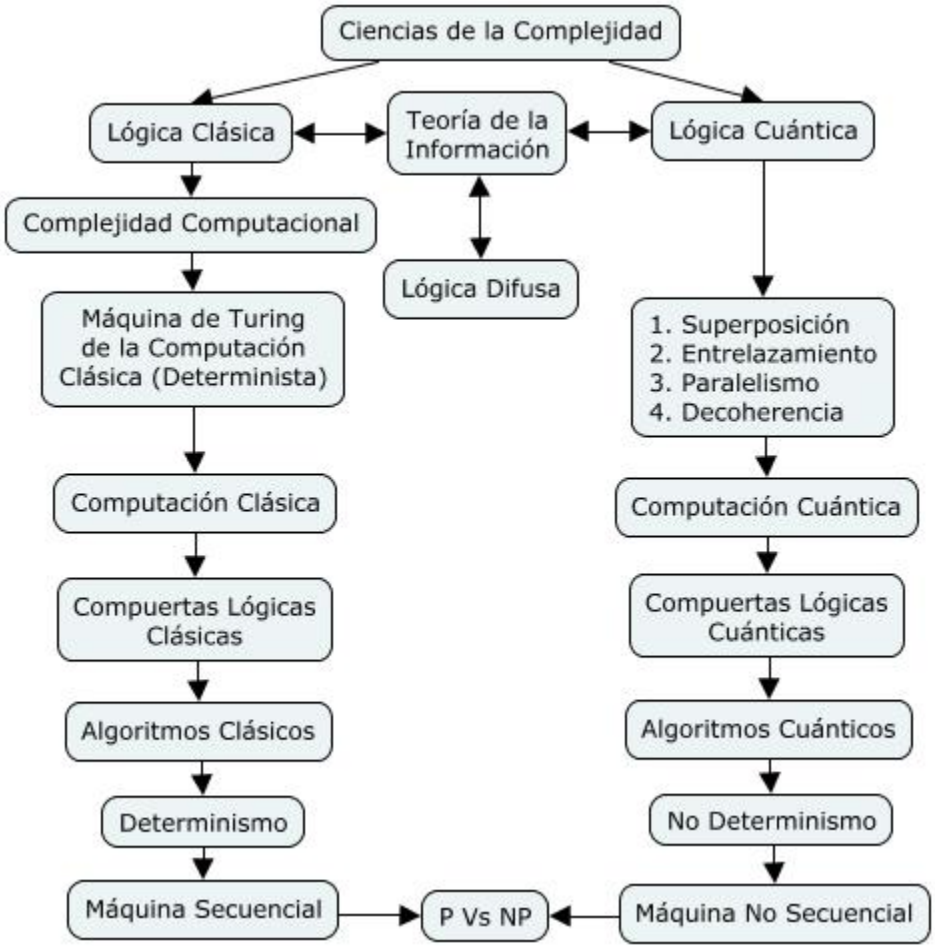


Figura 2.0.1: Mapa conceptual de la tesis, realizado con el software libre CmapTools.

2.1. Descripción del Problema

Las ciencias de la complejidad estudian tres ejes fundamentales: los problemas P vs NP, la lógica cuántica y la teoría de los sistemas dinámicos (Maldonado, 2010). El primer eje está dentro de la complejidad computacional, el segundo eje hace uso de los principios de la física cuántica, y el tercer eje no será tratado en este documento. La complejidad computacional está abordada desde la lógica clásica, y estudia la relación entre algoritmo, espacio en memoria, tiempo de ejecución, comprender qué es una Máquina de Turing, computabilidad, teorema de Gödel, y la relación entre razonamiento humano, funcionamiento biológico del cerebro, e inteligencia artificial, porque lo que se busca es simular procesos de pensamiento mediante algoritmos, entender la estructura de un lenguaje, de una lingüística, todo ello para resolver problemas imitando el comportamiento del cerebro humano, utilizando la inteligencia artificial. «Para que la inteligencia artificial funcione una computadora debe ser capaz de reconocer y aprender nuevos patrones» (Ciencia plus, 2021).

Se hace necesario entrelazar el eje de la complejidad computacional y el eje de la lógica cuántica, interrelacionarlos de alguna manera, para determinar cómo aportan a las ciencias de la complejidad; es por ello que, en este trabajo se empieza a investigar la definición formal de algoritmo, porque en la teoría de la complejidad computacional, los problemas se resuelven modelando con algoritmos. Dado que la tesis de Church-Turing, nos da una definición formal de algoritmo, se observa la necesidad de estudiar las Máquinas de Turing, porque es en las Máquinas de Turing donde se resuelven o ejecutan los algoritmos. Hay una relación tan estrecha entre algoritmo y Máquina de Turing que, Sipser (2006) nos presenta a los algoritmos como conjunto de instrucciones que se pueden resolver por medio de una Máquina de Turing.

Entendemos por Máquina de Turing, a una estructura matemática que hace operaciones, específicamente ejecuta algoritmos. La estructura «teórica» recibe un dato de entrada a través de una cinta, lo lee, luego escribe otro dato o respuesta en la misma cinta, y por último toma la decisión de parar, moverse a la derecha o moverse a la izquierda de la cinta, para de esa manera terminar su ejecución o volver a comenzar de nuevo.

Dado que las Máquinas de Turing son modelos teóricos de los computadores actuales, y que los computadores actuales funcionan bajo la lógica clásica, que es la misma lógica binaria, de verdadero y falso, booleana, nos interesa estudiar las compuertas de la lógica clásica, las cuales se utilizan para construir algoritmos en la computación clásica. Bajo estos algoritmos se abordan los problemas P vs NP en la computación clásica.

Las siguientes relaciones: ¿P = NP?, ¿P $\neq$ NP?, ¿NP $\geq$ P?, ¿P $\in$ NP?, no han podido ser resueltas hasta la fecha, y son el problema fundamental de la complejidad computacional, a la vez que hacen parte de los problemas del milenio (Maldonado, 2010).

Con respecto a los problemas P vs NP, se hace un estudio detallado de su definición, ya que a simple vista o lectura nunca queda clara su definición. Se entiende por un problema Polinomial (P), a un problema que se soluciona en un tiempo razonable, un tiempo tratable, un tiempo que una persona puede esperar sin problema; este tiempo se representa a través

de funciones polinómicas, por ejemplo, cuadráticas o cúbicas. Se entiende por un problema No Polinomial (NP), a un problema que sí se puede resolver, pero, el tiempo que tarda en resolverse es demasiado grande, por ejemplo, descomponer un número de mil dígitos en sus factores primos, puede tardar $4{,}25x10^{25}$ años (Vélez, 2001), resolver las torres de Hanoi para 64 discos tarda 584.942´417.355 años, moviendo un solo disco por segundo (ver capítulo 6). Es de aclarar que los problemas NP se pueden resolver en tiempos polinómicos o razonables, pero con algoritmos no determinísticos, y por ahora nuestros computadores clásicos son determinísticos. No se puede realizar más de una operación en una Máquina de Turing, es una máquina secuencial que hace lo que tiene que hacer (Arjona, 2007). Por último, se consideran los problemas No Polinomiales Completos (NP-Completos), estos problemas no se saben si tienen solución en tiempo polinomial, son problemas equivalentes a problemas decidibles, o sea problemas cuya respuesta es «Si» o «No», pero, por ahora no se sabe si tienen solución en tiempo polinomial. Lo que sí se sabe, es que todos estos problemas NP-Completos son equivalentes, o sea que, si pudiéramos resolver uno solo en tiempo polinomial, con ese mismo algoritmo resolveríamos todos (Arjona, 2007).

Según Penrose (1996), el problema más importante de la teoría de la complejidad, es determinar si NP = P, o sea, resolver mediante una Máquina de Turing determinista un problema no polinomial en un tiempo polinomial. Por ello, para Penrose (1996), la teoría de la complejidad trata de la complejidad computacional.

Para Deutsch (1985), citado por Penrose (1996), en algún momento se construirá un dispositivo cuántico que mejore el rendimiento de una Máquina de Turing. Esto quiere decir que, en la lógica cuántica, un problema de tipo No Polinomial podría ser resuelto en tiempo polinomial, demostrando en esa lógica que NP = P. Actualmente existen algoritmos cuánticos que reducen la complejidad de los algoritmos clásicos, por ejemplo: algoritmo de Deutsch-Jozsa, algoritmo de Shor y algoritmo de Grover. Sin embargo, aún no existe una computadora cuántica, con la capacidad de ejecutar estos algoritmos de manera más eficiente que una computadora clásica ejecuta sus algoritmos clásicos. O sea que, la computación cuántica es una ciencia en pleno desarrollo, tanto en el hardware, el software y sus estructuras algorítmicas (García, 2020).

La tesis de Deutsch (1985), citado por Penrose (1996), tiene mucho sentido también en matemáticas, ya que se explica en el teorema de Gödel, «un sistema matemático formal, debe contener algunos enunciados que no son demostrables ni indemostrables con los medios permitidos dentro del sistema», ello justificaría que en la lógica clásica existan problemas de tipo No Polinomial (NP), y que para poderlos resolver o tratar, se debe cambiar de sistema matemático formal, es decir, cambiarse de la lógica clásica de la computación clásica, a otro tipo de lógica, para nuestro caso sería a la lógica cuántica de la computación cuántica. De esta manera, podríamos resolver el problema P vs NP.

La tesis de Deutsch (1985), citado por Penrose (1996), también tiene sentido en biología, ya que la inteligencia artificial busca imitar el comportamiento del cerebro humano, pero la lógica que se aborda es clásica, de la computación clásica, sin embargo, ya se conoce que «el comportamiento del cerebro humano es cuántico» (Ciencia plus, 2021), por lo tanto, para

imitar el comportamiento del cerebro humano se necesita de una lógica cuántica a través de la computación cuántica. Surgiendo de esta manera la inteligencia artificial cuántica.

La aplicación de la lógica cuántica en la física cuántica es más que evidente, ya que surge de sus principios cuánticos.

La aplicación de la lógica cuántica en la informática, se estudia con cierta profundidad en este documento.

Figura 2.1.1: Gato de Schrödinger (Europa Press, 2019)

Erwin Schrödinger (1935), citado por Penrose (1996), quería mostrar lo absurdo de la mecánica cuántica, para ello ideó un experimento mental que más tarde se llamó el gato de Schrödinger, el cual consistía en meter un gato dentro de una habitación totalmente cerrada y vacía, en cuyo interior habría un átomo radiactivo que se desintegra cuánticamente, donde su función de onda incluye ambos estados, desintegrado y no desintegrado, esto quiere decir que el átomo tiene la probabilidad de 1/2 de estar desintegrado y la probabilidad de 1/2 de no estarlo. Si el átomo se desintegra se activa un mecanismo que expande un gas venenoso y el gato muere, y en caso contrario no pasaría nada y el gato vive, ver figura 2.1.1. Como la vida del gato depende del estado cuántico del átomo, esto quiere decir que en la medida en que el estado del átomo esté en superposición, es decir que el átomo está 50 % desintegrado y 50 % no desintegrado, en esa medida, el gato estará 50 % vivo y 50 % muerto, es decir el gato estaría vivo y muerto a la vez. Sólo cuando se rompe la coherencia cuántica, es decir cuando se interrumpe el aislamiento del sistema cuántico, es cuando el átomo toma el estado desintegrado o no, y por consiguiente el gato toma el estado de vivo o muerto. Esta interpretación choca con la intuición humana y rompe con nuestros estándares de la realidad. Sin embargo, hay otra interpretación que raya en las fronteras de la filosofía y la psicología, el papel de la conciencia, ¿el observador rompe la armonía del sistema al ser consciente del estado en que se encuentra el sistema? Por otro lado, hay otras interpretaciones en que se cree que el gato entra en dos universos paralelos, uno en el que está vivo y otro en el que está muerto, esta interpretación es muy científica para algunos, pero para otros es metafísica, un reto para la mente, la filosofía y la psicología. Una posible solución al experimento mental del gato de Schrödinger, es que este experimento es imposible de llevar a cabo, porque el gato es un

sistema compuesto de muchas partículas, y la interacción entre dichas partículas rompería la coherencia cuántica, es decir que el gato no podría comportarse de manera cuántica como lo haría un solo átomo; en otras palabras, el gato no podría estar en estado de superposición, de vivo y muerto a la vez, la interacción del sistema complejo del gato también rompería la coherencia cuántica del átomo radiactivo, por lo tanto, veríamos un experimento no cuántico, donde el gato moriría si el átomo se desintegra y viviría si el átomo no se desintegra, y el átomo radiactivo no podría estar desintegrado y no desintegrado a la vez. Aunque al parecer se haya resuelto esta paradoja, no se puede olvidar que es un experimento mental, y está lleno de suposiciones, tan complejo que aún origina complejas disertaciones en la física, la filosofía y la psicología.

Desde mi punto de vista, entendiendo que la física cuántica es una ciencia muy difusa, que aún no está terminada ni estudiada completamente, se me ocurre pensar en un experimento social, supongamos que hay una lotería de mil números o posibilidades, y que mil personas compran un boleto, antes de que se juegue la lotería, todos los jugadores son ganadores, todos son potencialmente ganadores, todos podrían ganar la lotería, y se produce un efecto psicológico muy placentero en los apostadores, el efecto de poderse ganar la lotería; este efecto se conoce como superposición cuántica, donde todos los estados son posibles, hay una coherencia entre todos los estados, porque están en sincronía, ningún jugador podría considerar que ya perdió. ¿En qué momento se pierde la coherencia?, o sea, ¿en qué momento los estados (jugadores) entran en decoherencia? ¿en qué momento los estados (jugadores) pueden dejar de considerar que pueden ganar?, la respuesta es obvia, el sistema colapsa, o se decide por un resultado, cuando la lotería juega, así un jugador jamás se entere del resultado del número ganador, tal vez él siga pensando que puede ganar, pero si no es el ganador, nunca lo podrá ser, igual pasa si una persona gana la lotería y no la reclama, igual él es el ganador.

Hay una aplicación de la lógica cuántica muy evidente para los científicos informáticos, y que muchas personas de otras áreas desconocen, es la lingüística, ya que los computadores trabajan con lenguajes, gramáticas y alfabetos. En este orden de ideas, la lógica cuántica estudiaría una lógica difusa dentro del cerebro humano, de ¿cómo el cerebro entiende la semántica? ¿cómo se representa el lenguaje en el cerebro humano? ¿qué relación hay entre significado y significante? ¿cómo a través de un estímulo se forma una imagen, se elabora un concepto y luego, se comunica dicho concepto? Sin embargo, en un cerebro cuántico humano hay otras funciones del lenguaje y psicológicas: existe la memoria, el pensamiento, la inteligencia, la autodeterminación, la espiritualidad, las pulsiones básicas, los trastornos de la personalidad, y un sin fin de universos por explorar. "Estamos en un estado en el que podemos comenzar a relacionar la física fundamental con conceptos en biología, como la memoria y el aprendizaje" (Ciencia plus, 2021).

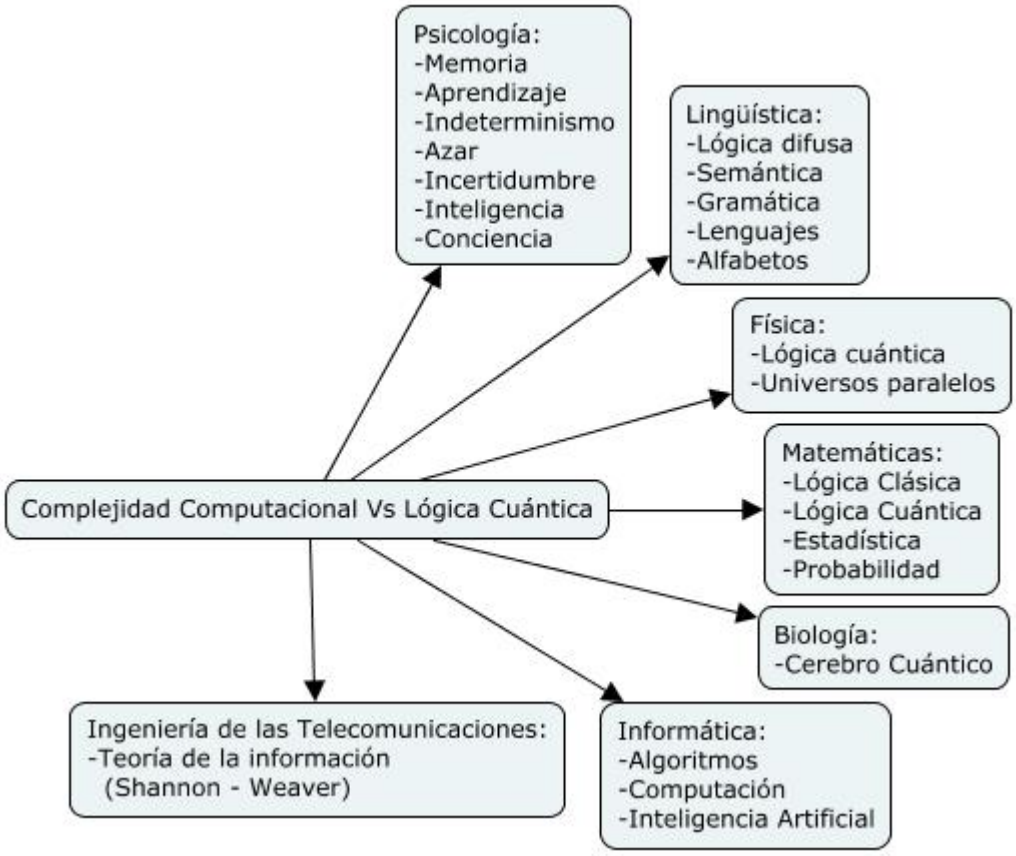


Figura 2.1.2: Interdisciplinariedad en la complejidad computacional vs lógica cuántica, realizado con el software libre CmapTools.

Se sabe que la física clásica y la lógica clásica son deterministas, entendiendo el determinismo como que, «el conocimiento exacto de las leyes del movimiento y de la configuración del universo en un momento dado, permitiría en principio, la predicción de la historia completa de éste» (Gell-Mann, 2003). Sin embargo, el universo es mecanocuántico, esto quiere decir que nuestro universo no es determinista; en palabras de Gell-Mann (2003), el no determinismo del universo implica que, «aun conociendo su estado inicial y las leyes fundamentales de la materia, sólo puede calcularse un conjunto de probabilidades para las diferentes historias posibles». Inclusive, la teoría del caos, regida por leyes y ecuaciones, supone que podamos predecir comportamientos futuros, sin embargo, se sabe que nada puede medirse con absoluta precisión, por lo tanto, una pequeñísima imprecisión en las condiciones iniciales, daría una grandísima diferencia en los resultados finales, a esta indeterminación se le llama no determinismo caótico, y se superpone al no determinismo cuántico, originando campos de la ciencia fascinantes y poco estudiados (Gell-Mann, 2003). En esta investigación de tesis, se estudia sólo el indeterminismo cuántico y no se estudia el indeterminismo caótico, aquí se busca relacionar la complejidad computacional con la lógica cuántica, no considerando los sistemas dinámicos. Relacionar el no determinismo cuántico con el no determinismo caótico, es un tema muy desconocido y de frontera que escapa a las posibilidades de este documento de tesis.

2.2. Sistematización del Problema

Avanzar en la solución de un problema a través de un algoritmo, no es trabajo de un computador o una máquina, es un problema que primero se resuelve en la mente humana, y luego sí se modela a través de una serie de pasos en un lenguaje de máquina para que lo resuelva un computador, por ello, se llega a la necesidad de investigar la injerencia de la lógica cuántica en los problemas no polinomiales o irresolubles de la lógica clásica. Para llevar a cabo esta investigación es necesario abordar las siguientes preguntas: ¿cuál es el concepto de Máquinas de Turing, computabilidad y algoritmo?, ¿qué es la complejidad computacional? y, ¿cómo se interrelaciona y se diferencia la lógica cuántica y la lógica clásica?

2.3. Enunciación del Problema

Se plantea el siguiente problema de investigación:

¿Por qué un problema de tipo no polinomial (NP), dentro de la lógica de la computación clásica, se puede convertir en un problema de tipo polinomial (P), dentro de la lógica de la computación cuántica?

Capítulo 3

Antecedentes y Justificación

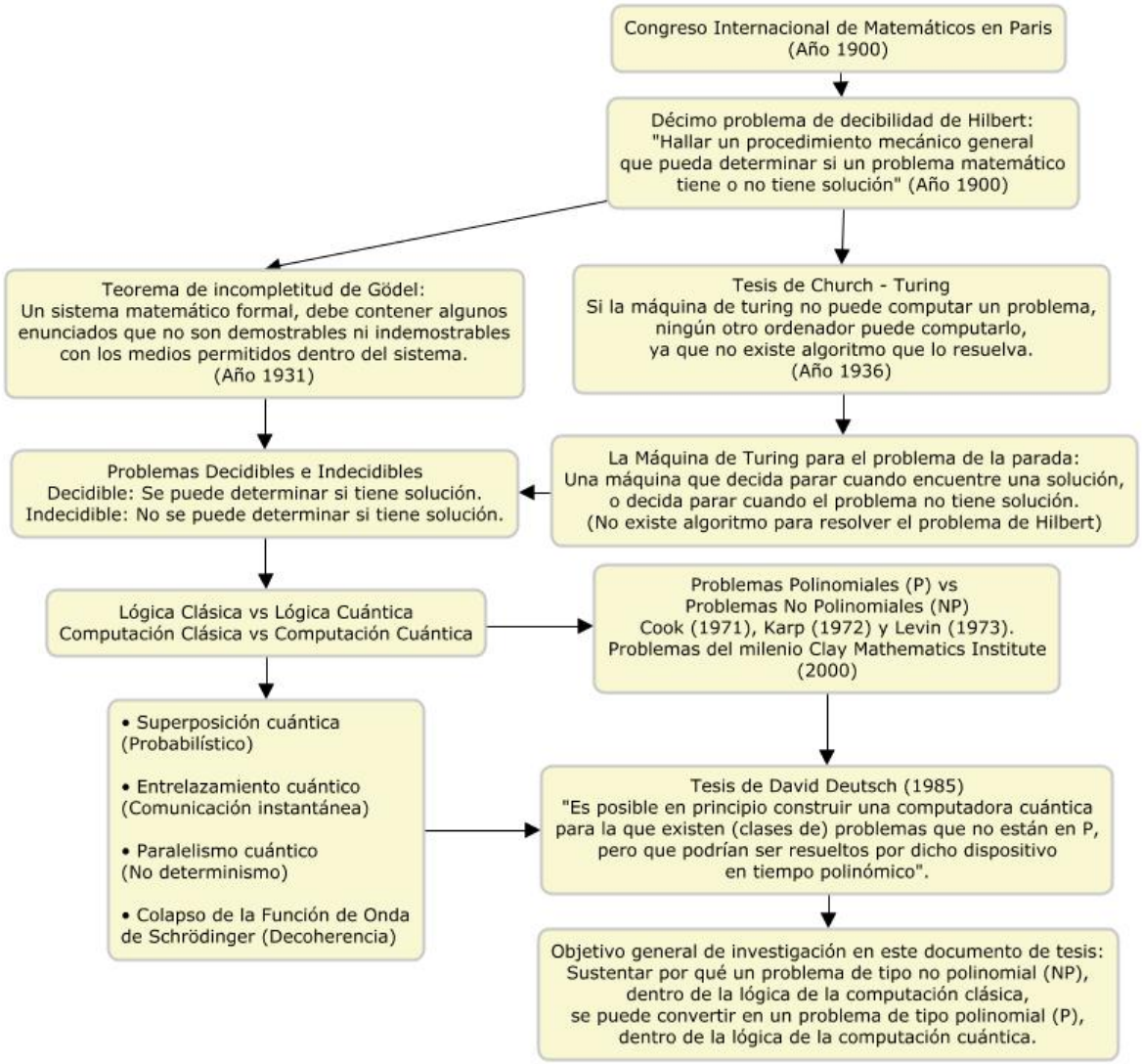


Figura 3.0.1: Mapa conceptual de los Antecedentes y Justificación, realizado con el software libre CmapTools.

Este trabajo se basa en uno de los problemas del milenio, cuyo premio para resolverlo es de un millón de dólares, sin contar la fama y los premios que han fijado algunas empresas que lideran el emporio económico del internet, como Google. En la página http://www.claymath.org/millennium-problems, podemos leer cada uno de dichos problemas, sin embargo, el problema que nos compete es el P vs NP:

«El problema de P versus NP es determinar si cada lenguaje aceptado por algún algoritmo no determinista en tiempo polinomial también es aceptado por algún algoritmo (determinista) en tiempo polinomial» (Cook, 2006).

Lo anterior quiere decir que, si los problemas No Polinomiales (NP) también se pueden convertir en problemas de tipo Polinomial (P), es decir que, si un problema No Polinomial se puede resolver mediante un algoritmo determinista en un tiempo razonable. En conclusión, lo que se pregunta es que si NP = P. Se sabe que la mayoría de los matemáticos sospechan que NP $\neq$ P, pero nadie ha sido capaz de demostrar dicha suposición. Quien sea capaz de demostrar este problema, será el afortunado ganador de un millón de dólares por parte del «Clay Mathematics Institute», (CMI), sin embargo, se debe cumplir las siguientes reglas:

1. CMI no acepta la presentación directa de propuestas de soluciones
2. El documento es una declaración completa de las reglas y procedimientos: CMI no ofrecerá más orientación o consejo
3. Antes de que CMI considere una solución propuesta, se deben cumplir las tres condiciones siguientes: (i) la solución propuesta debe publicarse en un punto de venta calificado, y (ii) deben haber transcurrido al menos dos años desde la publicación, y (iii) la solución propuesta debe haber recibido la aceptación general en la comunidad matemática mundial

(Clay Mathematics Institute, 2018).

Los informáticos teóricos saben que este tipo de problemas se trabajan en un modelo formal de computadora, porque se trata de representar algoritmos, lenguajes formales y funciones computables, «el modelo informático estándar en la teoría de la computabilidad es la Máquina de Turing» (Cook, 2006). Esto quiere decir que, aunque una Máquina de Turing, no es una máquina física, sí es un modelo teórico de cómo un computador actual ejecuta lógicamente un algoritmo, más adelante en este documento de tesis, se explicarán las Máquinas de Turing. Por lo anterior, para estudiar los problemas no polinomiales, es necesario estudiar las Máquinas de Turing, la Lógica Computacional Clásica y la Complejidad Computacional, los cuales son capítulos muy importantes en este documento de tesis.

Las Máquinas de Turing, y el concepto formal de algoritmo, tienen su origen en el año de 1936, donde Alan Turing, un matemático inglés, científico de la computación, trata de resolver el décimo problema de Hilbert, planteado en el Congreso Internacional de Matemáticos en París, en el año de 1900 por el famoso Matemático David Hilbert (Penrose, 1996). Dicho problema reza: «Hallar un proceso según el cual pueda ser determinado por un número finito de acciones, si un polinomio con coeficientes de números enteros tiene soluciones

enteras» (Sipser, 2006). Por lo que Alan Turing, lo interpreta como un problema de decisión, en su artículo «On computable numbers with an application to the entscheidnungsproblem» (Turing, 1936, citado por Penrose, 1996), que traduce «Sobre números computables con una aplicación al problema de decisión».

El problema planteado por Hilbert, es equivalente a: hallar un procedimiento mecánico general que pueda determinar si un problema matemático tiene o no tiene solución. EL término «procedimiento mecánico» fue interpretado como «algoritmo», sin embargo, no había una definición precisa de algoritmo, se podían crear algoritmos para ciertas tareas, pero era difícil demostrar que no existía un algoritmo para una tarea en particular. El progreso en el décimo problema de Hilbert tuvo que esperar a las investigaciones de Alonzo Church y Alan Turing en 1936, sobre una definición precisa de algoritmo. La tesis de Church-Turing proporciona la definición de algoritmo como un conjunto de instrucciones computables, es decir, que se resuelven por una Máquina de Turing. Luego, en 1970, el matemático Yuri Matiyasevich, basado en los trabajos de Julia Robinson, Martin Davis y Hilary Putnam, demostró que no existía un algoritmo que pudiera resolver este problema, por ello este problema fue considerado indecidible, es decir que, para este caso, no se puede determinar un algoritmo que decida si el problema tiene solución o no (Sipser, 2006).

Turing desarrolló originalmente sus ideas, con el objetivo de solucionar el muy general problema de decisión de Hilbert, el cual es equivalente a sistematizar algorítmicamente a la matemática; en términos generales Hilbert está preguntando: ¿existe algún procedimiento mecánico para responder a todas las cuestiones matemáticas dentro de un amplio, pero bien definido marco? Ante esta pregunta, Turing demostró junto con Church, que no puede haber un algoritmo general para decidir cuestiones matemáticas, en términos generales, la insolubilidad algorítmica de familias de problemas. Para este tipo de problemas, Turing planteó una máquina que decida parar cuando encuentre una solución, o decida parar cuando el problema no tiene solución, a esto se le llamó el problema de la detención o de la parada, y se determinó que es imposible resolver el problema de la parada por alguna Máquina de Turing, lo que significa que no se puede resolver algorítmicamente (Penrose, 1996).

El programa de Hilbert para sistematizar algorítmicamente a la matemática, fue primeramente truncado por los teoremas de incompletitud de Gödel en 1931, los cuales establecen que cualquier sistema matemático formal de axiomas y reglas de inferencia, que esté libre de contradicción, contiene enunciados que no son demostrables ni indemostrables con los medios permitidos dentro del sistema. El segundo teorema derivado del anterior, establece que, si el sistema matemático no conlleva a contradicciones, esto tampoco se puede demostrar dentro del mismo sistema (Penrose, 1996).

Gödel, Turing y Church, nos llevan a los conceptos de la decidibilidad e indecidibilidad. Los problemas indecidibles o no computables, son enunciados que no se puede demostrar o refutar, e inclusive no podemos determinar si tienen solución, lo único que si tenemos claro es que no existe un algoritmo que resuelva dichos problemas. Los problemas decidibles o computables, son los enunciados para los cuales si existen algoritmos que los resuelve.

Se sabe que una sentencia o enunciado, se llama independiente o indecidible en un sistema matemático formal, si este enunciado no se puede demostrar ni refutar dentro del conjunto de reglas del sistema matemático. Es por ello que, al analizar los problemas de tipo No Polinomiales, se observa que, aunque algunos son decidibles, pero intratables, o de una complejidad en tiempo exorbitante, podrían ser resueltos en lógicas diferentes a la lógica clásica, esto se deduce de la tesis de Deutsch (1985), citado por Penrose (1996):

«Es posible en principio construir una computadora cuántica para la que existen (clases de) problemas que no están en P, pero que podrían ser resueltos por dicho dispositivo en tiempo polinómico... subsiste la posibilidad teórica de que un dispositivo físico cuántico mejoraría una máquina de Turing».

Por la tesis de Deutsch (1985), citado por Penrose (1996), es que se establece el siguiente objetivo general de investigación en este documento de tesis:

Sustentar por qué un problema de tipo no polinomial (NP), dentro de la lógica de la computación clásica, se puede convertir en un problema de tipo polinomial (P), dentro de la lógica de la computación cuántica.

Este documento de tesis aborda el problema fundamental de la Complejidad Computacional, en cuanto al estudio de los problemas Polinomiales (P) y No Polinomiales (NP); el estudio de la Teoría de la Información, y el estudio de la Lógica Cuántica. En palabras de Maldonado (2013):

«De manera puntual y franca: las matemáticas de las ciencias de la complejidad son matemáticas de sistemas discretos, y, en consecuencia, los temas referentes a la teoría de la información, los problemas de computación, y las lógicas no-clásicas constituyen, si cabe la expresión, el núcleo mitocondrial del estudio de la complejidad. Y por ello mismo, los problemas formulados por Cook, Karp y Levin».

Maldonado (2013) se refiere a los problemas formulados por Cook, Karp y Levin, a los problemas P vs NP.

En cuanto a la lógica cuántica, se estudian las principales compuertas cuánticas, partiendo de los principios fundamentales de la física cuántica, en resumen, son:

- Superposición cuántica (Probabilístico)
- Entrelazamiento cuántico (Comunicación instantánea)
- Paralelismo cuántico (No determinismo)
- Colapso de la Función de Onda de Schrödinger (Decoherencia)

Las propiedades anteriores, especialmente el no determinismo de la física cuántica, hacen de la lógica cuántica un modelo matemático formal, muy efectivo a la hora de reducir la com-

plejidad temporal de muchos problemas de tipo no polinomial.

La anterior argumentación será explicada con mucho detalle en el siguiente documento de tesis, haciéndose énfasis de que este documento no es una exposición o exploración de la física cuántica, es más una exposición o exploración de la lógica de la computación cuántica, conceptos muy diferentes.

Para terminar, reitero la importancia de este trabajo de investigación:

1. La lógica cuántica y los problemas P vs NP son dos de los tres ejes principales de las ciencias de la complejidad (Maldonado, 2010), el tercer eje es la teoría de los sistemas dinámicos, pero este tema no hace parte de esta investigación.

2. Relacionar los problemas P vs NP con la lógica cuántica es una ciencia en desarrollo, especialmente porque hay un algoritmo cuántico (algoritmo de Shor) que «amenaza» con solucionar la descomposición en factores primos de números grandes (el cual es un problema NP), y a la vez destruir la seguridad informática actual. Aunque ya se sabe que la criptografía cuántica podrá resolver este tipo de «amenaza» (García, 2020).

3. La relación entre los problemas P vs NP hace parte de los problemas del milenio (Maldonado, 2010).

Capítulo 4

Objetivos de la Investigación

4.1. Objetivo General

Sustentar por qué un problema de tipo no polinomial (NP), dentro de la lógica de la computación clásica, se puede convertir en un problema de tipo polinomial (P), dentro de la lógica de la computación cuántica.

4.2. Objetivos Específicos

- Explorar el concepto de Máquinas de Turing, computabilidad y algoritmo.
- Explorar el concepto de complejidad computacional.
- Interrelacionar y diferenciar la lógica cuántica y la lógica clásica.

Capítulo 5

Metodología

Esta investigación es de alcance exploratorio, ya que la relación entre la complejidad computacional y la lógica cuántica ha sido muy poco estudiada, y es una ciencia en desarrollo. Por otro lado, se busca describir las características de la complejidad computacional y de la lógica cuántica, correlacionar la complejidad computacional con la lógica cuántica, buscando cómo están relacionadas; también tiene cierto enfoque descriptivo, porque se describen las características de la complejidad computacional y de la lógica cuántica; también busca explicar por qué algunos problemas No Polinomiales (NP), dentro de la lógica clásica, podrían ser Polinomiales (P) dentro de la lógica cuántica.

El diseño de esta investigación es documental, tipo monográfica. Se ha investigado con profundidad el estado del arte de la computación cuántica, y se ha hecho un parangón entre la computación clásica y la computación cuántica, para establecer por qué podría darse una supremacía de la una frente a la otra en la resolución de problemas de tipo no polinomial (NP). Además, se ha investigado la teoría matemática de la complejidad y las lógicas no clásicas como la cuántica, como ejes de las ciencias de la complejidad. Se ha investigado la interpretación y el origen de la entropía de Shannon, desde la teoría matemática de la información, los elementos de la comunicación y la transmisión de información desde los aportes de Shannon y Weaver. Se ha expuesto la complejidad computacional, en la resolución de problemas de tipo P y NP, a partir de algoritmos deterministas y no deterministas, dentro de estructuras matemáticas conocidas como Máquinas de Turing.

Cada capítulo dentro del documento está direccionado para entender la complejidad computacional, la teoría matemática de la complejidad, y la lógica de la computación clásica frente a la lógica de la computación cuántica, teniendo siempre el objetivo de seguir la línea de la teoría de la complejidad, y sustentar la tesis de la supremacía de la computación cuántica frente a la computación clásica, en cuanto a la resolución de problemas no polinomiales (NP).

Esta investigación es de tipo documental, basada principalmente en la base de datos de la plataforma de Google Académico, toma como referencia los autores más reconocidos, investigaciones actualizadas y documentos en diferentes idiomas. Se construye un documento de consulta que estudia la teoría matemática de la complejidad, la teoría matemática de la información, y la supremacía de la computación cuántica frente a la computación clásica respecto

a la resolución de problemas complejos (no polinomiales).

Esta investigación no es experimental, ni de campo, es de diseño documental, de alcance exploratoria, tipo monografía, con el propósito de ser fundamental, orientada hacia la complejidad computacional y la lógica cuántica.

Este documento abarca conceptos en las matemáticas, la informática, la física, la filosofía, la psicología, la biología y las telecomunicaciones, siempre haciendo uso de las herramientas que nos expone la teoría de la complejidad. Por ello, aparte de tener el objetivo general de investigar la supremacía cuántica frente a problemas de tipo no polinomial (NP), se persigue el objetivo de construir un documento científico de la teoría de la complejidad computacional y la lógica cuántica. Dicho propósito es de tipo cognitivo, comprensivo y metódico, más inclinado hacia las ciencias de la complejidad que hacia el pensamiento complejo como método.

Capítulo 6

Didáctica de los Problemas P vs NP

"Si tuviese que reducir toda la psicología educativa a un solo principio, enunciaría éste: de todos los factores que influyen en el aprendizaje, el más importante consiste en lo que el alumno ya sabe. Averígüese esto, y enséñese consecuentemente" (Ausubel 1976, citado por Palmero, 2011).

Este capítulo va a tener una construcción no formal, se van a exponer varios ejemplos de la complejidad computacional, sin el lenguaje riguroso que exige la construcción de la ciencia. El objetivo de este capítulo es poder explicar los problemas P vs NP para personas que no tienen una formación matemática, o para estudiantes de básica primaria, o de básica secundaria y media. El lenguaje que se va a utilizar es muy intuitivo, la idea de este capítulo es fundar unas bases en los estudiantes que no tienen fuertes conocimientos matemáticos, para que se motiven a seguir investigando. Sabemos que la didáctica puede hacer uso de diferentes estrategias para que el estudiante acceda a un aprendizaje significativo, significativo en el sentido de que este aprendizaje sea considerado relevante, importante y necesario por parte del estudiante. Pérez Gómez (2006), citado por Palmero (2011), considera aprendizaje relevante como "aquel tipo de aprendizaje significativo que por su sentido e importancia para el individuo provoca inestabilidad cognitiva, conflicto cognitivo, duda e interrogación, porque les hace repensar sus esquemas clásicos de interpretación al darse cuenta de que son insuficientes y les hace abrirse a la posibilidad de construir nuevos esquemas de interpretación de la realidad que son y que incluyen conocimientos, habilidades, actitudes y comportamientos en parte nuevos".

Una construcción un poco más científica y formal de la complejidad computacional, se puede encontrar en los capítulos siguientes de este documento.

Se sabe que la teoría del aprendizaje significativo ha cambiado mucho desde que David P. Ausubel la propuso en 1963, ya que es una teoría de la psicología que centra la atención en lo que ocurre en el aula cuando los estudiantes aprenden (Ausubel 1976, citado por Palmero, 2011), y también es una teoría del aprendizaje porque esa es su finalidad (Rodríguez, 2004 a, 2008, citado por Palmero, 2011). Por lo anterior, Palmero (2011), argumenta que la teoría del aprendizaje significativo es una teoría que se ocupa de lo que ocurre en el aula, y de cómo facilitar los aprendizajes que en ella se generan. Sin embargo, Palmero (2011) nos dice que

la teoría del aprendizaje significativo es un constructo dinámico que ha evolucionado junto con los avances de la psicología cognitiva, ya que «Ausubel insiste en la interacción entre los nuevos conocimientos y los conocimientos previos para que haya aprendizaje significativo, pero no da cuenta ni del proceso mismo, ni de las condiciones y características de esa interacción» (Palmero, 2011).

Se sabe que «El proceso de enseñanza-aprendizaje es complejo, multifactorial, de múltiples interacciones, donde las condiciones son definitivamente las que favorecen o dificultan el propio proceso y el resultado. Existen múltiples alternativas que deben analizarse en función de los resultados esperados y así activar los procesos necesarios para alcanzarlos» (Addine y otros, 2020). Podemos intuir que la educación como proceso de enseñanza-aprendizaje, depende de muchísimas variables, algunas tal vez desconocemos a la hora de enseñar, por lo que las interacciones son muchísimas y algunas posiblemente desconocidas también a la hora de enseñar. Además, podemos observar que la educación, es un sistema dinámico, en el sentido en que Montealegre y otros (2002), definen un sistema dinámico, con la característica de evolucionar con el tiempo y retroalimentarse a sí mismo, esto es lo que se conoce como procesos iterativos no lineales. Además, en el proceso de enseñanza, influyen muchos factores, y agentes, por ejemplo: Palmero (2011) citando a Gowin (1981), nos dice que «la enseñanza se consuma cuando el significado del material que el alumno capta es el significado que el profesor pretende que ese material tenga para el alumno", y que por ello se establece una relación triádica entre profesor, alumno y materiales educativos del currículo que tiendan a compartir significados.

Se ha expuesto una idea muy general sobre el complejo proceso de enseñanza-aprendizaje, pero este no es el objetivo de este capítulo, es sólo una manera de exponer lo difícil y complejo que es el proceso educativo. Por ello ahora si nos entramos en el objetivo principal de este capítulo.

6.1. Matrioshka (Muñecas rusas)

Se le llama Matrioshka a una famosa muñeca rusa de madera, lo curioso es que si se abre esta muñeca, por dentro encuentras otra muñeca, y así sucesivamente hasta encontrar una muñeca similar pero muy pequeña.

Figura 6.1.1: Muñeca rusa en situación inicial (Khan Academy, 2021).

Figura 6.1.2: Muñeca rusa en situación final (Khan Academy, 2021).

Antes se va a dar una idea no formal de lo que es un algoritmo. Se entiende por algoritmo, a una colección de instrucciones simples para realizar alguna tarea (Sipser, 2006).

El ejemplo de las muñecas rusas se utiliza para explicar el concepto de recursividad, que es la técnica que consiste en que un algoritmo se llame o se ejecute a sí mismo. De esa manera, el algoritmo está dentro de sí mismo, y así sucesivamente hasta que sea necesario, como cuando aplicamos el algoritmo o procedimiento de destapar una muñeca dentro de otra. Esta técnica es muy fácil de explicar en el lenguaje de la programación informática, pero ese nivel no es la finalidad de este capítulo.

6.2. Torres de Hanoi

El juego de las Torres de Hanoi consiste en ir cambiando los discos de la torre 1 a la torre 3, con la condición de que no se puede mover más de un disco a la vez, y que no puede colocarse un disco grande sobre uno pequeño (Uterra.com, 2021). Por ejemplo, se desarrolla el juego para un número «n» igual a tres discos, o sea n = 3.

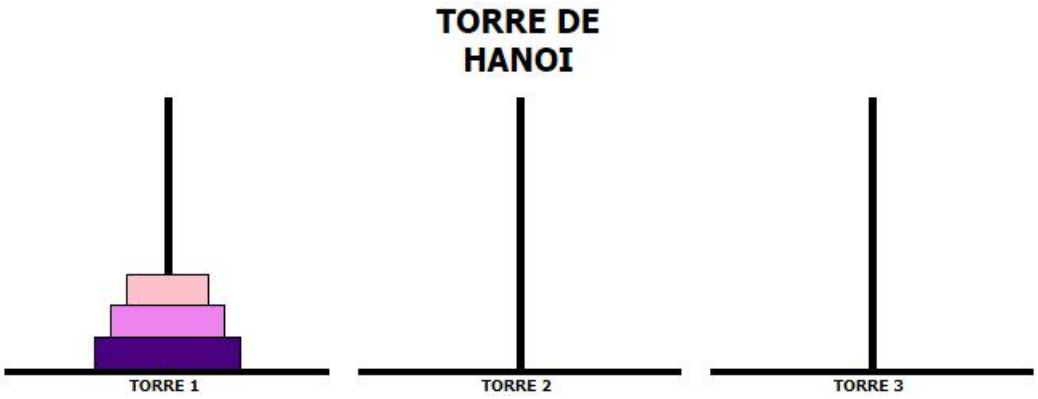


Figura 6.2.1: Torres de Hanoi de tres discos, posición inicial (Uterra.com, 2021).

Figura 6.2.2: Torres de Hanoi de tres discos, primer movimiento (Uterra.com, 2021).

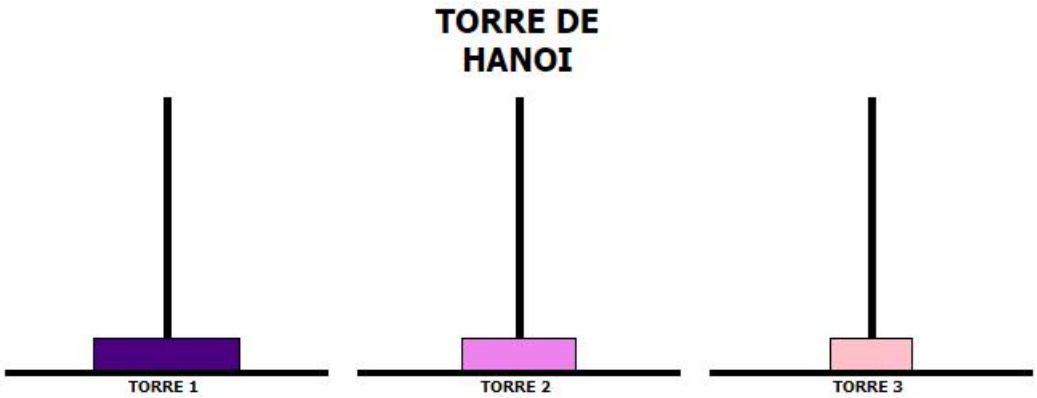


Figura 6.2.3: Torres de Hanoi de tres discos, segundo movimiento (Uterra.com, 2021).

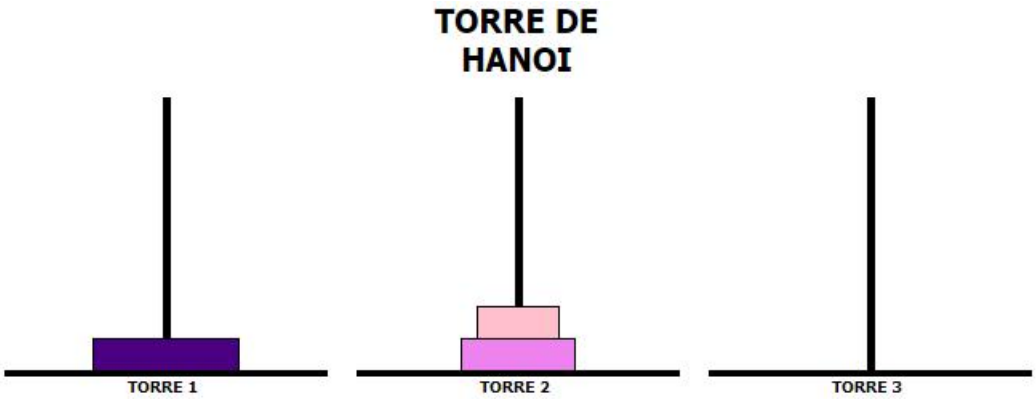


Figura 6.2.4: Torres de Hanoi de tres discos, tercer movimiento (Uterra.com, 2021).

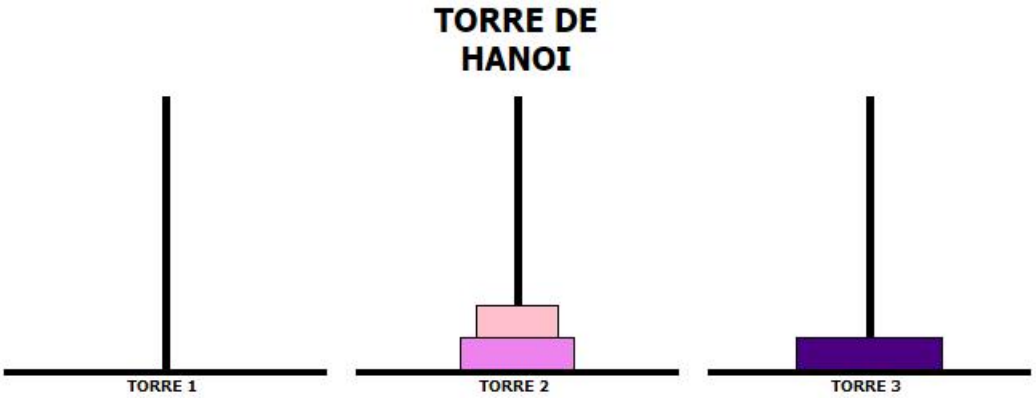


Figura 6.2.5: Torres de Hanoi de tres discos, cuarto movimiento (Uterra.com, 2021).

Figura 6.2.6: Torres de Hanoi de tres discos, quinto movimiento (Uterra.com, 2021).

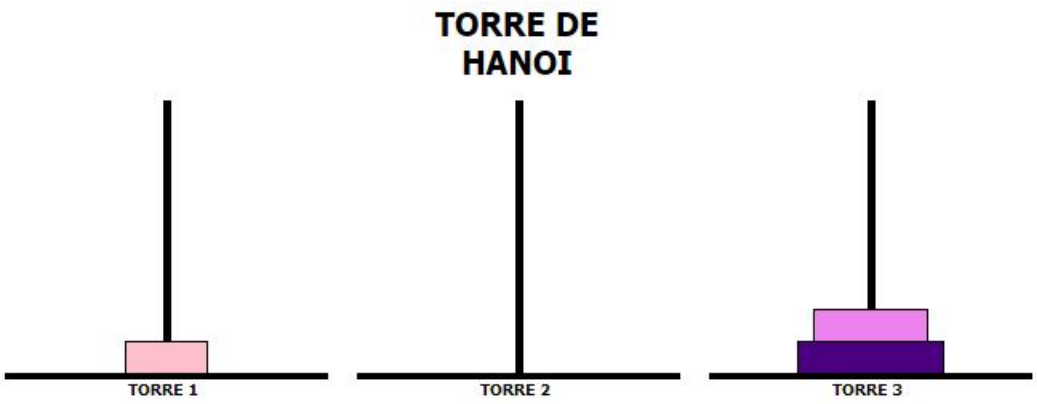


Figura 6.2.7: Torres de Hanoi de tres discos, sexto movimiento (Uterra.com, 2021).

Figura 6.2.8: Torres de Hanoi de tres discos, séptimo y último movimiento (Uterra.com, 2021).

Se observa que si se juega las Torres de Hanoi para n = 3 discos, se puede terminar el juego con un mínimo de 7 pasos o movimientos. Si se juega las Torres de Hanoi para n = 4 discos, se puede terminar el juego con un mínimo de 15 pasos o movimientos. Si se juega las Torres de Hanoi para n = 5 discos, se puede terminar el juego con un mínimo de 31 pasos o movimientos. Si se juega las Torres de Hanoi para n = 6 discos, se puede terminar el juego con un mínimo de 63 pasos o movimientos. Si se juega las Torres de Hanoi para n = 7 discos, se puede terminar el juego con un mínimo de 127 pasos o movimientos. Y así se observa que para «n» discos se puede terminar el juego en un mínimo de $2^n - 1$ pasos.

El anterior juego se puede jugar y verificar en la página web:
http://www.uterra.com/juegos/torre_hanoi.php

Para la siguiente explicación, «tiempo polinomial» significa tiempo corto o razonable, porque se puede hallar por una función polinomial, mientras que «tiempo no polinomial» significa tiempo exageradamente largo, tiempo no razonable, tiempo que no se puede tratar o manejar, la expresión «tiempo no polinomial» viene de que ese tiempo se calcula con fun-

ciones exponenciales y no con polinomios.

En el juego de las Torres de Hanoi, se observa un crecimiento exponencial en el mínimo de pasos que se necesita para resolver el juego. Es un problema de tipo polinomial para valores de «n» pequeños, porque se necesitan tiempos medibles, razonables o polinomiales, pero si el valor de «n» es grande, por ejemplo si n = 64, el tiempo no sería razonable o tratable, porque este problema se resolvería en un tiempo mucho mayor que la edad del universo desde el Big Bang hasta ahora. Este tipo de problemas se vuelven intratables pero sí se pueden resolver, se les llama problemas No Polinomiales (NP), porque se resuelven con algoritmos deterministas en tiempos no polinomiales; para resolverse en tiempos polinomiales o razonables, tendrían que resolverse con algoritmos no deterministas (Maldonado, 2010).

También hay otra clase de problemas, son los problemas NP-Completos, estos problemas son los que no se sabe si tienen solución en tiempo polinomial, pero sí se sabe que son todos equivalentes, quiere decir que si se pudiera resolver al menos uno en tiempo polinomial, con el mismo algoritmo se resolverían todos (Arjona, 2007).

Existe una leyenda que dice que, cuando se termine de resolver el juego de las Torres de Hanoi de 64 discos, en ese momento se acabaría el mundo. Lo anterior es cierto, pero no es porque haya un misterio, es sólo porque para resolver las Torres de Hanoi de 64 discos, se necesitan $2^{64} - 1$ = 18.446.744.073.709.551.615 de pasos, y si alguien pudiera hacer un sólo paso en un segundo, gastaría 18´´´446.744´´073.709´551.615 segundos, y ese número es demasiado grande, en letras quedaría: dieciocho trillones, cuatrocientos cuarenta y seis mil setecientos cuarenta y cuatro billones, setenta y tres mil setecientos nueve millones, quinientos cincuenta y un mil seicientos quince segundos, y este número gigante en segundos, al convertirse en años, quedaría: 584.942´417.355 años, y en letras sería: quinientos ochenta y cuatro mil novecientos cuarenta y dos millones, cuatrocientos diecisiete mil trescientos cincuenta y cinco años. La edad estimada del universo desde el Big Bang es de 13.770 millones de años, nuestro sol se extinguirá aproximadamente en 5.000 millones de años, estas son cifras muchísimo más pequeñas que el número de años que se necesitan para resolver un juego de Torre de Hanoi de 64 discos.

Capítulo 7

Máquinas de Turing

7.1. Máquinas de Turing Deterministas

«Para dar una definición matemáticamente precisa de algoritmo, Alan Turing ideó un dispositivo imaginario al que denominó Máquina de Computación Lógica, pero que ha recibido en su honor el nombre de Máquina de Turing, y definió un algoritmo como cualquier conjunto de instrucciones para dicha máquina» (Pérez, 2010).

Una Máquina de Turing MT, es un modelo matemático similar a un autómata finito pero con una memoria ilimitada y sin restricciones, es un modelo mucho más preciso de una computadora de uso general. Una Máquina de Turing puede hacer todo lo que una computadora real puede hacer. El modelo de la Máquina de Turing utiliza una cinta infinita con un cabezal o aguja que lee, escribe y borra símbolos, moviéndose de forma lateral hacia la izquierda o hacia la derecha. Toda la cinta está marcada con espacios o cuadros en blanco, y es allí donde el cabezal de la máquina escribe y borra símbolos (Sipser, 2006).

La definición de Máquina de Turing Determinista MT, según Jurado (2008), es:

Definición 1. Una Maquina de Turing MT es una 7-tupla, (Q, Σ, Γ, f, q_0, $\square$, F), donde Q, Σ, Γ, son conjuntos finitos y

1. Q es el conjunto finito de estados.
2. Σ es el alfabeto de la cadena de entrada, no conteniendo el símbolo del espacio en blanco «$\square$».
3. Γ es el alfabeto admitido por la cinta, donde $\square \in \Gamma$ y $\Sigma \subseteq \Gamma$.
4. $q_0 \in Q$ es el estado inicial.
5. $\square$ es el espacio en blanco, $\square \in \Gamma$ pero $\square \notin \Sigma$, $\square$ representa una casilla vacía de la cinta de escritura de la Máquina de Turing.
6. $F \subset Q$ es el conjunto de estados finales.

7. f: Q x $\Gamma \longrightarrow Q$ x Γ x $\{I, D, P\}$es la función de transición, donde $\{I, D, P\}$ son los movimientos del cabezal de la máquina, hacia la izquierda I, hacia la derecha D o parada P. En este item, Jurado Málaga, E. (2008), no menciona la acción del cabezal en Parada (P) o en Stop (S), pero para nuestro caso es muy necesario porque el software que se expone en este documento, JFLAP, si maneja la acción de parada (S), y muchos autores sí lo consideran indispensable, por ejemplo Arora, S., & Barak, B. (2009).

En palabras de Sipser (2006), el Lenguaje de una máquina M, o el Lenguaje Reconocido por la máquina M, se denota como L(M).

Definición 2. Se llama un lenguaje Reconocible Turing, si alguna Máquina de Turing lo reconoce.

Una Máquina es decisoria si toma una decisión de aceptar o rechazar un dato de entrada, no quedando en bucle. De esta manera si una máquina se demora siendo decisoria, podemos saber que la máquina está tardando en tomar una decisión pero no está en bucle.

Definición 3. Se llama un lenguaje decidible-Turing o simplemente lenguaje decidible si alguna máquina de Turing lo acepta o lo rechaza.

Ejemplo 4. Describamos una máquina de Turing (TM), llamada M2 que decide el lenguaje A = $\{O^{2^n} \mid n \geq 0\}$, el lenguaje consiste de todas las cadenas de Os cuya longitud es una potencia de 2.

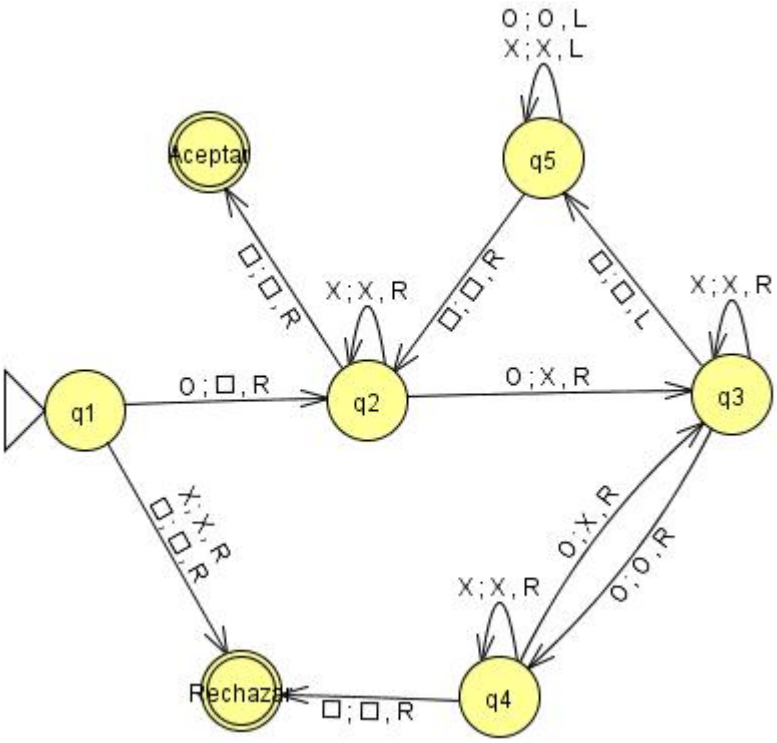


Figura 7.1.1: Máquina de Turing que reconoce o decide si una cadena de Os es una potencia de 2 (Sipser, 2006).

La máquina M2 cumple las siguientes condiciones para una cadena de entrada λ:

1. La primera O la cambia por la casilla vacía, luego recorre la cinta de izquierda a derecha marcando con una X cada O de por medio.
2. Si en el paso 1 la cinta contiene una única O, es aceptada.
3. Si en el paso 1 la cinta contiene más de una O y el número de parejas de Os era impar, se rechaza.
4. Regrese la cabeza del lector de la máquina, al extremo izquierdo de la cinta.
5. Vaya al paso 1.

Se trata de describir una máquina que sea capaz de determinar si una cadena de Os es una potencia del número dos.

$Q = \{q_1, q_2, q_3, q_4, q_5, q_{aceptar}, q_{rechazar}\}$, conjunto de estados.
$\Sigma = \{O\}$, alfabeto de la cadena de entrada.
$\Gamma = \{O, X, \square\}$, alfabeto leído por la cinta.

La función de transición f se describe en la figura 7.1.1 (Sipser, 2006).

El lector puede ingresar cualquier cadena, por ejemplo OOOOOOOOOOOOOO, al ingresarla en la máquina anterior queda:

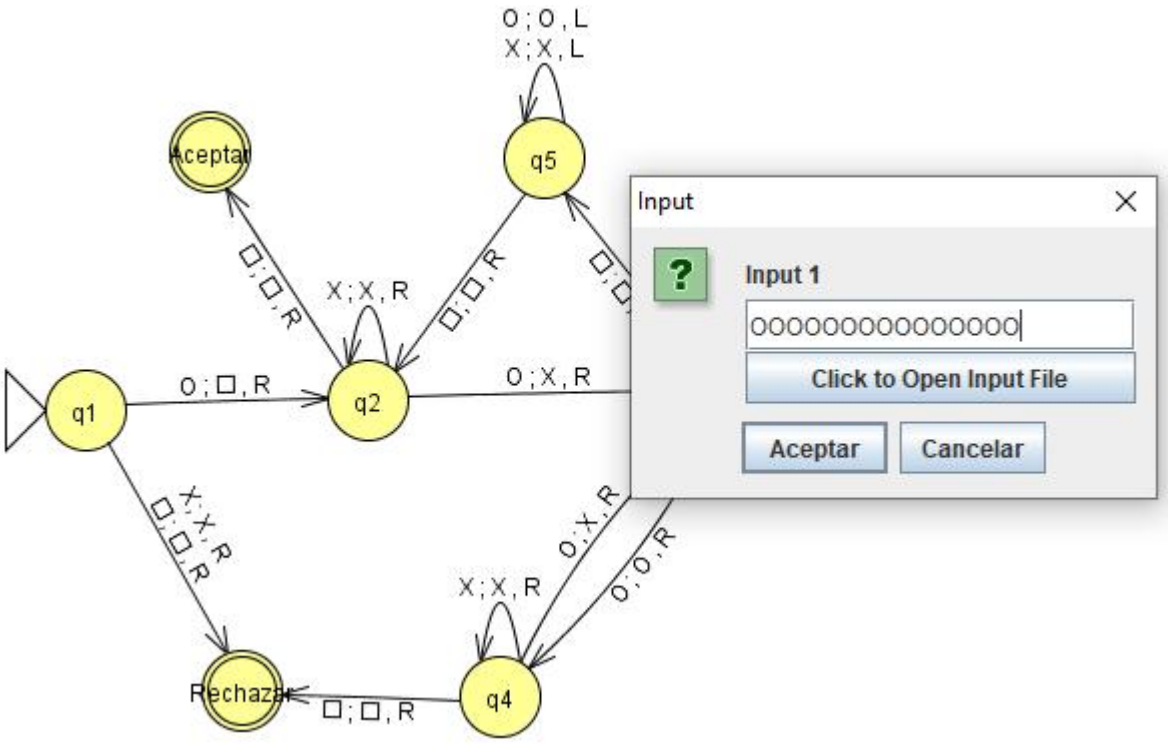


Figura 7.1.2: Ejemplo de ingreso de cadena de la Máquina de Turing, con el programa JFLAP.

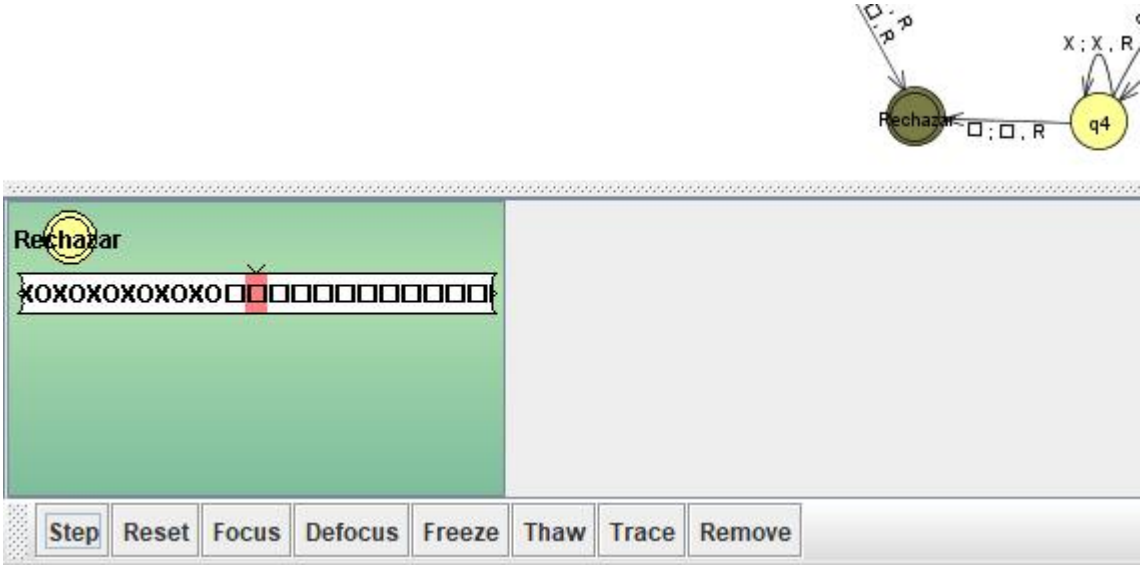


Figura 7.1.3: Resultado de ingresar la cadena OOOOOOOOOOOOOO, con el programa JFLAP

Se observa en la última figura, que la cadena insertada fue rechazada, debido a que la cadena OOOOOOOOOOOOOO, contiene quince Os, y este número no es una potencia de 2.

Para instalar el programa JFLAP, se puede descargar en el enlace: http://www.jflap.org/jflaptmp/ , y buscar la versión que esté vigente, en este caso se trabajó con la versión JFLAP7.1.jar, la extensión del programa está en .jar, lo que significa que es un programa ejecutado en el lenguaje de programación JAVA, por lo tanto, antes de ejecutar este programa, se debe tener instalado en el computador el programa que ejecuta aplicaciones en la plataforma de JAVA, este programa se llama JDK, dicho programa se puede encontrar en la dirección: https://www.oracle.com/technetwork/es/java/javase/downloads/index.html , y buscar la versión JDK más reciente, en este caso se descargó la versión JDK8. Se recuerda que los computadores tienen dos tipos de sistema, uno de 32bits u otro de 64 bits, por eso cuando se descargue la versión del programa JDK, se debe elegir el que diga x86 si su sistema es de 32 bits, o x64 si su sistema es de 64 bits.

También se puede ejecutar la máquina de manera manual teniendo en cuenta la figura 7.1.1, para ello se comienza en el estado q_1, el cual es el estado inicial. La notación (O, □, R) significa que si se lee una O se debe escribir en esa casilla un cuadro en blanco y el cabezal de lecto escritura de la máquina se corre a la derecha en la cinta. La notación (O, X, R) significa que si se lee una O se debe escribir en esa casilla una X y el cabezal de lecto escritura se corre a la derecha; de esa manera se lee y se ejecuta cada letra de la cadena de entrada.

Ejemplo 5. Complemento binario. Esta Máquina de Turing hace que una cadena de entrada compuesta por 0 y 1, sea transformada en otra cadena de salida, donde cada 1 es cambiado por cero, y cada 0 es cambiado por 1. Por ejemplo, si insertamos la cadena 11010 se obtiene como resultado 00101.

Figura 7.1.4: Representación del complemento a binario en una Máquina de Turing, bajo el programa JFLAP.

Conjunto de estados $Q = \{q_0, q_1\}$, conjunto de estados finales $F = \{q_1\}$, alfabeto de la cadena de entrada $\Sigma = \{0, 1\}$,alfabeto admitido por la cinta $\Gamma = \{0, 1, \square\}$, el programa JFLAP

utiliza la notación □ para referirse al espacio en blanco o casilla vacía de la cinta.

El círculo con q_0 representa al estado inicial de la máquina, el estado inicial está marcado con un triángulo apuntando al círculo. Cada flecha o arco representa la función de transición, f: Q x $\Gamma \longrightarrow Q$ x Γ x $\{I, D, P\}$, significa que si estamos ubicados en un estado y se lee una entrada, entonces nos ubicamos en otro o el mismo estado, y se escribe un símbolo admitido por el alfabeto de la cinta, además el cabezal de la cinta hace un movimiento hacia la Izquierda, hacia la Derecha o se queda Parado. El programa JFLAP utiliza la notación en inglés Left, Right and Stop.

En la figura aparecen unos arcos o flechas con la notación (1; 0, R), esto representa cada función de transición; la máquina se ubica en un estado q_0, lee un 1, escribe un 0, y mueve el cabezal de lecto-escritura hacia la derecha. La notación (0; 1, R) significa que la máquina lee un 0, escribe un 1 y mueve el cabezal de lecto-escritura hacia la derecha. La notación (□; □, S) significa que la máquina lee una casilla vacía, lo cual indica que la cadena de entrada ya terminó, escribe un espacio vacío □ y el cabezal queda en Stop.

El círculo con otro círculo dentro, representa un estado final, indica que la cadena de entrada ha sido leída y ha terminado su ejecución porque ha llegado a un estado final.

En el ejemplo, si insertamos la cadena 11010 se obtiene como resultado 00101. En la siguiente imagen se expone la ejecución final con JFLAP:

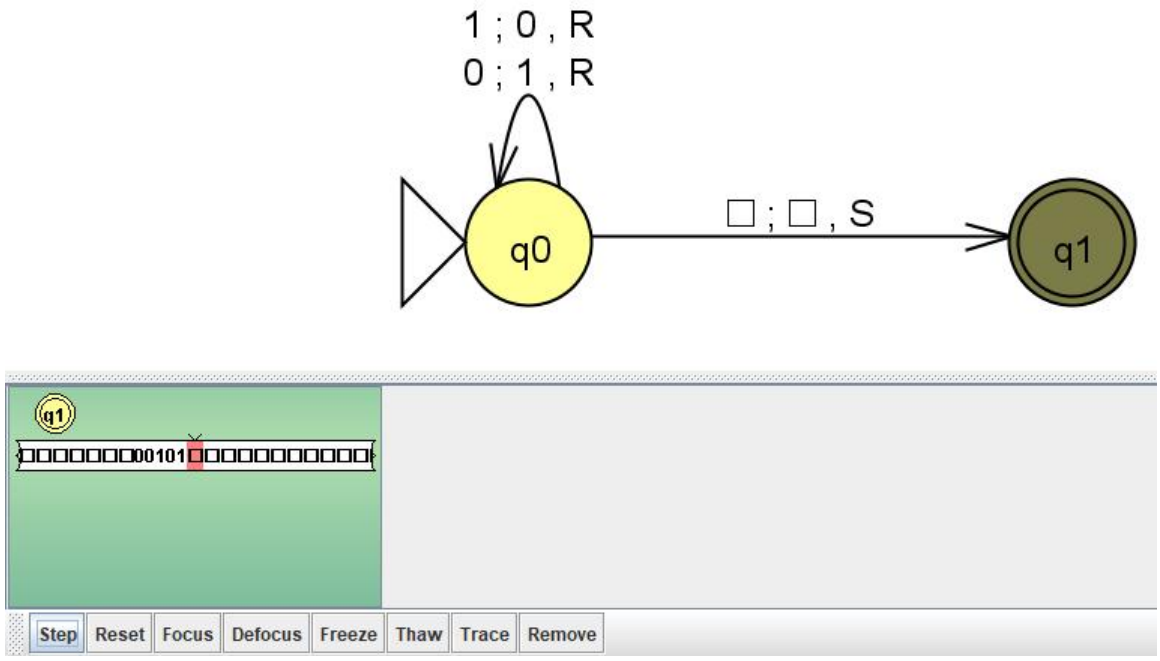


Figura 7.1.5: Representación del complemento a binario en una Máquina de Turing, con el programa JFLAP, para una cadena determinada.

Ejemplo 6. Paridad. Dado una cadena compuesta por 0 y 1, determinar si el número de 1s es par o impar, si llegase a ser par se añade un cero al final, y si es impar se añade un 1 al final.

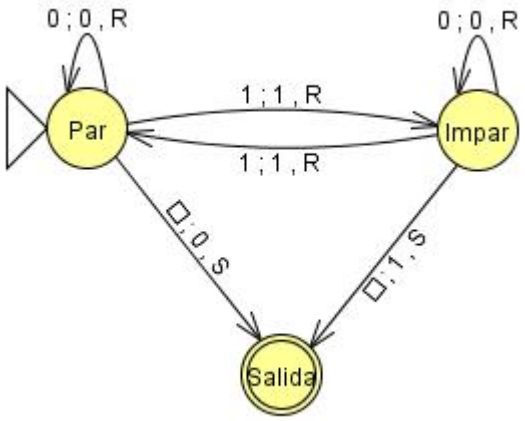


Figura 7.1.6: Máquina de Turing para marcar un número binario par o impar.

El estado inicial se llamará «Par», porque aún no se ha insertado algún 1, el número de unos en el momento sería cero, y el cero es par; en el momento de ingresar algún uno, saltaría al estado que se llamará «Impar», porque en este momento habría una cantidad impar de unos, pero si recibimos otro uno (1) saltaría al estado «Par», y en este estado se tendría una cantidad de unos par. Se recuerda que sólo se está teniendo en cuenta si el número de unos es par o impar, independientemente de la cantidad de ceros, y tampoco se registra la cantidad de unos, sólamente se registra si esa cantidad de unos es par o impar. Así se mantiene la máquina indefinidamente hasta que termine la cadena de entrada, en ese momento llegaría una casilla vacía, también llamada espacio en blanco. En el momento de llegar una casilla vacía, le indicaría a la máquina que la cadena de entrada ha terminado, por ello se registra en esta casilla un cero si el número de unos de la cadena es impar, o se registra en esta casilla un uno. Por ello si se ingresa cualquier cadena, por ejemplo 10111001, la salida será 101110011, donde en la casilla final se registraría un 1, porque en la cadena de entrada había un número impar de unos. Pero si se ingresa otra cadena, 0010111, la salida será 00101110, donde en la casilla final se registra un 0, porque en la cadena de entrada había un número par de unos. Para sucesivos ejemplos en JFLAP, no se va a explicar a fondo cada estado y cada función de transición, debido a que en los ejemplos anteriores se explicó muy detenidamente el funcionamiento de una máquina representada en JFLAP, a menos que la situación tenga algún elemento distinto.

7.2. Máquinas de Turing Multicintas

En este caso la Máquina de Turing (MT), dispone de k cintas y por ello tiene k cabezas de lectura - escritura, una para cada cinta funcionando de manera independiente. Este tipo

de máquinas no son más potentes, son equivalentes a otras máquinas de una cinta, siempre se puede construir una máquina de una cinta que sea equivalente a una máquina multicinta (Jurado, 2008).

Como ejemplos de máquinas multicintas se expone:

Ejemplo 7. Sumador de números binarios.

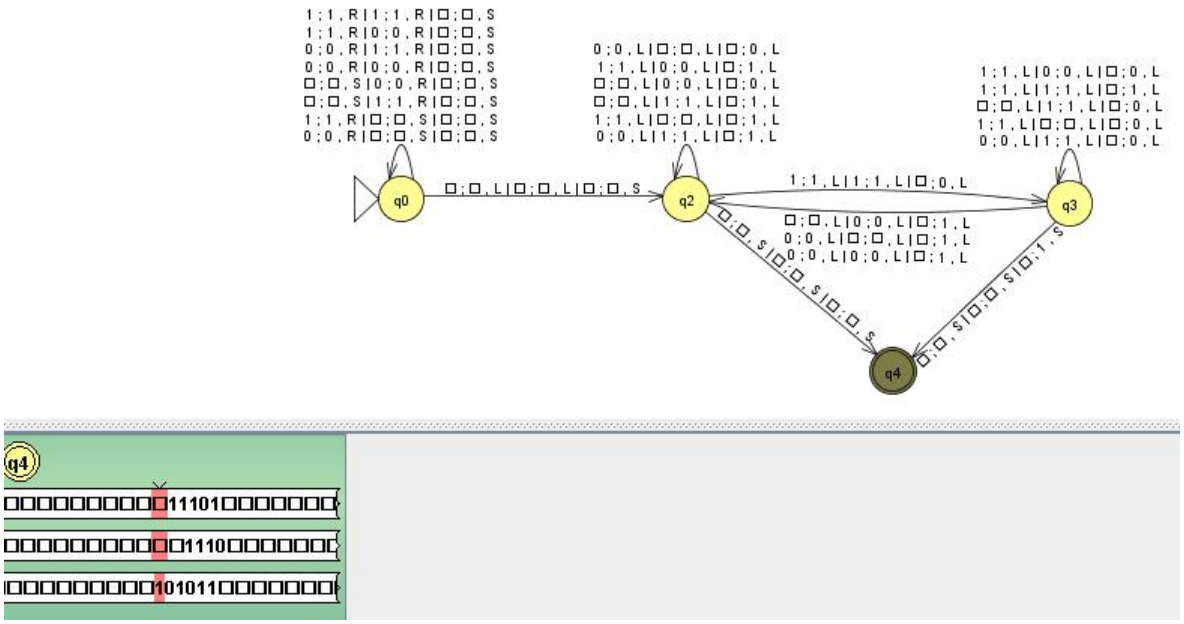


Figura 7.2.1: Máquina de Turing en el programa JFLAP.

La anterior Máquina de Turing es un sumador de números binarios, utiliza tres cintas, en las dos primeras cintas se insertan los sumandos. La tercera cinta se deja vacía, debido a que allí la máquina va a escribir la respuesta de sumar los dos binarios que se insertaron en las dos primeras cintas.

El estado q_0 es un estado inicial, se utiliza para alinear los dos binarios que se van a sumar, debido a que toca alinearlos de derecha a izquierda para poder hacer la suma convencional. En este estado, cada función de transición considera cada posibilidad en que los binarios estén desalineados. Por ejemplo, la función $\square;\square,S|1;1,R|\square;\square,S$, corresponde a tres cintas, donde la primera cinta indica que si se lee una casilla vacía, entonces se debe escribir otra casilla vacía y el cabezal de la máquina debe quedarse parado. En la segunda cinta, indica que si se lee un 1, entonces se debe escribir un 1 y el cabezal de la máquina debe moverse a la derecha. La tercera cinta indica que si se lee una casilla vacía, entonces se debe escribir una casilla vacía y el cabezal de la cinta debe quedarse parado. De esta manera se explica los términos de las funciones de transición, sin embargo es indispensable descargar el software

JFLAP y ejecutar la máquina, de lo contrario no se podría entender completamente la operación de esta máquina.

El estado $q2$ se utiliza para sumar cada dígito binario de los sumandos, siempre y cuando no se lleve algún uno en la suma, por ejemplo si suma 1+1=0 y lleva 1. El estado $q3$ se utiliza para sumar cada dígito binario de los sumandos, siempre y cuando sí se lleve algún uno en la suma. El estado q_4 se utiliza cuando ya no hay dígitos para sumar y sólo quedan casillas vacías, este estado indica que la operación ha terminado y que ya hay una respuesta en la tercera cinta, por eso se le llama estado final.

De nuevo se recuerda que, para instalar el programa JFLAP, se debe instalar primero el programa JDK, ya que JFLAP es un programa que está hecho en JAVA. Se recuerda que los computadores tienen los tipos de sistema de 32bits o de 64 bits, por eso cuando se descargue el programa JDK, se debe elegir x86 para 32 bits y x64 para 64 bits. Debajo de la figura 7.1.3, se explicó muy bien este proceso.

Ejemplo 8. Contador del símbolo 1 dentro del alfabeto de la cadena de entrada $\Sigma = \{1\}$.

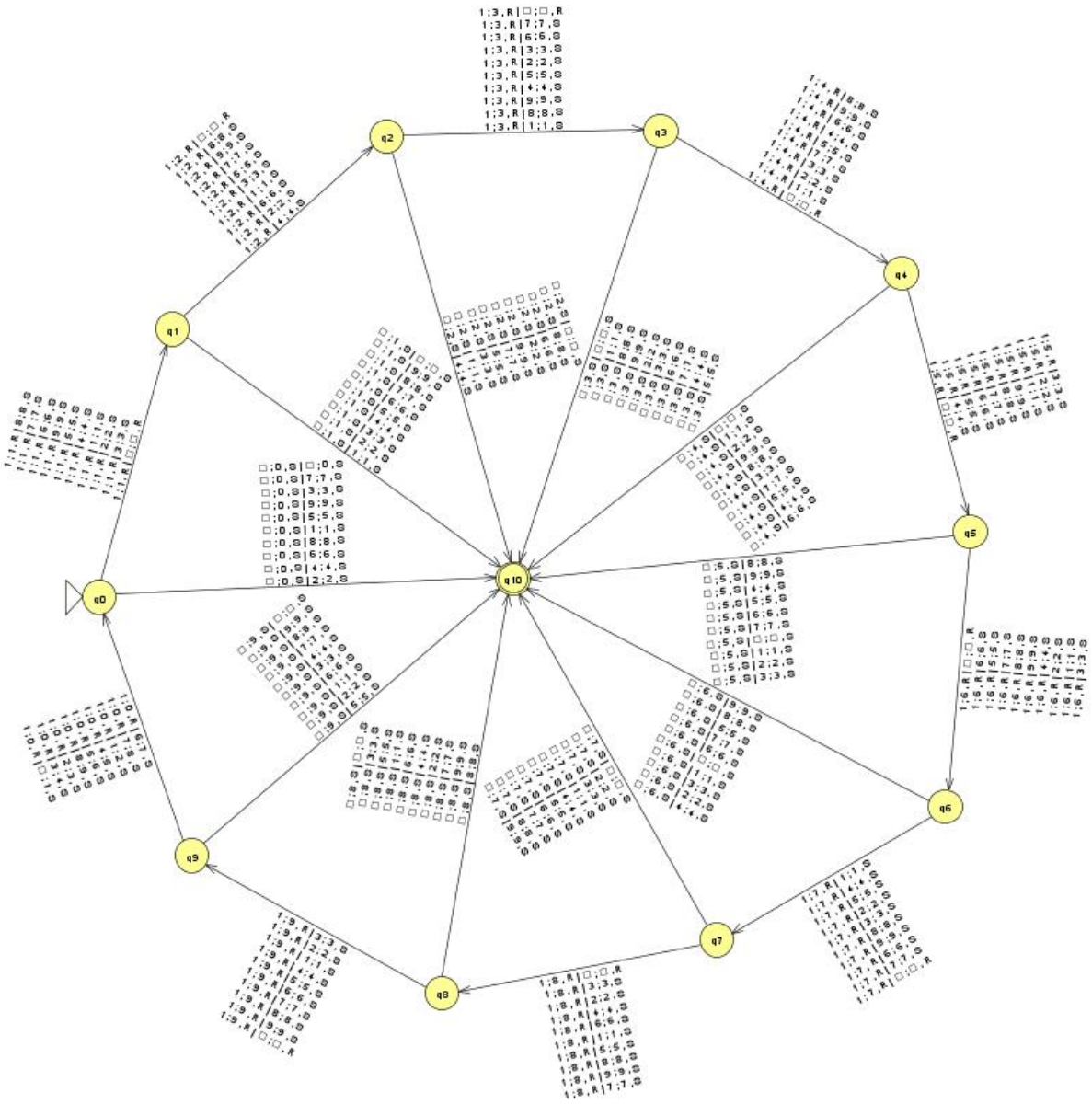


Figura 7.2.2: Máquina de Turing que recibe una cadena de unos y los cuenta.

La figura anterior no se puede distinguir claramente en la imagen, porque tiene once estados, dos cintas, y 200 funciones de transición para cada cinta, para un total de 400 funciones de transición. Esta máquina lee una cadena de unos 11111.... de longitud máximo de 99 unos. No se hizo más grande porque no era necesario en este libro, el objetivo es mostrar que las operaciones con la base diez son muy extensas en las Máquinas de Turing, por eso en informática es mejor operar en bases más pequeñas como la base binaria, este ejercicio lee una cadena de unos y representa su longitud en términos de números en base diez, por eso tiene diez estados y uno final para mostrar el resultado. Tiene dos cintas, en la primera cinta escribimos la cadena de entrada, esa cadena la pide la máquina, luego la máquina se ejecuta sola, la máquina va escribiendo las unidades en la primera cinta y las decenas en la segunda

cinta, por ejemplo, si ingresamos la cadena 111111111111111111111111 que tiene 24 unos, al ejecutarlo en la máquina resultará:

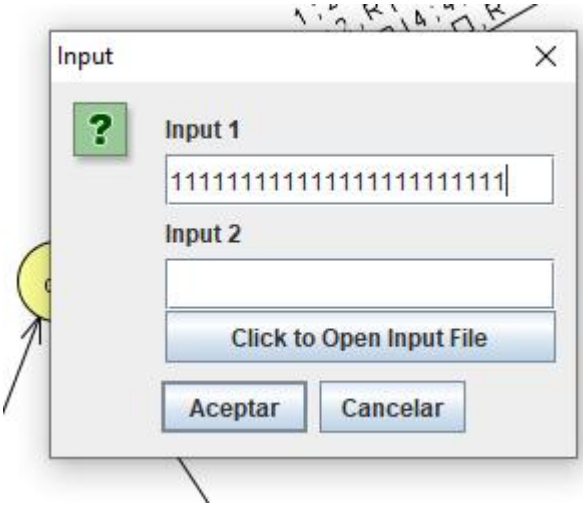


Figura 7.2.3: Ingreso de la cadena de entrada.

Al hacer click en aceptar daría la respuesta:

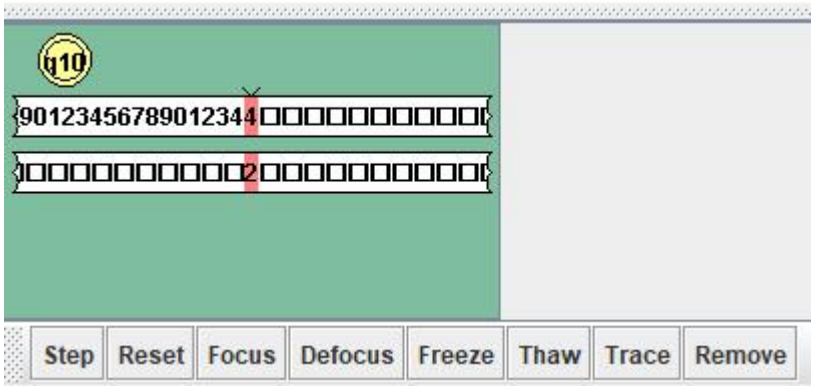


Figura 7.2.4: Resultado de contar dentro de una cadena 24 unos.

Con el fin de mostrar estas funciones de transición, con sus cintas y estados, se pegan las siguientes imágenes:

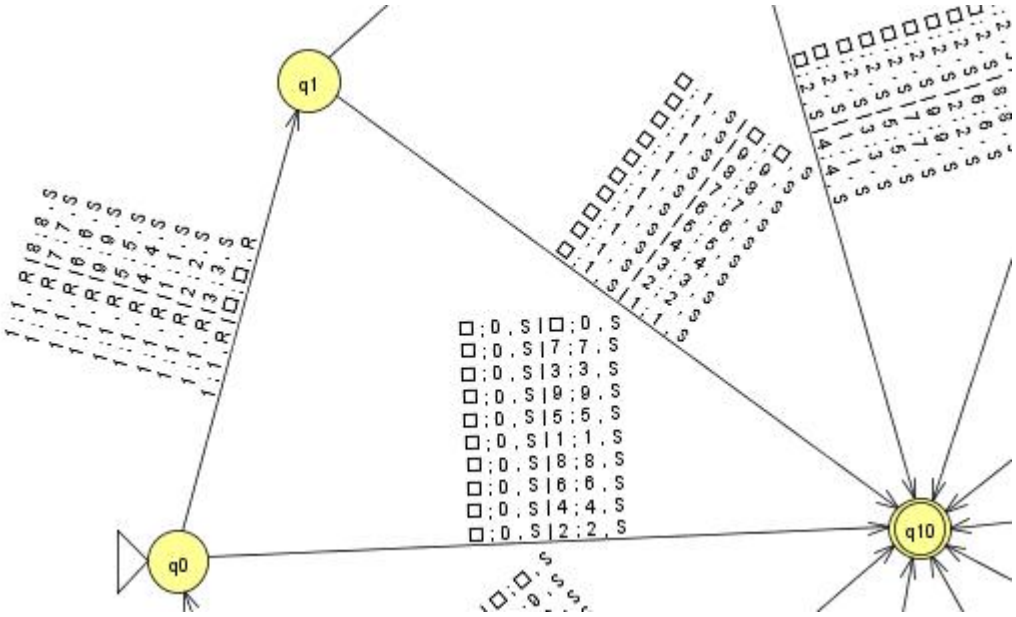


Figura 7.2.5: Estados 0 y 1.

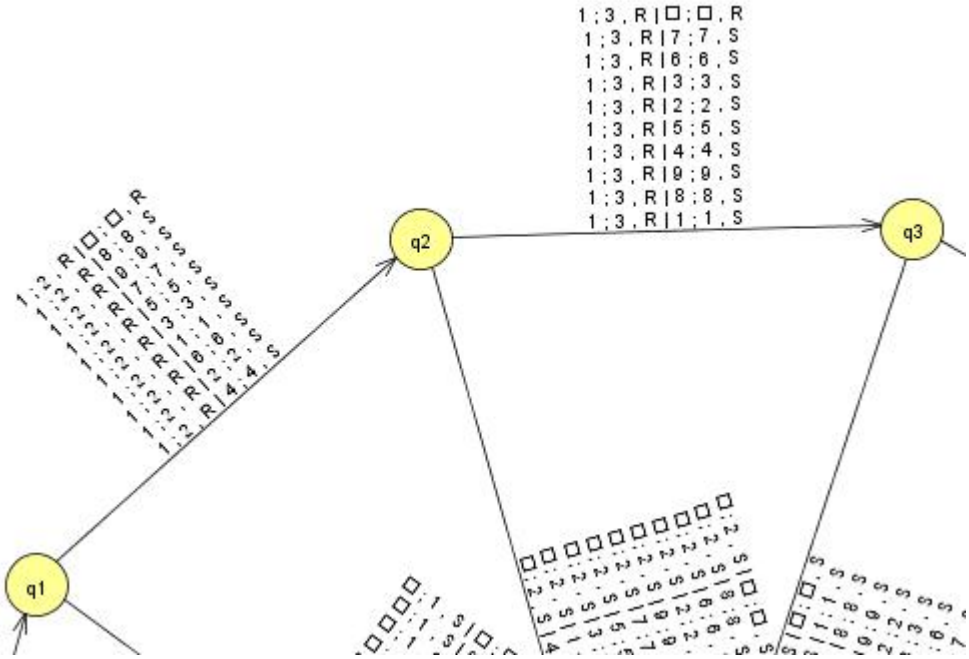


Figura 7.2.6: Estados 1, 2 y 3.

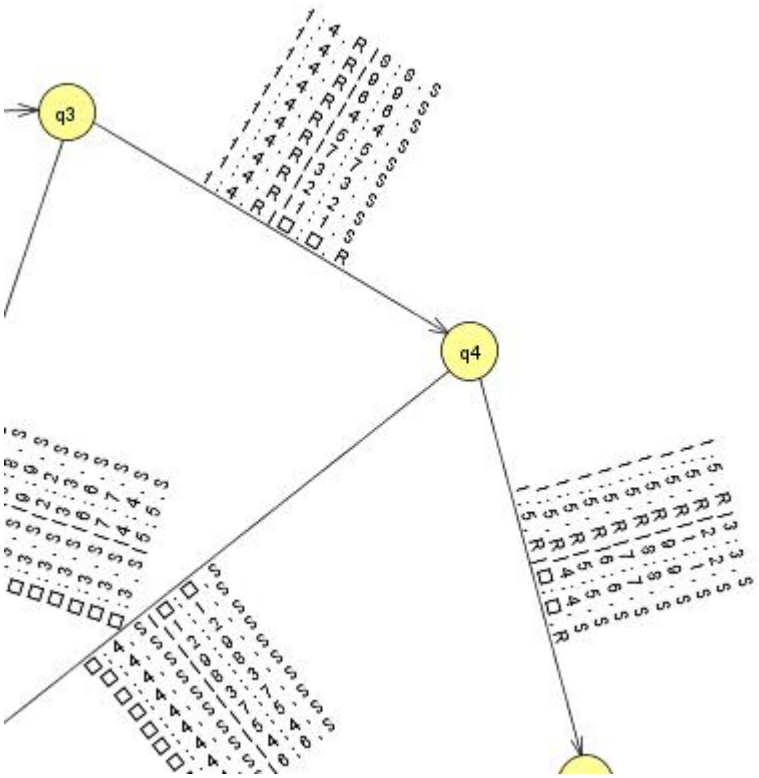


Figura 7.2.7: Estados 3 y 4.

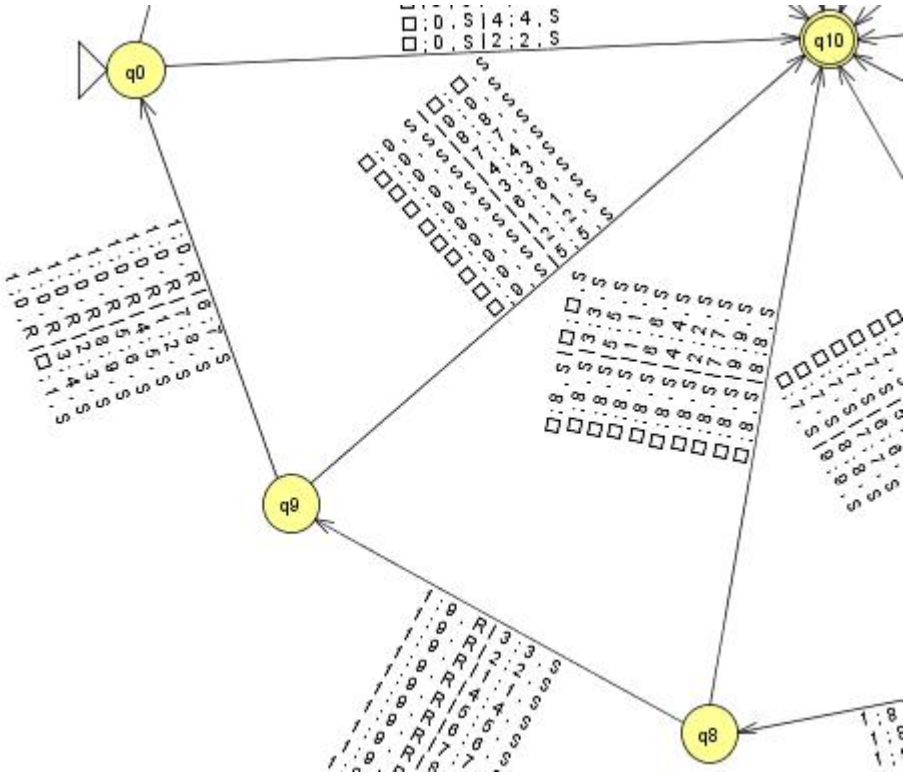


Figura 7.2.8: Estados 9 y 0.

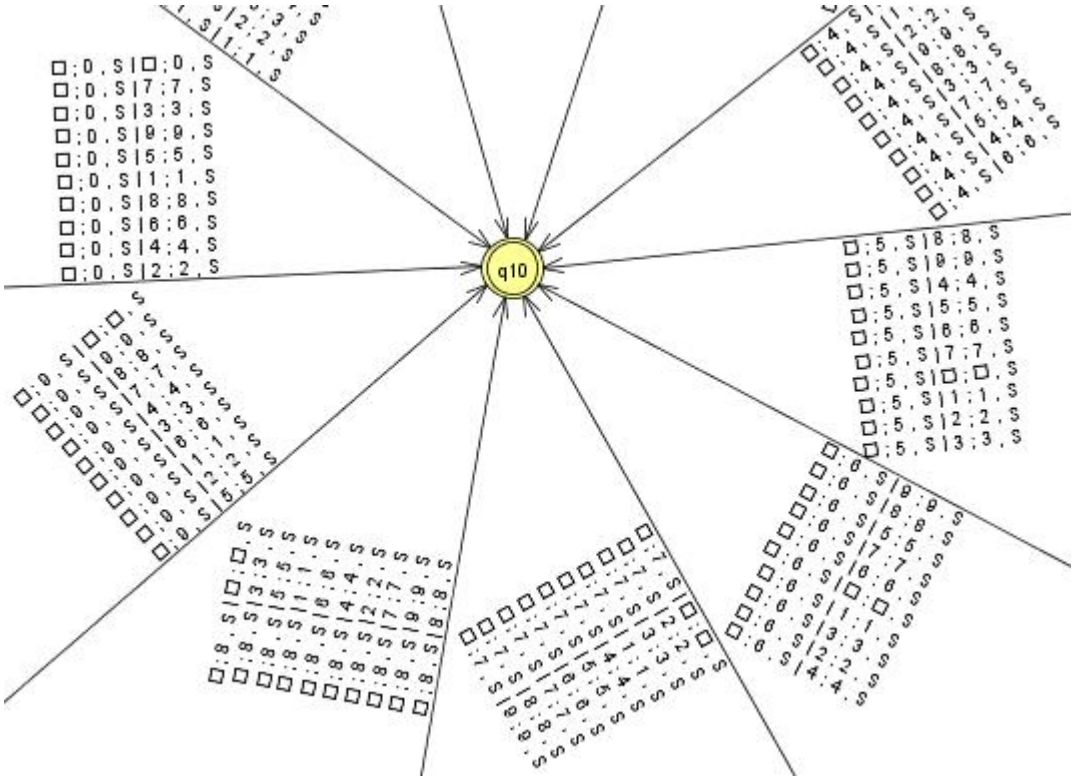


Figura 7.2.9: Estado 10.

7.3. Máquinas de Turing No Deterministas

En cualquier punto de un cálculo, la máquina puede proceder de acuerdo con varias posibilidades. Todos los elementos de una máquina no determinista son iguales a los elementos de una máquina determinista, con excepción de la función de transición. La función de transición para una máquina de Turing no determinista tiene la forma:

$$f\colon Q\,x\,\Gamma \longrightarrow P\,(Qx\Gamma x\{I,D,P\})$$

Donde $P\,(Qx\Gamma x\{I,D,P\})$ es el conjunto de partes del producto cartesiano de los conjuntos Q, Γ y $\{I,D,P\}$.

Recordamos que Q es el conjunto de estados de la Máquina de Turing, Γ es el conjunto del alfabeto de la cinta de la máquina, incluido el espacio en blanco. Y $\{I,D,P\}$ son los movimientos del cabezal de lecto escritura de la máquina, hacia la izquierda I, hacia la derecha D o parado P; después de leer una entrada, escribir y cambiar de estado.

El cómputo de una Máquina de Turing No Determinista es un árbol cuyas ramas corresponden a diferentes posibilidades para la máquina. Si alguna rama del cálculo lleva al estado

de aceptación, la máquina acepta su entrada. La anterior definición de Máquinas No Deterministas es tomada de Sipser (2006).

La Máquina No Determinista se ve obligada a considerar varias posibilidades en sus estados, alfabeto de entrada y movimientos laterales, hasta alcanzar su estado de aceptación, no es una cuestión de azar sino de ensayo y error. Por ello es muy diferente una Máquina No Determinista a una Máquina Probabilística.

El no determinismo no afecta el poder del modelo de las Máquinas de Turing, debido a que toda Máquina de Turing no determinista tiene una Máquina de Turing determinista equivalente (Sipser, 2006).

En modelos deterministas la máquina no puede estar en diferentes estados al mismo tiempo, en modelos no deterministas la máquina sí puede estar en múltiples estados al mismo tiempo (Martínez, 2016).

Ejemplo 9. Si tenemos un conjunto de enteros $\{a, b, c\}$, hallar un subconjunto tal que la suma de sus elementos de igual al entero 0.

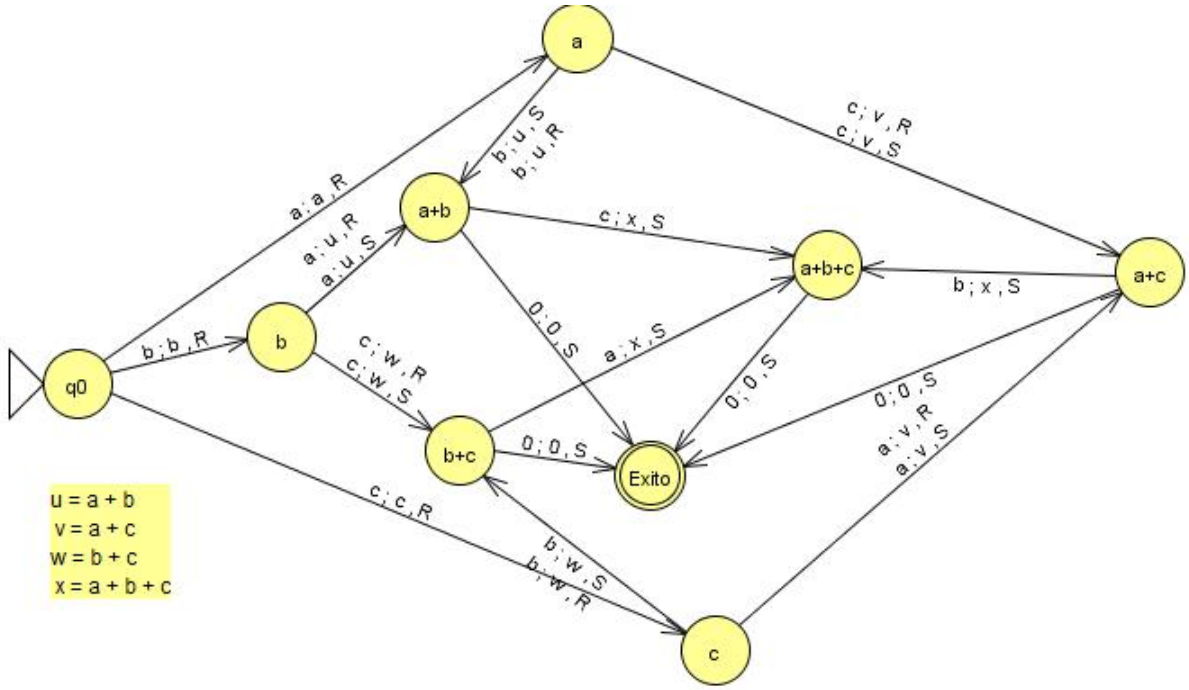


Figura 7.3.1: Máquina no determinista para determinar cuándo los subconjuntos de un conjunto de tres elementos suman cero.

Por ejemplo, sea el conjunto $H = \{-3, -1, +4\}$, debemos hallar el subconjunto cuya suma nos de el número cero. Para utilizar el diagrama anterior, sabemos que $H = \{a, b, c\}$

luego $a=-3$,$b=-1$,$c=+4$. Como sabemos, todos los suconjuntos de H forman el conjunto de partes de H, o sea $P(H)$ = {ϕ, {-3}, {-1}, {+4}, {-3, -1}, {-3, +4}, {-1, +4}, {-3, -1, +4}}, pero, como se trata de hacer la suma de los elementos de cada subconjunto, es necesario que elijamos los subconjuntos que tengan dos o más elementos. Por lo anterior, los subconjuntos que nos interesa son: $\{-3,-1\},\{-3,+4\},\{-1,+4\},\{-3,-1,+4\}$, para verlos como letras sería: $\{a,b\},\{a,c\},\{b,c\},\{a,b,c\}$. Otra situación a tener en cuenta es que la operación de la suma es conmutativa, por ello no importa el orden de los elementos: $ab=ba$,$ac=ca$,$bc=cb$,$abc=bca=bac=cba=cab=acb$. Luego ya hemos definido el alfabeto de entrada y el lenguaje que reconoce la máquina, nuestra máquina nunca va a reconocer las cadenas *aabc*, *cbab*, *bb*,*a*,*b*, por ejemplo. Después de tener claro el lenguaje, vamos a ensayar una cadena que pertenezca al lenguaje de la máquina, por ejemplo *cb*,comenzamos en un estado inicial q_0, la máquina cuando lee la entrada *c*,escribe la letra *c* en la cinta y mueve el cabezal de lectura hacia la derecha de la cinta, luego la flecha en el grafo nos indica que la máquina queda en el estado *c*, salta al estado *c*. Estando en el estado *c*,la máquina recibe la entrada *b*, recordemos que la cadena de entrada es *cb*. Pero en el estado *c*,la flecha que indica la entrada *b*,nos dice que hay dos funciones de transición para el mismo estado y la misma entrada, (b,w,S) y (b,w,R), la primera función nos dice que si ingresamos *b*,el cabezal de la máquina escribe en la cinta una $w=b+c$,y que la máquina se para $S=Stop$,esto se hace para crear un nodo de donde saldrán dos flechas, una para evaluar si la suma $w=b+c=0$, y otra para recibir otra letra de entrada, suponiendo que la cadena de entrada pueda tener hasta tres letras.

En la siguiente gráfica de Sipser (2006), observamos una diferencia entre la computación determinista y la no determinista:

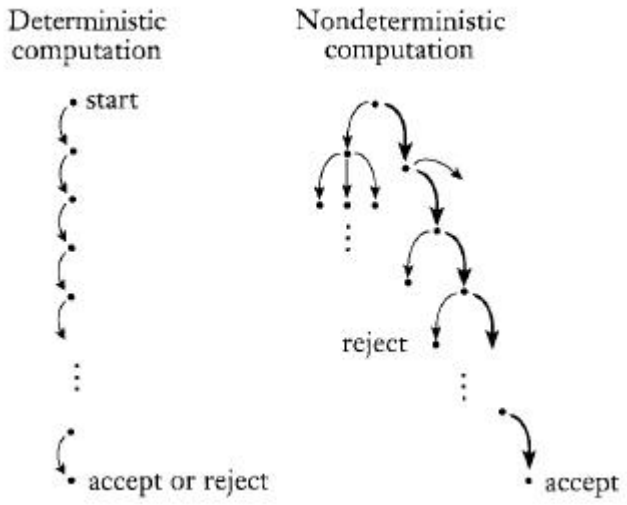


Figura 7.3.2: Cálculos deterministas y no deterministas con una rama de aceptación (Sipser, 2006)

Se observa que el árbol de decisión en modelos deterministas son lineales, en una sola dirección, como llevando una sola secuencia a la vez, mientras que el árbol de decisión en modelos no deterministas, sigue un árbol de decisión no lineal, en cada nodo se puede crear

otras ramas, y en cada rama puede crearse otro nodo con otras ramas, al final, en cualquiera de estas ramas puede haber una salida de aceptación o de rechazo. Observamos que el no determinismo no tiene nada que ver con el azar, las Máquinas de Turing no deterministas no son las mismas Máquinas de Turing probailísticas. En una máquina probabilística se toma una sola decisión en cada nodo, dependiendo de la probabilidad obtenida, mientras que en las máquinas no deterministas, en un nodo se toman todas las decisiones, el algoritmo se ejecuta en todas las ramas hasta que llegue a un estado de aceptación o de rechazo. Es como si una persona tomara un laberinto de tamaño humano, y llegara a un punto nodo o vértice del grafo o árbol, y en ese nodo haya por ejemplo, tres caminos diferentes a seguir, una «persona probabilística» acciona un mecanismo de azar (tirar un dado, el tin marin de do pin güe, lanzar una moneda, etc...) para decidir por cuál rama o camino seguir, mientras que la «persona no determinística» debe tomar los tres caminos al tiempo, para ello la «persona» se debe dividir en tres personas diferentes, y cada una seguir un camino de los tres posibles, todo el proceso en modelos no deterministas, termina cuando se ha encontrado la meta, o cuando se ha verificado que no es posible hallar dicha meta.

Ejemplo 10. Sea el alfabeto de entrada $\Sigma = \{0,1,2,3\}$, el conjunto de estados Q = $\{q_{inicial}$, q_1, q_2, q_3, $q_{final}\}$, el alfabeto admitido por la cinta $\Gamma = \{\square,0,1,2,3\}$, construir una Máquina de Turing que reconozca un lenguaje o cadenas formadas por el alfabeto Σ,de tal manera que algún número de la cadena, diferente de cero, indique la posición del cero, dicha posición del cero debe ser leído de derecha a izquierda. Por ejemplo, las siguientes cadenas sí pertenecen a ese lenguaje $013, 3012, 2301, 0213, \ldots$ y las siguientes cadenas no pertenecen a ese lenguaje $3210, 0, 2103, 0231, 2031, 1302, \ldots$.

En el siguiente gráfico queda construída la Máquina de Turing solicitada, es una máquina de reconocimiento de lenguaje.

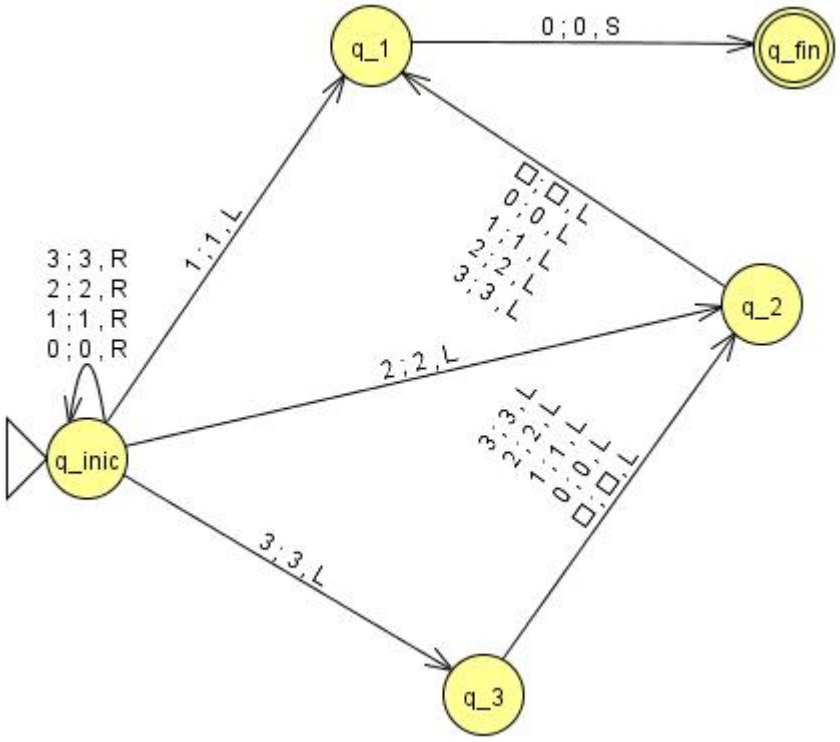


Figura 7.3.3: Máquina de Turing No Determinista, para reconocer un lenguaje específico.

Capítulo 8

Computación Clásica

La computación clásica funciona con compuertas lógicas, cuyo origen está solo en dos operaciones, prendido y apagado, o en términos del Álgebra de Boole ceros y unos. Sabemos que un Bit es la abreviación de Binary Digit (dígito binario), y que es la menor unidad de información de una computadora; un Bit es como una casilla de una cinta donde sólo cabe una sóla entrada, un cero o un uno, en un circuito lógico el cero representa que no hay paso de corriente, y el uno representa que sí hay paso de corriente; en términos lógicos, el cero representa un valor de verdad de «falso» y el uno representa un valor de verdad de «verdadero». Los símbolos y caracteres utilizados en nuestro computador clásico, están representados en la tabla ASCII (Código Estadounidense Estándar para el Intercambio de Información). Cada símbolo está representado por una cadena de ceros y unos, de longitud de 8 casillas, cada casilla (Bit) puede tomar un valor de cero o uno, a esta cadena de ocho Bit se le llama Byte; un ejemplo de un Byte podría ser: 01000000, el cual representa al número 64 en el sistema de numeración decimal, y combinando la tecla ALT simultáneamente con el número 64 nos daría: ALT+64 = @. Pero los Byte del código ASCII no sólo sirven para representar símbolos, también pueden representar palabras, por ejemplo, la siguiente palabra «Computación», quedaría representada así:

C = ALT + 67
o = ALT + 111
m = ALT + 109
p = ALT + 112
u = ALT + 117
t = ALT + 116
a = ALT + 97
c = ALT + 99
i = ALT + 105
ó = ALT + 162
n = ALT + 110

La computadora no interpreta la letra mayúscula C, ella sólo trabaja la cadena de ceros y unos de longitud ocho que representa a la letra mayúscula C. El Byte que representa a la letra mayúscula C es 01000011. Por lo anterior, cuando escribimos con el teclado «Computación»,

la computadora tendrá en su memoria las siguientes cadenas de ceros y unos:

C = ALT + 67 = 01000011
o = ALT + 111 = 01101111
m = ALT + 109 = 01101101
p = ALT + 112 = 01110000
u = ALT + 117 = 01110101
t = ALT + 116 = 01110100
a = ALT + 97 = 01100001
c = ALT + 99 = 01100011
i = ALT + 105 = 01101001
ó = ALT + 162 = 10100010
n = ALT + 110 = 01101110

Por lo anterior, una cadena de texto que represente a la palabra «Computación», quedará representada en la computadora con la siguiente cadena de ceros y unos, agrupando a su vez a 11 cadenas de ocho bits cada una.

«Computación» = «01000011 01101111 01101101 01110000 01110101 01110100 01100001 01100011 01101001 10100010 01101110»

Sin embargo, si vamos a representar una variable numérica, por ejemplo 762.189, quedaría representado en el número binario: 10111010000101001101, el cual tiene veinte bits y no está agrupado en paquetes de 8 bits (Grupo de Computación Cuántica, 2003).

Para hacer operaciones en la computación clásica, por ejemplo sumar dos números binarios, utilizamos compuertas lógicas.

8.1. Compuertas de la Lógica Clásica

8.1.1. Compuerta Si o Buffer

Esta compuerta es similar a un cable continuo o un interruptor, no afecta el paso de corriente o de señal, si entra un uno saldrá un uno, si entra un cero saldrá un cero. Si entra corriente sale corriente, si no entra corriente entonces tampoco sale corriente, tiene una sola entrada y una sola salida.

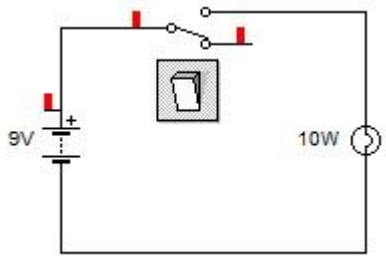


Figura 8.1.1: Circuito eléctrico abierto, para la compuerta «Si».

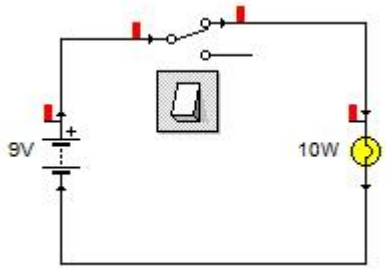


Figura 8.1.2: Circuito eléctrico cerrado, para la compuerta «Si».

Entrada «A»	Salida «Si A»
1	1
0	0

Cuadro 8.1: Tabla de verdad para la compuerta «Si».

8.1.2. Compuerta No, Not, Inversor, ¬

Esta compuerta invierte el valor de la entrada, da como resultado el complemento o la negación. Si entra un uno la salida será un cero, y si entra un cero la salida será un uno.

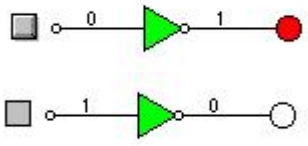


Figura 8.1.3: Representación gráfica de la compuerta «No».

Entrada «A»	Salida «No A»
1	0
0	1

Cuadro 8.2: Tabla de verdad para la compuerta «No».

En electrónica el símbolo de la compuerta NOT es:

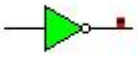


Figura 8.1.4: Símbolo gráfico de la compuerta «No».

8.1.3. Compuerta AND, Y, ∧

En términos de la lógica clásica es la equivalente del conectivo lógico «Y», o también conocida como Conjunción. En este circuito sólo pasa corriente si las dos entradas tienen corriente.

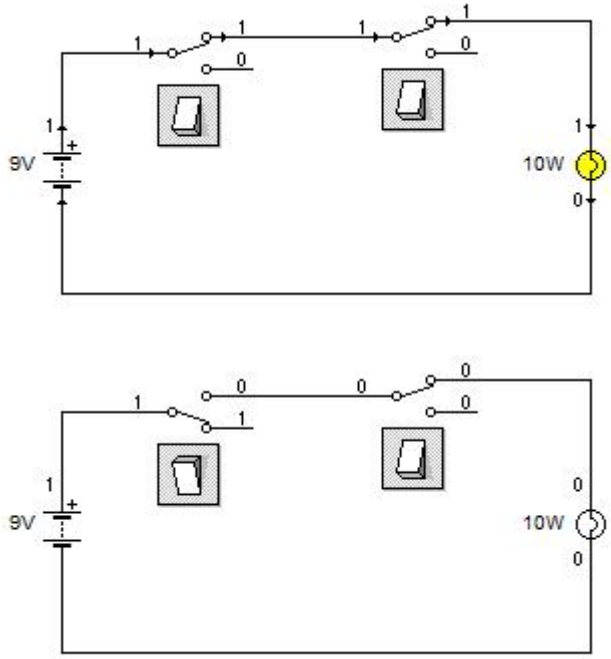


Figura 8.1.5: Circuito eléctrico para la compuerta «Y».

Entrada «A»	Entrada «B»	Salida «A Y B»
1	1	1
1	0	0
0	1	0
0	0	0

Cuadro 8.3: Tabla de verdad para la compuerta «Y».

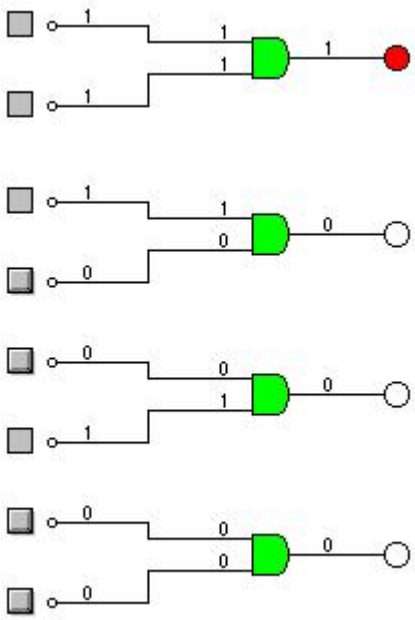


Figura 8.1.6: Representación gráfica de la compuerta «Y».

En electrónica el símbolo de la compuerta AND es:

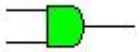


Figura 8.1.7: Símbolo gráfico de la compuerta «Y».

8.1.4. Compuerta OR, O, $\vee$

En términos de la lógica clásica es la equivalente del conectivo lógico «O», también conocida como Disjunción Inclusiva. En este circuito sólo pasa corriente si las dos entradas tienen corriente o una de las dos.

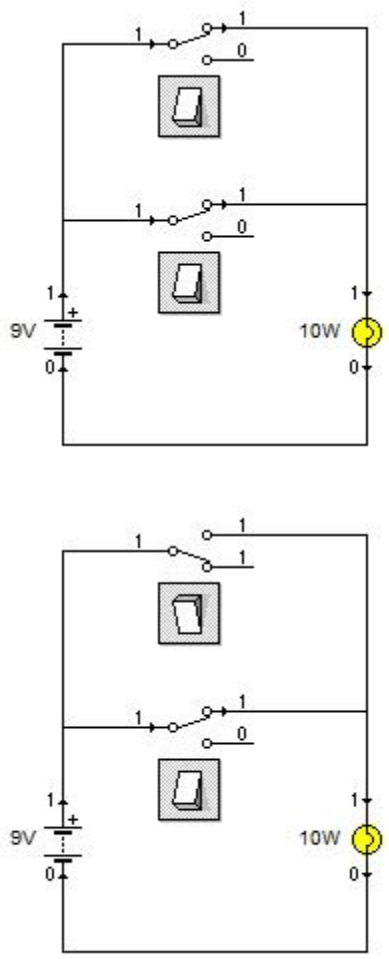


Figura 8.1.8: Circuito eléctrico cerrado, para la compuerta «O».

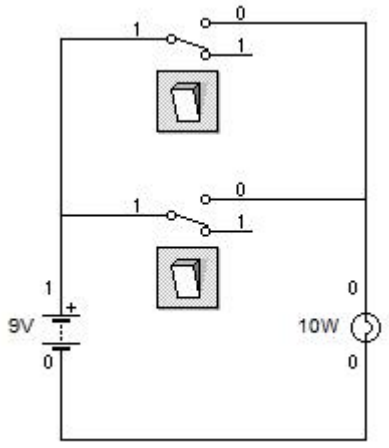


Figura 8.1.9: Circuito eléctrico abierto, para la compuerta «O».

Entrada «A»	Entrada «B»	Salida «A O B»
1	1	1
1	0	1
0	1	1
0	0	0

Cuadro 8.4: Tabla de verdad para la compuerta «O»

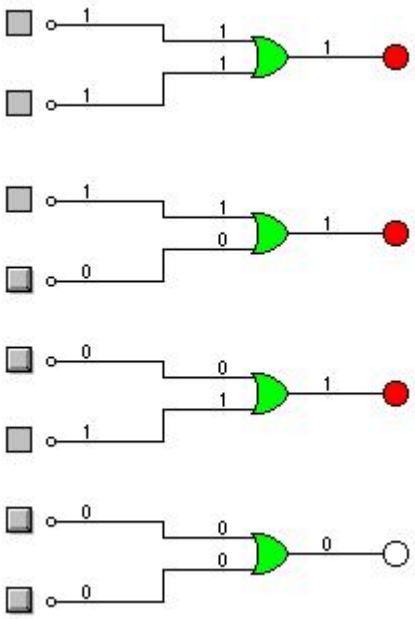


Figura 8.1.10: Representación gráfica para la compuerta de la «O».

En electrónica el símbolo de la compuerta OR es:

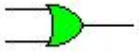


Figura 8.1.11: Símbolo gráfico de la compuerta «O».

8.1.5. Compuerta XOR, Ô, ⊻

En términos de la lógica clásica es la equivalente del conectivo lógico «O - exclusiva», también conocida como Disjunción Exclusiva. En este circuito sólo pasa corriente si sólo una de las dos entradas tiene corriente, pero no ambas.

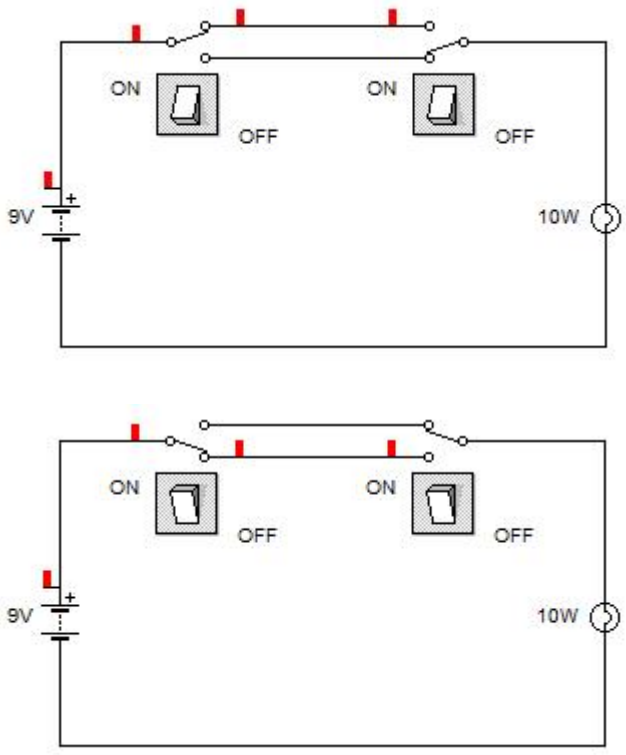


Figura 8.1.12: Circuito eléctrico abierto, para la compuerta de la «O exclusiva».

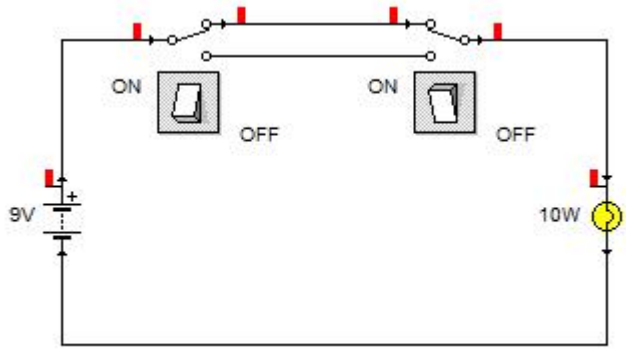


Figura 8.1.13: Circuito eléctrico cerrado, para la compuerta de la «O exclusiva».

Entrada «A»	Entrada «B»	Salida «A Ô B»
1	1	0
1	0	1
0	1	1
0	0	0

Cuadro 8.5: Tabla de verdad para la compuerta de la «O exclusiva».

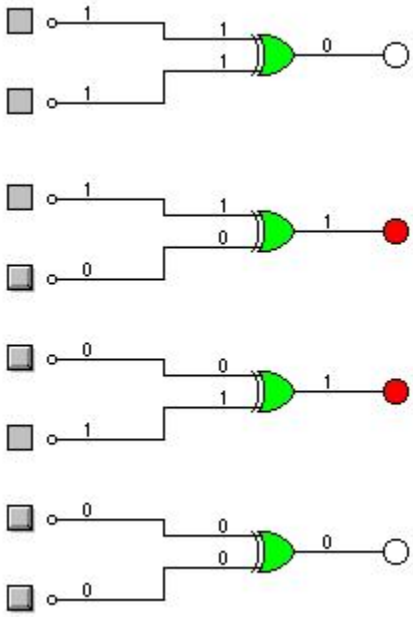


Figura 8.1.14: Representación gráfica para la compuerta de la «O exclusiva».

En electrónica el símbolo de la compuerta XOR es:

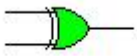


Figura 8.1.15: Símbolo gráfico de la compuerta «O exclusiva».

8.1.6. Negación o complemento de compuertas

Para cada una de las compuertas lógicas también existe su negación o complemento, su circuito lógico, la tabla de verdad y su símbolo lógico en la electrónica. A continuación se expone su nombre y su símbolo lógico, y se deja al lector investigar lo que considere necesario.

Figura 8.1.16: Símbolo gráfico de la compuerta «NOT».

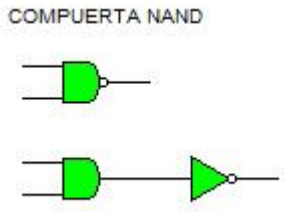


Figura 8.1.17: Símbolo gráfico de la compuerta «NAND».

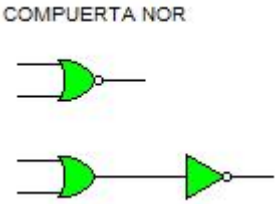


Figura 8.1.18: Símbolo gráfico de la compuerta «NOR».

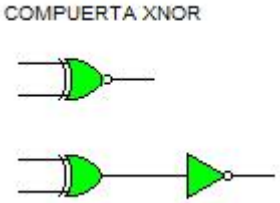


Figura 8.1.19: Símbolo gráfico de la compuerta «XNOR».

8.1.7. Suma de números binarios utilizando compuertas lógicas

El siguiente circuito lógico, representa un algoritmo, o trozo de código que utilizado indefinidamente puede hallar la suma de cualesquier dos cadenas de ceros y unos.

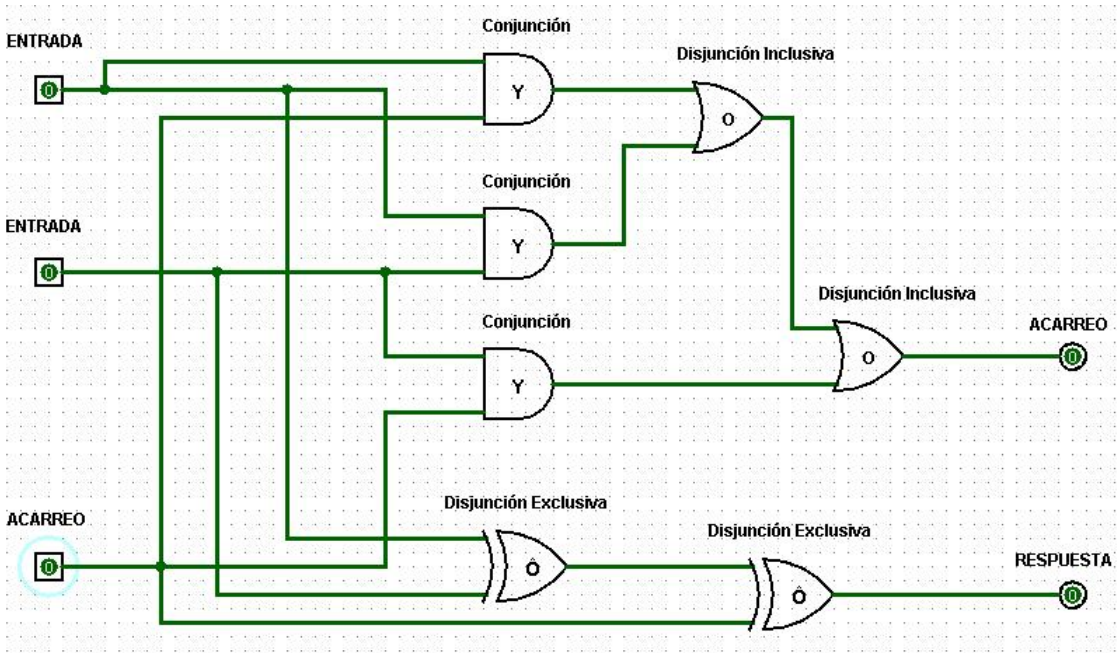


Figura 8.1.20: Sumador de números binarios utilizando compuertas lógicas.

Ejemplo 11. Sumar los números binarios 101 y 11, utilizando un circuito de compuertas lógicas.

$$\begin{array}{rccc} & 1 & 0 & 1 \\ + & & 1 & 1 \\ \hline \end{array}$$

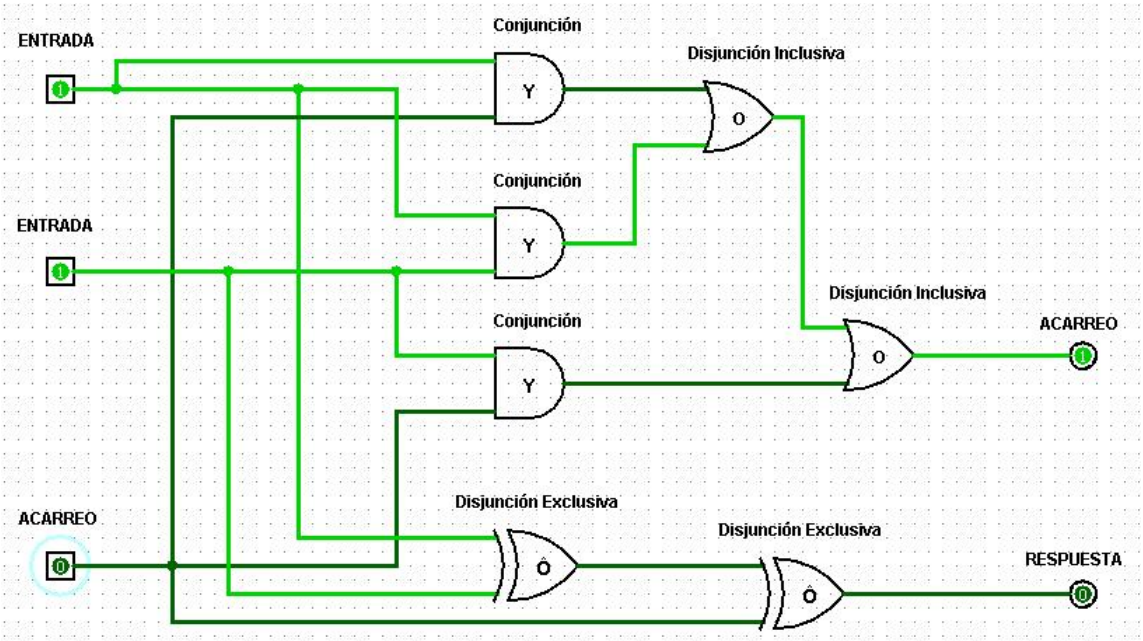


Figura 8.1.21: Circuito para la suma de los números binarios 1 + 1, con acarreo de entrada 0.

La figura 8.1.21 muestra que si entra 1 y 1, con acarreo 0, la primera respuesta es 0 con acarreo 1.

$$\begin{array}{rccc} & 1 & 0 & 1 \\ + & & 1 & 1 \\ \hline & & & 0 \end{array}$$ 1 + 1 = 0, acarreo 1.

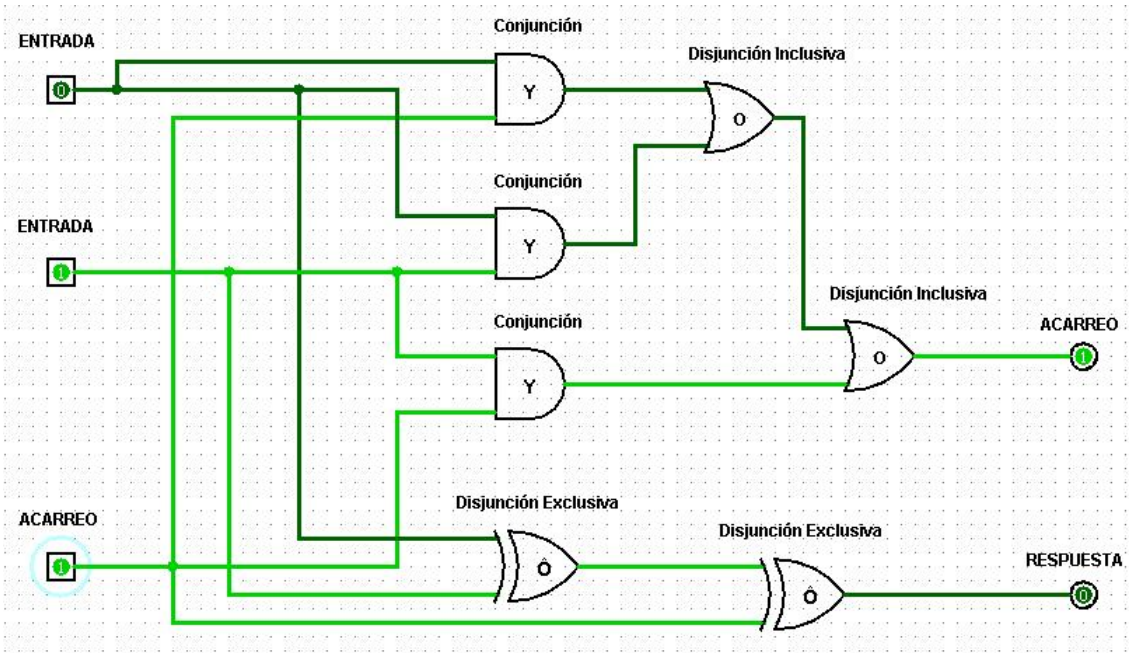


Figura 8.1.22: Circuito para la suma de los números binarios 0 + 1, con acarreo de entrada 1.

La figura 8.1.22 muestra que si entra 0 y 1, con acarreo 1, la segunda respuesta es 0 con acarreo 1.

$$\begin{array}{rccc} & 1 & 0 & 1 \\ + & & 1 & 1 \\ \hline & & 0 & 0 \end{array}$$ 0 + 1 = 0, acarreo 1.

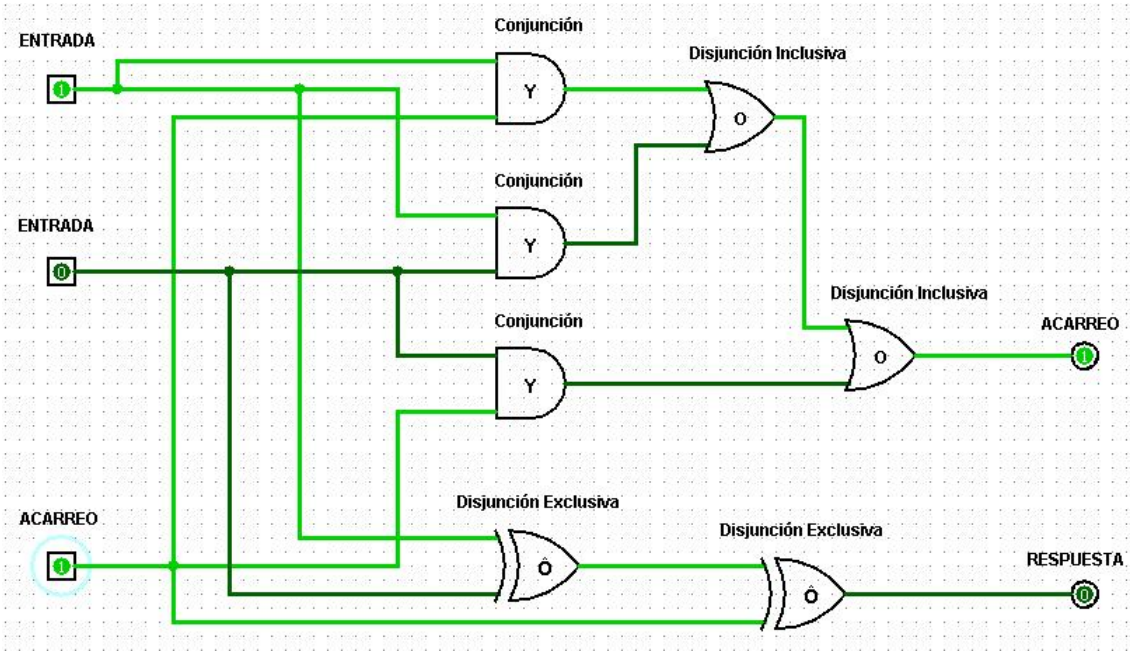


Figura 8.1.23: Circuito para la suma de los números binarios 1 + 0, con acarreo de entrada 1.

La figura 8.1.23 muestra que si entra 1 y 0, con acarreo 1, la tercera respuesta es 0 con acarreo 1.

$$\begin{array}{cccc} & 1 & 0 & 1 \\ + & & 1 & 1 \\ \hline 1 & 0 & 0 & 0 \end{array}$$

1 + 0 = 0, acarreo 1. Pero como ya no hay más dígitos en los sumandos, entonces el acarreo se coloca en la casilla siguiente del resultado.

En un circuito eléctrico o lógico, las compuertas lógicas son interruptores, ubicados de tal manera que realizan operaciones lógicas. Estas operaciones obedecen a la lógica clásica de verdadero y falso, en matemáticas también le llamamos Ágebra de Boole. En la práctica un interruptor es remplazado por un transistor. En electrónica sabemos que un transistor tiene dos usos básicos, uno como amplificador de una señal eléctrica y otro como interruptor de una corriente. Para el caso de este documento, podemos utilizar los transistores NPN como interruptores, cada uno de ellos tiene tres pines, uno llamado base, otro colector y otro emisor. Para que un transistor NPN no interrumpa el paso de una corriente desde el pin colector hacia el pin emisor, se le debe ingresar una corriente o señal de habilitación por el pin de base, a esto se le llama estado de saturación del transistor. Si por el pin de base no le ingresamos una corriente o señal de habilitación, entonces el transistor se cierra, no permitiendo el paso de corriente desde el pin colector hacia el pin emisor, a esto se le llama estado de corte del transistor. En este documento no se profundiza en el conocimiento de los transistores o componentes electrónicos, debido a que el objetivo es exponer la lógica computacional clásica dentro de circuitos eléctricos.

Gracias a la invención de los transistores y al descubrimiento de los semiconductores, combinando la física de materiales con la informática teórica, fue que se pudo inventar el computador. Toda la teoría estaba lista, sólo se necesitaban elementos físicos que pudieran representar en la práctica la lógica computacional. Es lo mismo que tenemos ahora con la computación cuántica, estamos avanzados en lógica cuántica pero no hemos podido avanzar mucho en hardware y algoritmos cuánticos.

Capítulo 9

Complejidad

9.1. ¿Qué es la Complejidad?

No existe una sola definición de complejidad, Maldonado (2010). Para Maldonado (2010), existen tres ejes que conciernen específicamente a las ciencias de la complejidad. Estos ejes son, sin importar el orden:

1. La teoría matemática de la complejidad; la cual trata de los problemas P vs NP.
2. Las relaciones entre el universo microscópico y el universo macroscópico; estas se refieren al tema de una de las lógicas no clásicas, la lógica cuántica.
3. La teoría de los sistemas dinámicos.

En cuanto a una definición precisa de complejidad Zoya (2014), establece que «este espacio controversial se encuentra integrado por tres enfoques teórico-metodológicos primordiales: La propuesta del pensamiento complejo formulada por Edgar Morin, la teoría de los sistemas complejos elaborada por Rolando García, y el enfoque de las así llamadas "ciencias de la complejidad" o "ciencias de los sistemas complejos"».

Zoya (2014), relata que Maldonado (1999), ha planteado tres líneas en la comprensión de la complejidad: La complejidad como método, la complejidad como cosmovisión, y la complejidad como ciencia o ciencias de la complejidad. Para Zoya (2014), Maldonado (2001), identifica la obra de Edgar Morin y su propuesta del pensamiento complejo en la línea de la complejidad como método, en otras palabras como una «hermenéutica», una «filosofía del sujeto» y en el límite de «una filosofía de la conciencia». Además Zoya (2014), nos dice que Maldonado (2001), cree que el pensamiento complejo es más que un método, nos cita que es una «actitud general hacia el mundo, la naturaleza, la vida, y hacia el propio conocimiento». En la segunda línea, la complejidad como cosmovisión, Zoya (2014), nos dice que Maldonado la identifica con el pensamiento sistémico, y que para Maldonado esta línea la desarrolla el físico Capra, Bateson y la Escuela de Palo Alto. Para el tercer enfoque, Zoya (2014), nos dice que Maldonado se refiere a sistemas complejos no lineales y a sistemas dinámicos no lineales.

Según Zoya (2014), Maldonado (2007) propone demarcar dos formas básicas de entender el término complejidad, estas dos formas son: el pensamiento complejo de Edgar Morin, y

las ciencias de la complejidad o sistemas complejos adaptativos.

Para Zoya (2014), Carlos Reynoso propone dos grandes categorías para entender el término de complejidad, estas son: los paradigmas globales de la complejidad y los algoritmos de la complejidad. Zoya (2014), nos cuenta que, bajo la categoría de «los paradigmas globales de la complejidad», Reynoso ubica, además de la propia obra de Morin, a la teoría de la información, la teoría general de los sistemas, la cibernética, la teoría de las catástrofes, la teoría de las estructuras disipativas, la autopoiesis, entre otras; y que para Reynoso (2006), en esta categoría se agrupan grandes construcciones filosóficas sin demasiado sustento experimental. Zoya (2014), nos cuenta además que, bajo la categoría de «los algoritmos de la complejidad» según Reynoso, está el estudio de los sistemas complejos, los autómatas celulares, las redes booleanas aleatorias, los modelos basados en agentes, el algoritmo genético, y demás técnicas de modelación. Estos algoritmos se encuentran implementados en lenguaje formal, sea éste matemático o computacional. Así con las dos categorías de Reynoso, se plantea un lenguaje natural o teorías discursivas de la complejidad, y un lenguaje formal o práctico (Zoya, 2014).

Para Zoya (2014), hay una controversia en el alcance del término complejidad, «no por sus límites sino por la clase de problemas que abarca», de esa manera queda al descubierto dos formas de entender la complejidad, una desde el pensamiento complejo de Edgar Morin y la otra desde «las llamadas ciencias de la complejidad», según Zoya (2014), Morin hizo una distinción entre las dos formas de abordar la complejidad, a su teoría del pensamiento complejo le llamó «Complejidad General» y a la teoría de la complejidad desde «las ciencias de la complejidad», le llamó «Complejidad Restringida» (Morin, 2005).

Morin (2005), reconoce tres nociones que están presentes en la complejidad restringida, la noción de sistema, emergencia y caos, no obstante, considera que la complejidad restringida sólo busca crear leyes de complejidad, avanzar en la ciencia, evitando lo fundamental, que para este autor es lo epistemológico, lo cognitivo y lo paradigmático.

«La complejidad restringida ha permitido realizar importantes avances en la formalización, en las posibilidades de modelado, que por sí mismas promueven el potencial interdisciplinario. Pero permanecemos en la epistemología de la ciencia clásica. Cuando buscamos "leyes de complejidad", todavía colgamos la complejidad como una especie de vagón detrás de la locomotora real, la que produce leyes. Formó un híbrido entre los principios de la ciencia clásica y los avances hacia su más allá. En realidad, evitamos el problema fundamental de la complejidad que es epistemológico, cognitivo, paradigmático. En cierto modo, reconocemos la complejidad, pero descomplejándola. Como resultado, abrimos la brecha, luego intentamos taparla: el paradigma de la ciencia clásica permanece, solo resquebrajado» (Morin, 2005).

Para Morin (2005), un replantamiento epistemológico es una organización del conocimiento, un replanteamiento paradigmático es un modelo o un método que tiene la ciencia y debe ser replanteado, por ejemplo, Morin considera que la ciencia clásica está controlada por la simplificación y la disyunción del conocimiento, esto quiere decir que el método de la ciencia clásica es reduccionista, determinista y no hay integración entre sus disciplinas, por

lo que Morin propone un paradigma de complejidad para la ciencia, esto quiere decir que la ciencia debe ser modelada por un principio de separacíon entre sus disciplinas, pero que, trate de encontrar la relación entre cada una de ellas.

«Dado que un paradigma de simplificación controla la ciencia clásica, al imponer un principio de reducción y un principio de disyunción a todo conocimiento, debería existir un paradigma de complejidad que imponga un principio de distinción y un principio de conjunción.» (Morin, 2005).

Respecto al principio de disyunción en la ciencia clásica, (Morin, 2005) nos dice que es la separación de disciplinas sin buscar un principio de unificación.

«El principio de disyunción, que consiste en aislar y separar las dificultades cognitivas entre sí, lo que ha llevado a la separación entre disciplinas que se han vuelto herméticas entre sí» (Morin, 2005).

Lo que Morin (2005) busca es desligarse de la ciencia clásica, de lo que considera que es el método de la ciencia clásica, método que lo resume en tres principios: 1) El principio del determinismo universal, conocer un evento del pasado y predecir un evento del futuro. 2) El principio de reducción, que consiste en conocer el todo a partir de sus partes. 3) El principio de disyunción, que consiste en aislar y separar las dificultades cognitivas entre sí, llevando a la separación del conocimiento a través de disciplinas.

En la medida en que Morin (2005) busca desligarse de la ciencia clásica, tiene el objetivo de buscar un conocimiento distinguido pero integrador, en palabras de Morin (2005), «*un principio de distinción y un principio de conjunción*». Para Morin (2005), «los fenómenos se presentan de manera confusa e incierta, pero la misión de la ciencia es descubrir, detrás de estas apariencias, el orden oculto que es la auténtica realidad del universo». Para Morin (2005), «la complejidad es invisible en la división disciplinaria de la realidad».

Los principios de la complejidad generalizada, o también llamada pensamiento complejo, tiene los principios antagónicos a los principios que describe Morin (2005) sobre de la ciencia clásica. Estos principios de la complejidad generalizada son para Morin (2005):

1. Principio de implicación mutua de todas las partes; contrario al principio de reducción.

2. Principio de distinción y conjunción; contrario al principio de disyunción, debe mantener la distinción entre los agentes del conocimiento pero, debe tratar de establecer la relación entre ellos.

3. Principio de concepción de una relación entre orden, desorden y organización; contrario al principio de determinismo universal, en mi opinión, no es la búsqueda de un resultado, de un valor o parámetro, es la búsqueda de un equilibrio o de una armonía entre las partes.

En definitiva, se puede creer que la idea de Morin sobre pensamiento complejo, o de lo que él mismo llama complejidad general, además de ser una filosofía del pensamiento, también

es una teoría del conocimiento, en palabras de Morin (2005), «el conocimiento del conocimiento es un requisito del pensamiento complejo».

«Además, creo que será necesario llegar cada vez más a un conocimiento científico que integre el conocimiento del espíritu humano con el conocimiento del objeto que este espíritu capta y reconozca la inseparabilidad del objeto y el sujeto» (Morin, 2005).

Para Morin (2005), la diferencia entre la complejidad restringida y la complejidad general es su objeto de estudio, para él la complejidad restringida se ocupa principalmente de los sistemas dinámicos llamados complejos, nos dice que este campo está dentro de la ciencia, por ello él le llama Ciencias de la Complejidad. Para Morin (2005), «la complejidad general no sólo abarca el campo de la complejidad restringida, sino también a nuestro conocimiento como ser humano, individuo, persona y ciudadano».

Morin (2005), fija una ruptura entre la complejidad restringida y la complejidad general, desde la primera hacia la segunda; él argumenta que la complejidad general sí reconoce y contiene a la complejidad restringida, pero que es la complejidad restringida la que rechaza la complejidad general.

«Lamentablemente, la complejidad restringida rechaza la complejidad generalizada, que le parece pura charlatanería, pura filosofía. La rechaza porque no ha hecho la revolución epistemológica y paradigmática que obliga la complejidad. Sin duda esto sucederá. Pero mientras tanto, vemos que la problemática de la complejidad ha invadido todos nuestros horizontes y repito "problemática", porque es un error pensar que encontraremos en la complejidad un método que podamos aplicar automáticamente en el mundo y sobre todo» (Morin, 2005).

9.2. Ciencias de la Complejidad

Es interesante abordar la definición de sistema complejo que se plantea a continuación; «Un sistema compuesto por un número (generalmente grande) de entidades, procesos o agentes que interactúan (generalmente fuertemente), cuya comprensión requiere el desarrollo o el uso de nuevas herramientas científicas, modelos no lineales, descripciones fuera de equilibrio y simulaciones en computadoras» (Advances in Complex Systems Journal, como se citó en Petty, 2014). Se estudia cómo las partes del sistema interactúan para constituir el sistema, y cómo este sistema interactúa con su entorno.

Las ciencias de la complejidad son fundamentalmente el resultado del desarollo del computador, obedecen a la existencia y al trabajo con computación, y contribuyen, a su vez, al desarrollo de la computación y en general de los sistemas informáticos (Maldonado, 2020). Maldonado (2010), también enfatiza que las ciencias de la complejidad son el fundamento del modelamiento y la simulación de sistemas, fenómenos y comportamientos complejos. Y, a través de la complejidad, las lógicas no clásicas estudian los sistemas dinámicos no lineales, el modelamiento y la simulación. Maldonado (2010), se refiere a algunas lógicas no clásicas

como: la lógica difusa, la lógica de la relevancia, la lógica paraconsistente, la lógica libre, la lógica epistémica, la lógica temporal, la lógica cuántica, la lógica intuicionista y la lógica de fábrica (o fabricación). También se refiere a que la lógica difusa es la única lógica no clásica que se ha podido incorporar en la computación, desde el punto de vista informacional, algorítmico y computacional; siendo el trabajo de incorporar las demás lógicas en la computación, una tarea abierta y de grandes avances recientes.

El objetivo de la complejidad es transformar la linealidad en no linealidad, cuando sea posible, esto acarrea una transformación radical del fenómeno estudiado (Maldonado, 2010).

La teoría de la complejidad trata la complejidad de algoritmos, del estudio de la solución de las clases o familias de algoritmos, y el problema más importante no resuelto de la teoría de la complejidad, es determinar si es posible resolver con cualquier dispositivo similar a una Máquina de Turing, un problema NP Completo en tiempo polinómico y por consiguiente determinar si NP es igual a P (Penrose, 1996).

Por lo anterior podemos concluir que la teoría de la complejidad está muy estrechamente relacionada con la teoría de la complejidad computacional.

Es posible construir una computadora cuántica que resuelva clases de problemas no polinomiales en tiempos polinomiales, o sea que en la computación cuántica, la cual se rige por la lógica cuántica, algunos problemas NP se convertirían en problemas de tipo P (Deutsch, 1985, citado por Penrose, 1996). A la fecha la computación cuántica está en pleno desarrollo, aunque todavía no ha llegado la supremacía cuántica, está vigente la aseveración de Penrose (1996), de que «subsiste la posibilidad teórica de que un dispositivo físico cuántico mejoraría una Máquina de Turing». Lo cual significa que un problema que no pueda resolver un computador clásico, bajo la lógica clásica, sí lo podría resolver un computador cuántico, bajo la lógica cuántica, esto se conoce también como supremacía cuántica.

Capítulo 10

Teoría Matemática de la Información

10.1. Probabilidad frecuentista

En la definición de probabilidad frecuentista, nos dice que si tenemos una cantidad «N» de resultados posibles, y cada resultado no puede suceder simultáneamente con otro, es decir que sólo podrá ocurrir un sólo evento o resultado del conjunto del espacio posible (espacio muestral), sabemos que la suma de todas las probabilidades de ocurrencia da el número uno. Por ejemplo; lancemos un dado, el espacio muestral, o posibles resultados son: $S = 1,2,3,4,5,6$, observamos que hay seis posibles resultados o valores que pueden salir, al lanzar un dado una sola vez. Si alguien preguntara cuál es la probabilidad de que salga el número cinco, como hay un sólo cinco y hay seis posibles resultados, la probabilidad sería $1/6$. Y así nos damos cuenta de que cada número tiene la misma probabilidad de salir. Si otra persona preguntara cuál es la probabilidad de que al lanzar un solo dado salga el número siete, observamos que no hay ese número dentro de las posibilidades o espacio muestral, por ello la probabilidad sería $\frac{0}{6} = 0$. La suma de todas las probabilidades nos daría $1/6+1/6+1/6+1/6+1/6+1/6 = 1$.

Supongamos un ejemplo donde cada evento o resultado tenga diferentes probabilidades de salir, por ejemplo, una bolsa con quince bolas de colores, dentro de la bolsa hay dos rojas, una azul, cinco amarillas y siete verdes. Si alguien preguntara cuál es la probabilidad de que al meter la mano se saque la bola verde, la respuesta sería $\frac{7}{15}$, ya que hay siete bolas verdes dentro de un conjunto posible de quince bolas como tamaño muestral.

S = {Roja, Roja, Azul, Amarilla, Amarilla, Amarilla, Amarilla, Amarilla, Verde, Verde, Verde, Verde, Verde, Verde, Verde}

La suma de todas las probabilidades para ese ejemplo, sería: $2/15 + 1/15 + 5/15 + 7/15 = 1$.

Podemos utilizar la notación $P(x_i)$, para representar la probabilidad de obtener el resultado o evento x_i . Por ejemplo, en el caso del dado, la probabilidad de obtener el número tres, es: $P(3) = 1/6$. En el caso de la bolsa con quince bolas de colores, la probabilidad de obtener una bola amarilla, es: $P(Amarilla) = 5/15 = 1/3$. Podemos concluir que la probabilidad frecuentista, es la división entre el número de casos favorables del espacio muestral, y

el número total de casos posibles.

$$P(x_i) = \frac{Número\ de\ casos\ favorables\ de\ x_i}{Número\ total\ de\ casos\ posibles}$$

10.2. Probabilidad condicional

La probabilidad condicional se refiere a encontrar la probabilidad de ocurrir un evento A, sabiendo que ya ha ocurrido otro evento B. Sea la notación:

$P(A|B) = Probabilidad\ de\ que\ ocurra\ el\ evento\ A,\ sabiendo\ que\ ya\ ha\ ocurrido\ el\ evento\ B.$

Como sabemos que el evento B ya ha ocurrido, entonces la probabilidad $P(A|B)$, es la división entre los casos favorables, que en este caso es la probabilidad de que ocurran simultáneamente los eventos A y B, dividido por la probabilidad total, que en este caso sabemos que es $P(B)$.

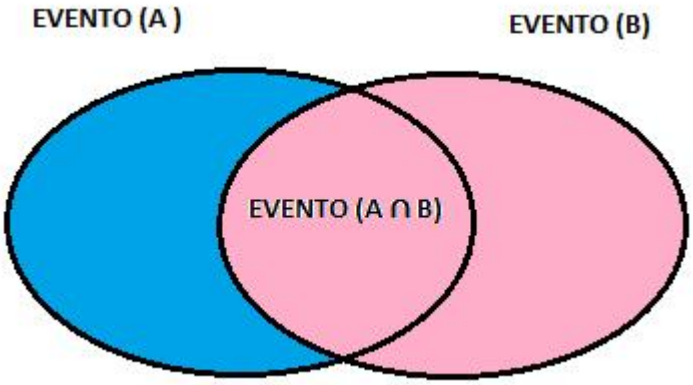


Figura 10.2.1: Probabilidad Condicional.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Ejemplo 12. Se lanza un dado y se sabe que el resultado es un número par. ¿Cuál es la probabilidad de que ese número sea divisible por el número tres?

El espacio muestral es S = {1, 2, 3, 4, 5, 6}.

EL evento de ser par es del conjunto B = {2, 4, 6}, luego P(B) = 3/6 = 1/2, debido a que el espacio muestral es el conjunto S.

El evento de ser divisible por el número tres es del conjunto A = {3, 6}.

Sabemos que A∩B = {6}, luego P(A∩B) = 1/6, debido a que el espacio muestral es el conjunto S.

Aplicando la fórmula: $P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{1/6}{3/6} = \frac{1}{3}$.

Tenemos que la probabilidad de que al lanzar una dado, el resultado sea un número divisible por tres, sabiendo que ha salido un número par, es de 1/3.

No es necesario aplicar siempre la fórmula de probabilidad condicional, en muchas ocasiones es más práctico desarrollar el ejercicio de manera intuitiva, como debería ser más útil. El espacio muestral restringido es de tres elementos, ya que ha salido un número par, y los números pares en el dado son sólo tres. Los números divisibles por tres son dos números, pero de esos sólo uno es par, luego, la respuesta tendría que ser 1/3.

10.3. Teorema de Bayes

El teorema de Bayes corresponde a tener la probabilidad condicional en ambos sentidos, por ejemplo:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \text{ y } P(B|A) = \frac{P(B \cap A)}{P(A)}$$

Despejando la ecuación $P(B|A) = \frac{P(B \cap A)}{P(A)}$, obtenemos:

$$P(B|A).P(A) = P(B \cap A)$$

Como sabemos que $P(A \cap B) = P(B \cap A)$, luego:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A).P(A)}{P(B)}$$

Por lo tanto se obtiene el famoso teorema de Bayes:

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

Para un conjunto de eventos A_i, y un evento que ya ha ocurrido B, sabiendo que los eventos A_i son mutuamente excluyentes (la intersección entre ellos es vacía), y además la unión de todos los eventos A_i conforman todo el conjunto del espacio muestral, o sea que A_i es una partición del espacio muestral, tenemos el siguiente teorema de Bayes generalizado:

$$P(A_i|B) = \frac{P(B|A_i).P(A_i)}{\sum_{k=1}^{n} P(B|A_k).P(A_k)}$$

10.4. Problema de Monty Hall

Un problema muy famoso de probabilidad que nos ha confundido desde su publicación, en un programa de la televisión estadounidense, el cual se emitió entre 1963 y 1986, cuyo nombre proviene del presentador de dicho programa, es el problema de Monty Hall. Se trata de un presentador que muestra al concursante tres puertas cerradas, sólo el presentador sabe dónde está el premio. El presentador le da a escoger al concursante una puerta, después que que el concursante elige, el presentador no abre dicha puerta, sino que le abre otra puerta donde él sabe que no está el premio, y le pregunta al concursante que si desea cambiar de opinión, que si desea elegir otra puerta. La pregunta es si la probabilidad de ganar del concursante, aumenta o disminuye al cambiar de puerta.

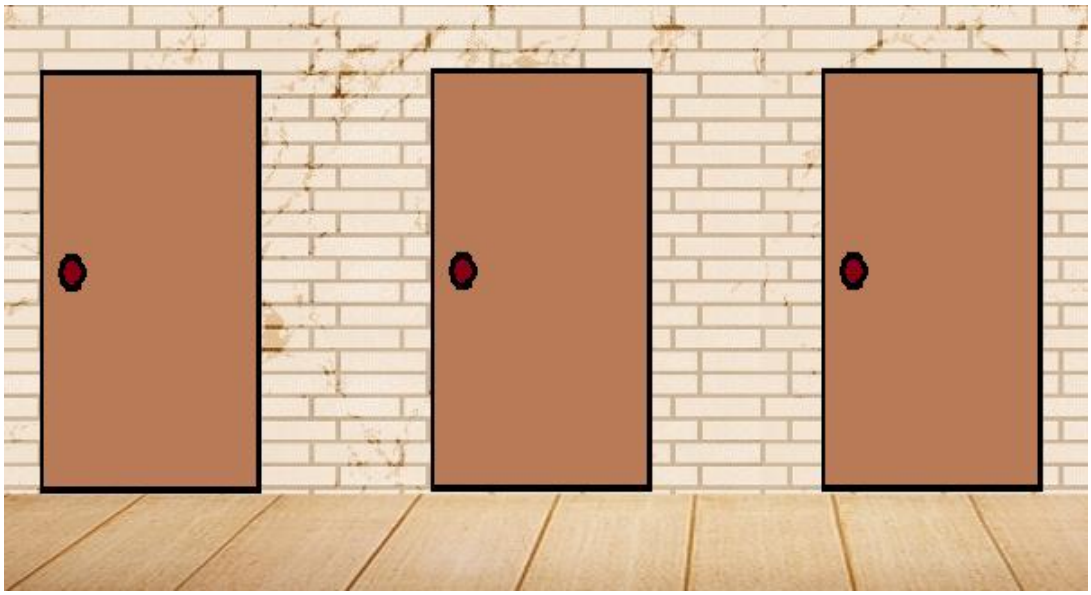


Figura 10.4.1: Tres puertas para ambientar el experimento de Monty Hall

En mi concepto es un problema de la física cuántica para el concursante, pero no para el presentador. En la primera situación la probabilidad de ganar es de $\frac{1}{3}$, en ese momento, para el concursante el premio se encuentra en cualquiera de las tres puertas, como en un estado de superposición. En el momento en que el presentador abre una puerta donde no está el premio, hace que en la mente del concursante, el premio colapse o se decida estar en cualquiera de las dos puertas restantes. Sin embargo, debido a que el sistema ha colapsado, si el concursante no cambia de opinión, sigue teniendo la misma probabilidad de $\frac{1}{3}$ de ganar. Si el concursante, revuelve el nuevo sistema de las dos puertas, y vuelve a elegir al azar, el nuevo sistema debe hacer colapsar el premio en cualquiera de las dos puertas, con una probabilidad de $\frac{1}{2}$ de ganar.

10.5. Teoría de la Información

El artículo de Claude E. Shannon "A Mathematical Theory of Communication" apareció en dos partes en las ediciones de julio y octubre de 1948 en el «Bell System Technical

Journal». En 1948, Shannon incorporó "nuevos factores" a la teoría de la transmisión de información, "en particular el efecto del ruido en el canal, y los ahorros posibles debido a la estructura estadística del mensaje original y debido a la naturaleza del destino final de la información" (Shannon, 1948, como se citó en Seising, 2012).

Es muy probable que el artículo de Shannon no se hubiera hecho famoso sin la ayuda del popular texto de Weaver, “The Mathematics of Communication” (Weaver, 1948 a), el cual reinterpretó el trabajo de Shannon para un público científico más amplio (Seising, 2012).

Los sistemas de comunicación de Shannon tienen las siguientes partes:

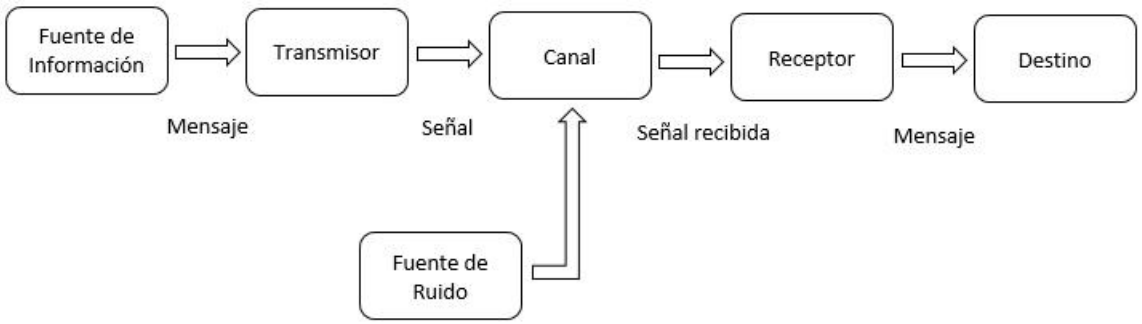


Figura 10.5.1: Diagrama de comunicación de Shannon en la versión de Weaver (1948 a), citado por Seising (2012).

- La fuente de información o emisor, produce una serie de mensajes que deben entregarse al destinatario, pasando primero por un proceso de conversión en algún tipo de señal, para ser transportada y luego decodificada para que la pueda interpretar el destinatario.
- El transmisor transforma el mensaje de alguna manera para que pueda producir señales que pueda transmitir a través del canal. El mensaje es transformado en una corriente eléctrica, en una onda electromagnética, etc. de esa manera es llevado al canal.
- El canal es el medio utilizado para la transmisión. Los canales pueden ser cables, haces de luz, espacio electromagnético, etc.
- El receptor debe realizar la operación contraria a la del transmisor y de esta manera reconstruye el mensaje original a partir de la señal transmitida. EL receptor convierte una señal eléctrica, óptica o electromagnética en un mensaje que pueda entender el destinatario.
- El destino es la persona o entidad que debe recibir el mensaje.

  (Seising, 2012).

Para Shannon “El problema fundamental de la comunicación es el de reproducir en un punto, ya sea exactamente o aproximadamente, un mensaje seleccionado en otro punto " (Shannon, 1948, como se citó en Seising, 2012). Esto quiere decir que el problema es transportar un mensaje y poderlo interpretar en su lugar de destino, reduciendo las posibles interferencias o

fuentes de ruido.

Para Weaver en la comunicación hay problemas en tres niveles:

1. Técnico, tienen que ver con la precisión de la transferencia de información del remitente al receptor, implica la exactitud con la que se pueden transmitir los símbolos.

2. Semántico, se refiere a la interpretación del significado por parte del receptor, en comparación con el significado pretendido por el remitente. Trata sobre la precisión con la que la señal trasnmitida transporta el significado deseado.

3. Influyente, se refiere a la efectiviadad del mensaje, en cuanto influya en la conducta del receptor.

   (Seising, 2012)

Para Weaver, la teoría de Shannon no tocaba los problemas contenidos en los niveles 2) y 3), para Shannon el concepto de transmisión de información era meramente técnico (Seising, 2012).

Weaver fue más allá de la teoría de Shannon, aunque la teoría de Weaver se aplica a problemas de nivel técnico, también es útil y sugerente en los niveles de semántica y eficacia. Weaver declaró que el diagrama formal de Shannon de un sistema de comunicación (ver figura 10.5.1) se le podría adicionar otra caja llamada “Receptor Semántico” interpuesta entre el receptor de ingeniería (que cambia señales a mensajes) y el destino, su función es que debe hacer coincidir las características semánticas estadísticas del mensaje con las capacidades semánticas estadísticas de la totalidad de destinatarios, o sea, interpretar el mensaje deacuerdo con las capacidades del destinatario. De manera similar, se puede imaginar otro cuadro en el diagrama que, insertado entre la fuente de información y el transmisor, estaría etiquetado como "Ruido semántico" (que no debe confundirse con "ruido de ingeniería"). Esto representaría distorsiones de significado introducidas por la fuente de información, tal como un orador, que no son intencionales pero que sin embargo afectan al destinatario, u oyente. Y el problema de la decodificación semántica debe tener en cuenta este ruido semántico. También es posible pensar en un tratamiento o ajuste del mensaje original, que haría que, la suma del significado del mensaje, más el ruido semántico, sea igual al significado del mensaje total deseado en el destino" (Shannon, C.E., Weaver, W., 1949, citado por Seising, 2012).

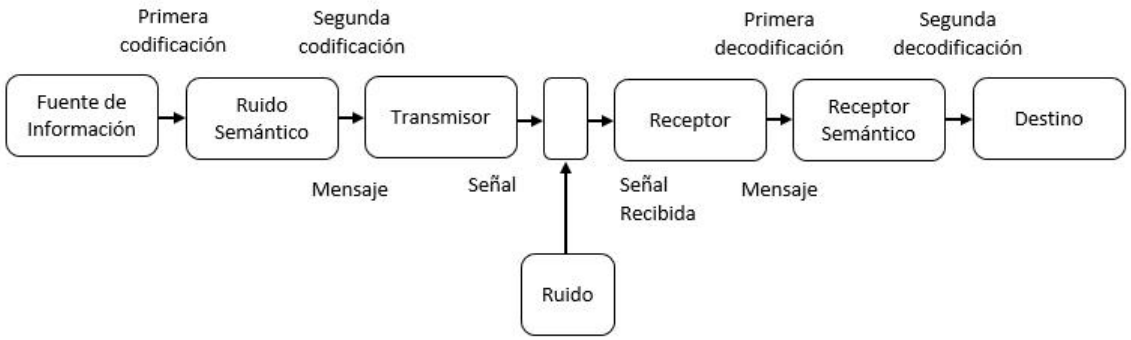


Figura 10.5.2: Diagrama de Shannon con la adición de Weaver de las dos casillas "Receptor semántico" y "Ruido semántico", en la versión de Seising (2012).

10.6. Entropía de Shannon

Es difícil medir la complejidad de cualquier sistema, especialmente porque pueden influir muchas variables. Seising (2012) citando a Weaver (1948 b) en su libro «Science and Complexity», establece que existen problemas de complejidad desorganizada, en los que pueden haber problemas que manejen dos mil millones de variables, y los define de la siguiente manera: “es un problema en el que el número de variables es muy grande, y en el que cada una de las muchas variables tiene un comportamiento individualmente errático, o quizás totalmente desconocido. Sin embargo, a pesar de este comportamiento desordenado, o desconocido, de todas las variables individuales, el sistema en su conjunto posee ciertas propiedades promedio ordenadas y analizables”. Estos problemas hacen uso de poderosas técnicas de la teoría de la probabilidad y de la mecánica estadística.

Por otro lado, Weaver (1948 b) también propone la clasificación de complejidad organizada, la cual involucra problemas que la ciencia poco ha explorado o conquistado, son problemas que no se pueden reducir a una fórmula sencilla ni pueden resolverse con los métodos de la estadística o de la teoría de la probabilidad.

Para Seising (2012), los problemas de complejidad organizada y complejidad desorganizada, fueron abordados por la ciencia en la segunda mitad del siglo XX, mediante la autoorganización, sinergia, teoría del caos, fractales, y mediante el enfoque de la teoría de sistemas y conjuntos difusos de Lotfi Zadeh.

Podríamos medir la complejidad en un sistema determinado, para un observador particular, en un contexto particular y un propósito particular. Por ejemplo, en la teoría de la información, reducimos cualquier evento a la probabilidad de que ocurra y damos una medida de la información que obtenemos de él.

Sean A y B, dos eventos distintos, llamamos P(A) a la probabilidad de que ocurra el evento A, por lo tanto P($\sim$A) es la probabilidad de que no ocurra el evento A. Definimos P(A$\wedge$B) como la probabilidad de que ocurra A y B simultáneamente, siendo $\wedge$ el símbolo de

la conjunción. Además, P(A∨B) es la probabilidad de que ocurra el evento A o B, siendo ∨ el símbolo de una disjunción inclusiva. Tenemos las siguientes propiedades:

- P(∼A) = 1 - P(A). Porque la probabilidad es un número real mayor o igual a cero y menor o igual a uno.
- P(A∨B) = P(A) + P(B) - P(A∧B).
- Si P(A∧B) = 0, entonces los eventos A y B son mutuamente excluyentes.

De la probabilidad condicional sabemos que P(A|B) = Probabilidad de que ocurra el evento A, sabiendo que ya ha ocurrido el evento B.

La probabilidad P(A∧B) = P(A|B).P(B), pero, sabemos que, si dos eventos son independientes, o sea que los resultados de uno son independientes de los del otro, tenemos que P(A|B) = P(A) o también P(B|A) = P(B), por lo tanto:

$$P(A \wedge B) = P(\ A|B\ ).P(B) = P(A).P(B)$$

$$P(A \wedge B) = P(A).P(B)$$

Ignoramos adrede cualquier característica particular del evento, sólo nos interesa observar si sucede o no, es decir, nos interesa sólo la probabilidad de ocurrencia del evento. Nuestro evento será observar un símbolo cuya probabilidad de ocurrir es «$0 \leq p_i \leq 1$», por lo tanto definimos una «función de información» en términos de la probabilidad «p_i».

Formamos el espacio muestral de nuestros eventos, pero no vamos a describir los eventos sino la probabilidad de cada uno de ellos. El conjunto con las probabilidades del espacio muestral sería:

$\{p_1, p_2, p_3, p_4,..., p_m\}$, sabemos que $0 \leq p_i \leq 1$, y, $\sum_{i=1}^{m} p_i = 1$, por la definición de la probabilidad.

Queremos encontrar cierta probabilidad dentro del conjunto del espacio muestral, para ello vamos a hacer particiones de dos en dos, primero dividir el conjunto del espacio muestral en dos conjuntos, luego cada conjunto resultante de nuevo se divide en dos conjuntos, y así sucesivamente hasta que sólo haya un elemento p_i en cada conjunto, $\{p_1\}$, $\{p_2\}$, $\{p_3\}$, $\{p_4\}$,..., $\{p_m\}$. Para llegar a esto hemos hecho n particiones o pasos, lo importante por ahora es encontrar ese valor de n, el cual nos permite analizar cada una de las probabilidades del espacio muestral.

$\{p_1, p_2\}$, para m = 2
$\{p_1\}$, $\{p_2\}$, se particiona en un sólo paso.
- -
$\{p_1, p_2, p_3\}$, para m = 3
$\{p_1, p_2\}$, $\{p_3\}$
$\{p_1\}$, $\{p_2\}$, $\{p_3\}$, se particiona en dos pasos.

- -
$\{p_1, p_2, p_3, p_4\}$, para m = 4
$\{p_1, p_2\}, \{p_3, p_4\}$
$\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}$, se particiona en dos pasos.
- -
$\{p_1, p_2, p_3, p_4, p_5\}$, para m = 5
$\{p_1, p_2, p_3\}, \{p_4, p_5\}$
$\{p_1, p_2\}, \{p_3\}, \{p_4\}, \{p_5\}$
$\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}, \{p_5\}$, se particiona en tres pasos.
- -
$\{p_1, p_2, p_3, p_4, p_5, p_6\}$, para m = 6
$\{p_1, p_2, p_3\}, \{p_4, p_5, p_6\}$
$\{p_1, p_2\}, \{p_3\}, \{p_4, p_5\}, \{p_6\}$
$\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}, \{p_5\}, \{p_6\}$, se particiona en tres pasos.
- -
$\{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$, para m = 7
$\{p_1, p_2, p_3, p_4\}, \{p_5, p_6, p_7\}$
$\{p_1, p_2\}, \{p_3, p_4\}, \{p_5, p_6\}, \{p_7\}$
$\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}, \{p_5\}, \{p_6\}, \{p_7\}$, se particiona en tres pasos.
- -
$\{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$, para m = 8
$\{p_1, p_2, p_3, p_4\}, \{p_5, p_6, p_7, p_8\}$
$\{p_1, p_2\}, \{p_3, p_4\}, \{p_5, p_6\}, \{p_7, p_8\}$
$\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}, \{p_5\}, \{p_6\}, \{p_7\}, \{p_8\}$, se particiona en tres pasos.
- -
$\{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$, para m = 9
$\{p_1, p_2, p_3, p_4, p_5\}, \{p_6, p_7, p_8, p_9\}$
$\{p_1, p_2, p_3\}, \{p_4, p_5\}, \{p_6, p_7\}, \{p_8, p_9\}$
$\{p_1, p_2\}, \{p_3\}, \{p_4\}, \{p_5\}, \{p_6\}, \{p_7\}, \{p_8\}, \{p_9\}$
$\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}, \{p_5\}, \{p_6\}, \{p_7\}, \{p_8\}, \{p_9\}$, se particiona en cuatro pasos.

Observamos que:

Para m = 2, n = 1 paso.
Para m = 3, n = 2 pasos.
Para m = 4, n = 2 pasos.
Para m = 5, n = 3 pasos.
Para m = 6, n = 3 pasos.
Para m = 7, n = 3 pasos.
Para m = 8, n = 3 pasos.
Para m = 9, n = 4 pasos.
Para m = 10, n = 4 pasos.
Para m = 11, n = 4 pasos.
Para m = 12, n = 4 pasos.
Para m = 13, n = 4 pasos.

Para m = 14, n = 4 pasos.
Para m = 15, n = 4 pasos.
Para m = 16, n = 4 pasos.

Por el método de inducción matemática podemos demostrar que $2^{n-1} < m \leq 2^n$, para cualquier n número de pasos, por ello podemos escoger a $m = 2^n$ como representante de la clase $[m]$ de números que cumplen la desigualdad $2^{n-1} < m \leq 2^n$. Obviamente que m y n son números enteros positivos.

Hemos obtenido la siguiente ecuación: $m = 2^n$. Luego tenemos:

$$\text{Si } m = 2^n, \text{ entonces, } \log_2 m = n.$$

Esa función del logaritmo $n = \log_2 m$, es la función que nos permite hallar información acerca de la probabilidad de cada uno de los elementos del espacio muestral. Si hubiéramos hecho particiones de base tres, o cuatro, tendríamos que utilizar esas mismas bases para el logaritmo.

Luego tenemos que la función de la información asociada a cada valor de probabilidad sería: $I_i = \log_2 p_i$, pero, como sabemos que las probabilidades son valores reales entre el número cero y el número uno, el logaritmo sería una cantidad negativa, por ello tendríamos que multiplicar por el signo menos, para que el valor de la información nos de positivo, de esa manera la función de información quedaría así:

$$I_i = -\log_2 p_i, \text{ y esto también sería igual a } I_i = \log_2(1/p_i).$$

La probabilidad de obtener cada información $I_{i,}$sería $H_i = p_i.I_i$. EL valor mínimo de la probabilidad de obtener toda la información, o grado mínimo de incertidumbre de una fuente de información, es:

$$H = \sum_{i=1}^{m} p_i.I_i = -\sum_{i=1}^{m} p_i.(\log_2 p_i) = \sum_{i=1}^{m} p_i.\log_2(1/p_i)$$

La gráfica que represente la mínima incertidumbre depende de la cantidad de variables que se van a utilizar, supongamos un evento binario, una caja donde sólo cabe el número uno o el número cero, a esta casilla se le llama un bits de información. A la probabilidad de obtener el número uno le vamos a llamar p_1, y, a la probabilidad de obtener el número cero le vamos a llamar p_0. Por lo tanto, a la mínima incertidumbre de obtener el número «uno» le vamos a llamar $H_1 = -p_1.log_2 p_1$, y, a la mínima incertidumbre de obtener el número «cero» le vamos a llamar $H_0 = -p_0.log_2 p_0$. EL valor mínimo de incertidumbre de obtener toda la información es $H = -\sum_{i=1}^{m} p_i.(\log_2 p_i)$, y para nuestro caso sería:

$$H = H_0 + H_1 = -p_0.log_2 p_0 - p_1.log_2 p_1$$

Sabemos que las probabilidades tienen valores en el intervalo cerrado $[0,1]$, y como sólo son dos eventos en el espacio muestral, «cero» o «uno», se cumple que $p_0 + p_1 = 1$, y dado que conviene utilizar una sola variable para graficar con respecto al eje «X», es necesario representar cualquiera de las variables en términos de la segunda, podría ser $p_0 = 1 - p_1$. Por lo tanto, la ecuación $H = -p_0.log_2 p_0 - p_1.log_2 p_1$, quedaría así:

$$H = -(1-p_1).log_2(1-p_1) - p_1.log_2 p_1$$

Para efectos de la gráfica en el plano X-Y, la anterior ecuación quedaría así:

$$y = -(1-x).log_2(1-x) - x.log_2 x$$

Ahora, introducimos la anterior fórmula en cualquier programa de gráficas y se obtiene la siguiente:

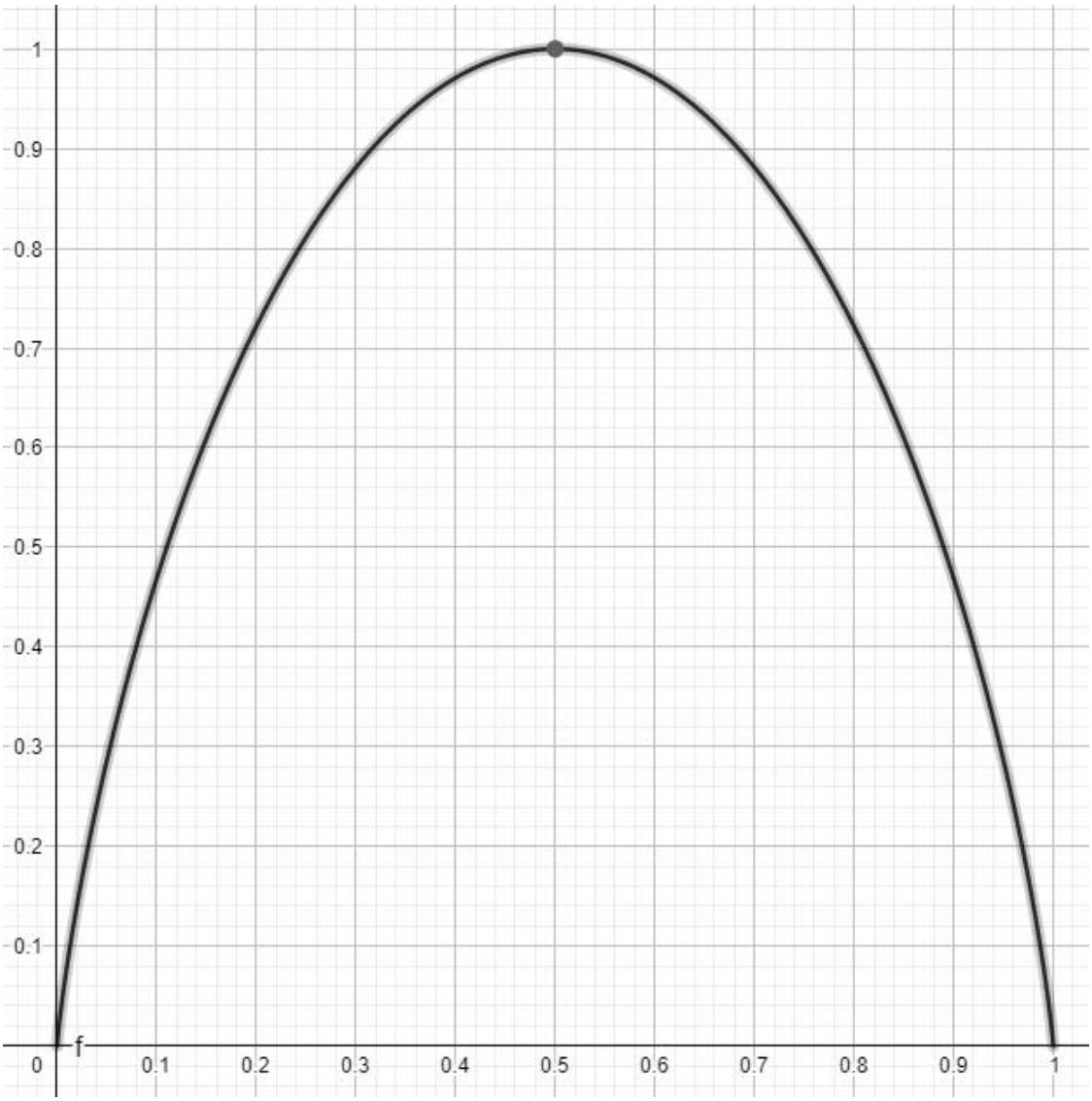


Figura 10.6.1: Incertidumbre mínima en un bits de información

Para el anterior gráfico se utilizó el programa Geogebra online, en la página web:

https://www.geogebra.org/graphing?lang=es

Al valor mínimo de incertidumbre en una fuente de información se le llama entropía máxima de información. También significa que la mínima cantidad de bits que necesita un

canal para que no haya pérdida de informacón es $H = \sum_{i=1}^{m} p_i.\log_2(1/p_i)$. En complejidad de algoritmos, H determina la mínima cantidad de pasos o instrucciones en la que podemos reducir el orden de complejidad de un algoritmo. Es decir, que H siempre va a ser un número real positivo, y nos indica la máxima compresión que podemos hacer de la información, o también que H determina la mínima cantidad de instrucciones para optimizar un algoritmo. Esta teoría es muy útil en la encriptación o codificación de mensajes y en la optimización de algoritmos. Al valor numérico de H, se conoce como entropía, y es un concepto muy utilizado en la informática teórica.

Ejemplo 13. Encontrar el género de una persona que va a un bar a jugar billar. Espacio muestral = {Másculino, Femenino}. Se sabe que la probabilidad de que el género de esa persona sea Masculino es de P(M) = 9/10, y la probabilidad de que el género de esa persona sea Femenino es de P(F) = 1/10. (Este ejemplo es imaginario y no obedece a ninguna estadística real).

La información asociada al género masculino es:

$$I_M = -\log_2 p_M = -\log_2(9/10) = +0{,}1520$$

La información asociada al género femenino es:

$$I_F = -\log_2 p_F = -\log_2(1/10) = +3{,}3219$$

Lo que significa que un evento arroja más cantidad de información entre menos probabilidad tenga de suceder, y un evento entre más probabilidad tenga de suceder menos cantidad de información arroja.

La entropía del sistema sería:

$$H = -\sum_{i=1}^{2} p_i.(\log_2 p_i) = -[(9/10).\log_2(9/10) + (1/10).\log_2(1/10)] = 0.4690$$

Este valor de la entropía, en este ejemplo, significa que la probabilidad mínima de la información obtenida es de 0.4690. O también que, la incertidumbre mínima de conocer el género de esa persona es de 0.4690.

10.6.1. Curva de Entropía Máxima de Shannon en un Evento Binario

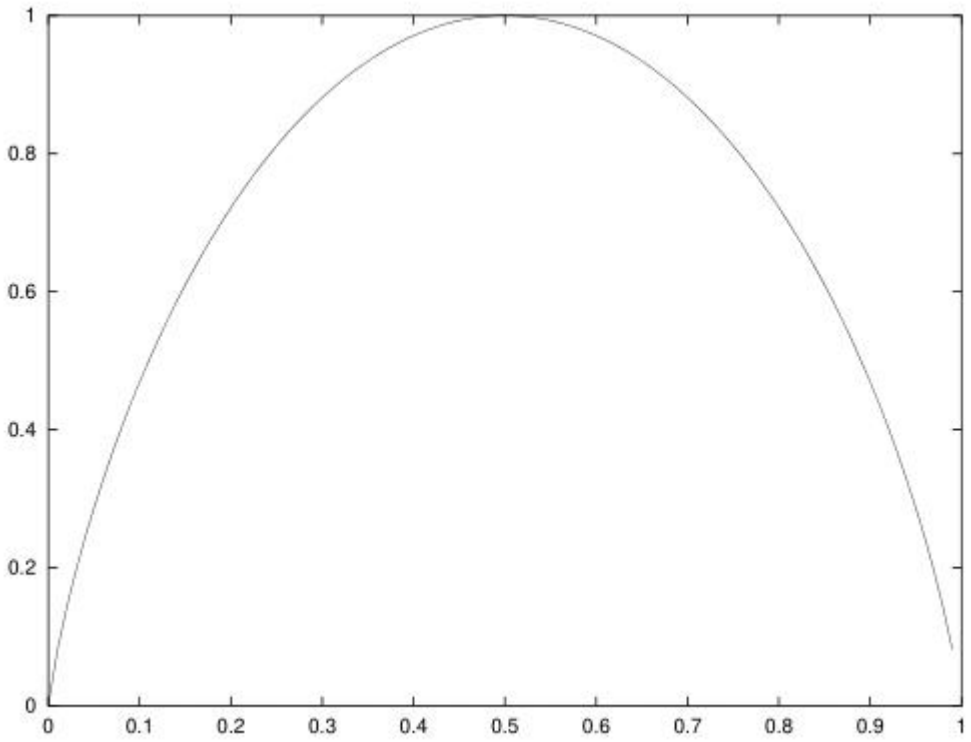


Figura 10.6.2: Curva de entropía máxima para un bits de información (López, 2002).

En una distribución uniforme de probabilidad se presenta la entropía máxima, ya que hay mayor grado de incertidumbre, todos los datos tendrán la misma probabilidad de salir. En la gráfica se observa que para una variable que ocupa un bits de información, ya sea el 0 o el 1, cuando la probabilidad de la variable es 0.5, entonces la entropía es máxima, en este caso es igual a 1. La entropía indica el grado de incertidumbre de un fenómeno aleatorio (López, 2002).

Se sabe que la variable sólo puede tomar los valores 0 o 1, si un valor tiene la probabilidad de 0 %, significa que el otro tiene la probabilidad del 100 %, por lo que no hay incertidumbre, y la entropía es mínima. Lo mismo significa cuando la probabilidad de un dato es del 100 % y la probabilidad del otro es del 0 %. En ambos extremos la incertidumbre es mínima. El hecho de tener dos posibles valores 0 o 1 para la variable, explica la simetría de la gráfica anterior.

Una vez que se sabe el resultado de la variable, desaparece el grado de incertidumbre y se genera la información, por este motivo, puede considerarse la entropía como una medida de la cantidad de información que proporciona un experimento o fenómeno. Pero esta medida de la cantidad de información, es muy útil si se halla la cantidad mínima de información que es la que nos puede proporcionar la entropía (López, 2002).

Capítulo 11

Complejidad Computacional

La Complejidad Computacional se relaciona con el tiempo de duración de ejecución de algoritmos computacionales. En palabras de Gell-Mann (1995), la complejidad computacional en sí está relacionada con el menor tiempo (o número de pasos) necesarios para realizar un determinado cálculo. En este capítulo se busca responder a la pregunda de ¿cómo se determina el tiempo de ejecución de un algoritmo en una Máquina de Turing? Se estudian algoritmos en las Máquinas de Turing, porque son algoritmos computacionales, y a la fecha de hoy no existe un modelo matemático más perfecto del procesamiento de una computadora que una Máquina de Turing. Aquí se entiende por máquina no un objeto o una estructura física, se entiende por máquina como un modelo teórico de una estructura matemática de procesamiento de información.

El concepto de algoritmo se ha utilizado durante toda la historia de la matemática, pero la noción de algortimo no se definió con precisión hasta el siglo XX. En el año 1900, el famoso matemático David Hilbert, pronunció un famoso discurso en el Congreso Internacional de Matemáticos en París, en este congreso identificó veintitrés problemas matemáticos como un desafío para el siglo venidero, el décimo problema de su lista se refería a los algoritmos: «Hallar un proceso según el cual pueda ser determinado por un número finito de acciones, si un polinomio con coeficientes de números enteros tiene soluciones enteras», en este problema Hilbert estaba pidiendo que se hallara un algoritmo, y ya estaba suponiendo que este algoritmo existía, este algoritmo debía determinar si el polinomio con coeficientes enteros tenía solución o no dentro del conjunto de los números enteros, esto es lo que se conoce como un problema de decisión, un problema de decidibilidad. Luego, en 1970, el matemático Yuri Matiyasevich, basado en los trabajos de Julia Robinson, Martin Davis y Hilary Putnam, demostró que no existía un algoritmo que puediera resolver este problema, por ello este problema fue considerado indecidible, es decir que para este caso, no se puede determinar un algoritmo que decida si el problema tiene o no tiene solución. La dificultad radicaba en que no existía una definición clara de algoritmo, se podían crear algoritmos para ciertas tareas, pero era difícil demostrar que no existía un algoritmo para una tarea en particular. El progreso en el décimo problema de Hilbert, tuvo que esperar a las investigaciones de Alonzo Church y Alan Turing en 1936, sobre una definición precisa de algoritmo. La tesis de Church-Turing proporciona la definición de algoritmo necesaria para resolver el décimo problema de Hilbert (Sipser, 2006).

El décimo problema de Hilbert también se puede expresar así:

Sea D = { p / p es un polinomio con una raíz entera}

El décimo problema de Hilbert pregunta en esencia, si el conjunto D es decidible, la respuesta como ya sabemos, es negativa. Es decir no se puede hallar un algoritmo que diga si el conjunto D se puede solucionar o no. Sí se puede demostrar que el conjunto D es reconocible por una Máquina de Turing, pero no se puede resolver mediante dicha máquina de cómputo (Sipser, 2006).

La tesis de Church-Turing establece la equivalencia entre algoritmos y Máquinas de Turing, en conclusión, un algoritmo es un conjunto de instrucciones computables, es decir que se puede resolver mediante una Máquina de Turing.

11.1. Problemas de Clase P

Los algoritmos de complejidad polinómica son tratables porque se pueden resolver de manera práctica, los problemas que tienen algoritmos con esta complejidad forman la clase P.

La clase P consiste de todos aquellos problemas de decisión que pueden resolverse en tiempo polinomial por una computadora determinística (Arjona, 2007).

11.2. Problemas de Clase NP

Los problemas de clase NP, se refieren a problemas no deterministas con solución polinómica. Ejemplo: Si tenemos un conjunto de enteros $\{-2,-3,15,14,7,-10\}$, hallar un subconjunto tal que la suma de sus elementos de igual a cero. Para este ejercicio no podemos aplicar un algoritmo definido (determinístico), nos toca aplicar algoritmos no determinísticos, que en este caso sería por tanteo hasta encontrar la respuesta que satisface la condición (Mañas, 1997). Se trata de hallar todas las combinaciones de subconjuntos de al menos dos elementos,

$$\binom{6}{2}+\binom{6}{3}+\binom{6}{4}+\binom{6}{5}+\binom{6}{6}=\sum_{i=2}^{6}\binom{6}{i}=57$$

Lo cual quiere decir que en un conjunto de 6 elementos hay 57 subjconjuntos de elementos diferentes que se pueden sumar entre sí. Y generalizando podemos decir que si un conjunto tiene n elementos, habrían

$$\binom{n}{2}+\binom{n}{3}+\binom{n}{4}+\ldots+\binom{n}{n}=\sum_{i=2}^{n}\binom{n}{i}=2^n-n-1$$

Dado el caso de que haciendo todas las combinaciones posibles, no se encuentra algún subconjunto cuya suma de sus elementos sea cero, podríamos concluir que el ejercicio no tiene solución. Pero en este caso sí se ha encontrado al menos una solución: $\{-2,-3,-10,15\}$.

Los problemas de clase NP, son problemas que no se resuelven en tiempo polinomial con algoritmos determinístas (Maldonado, 2010), NP también significa problema que se soluciona con un algoritmo no determinísta en tiempo polinomial, eso quiere decir que para que el problema NP se pueda resolver en un tiempo razonable o tratable, se debe emplear un método no determinísta, de lo contrario el problema sería intratable o de solución no polinomial. Sin embargo, una vez hallada la solución, sí se puede verificar con un algoritmo determinista o polinomial.

La clase NP se define como el conjunto de todos los problemas de decisión que pueden resolverse por una computadora no determinística en tiempo polinomial (Arjona, 2007).

Los problemas de tipo P se consideran tratables, es decir que se pueden solucionar en un tiempo aceptable, razonable o de duración polinomial. Sin embargo, los problemas de tipo NP que no están en P, se consideran intratables, o sea que aunque tengan solución, dicha solución podría tardar más años que la edad del universo (Penrose, 1996).

11.3. Problemas de Clase NP Completos

Los problemas de clase NP-Completos, son problemas que sólo se resuelven en tiempos no polinómicos, son problemas intratables, estos problemas son una aplicación del teorema de Cook, este teorema establece que hay una clase de problemas NP que se pueden transformar en problemas equivalentes, estas transformaciones se dan en tiempos polinómicos, y por el hecho de ser problemas equivalentes, se observa que si encontramos un algoritmo que resuelva un sólo problema NP-Completo, este algoritmo también resolvería todos los problemas de este tipo (Arjona, 2007).

Una Máquina de Turing, frecuente en el teorema de Cook, no puede estar en dos estados al mismo tiempo, el término «no determinista» quiere decir que la máquina no recorre una sola secuencia o ruta, sino varios caminos, no todos al tiempo sino uno por uno. No se puede realizar más de una operación en una Máquina de Turing, es una máquina secuencial que hace lo que tiene que hacer (Arjona, 2007). Si una máquina es no determinista, significa que dado un estado y una misma entrada, tomará varios caminos o transiciones a otros estados, no de manera simultánea sino secuencial; es como si una sola persona recorriera todo un laberinto hasta encontrar la salida.

Arjona (2007), concluye que cualquier par de problemas NP-Completos son mutuamente transformables entre sí, en tiempo polinomial. Dice que resolviendo uno solo, se podrán resolver todos, y que sin embargo, nadie ha encontrado un algoritmo para ningún problema NP-Completo en tiempo polinomial.

Los problemas NP contienen a los problemas de tipo NP- Completos, son problemas que se resuelven en Máquinas de Turing no deterministas en tiempos polinomiales; la diferencia es que los problemas de tipo NP-Completos son equivalentes entre sí, quiere decir que al poder resolver uno de ellos se podrían resolver todos, bajo el mismo algoritmo.

Un problema del tipo NP completo, significa que cualquier otro problema NP puede ser reducido a éste en un tiempo polinómico; o sea que si se pudiera encontrar un algoritmo que resolviera el problema en tiempo polinómico, es decir, que probara que el problema está en P, se concluiría que todos los problemas NP están realmente en P (Penrose, 1996).

11.4. Teoría Matemática de la Complejidad

La teoría matemática de la complejidad trata de las relaciones entre P y N-P. Estos problemas constituye la manera como se desarrolla el modelamiento y la simulación de sistemas complejos y, en general, el estudio y la investigación en ciencias de la complejidad (Maldonado, 2010).

Los problemas de la complejidad computacional tratan de estudiar los tiempos de computación para resolver los problemas P y NP, estudian las relaciones posibles entre este tipo de problemas: $¿P = NP?$, $¿P \neq NP?$, $¿NP \geq P?$, $¿P \in NP?$, ninguna de estas relaciones han podido ser resueltas hasta la fecha, y además hacen parte de los siete problemas del milenio (Maldonado, 2010).

11.5. Complejidad de un Algoritmo

Hallar la complejidad de un algortimo es hallar cuál es el tiempo de ejecución del algoritmo. Para ello hay dos maneras; una es analítica, en la cual se estudia el código que representa el algoritmo dentro de un programa (software) de computación, la otra manera es de tipo experimental, se miden los tiempos de ejecución y se analizan de manera estadística (Mañas, 1997).

Analizar un código de programación de manera teórica no es tan sencillo y práctico, ya que eso depende de muchos factores; las librerías que se utilicen, los llamados de funciones o de variables, los tipos de variables, etc. Y es completamente imposible cuando no conocemos el código fuente en que está escrito el algoritmo (Mañas, 1997).

En el método experimental, para medir el tiempo de ejecución del código de programación del algoritmo, según Mañas, 1997, se tiene en cuenta lo siguiente:

1. Se ejecuta el código para varios valores de N.
2. Tabulamos los tiempos de ejecución del programa, para los diferentes valores de N.
3. Graficamos los tiempos de ejecución, analizando a qué tipo de gráfica se ajusta mejor

La gráfica que más se ajuste según el coeficiente de correlación de la regresión, determina el tipo de complejidad del código de programación del algoritmo.

Debemos tener en cuenta la siguiente notación:

T(N) – tiempo de ejecución en función del tamaño N del problema. La variable N se refiere al número máximo que puede tomar la variable dentro del problema, especialmente si la variable es de tipo recursiva.

«Para cada problema determinaremos una medida N de su tamaño (por número de datos) e intentaremos hallar respuestas en función de dicho N. El concepto exacto que mide N depende de la naturaleza del problema. Así, para un vector se suele utilizar como N su longitud; para una matriz, el número de elementos que la componen; para un grafo, puede ser el número de nodos (a veces es más importante considerar el número de arcos, dependiendo del tipo de problema a resolver); en un fichero se suele usar el número de registros, etc. Es imposible dar una regla general, pues cada problema tiene su propia lógica de coste» (Mañas, 1997).

El tiempo de ejecución de un programa en función de N, representado mediante T(N), se había dicho que se podía hallar de dos maneras, una experimental cronometrando la ejecución del programa para valores grandes de N; y la otra de forma analítica sobre el código de programación, contando las instrucciones y multiplicando por el tiempo requerido por cada instrucción (Mañas, 1997).

La notación O(N), se refiere al grado u orden de complejidad para un determinado código de programación, se recuerda que el código de programación es diferente del algoritmo del problema, ya que el algortimo es el procedimiento que va a desarrollar el problema y el código es el lenguaje de programación en que se va a escribir o a representar el algoritmo. Los lenguajes de programación pueden ser C, C++, Java, Python, Visual Basic, etc...

Podemos entender O(N) como la función que más se ajusta de manera asintótica al tiempo T(N), para valores muy grandes de N.

Se observa que las funciones más comunes, para estimar el tiempo que se demora en ejecutar el código de un algoritmo, de cierto tamaño N, representado mediante la función T(N), son:

Tiempo de Ejecución $T(N)$	Orden de Complejidad	Clase de Orden
$T(N) = 1$, Valor constante	$O(N) = 1$, Valor constante	Orden Constante
$T(N) = log(N)$	$O(N) = log(N)$	Orden Logarítmico
$T(N) = N$	$O(N) = N$	Orden Lineal
$T(N) = Nxlog(N)$	$O(N) = Nxlog(N)$	
$T(N) = N^2$	$O(N) = N^2$	Orden Cuadrático
$T(N) = N^a, a = constante$	$O(N) = N^a, a = constante$	Orden Polinomial $(a > 2)$
$T(N) = a^N, a = constante$	$O(N) = a^N, a = constante$	Orden Exponencial $(a > 1)$
$T(N) = N!$	$O(N) = N!$	Orden Factorial

Cuadro 11.1: Funciones de referencia (Mañas, 1997)

T(N) = 1 u O(N) = 1, se refiere a que la instrucción demora una mínima unidad de medida de tiempo computacional, de valor constante, o sea una millonésima parte de un segundo $1\mu s$.

Se enfatiza que al hacer medidas de tipo experimental, se debe realizar la operación un número elevado de veces, con valores de N que arrojen datos considerables (grandes), los valores de N deben ser aleatorios para quedarnos con un valor promedio o tendencia (Mañas, 1997).

Se puede pensar en la Máquina de Turing como una computadora moderna simplificada, con la cinta de la máquina correspondiente a la memoria de una computadora, y la función de transición y el registro correspondiente a la unidad central de procesamiento (CPU) de la computadora. Sin embargo, es mejor pensar en las Máquinas de Turing como simplemente una forma formal de describir algoritmos. A pesar de que los algoritmos a menudo se describen mejor con texto en inglés, a veces es útil expresarlos mediante tal formalismo para discutir matemáticamente sobre ellos. (Del mismo modo, uno necesita expresar un algoritmo en un lenguaje de programación para ejecutarlo en una computadora), (Arora, S., 2009).

De lo anterior deducimos que una Máquina de Turing tiene dos interpretaciones, una como estructura matemática que representa algoritmos; o también como la representación teórica de un computador, en el sentido de recibir entradas, ejecutar un proceso, escribir, borrar o almacenar información y mover el cabezal de lectura hacia cierta orientación para conttinuar con su operación. Sin embargo, en este libro nos interesa es la complejidad de los algoritmos dentro de un código o lenguaje de programación y la complejidad de un algoritmo dentro de una Máquina de Turing, siendo analizadas estas máquinas como estructuras matemáticas. Para esta investigación son muy importantes las Máquinas de Turing, ya que la tesis de Church - Turing establece que «si una Máquina de Turing no puede resolver un problema, entonces ningún computador puede hacerlo, ya que no existe algoritmo para obtener una solución» (Cortez, 2004).

Otro aspecto que nos interesa en esta investigación, además de la teoría de la complejidad, es la teoría de la computabilidad. Sabemos que la teoría de la complejidad estudia la eficiencia de los algoritmos en función de los recursos que utiliza en un sistema computacional, estos recursos pueden ser espacio o memoria, y tiempo, (Cortez, 2004). Mientras que la teoría

de la computabilidad estudia la abstracción de los algoritmos, la construcción de algoritmos sin emplear un lenguaje de programación específico, para este caso se utilizan las Máquinas de Turing como un modelo de máquina isomorfa a cualquier otro sistema computacional (Cortez, 2004). Se hace énfasis en que los sistemas computacionales no siempre son físicos, también pueden ser teóricos.

Capítulo 12

Computación Cuántica

Peter Shor en 1994, dio impulso a la Computación Cuántica, al proponer un algoritmo denominado algoritmo de Shor, el cual descompone un número en sus factores primos, ejecutándose en computadores cuánticos que aún no se han construido (Vélez, 2001). Desde la práctica un computador clásico necesita apróximadamente 4,25x10^{25}años para factorizar un número de 1000 dígitos, sin embargo, en teoría un computador cuántico realizaría dicha tarea en sólo 307 días (Vélez, 2001). Por ello la seguridad informática tradicional está en riesgo, actualmente los computadores cuánticos están en construcción, los modelos teóricos no se han puesto totalmente en la práctica, la capacidad de los computadores cuánticos en la actualidad no tienen la capacidad de hacer tareas grandes.

En octubre de 2019, Google anunció que había logrado la supremacía cuántica, es decir, un ordenador cuántico basado en su procesador Sycamore de 54 qubits, capaz de realizar un cálculo en 200 segundos que habría llevado 10.000 años a un superordenador convencional, aunque IBM pone en duda este hito (García, 2020). IBM ya construyó un computador cuántico capáz de aplicar el algoritmo de Shor, este algoritmo cuántico es capáz de factorizar números gigantes en sus factores primos, en ese momento el ordenador de IBM sólo pudo factorizar hasta el número 15 en el año 2012. Actualmente la empresa D-Wave (Quantum Computing Company) publicita en su página web: https://www.dwavesys.com/d-wave-two-system , su computadora cuántica 2000Q, en donde afirma que es la computadora cuántica más avanzada del mundo. Cuando las computadoras cuánticas estén desarrolladas completamente, no habrá seguridad informática en el mundo, por ejemplo sus algoritmos cuánticos podrán romper cualquier clave RSA basada en números primos, por ello será muy necesario que la seguridad informática se construya con base a la criptografía cuántica. El autor García, 2020, establece que «la criptografía cuántica permite distribuir claves privadas a través de un canal cuántico, con la ventaja de que cualquier intento de medir la clave será detectado, ya que es imposible observar un qubit sin dejar rastro (debido a la propiedad de entrelazamiento).» Además dicho autor, nos recuerda que las claves cuánticas ya se están aplicando, dice que en el 2007 hubo una videoconferencia cifrada con claves cuánticas entre China y Austria, separados una distancia de 7.600 Km.

Para García (2020), la ventaja que tienen los computadores cuánticos es que resuelven problemas con complejidad lineal, aquellos problemas que en un computador clásico es de

complejidad exponencial. En los términos de esta tesis de grado, lo que establece García, es que un computador cuántico convierte los problemas no polinomiales en problemas polinomiales, hablando de los problemas no polinomiales como aquellos que un computador clásico resuelve en tiempos no polinomiales, o tiempos demasiado grandes. Por ejemplo un problema que el mejor computador clásico del mundo resuelve en 10 mil años, un computador cuántico lo podría resolver en sólo 200 segundos. Factorizar claves de 2048 bits en promedio tardaría $4x10^{16}$años, con un ordenador clásico; pero con un ordenador cuántico la factorización de claves tardaría sólo unos segundos (García, 2020).

12.1. Mecánica Cuántica

Todo sistema cuántico se desarrolla dentro de un espacio de Hilbert, el cual puede ser de dimensión finita o infinita, el sistema se describe mediante una función de onda o estado. La ecuación de onda o de estado, se le llama también ecuación de Schrödinger:

$$i\hbar\frac{\partial}{\partial t} \mid \Psi > = H \mid \Psi >$$

Donde Hes el Hamiltoniano del sistema cuántico, o también la energía necesaria para un cambio de estado. La función Ψdescribe la dinámica o el comportamiento del sistema cuántico, nos dice cómo cambia los elementos del sistema con el tiempo.

12.2. Espacio Vectorial de Hilbert de Dimensión Finita en un Campo Escalar Complejo.

En la notación de Dirac, se escribe como $\mid u >$, para representar un vector columna, y $< u \mid$ para representar un vector fila.

A un elemento $\mid u >$ se le llama vector o función de onda de un espacio de Hilbert $\mathscr{H}$, si cumple las siguientes condiciones:

Para todo $a, b \in \mathbb{C}$, campo escalar complejo.

1. $a. \mid u > \in \mathscr{H}$, y $(\mid u > + \mid v >) \in \mathscr{H}$, propiedad clausurativa.
2. $\mid u > +(\mid v > + \mid w >) = (\mid u > + \mid v >) + \mid w >$, y, $a.(b. \mid u >) = (a.b). \mid u >$, propiedad asociativa.
3. $\exists \mid 0 > / \mid u > + \mid 0 > = \mid u >$, existencia del elemento neutro.
4. $\exists - \mid u > / - \mid u > + \mid u > = \mid 0 >$, existencia del elemento opuesto aditivo.
5. $\mid u > + \mid v > = \mid v > + \mid u >$, propiedad conmutativa.
6. $a.(\mid u > + \mid v >) = a. \mid u > + a. \mid v >$, y, $(a+b). \mid u > = a. \mid u > + b. \mid u >$, propiedad distributiva. Hasta aquí el espacio de Hilbert es un espacio vectorial.

7. Existencia de un producto escalar: $< u \mid v > = x \in \mathbb{C}$
8. $< u \mid v >^* = < v \mid u >$, quiere decir que el conjugado de un producto es igual a conmutar el orden de los factores. Recordemos el concepto del conjugado de un número complejo: $(a+bi)^* = (a-bi)$.
9. $< u \mid u > \geq 0 \in \mathbb{R}$
10. $Si < u \mid u > = 0 \in \mathbb{R} \Longrightarrow \mid u > = \mid 0 > \in \mathscr{H}$
11. $Si \mid u > = a. \mid v > \Longrightarrow < u \mid u > = a^*.a < v \mid v >$
12. Existencia de la norma: $\| \mid u > \| = \sqrt{\mid < u \mid u > \mid}$
13. Desigualdad triangular: $\| \mid u > + \mid v > \| \leq \| \mid u > \| + \| \mid v > \|$
14. El espacio debe ser completo.
15. Cada elemento del espacio debe tener un dual, o sea $\mid u >^* = < u \mid \in \mathscr{H}^*$, para el caso de vectores de dimensión finita, para todo vector columna existe su vector fila con sus elementos conjugados.
16. El producto de vectores $< u \mid v >$ está definido de la siguiente manera: $< u \mid . \mid v > = < u \mid v >$. Lo cual quiere decir que dos vectores se multiplican si el primero es un vector fila y el segundo es un vector columna. Esto porque la multiplicación de vectores en este caso es similar a multiplicar matrices. Sea A_{1xn}una matriz de 1 fila por n columnas, esta matriz sólo se puede multiplicar por otra matriz B_{nx1} de n filas por 1 columna. De esa manera el resultado de multiplicar estos dos vectores o matrices, siempre dará como resultado un vector o matriz C_{1x1}, siendo este resultado un número del campo escalar, para este caso es un número del campo escalar complejo. O sea $A_{1xn}.B_{nx1} = C_{1x1}$. Por ello se concluye que el espacio dual se utiliza para poder definir el producto interior entre dos vectores del espacio de Hilbert $\mathscr{H}$.
17. $< u \mid .(\mid v > + \mid w >) = < u \mid v > + < u \mid w >$
18. $[a. \mid u >]^* = a^*. < u \mid$
19. $[a. \mid u > + b. \mid v >]^* = a^*. < u \mid + b^*. < v \mid$
20. Si $< u \mid v > = 0$ entonces $\mid u >$ y $\mid v >$ son vectores ortogonales.
21. $\mid < u \mid v > \mid \leq \sqrt{\mid < u \mid u > \mid}.\sqrt{\mid < v \mid v > \mid}$. Desigualdad de Schwarz.

Adaptado del canal de youtube de Javier García, 17 de febrero de 2016.

La computación cuántica hace uso de la mecánica cuántica para su desarrollo. Los principios básicos de la mecánica cuántica descritos por Prieto (2019), son:

1. Superposición: Un elemento puede estar en dos o más estados simultáneamente, esto sucede antes de interactuar con el medio externo.

2. Entrelazamiento: Dos elementos están entrelazados cuando el cambio en uno de ellos nos produce un cambio en el otro, o cuando al conocer el estado de un elemento nos podemos informar del estado del otro.

3. Colapso de la Función de Onda: Ocurre cuando observamos un elemento que está en superposición, o cuando este elemento interactúa con el medio, automáticamente el elemento colapsa, es decir toma un valor o un estado. Por ejemplo, una moneda girando en el aire, ella está en estado de superposición, tomando varios valores al tiempo, cara o sello en ese caso, pero cuando la moneda toca el suelo, automáticamente la moneda se decide por un valor o estado, el cual puede ser cara o sello, pero no ambos.

12.3. Los Qubits

Un qubits es la equivalencia de un bits de la computación clásica, un qubits es la unidad mínima de información en la computación cuántica, también se le llama bits cuántico. Los qubits son elementos del espacio vectorial de Hilbert en dos dimensiones, con un campo escalar complejo, se representan con la notación de Dirac, o sea paréntesis llamados Kets $|0>$, $|1>$, estos vectores forman una base ortonormal del espacio vectorial en dos dimensiones (Sicard, 1999). Se escriben como vectores columnas de la siguiente manera:

$$|0> = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1> = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Teniendo en cuenta que el campo escalar es complejo, se sabe que $0 = 0 + 0i$, y que $1 = 1 + 0i$. Para poder construir el producto interior o producto escalar es necesario utilizar el dual de cada vector. Los vectores duales son:

$$|0>^* = \begin{pmatrix} 1 \\ 0 \end{pmatrix}^* = \begin{pmatrix} 1^* & 0^* \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix} = <0|, \qquad |1>^* = \begin{pmatrix} 0 \\ 1 \end{pmatrix}^* = \begin{pmatrix} 0^* & 1^* \end{pmatrix} = \begin{pmatrix} 0 & 1 \end{pmatrix} = <1|$$

Donde $1^* = (1+0i)^* = 1-0i = 1$, y $0^* = (0+0i)^* = 0-0i = 0$, por lo que el asterisco en un número complejo indica el conjugado, y el asterisco en un vector indica el vector dual.

Recordamos que el dual de un vector columna es el conjugado del vector fila. En términos de matrices, el dual de una matriz es la matriz transpuesta de la matriz conjugada.

El producto escalar de dos qubits está definido como:

$$|0>^* . |0> = <0| . |0> = <0|0> = \begin{pmatrix} 1 & 0 \end{pmatrix} . \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1$$

$$|1>^* . |1> = <1| . |1> = <1|1> = \begin{pmatrix} 0 & 1 \end{pmatrix} . \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 1$$

$$|0>^* . |1> = <0| . |1> = <0|1> = \begin{pmatrix} 1 & 0 \end{pmatrix} . \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 0$$

$$| 1 >^{*} . | 0 > = < 1 | . | 0 > = < 1 | 0 > = \begin{pmatrix} 0 & 1 \end{pmatrix} . \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0$$

Por lo anterior las bases $| 0 > = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $| 1 > = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ son ortonormales.

A los vectores duales de los Kets $| 0 >, | 1 >$, se les llama Bra $< 0 |, < 1 |$, en la notación de Dirac.

El 1-qubit más general que se puede formar, es una combinación lineal de las bases ortonormales, así:

$$| w > = a. | 0 > + b. | 1 >, \text{ donde } a, b \in \mathbb{C}; \text{ y } | a |^2 + | b |^2 = 1$$

La suma de los cuadrados de cada uno de los módulos de los complejos a y b es igual a uno, ya que el módulo de estos coeficientes determinan la probabilidad o densidad de probabilidad de ocurrir cada qubits. Por ejemplo:

$$| w > = a. | 0 > + b. | 1 > = a. \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b. \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$$

Por ejemplo supongamos que $| a |^2 = 0{,}20$ y $| b |^2 = 0{,}80$, tiene que cumplirse $| a |^2 + | b |^2 = 1$, lo que significa que existe la probabilidad en un 20% de que la función de onda, qubits o vector $| w >$ colapse o tome el valor de $| 0 >$, y una probabilidad del 80% de que colapse hacia $| 1 >$.

Si tenemos n espacios de memoria, en la computación clásica cabrían n bits, los cuales pueden tomar los valores cero ”0” o uno ”1”, mientras que si tenemos n espacios de memoria, en la computación clásica cabrían 2^n qubits, porque un sólo espacio de memoria puede contener una superposición de dos qubits, esta superposición está definida como se hizo anteriormente: $| w > = a. | 0 > + b. | 1 >$. Por ello podemos decir que, con la superposición cuántica, la capacidad para almacenar información de un computador cuántico es exponencial, mientras que en la computación clásica es lineal.

Una función de onda o qubits $| w >$, colapsa o se decide por un valor $| 0 >$ o $| 1 >$, cuando es observado o medido, esta acción provoca una interacción del qubits con el medio externo, este es un principio de la física cuántica. Se sabe que los sistemas cuánticos tienen una propagación de onda, o como campos, y se detectan como partículas. Murray Gell-Man (2003), describe la Teoría Cuántica de Campos así:

«Toda teoría respetable sobre las partículas elementales se desarrolla dentro del marco de la teoría cuántica de campos, que incluye tanto el modelo estándar como la teoría de supercuerdas. La teoría cuántica de campos está basada en tres supuestos fundamentales: la validez de la mecánica cuántica, la validez del principio de relatividad de Einstein (la relatividad especial cuando no se incluye la gravedad, y la relatividad general en caso

contrario) y la localidad (es decir, todas las fuerzas fundamentales surgen de procesos locales y no de la acción a distancia). Esos procesos locales incluyen la emisión y absorción de partículas».

En término generales establece que la materia genera campos o líneas de fuerza y que los campos de fuerza están cuantizados, hay un comportamiento de la materia como ondas, partículas y líneas de fuerzas.

«Cuando se dice que la materia está compuesta de partículas elementales —es decir, fermiones y bosones— debería hacerse notar que, bajo ciertas condiciones, algunos de estos bosones pueden comportarse como campos más que como partículas (por ejemplo, el campo eléctrico que rodea una carga). Los fermiones pueden describirse también en términos de campos que, aunque no se comportan en absoluto de manera clásica, pueden sin embargo asociarse, en cierto sentido, con fuerzas» (Gell-Mann, 2003).

En esta decripción vaga del universo cuántico, falta hablar de la energía, para ello Gell-Mann (2003), nos recuerda que materia y energía están en un ciclo de causa y efecto interminable:

«Toda la materia posee energía, y toda energía está asociada con materia. Cuando se habla a la ligera de la conversión de materia en energía (o viceversa), se está diciendo simplemente que ciertas clases de materia se están convirtiendo en otras. Por ejemplo, un electrón y su antipartícula asociada de carga opuesta, el positrón, pueden interaccionar y convertirse en dos fotones, un proceso que a menudo se describe como «aniquilación» o incluso «aniquilación de materia para producir energía». Sin embargo, no es más que una simple transformación de materia o, si se prefiere, de energía.»

12.4. El Qubit y sus Características

Un Qubit es la unidad mínima de información en la computación cuántica, también se le llama bits cuántico. También representa un comportamiento cuántico de una partícula cuántica.

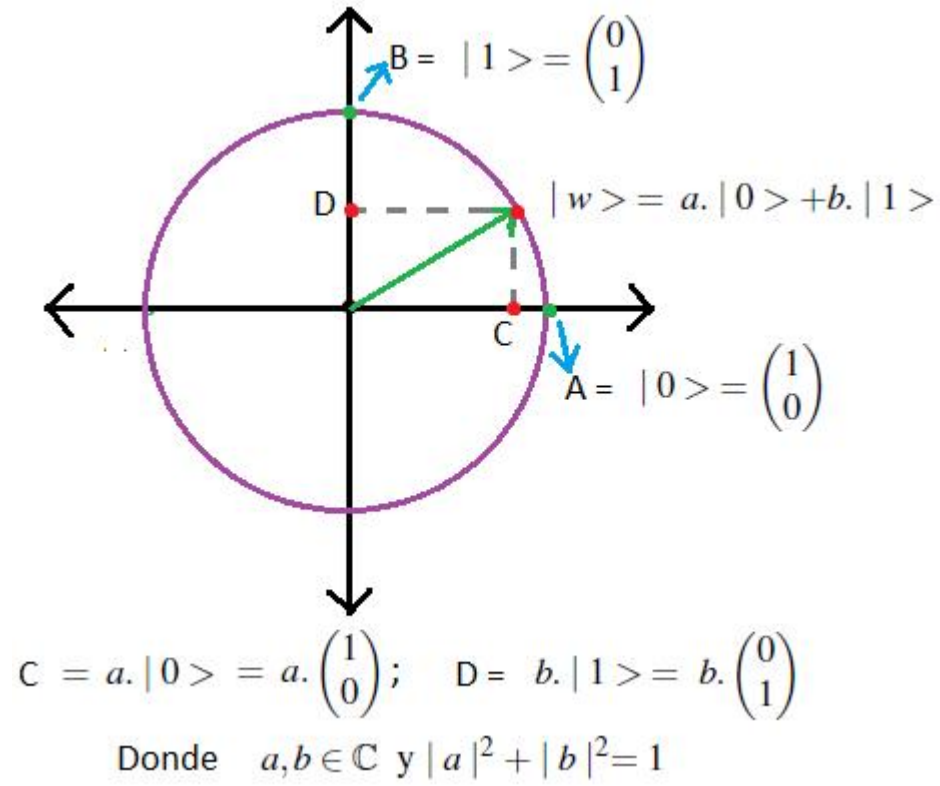


Figura 12.4.1: Gráfica bidimensional del Qubit. (Interpretación propia).

- **Superposición:** El Qubit $|w>$, tiene la probabilidad $|a|$ de colapsar hacia el Qubit $|0>$, y tiene la probabilidad $|b|$ de colapsar hacia el Qubit $|1>$.
- **Entrelazamiento:** Dos Qubits están entrelazados cuando el cambio en uno de ellos nos produce un cambio en el otro, o cuando al conocer el estado de un Qubit nos podemos informar del estado del otro.
- **Paralelismo:** Es la capacidad de una función o algoritmo de evaluar Qubits que toman valores simultáneos, Qubits que están en superposición cuántica.

12.5. Compuertas Cuánticas Sobre Un Qubits

Una compuerta cuántica es una operación, un operador o matriz que modifica el estado cuántico o qubits. Dicha matriz está compuesta de números complejos.

12.5.1. Compuerta Cuántica X o NOT

Esta compuerta intercambia el valor de probabilidad de leer los qubits $|0>, |1>$. Se representa mediante la siguiente matriz:

Compuerta Not: $X = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$. Sabemos que: $|\Psi> = a.|0> + b.|1> = \begin{pmatrix}a\\b\end{pmatrix}$

Entonces: $|\Psi' > = X.|\Psi > = X.\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix} = b.|0> + a.|1>$

De lo anterior podemos observar que:

Si $|0> = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, entonces $X.|0> = X.\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1>$

Si $|1> = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, entonces $X.|1> = X.\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0>$

También se observa que la compuerta X o Not, cambia un qubits $|0>$ por un qubits $|1>$, y viceversa.

12.5.2. Compuerta Cuántica U o Unidad.

Esta compuerta no hace nada, deja tal cual los valores de probabilidad de leer los qubits $|0>, |1>$. Se representa mediante la siguiente matriz:

Compuerta U: $U = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Sabemos que: $|\Psi > = a.|0> + b.|1> = \begin{pmatrix} a \\ b \end{pmatrix}$

Entonces:

$$|\Psi' > = U.|\Psi > = U.\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} = a.|0> + b.|1> = |\Psi >$$

De lo anterior podemos observar que:

Si $|0> = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, entonces $U.|0> = U.\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0>$

Si $|1> = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, entonces $U.|1> = U.\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1>$

Se observa que la compuerta U, es la matriz identidad, no cambia nada.

12.5.3. Compuerta Cuántica H o Hadamard.

Esta compuerta pone en estado de superposición a los qubits $|0>$ y $|1>$. Se representa mediante la siguiente matriz:

$$\text{Compuerta H: } H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Sabemos que: $|\Psi > = a.|0> + b.|1> = \begin{pmatrix} a \\ b \end{pmatrix}$

Entonces:

$$|\Psi'> = H.|\Psi> = H.\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.\begin{pmatrix} a \\ b \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} a+b \\ a-b \end{pmatrix} =$$

$$\frac{1}{\sqrt{2}}.(a+b).|0> + \frac{1}{\sqrt{2}}.(a-b).|1>$$

De lo anterior podemos observar que:

$$\text{Si } |0> = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \text{ entonces } H.|0> = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} =$$

$$\frac{1}{\sqrt{2}}.\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right] = \frac{1}{\sqrt{2}}(|0> + |1>) = \frac{1}{\sqrt{2}}.|0> + \frac{1}{\sqrt{2}}.|1>$$

Esto quiere decir que el estado $|0>$ pasa a ser una superposición de los estados $|0>$ y $|1>$, con una densidad de probabilidad de $\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$ cada estado. O sea que el qubits $|0>$ puede colapsar hacia $|0>$ o $|1>$, con una probabilidad del 50 %.

Lo mismo ocurre con el estado o qubits $|1>$.

$$\text{Si } |1> = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \text{ entonces } H.|1> = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix} =$$

$$\frac{1}{\sqrt{2}}.\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right] = \frac{1}{\sqrt{2}}(|0> - |1>) = \frac{1}{\sqrt{2}}.|0> - \frac{1}{\sqrt{2}}.|1>$$

Esto también quiere decir que el estado $|1>$ pasa a ser una superposición de los estados $|0>$ y $|1>$, con una densidad de probabilidad de $\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$ cada estado. O sea que el qubits $|1>$ puede colapsar hacia $|0>$ o $|1>$, con una probabilidad del 50 %.

Adaptado del canal de youtube de geneXCodeNow, 13 de febrero del 2020.

12.6. Compuertas Cuánticas Sobre Dos Qubits

Hemos visto compuertas cuánticas sobre u sólo qubits, como operaciones unarias, ahora veremos compuertas sobre dos qubits, como operaciones binarias.

12.6.1. Compuerta Cuántica CNOT: Controlled Not.

Esta compuerta actúa sobre dos qubits, al primero se le llama controlador, y es el que indica qué hay que hacer con el segundo qubits.

Si el primer qubits es cero, $|\Psi> = |0>$, el segundo qubits queda igual, no se le hace ningún cambio.

Si el primer qubits es uno, $|\Psi> = |1>$, al segundo qubits se le aplica la puerta X (NOT), es decir que al segundo qubits se le cambia su estado.

Para el caso donde se relacionan dos o más qubits, es necesario definir el Producto Tensorial o de Kroneker, entre vectores:

$$\text{Sea } |\Psi_1> = a.|0> + b.|1> = \begin{pmatrix} a \\ b \end{pmatrix} \quad \text{y} \quad |\Psi_2> = c.|0> + d.|1> = \begin{pmatrix} c \\ d \end{pmatrix}$$

Para formar dos qubits conjuntos, $|\Psi_1\Psi_2>$, utilizamos el producto tensorial:

$$|\Psi_1\Psi_2> = \begin{pmatrix} a \\ b \end{pmatrix} \bigotimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a.c \\ a.d \\ b.c \\ b.d \end{pmatrix}$$

Sabemos que $|0> = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, y $|1> = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Luego podemos formar qubits conjuntos, estos son estados compuestos de dos o más qubits. Analicemos los estados compuestos de dos qubits:

$$|0> \bigotimes |0> = |00> = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \bigotimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$|0> \bigotimes |1> = |01> = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \bigotimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$|1> \bigotimes |0> = |10> = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \bigotimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$|1> \bigotimes |1> = |11> = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \bigotimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

La compuerta cuántica CNOT, se representa mediante la matriz:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Luego, si aplicamos esta matriz a un estado $|\Psi_1\Psi_2> = \begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a.c \\ a.d \\ b.c \\ b.d \end{pmatrix}$,

obtenemos:

$$|\Psi'_1\Psi'_2> = CNOT|\Psi_1\Psi_2> = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} a.c \\ a.d \\ b.c \\ b.d \end{pmatrix} = \begin{pmatrix} a.c \\ a.d \\ b.d \\ b.c \end{pmatrix}$$

Sabemos que $|a.c|^2 = P(00)$, es decir que el valor de $|a.c|^2$es la probabilidad de que el qubits $|\Psi'_1\Psi'_2>$ colapse o se convierta en el qubits $|00>$. Lo mismo podemos observar para:

$|a.d|^2 = P(01)$, es la probabilidad de que el qubits $|\Psi'_1\Psi'_2>$ colapse en $|01>$.
$|b.d|^2 = P(10)$, es la probabilidad de que el qubits $|\Psi'_1\Psi'_2>$ colapse en $|10>$.
$|b.c|^2 = P(11)$, es la probabilidad de que el qubits $|\Psi'_1\Psi'_2>$ colapse en $|11>$.

De lo anterior, podemos observar que la Compuerta CNOT lo que hace es que el qubits $|\Psi_1\Psi_2>$ intercambie los valores de las probabilidades $|b.c|^2 = P(11)$ con $|b.d|^2 = P(10)$.

También podemos observar la compuerta CNOT como una negación condicionada, que se aplica a dos qubits, tal que si el primer qubits es $|0>$ entonces el segundo qubits no cambia, pero si el primer qubits es $|1>$ entonces el segundo qubits cambia a su negación o complemento. Por ejemplo:

$$\text{CNOT}.|00> = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00>$$

$$\text{CNOT}.|01> = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01>$$

$$\text{CNOT}.|10> = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |11>$$

$$\text{CNOT.}|\,11> = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |\,10>$$

Adaptado del canal de youtube de geneXCodeNow, 24 de Marzo del 2020.

12.7. Reversibilidad de la Compuertas Cuánticas

Esto quiere decir que si aplicamos dos veces seguidas la misma compuerta cuántica, el resultado será como si no hubieramos hecho nada, es decir como si hubiéramos aplicado la compuerta unitaria, esto es debido a que las compuertas cuánticas son matrices Unitarias y Hermitianas, o sea Ortogonales y Simétricas.

Por ejemplo:

$$X.X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = U$$

$$H.H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = U$$

$$CNOT.CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = U$$

Adaptado del canal de youtube de geneXCodeNow, 24 de Marzo del 2020.

12.8. Paralelismo Cuántico

El paralelismo cuántico es la capacidad para evaluar una función $f(x)$ en varios valores del dominio, de forma simultánea, utilizando para ello los estados de superposición de los qubits. Sabemos que un qubits puede estar en estado de superposición si tiene una probabilidad a^2 de colapsar hacia el estado $|\,0>$ y otra probabilidad b^2 de colapsar hacia el estado $|\,1>$. Es decir que el qubit se puede representar como una combinación $|\,q> = a.\,|\,0> + b.\,|\,1>$, donde $a, b \in \mathbb{C}$; y $|\,a\,|^2 + |\,b\,|^2 = 1$ (Jiménez y otros, 2020).

12.9. Algoritmos Cuánticos

En un Congreso Internacional de Matemáticos en París, el cual se llevó a cabo en el año 1900, el gran matemático alemán David Hilbert, expuso 23 problemas o retos matemáticos que a esa fecha no se les conocía solución. Uno de ellos, el décimo problema de Hilbert, pedía idear un procedimiento o proceso finito, que ahora entendemos como algoritmo, que permita determinar si cualquier polinomio con coeficientes enteros, tiene o no tiene solución en variables enteras (Videla, 2006). Un caso particular de este problema es el famoso «Último

teorema de Fermat». A este tipo de polinomios se les llama ecuaciones diofánticas.

En la formulación original del décimo problema de Hilbert, no se pide encontrar las soluciones a este tipo de polinomios, lo que solicita Hilbert es que se responda si tiene solución o no, a este tipo de problemas se les conoce como problemas de decisión.

El matemático inglés Alán Turing, inventó una estructura matemática, entre el año 1935 y 1936, para tratar el décimo problema de Hilbert, a este tipo de problemas Alán Turing le llamó «Entscheidungsproblem», que traduce del alemán como «Problemas de decisión». Hoy en día a esta estructura matemática se le conoce como «Máquina de Turing», y no es un objeto físico, es un elemento de la matemática abstracta (Penrose, 1996). Muchos académicos de áreas distintas de la matemática computacional, o informática teórica, creen que la palabra «máquina» se refiere a algún tipo de mecanismo arcaico de los inicios de la computación, nada más lejos de la realidad; Penrose (1996), afirma que, «todas las computadoras modernas de tipo general son en efecto, máquinas universales de Turing».

Penrose (1996), afirma, citando a David Deutsch (1985), que es posible construir una computadora cuántica que resuelve problemas de tipo No Polinomial (NP), como si fueran de tipo Polinomial (P). Es decir, los computadores cuánticos podrán resolver en tiempo Polinomial (P) algunos problemas que para la computación clásica son de tipo No Polinomial (NP). De aquí podemos concluir que los problemas son de clase P o NP, dependiendo de la lógica que los aborde. Algunos problemas de tipo No Polinomial en la Lógica Clásica serán de tipo Polinomial en la Lógica Cuántica; un ejemplo de ello es la descomposición en factores primos de números grandes, en la lógica clásica que es secuencial y unidimensional, se factorizan claves de 2048 bits en $4x10^{16}$años, mientras que en la lógica cuántica, a través del algortimo de Shor, este tipo de descomposiciones factoriales se resuelven en segundos (García, 2020).

12.9.1. Algoritmo Deutsch

Fue el primer algoritmo cuántico que superó a la versión de su algoritmo clásico, esto demostró que las computadoras cuánticas superan a las computadoras clásicas, al menos en algunos problemas específicos, (The Jupyter Book Community, 2020), esto se debe a que la computación cuántica reduce la complejidad de un algoritmo clásico.

Sea $f : \{0,1\} \longrightarrow \{0,1\}$, una función booleana. Se dice que f es una función constante si $f(0) = f(1)$, de lo contrario se dice que f es una función balanceada. ¿Si evaluamos una sola vez la función f, será posible determinar si esta función es constante o balanceada? Este problema es conocido como el problema de Deutsch (Sicard, 1999).

En la computación clásica no se puede determinar el tipo de función con sólo una evaluación, nos tocaría hacer dos evaluaciones. En la computación Cuántica sí es posible evaluar una misma función en dos valores diferentes al mismo tiempo, esto se conoce como «Paralelismo Cuántico» (Sicard, 1999).

Este algoritmo no nos mostrará los valores de las evaluaciones $f(0)$ o $f(1)$, más bien nos dirá si $f(0) = f(1)$ ó $f(0) \neq f(1)$, para nuestro caso nos dirá si la función es constante o balanceada (Jiménez y otros, 2020).

El algoritmo de Deutsch resuelve el problema para un único qubit, la generalización para n qubits se conoce como el algortimo de Deutsch-Jozsa (Jiménez y otros, 2020).

Capítulo 13

Análisis y Discusión de Resultados

Este trabajo se basa en lógicas deterministas y lógicas no deterministas, por ello se investigó estos conceptos. El determinismo es la posibilidad de predecir los resultados futuros a partir de los resultados iniciales, es como si metiéramos en una fórmula o en una ecuación los datos iniciales y de esa manera obtuviéramos los resultados futuros. El no determinismo toma los valores iniciales y arroja un resultado de un conjunto posible de respuestas, el mecanismo puede ser aleatorio o depender del comportamiento cuántico del sistema. El determinismo se representa en grafos lineales dado a que es secuencial, mientras que el no determinismo está representado por grafos no lineales, ya que no es secuencial.

Hay dos clases de Máquinas de Turing (MT) según su lógica, las máquinas deterministas que obedecen a la lógica clásica, y las máquinas no deterministas que obedecen a las lógicas no clásicas; la lógica clásica es el fundamento teórico de los computadores clásicos, mientras que las lógicas no clásicas son el fundamento de la computación cuántica; la lógica clásica es determinista y secuencial, mientras que la lógica cuántica es no determinista, probabilística y difusa. Es por ello que los computadores clásicos son deterministas y sólo pueden ejecutar algoritmos deterministas, mientras que los computadores cuánticos sí pueden ejecutar algoritmos no deterministas.

Los problemas polinomiales (P) y los problemas no polinomiales (NP), sí tienen solución en computadores clásicos, lo que no puede hacer un computador clásico es resolver un problema no polinomial (NP) en tiempo polinomial (P), es decir, resolver un problema de tiempo gigantesco en tiempo corto. Los computadores clásicos son deterministas, el tiempo para resolver un problema polinomial es muy corto, y el tiempo para resolver un problema no polinomial es muy largo, extremadamente gigantesco, es por ello que un problema no polinomial necesita ser resuelto en un computador no determinista, y este tipo de máquinas son computadores cuánticos. En la computación cuántica los problemas no polinomiales se resuelven en tiempos polinomiales, porque se utilizan algoritmos no deterministas en computadores no deterministas (Máquinas de Turing no deterministas). De aquí podemos concluir que las Máquinas de Turing no deterministas son computadores cuánticos, y las Máquinas de Turing deterministas son computadores clásicos.

Si queremos resolver un problema no polinomial (NP) en tiempo polinomial, debemos

utilizar algoritmos no deterministas, y la única máquina que desarrolla este tipo de algoritmos es una máquina cuántica.

Los problemas polinomiales (P) son problemas que sí tienen solución, y se pueden resolver en Máquinas de Turing deterministas en tiempos polinomiales, o sea en computadores clásicos. Un tiempo polinomial significa que es corto, aceptable, segundos, minutos, horas, ... o sea que una persona lo puede esperar sin perder la utilidad de la respuesta, es un tiempo determinado por un polinomio y por ello se llaman problemas polinomiales (P). Mientras que un tiempo no polinomial son tiempos inmanejables, intratables, que duran muchos años o milenios, algunos tiempos de orden exponencial tardarían más tiempo que la edad del universo.

Los problemas no polinomiales (NP) se definen de dos maneras:

- Son problemas que se resuelven por un computador clásico en tiempos no polinomiales, es decir, en tiempos exponenciales, tiempos muy grandes, siglos, milenios, etc. Tiempos que una persona no podría esperar, o que el tiempo de respuesta ya no sería útil esperarlo.
- Son problemas que se resuelven con algoritmos no deterministas en tiempos polinomiales. Aclarando que los computadores clásicos son deterministas y no pueden ejecutar algoritmos no deterministas, por ello este tipo de problemas no se pueden resolver en computadores clásicos en tiempos polinomiales. O sea que, si un problema NP se resuelve en un computador clásico, el tiempo de espera es astronómico, o sea no polinomial.

Los problemas NP contienen a los problemas de tipo NP-Completos, el teorema de Cook nos dice que los problemas NP-Completos son equivalentes entre sí, quiere decir que al poder resolver uno de ellos se podrían resolver todos bajo el mismo algoritmo. El teorema de Cook establece que todos los problemas NP-Completos son equivalentes a un problema de decisión; si un problema NP-Completo se pudiera resolver en máquinas deterministas en tiempos polinomiales, entonces NP sería igual a P (NP = P). Si se pudiera determinar que un problema NP-Completo no tiene solución, entonces ningún problema NP-Completo tendría solución, y quedaría demostrado que NP $\neq$P. El problema principal de P vs NP dentro de la Complejidad Computacional con la lógica clásica, es determinar si un problema no polinomial se puede resolver con un algoritmo determinista en tiempo polinomial. En este documento de tesis hemos sustentado por qué en la lógica cuántica, los problemas no polinomiales se pueden llegar a resolver en tiempos polinomiales con la computación cuántica, es decir que sustentamos por qué en la computación cuántica podría llegar a ser P = NP, basados en que la lógica cuántica desarrolla algoritmos no deterministas y los problemas NP se convierten en problemas polinomiales en Máquinas de Turing no deterministas.

Aunque sabemos que las máquinas probabilísticas son también no deterministas, es necesario aclarar que el no determinismo no siempre es cuestión de azar, sino más bien de que no hay una función matemática que determina una sola respuesta, por lo tanto, hay una relación matemática que, ante unas condiciones iniciales, puede originar cualquier respuesta dentro de un conjunto establecido. Como cuando voy al restaurante y le pido al mesero que escoja

el menú de mi almuerzo, sería distinto si yo escojo al azar, o si yo determino que a cada día de la semana le corresponda un determinado plato.

Lo que no queda claro es qué criterio utiliza la máquina no determinista, para escoger una respuesta dentro de un conjunto posible a determinadas condiciones iniciales; algunos como Martínez (2016), opinan que la máquina toma respuestas simultáneas, otros como Arjona (2007), describen el proceso no determinista como elegir una respuesta por tanteo, pero no explican qué hace que la máquina tome dicha decisión, es claro que hay una relación de transición para tomar decisiones, pero ¿cómo se ejecuta esta relación cuando el proceso no es determinístico ni probabilístico? Mientras que, si fuera una máquina probabilística, sería sencillo decir que se activa un mecanismo de azar que toma cualquier decisión dentro de un espacio muestral.

La respuesta de la pregunta anterior sería «clara», cuando la Máquina de Turing es cuántica, porque sabemos que la capacidad de la computación cuántica radica en sus características cuánticas, especialmente la superposición, el entrelazamiento y el paralelismo cuántico. La superposición le da la capacidad a los qubits de manejar muchos más estados; el entrelazamiento le da la capacidad de comunicarse o interactuar a distancia entre dos estados; pero el paralelismo cuántico le da la capacidad a un computador cuántico de ejecutar algoritmos de manera simultánea, esta capacidad es la que nos brindaría una Máquina de Turing no determinista en la lógica cuántica. Por ello podemos concluir que, un problema de tipo no polinomial (NP) dentro de la lógica de la computación clásica, se podría convertir en un problema de tipo polinomial (P) dentro de la lógica de la computación cuántica, utilizando algoritmos cuánticos que hacen uso de procesos simultáneos a través del paralelismo cuántico. La supremacía de la lógica cuántica en problemas no polinomiales (NP), sobre la lógica clásica, ya la había enunciado Deutsch (1985), citado por Penrose (1996), pero este documento tenía el complicado y extenso trabajo de explicar y sustentar esta tesis.

13.1. Conclusiones

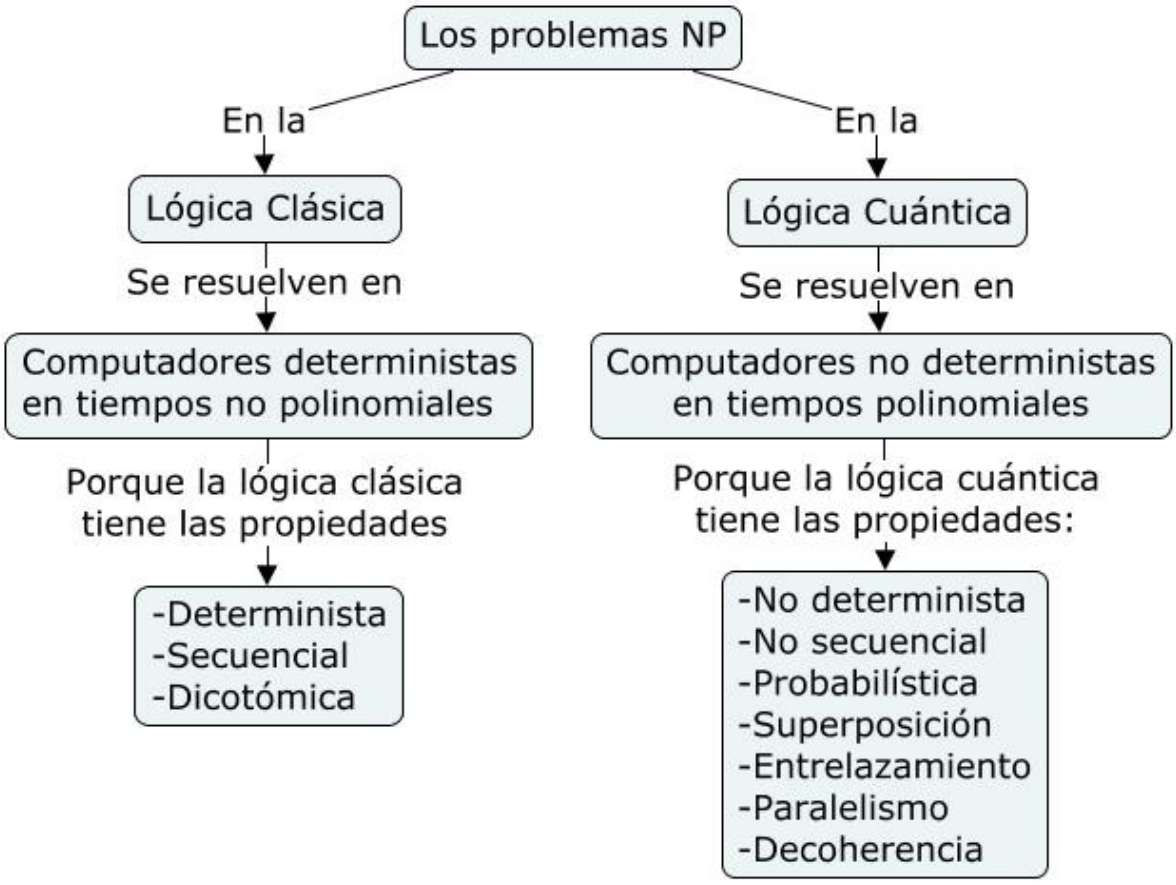


Figura 13.1.1: Mapa conceptual de las conclusiones, realizado con el software libre CmapTools.

1. En esta tesis se sustentó por qué un problema de tipo no polinomial (NP), dentro de la lógica de la computación clásica, se puede convertir en un problema de tipo polinomial (P), dentro de la lógica de la computación cuántica, exponiéndose que en la lógica de la computación clásica los problemas no polinomiales se resuelven con algoritmos no deterministas en tiempos polinomiales, se encontró que los computadores en la lógica clásica son deterministas, eso quiere decir que los computadores que utilizamos para nuestros trabajos, son secuenciales y no pueden ejecutar algoritmos no deterministas. En la lógica de la computación cuántica los computadores cuánticos sí pueden ejecutar algoritmos no deterministas, mediante el paralelismo cuántico, el cual nos da la capacidad de ejecutar procesos simultáneos. Es la capacidad de ejecutar algoritmos no deterministas, lo que le da la ventaja a la computación cuántica sobre la computación clásica, a la hora de resolver problemas no polinomiales en tiempos polinomiales, ya que como se observó, los computadores en la lógica clásica no pueden ejecutar algoritmos no deterministas, ya que los computadores clásicos son secuenciales.

2. En esta tesis se exploró el concepto de Máquinas de Turing, computabilidad y algoritmo, encontrándose que los computadores clásicos actuales son Máquinas de Turing deterministas, que un algoritmo es un proceso que se puede resolver o calcular por una Máquina de Turing, y que los problemas computables son los enunciados para los cuales sí existen algoritmos que los resuelve.

3. En esta tesis se exploró el concepto de complejidad computacional, llegándose a comprender esta complejidad como la ciencia que estudia la eficiencia de los algoritmos, en función de los recursos de memoria y tiempo que se utilizan en un sistema computacional; estos sistemas computacionales pueden ser físicos como un computador o teóricos como una Máquina de Turing. Se ha observado que la complejidad computacional está desarrollada en la lógica clásica, aún no existe una complejidad computacional desarrollada en la lógica cuántica, posiblemente porque, aunque se haya desarrollado la lógica cuántica, falta mucho desarrollo en la computación cuántica, algoritmos cuánticos, software cuántico y hardware cuántico. En palabras de Penrose (1996) el problema más importante en la teoría de la complejidad es la relación entre los problemas polinomiales (P) y no polinomiales (NP), y según Maldonado (2010), la relación entre P vs NP constituye el estudio y la investigación en Ciencias de la Complejidad.

4. En esta tesis se interrelacionó la lógica cuántica y la lógica clásica, observándose que la lógica cuántica es no determinista y no secuencial, mientras que la lógica clásica es todo lo contrario, determinista y secuencial. Dado que los problemas NP se pueden resolver en tiempos polinómicos con algoritmos no deterministas, concluimos que la lógica clásica no los puede resolver en tiempo polinomial, porque esta lógica no puede ejecutar algoritmos no deterministas, mientras que la lógica cuántica sí los puede resolver en tiempo polinomial, porque esta lógica sí puede ejecutar algoritmos no deterministas. A los algoritmos no deterministas les llamamos algoritmos cuánticos, dentro de la lógica cuántica, es por ello que la computación cuántica sí puede resolver problemas en tiempos polinómicos que la lógica clásica no puede, porque la propiedad del paralelismo cuántico de la lógica cuántica, le da la capacidad a la computación cuántica de ejecutar procesos simultáneos, en otras palabras, ejecutar algoritmos no deterministas.

Bibliografía

Addine, F., Recarey, S., Fuxá, M., & Fernández, S. (2020). Didáctica: teoría y práctica. Editorial Pueblo y Educación.

Arjona, J. L. O. (2007). Breves Notas sobre Complejidad.

Arora, S., & Barak, B. (2009). Computational complexity: a modern approach. Cambridge University Press.

Ausubel, D. P. (1976). Psicología educativa. Un punto de vista cognoscitivo. México: Ed. Trillas.

Cárdenas, M. M., Betancourth, G. L., & Quiza, L. A. P. (2002). Fundamentos de los Sistemas Dinámicos: La interdisciplinariedad desde los Sistemas No lineales. Universidad Surcolombiana.

Ciencia plus. (2021, 1 febrero). Primeros pasos hacia un cerebro cuántico. europapress.es. https://www.europapress.es/ciencia/laboratorio/noticia-primeros-pasos-cerebro-cuantico-20210201173554.html

Clay Mathematics Institute, C. M. I. (2018, 27 septiembre). Rules for the Millennium Prizes | Clay Mathematics Institute. Reglas de los premios Millennium. http://www.claymath.org/millennium-problems/rules-millennium-prizes

Cook, S. (2006). The P versus NP problem. The millennium prize problems, 87-104. Recuperado el 01 de febrero de 2021 de http://www.claymath.org/sites/default/files/pvsnp.pdf

Cortez, A. (2004). Teoría de la complejidad computacional y teoría de la computabilidad. RISI, 1(1), 102-105.

Deutsch, D. (1985), "Quantum theory, the Church-Turing principle and the universal quantum computer", Proc. Roy. Soc. (Lond.), A400, pp. 97-117.

Europa Press. (2019, 5 junio). Físicos logran salvar al gato de Schrödinger con un experimento. El Tiempo. https://www.eltiempo.com/vida/ciencia/cientificos-salvan-al-gato-de-schroedinger-con-un-experimento-370942

García, A. G. (2020). Computación cuántica y aplicaciones. Revista general de marina, 278(4), 635-640.

Gell-Mann, M. (1995). What is Complexity? Remarks on simpicity and complexity by the Nobel Prizewinning author of The Quark and the Jaguar. Complexity, 1.

Gell-Mann, M. (2003). El Quark y el Jaguar. Aventuras en lo simple y lo complejo. Tusquets Editores, S.A. Barcelona.

geneXCodeNow. (13 de febrero del 2020). Curso programación cuántica. T3 - Puertas Cuánticas. [Archivo de Video].
Youtube. https://www.youtube.com/watch?v=_uPBTPy7amI&t=2s

geneXCodeNow. (24 de marzo del 2020). Curso computación cuántica. T4 - Puertas Cuánticas II. [Archivo de Video]. Youtube. https://youtu.be/YpxObyiC6eE

Gowin, D. B. (1981). Educating. Ithaca, N.Y.: Cornell University Press.

Grupo de Computación Cuántica. (2003). Introducción al modelo cuántico de computación. Technical Report No. 19. Junio.

Javier García. (17 de febrero de 2016). 27.- La Naturaleza es un Espacio de Hilbert [Archivo de Video]. Youtube. https://www.youtube.com/watch?v=rDd-xH0ZFEs

Jiménez Jiménez, J. J., Regueira Fernández, A., & Sánchez Vélez, J. J. (2020). Diseño de algoritmos y circuitos cuánticos.

Jurado Málaga, E. (2008). Teoría de autómatas y lenguajes formales. Universidad de Extremadura. Servicio de Publicaciones.

Khan Academy. (2021). Recursividad. https://es.khanacademy.org/computing/computer-science/algorithms/recursive-algorithms/a/recursion

López García, C. A., & Fernández Veiga, M. (2002). Teoría de la Información y Codificación. Enxeñaría telemática.

Maldonado, C. E. (1999). Esbozo de una filosofía de la lógica de la complejidad. Visiones sobre la complejidad, 9-27.

Maldonado Carlos, E. (2001). Visiones sobre la complejidad. Pp5-42.

Maldonado, C. E. (2007). Complejidad: ciencia, pensamiento y aplicaciones. Books, 1.

Maldonado, C. E., & Gómez Cruz, N. A. (2010). Modelamiento y simulación de sistemas complejos. Editorial Universidad del Rosario.

Maldonado, C. E. (2013). Un problema fundamental en la investigación: Los problemas P vs. NP. Revista Logos, Ciencia & Tecnología, 4(2), 10-20.

Mañas, J. A. (1997). Análisis de algoritmos: Complejidad.

Martínez, I. A. R. (2016). Sobre la complejidad temporal de los algoritmos que buscan clanes.

Morin, E. (2005, June). Complexité restreinte, complexité générale. In Intelligence de la complexité: épistémologie et pragmatique, Colloque de Cerisy (pp. 28-50).

Palmero, M. L. R. (2011). La teoría del aprendizaje significativo: una revisión aplicable a la escuela actual. IN. Investigació i Innovació Educativa i Socioeducativa, 3(1), 29-50.

Penrose, R., & Jos˘Javier Garca' S'nz. (1996). La mente nueva del emperador. Consejo Nacional de Ciencia y Tecnologa'. Fondo de Cultura Econmica.

Pérez Gómez, A. (2006). A favor de la escuela educativa en la sociedad de la información y de la perplejidad. En Gimeno Sacristán, J. (comp.). La reforma necesaria: entre la política educativa y la práctica escolar. Ed. Morata/Gobierno de Cantabria. Madrid. Págs. 95-108.

Pérez, J. M. (2010). Inteligencia computacional inspirada en la vida (Vol. 36). Servicio Publicaciones UMA.

Petty, M. D. (2014). Modeling and validation challenges for Complex Systems. M S Journal, 9(1), 25-35.

Prieto Gil, M. (2019). Algoritmos cuánticos para la resolución del problema de satisfacibilidad booleana (Bachelor's thesis).

Reynoso, C. (2006). Complejidad y caos: una exploración antropológica. Argentina: Sb.

Rodríguez Palmero, M. L. (2004 a). La Teoría del Aprendizaje Significativo. Ponencia presentada en la FirstIntenationalConferenceon Concept Mapping. Pamplona (España), 14-17 de septiembre. Págs. 535-544.

Rodríguez Palmero, M. L. (2004 b). Aprendizaje significativo e interacción personal. En Moreira, M. A., Caballero Sahelices, C. y Rodríguez Palmero, M. L. Aprendizaje Significativo: Interacción personal, Progresividad y Lenguaje. Universidad de Burgos. Servicio de Publicaciones. Págs. 15-46.

Rodríguez Palmero, M. L. (2008). La Teoría del Aprendizaje Significativo. En Rodríguez Palmero, M. L. (org.): La Teoría del Aprendizaje Significativo en la perspectiva de la Psicología Cognitiva. Barcelona: Ed. Octaedro. Págs. 7-45.

Schrödinger, E. (1935), "Die gegenwartige Situation in der Quantenmechanik, Naturwissenschaften", 23, pp. 807-812, 823-828 844-849 (Traducido al inglés por J. T. Trimmer en Proc. Amer. Phil Soc 124 1980, pp. 323-338; y enJ. A. Wheeler y W. H. Zurek (comps.), Quantum theory and measurement, Princeton University Press, 1983

Seising, Rudolf. (2012). Warren weaver's "science and complexity" revisited. Studies in Fuzziness and Soft Computing. 273. 55-87. 10.1007/978-3-642-24672-2_3.

Shannon, C.E. (1948). A Mathematical Theory of Communication. The Bell System Technical Journal 27, 379–423, 623–656.

Shannon, C.E., Weaver, W. (1949). The Mathematical Theory of Communication. Univ. of Illinois Press, Urbana.

Sicard, A. (1999). Algunos elementos introductorios acerca de la computación cuántica. Memorias VII Encuentro ERM, Universidad de Antioquia, Medellín, agosto, 23.

Sipser, M. Introduction to the Theory of Computation. 2006. Thomson Course Tecnology.

The Jupyter Book Community. (2020). Deutsch-Jozsa Algorithm. 19 de octubre de 2020, de qiskit.org Sitio web: https://qiskit.org/textbook/ch-algorithms/deutsch-jozsa.html

Turing, A. (1936). On computable numbers with an application to the entscheidnungsproblem, Proc. London Math. Soc. 42, 230–265.

Uterra.com. (2021). El juego de los discos: La Torre de Hanoi. http://www.uterra.com/juegos/torre_hanoi.php

Vélez, M., & Sicard, A. (2001). El formalismo de la teoría gauge en la computación cuántica. Revista colombiana de física, 33(2).

Videla, C. R. (2006). El décimo problema de Hilbert, curvas elípticas y la conjetura de Mazur. Lecturas matemáticas, 27(3), 185-209.

Weaver, W. (1948 a). The Mathematics of Communication. Scientific American 181, 11–15.

Weaver, W. (1948 b). Science and Complexity. American Scientist 36, 536–544.

Zoya, L. G. R., & Zoya, P. G. R. (2014). El espacio controversial de los sistemas com- plejos. Estudios de filosofía, (50), 103-129.