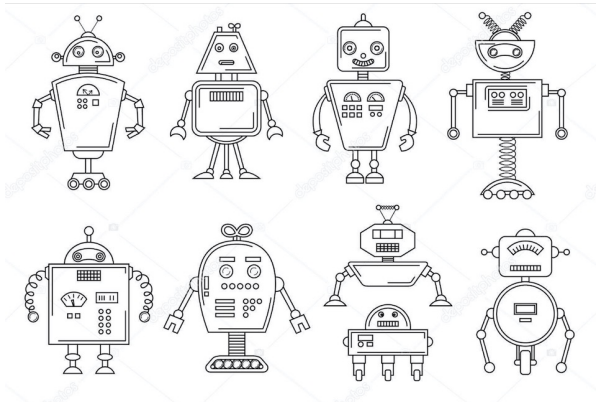
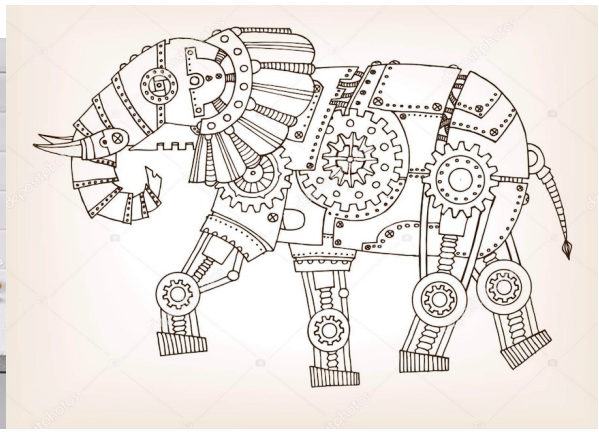
Lenguajes Formales, Autómatas y Computabilidad
Teoría de Lenguajes



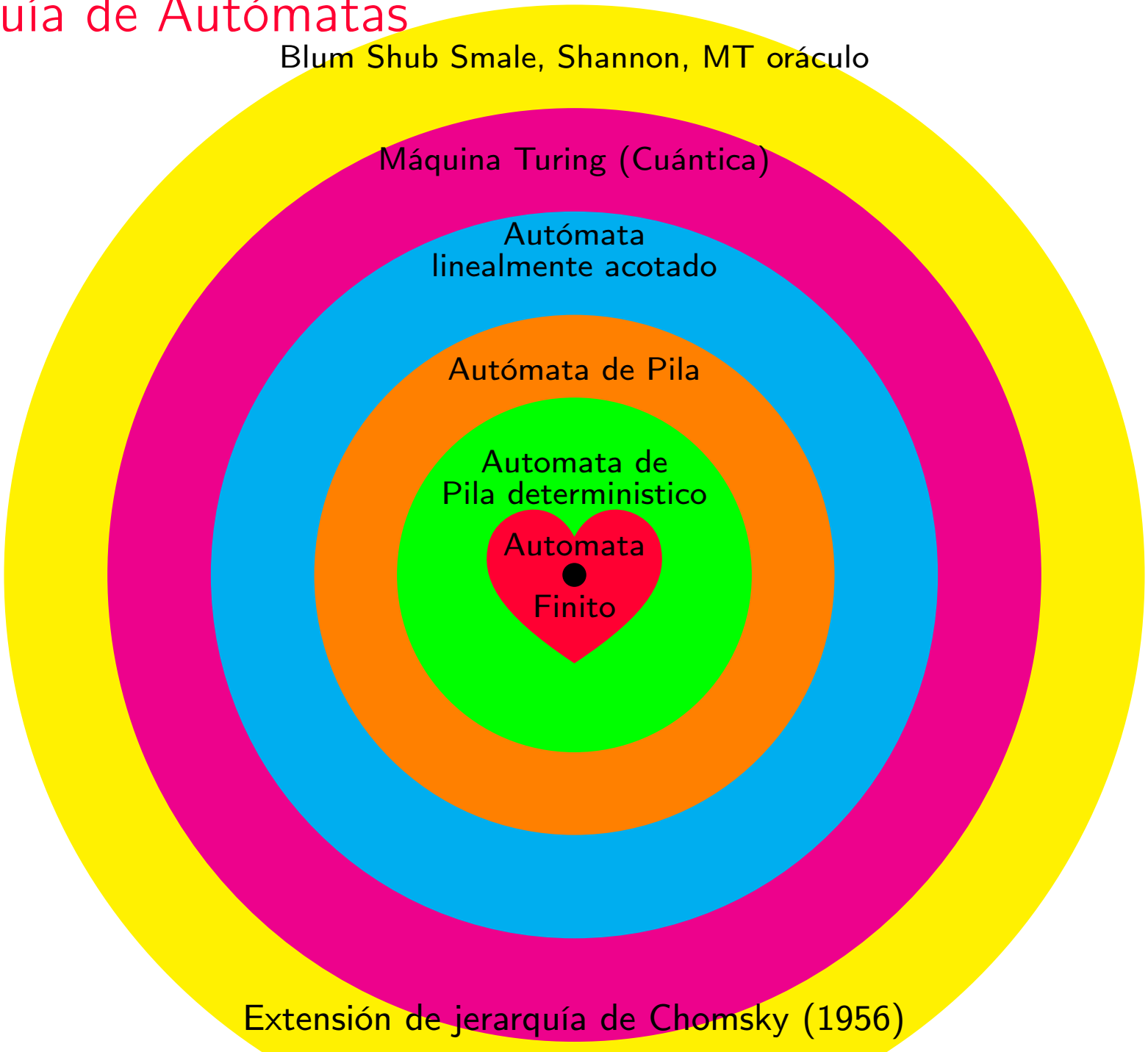
Teórica Final: Fiesta de autómatas



Autómatas y los lenguajes que reconocen

Autómata finito deterministico o no-determinístico	lenguajes regulares
Autómata de pila determinístico	lenguajes libres de contexto determinístico (lenguajes LR)
Autómata de pila no determinístico	lenguajes libres de contexto
Autómatas linealmente acotados	lenguajes sensitivos al contexto
Máquina de Turing	lenguajes recursivamente enumerables

Jeraquía de Autómatas



Autómatas computan funciones ¿de dónde a donde?

Aceptadores $\Sigma^* \to \{0,1\}$, Transductores $\Sigma^* \to \Sigma^*$
Más en general, subconjuntos particulares de funciones

$$\mathbb{N} \to \mathbb{N}$$
$$\mathbb{R} \to \mathbb{R}$$

Los modelos de cómputo que vimos
AFD $A = (Q, \Sigma, \delta, q_0, F)$,
$\delta : Q \times \Sigma \to Q$, $\mathcal{L}(A)$

AP $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$,
$\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$, $\mathcal{L}(A)$

MT $M = (Q, \Sigma \cup \{\mathbf{B}\}, \delta, q_0, q_f)$,
$\delta : Q \times \Sigma \cup \{\mathbf{B}\} \to Q \times \Sigma \cup \{\mathbf{B}\} \cup \{L, R\}$,
$f_M : \mathbb{N} \to \mathbb{N}$, $dominio(F_M) = \mathcal{L}(M)$.

Propiedades de clausura

Lenguajes Regulares: union, intersección, complemento, concatenación, clausura Kleene, reversa

Lenguajes libres de contexto: unión, concatenación, clausura Kleene, reversa, interseccion entre lenguaje libre de contexto y lenguaje regular

Lenguajes libres de contexto determinísticos: complemento, reversa, interseccion entre lenguaje libre de contexto determinisitico y lenguaje regular

Lenguajes c.e: unión, concatenación, clausura Kleene, reversa

Problemas decidibles

Regulares:
$x \in L$ (en tiempo lineal en longitud x),
L es vacío?,
L es finito?,
L es infinito?,
$L_1 = L_2$? , AFD A es minimo?

Libres de contexto determinístico:
$x \in L$ (en tiempo lineal en longitud de x),
L es vacío?,
L es finito?,
L es infinito?

Libres de contexto:
$x \in L$ (en tiempo cubico en longitud de x),
¿L es vacío?,
¿L es finito?,
¿L es infinito?

Computables:
¿ $x \in L$? ¿ $x \notin L$?

Computablemente enumerables: ?

Problemas indecidibles

Regulares:

Libres de contexto:
¿L es regular ?
¿L es libre de contexto determinístico?

Computables:
¿L es vacío?,
¿L es finito?
¿L es regular/libre de contexto?
¿$L_1 = L_2$?

Computablemente enumerables:
¿$x \in L$?
¿$x \notin L$?,
¿L es finito/regular/libre de contexto / computable?
¿$L_1 = L_2$?

¿Se cuelgan?

Decimos que un AF o un AP se cuelga cuando hace infinitas transiciones sin leer de la entrada Decimos que un MT se cuelga cuando hace infinitas transiciones sin llegar a un estado final.

AFD: no se cuelga para ninguna entrada.

AFND: no se cuelga para ninguna entrada .

AFND-λ: se puede decidir si tiene ciclo de λ y transformarlo en otro sin ciclos λ.

APD: se puede inspeccionar y decidir si tiene configuraciones cilcantes, y transformarlo en otro APD que no.

AP: dada una entrada $w \in \Sigma^*$ se puede decidir si tiene una configuración ciclante o no. Sin embargo, no se puede decidir si para todo $w \in \Sigma^*$ el AP no entra en configuración ciclante.

MT: HALT(x, y) es parcialmente computable pero no totalmente computable.

¿Qué poder computacional tienen estos autómatas?

¿El azar agrega poder computacional?

- Autómata finito cuántico
- Autómata finito
- Autómata finito de doble vía
- Autómata finito con un contador
- Autómata finito con múltiples cintas
- Autómata de pila
- Autómata de pila de doble via
- Autómata finito con dos contadores
- Autómata finito de dos pilas
- Autómata finito con una cola
- Máquina de Turing con múltiples cintas
- Máquina de Turing no determinística
- Máquina de Turing probabilística
- Máquina de Turing con oráculo
- Máquina de Turing cuántica
- Modelo de cómputo de Blum-Shub-Smale
- Máquina de Shannon 1941 de reales en reales
  –General Purpose Analog Computer (GPAC) —

Autómata finito de doble vía

Un autómata finito de doble vía es $M = (Q, \Sigma, L, R, \delta, s, t, r)$ donde
Q es conjunto finito de estados
Σ es al alfabeto de entrada
L marcador izquierdo
R marcador derecho
$\delta : Q \times (\Sigma \cup \{L, R\}) \to Q \times \{\text{left}, \text{right}\}$,
s el estado inicial,
t estado final (aceptación)
r estado de no-aceptación

- para todo $q \in Q$,
  $\delta(q, L) = (q', \text{right})$ para algún $q' \in Q$.
  $\delta(q, R) = (q', \text{left})$ para algún $q' \in Q$.
  Debe haber alguna transición posible cuando el puntero llega al marcador izquierdo o derecho.
- for all symbols $\sigma \in \Sigma \cup \{L\}$,
  $\delta(t, \sigma) = (t, \text{right})$, $\delta(r, \sigma) = (r, \text{right})$, $\delta(t, R) = (t, \text{left})$,
  $\delta(r, R) = (r, \text{left})$
  En los estados t y r , aceptación y no-aceptación, el puntero va a marcador derecha y cicla alli.

Autómata finito no determinístico de doble vía

Un autómata finito no-determinístico de dos vías (2NFA)
$M = (Q, \Sigma, L, R, \delta, s, t, r)$ donde

$$\delta : Q \times (\Sigma \cup \{L, R\}) \to 2^{Q \times \{\text{left}, \text{right}\}}.$$

Acepta una cadena de entrada si hay al menos un cómputo que acepta.

Autómata finito de doble via

Un autómata de doble vía es un autómata finito que puede releer su entrada

Teorema (Rabin and Scott 1959)

Los autómatas finitos deterministicos de doble vía son equivalentes a los autómatas finitos determinísticos.

El autómata finito determinístico equivalente a un autómata finito determinístico de doble vía puede requerir una cantidad exponencialmente mayor de estados.

Teorema (Rabin and Scott 1959)

Los autómatas finitos no-deterministicos de doble vía reconocen exactamente los lenguajes regulares.

Autómata finito con un contador

Un contador es una variable entera para la cual solamente podemos preguntar si es igual a 0. Las operaciones sobre el contador son sumar 1 y restar 1.

Un autómata contador es un autómata de pila que tiene solamente dos símbolos de pila, A y el símbolo inicial Γ.

Equivalentemente, un autómata contador es un autómata finito no determinístico con una celda de memoria adicional que contiene un número entero no negativo (de tamaño arbitrario) que puede ser incrementado, decrementado y se puede preguntar si es cero.

Lenguajes de contadores

La clase de autómatas contadores reconocen a un supraconjunto de los lenguajes regulares y un subconjunto propio de los lenguajes libres de contexto.

Por ejemplo, $\{a^n b^n : n \in \mathbb{N}\}$ no es regular y es aceptado por un autómata contador: supongamos el símbolo A para contar el número de as en una cadena de entrada x, escribiendo en la pila A por cada a en x), y luego borrando una A por cada b en x.

Máquinas de Turing

$M = (Q, \Sigma \cup \{\mathbf{B}\}, \delta, q_0, q_f)$
$\delta : \mathbb{Q} \times (\Sigma \cup \{\mathbf{B}\}) \to Q \times (\Sigma \cup \{\mathbf{B}\} \cup \{L, R\}$.
Define $f_M : \Sigma^* \to \Sigma^*$ parcial computable, que interpetamos como $f_N : \mathbb{N} \to \mathbb{N}$.
$\mathcal{L}(M) = domino(f_M)$.

Hay una máquina universal U

El poder computacional de una máquina de Turing

Las funciones computadas por las máquinas de Turing (llamadas funciones parciales computables) es el conjunto más pequeño de funciones parciales $\mathbb{N}^n \to \mathbb{N}$ (para cuaqlueir $n \in \mathbb{N}$) que

- contiene la función constante cero, la función sucesor y las funciones de proyección,
- está cerrado por las operaciones de composición, recursión primitiva y minimización no acotada.

El cómputo $u : \mathbb{N} \to \mathbb{N}$ en máquina de Turing se obtiene en tiempo y espacio discretos.
Consideramos una mapeo computable biyectivo entre $\mathbb{N} \times \mathbb{N}$ y $\mathbb{N}$,

$$< i, j >:= \frac{1}{2}(i + j - 2)(i + j - 1) + i,$$

para $i, j \in \{1, 2, 3, \dots\}$
Hay una máquina de Turing universal $U : \mathbb{N} \to \mathbb{N}$, $U(\langle i, n \rangle)$.

Autómata finito dos contadores = Máquina de Turing

Teorema (Teorema 8.15 Hopcroft, Motwani, Ullman)

1. *Para cada autómata finito con dos contadores M existe una máquina de Turing M' tal que $\mathcal{L}(M) = \mathcal{L}(M')$.*
2. *Para cada máquina de Turing M existe un autómata finito con dos contadores tal que $\mathcal{L}(M) = \mathcal{L}(M')$.*

Autómata de dos pilas = Máquina de Turing

Teorema (Teorema 8.13 Hopcroft, Motwani, Ullman)

1. *Para cada autómata finito con dos pilas M existe una máquina de Turing M' tal que $\mathcal{L}(M) = \mathcal{L}(M')$.*
2. *Para cada máquina de Turing M existe un autómata con dos pilas tal que $\mathcal{L}(M) = \mathcal{L}(M')$.*

Otras máquinas de Turing: múltiples cintas, no-determinsmo

Teorema

Para cada máquina de Turing M con m cintas existe un máquina de Turing M' con una sola cinta tal que $\mathcal{L}(M) = \mathcal{L}(M')$.

Teorema

Para cada máquina de Turing M no deterministica existe un máquina de Turing M' determinística tal que $\mathcal{L}(M) = \mathcal{L}(M')$.

Máquinas de Turing Probabilísticas

Definición

Una máquina de Turing probabilística es una máquina de Turing no-determinística que elige de manera aleatoria entre las transiciones disponibles, de acuerdo a una distribución de probabilidades. La aceptación de una cadena se define acotando el error de equivocación: la probabilidad de la respuesta incorrecta es menor que $1/3$.

La elección de $1/3$ en la definición puede reemplazarse por cualquier valor ente 0 y $1/2$ (exclusive) por ejemplo exigiendo error menor 2^{-n-c} donde c una constante positiva y n es la longitud de la entrada.

Notar que una máquina de Turing probabilística tiene resultados estocásticos, sobre un mismo input puede tener diferentes corridas o no terminar; es decir, puede aceptar en una ejecución y no terminar en otra.

Máquinas de Turing Probabilísticas

Teorema

Las máquinas de Turing probabilísticas aceptan los mismos mismos lenguajes que las determinísticas,

Porque las máquinas de Turing probabilísticas son máquinas no determinísticas, que ya sabemos que aceptan los mismos lenguajes que las determinísticas.

¿La aleatoriedad acelera la ejecución?

P: Problemas que pueden resolverse por medio de una máquina de Turing determinística en tiempo Polinomial en el tamaño de la entrada.

NP: Problemas que pueden resolverse por medio de una máquina de Turing No-deteminística en tiempo Polinomial en el tamaño de la entrada.

BPP: bounded-error probabilistic polynomial time (BPP).
Problemas que pueden resolverse con error acotado (para todas las instancias) en una máquina de Turing probabilística en tiempo polinomial en el tamaño de la entrada.

¿La aleatoriedad acelera la ejecución?

BPP se puede definir usando solamente máquinas de Turing determinísticas.

Un lenguaje L está en BPP si y solo si hay un polinomio p y una máquina de Turing determinística M tal que ejecuta en tiempo polinomial sobre todos los inputs (x, y), donde x corresponde al input de la máquina probabilística e y corresponde a los bits aleatorios que hubieran determinado la ejecución de máquina probabilística:

- para todo $x \in L$, la fracción de cadenas y de longitud $p(|x|)$ que satisfacen $M(x, y) = 1$ is mayor o igual que $2/3$,
- para todo $x \notin L$, la fracción de tales cadenas y que satisfacen $M(x, y) = 1$ es menor o igual que $1/3$.

Adleman, L. M. (1978). Two theorems on random polynomial time. Proceedings of the Nineteenth Annual IEEE Symposium on Foundations of Computing. pp. 75–83.

Ejemplo en BPP

Un problema en BPP y aún no se sabe si está en P es determinar si un polinomio dado es el polinomio 0: el problema es determinar si hay una asignación de valores a las variables tal que al evaluar un polinomio sobre esas variables el resultado es distinto de 0.

Primalidad estaba en BPP hasta que en 2002 se demostró en P: Manindra Agrawal, Neeraj Kayal and Nitin Saxena (AKS) dieron un algoritmo determinístico polinomial que decide si número entero dado es primo.

La clase BPP está cerrada por complemento, unión e intersección.

¿La aleatoriedad acelera la ejecución?

$P \neq NP$? Es la pregunta abierta más importante en la teoría de complejidad computacional: ¿todo problema que requiere una cantidad de operaciones polinomial (tiempo polinomial) en el tamaño de la entrada en una máquina de Turing no determinística no puede resolverse en tiempo polinomial en una máquina determinística?

$P = BPP$? ¿todo problema que puede resolverse en tiempo polinomial en una máquina de Turing probabilística también lo hace así en una determinística?

Máquinas de Turing Cuánticas

Paul Benioff y Yuri Manin (1980), Richard Feynman (1982) y David Deutsch (1985)

Definición

Una máquina de Turing cuántica es $M = \langle Q, \Sigma, \delta, q_0, q_f, P \rangle$ con una cinta infinita (en ambas direcciones) y una cabeza lectora-grabadora

- El conjunto de estados $Q = (c_1, \ldots c_q)$, $c_i \in \mathbb{C}^q)$, con q_0 estado inicial y q_f estado final.
- Σ es el alfabeto.
- La función de transición $\delta : Q \times \Sigma \to \mathbb{C}^{\Sigma \times Q \times \{L,R\}}$

La función de transición da para cada estado actual p, símbolo actual a, nuevo símbolo b, nuevo estado q, acción L o R, un número complejo (computable). El cómputo ahora es una secuencia de combinaciones lineales de configuraciones instantáneas. Se pide que esa transformación lineal sea unitaria. Esto se hace para poder interpretar al modulo cuadrado de δ como una probabilidad. El cómputo termina cuando todas las configuraciones en la llamada "superposición" están en el estado q_f.

El cómputo opera en espacio continuo y tiempo discreto.

Ejemplo de algoritmo para Máquina de Turing Cuánticas

Algoritmo de Shor para factorización de números enteros, 1994.

Corre en tiempo polinomial en una máquina de Turing cuántica:
para factorizar el número N requiere tiempo polinomial en $\log N$ que es la longitud de la entrada.

Específicamente requiere una cantidad de compuertas cuánticas del orden $O((\log N)^2(\log\log N)(\log\log\log N))$ usando multiplicación rápida.
Es significativamente más rápido que el algoritmo clásico de factorización de enteros que corre en tiempo sub-exponencial $O(e^{1,9(\log N)^{1/3}}(\log\log N)^{2/3})$.

La eficiencia del algoritmo de Shor se debe a la eficiencia de la transformada de Fourier cuántica y la exponenciacón modular por repetición de elevar al cuadrado.

Supremacía cuántica

Experimento realizado por Google y NASA en Octubre 2019 en la máquina cuántica de 53 q-bits. El resultado del experimento se llama Supremacía Cuántica, ya que demuestra que las computadoras cuánticas pueden resolver problemas exponencialmente más rápido que las computadoras no-cuánticas.

- Quantum Complexity Theory Ethan Bernstein and Umesh Vazirani. SIAM Journal on Computing, 26(5), 1411–1473, 1997 https://doi.org/10.1137/S0097539796300921

- NASA/TP-2019-220319 Quantum Supremacy Using a Programmable Superconducting Processor Eleanor G. Rieffel NASA Ames Research Center Agosto 2019 https://www-2.dc.uba.ar/staff/becher/supremacia.pdf

https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.132.030601
Verifying Quantum Advantage Experiments with Multiple Amplitude Tensor Network Contraction Yong Liu, Yaojian Chen, Chu Guo, Jiawei Song, Xinmin Shi, Lin Gan, Wenzhao Wu, Wei Wu, Haohuan Fu, Xin Liu, Dexun Chen, Zhifeng Zhao, Guangwen Yang, and Jiangang Gao Physical Review Letters 132, 030601 – Published 16 January 2024
NOTA: Physical Review Letters is the world's premier physics letter journal and the American Physical Society's flagship publication - Nota de Google sobre Supremacía cuántica October 23, 2019
https://ai.googleblog.com/2019/10/quantum-supremacy-using-programmable.html

https://www.nature.com/articles/s41586-019-1666-5

- Video de J. Martinis, Caltech, November 1, 2019 Quantum supremacy using a programmable superconducting processor

Autómata finito cuánticos

Un grupo es un conjunto de elementos con una operación binaria que es asociativa, tiene un elemento identidad y cada elemento tiene un inverso. Los lenguajes regulares de grupo son exactamente los lenguajes reconocidos por los autómatas finitos de permutación.

Un autómata finito determinista $A = (Q, \Sigma, \delta, q_0, F)$ es de permutación si cada símbolo del alfabeto induce una permutación sobre el conjunto de estados. Como las permutaciones son funciones invertibles, forman un grupo.

Ejemplo: El lenguaje de todas las palabras sobre $\{a, b\}$ con un número par de as y un número par de bs es un lenguaje regular de grupo.
Son reversibles. Subconjunto propio de los lenguajes regularees. Excluye a los lenguajes finitos.
AFQ $A = (Q, \Sigma, \delta, q_0, q_f, P)$,
$Q = (c_1, \ldots c_q)$, $c_i \in \mathbb{C}^q$),
$\delta : Q \times \Sigma \to \mathbb{C}^{Q \times \Sigma}$
$\mathcal{L}(A)$ usando δ, q_0, q_f, P.

Máquinas de Turing con oráculo

Agreguemos a una máquina de Turing una cinta infinita llena de 0s y 1s y agreguemos la instrucción de ler una celda en esta cinta. Estas son las Máquinas de Turing con oráculo.

Estas tienen más poder de cómputo que una máqina de Turing tradicional porque computan computan una clase de funciones más grande la clase de las funciones parciales computables.

Por ejemplo, consideremos el oráculo que codifica la la detención de las máquinas de Turing sin oráculos. Este oráculo no sirve para determinar si las máquinas de Turing con dicho oráculo se detendrán o no. Este hecho crea una jerarquía de máquinas, conocida como la jerarquía aritmética, en la cual cada una tiene un mayor potencial y a su vez un mayor problema de detención.

La computación interactiva donde el usuario interactúa con el programa , no es otra cosa que un cómputo con oráculo.

Máquinas idealizada BSS, tiempo discreto espacio continuo

Un modelo de Blum Shub Smale de cómputo idealizado que manipula números reales con infinita precisión:

- Los números reales son entidades básicas (con precisión infinita)
- Un paso computacional es una operación aritmética o una comparación
  (costo unitario)
- Los inputs y los outputs son vectores en $\mathbb{R}^n$. Si $x \in \mathbb{R}^n$ entonces $long(x) = n$.
- Los problemas de decisión se formalizan como subconjuntos de $\mathbb{R}^n$.

Operan en tiempo discreto y espacio continuo

Blum, Lenore; Shub, Mike; Smale, Steve (1989). "On a Theory of Computation and Complexity over the Real Numbers: NP-completeness, Recursive Functions and Universal Machines". Bulletin of the American Mathematical Society 21 (1): 1–46.

Máquina de Shannon, tiempo y espacio continuo

La computadora analógica de propósito general de Shannon creada en 1931 (en MIT con la supervisión de Vannevar Bush). El modelo de cómputo se explica mediante un conjunto de ecuaciones diferenciales. Opera en tiempo continuo y espacio continuo.

$$dy/dx = r(x, y)$$

$y(x)$ es la función que estamos modelando
dy/dx es su derivada con respecto a x
$r(x, y)$ es alguna expresión que relaciona x, y y la derivada

Ejemplo: $dy/dx = 2x^3$

A Survey on Analog Models of Computation Olivier Bournez and Amaury Pouly, 2018 "Handbook of Computability and Complexity in Analysis "Series:Theory and Applications of Computability Publisher: Springer in cooperation with the Association Computablity in Europe. Editors: Vasco Brattka and Peter Hertling, Handbook of Computability and Complexity in Analysis, 2018 https://arxiv.org/abs/1805.05729

Video june 2020 `https://www.youtube.com/watch?v=cW-epEUCIcE`

Máquina de Shannon, tiempo y espacio continuo

Caracterizaron la clase P de lenguajes computables en tiempo polinomial en términos de ecuaciones diferenciales cuya parte derecha es polinomial.

Polynomial Time corresponds to Solutions of Polynomial Ordinary Differential Equations of Polynomial Length Olivier Bournez, Daniel Silva Graça and Amaury Pouly Journal of the ACM (JACM), 64:38:1-38:76, 2017

On the Functions Generated by the General Purpose Analog Computer (arXiv) Olivier Bournez, Daniel Silva Graça and Amaury Pouly Information and Computation, 2017