Introducción

Carlos Linares López

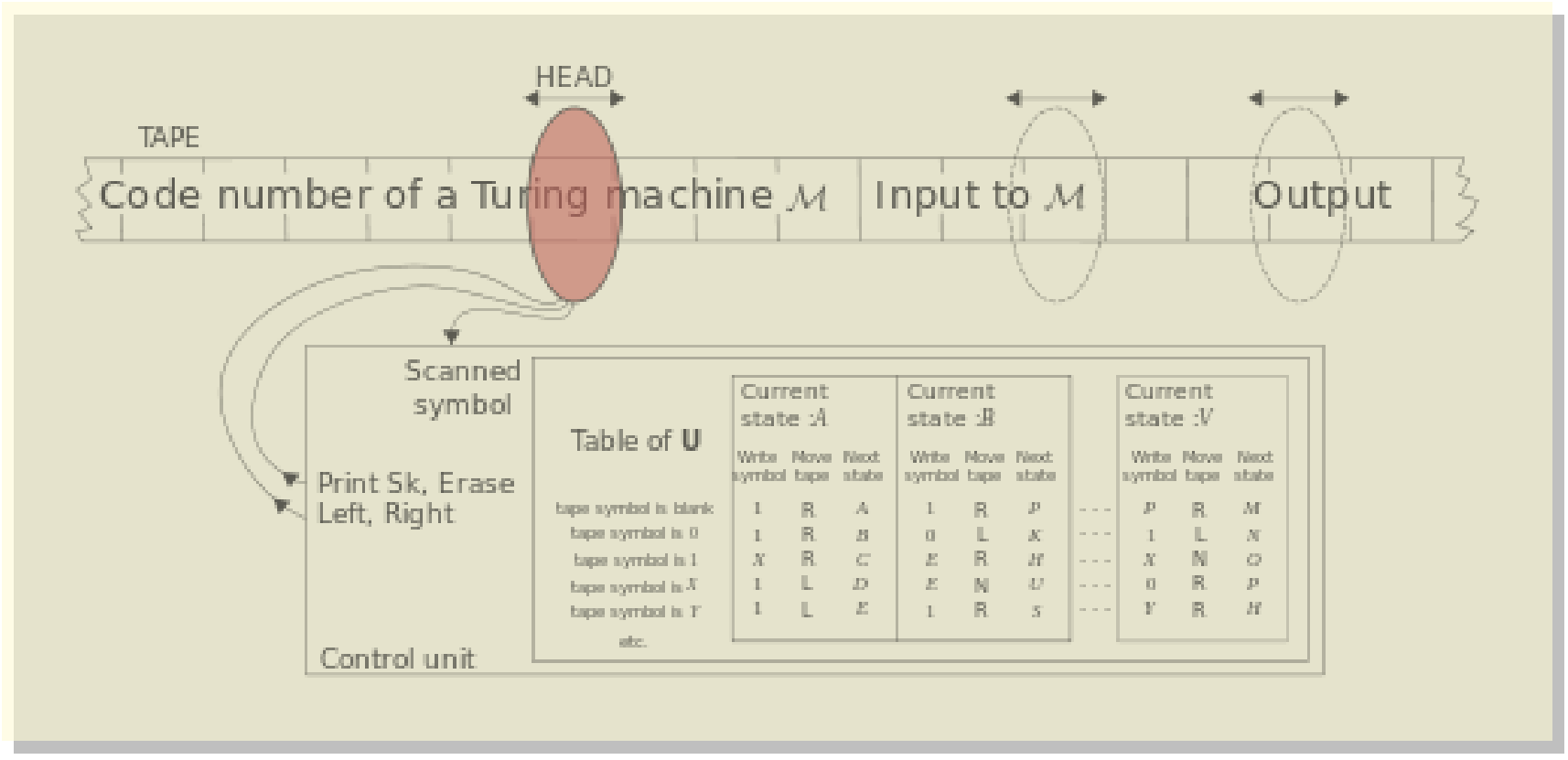
Grupo de Planificación y Aprendizaje (PLG)
Departamento de Informática
Escuela Politécnica Superior
Universidad Carlos III de Madrid

3 de septiembre de 2019

¿Qué es un ordenador? I

Un ordenador ¿es un *dispositivo secuencial simbólico (discreto)*?

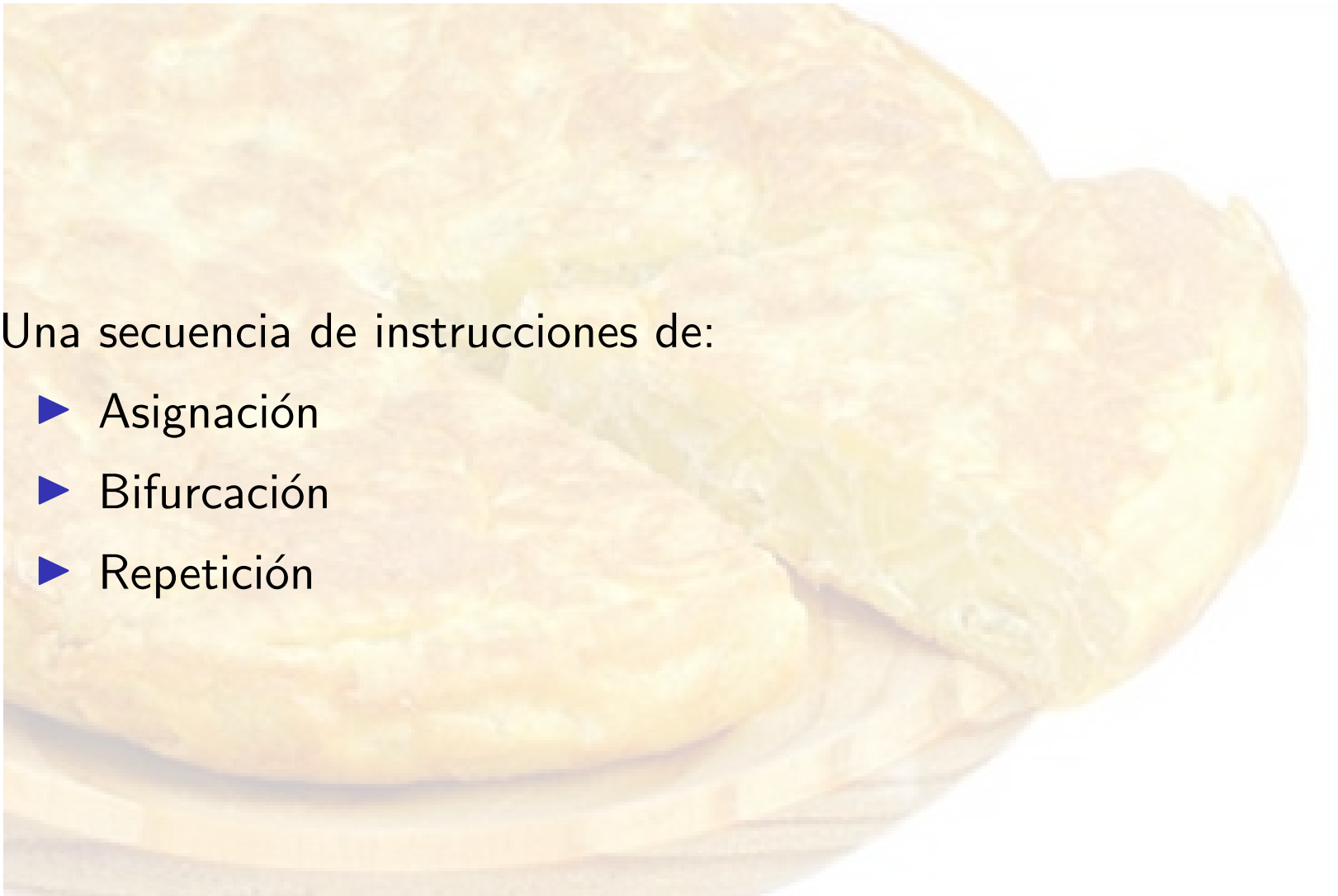
¿Qué es un ordenador? II



Máquina Universal de Turing

Una máquina de Turing $\mathcal{M}$ es *Universal* si puede ejecutar el algoritmo implementado en cualquier otra máquina de Turing $\mathcal{M}'$

¿Qué es un algoritmo? I



Una secuencia de instrucciones de:

- Asignación
- Bifurcación
- Repetición

¿Qué es un algoritmo? II

Consiste de:

- Entrada
- Procesamiento
- Salida

Entrada

No tiene por qué existir

Salida I

No tienen por qué parar ... ¡hay algoritmos que se ejecutan infinitamente! como la criba de Eratóstenes o la refutación de teoremas por contraejemplo

Salida II

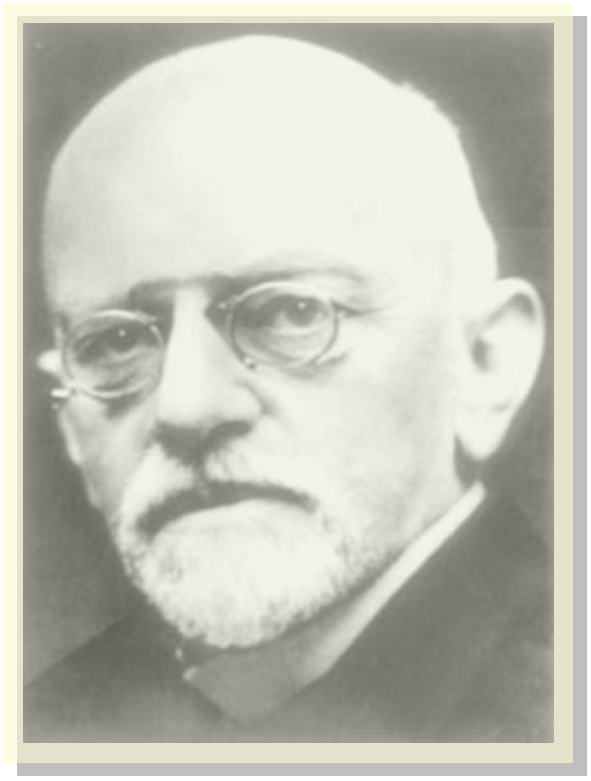
Problema de la parada

Dado un programa $\mathcal{P}$ y una máquina universal de Turing $\mathcal{M}$ determinar si la máquina $\mathcal{M}$ se detendrá eventualmente durante la ejecución de $\mathcal{P}$

Es un problema ¡indecidible!

Salida III

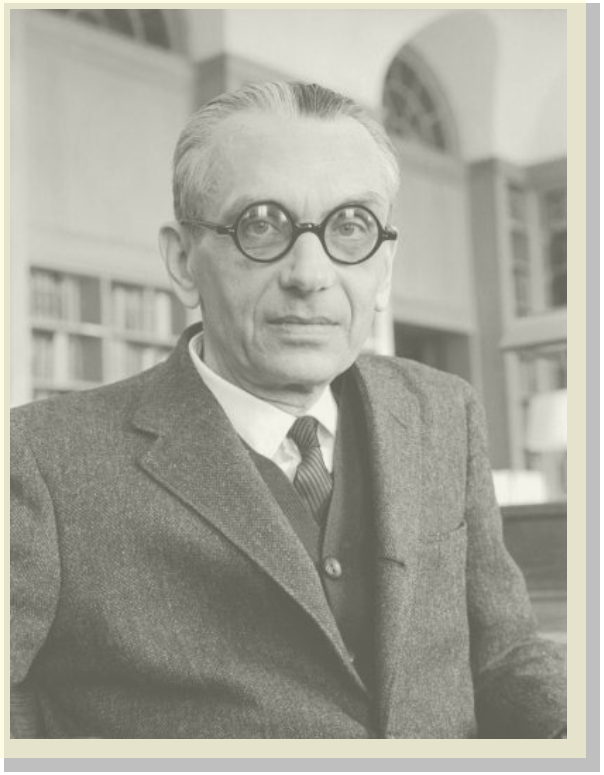
¡No es un resultado nuevo!



"Dada una ecuación diofántica con una cantidad arbitraria de incógnitas y coeficientes racionales, encontrar un proceso que decida en un número finito de pasos si es posible resolverla en el conjunto de los números racionales."

Encontrar un algoritmo general que decida si una fórmula del *cálculo de primer orden* es un teorema o no.

Salida IV



En 1931, Gödel demuestra su teorema de incertidumbre (el décimo problema de Hilbert)

Salida V

Se conjetura que son problemas no computables:

Problema de los pares primos ¿El número de números primos cuya diferencia es 2 es infinito?

$3n + 1$ La siguiente serie ¿converge siempre a 1?

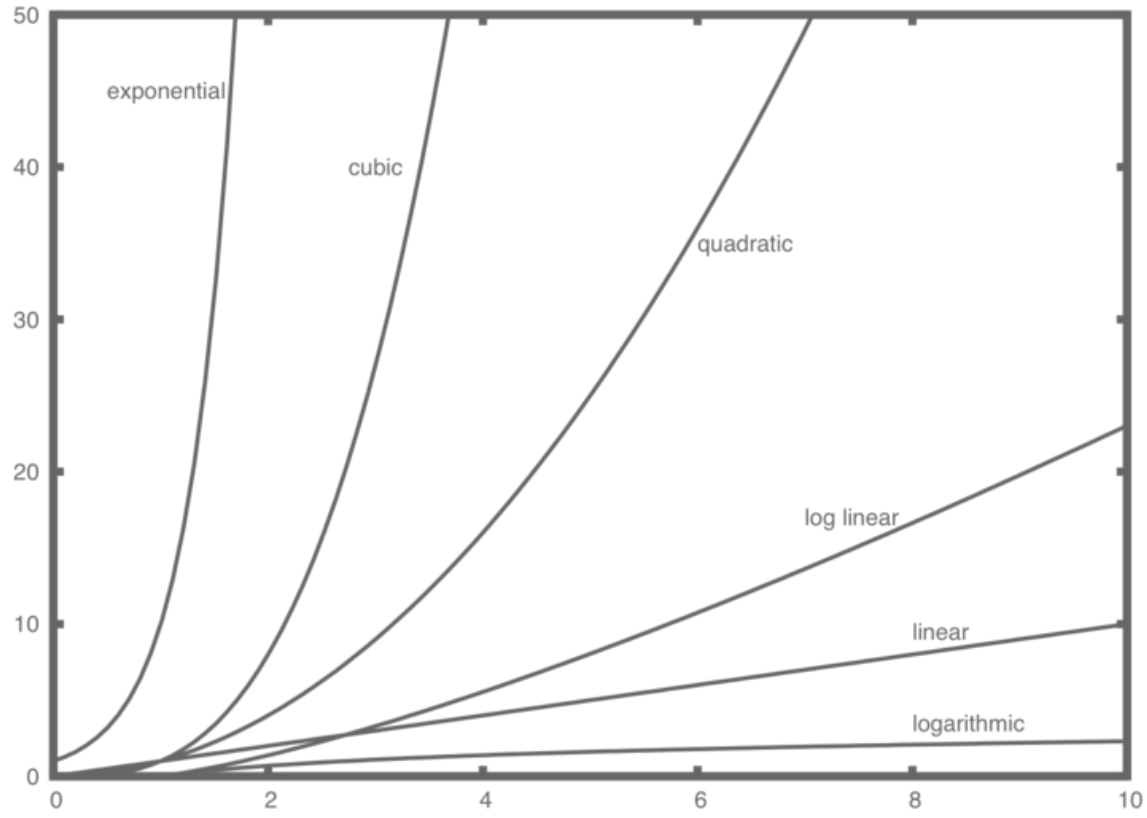
$$x_{n+1} = \begin{cases} 3x_n + 1 & \text{n impar} \\ \frac{x_n}{2} & \text{n par} \end{cases}$$

Por lo tanto, ¿qué es un algoritmo?

Fundamentalmente procesamiento

Análisis de algoritmos I

Tiempo y memoria de ejecución son recursos críticos



Análisis de algoritmos II

Problema de la suma del subconjunto de números

Dado un conjunto $\mathcal{S} = s_1, s_2, ..., s_n$ de n números, determinar si hay algún subconjunto $\mathcal{S}' \subseteq \mathcal{S}$ cuya suma sea exactamente igual a un valor T

El algoritmo que estás pensando tardaría más de un siglo en el ordenador más potente del mundo (Cray XT5, Oak Ridge, USA) con sólo 83 números
Sin embargo, ¡existen algoritmos pseudo-polinomiales!

Paralelismo

Las máquinas Universales de Turing pueden simular el paralelismo, aunque no necesariamente de forma eficiente.
No resuelven problemas incompletos ni cambian la clase de complejidad de los problemas

Computación cuántica

Scott Aaronson

"*Quantum computers are not known to be able to solve NP-complete problems in polynomial time, and can be simulated classically with exponential slowdown*"

La mejor promesa de la computación cuántica es que no es polinomialmente transformable en Máquinas Universales de Turing.

Algoritmos bio-inspirados

¡Los ordenadores no son cerebros! (aunque Turing lo creyera) y, en su lugar, son procesadores cuánticos.
¡Los ordenadores no se rigen por las leyes de la naturaleza! y, el Universo es también un procesador cuántico
Cualquier algoritmo bio-inspirado se puede simular con una máquina Universal de Turing, y algunos modelos ni siquiera resuelven problemas lineales.

¿Qué hace ...? I

... Inteligencia Artificial

Intentar encontrar modelos computacionales de procesos naturales que evidencian *inteligencia*

¡Y no tienen por qué hacerlo de la misma manera!

¿Qué hace ...? II

... Informática

¡Resolver problemas!

¿Los ordenadores resuelven?

"*The purpose of computing is insight, not numbers*"

(Richard Hamming)

Taxonomía (I)

Programación Lineal (Simplex) $f(x)$ es una *función lineal* y $S \subseteq \mathbb{R}^{+n}$ se representa con un conjunto de restricciones lineales $g(x) \leq b$ que se consideran iterativamente

- **Programación Entera** (Branch and Bound) Impone restricciones adicionales de integridad sobre algunas *variables de decisión*, $x_i = a_i$

Programación Dinámica (Bellman) Definición recursiva del método de maximización

Optimización en redes (Vogel, método húngaro, MST, SPP) Analiza el flujo de asignaciones o recorrido en redes

Programación no Lineal (Punto Interior) $f(x)$ y/o sus restricciones pueden ser no lineales

Taxonomía (II)

Problemas de satisfabilidad Identificación de modelos que satisfacen una definición $\langle X, D, C \rangle$ donde $X = \{x_1, \ldots, x_n\}$ es un conjunto de variables con dominios $D = \{D_1, \ldots, D_n\}$ y R es el conjunto de restricciones

- **Satisfabilidad lógica** (DPLL) X es un conjunto de proposiciones con $D_i = \{\bot, \top\}$ y las restricciones se describen como CNFs
- **Satisfacción de restricciones** (Propagación) Las restricciones se representan como relaciones $S_i \subseteq X$ que representan las asignaciones simultaneas legales

Taxonomía (III)

Optimización global (Heurística) en presencia de mínimos locales y se busca la mejor solución global

- **Analítica** (Gradiente, Newton-Raphson) Caracterización analítica de regiones convexas por las que se progresa hasta la solución
- **Heurística** (A^*) Definición de problemas como grafos donde la solución consiste en un camino $\langle s, t \rangle$
- **Metaheurística** (*Simulated annealing*) simulación estocástica del proceso de resolución dominado por variables externas

Desafortunadamente, no veremos ...

Análisis de algoritmos ($O(\cdot)$, $\Theta(\cdot)$, $\Omega(cdotp)$, $o(\cdot)$) Análisis del comportamiento del peor caso, medio y mejor

Complejidad algorítmica (NP-Completitud) Análisis de la complejidad del uso de recursos (tiempo y/o espacio) de un conjunto relacionado de problemas (*Gödel's Lost Letter and P=NP*, 7 Noviembre 2010)

Algoritmos de aproximación (*derandomization*) modelos de aproximación que garantizan la solución en un determinado intervalo

Teoría de scheduling/planning (representación, resolución) Modelos independientes de resolución de problemas con recursos o sin ellos

¡Entrénate!

Juega en http://www.projecteuler.net
Juega en http://www.hackerrank.com
Pregunta en http://cs.stackexchange.com