

Turing y la computación programable

Juan Seguro

Pregunta central

¿Cómo separó la idea de un procedimiento universal de cómputo la computación de una máquina concreta e hizo de los programas, en vez de los mecanismos fijos, una forma general de organizar trabajo simbólico?

Contexto histórico y evolutivo

Las calculadoras mecánicas y los sistemas de tarjetas perforadas existían antes de los ordenadores electrónicos, pero solían diseñarse en torno a operaciones o flujos de trabajo restringidos. En 1936–1937 Alan Turing formalizó la computación mediante una máquina abstracta que manipula símbolos según reglas finitas y mostró que una máquina universal podía simular cualquier máquina de esa clase. Era una respuesta matemática a una cuestión de lógica, no un plano de todos los ordenadores posteriores. Durante la década de 1940, la conmutación electrónica, el criptoanálisis de guerra, los conceptos de programa almacenado y grupos de ingeniería en Gran Bretaña y Estados Unidos convirtieron ideas relacionadas en máquinas físicas. El informe EDVAC circuló en 1945; el Manchester Baby ejecutó un programa almacenado en 1948; y EDSAC entró en operación práctica en 1949. La discusión de Turing de 1950 sobre maquinaria de computación amplió después la cuestión desde el cálculo hacia el aprendizaje y el comportamiento inteligente.

¿Qué podían saber las comunidades antes de esta transición?

Calculistas humanos, calculadoras de escritorio, tabuladoras y máquinas de propósito específico podían ejecutar procedimientos largos. La lógica formal había aclarado algoritmos y reglas simbólicas, mientras que la conmutación eléctrica mostraba que los circuitos podían implementar estados discretos. La limitación era una fragmentación conceptual: procedimiento, datos y máquina se trataban a menudo como estrechamente ligados a una sola tarea.

¿Qué cambió?

La computación se convirtió en una clase formal de procedimientos de procesamiento simbólico

El artículo de Turing de 1936 definió números computables mediante una máquina abstracta deliberadamente sencilla. La importancia no estaba en la metáfora de la cinta, sino en demostrar que procedimientos complejos podían descomponerse en pasos elementales gobernados por reglas.

La universalidad separó arquitectura de máquina e identidad de tarea

Una máquina universal podía codificar la descripción de otra máquina y simularla. Esto hizo posible razonar sobre un dispositivo de cómputo de propósito general cuyo comportamiento cambia mediante instrucciones simbólicas. Las máquinas posteriores de programa almacenado implementaron esta separación en forma de ingeniería, pero Turing no fue el único inventor del ordenador electrónico programable: hardware, memoria, conmutación y arquitectura surgieron a través de múltiples grupos.

Los programas se convirtieron en objetos analizables, almacenables y modificables

Una vez representadas simbólicamente, las instrucciones pueden copiarse, depurarse, compararse y transformarse. En 1950 Turing discutió explícitamente ordenadores digitales y máquinas de aprendizaje, ayudando a conectar la computación formal con cuestiones posteriores de inteligencia artificial.

¿Por qué pudo surgir esta capacidad?

La transición requirió lógica matemática, notación para procedimientos formales, conmutación electromecánica y electrónica, inversión bélica, tecnologías de memoria y equipos capaces de traducir algoritmos en operaciones de máquina. También exigió un cambio institucional desde cálculos únicos hacia infraestructura computacional reutilizable.

¿Qué formas posteriores de conocimiento hizo posibles?

La computación programable permitió simulación numérica, procesamiento automatizado de datos, compiladores y sistemas de software, comunicación digital, análisis de secuencias moleculares, teledetección y aprendizaje automático. Creó además nuevos objetos de estudio como complejidad, computabilidad, corrección del software y límites algorítmicos.

Nuevos límites y riesgos

La universalidad no hace tratables todos los problemas. Los límites de computabilidad, la memoria finita, el error numérico, la complejidad y la calidad de datos restringen lo que las máquinas pueden ofrecer. El software crea además opacidad y dependencia: un resultado puede depender de código que pocos usuarios inspeccionan. La programabilidad puede automatizar con la misma facilidad formal procedimientos beneficiosos y perjudiciales.

¿Qué cambió en la arquitectura del conocimiento?

Un procedimiento se convirtió en un **objeto simbólico portátil ejecutable por una máquina general**. El conocimiento podía codificarse no solo como proposiciones o ecuaciones, sino como instrucciones que transforman representaciones paso a paso.

Pérdida, exclusión e incertidumbre

El papel de Turing es fundacional, pero una narrativa de un solo héroe borraría a Church, Gödel, Post, von Neumann, Flowers, Kilburn, Williams y numerosos ingenieros y operadores. Universalidad formal, construcción electrónica y adopción del programa almacenado son transiciones distintas. PIO-082 trata, por tanto, la contribución de Turing como bisagra conceptual dentro de una historia tecnológica distribuida.

Mapa de dependencias

- INDIRECTO ← PIO-056 Probabilidad, censos y error: de Pascal a Quetelet
- INDIRECTO ← PIO-074 Redes eléctricas y sistemas tecnológicos
- PARALELO → PIO-084 Shannon y la información cuantificable
- DIRECTO → PIO-090 Redes, simulación y aprendizaje automático

Referencias

1. Turing, A. M. «On Computable Numbers, with an Application to the Entscheidungsproblem.» *Proceedings of the London Mathematical Society* s2-42(1), 230–265 (1937). DOI: 10.1112/plms/s2-42.1.230.
2. Turing, A. M. «Computing Machinery and Intelligence.» *Mind* LIX(236), 433–460 (1950). DOI: 10.1093/mind/LIX.236.433.
3. Copeland, B. J., ed. *The Essential Turing*. Clarendon Press, Oxford (2004).