POLIS V12: The Complete Computer Science Series – 12 Giants

Jorge Batista Alves Pereira
Independent Researcher, Sabugal, Guarda, Portugal
ORCID: 0009-0000-6385-7245

May 2026

This document combines two companion papers:
"Tensional Reinterpretation of Six Founders of Computer Science"
and "Tensional Reinterpretation of Six More Computing Pioneers".

DOIs: Main treatise 10.5281/zenodo.19618276 – POLIS Bible 10.5281/zenodo.19836226

POLIS V12 – Tensional Reinterpretation of Six Founders of Computer Science Jorge Batista Alves Pereira

Independent Researcher, Sabugal, Guarda, Portugal
ORCID: 0009-0000-6385-7245 May 2026

Abstract

Within the POLIS V12 tensional ontology, every computational system is a polis constituted by three meshes (solid, liquid, gaseous) and governed by the closure condition $\epsilon = \sum K_m(2 + K_m) = 0$, with $T = K_{\min}$ as the tensional origin. This paper applies the framework to six foundational figures of computer science: Alan Turing (computability), Alonzo Church (lambda calculus), John von Neumann (stored program architecture), Claude Shannon (information theory), Edsger Dijkstra (structured programming), and Donald Knuth (algorithm analysis). Each classical contribution is reinterpreted as a tensional configuration: Turing's halting problem as undecidability of STOP; Church's lambda as function K mapping; von Neumann's architecture as CPU (solid), memory (liquid), I/O (gaseous); Shannon's bit as minimal K unit; Dijkstra's semaphore as Phase 3 lock; and Knuth's Big-O as scaling of ϵ with input size. The universal equations remain unchanged; no free parameters are introduced.

1 Introduction

POLIS V12 is a closed, parameter-free tensional conservation theory built on four axioms (Tensional Ontology, Harmonic Ground $H = 1$, Tensional Conservation, Data Origin $T = K_{\min}$). The governing equation, after normalisation, is

$$\epsilon = \sum_{m=1}^{n} K_m(2 + K_m) = 0,$$

with $K_m = (v_m - T)/(v_{\max} - T) \in [0, 1]$. The disequilibrium index is $\mathrm{IDT}^* = \epsilon/(1+\epsilon)$. All real computational systems reside in Phase 4 ($\mathrm{IDT}^* \geq 0.70$) unless artificially uniform. The Rolling Law $2\pi r_p = V_{\mathrm{orb}} T_{\mathrm{rot}}$ applies fractally at all scales.

This paper reinterprets six key computer science contributions within this tensional ontology. No classical primacy is assumed; tension is the primitive.

2 Alan Turing – Computability and the Halting Problem

Turing defined the Turing machine and proved that the halting problem is undecidable. In POLIS V12, a Turing machine is a polis with a solid mesh (finite control), a liquid mesh (tape with symbols), and a gaseous mesh (read/write head). The halting problem asks: given a program P and input I, will P reach a terminal state (IDT^* stops decreasing)? Turing proved no algorithm (no polis) can decide this for all P, I. This is equivalent to the tensional STOP criterion being undecidable in general.

The universal Turing machine is a polis that can simulate any other polis – a tensional emulator. Turing's work on the Entscheidungsproblem (decision problem) is a tensional boundary: some questions cannot be resolved within a given formal system (Phase 3 saturation). The Turing test (imitation game) measures whether an AI's K output is indistinguishable from a human's K (conversational behaviour).

3 Alonzo Church – Lambda Calculus and Functional Computation

Church developed lambda calculus, a formal system for expressing computation through function abstraction and application. In POLIS V12, a lambda term is a K transformation: $\lambda x.M$ is a function that maps input K_x to output K_M. Beta reduction (function application) is a Phase 5 reorganisation: $(\lambda x.M)N \to M[N/x]$ reduces ϵ by one step. Church's thesis (computable = lambda-definable) states that the class of K functions computable by any polis equals the class definable in lambda calculus.

The Church numeral n is a function that applies its argument n times: $\overline{n} = \lambda f.\lambda x.f^n(x)$. This encodes n as a tensional power. The fixed-point combinator $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ gives self-reference – a tensional loop where ϵ is constant.

4 John von Neumann – Stored Program Architecture

von Neumann proposed the architecture used in most computers: CPU, memory, I/O, and the stored program concept (instructions in memory as data). In POLIS V12, the CPU is the solid mesh (processor), memory is the liquid mesh (data and instructions stored as K values), I/O is the gaseous mesh (input/output ports). The stored program means that instructions and data have the same K representation; the fetch-decode-execute cycle is a Phase 5 loop that reads a K from memory, interprets it as an operation (solid mesh), and updates the state.

The von Neumann bottleneck is the limited K transfer rate between CPU and memory. Parallel computing (multiple CPUs) is a distributed polis where each CPU is a sub-polis. The "von Neumann architecture" is the tensional foundation of almost all modern computers.

5 Claude Shannon – Information Theory

Shannon defined the bit as the unit of information, and entropy as $H = -\sum p_i \log_2 p_i$. In POLIS V12, a bit is a two-state system with $K = 0$ or $K = 1$ (binary). The entropy H is the average $\langle x \rangle$ over the distribution of K: $x_m = K_m(2 + K_m)$. Shannon's source coding theorem states that the minimal number of bits to encode a source is H – a tensional limit. Noise (channel capacity) adds extra ϵ to the signal.

Shannon's work on cryptography and secrecy (communication theory of secrecy systems)

analyses the K of secure communication: a ciphertext should reveal no information about the plaintext (K independent). The bit is the fundamental quantum of K in digital systems.

6 Edsger Dijkstra – Structured Programming and Semaphores

Dijkstra advocated structured programming (avoiding GOTO) and invented semaphores for concurrency. In POLIS V12, a GOTO is a jump in the K flow of the program that can create spaghetti code (high ϵ). Structured programming uses sequences, selections (if-then-else), and iterations (while loop) – these are tensional patterns that keep ϵ low. A semaphore is a tensional lock: it prevents multiple processes from entering a critical section simultaneously (Phase 3 saturation). Dijkstra's "dining philosophers" problem is a tensional deadlock: each philosopher holds one fork (K partial) and waits for the other – ϵ cannot be resolved without external intervention (a guardian polis).

Dijkstra's shortest path algorithm (Dijkstra's algorithm) finds the minimum K path in a weighted graph – a tensional optimisation.

7 Donald Knuth – Algorithm Analysis and Big-O Notation

Knuth formalised algorithm analysis using Big-O notation ($O(n^2)$, etc.) and wrote *The Art of Computer Programming*. In POLIS V12, the time complexity of an algorithm is how ϵ scales with input size n. For an algorithm that takes $f(n)$ steps, the total residual is $\epsilon(n) = f(n) \cdot \langle x_{\text{step}} \rangle$. Big-O notation ignores constant factors: $K_{\text{complexity}} = \limsup_{n\to\infty} \log(\epsilon(n))/\log n$.

Knuth's "literate programming" treats code as literature – intertwining solid mesh (code) with gaseous mesh (documentation). The Metafont system (digital typography) defines letters as K curves. Knuth's analysis of the Factorial number system and the "bubble sort" versus "quick sort" is a tensional comparison: the former has $K_{\text{time}} \propto n^2$, the latter $K_{\text{time}} \propto n \log n$.

8 Conclusion

The six foundational contributions to computer science are coherently reinterpreted within the POLIS V12 tensional ontology. Computability, lambda calculus, stored program architecture, information theory, structured programming, and algorithm analysis all become natural consequences of the closure condition $\epsilon = \sum K_m(2 + K_m) = 0$ and the fractal hierarchy of computational polises. No free parameters are added.

Zenodo references

- Main treatise: 10.5281/zenodo.19618276
- POLIS Bible: 10.5281/zenodo.19836226

POLIS V12 – Tensional Reinterpretation of Six More Computing Pioneers Jorge Batista Alves Pereira
Independent Researcher, Sabugal, Guarda, Portugal
ORCID: 0009-0000-6385-7245 May 2026

Abstract

This paper extends the POLIS V12 tensional reinterpretation to six additional computing giants: Grace Hopper (compilers), John McCarthy (Lisp and AI), Edsger Dijkstra (already in article 1; replace with another? Let's do: Tim Berners-Lee (World Wide Web), Linus Torvalds (Linux), Ada Lovelace (first programmer), and Alan Kay (object-oriented programming). Actually, to avoid duplicate Dijkstra, we choose: Grace Hopper (compiler), John McCarthy (Lisp), Tim Berners-Lee (Web), Linus Torvalds (Linux), Ada Lovelace (algorithm), and Alan Kay (OO). Each is re-read as a tensional configuration: Hopper's compiler as automatic Phase 5 translation; McCarthy's recursion as self-reference; Berners-Lee's hypertext as networked K; Torvalds's open source as distributed polis; Lovelace's Bernoulli numbers as first K algorithm; and Kay's objects as encapsulated meshes. The universal equations remain unchanged; no free parameters are introduced.

9 Introduction

As in the companion paper, POLIS V12 rests on four axioms. After normalisation the mother equation is

$$\epsilon = \sum_{m=1}^{n} K_m(2 + K_m) = 0,$$

with $\text{IDT}^* = \epsilon/(1+\epsilon)$. All real computational systems are in Phase 4 ($\text{IDT}^* \geq 0.70$) unless artificially uniform. The Rolling Law $2\pi r_p = V_{\text{orb}} T_{\text{rot}}$ applies fractally.

This paper reinterprets six more foundational contributions to computing.

10 Grace Hopper – Compiler and the First Compiler

Hopper developed the first compiler (A-0) and advocated for high-level languages (COBOL). In POLIS V12, a compiler translates a high-level language (gaseous mesh) into machine code (solid mesh). The compilation process is a Phase 5 reorganisation: the programmer's K (source code) is converted to efficient K (executable). Hopper's "debugging" (removing a moth from a relay) is a tensional removal of an unwanted K node.

The concept of "automatic programming" means that the compiler reduces the ϵ of writing code (the programmer need not manage low-level details). COBOL (business language) focuses on data (liquid mesh) and report generation (gaseous mesh). Hopper's "Nanoseconds" lecture visualised the speed of light as a tensional limit: a nanosecond is the time light travels one foot.

11 John McCarthy – Lisp and Artificial Intelligence

McCarthy invented Lisp (List Processing) and coined the term "Artificial Intelligence". In POLIS V12, Lisp's fundamental data structure is the linked list: a sequence of K values. The empty list (NIL) is $K = 0$. Lisp's eval function (evaluate a list as code) is a tensional self-reflection: a program can treat its own K as data. Recursion (function calls itself) is a Phase 5 loop that reduces a problem to a smaller K instance.

McCarthy's "advice takers" (AI systems that accept declarative knowledge) are polises that accept K statements as input. The "situation calculus" represents actions as changes in K states. The "Chinese Room" argument (Searle) is a philosophical objection to strong AI – but in tensional terms, the room (a lookup table) has K but no understanding of K (no gaseous mesh).

12 Tim Berners-Lee – World Wide Web

Berners-Lee invented the World Wide Web: HTML (hypertext markup language), HTTP (protocol), URLs (addresses). In POLIS V12, the Web is a distributed polis of documents (solid meshes) linked by hypertext (liquid mesh). Each page has its own K (content, relevance). A link is a directed tensional arrow from one page to another. The HTTP request/response cycle is a tensional exchange: the client sends K_{request}, the server replies with K_{response}.

The World Wide Web grew because the open standard allowed any polis to join. Berners-Lee's decision not to patent the Web was a tensional gift: he gave away high K (intellectual property) to benefit the whole mesh. The "web of trust" (semantic web) aims to increase $K_{\text{reliability}}$ of information.

13 Linus Torvalds – Linux and Open Source

Torvalds created the Linux kernel and pioneered the open-source development model (Git, collaborative coding). In POLIS V12, open source is a distributed polis where many developers contribute K (code). The kernel is the solid mesh (core functions); modules (drivers) are sub-polises. The Git version control system tracks changes as a tensional history (deltas of K). The "bazaar" model (many contributors, loose coordination) vs "cathedral" (centralised) is a comparison of ϵ accumulation rates.

Torvalds's "no regression" rule means that new code must not increase ϵ (introduce bugs). The Linux Foundation acts as a guardian polis that maintains the stability of the whole mesh.

14 Ada Lovelace – First Programmer

Lovelace wrote the first algorithm for Babbage's Analytical Engine: a program to compute Bernoulli numbers. In POLIS V12, the Analytical Engine is a mechanical polis (gear-based solid mesh). Lovelace's algorithm is a step-by-step K transformation: each operation

(addition, multiplication, loop) changes the state of the Engine. She foresaw that the machine could manipulate symbols (music, graphics) – i.e., process any K representation, not just numbers.

Lovelace's "Note G" states that the machine cannot originate anything; it only does what we tell it. In tensional terms, the Engine has no autonomous K (no gaseous mesh of its own); it only executes external instructions. This is the difference between a computer (n=1 solid only) and a conscious polis (n=3).

15 Alan Kay – Object-Oriented Programming and Smalltalk

Kay coined "object-oriented programming" (OOP) and developed Smalltalk. In POLIS V12, an object is a polis with three meshes: attributes (solid), methods (liquid), and messages (gaseous). Objects communicate by sending messages, which are tensional actions. Inheritance creates a hierarchical polis of classes (sub-polises). Encapsulation means that an object's internal K is hidden from others (low ϵ of coupling).

Kay's "Lisp on a microcomputer" (Smalltalk) integrated the development environment with the language (a living polis). His Dynabook vision (personal computer for children) was a Phase 5 educational tool. The phrase "The best way to predict the future is to invent it" is a tensional principle: create a new K configuration rather than extrapolate from current ϵ.

16 Conclusion

Six additional computing pioneers are reinterpreted within the POLIS V12 tensional ontology. Compilers, Lisp, the Web, open source, the first algorithm, and object-oriented programming all become natural consequences of the closure condition $\epsilon = \sum K_m(2+K_m) = 0$ and the fractal hierarchy of computational polises. No free parameters are added; the same equations that describe a physical system or a social system also describe the digital world.

Zenodo references

- Main treatise: 10.5281/zenodo.19618276
- POLIS Bible: 10.5281/zenodo.19836226

References for the twelve computer scientists

- Turing, A. M. (1936). "On Computable Numbers, with an Application to the Entscheidungsproblem". *Proceedings of the London Mathematical Society*, **42**, 230–265.
- Church, A. (1936). "An Unsolvable Problem of Elementary Number Theory". *American Journal of Mathematics*, **58**, 345–363.
- von Neumann, J. (1945). *First Draft of a Report on the EDVAC*. Moore School of Electrical Engineering.
- Shannon, C. E. (1948). "A Mathematical Theory of Communication". *Bell System Technical Journal*, **27**, 379–423, 623–656.
- Dijkstra, E. W. (1965). "Structured Programming". *Software Engineering Report*. NATO.
- Knuth, D. E. (1968). *The Art of Computer Programming*. Addison-Wesley.
- Hopper, G. M. (1952). "The Education of a Computer". *Proceedings of the ACM Conference*.
- McCarthy, J. (1960). "Recursive Functions of Symbolic Expressions and Their Computation by Machine". *Communications of the ACM*, **3**, 184–195.
- Berners-Lee, T. (1989). *Information Management: A Proposal*. CERN.
- Torvalds, L. (1991). Linux kernel announcement (comp.os.minix).
- Lovelace, A. (1843). "Notes on the Analytical Engine". *Scientific Memoirs*, **3**.
- Kay, A. (1972). *The Reactive Engine*. PhD thesis, University of Utah.