

KNO-022 · Computation as External Memory

KNO · Ways of Knowing

Juan Segurola

v0.3 · EN

All Rights Reserved

Article Abstract

A computer does more than calculate quickly. It can store intermediate states, apply explicit procedures repeatedly, preserve symbolic structures and make operations inspectable outside the limits of unaided human memory. That makes computation a form of external cognitive infrastructure. Yet outsourcing memory and procedure changes what people remember, what they notice and where errors can hide. A saved file can preserve a result while losing the environment that produced it; an automated pipeline can increase consistency while repeating the same mistake thousands of times. This article asks what computation contributes when it is used as an external memory for science and reasoning: persistence, repeatability, scale, shareable representations and executable procedures. It also asks what computation cannot supply by itself: meaningful variables, justified assumptions, trustworthy inputs or an interpretation of why a result matters.

In 60 Seconds

Humans routinely move cognitive work into the environment: we write lists, draw diagrams, set reminders and arrange objects so that the world carries part of the task. Cognitive-offloading research treats this as a general strategy rather than an anomaly.[1] Computers extend the same logic. They can retain exact symbolic states, rerun procedures and transform representations at scales that unaided memory cannot manage. External representations also change the structure of thinking, not merely its storage burden.[2] Search engines and networked information can shift memory toward knowing where information can be found.[3] In science, executable code can preserve a procedure more precisely than prose, especially when documentation and computation are kept together.[4] But computation is not neutral memory. File formats, software versions, interfaces and automation all shape what survives and what becomes easy to do. A computer can preserve an error perfectly. Its epistemic value comes from coupling external memory to provenance, interpretation and checks.

Why This Question Matters

Many scientific results now depend on chains of computation too long to reproduce mentally and too detailed to describe completely in prose. The practical question is no longer whether humans use external memory, but how reliable that external memory is. Scripts, notebooks, databases and version histories can preserve reasoning steps across time and across people. They can also detach users from the assumptions embedded in those steps. Understanding computation as external memory helps separate three things that are often conflated: storing information, executing a procedure and understanding what the procedure means.

1. Memory Can Be Moved into the Environment

The Question What changes when information needed for a task is stored outside the person performing it? External memory can reduce internal load, but it can also reorganize reasoning, create new dependencies and change what a user must know in order to retrieve, execute and interpret the stored information.

Cognitive offloading uses physical action or external storage to reduce internal cognitive demand. Risko and Gilbert review evidence that people restructure tasks by writing, arranging or storing information when relying on memory alone would be costly.[1] The gain is persistence and shareability, not immunity to error: an external record can be lost, mislabelled or stripped of context.

Claim C01. People use external actions and representations to reduce internal cognitive demand. [1,2]

Where That Leaves Us

External memory changes who can inspect information, how long it can persist and which errors become possible. Writing and storing reduce some cognitive burdens while creating dependence on labels, location and context. Computation extends this tradeoff by making external states executable. The next question is how representations themselves reshape the work of reasoning rather than merely holding facts.

2. External Representations Change the Cost of Thinking

Kirsh argues that external representations do more than save memory: they can change the cost structure of inference, create persistent referents and provide shareable objects of thought.[2] A diagram, table or symbolic layout can make relations simultaneously available that would be cumbersome to maintain internally. Computation adds the ability to transform one external representation into another by explicit rules.

Claim C02. External representations can alter the cost structure of reasoning, not only memory load. [2]

Where That Leaves Us

External representations reorganize operations, not just storage. A table, plot or symbolic structure changes what can be compared at once and which relations become easy to inspect. Computation adds rule-governed transformation between representations. That leads to a stronger form of external memory: a stored state can be paired with an executable procedure that changes it predictably.

3. A Program Stores a Procedure, Not Only a Result

A saved answer records an endpoint; a program can preserve a sequence of operations capable of regenerating that endpoint from specified inputs. Knuth's literate-programming proposal made the link between executable procedure and human explanation explicit.[4] Code can therefore externalize procedural memory with more operational detail than prose alone, although documentation is still needed to connect machine operations to scientific meaning.

Claim C03. Code can preserve an executable procedure in addition to a stored result. [4]

Claim C10. Literate programming links executable procedure with human-readable explanation. [4]

Where That Leaves Us

Code can externalize procedural memory: not only what answer was reached, but how a system can reach it again from inputs. That increases precision and repeatability for complex workflows, provided the code remains interpretable. Execution alone does not reveal why a variable represents what it claims to represent. The next question is what happens when remembered procedure depends on a changing software environment.

4. Exact Files Can Stop Being Executable

A script may remain byte-for-byte unchanged while libraries alter defaults, operating systems remove interfaces, services disappear and formats age. The text of a program therefore does not fully specify its future behaviour unless relevant dependencies and environment are also known. Containers and environment manifests can reduce ambiguity, but long-lived computational memory still requires migration and testing, not just copying.

Claim C04 - Declared synthesis. Executable preservation depends on software and environment context as well as file identity. **Declared synthesis:** [4] supports preserving computational environment information; the claim that executable preservation therefore depends on that context in addition to file identity is the synthesis.

Where That Leaves Us

Digital persistence has layers. Byte identity protects against one kind of change, while executability depends on software and hardware context, and scientific meaning depends on metadata and interpretation. A preserved script can therefore be intact yet unusable. This exposes the next limit of automation: when a procedure does run reliably, it may repeat a mistaken assumption with extraordinary consistency.

5. Automation Multiplies Consistency and Error

Automation applies the same explicit procedure repeatedly without fatigue, making it possible to process volumes no person could handle manually. That consistency is conditional on the procedure being appropriate. A unit-conversion error inside a loop is not diluted by repetition; it is replicated. Computational scale therefore increases the value of tests, validation data, invariants and independent checks because systematic error can be amplified with perfect regularity.

Claim C05 - Declared synthesis. Automation can reduce variability of execution while amplifying systematic procedural error. **Declared synthesis:** no source is assigned as direct support; both the reduction of run-to-run variability and the possibility of consistently repeating the same procedural error are methodological synthesis here.

Where That Leaves Us

Automation changes the distribution of error. It can suppress inconsistency caused by repeated manual execution, yet magnify any mistake embedded in the procedure. The more widely a pipeline is reused, the more consequential its assumptions become. That makes computational memory inseparable from verification: stored procedures need tests that can fail. The next question is what users remember once machines reliably remember for them.

6. Knowing Where Can Replace Remembering What

Sparrow, Liu and Wegner reported experiments in which expectations about later computer access affected what participants remembered, including memory for where information could be found. [3] The result does not show that the internet simply destroys memory. It illustrates a shift in allocation: when external retrieval is expected, internal memory can prioritize access paths over content, creating dependence on the stability of the external system.

Claim C06. Expectations of later computer access can change what people remember about information and its location. [3]

Claim C09 - Declared synthesis. External computational memory creates dependency on identifiers, interfaces and institutions. **Declared synthesis:** [1,3] support distributed cognition and persistent computational resources; describing their use as a dependency on identifiers, interfaces and institutions is the synthesis.

Where That Leaves Us

External access can change what internal memory is used for. People may retain routes to information rather than the information itself when reliable retrieval is expected. This is efficient while the external system persists, but it creates dependency on identifiers, interfaces and institutions. The next question is how those dependencies are coordinated when computation is distributed across people, artifacts and procedures.

7. Computation Can Be Distributed Across People and Artifacts

Hutchins' study of ship navigation treats cognitive work as distributed across people, instruments, charts, procedures and communication rather than located wholly in one navigator.[5] Modern scientific workflows have similar architecture: sensors produce records, scripts transform them, databases preserve state and specialists validate stages. Reliability can emerge from coordinated division of labour, while responsibility and meaning can also become diffuse at handoffs.

Claim C07. Distributed cognitive systems can span people, artifacts and procedures. [5]

Claim C08. No single participant need represent every transformation in a distributed cognitive workflow. [5]

Where That Leaves Us

Computational knowledge can be distributed across humans and artifacts. No individual needs to remember every intermediate state if the system preserves them and coordinates transformations reliably. This expands collective capability while making provenance more important: responsibility and meaning can be lost at handoffs. KNO-023 will examine the human-group side of this architecture; first, KNO-022 must set its boundary with understanding itself.

8. External Memory Does Not Supply Meaning

Licklider imagined a partnership in which machines perform routinizable work while humans formulate goals, hypotheses, criteria and evaluations.[6] The language is dated, but the methodological warning remains useful. A computer can preserve and execute formal operations without deciding whether the inputs answer the right question. Meaning still enters through modelling choices, measurement conventions and interpretation.

Claim C11 - Declared synthesis. Computation can preserve operations without establishing the scientific meaning of their inputs. **Declared synthesis:** [4,6] support preserving workflows and software-dependent operations; the contrast that preserved operations do not by themselves establish the scientific meaning of inputs is the synthesis.

Claim C12 - Declared synthesis. Computation is epistemically strongest when stored states and procedures retain provenance and interpretation. **Declared synthesis:** [2,4,5] support provenance, reproducible procedures and retained computational context; ranking such systems as epistemically strongest when those elements remain interpretable is the synthesis.

Where That Leaves Us

Computation is external memory plus executable transformation, not an autonomous source of meaning. It can preserve states, procedures and coordination across time, but the scientific significance of those objects still depends on justified representations and questions. This sets the series boundary: KNO-022 owns computational offloading and executable memory; KNO-023 turns to how human groups distribute and combine knowledge.

What We Know, and How

- People use external representations and actions to reduce internal cognitive demand.[1,2]
- External representations can change the structure and cost of reasoning, not only storage requirements.[2]
- Expectations of external access can affect what people remember about information and its location.[3]
- Executable code can preserve procedures with greater operational specificity than prose alone.[4]

What We Still Don't Know

- How different forms of cognitive offloading reshape long-term learning rather than immediate task performance.
- Which computational dependencies are necessary to preserve for century-scale executability.
- How much automation changes users' ability to detect conceptual rather than implementation errors.
- Where responsibility should sit in highly distributed human-computer workflows.

What Could Change Our Minds

- Evidence that external representations systematically impair reasoning even when task demands and retrieval costs are controlled.
- Long-term studies showing that current reproducibility tools fail to preserve executable scientific workflows despite complete metadata.
- Demonstrations that automation errors are reliably caught without independent validation, reducing the need for redundant checks.
- New cognitive models that explain offloading effects without any functional role for external memory or representation.

If You Remember Only One Thing

Computation becomes epistemically powerful when it preserves not only results, but the states, procedures and provenance needed to understand how those results were produced.

Related Articles

- KNO-008 · Instruments That Extend the Senses
- KNO-019 · Data, Tables and Records
- KNO-021 · Open Science and Reproducibility

References

[1] Evan F. Risko; Sam J. Gilbert. Cognitive Offloading. *Trends in Cognitive Sciences* 20(9), 676–688 (2016). DOI 10.1016/j.tics.2016.07.002. <https://doi.org/10.1016/j.tics.2016.07.002> [2] David Kirsh. Thinking with external representations. *AI & Society* 25, 441–454 (2010). DOI 10.1007/s00146-010-0272-8. <https://doi.org/10.1007/s00146-010-0272-8> [3] Betsy Sparrow; Jenny Liu; Daniel M. Wegner. Google effects on memory: cognitive consequences of having information at our fingertips. *Science* 333(6043), 776–778 (2011). DOI 10.1126/science.1207745. <https://doi.org/10.1126/science.1207745> [4] D. E. Knuth. Literate Programming. *The Computer Journal* 27(2), 97–111 (1984). DOI 10.1093/comjnl/27.2.97. <https://doi.org/10.1093/comjnl/27.2.97> [5] Edwin Hutchins. *Cognition in the Wild*. The MIT Press (1995). DOI 10.7551/mitpress/1881.001.0001. <https://doi.org/10.7551/mitpress/1881.001.0001> [6] J. C. R. Licklider. Man-Computer Symbiosis. *IRE Transactions on Human Factors in Electronics* HFE-1, 4–11 (1960).