Objectifying Reason and Computation: Computable-Once Reasoning Objects for Efficient AI

Eric Robert Lawson

November 9, 2025

Abstract

This article explores a novel paradigm in which reasoning itself is treated as a first-class, composable object. By objectifying reasoning, we introduce the concept of *computable-once objects* — reasoning artifacts that, once derived, can be stored, retrieved, and combined without redundant recomputation. This approach enables emergent reasoning, lazy evaluation, and efficient management of combinatorial reasoning spaces, with direct applications to AI, symbolic computation, and complex decision-making tasks such as chess. We discuss the implications of this framework for computation, domain-specific languages, and the future of automated reasoning systems.

Contents

1 Introduction

Traditional computation treats reasoning as a process that is often recomputed each time it is needed. In complex domains, such as strategic gameplay or symbolic mathematics, this results in significant redundant computation and wasted resources.

In contrast, treating reasoning as a *first-class object* allows computation to be stored, reused, and composed, avoiding redundant recalculation. For instance, in combinatorial domains like chess, this enables lazy evaluation, compositional reasoning, and efficient exploration of derivative states.

2 Reasoning as Computable-Once Objects

A *computable-once object* is a reasoning primitive that, once calculated, can be stored and reused across multiple contexts without recalculation. Formally, let R denote a reasoning object and C a context:

$$R[C] \text{ can be retrieved from storage after initial computation.}$$

This separates **computation** from **retrieval and assimilation**, enabling efficient reuse in complex reasoning tasks.

2.1 Example: Chess Reasoning Space

Consider a reasoning object representing a chess game state S. Traditional evaluation recomputes all derivative states each time. Using computable-once objects:

- S is computed once and reused in meta-analyses (strategy comparisons, counterfactual exploration).
- Multiple S_i objects can be combined into higher-order reasoning constructs without redundant computation.
- Lazy evaluation allows exploration of derivative states only as needed, mitigating combinatorial explosion.

2.2 Implications for Combinatorial Computation

Storing and reusing reasoning objects reduces combinatorial blowup. For instance:

- Symbolic DAGs representing combinatorial expansions can be reused as nodes.
- Bell-polynomial derivatives or layer/POT generator operations need not be recomputed for identical substructures.

3 Bridging Computation and Reasoning

This paradigm redefines computation:

1. **Reuse of computation:** Precomputed objects are accessed as needed, avoiding redundant calculation.
2. **Context-aware retrieval:** Objects adapt to new reasoning contexts without recomputation.
3. **Meta-reasoning:** Higher-order structures leverage precomputed objects to explore emergent reasoning.

The above principles can be directly encoded in a DSL, enabling AI systems to operationalize the compute-once paradigm efficiently.

3.1 DSL Implications

A domain-specific language (DSL) built on this paradigm can directly implement these principles:

- Primitives for object creation, retrieval, and composition
- Lazy evaluation of reasoning DAGs to defer computation
- Meta-RDU support for emergent reasoning across layers

This allows AI systems to *autonomously manipulate and integrate reasoning objects*, bridging the gap between symbolic and automated reasoning.

With these primitives in place, reasoning objects can be fully instantiated, manipulated, and assimilated within a structured substrate.

3.2 From Process to Object: Operationalizing Reasoning

Traditionally, reasoning has been seen as an ongoing process — a sequence of inferential steps executed transiently within a computational or cognitive system. In contrast, the *compute-once object* reframes reasoning as something that can be stored, reconfigured, and acted upon as an explicit entity.

Operationalizing reasoning as an object introduces a new computational perspective. The act of reasoning no longer ends with a single result. Instead, it produces a manipulable artifact — the reasoning object. This artifact can undergo further computational operations, such as optimization, recontextualization, or assimilation into broader reasoning spaces.

When reasoning spaces are compatible, precomputed reasoning objects can be combined to form composite or emergent structures. This transforms reasoning from a purely procedural process into a composable substrate of interrelated artifacts. What was once a theoretical or implicit branching of possibilities — a transient exploration of reasoning paths — becomes a persistent and reusable structure.

The compute-once paradigm does more than eliminate redundant recomputation; it also broadens the scope of computation. Instead of computing narrowly defined outputs, systems compute relative to an evolving reasoning space. This space is continuously constructed, objectified, and optimized. Computation thus becomes an act of reasoning-space construction, enabling a form of meta-efficiency that goes beyond traditional algorithmic repetition.

3.3 From Conceptual to Operational: Assimilating Compute-Once Objects

Assimilation is the process by which a reasoning object is integrated into the broader reasoning substrate, abstracting its operational paths while preserving strategic and inferential content. This allows objects to participate in higher-order operations, meta-reasoning, and cross-domain integration without recomputation.

While we already conceptually understand compute-once objects in domains like chess, their current form is largely **observational and static**. Recorded games, typically notated in algebraic notation, serve as proof-like artifacts: sequences of moves that humans can interpret and analyze. In this sense, they are akin to tapes or logs of reasoning, capturing complete games without providing operational leverage for automated reasoning agents.

However, these conceptual compute-once objects are **not yet part of a reasoning substrate** that allows for machine manipulation, traversal, or composition. They exist as records, not as **active reasoning units** capable of integration into a larger, evolving reasoning space.

By introducing a universal reasoning substrate and an accompanying DSL, we can operationalize these objects. Specifically:

- Each recorded game becomes a *computable-once reasoning object*, fully instantiated within a reasoning space.
- Precomputed moves, position evaluations, and strategic patterns are stored as structured DAG nodes, forming a semi-structured reasoning network.
- Assimilation of multiple games enables the construction of composite reasoning objects, supporting higher-order strategic analysis without recomputation.
- Lazy DAG evaluation allows the system to defer computation until necessary, mitigating combinatorial explosion.

Moreover, this framework enables **self-referential reasoning**. Gaps in the reasoning space can be filled using predictive models or other computational engines, producing new reasoning objects that are then assimilated back into the substrate. As a result, the reasoning space becomes **both operationally complete and extensible**, allowing AI systems to learn, generate, and integrate new knowledge dynamically.

In practice, this approach transforms a repository of static chess games into a **fully operational reasoning substrate**. The process can be summarized as follows:

1. Recorded games are imported as structured reasoning objects.
2. Objects are assimilated into a DAG representing the chess reasoning space.
3. Machine learning models can be trained on this structured space — via supervised learning, reinforcement learning, or self-play — producing new reasoning objects that can be assimilated back into the substrate.
4. New objects are assimilated, expanding the substrate and enabling meta-reasoning across games and strategies.

Importantly, assimilation removes the need for direct traversal of individual game paths. Each reasoning object abstracts and encodes the strategic and evaluative content of a game, embedding it into the potential-branch reasoning space for direct, composable use.

This paradigm illustrates the **pragmatic utility of the compute-once concept**: rather than recomputing reasoning paths from scratch, AI agents operate on **already-computed, manipulable reasoning objects**, enabling efficient, composable, and scalable reasoning across complex domains.

3.4 Integrating Machine Learning with Structured Reasoning Spaces

Traditional machine learning often operates without explicit awareness of the broader reasoning context. Models are trained on data points in isolation, without considering how each observation contributes to a structured space of reasoning. For example, a data annotator labels individual instances without defining the relationships or operational dependencies between them.

In contrast, within a structured reasoning substrate, context emerges naturally from the reasoning space itself and the reward function that defines its operations. Consider the chess reasoning space: the reward function is defined to achieve a win condition, pruning loss branches and draw

branches strategically. At any position, the reward can be locally evaluated to determine which paths in the reasoning space lead to better outcomes, without requiring enumeration of all possible branches — thereby mitigating combinatorial explosion.

This structural approach offers explicit advantages over unstructured methods:

- Learning occurs relative to a fully instantiated reasoning space, rather than moment-to-moment observations.
- Models can generate new reasoning objects, which are assimilated back into the substrate and examined with respect to the broader reasoning space.
- The self-referential nature of reasoning objects — the meta-RDU instantiation — allows derivative spaces to be described, explored, and evaluated recursively.
- Reward functions can be measured post-hoc against these reasoning objects, enabling feedback to inform learning in a context-aware manner.

By integrating machine learning in this way, predictive models shift from solving narrowly defined tasks to operating over an **objectified reasoning space**. The compute-once paradigm ensures that learning, generation, and evaluation of reasoning objects are efficient, composable, and inherently explainable. This approach opens entirely new avenues for hybrid AI systems that combine symbolic reasoning with data-driven learning, fully exploiting the structural and meta-referential capabilities of the reasoning substrate.

Machine learning models interact with the DSL by producing reasoning objects that are immediately assimilated into the substrate, demonstrating the operational utility of the DSL's primitives in a feedback loop. They can also act as agents that construct reasoning layers; for example, they can create reasoning objects that represent the next move in a game, enabling dynamic expansion of the reasoning space.

3.5 Network-of-Networks: Combinatorial Primitives and Cross-Domain Reasoning

The objects produced and assimilated by ML models naturally extend into a network-of-networks, enabling structured cross-domain reasoning. Here, computation extends beyond a single reasoning space. It enables cross-referencing and operations across multiple domains. For example, ML models can guide structured reasoning navigation within specific contexts, producing objects that inform broader substrate operations.

The reasoning substrate developed through computable-once objects naturally forms a **network-of-networks**. In this structure, each network corresponds to a domain or sub-domain — for example, medicine, chemistry, or domain-specific procedures — and the nodes within these networks are composed of fully instantiated reasoning objects.

In our prototypes, combinatorial primitives were used to define multinomial structures that capture the relationships and potential interactions between reasoning objects. These primitives allow the substrate to represent and manipulate *all feasible combinations of reasoning elements* within and across domains without redundantly recomputing shared substructures. By operationalizing these primitives within the DSL, we can instantiate reasoning objects as nodes in a structured, hierarchical network, where edges encode operational, strategic, or inferential dependencies.

This network-of-networks approach shifts the computational paradigm from narrowly scoped, isolated calculations to the manipulation of structured, objectified reasoning across multiple domains. Generalizable primitives — such as structural axioms, heuristics, or strategy motifs — can

be assimilated into multiple networks, enabling meta-reasoning, efficient traversal, and compositional operations without recomputation.

The resulting hierarchical and composable substrate ensures that reasoning objects are both **grounded** in domain-specific knowledge and **extensible** to novel contexts. AI systems can integrate, traverse, and reason across these domains efficiently, while maintaining transparency and auditability. Importantly, this approach establishes a concrete foundation for the DSL, demonstrating how combinatorial primitives, compute-once reasoning objects, and network-of-networks organization together operationalize a multi-domain reasoning substrate that transcends traditional, narrowly focused algorithmic processes.

4 Conclusion

By treating reasoning as a first-class object and leveraging computable-once primitives, we open the door to a new paradigm of efficient, composable, and emergent computation. This framework supports **direct assimilation of reasoning artifacts into a structured substrate**, enabling hybrid AI systems to integrate symbolic and data-driven learning. It provides a natural foundation for the next generation of reasoning-oriented DSLs, where reasoning objects are not merely evaluated but **operationally manipulated, composed, and reused across domains**.

Crucially, the Python prototypes themselves — including the multinomial combinatorial primitives underlying the POT generator function — can be instantiated as **compute-once objects in the mathematical domain**. These primitives provide cross-domain applicability: the same combinatorial structures that organize reasoning in AI can be formalized and manipulated within pure mathematics, creating reusable, composable, and operational artifacts. Rarely, if ever, has the multinomial process been defined in terms of such combinatorial primitives and applied across such a broad contextual landscape. This dual utility demonstrates that reasoning structures are not limited to a single domain but instead form a universal substrate, capable of both symbolic computation and applied operationalization.

While chess has served as a useful grounding example, the principles outlined here extend far beyond any single domain. The framework demonstrates how innovations in one area can naturally translate to others, emerging from the structure of reasoning itself. What may initially appear as a niche innovation can, in fact, have broad applicability, highlighting the generality, cross-domain reach, and mathematical grounding of the compute-once reasoning paradigm.