

The Operator Intelligence Benchmark: Domain Construction, Constraint Contraction, and Certified Solution Emergence in Neurosymbolic AI

Thanh Trung Phan

Independent Researcher, Vietnam

pttrung@operatorology.com | ORCID: 0009-0000-7520-6781

Abstract

Current reasoning benchmarks typically score whether a system reaches the correct answer, while giving much less attention to how the candidate solution space is represented, contracted, and certified. We introduce OIBench 1.0, a benchmark for a narrow and operational notion of Operator Intelligence: the ability to construct or contract a candidate domain through reusable operators while preserving admissible solutions and producing a checkable certificate. OIBench contains 1,000 deterministic tasks across permutation, Boolean, modular-arithmetic, and graph-coloring families, with explicit train/development/test splits, exact symbolic semantics, domain sizes from 120 to 6,561 candidates, and 6-29 admissible constraints per task. We define three evaluation tracks: operator selection, domain construction, and certified operator construction. The released reference implementation fully instantiates the first track and provides exact verification interfaces for the latter two. Metrics include solution survival, certificate validity, candidate-evaluation cost, operator steps, compression bits, and domain-trajectory statistics. On 200 test tasks, all reference methods preserve the unique certified solution. Median candidate-evaluation cost relative to exhaustive enumeration is 0.133 for static ordering, 0.127 for random ordering, 0.090 for a compact neural operator selector, 0.079 for 32-sample greedy selection, and 0.078 for exact greedy selection. The neural selector significantly reduces cost relative to random ordering but generalizes unevenly across reasoning families, exposing a useful separation between learned operator ranking and symbolic verification. OIBench is intended as a reproducible benchmark for neurosymbolic systems that aim to move beyond answer generation toward explicit domain construction, constraint contraction, and verifiable solution emergence.

Keywords: neurosymbolic AI; benchmark; operator intelligence; constraint reasoning; domain contraction; symbolic verification; certified reasoning

1. Introduction

Neurosymbolic artificial intelligence seeks to combine the learning and adaptive strengths of neural models with the explicit structure, reasoning, and verifiability of symbolic systems [1-6]. This integration has produced differentiable logic, neural predicates in probabilistic programs, neural theorem proving, semantic losses, and systems that jointly learn and reason. Yet evaluation practice still often collapses the behavior of a reasoner to a final-answer metric: accuracy, exact match, proof success, or task reward. Those measures are indispensable, but they do not directly reveal whether a system has represented the problem in a useful domain, selected informative reasoning operations, preserved valid hypotheses, or produced a certificate that explains why the final answer survived.

This paper studies a narrower question. Suppose a problem admits a finite candidate domain D_0 , a set of admissibility constraints C , and an operator library O whose elements transform or restrict the domain. A reasoner can search candidates directly, or it can contract the domain by applying operators whose effects are symbolically checkable. If repeated contraction yields a singleton $DT=\{x^*\}$ while x^* remains valid at every step, the final answer is not merely guessed: it emerges as the only surviving admissible configuration. We use the term Operator Intelligence for the benchmarkable capability to organize that process. The term is intentionally operational rather than anthropomorphic; OIBench does not claim a new form of general intelligence.

The distinction matters for neurosymbolic AI. A neural component can rank candidate operators, propose relations, or predict which restriction is likely to be informative. A symbolic component can then execute the proposal, reject invalid operations, preserve invariants, and certify the resulting contraction. This architecture separates heuristic guidance from logical authority. The neural subsystem may be uncertain; the symbolic verifier is responsible for correctness.

OIBench 1.0 turns this idea into a benchmark rather than a manifesto. It contributes: (i) four families of exact finite-domain reasoning tasks with 1,000 reproducibly generated instances; (ii) a common operator schema and reference symbolic semantics; (iii) metrics for domain contraction, solution survival, certificate validity, and computational cost; (iv) three benchmark tracks spanning operator selection through domain construction and certified operator invention; and (v) reference baselines ranging from exhaustive enumeration to a compact neural operator selector and exact greedy upper bound. The benchmark is designed so that a system can be wrong in informative ways: it may solve the task but contract inefficiently, contract quickly but eliminate the solution, rank operators well in one family but transfer poorly, or produce an answer without a valid certificate.

Our experimental results demonstrate that the benchmark separates these behaviors. A small neural selector trained to predict operator gain uses substantially fewer candidate evaluations than random or static ordering while retaining symbolic correctness. At the same time, a cheap sample-based symbolic heuristic remains stronger overall, and the neural selector underperforms random ordering on the coloring family. This negative transfer is useful: the benchmark does not reward the mere presence of a neural component, but tests whether learned guidance actually improves certified reasoning.

2. Operator Intelligence as a Benchmarkable Capability

Let a reasoning instance be represented by a tuple $P=(X,C,I,R,O)$, where X is a candidate state space, C is a set of admissibility constraints, I denotes invariants that must be preserved, R denotes explicit relations among states or variables, and O is a set of operators. In OIBench 1.0, each supplied symbolic constraint induces a filtering operator on a current domain D subseteq X .

$$D_{t+1} = \Omega_t(D_t) = \{x \text{ in } D_t : c_t(x) = \text{true}\}.$$

A valid contraction operator must not eliminate the ground-truth solution. More generally, for tasks with multiple admissible solutions it must preserve the target solution set. The strict single-solution release used here makes this condition directly auditable.

Solution survival: x^* in D_t for every t .

The ideal endpoint is a singleton domain. In that case the reasoning trajectory itself acts as a certificate: every eliminated candidate violates at least one verified constraint, while the survivor satisfies the full constraint set.

$$D_0 \rightarrow D_1 \rightarrow \dots \rightarrow D_T = \{x^*\}.$$

This formulation distinguishes three capabilities that are often conflated. Search chooses among candidate answers inside a fixed representation. Operator selection chooses which admissible restriction to apply next. Domain construction chooses the state representation and the admissible candidate set itself. OIBench exposes these as separate tracks so that progress in one cannot be silently credited to another.

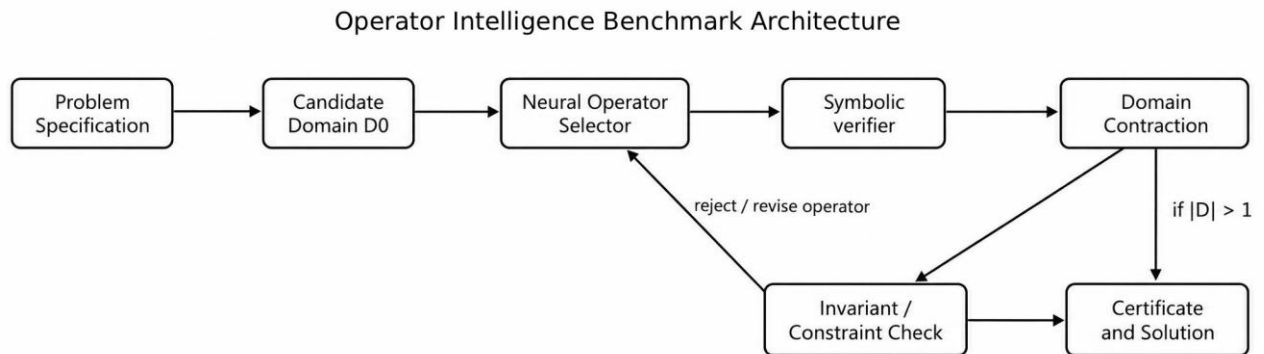


Figure 1. Benchmark architecture. A neural module may rank candidate operators, but every accepted contraction is executed and certified by exact symbolic semantics.

3. Benchmark Design

3.1 Task families

OIBench 1.0 contains four synthetic but structurally distinct finite-domain families. Synthetic generation is deliberate: exact extensional ground truth is needed to verify solution survival, operator semantics, and certificates without ambiguity. Each family uses different variable types and constraint vocabularies, reducing the risk that a policy can succeed by memorizing one operator grammar.

Family	State space	Typical domain	Operator examples	Reasoning property
Permutation	Permutation of n symbols	120-5,040	equality, order, adjacency, distance, parity	bijection + relational constraints
Boolean	n -bit vectors	512-4,096	XOR, implication, clauses, exact count, parity	logical restriction + cardinality
Modular	3 integer coordinates	512-2,197	linear congruence, sum, product residue, order	arithmetic consistency
Coloring	3-color assignments	729-6,561	edge inequality, fixed color, color count, same color	graph-relational consistency

Table 1. OIBench task families and the reasoning structure each family emphasizes.

3.2 Dataset generation and splits

The benchmark generator uses release seed 20260913. For each family, 250 tasks are produced, approximately balanced across easy, medium, and hard difficulty. A hidden solution is sampled first. Candidate constraints are then generated only when they are true of that solution. Constraints are accumulated until the complete set isolates the solution uniquely, after which additional valid redundant constraints are added so that operator ordering remains nontrivial. The complete set is shuffled before serialization. Every instance is re-verified by exhaustive enumeration before release.

The resulting 1,000 tasks are stratified by family into 600 training, 200 development, and 200 test instances. Across all tasks, explicit reference-domain size ranges from 120 to 6,561 candidates (median 1,331), while the number of admissible constraints ranges from 6 to 29 (median 15). The structured JSONL files include the family specification, operator pool, unique solution, and a SHA-256 solution hash. A hidden-solution copy can be derived for blind evaluation or server-side scoring.

3.3 Evaluation tracks

Track A - Operator Selection. The solver receives the task specification and the admissible operator pool. It must order or adaptively select operators so as to minimize reasoning cost while preserving the solution and ending with a valid certificate. All released baselines operate in this track.

Track B - Domain Construction. The solver receives the structured problem specification but is free to construct an explicit or implicit candidate domain. The reference enumerator defines extensional semantics, enabling evaluation of whether the constructed domain preserves the true solution and whether its representation reduces subsequent reasoning cost. OIBench provides interfaces and reference semantics for this track, but does not claim that the supplied baseline autonomously invents representations.

Track C - Certified Operator Construction. The solver may derive additional restrictions not included in the supplied pool. Each derived operator must be accompanied by a certificate or executable predicate that the benchmark verifier can test against the reference semantics. This track is intended for future theorem-guided, program-synthesis, and LLM-assisted systems.

4. Metrics

OIBench deliberately separates correctness from efficiency. Final-answer accuracy alone cannot distinguish a certified contraction policy from a lucky guess.

Certificate validity = 1 iff $D_T = \{x^*\}$ and every applied operator is admissible.

Compression bits = $\log_2(|D_0| / \max(|D_T|, 1))$.

Cost ratio = candidate evaluations of method / candidate evaluations of exhaustive certification.

Candidate-evaluation cost counts the number of state-predicate evaluations performed by the reference explicit executor. This machine-independent primitive is preferable to wall-clock time for the core benchmark, while runtimes may be reported as secondary implementation measures. Operator steps count accepted restrictions until the domain becomes a singleton. The release also records normalized log-domain trajectories, enabling analysis of whether a method removes uncertainty early or late in a reasoning episode.

Metric	Desired direction	Interpretation
Certificate validity	1.0	Correct symbolic endpoint and admissible trajectory
Solution survival	1.0	True solution is never pruned
Candidate-evaluation cost	lower	Primitive symbolic work
Cost ratio vs enumeration	lower	Search reduction relative to exhaustive certification
Operator steps	lower	Number of accepted contractions
Compression bits	higher	Amount of state-space reduction

Table 2. Primary OIBench metrics.

5. Reference Baselines

We release five contraction policies plus exhaustive enumeration. Enumeration checks the full candidate domain against the complete constraint set and serves as a transparent certification baseline. Static ordering applies constraints in serialized order. Random ordering shuffles the supplied operator pool independently per task. Probe-32 estimates each remaining operator's selectivity on a random sample of at most 32 surviving candidates and chooses the most contractive nonempty operator. Exact greedy evaluates every remaining operator on the full current domain and therefore provides an expensive upper-reference for myopic contraction.

The neural reference baseline is intentionally compact. For a task state, each candidate operator is encoded by family identity, operator type, arity, normalized parameters, current and initial log-domain size, step fraction, and number of remaining operators. A two-hidden-layer multilayer perceptron predicts immediate information gain in bits. The model is trained on 10,208 operator-state examples generated from 180 training tasks, leaving most of the official training split unused so that the reference does not become a hyperparameter-heavy competition. On a development sample it achieves MAE 0.404 gain bits and $R^2=0.717$. At inference, the highest-ranked proposed operator is passed to the exact symbolic verifier; any proposal that would produce an empty domain is rejected.

The neural baseline is thus genuinely neurosymbolic in a minimal sense: selection is learned, but state transition and correctness remain symbolic. It is not intended as a state-of-the-art neural architecture; rather, it demonstrates how OIBench can separate learned guidance from exact logical execution.

6. Results

All six reference methods achieve 100% certificate validity on the 200-task test split. The principal difference is the amount of symbolic work required to reach a singleton domain.

Method	Success	Median steps	Median eval. cost	Median cost ratio	Mean cost ratio
Enumeration	1.000	15.0	29645	1.000	1.000
Static	1.000	10.0	3023	0.133	0.138
Random	1.000	10.0	2845	0.127	0.136
Neural selector	1.000	8.0	2334	0.090	0.106
Probe-32	1.000	5.0	2341	0.079	0.089
Exact greedy	1.000	5.0	2250	0.078	0.086

Table 3. Reference results on 200 held-out test tasks. Lower cost ratios are better; enumeration is normalized to 1.0.

Static and random ordering already reduce explicit work because they stop when the domain becomes a singleton, but learned or selectivity-aware ordering contracts substantially faster. The neural selector uses a median 9.0% of exhaustive candidate-evaluation cost, compared with 12.7% for random ordering. Probe-32 and exact greedy reach 7.9% and 7.8%, respectively. A paired one-sided Wilcoxon signed-rank test confirms lower cost for the neural selector than random ordering ($p=5.15e-14$; median neural/random task-cost ratio 0.736). Probe-32 and exact greedy also improve strongly over random ordering ($p=6.96e-34$ and $p=7.18e-35$). Static ordering is not significantly better than random ordering ($p=0.812$).

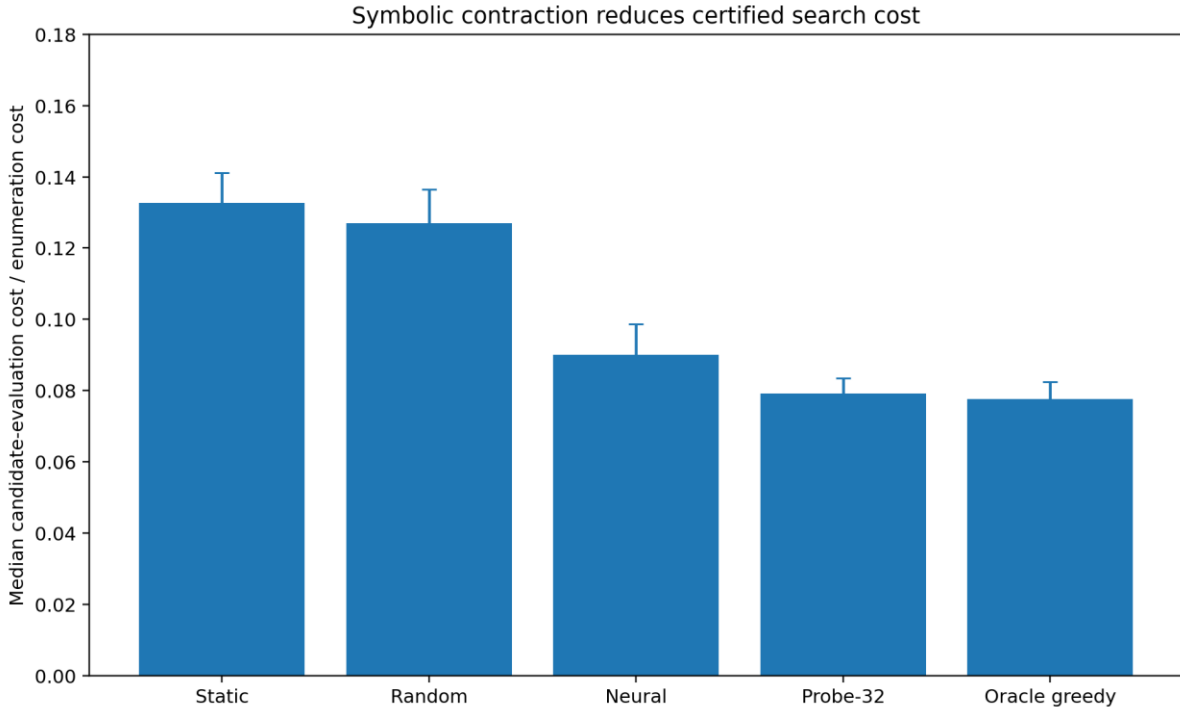


Figure 2. Median candidate-evaluation cost relative to exhaustive certification. Error bars are bootstrap 95% intervals over test tasks.

The benchmark also reveals a limitation that would be invisible under final-answer accuracy: the neural policy transfers unevenly across reasoning families. It improves on random ordering for permutation, Boolean, and modular tasks, but is worse on coloring (median cost ratio 0.138 for neural versus 0.123 for random). Probe-32 and exact greedy remain strong across all four families. The result suggests that the small neural baseline learns family-specific operator preferences rather than a fully transferable notion of symbolic selectivity.

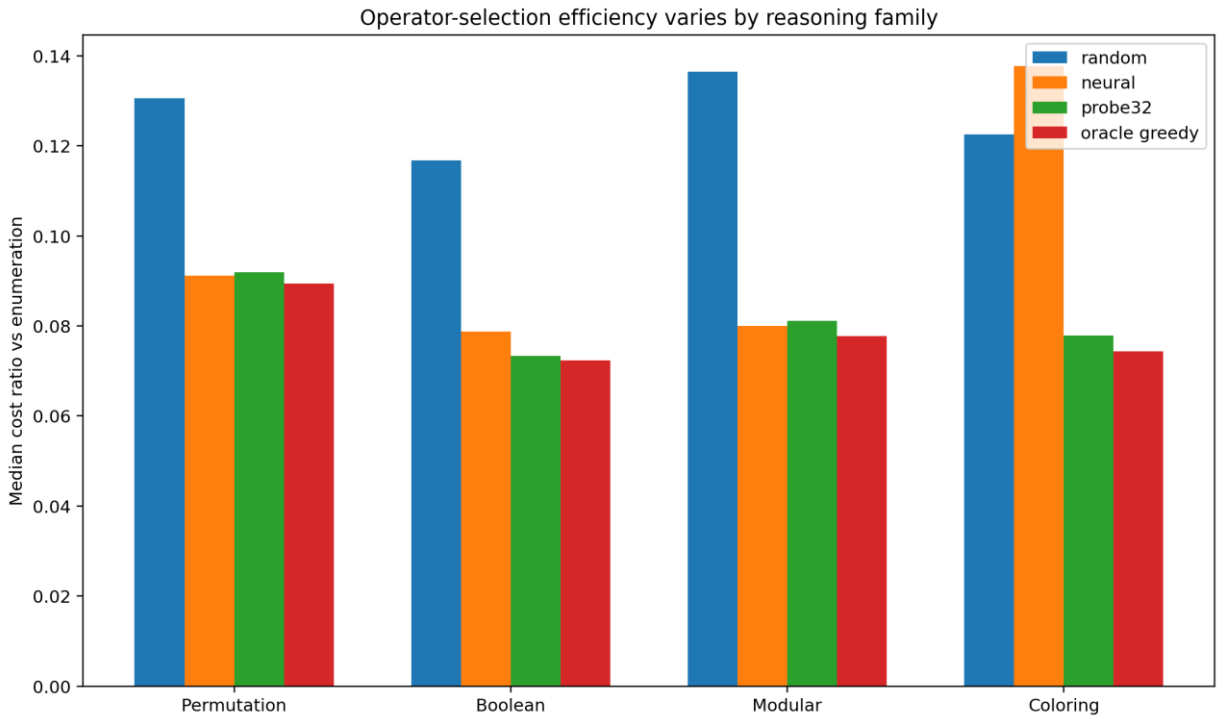


Figure 3. Median cost ratio by reasoning family. The simple neural policy generalizes unevenly, particularly on graph coloring.

Domain-trajectory curves provide a complementary view. Greedy and sampled-greedy policies remove large fractions of the candidate domain early, while static and random policies retain more uncertainty for longer. The neural selector falls between these regimes, consistent with its lower total cost but imperfect cross-family ranking.

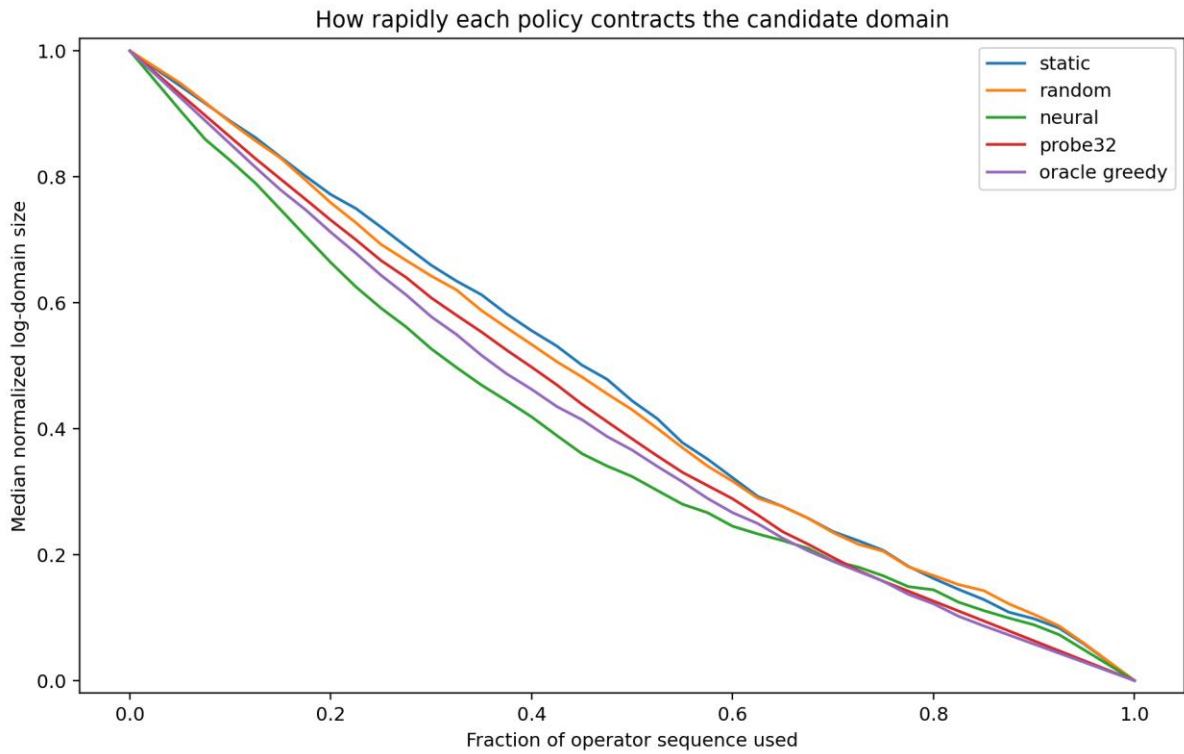


Figure 4. Median normalized log-domain size over normalized operator-sequence progress. Lower curves indicate earlier uncertainty reduction.

7. What OIBench Measures - and What It Does Not

OIBench is designed around a specific hypothesis: an important component of reasoning quality is not only whether a system reaches the right answer, but whether it constructs or contracts a representation in which invalid alternatives are eliminated efficiently and verifiably. In the strict finite setting, solution emergence has a precise meaning: after a sequence of certified contractions, the correct solution is the only admissible survivor. No metaphysical claim is intended.

This perspective complements, rather than replaces, existing neurosymbolic evaluation. Semantic-loss methods measure compatibility between neural outputs and symbolic knowledge [2]. Neural theorem provers learn continuous representations for symbolic inference [4]. DeepProbLog integrates neural predicates with probabilistic logic programs [3], and Logic Tensor Networks provide differentiable first-order reasoning [6]. OIBench focuses on a different axis: how a reasoner allocates symbolic work over a candidate domain, whether it preserves the valid solution, and whether the reasoning trajectory itself can be checked.

The benchmark also makes a useful distinction between heuristic intelligence and logical authority. A learned operator selector may be wrong about which restriction is most useful, as the coloring results demonstrate. That error need not compromise correctness if a symbolic verifier controls state transitions. This separation is attractive for trustworthy systems: neural components can optimize search or propose structure while exact constraints guard admissibility.

The current results should not be interpreted as evidence that neural ranking is inherently superior. Probe-32, which uses only a tiny symbolic sample, is both cheaper than exact greedy selection and more efficient than the reference neural policy. That is an important benchmark outcome. A useful benchmark should be able to falsify fashionable assumptions, including the assumption that adding a neural module necessarily improves reasoning.

8. Limitations and Threats to Validity

First, OIBench 1.0 uses synthetic finite domains. This enables exact verification, but does not capture the open-ended ambiguity of natural language, perception, scientific discovery, or large theorem-proving environments. Second, all released tasks have a unique solution. Multi-solution variants would require set-preservation metrics rather than singleton survival.

Third, Track A supplies a valid operator pool. It therefore evaluates operator selection and contraction, not unrestricted operator invention. Tracks B and C define interfaces for domain construction and certified derived operators, but the supplied reference experiments do not solve those harder problems. Fourth, candidate-evaluation cost is an implementation-neutral symbolic-work metric rather than a physical runtime model. Systems using compiled solvers, vectorized evaluation, SAT encodings, or proof caching may exhibit different wall-clock profiles.

Fifth, the neural baseline is intentionally modest and trained on only 180 of the 600 available training tasks. Its purpose is to establish a reproducible learned-selection reference, not a performance ceiling. Stronger graph neural networks, transformers over constraint sets, reinforcement learning, or large language model policies may substantially improve operator ranking. Finally, because test solutions are included in the research release for transparent verification, leaderboard-style use should employ a hidden-label test copy or server-side evaluator.

9. Reproducibility and Data Availability

The accompanying package contains deterministic generation code, exact constraint semantics, train/development/test JSONL files, reference solvers, the compact neural selector, per-task test metrics, statistical analysis, figures, and a package verifier. The release seed and software dependencies are recorded in machine-readable form. The benchmark can be regenerated from source and every released task can be rechecked by exhaustive enumeration. Before final journal submission, the package should be deposited in a persistent public repository and the resulting DOI inserted here.

10. Conclusion

OIBench 1.0 provides a concrete way to study a claim that is often expressed only metaphorically: instead of searching harder for an answer, a reasoning system can reshape the effective problem space so that invalid alternatives disappear and the valid solution becomes the certified survivor. The benchmark operationalizes this idea through finite

domains, explicit operators, exact symbolic verification, and metrics that separate correctness from contraction efficiency.

The first reference results already reveal nontrivial behavior. Learned operator ranking significantly reduces symbolic work relative to random ordering, yet a simple sample-based symbolic heuristic remains stronger and the neural policy transfers poorly to one family. These outcomes are useful precisely because correctness is held constant. OIBench therefore offers a test bed for neurosymbolic systems that aim to learn where to reason while retaining symbolic guarantees about what reasoning steps are allowed.

The broader research question remains open: can AI systems learn not only to select operators inside a supplied domain, but to construct better domains and invent new certified operators across problem classes? OIBench defines that progression as an explicit experimental agenda rather than assuming it has already been achieved.

Declarations

Funding. This research received no external funding.

Competing interests. The author declares no competing financial or personal interests that could have influenced the work reported in this manuscript.

Generative AI and AI-assisted technologies. Generative-AI tools were used to assist with language drafting, code development, debugging, and preparation of the submission package. The author reviewed the generated material, regenerated all reported computational outputs from deterministic code, and takes responsibility for the final content.

Author contributions (CRediT). Thanh Trung Phan: Conceptualization; Methodology; Software; Formal analysis; Investigation; Data curation; Validation; Visualization; Writing - original draft; Writing - review & editing.

References

- [1] A. d'Avila Garcez and L. C. Lamb, Neurosymbolic AI: The 3rd Wave, arXiv:2012.05876 (2020).
- [2] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Van den Broeck, A Semantic Loss Function for Deep Learning with Symbolic Knowledge, Proceedings of ICML 2018, PMLR 80, 5502-5511 (2018).
- [3] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt, DeepProbLog: Neural Probabilistic Logic Programming, Advances in Neural Information Processing Systems 31, 3749-3759 (2018).
- [4] T. Rocktaschel and S. Riedel, Learning Knowledge Base Inference with Neural Theorem Provers, Proceedings of the 5th Workshop on Automated Knowledge Base Construction, 45-50 (2016).
- [5] R. Evans and E. Grefenstette, Learning Explanatory Rules from Noisy Data, Journal of Artificial Intelligence Research 61, 1-64 (2018).
- [6] S. Badreddine, A. d'Avila Garcez, L. Serafini, and M. Spranger, Logic Tensor Networks, Artificial Intelligence 303, 103649 (2022).
- [7] G. Singh, R. Tommasini, S. Bhatia, and R. Mutharaju, Benchmarking Neuro-Symbolic Description Logic Reasoners: Existing Challenges and A Way Forward, Neurosymbolic Artificial Intelligence (2024).
- [8] R. Dechter, Constraint Processing, Morgan Kaufmann, San Francisco (2003).
- [9] P. Hitzler and M. K. Sarker (eds.), Neuro-Symbolic Artificial Intelligence: The State of the Art, IOS Press (2022).