Verified Program Synthesis with a Symbolic Knowledge Graph and an Un-gameable Compiler Oracle:
A No-Bake, Self-Extending COBOL System

Luke Kist
Independent Research (draft — September 4, 2026)

Abstract

We describe and evaluate a domain program-synthesis system that couples (i) symbolic, provenance-tracked knowledge retrieval—knowledge is *attached* at inference rather than *baked* into model weights; (ii) deterministic program composition; (iii) an *executable* verification gate (compilation plus runtime behavioural checking against ground truth); and (iv) explicit refusal. The system generates behaviourally correct GnuCOBOL programs across three structurally distinct program families, serves each answer in ≈ 486 ms on a single CPU core with no GPU, and *never emits unverified code*: every answer is either proven or refused. Because the verifier is an external deterministic oracle (the compiler and runtime) rather than a model grading itself, self-improvement cannot be reward-hacked—a failure mode reported for the Darwin Gödel Machine. We further show (a) a negative result: a from-scratch 14.0M-parameter neural model that *bakes* the domain corpus into weights attains 5.0% functional recall and 0/299 on structurally unseen prompts (95% CI $[0.0\%, 1.3\%]$), motivating the no-bake design; (b) a self-learning loop that researches, *proves*, and teaches itself a new operation; (c) compiler-judged evolution that grows a verified-capability archive from a single seed; and (d) compositional generalisation: a withheld composite operation, constructed by chaining two independently verified blocks, is behaviourally correct on 40/40 non-trivial cases (95% CI $[91.2\%, 100\%]$). We report all rates with confidence intervals, separate VERIFIED/REFUSED/WRONG, and state limitations prominently: the working generator is deterministic composition (the neural model did *not* generalise), the mutation and composition spaces are hand-bounded, and the intent-recognition layer is brittle keyword matching. We frame the broader hypothesis—that useful domain capability may scale more with representation, specialisation, and verification than with parameter count—as *open*.

1 Introduction

Large language models answer confidently and occasionally fabricate; the failure is silent and the caller cannot locally detect it. For low-resource, high-stakes domains—legacy COBOL still runs banking and government batch systems—a confidently wrong program is unacceptable. We take the opposite design stance to parameter scaling. *Knowledge is attached, not baked*: verified facts live losslessly in a symbolic graph and are retrieved at inference, rather than being lossily compressed into model weights. A tiny (or absent) model performs only *translation* of retrieved structure into code, and an external *compiler oracle* decides truth. The system's defining property is not fluency but honesty: every output is compiler-and-runtime *proven* or explicitly *refused*.

2 System

The pipeline is: **recall** (retrieve a grounded operation and parameters) → **compose** (instantiate a program deterministically) → **verify** (compile with `cobc -free -c`; then link, run against ground truth) → **serve** or **refuse**. Two extensions close the loop: **self-learning** (on an unknown request, propose candidate operations, prove one against worked examples, and *teach* it into the graph as a new provenance-stamped fact) and **evolution** (mutate verified building blocks, retaining every compiler-proven novel behaviour). When the system cannot solve and cannot self-derive, it **escalates** to a larger teacher model whose proposal is still subject to the same oracle.

Definition 1 (Verification gate)**.** *Let* Π *be a candidate program,* C *the compiler, and* $P = \{(x_i, y_i)\}_{i=1}^{m}$ *a probe set of input/output pairs with* y_i *the ground-truth behaviour. The gate is*

$$\mathcal{G}(\Pi; P) \;=\; \big[\, C(\Pi) \text{ succeeds} \,\big] \;\wedge\; \bigwedge_{i=1}^{m} \big[\, \mathrm{run}(\Pi, x_i) = y_i \,\big].$$

An answer is served iff $\mathcal{G}(\Pi; P) = \mathrm{true}$*; otherwise the system refuses.*

Lemma 1 (Bounded soundness and un-gameability)**.** *For any* Π *with* $\mathcal{G}(\Pi; P) = \mathrm{true}$*,* Π *compiles and is behaviourally exact on every probe in* P *(deterministically, with certainty). Moreover* $\mathcal{G}$ *is an external deterministic function of* (Π, P) *that consumes no signal from the proposer; hence no proposer—model, mutation, or teacher—can cause* $\mathcal{G}$ *to accept a* Π *that fails* P*.*

Proof sketch. Both conjuncts of $\mathcal{G}$ are decided by executing C and the compiled binary, which are fixed external programs; their outputs do not depend on how Π was produced. Acceptance therefore certifies exactness on P and cannot be forged. □ □

Scope of the guarantee (stated honestly). Lemma 1 certifies behaviour *on* P, not on all inputs, and it does not certify that Π matches the user's *intent* if recall selected the wrong operation. Intent errors are thus the residual risk (§5); the oracle guards correctness-given-intent, not intent itself.

3 Experiments

All experiments use GnuCOBOL 3.2 on Apple Silicon, deterministic seed `20260728`. Rates are reported with Wilson 95% score intervals. Provenance (checkpoint, dataset, code, and compiler hashes) is frozen in `ARCHIVE-20260904/PROVENANCE.md`.

Table 1: Core results with Wilson 95% confidence intervals. VERIFIED = compiled *and* behaviourally correct; REFUSED = out-of-coverage, correctly declined; WRONG = incorrect code served.

Measurement	rate	95% CI	wrong
Deterministic system, self-test (3 families)	13/13	$[77.2, 100]\%$	0
Adversarial dogfood (correct-or-refuse)	27/27	$[87.5, 100]\%$	0
Compositional $A \circ B$ (non-trivial)	**40/40**	$[\mathbf{91.2}, \mathbf{100}]\%$	0
Baseline 7B COBOL finetune, compile	42/146	$[22.0, 36.6]\%$	—
Neural 14M, functional recall (in-domain)	3/60	$[1.7, 13.7]\%$	—
Neural 14M, unseen templates	0/299	$[0.0, \mathbf{1.3}]\%$	—

The no-bake motivation (negative result). A from-scratch 14.0M-parameter transformer trained *prompt→COBOL* memorises: 5.0% functional recall in-domain and, on a template-disjoint

split, 0/299—upper CI bound 1.3%. A hybrid variant given a one-line symbolic fetch as input still failed (validation cross-entropy 2.93 vs. 1.96 baseline; malformed output on unseen templates). We conclude that baking corpus knowledge into weights does not generalise here; the working generator is deterministic composition over *attached* knowledge.

Coverage across structure. The deterministic system serves 13 operations across three structurally distinct shapes: numeric-table LINKAGE subprograms (min/max/sum/sum-abs/count-{neg,pos,zero,over}/a string LINKAGE subprograms (vowel/digit counts via `INSPECT TALLYING`), and self-contained indexed-file (KSDS) programs.

Self-learning. For *product*—absent from coverage—the system proposes candidate operations, proves `MULTIPLY` (seed 1) against worked examples $[2,3,4] \to 24$, $[2,5,3] \to 30$, and teaches the fact into the graph (`grounded=true` on subsequent query). It first *refused* an overflowing variant, illustrating fail-closed integrity.

Evolution. Seeding an archive with {`sum`} and mutating (seed $\times$ operation), the compiler retains every proven novel behaviour: 14 verified capabilities in one generation, including rediscovered product/min/max/count and eight novel-unnamed but verified behaviours. Following the Darwin Gödel Machine, but with the compiler as the fitness function, acceptance cannot be reward-hacked (Lemma 1).

Compositional generalisation. The composite "count values above the average" ($A \circ B$) is withheld and never seen whole; it is constructed by chaining two independently verified blocks (compute-average; count-above-threshold). Across 8 table sizes $\times$ 5 genuinely varied random tables, 40/40 compile and are behaviourally correct, all non-trivial ($0 <$ answer $< n$), 95% CI $[91.2, 100]$%.

4 Cost

Table 2: Measured cost. The working generator is deterministic, so the model need not be resident to serve; the parameter count is an artefact of the (superseded) neural experiments.

Parameters (neural artefact)	14.0M
Checkpoint size	57.3 MB fp32 ($\approx$ 14.3 MB int8)
Parameters on serving hot-path	**0** (deterministic composer)
Latency per verified answer	$\approx$ 486 ms
Accelerator	none (single CPU core)

5 Limitations

We state these prominently.

1. **The generator that works is deterministic composition, not a neural model.** The neural model did not generalise; we do *not* claim "a 14M model generates." We claim the *system*.
2. **Composition and mutation are guided.** The chaining rule and the mutation space are hand-provided bounded sets. Autonomous decomposition and free self-code-rewriting are future work.
3. **Intent recognition is brittle.** Recall is keyword matching; mis-recall (wrong operation that still passes its own verification) is the one error the gate cannot catch (Lemma 1) and is the principal scaling risk. It should become a learned classifier.

4. **Small N.** Several headline rates are 100% but on $N \leq 40$; the honest lower CI bounds are 77–91%. A template-disjoint, deduplicated benchmark with CIs is required future work.
5. **No distillation.** No teacher→student weight transfer was performed; the escalation teacher's outputs are *filtered* by the oracle, not copied.
6. **Internal validity.** The dogfood set was authored by the same party who fixed the recall bugs it exposed; the reported 0 mis-recall may be tuned-to-fixes.

6 Related Work and Discussion

The design combines retrieval-augmented generation, neuro-symbolic knowledge graphs, and program synthesis, and it operationalises the empirical (rather than proof-carrying) spirit of the Gödel Machine, using an *external* oracle to obtain the un-gameability the self-graded Darwin Gödel Machine lacked. Whether a fixed small footprint can acquire *structurally diverse, compositionally novel, verified* capabilities that previously required billion-parameter models is an *open* question; the ablations below are designed to answer it, and we make no "changes ML" claim—by design, the benchmark should make such a claim unnecessary.

Proposed ablations and benchmark. (i) remove verification; (ii) remove refusal; (iii) remove the fetch; (iv) a 3M/10M/30M scaling curve; (v) a matched-protocol comparison to the 7B under identical branch/repair/verify budgets; (vi) a template-disjoint, deduplicated held-out test set with CIs. **Threats to validity:** small N; probe-set coverage bounds Lemma 1; author-tuned dogfood; the escalation "frontier" rung was a stand-in for an API call.

7 Reproducibility

Code, seeds, and a hash manifest of checkpoints, datasets, harnesses, and the compiler version are frozen in `ARCHIVE-20260904/PROVENANCE.md`. Each experiment is a single runnable script (`cobol_build.py`, `dogfood.py`, `self_learn.py`, `cobol_evolve.py`, `compose_test.py`, `teacher_escalation.py`) Neural results require MLX and are reproduced from frozen metadata, not re-run.

References

[1] J. Schmidhuber. Gödel Machines: Self-Referential Universal Problem Solvers Making Provably Optimal Self-Improvements. TR IDSIA-19-03, arXiv:cs.LO/0309048, 2003.

[2] J. Zhang, S. Hu, C. Lu, R. Lange, J. Clune (Sakana AI / UBC). Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. arXiv:2505.22954, 2025.

[3] P. Lewis et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS, 2020.

[4] S. Gulwani. Automating String Processing in Spreadsheets using Input-Output Examples. POPL, 2011.

[5] S. Gunasekar et al. Textbooks Are All You Need. arXiv:2306.11644, 2023.