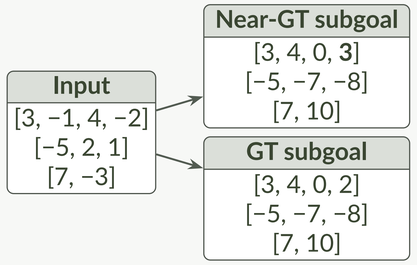
Solver-Aware Decompositions for Programming-by-Example, Zenkner et al.

1 Motivation

- **Programming-by-Example:** find a program that maps inputs to outputs.
- **Divide and conquer** is a promising way to scale synthesis to longer programs.
- **Divide:** a decomposer predicts an intermediate state (subgoal) that the synthesizer uses to construct the next program step.
- **Conquer:** a synthesizer (the solver) writes this step.
- **The catch:** decomposer and synthesizer are both trained on ground-truth (GT) subgoals, yet a near-GT subgoal can lie outside the DSL, and even the GT subgoal can be hard for the synthesizer.

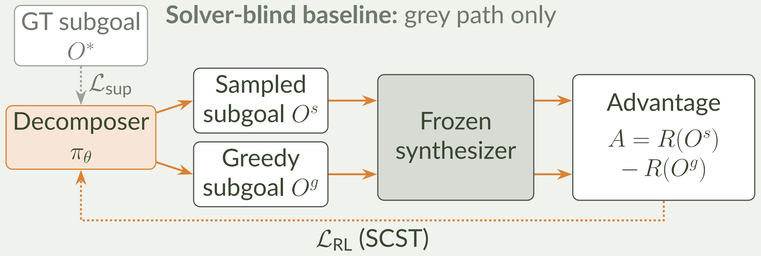
Should we optimize decompositions for similarity to ground truth or for downstream solver success?



× Near GT, but infeasible: no operation produces it; GT supervision barely penalizes it

× Decomposer not aligned to the synthesizer: realizable by ScanI1 (−), but synthesis fails

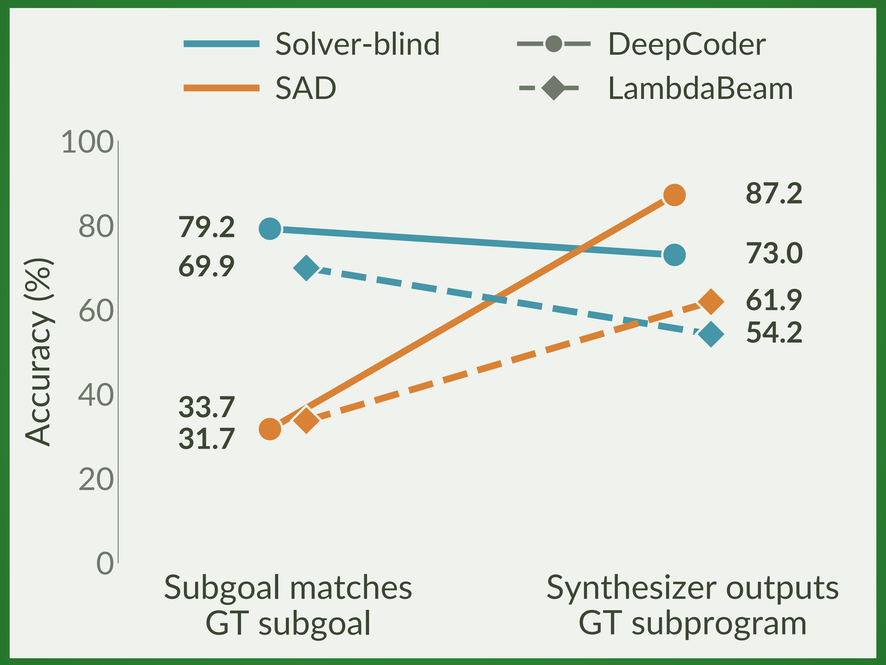
2 Solver-Aware Decomposition (SAD)



Solver-blind baseline: grey path only

- **Idea:** the frozen synthesizer scores sampled and greedy subgoals by how well it recovers the GT subprogram p^*; the difference A trains the decomposer.
- **Reward:** $R(O) = -\text{CE}(\text{Synth}(I, O), p^*)$
- **Supervised loss** $\mathcal{L}_{\text{sup}}$ keeps subgoals meaningful; without it, task accuracy is below 20 % (single seed).
- **Objective:** $\mathcal{L} = \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{RL}} - \lambda H(\pi_\theta)$, $\lambda = 0.01$

Dividing Requires Knowing How to Conquer



SAD's subgoals match the GT subgoals less often, yet lead the synthesizer to the GT subprogram more often.

Per step under teacher forcing, in-distribution, mean of 5 seeds

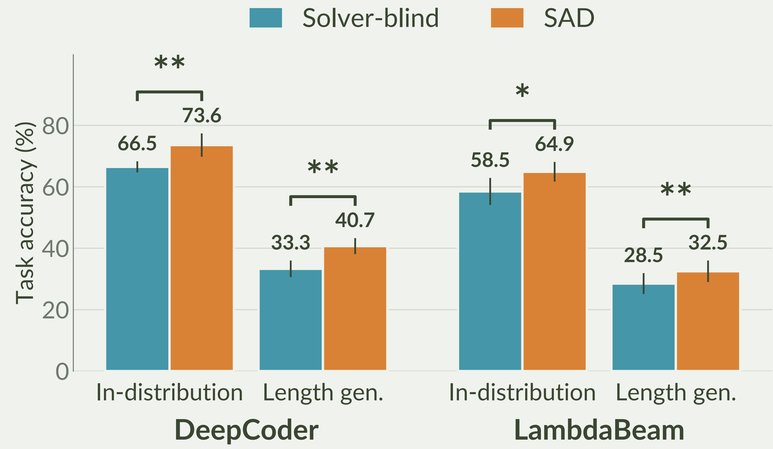
[1] J. S. Rule et al. Symbolic metaprogram search improves learning efficiency and explains rule learning in humans. *Nature Communications* 15, 6847 (2024).

TU Clausthal CORE

Paper & code

3 Results

Setup: DeepCoder and LambdaBeam list-processing DSLs. Length generalization: train on programs of length 1–4, test on length 5. Solver-blind uses the same synthesizer, data and beam search as SAD.



- **Gains hold under length generalization:** +7.4 and +4.0 points on DeepCoder and LambdaBeam, i.e., +22.4 % and +13.9 % relative.
- **Complementary:** SAD solves 63 (DeepCoder) and 110 (LambdaBeam) tasks that an oracle always using GT subgoals fails on, though the oracle solves more overall.
- **No choice, no gain:** on RobustFill, where the decomposition order is fixed, both reach 94.1 %.

Beyond the main benchmarks

- **Transfers zero-shot:** 42.3 % vs. 33.0 % on 129 list functions of Rule et al. [1].
- **Carries over to an LLM:** Llama-3.1-8B (LoRA), 34.5 % vs. 30.5 % on 200 DeepCoder tasks.

4 Limitations

- SAD must be retrained when the synthesizer changes.
- The reward needs a GT program and a learned solver.
- The LLM result is preliminary (single seed).

5 Takeaway

- GT subgoals are not always the subgoals a given synthesizer solves best.
- Decomposition quality depends on the synthesizer, and the decomposer can be trained against it.