Learning to Find Proofs and Theorems by Learning to Refine Search Strategies
The Case of Loop Invariant Synthesis

Jonathan Laurent1,2 and André Platzer1,2 (Carnegie Mellon University1, Karlsruhe Institute of Technology2)

Motivation
Can theorem proving be learned without examples of proofs or theorems?
• Automated theorem proving has crucial applications in many fields, including software verification.
• The dominant approach for scaling it up with machine-learning is to use imitation learning. However, human proof data is scarce (and nearly nonexistent in many domains).
• Reinforcement learning alleviates the need for human proofs but training tasks of suitable relevance and diversity are still needed (equally scarce).

Our approach
A teacher/solver architecture in which both agents use RL to refine generic expert-defined strategies expressed as nondeterministic programs.

Teacher (AlphaZero agent)
Solver (AlphaZero agent)

Evaluation Setting
Verifying imperative programs by generating loop invariants:
• Training data unavailable and hard to generate!
• No pre-existing deep-learning agent can generalize across instances.

Teacher Strategies
For RL to properly generalize across instances, diverse and relevant theorems (i.e. initial states in the strategy MDP) must be provided. Generating such theorems is often harder than proving them (for invariant synthesis, naive approaches based on rejection sampling produce low-quality training tasks).

Key insight: teacher agents can be implemented similarly to solver agents, by using RL to refine expert-defined strategies. To do so, we introduce the concept of a conditional generative strategy, which generates a problem in two steps:
1. Sample a set of random constraints.
2. Generate a problem nondeterministically and get rewarded for satisfying as many constraints as possible (amenable to learning).

A Flexible Strategy Language
We propose a flexible language for experts to define search strategies in the form of nondeterministic programs, using the CHOOSE, REWARD and EVENT operators.

A solver strategy for loop invariant synthesis:

```
def solve(strategy: str, program: Program) -> List[Formula]:
    # initialize
    init = And(Not(guard), post)
    # solve
    return inv
```

In this strategy, the loop body is solved nondeterministically.

Strategies in this language can be compiled by our tool into MDPs that are amenable to neural-guided search and RL. Non-final states correspond to nondeterministic choice points:

Experiments
• We implemented our strategy language along with a toolchain to write, debug and compile strategies into MDPs.
• We trained a teacher and a solver agent for invariant synthesis based on two strategies written in this language. We used Dynamic Graph Transformers with 2M parameters as neural oracles and trained both agents for 160K AlphaZero episodes (with 32 MCTS simulations per move).
• Training took 16 hours on a 10-core CPU and 1 Nvidia RTX 3080 GPU.

Policy % Problems solved
Random 18.4 ± 0.0
Network (untrained teacher) 39.7 ± 1.6
Network (trained teacher) 61.5 ± 0.4

Takeaway: the trained network can solve a majority of problems with no search at all despite never seeing those during training. Using an untrained teacher leads to an inferior solver with decreased generalization capabilities.

Conclusion and Future Work
We demonstrated the possibility of learning a theorem proving task (invariant synthesis) in the absence of both proof and theorem examples.

• Broader vision: interactive provers allow users to write teacher and solver strategies for various domains in a distributed way. A large language model is fine-tuned to serve as a shared oracle that generalizes across those.
• Future work:
  • Evaluation of our framework in other application domains
  • Intrinsic teacher rewards (curiosity, solver rewarding the teacher directly…)
  • Integration with large pretrained language models