Theorem-Proving Environments

Nathan Ng

CSC2547: Learning to Search

Theorem Proving

- What is a theorem?
  - Statement proven based on basis of previously established statements
    - Premise: If I attend UofT, I am a student
    - Premise: I attend UofT
    - Theorem: I am a student
- Why do we want to prove theorems more efficiently?
  - Integrated Circuit Design
  - Program Verification
  - Formulating large proofs (Kepler Conjecture)

Propositional Logic

- 0th-order logic
  - Deals with statements that are either true or false
  - $\neg(A \vee B) = \neg A \wedge \neg B$
  - Proving a proposition is true can be reduced to SAT-solving
- Problem: not expressive enough for many theorems
  - Prove that there are an infinite number of primes
    - Only have a finite number of variables to use!
  - Prove that if 1 < 4 and 4 < 9, then 1 < 9
    - No concept of relations!

Predicate Logic

- 1st-order logic
  - Defines **predicates** and **quantifiers** over variables
    - predicates: expression over variables (property or relationship)
    - quantifiers: describe a set of variables we would like to consider
      - all philosophers are scholars
      - **for all** philosopher(Y), **scholar**(y)
- Still not expressive enough!
  - Prove that the set of prime numbers is countable
    - need some way of expressing relationships between sets and predicates themselves

Higher Order Logic

- Defines set of predicates and quantifiers that can be applied to all domains
  - In first order logic, cannot express the predicates that A and B have some property in common
  - In higher order logic, we can write $\exists P, (P(A) \wedge P(B))$

What is an ATP?

- Automatic Theorem Prover
  - Can we program a computer to automatically prove theorems based on some core axioms?
  - very difficult problem
    - how does the computer know what action/strategy to take to reduce problem or solve subproblem?
    - higher order logics make procedures and verification more complex
- Can we build a framework for humans to use machines to help develop formal proofs?

What is an ITP?

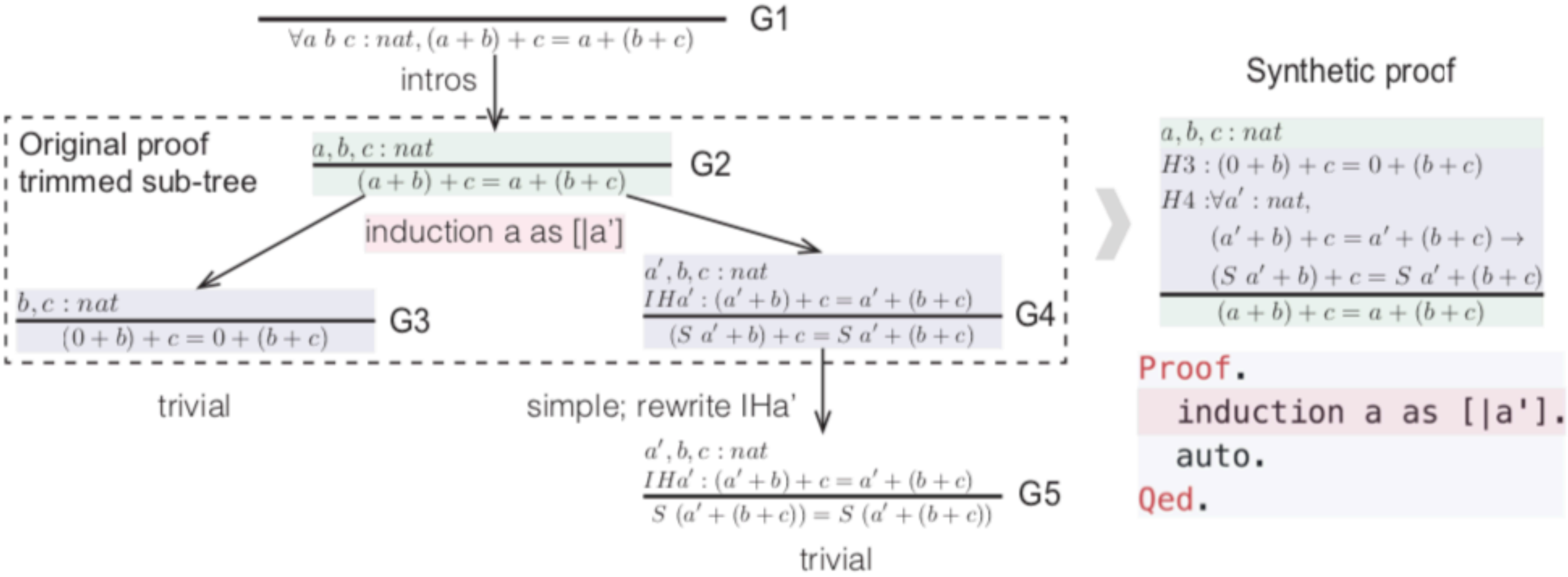
- Interactive Theorem Prover
  - Not automatic!
    - Machine-aided theorem proving, but ultimately human-driven
  - automatically check proof
  - build repositories of previously proven knowledge
  - abstracts away easy tasks so human can focus on hard ones
- Why is this useful?
  - logically sound
  - allows for meta-reasoning
  - can be automated
  - practical and effective

How do we use an ITP?

- input theorem to prove as a *goal*
- ITP provides *tactics* to manipulate goal
  - may include *arguments* of previously proven theorems
  - produces *subgoals* to prove
- once all subgoals can be proved, goal is proven
- goals and subgoals form tree structure

Partial Evaluation of Functional Logic Programs [Alpuente, 1998]

How do we use an ITP?



Learning to Prove Theorems via Interacting with Proof Assistants [Yang, 2019]

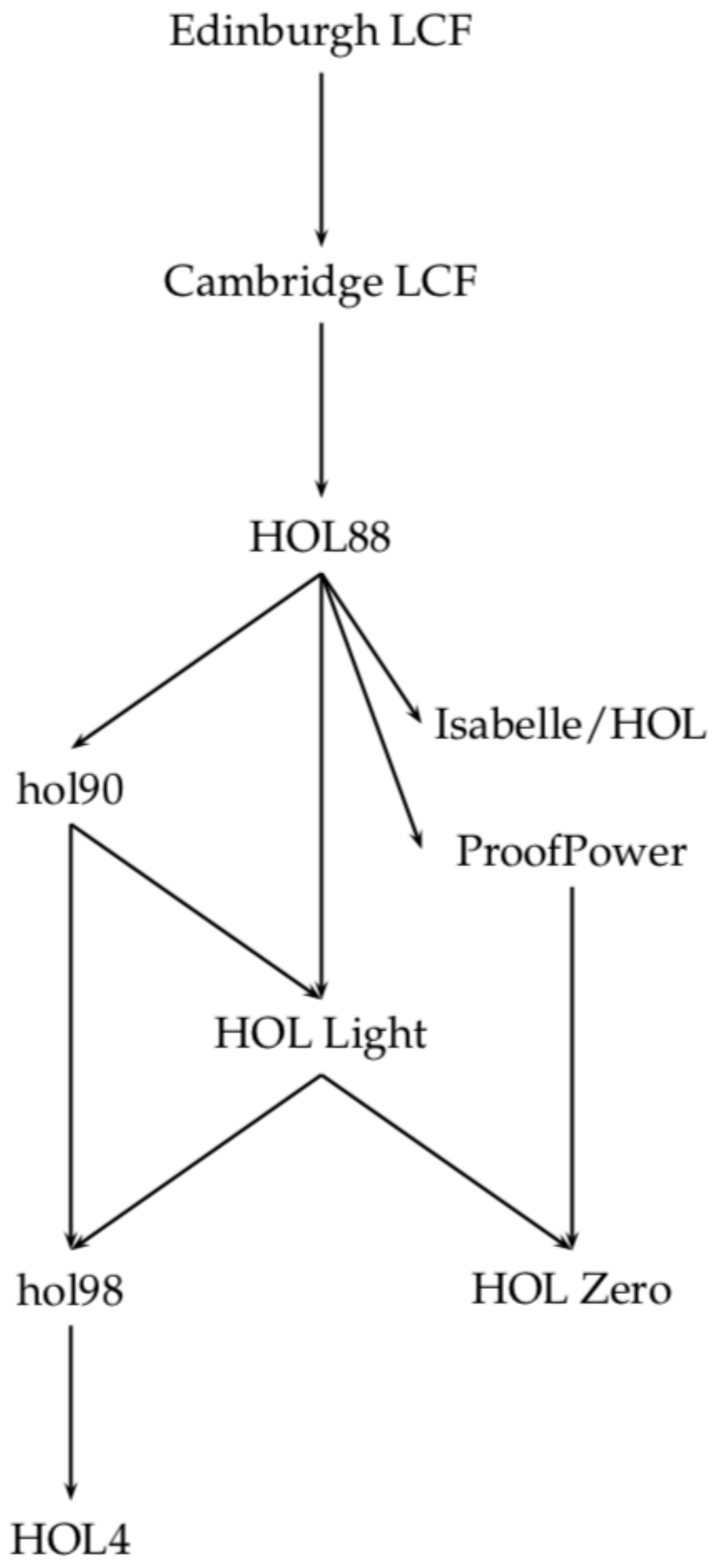
HOL

- Higher Order Logic (HOL)
  - small trusted kernel of theorems
  - abstract data types
  - new theorems built on top using library functions
    - what does this mean for all theorems in this system?

A Brief Introduction to Higher Order Logic [Nesi, 2011]

HOL Light

- Intended to be a foundationally simpler version of HOL
  - Kernel is only a few hundred lines of code
    - highly scrutinized and self-verified
- 10 basic primitive inference rules
- 3 mathematical axioms
- extendable and programmable
  - can build public libraries of systems of proofs/theorems
  - automate theorem proving processes



Interactive Theorem Proving [Tuerk, 2019]

Coq

- Another ITP similar to HOL
- Different logical basis allows for dependent types
  - matmul (nat n m p): mat n m -> mat m p -> mat n p
  - In HOL, need to explicitly describe this dependence
- Less “push-button” than HOL
  - more explicit but also easier to write more complicated proof automation

GamePad: A Learning Environment for Theorem Proving [Huang. 2019]

Other ITPs

- Mizar
- Isabelle
- HOL4
- Lean

GamePad: A Learning Environment for Theorem Proving [Huang. 2019]

Towards an ATP in an ITP Environment

- Much of ITP is still human-driven
  - What tactic should we use on a given subgoal?
  - What arguments and theorems should we use in a given tactic?
  - How do we balance exploration of other strategies with investigation of current ones?
- Can we learn policies to effectively solve these problems without the need for humans?

HOList: An Environment for Machine Learning of Higher-Order Theorem Proving

Kshitij Bansal, Sarah M. Loos, Markus N. Rabe, Christian Szegedy, and Stewart Wilcox

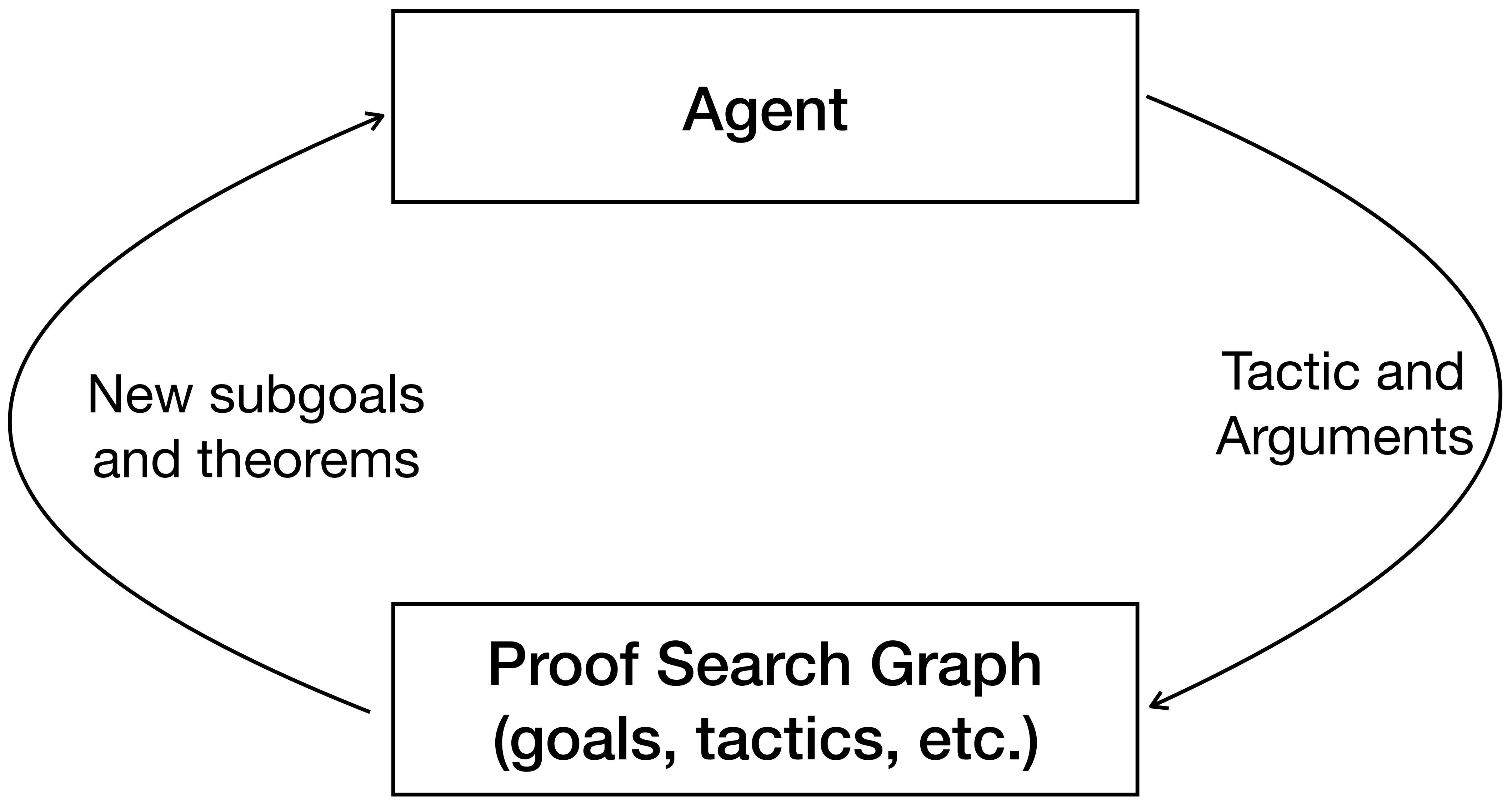
Imitation Learning

- From previous ITP proof logs, we have proof context, and human tactic/arguments
- Supervised learning on human examples
  - Given some proof context (goals, subgoals, proven theorems, etc.), decide what tactic and arguments to use
- Problem: limited by the amount of training examples humans can generate
  - System will learn to create proofs like humans, but what if this isn't the best way?

GamePad: A Learning Environment for Theorem Proving [Huang. 2019]

Reinforcement Learning

- Allow agent to learn which actions to take itself
- Formulation as RL Problem
  - state
    - Proof search graph
  - action
    - tactic/argument
  - reward
    - proving a goal or subgoal
  - transition
    - application of tactics to current graph



DeepHOL

- Can we build an effective reinforcement learning agent within the HOL Light environment?
  - Need some way to decide which tactic to apply to a goal
    - Rank tactics
    - Create arguments for each tactic
  - Keep track of goals and state of proof search in data structure (graph)

HOList: An Environment for Machine Learning of Higher Order Theorem Proving [Bansal, 2019]

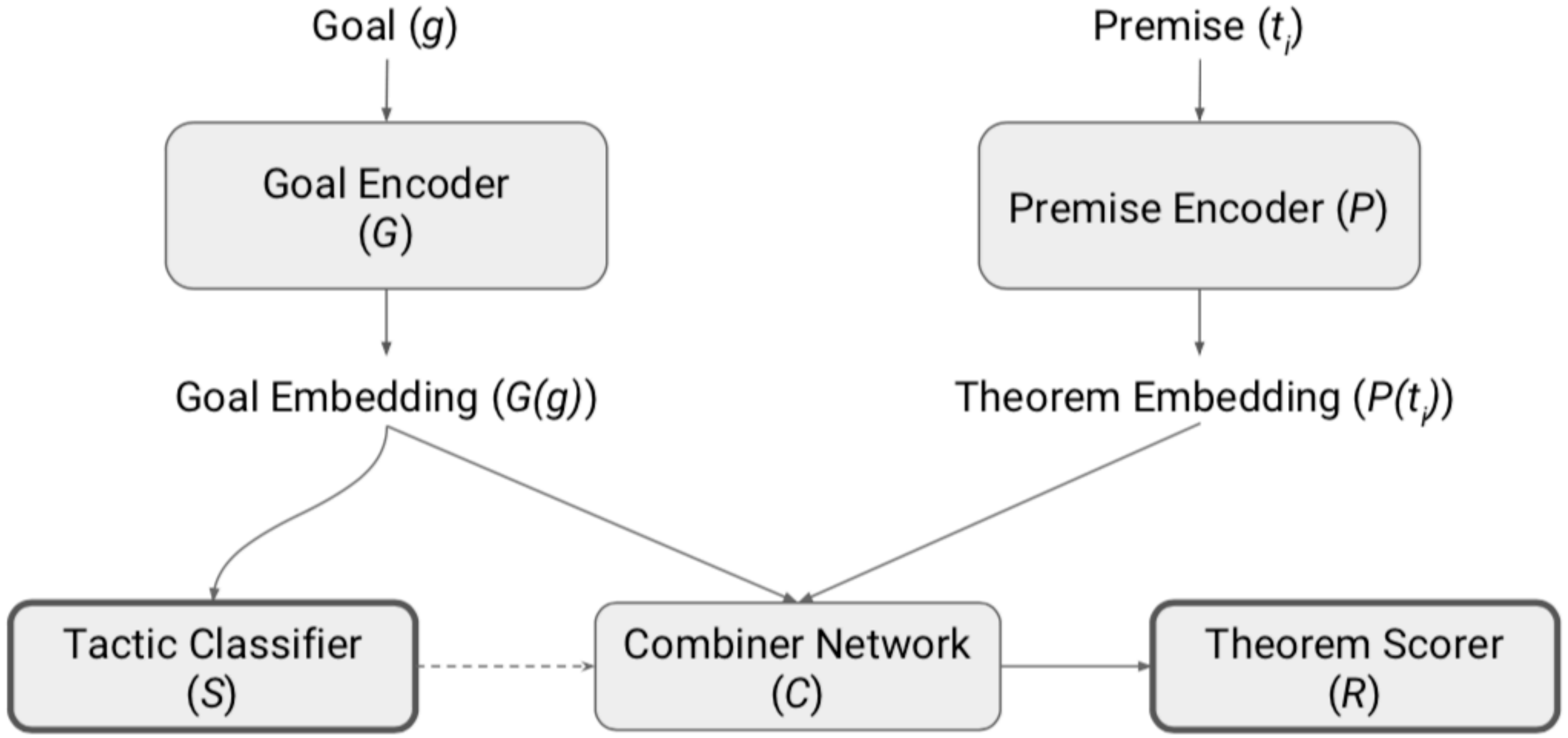
Dataset/Environment

- Proof export for HOL Light verification
- Theorem corpora for training and validation
  - core: theorems needed for tactics
  - complex: theorems of complex calculus
  - flyspeck: lemmas and theorems of Kepler Conjecture
- examples consist of goal, tactic, and arglist
  - goal: theorem to prove
  - tactic: tactic that led to a successful proof
  - arglist: arguments passed to tactic as arguments

HOList: An Environment for Machine Learning of Higher Order Theorem Proving [Bansal, 2019]

DeepHOL: Action Generator

- Two towers
  - Goal Encoder generates Goal Embedding
  - Premise Encoder generates Premise Embedding
- Goal embedding used to generate tactics to use
- Premise embedding, goal embedding, and selected tactic used to generate arguments to use



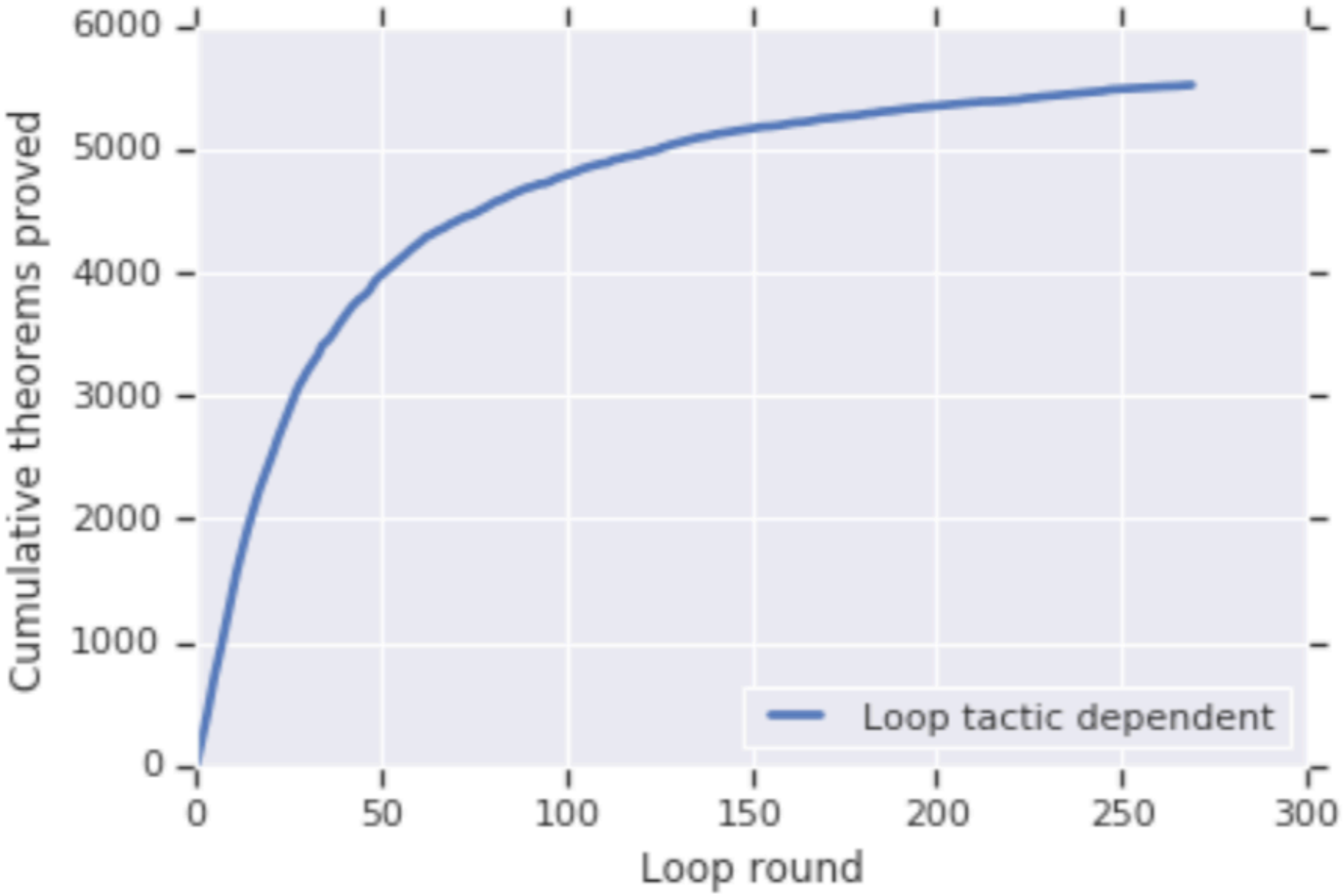
HOList: An Environment for Machine Learning of Higher Order Theorem Proving [Bansal, 2019]

Training the Action Generator

- Start training with supervised learning
  - use human proof logs
- Continue training with reinforcement learning loop
  - Trainer and multiple provers running continuously
    - each round consists of random sample of theorems
      - human training examples (optional)
      - previous experiment's generated examples (optional)
      - freshly generated examples
      - historical training loop examples

HOList: An Environment for Machine Learning of Higher Order Theorem Proving [Bansal, 2019]

Results



Description	Proof success
ASM_MESON_TAC	6.1%
ASM_MESON_TAC + argument selection	9.2%
WaveNet	31.72%
Deeper WaveNet	32.65%
Wider WaveNet	27.60%
Loop	36.3%
Trained on loop output	36.8%
Loop tactic dependent	**38.9**%

HOList: An Environment for Machine Learning of Higher Order Theorem Proving [Bansal, 2019]

Other Approaches

- GamePad: A Learning Environment for Theorem Proving
  - fewer theorems in dataset (1602 vs 29462)
  - proxy metrics of tactic prediction instead of actual theorem proving
  - also framed as RL problem with similar strategy
- Learning to Prove Theorems via Interacting with Proof Assistants
  - ASTactic uses encoder-decoder architecture
  - Supervised learning with teacher forcing instead of RL
  - use Coq outputs of human proof steps as training examples
- TacticToe: Learning to Prove with Tactics
  - Learn tactic predictor from human examples
  - Apply MTCS during proof tree search

HOList: An Environment for Machine Learning of Higher Order Theorem Proving [Bansal, 2019]

GamePad

- Tactic Prediction
  - What tactic should we apply next given some input proof state?
- Position Evaluation
  - How many steps do we have left before we reach a successful proof?
  - Should be dependent on tactic predictor
    - better predictor uses less steps

Table 2: Test accuracies for position evaluation (Pos) and tactic prediction (Tac). † indicates kernel-level. ‡ indicates mid-level without implicit arguments. For tactic argument prediction, we report validation recall for models with a minimum precision of 10%

Model	Pos†	Pos‡	Tac†	Tac‡	Tac† arguments
Constant	53.66	53.66	44.75	44.75	-
SVM	57.37	57.52	48.94	49.45	-
GRU	65.30	65.74	58.23	57.70	25.98
TreeLSTM	68.44	66.30	60.63	60.55	23.91

GamePad: A Learning Environment for Theorem Proving [Huang. 2019]

ASTactic

- Encoder-decoder architecture
  - Encoding proof state (context and premises) using TreeLSTM
  - Use encoder embedding to generate tactic
- Teacher forcing
  - How to expand proof tree if prediction is wrong?
  - Force input at next step to be correct even if previous prediction was wrong

Method	Success rate (%)
trivial	2.4
auto	2.9
intuition	4.4
easy	4.9
hammer (default time limit)	17.8
hammer (extended time limit)	24.8
ours	12.2
ours + auto	12.8
ours + hammer	**30.0**

GamePad: A Learning Environment for Theorem Proving [Huang. 2019]