Reinforcement Learning for Interactive Theorem Proving in HOL4

Minchao Wu[1] Michael Norrish[1,2] Christian Walder[1,2] Amir Dezfouli[2]

[1]Research School of Computer Science
Australian National University

[2]Data61, CSIRO

September 14, 2020

Overview

- Interface: HOL4 as an RL environment
  - Enables interaction with HOL4.
  - Monitor proof states on the Python side.

Overview

- Interface: HOL4 as an RL environment
  - Enables interaction with HOL4.
  - Monitor proof states on the Python side.
- Reinforcement learning settings

Overview

- Interface: HOL4 as an RL environment
  - Enables interaction with HOL4.
  - Monitor proof states on the Python side.
- Reinforcement learning settings
  - Policies for choosing proof states, tactics, and theorems or terms as arguments.

Overview

- Interface: HOL4 as an RL environment
  - Enables interaction with HOL4.
  - Monitor proof states on the Python side.
- Reinforcement learning settings
  - Policies for choosing proof states, tactics, and theorems or terms as arguments.
  - Learning: policy gradient

Environment

- An environment can be created by specifying an initial goal.

```
e = HolEnv(GOAL)
```

- An environment can be reset by providing a new goal.

```
e.reset(GOAL2)
```

- The basic function is querying HOL4 about tactic applications.

```
e.query("∀l. NULL l ⇒ l = []", "strip_tac")
```

Environment

The `e.step(action)` function applies `action` to the current state and generates the new state.

```
e.step(action)
```

step takes an action and returns the immediate reward received, and a Boolean value indicating whether the proof attempt has finished.

Demo

- A quick demo.

RL Formalization

- A goal $g \in \mathbb{G}$ is a HOL4 proposition.

RL Formalization

- A goal $g \in \mathbb{G}$ is a HOL4 proposition.
- A fringe is a finite set of goals.

RL Formalization

- A goal $g \in \mathbb{G}$ is a HOL4 proposition.
- A fringe is a finite set of goals.
  - A fringe consists of all the remaining goals.

RL Formalization

- A goal $g \in \mathbb{G}$ is a HOL4 proposition.
- A fringe is a finite set of goals.
  - A fringe consists of all the remaining goals.
  - The main goal is proved if everything in any one fringe is discharged.

RL Formalization

- A goal $g \in \mathbb{G}$ is a HOL4 proposition.
- A fringe is a finite set of goals.
  - A fringe consists of all the remaining goals.
  - The main goal is proved if everything in any one fringe is discharged.
- A *state* s is a finite sequence of fringes.

RL Formalization

- A goal $g \in \mathbb{G}$ is a HOL4 proposition.
- A fringe is a finite set of goals.
  - A fringe consists of all the remaining goals.
  - The main goal is proved if everything in any one fringe is discharged.
- A *state* s is a finite sequence of fringes.
  - A fringe can be referred by its index i, i.e., $s(i)$.

RL Formalization

- A goal $g \in \mathbb{G}$ is a HOL4 proposition.
- A fringe is a finite set of goals.
  - A fringe consists of all the remaining goals.
  - The main goal is proved if everything in any one fringe is discharged.
- A *state* s is a finite sequence of fringes.
  - A fringe can be referred by its index i, i.e., $s(i)$.
- A *reward* is a real number $r \in \mathbb{R}$.

Examples

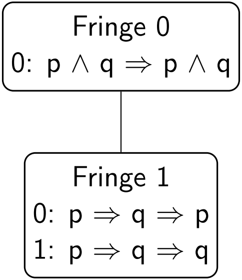

Figure: Example fringes and states

RL Formalization

- An action is a triple $(i, j, t) : \mathbb{N} \times \mathbb{N} \times \text{tactic}$.

RL Formalization

- An action is a triple $(i, j, t) : \mathbb{N} \times \mathbb{N} \times \text{tactic}$.
  - i selects the ith fringe in a state s.

RL Formalization

- An action is a triple $(i, j, t) : \mathbb{N} \times \mathbb{N} \times \mathrm{tactic}$.
  - i selects the ith fringe in a state s.
  - j selects the jth goal within fringe $s(i)$.

RL Formalization

- An action is a triple $(i, j, t) : \mathbb{N} \times \mathbb{N} \times \mathrm{tactic}$.
  - i selects the ith fringe in a state s.
  - j selects the jth goal within fringe $s(i)$.
  - t is a HOL4 tactic.

RL Formalization

- An action is a triple $(i, j, t) : \mathbb{N} \times \mathbb{N} \times \text{tactic}$.
  - i selects the ith fringe in a state s.
  - j selects the jth goal within fringe $s(i)$.
  - t is a HOL4 tactic.
- Example: $(0, 0, \texttt{fs[listTheory.MEM]})$

RL Formalization

- An action is a triple $(i, j, t) : \mathbb{N} \times \mathbb{N} \times \mathrm{tactic}$.
  - i selects the ith fringe in a state s.
  - j selects the jth goal within fringe $s(i)$.
  - t is a HOL4 tactic.
- Example: $(0, 0, \texttt{fs[listTheory.MEM]})$
- Rewards
  - Successful application: 0.1
  - Discharges the current goal completely: 0.2
  - Main goal proved: 5
  - Otherwise: -0.1

Example

Fringe 0

0: $p \wedge q \Rightarrow p \wedge q$

Figure: Example proof search

Example

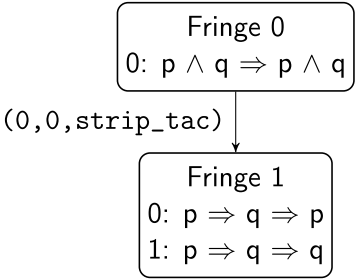

Figure: Example proof search

Example

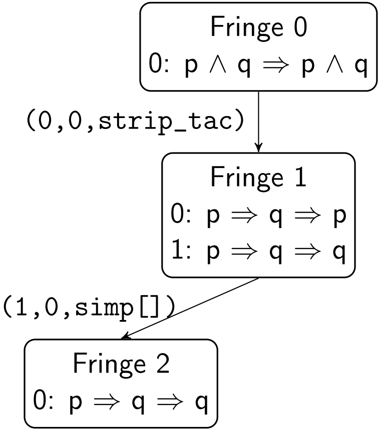

Figure: Example proof search

Example

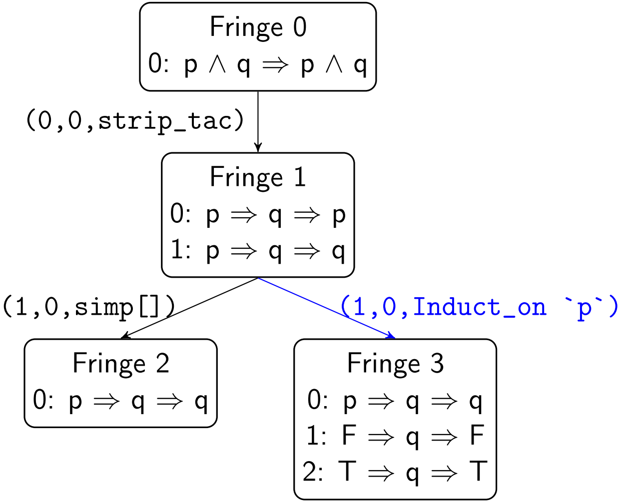

Figure: Example proof search

Example

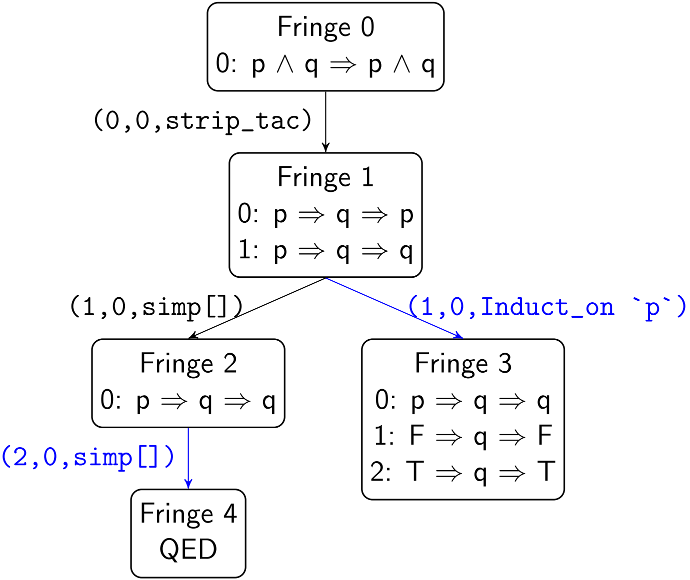

Figure: Example proof search

Choosing fringes

An action is a triple (i, j, t). Given state s.

- A value network $V_{\text{goal}} : \mathbb{G} \to \mathbb{R}$.

Choosing fringes

An action is a triple (i, j, t). Given state s.

- A value network $V_{\text{goal}} : \mathbb{G} \to \mathbb{R}$.
- The value v_i of fringe $s(i)$ is defined by:

$$v_i = \Sigma_{g \in s(i)} V_{\text{goal}}(g)$$

Choosing fringes

An action is a triple (i, j, t). Given state s.

- A value network $V_{\text{goal}} : \mathbb{G} \to \mathbb{R}$.
- The value v_i of fringe $s(i)$ is defined by:

$$v_i = \Sigma_{g \in s(i)} V_{\text{goal}}(g)$$

- Sample from the following distribution

$$\pi_{\text{fringe}}(s) = \text{Softmax}(v_1, ..., v_{|s|})$$

Choosing fringes

An action is a triple (i, j, t). Given state s.

- A value network $V_{\text{goal}} : \mathbb{G} \to \mathbb{R}$.
- The value v_i of fringe $s(i)$ is defined by:

$$v_i = \Sigma_{g \in s(i)} V_{\text{goal}}(g)$$

- Sample from the following distribution

$$\pi_{\text{fringe}}(s) = \text{Softmax}(v_1, ..., v_{|s|})$$

- By default, j is fixed to be 0. That is, we always deal with the first goal in a fringe.

Generating tactics

Suppose we are dealing with goal g.

- A tactic is either

Generating tactics

Suppose we are dealing with goal g.

- A tactic is either
  - A tactic name followed by a list of theorem names, or

Generating tactics

Suppose we are dealing with goal g.

- A tactic is either
  - A tactic name followed by a list of theorem names, or
  - A tactic name followed by a list of terms

Generating tactics

Suppose we are dealing with goal g.

- A tactic is either
  - A tactic name followed by a list of theorem names, or
  - A tactic name followed by a list of terms
- A value network

  $$V_{\text{tactic}} : \mathbb{G} \to \mathbb{R}^D$$

  where D is the total number of tactic names allowed.

Generating tactics

Suppose we are dealing with goal g.

- A tactic is either
  - A tactic name followed by a list of theorem names, or
  - A tactic name followed by a list of terms
- A value network
  $$V_{\text{tactic}} : \mathbb{G} \to \mathbb{R}^D$$
  where D is the total number of tactic names allowed.
- Sample from the following distribution
  $$\pi_{\text{tactic}}(g) = \text{Softmax}(V_{\text{tactic}}(g))$$

Argument policy

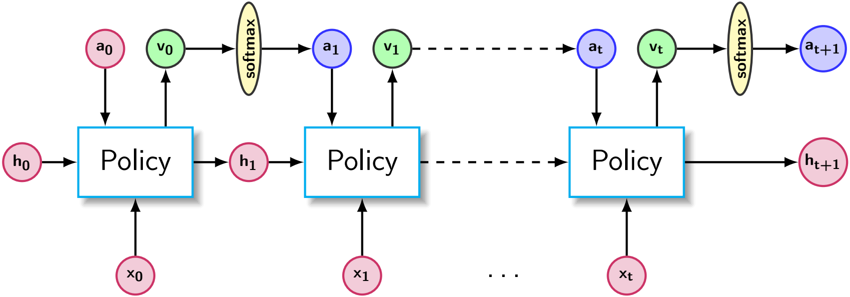

Figure: Generation of arguments. $\mathbf{x_i}$ is the candidate theorems. $\mathbf{h_i}$ is a hidden variable. $\mathbf{a_i}$ is a chosen argument. $\mathbf{v_i}$ is the values computed by the policy. Each theorem is represented by an N-dimensional tensor based on its tokenized expression in Polish notation. If we have M candidate theorems, then the shape of $\mathbf{x_i}$ is $M \times N$. The representations are computed by a separately trained transformer.

Generating arguments

Generation of arguments

Given a chosen goal g. Each theorem is represented by an N-dimensional tensor based on its tokenized expression. Suppose we have M candidate theorems.
Input: the chosen tactic or theorem $t \in \mathbb{R}^N$, the candidate theorems $X \in \mathbb{R}^{M \times N}$ and a hidden variable $h \in \mathbb{R}^N$.
Policy: $V_{\text{arg}} : \mathbb{R}^N \times \mathbb{R}^{M \times N} \times \mathbb{R}^N \to \mathbb{R}^N \times \mathbb{R}^M$
Initialize hidden variable h to t.
$l \leftarrow [t]$.

Loop for allowed length of arguments (e.g., 5):

$h, \mathbf{v} \leftarrow V_{\text{arg}}(t, X, h)$
$t \leftarrow$ sample from $\pi_{\text{arg}}(g) = \mathrm{Softmax}(\mathbf{v})$
$l \leftarrow l.\mathrm{append}(t)$

Return l and the associated (log) probabilities.

Generating actions

Given state s, we now have some (log) probabilities.

- $p(f|s)$ given by π_{fringe}.

Generating actions

Given state s, we now have some (log) probabilities.

- $p(f|s)$ given by π_{fringe}.
- $p(t|s,f)$ given by π_{tactic}.

Generating actions

Given state s, we now have some (log) probabilities.

- $p(f|s)$ given by π_{fringe}.
- $p(t|s, f)$ given by π_{tactic}.
- $p_0(c_0|s, f, t), ..., p_{l-1}(c_{l-1}|s, f, t, \mathbf{c}_{l-2})$ given by π_{arg}, where l is the length of arguments, and $\mathbf{c}_l = (c_0, ..., c_{l-1})$.

Generating actions

Given state s, we now have some (log) probabilities.

- $p(f|s)$ given by π_{fringe}.
- $p(t|s,f)$ given by π_{tactic}.
- $p_0(c_0|s,f,t), \ldots, p_{l-1}(c_{l-1}|s,f,t,\mathbf{c}_{l-2})$ given by π_{arg}, where l is the length of arguments, and $\mathbf{c}_l = (c_0, \ldots, c_{l-1})$.
- Let a be the chosen action. Now we have

$$\pi_\theta(a|s) = p(f|s)p(t|s,f)p_0(c_0|s,f,t)\prod_{i=1}^{l-1} p_i(c_i|s,f,t,\mathbf{c}_{i-1})$$

where θ is the parameters of $\{V_{\text{goal}}, V_{\text{tactic}}, V_{\text{arg}}\}$.

Baseline

REINFORCE(Williams (1988, 1992)):

We jointly train the policies:

$$\theta \leftarrow \theta + \alpha\gamma^t G_t \nabla_\theta \ln \pi_\theta(A_t|S_t)$$

given a trajectory $S_1, A_1, R_1, S_2, A_2, \ldots, S_T$.

Experiment with list

- 444 basic theorems from list theory.

Experiment with list

- 444 basic theorems from list theory.
- A small set of tactics.
  - `simp, fs, metis_tac, rw`
  - `irule, drule`
  - `Induct_on`
  - `strip_tac, EQ_TAC`

Experiment with list

- 444 basic theorems from list theory.
- A small set of tactics.
  - `simp`, `fs`, `metis_tac`, `rw`
  - `irule`, `drule`
  - `Induct_on`
  - `strip_tac`, `EQ_TAC`
- Only theorems that come before target g in library are allowed to be used to prove g.

Experiment with list

- 444 basic theorems from list theory.
- A small set of tactics.
  - `simp`, `fs`, `metis_tac`, `rw`
  - `irule`, `drule`
  - `Induct_on`
  - `strip_tac`, `EQ_TAC`
- Only theorems that come before target g in library are allowed to be used to prove g.
- A limited number of theorems are provable using this set of tactics (190$\sim$/443).

Preliminary results

	success/iter	success rate w.r.t total provable	success rate on validation
Random rollouts	42	21.2%	38.3%
Trained agent	149	75.3%	87.5%

Figure: An agent trained for 1000 iters performs significantly better than guessing. In each iteration, only one attempt for each theorem is allowed. There are 444 theorems in total and 198 of them are provable using the specified set of tactics. The validation set consists of equivalent forms of 20 easy theorems in the training set.

Preliminary results

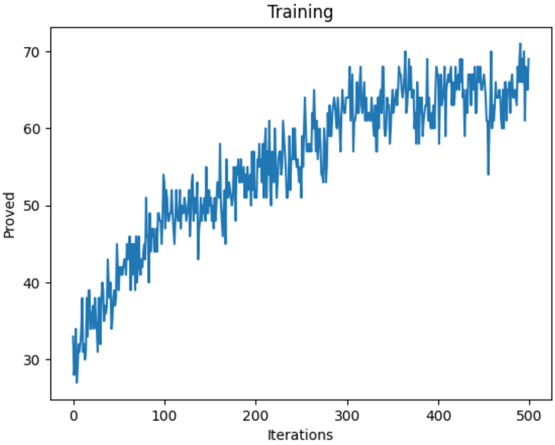

Figure: A typical training curve. In this experiment, the training set contains 87 theorems that are all provable. The performance of the agent keeps improving as training continues.