DT-Solver:
Automated Theorem Proving with Dynamic-Tree Sampling Guided by Proof-level Value Function

Presentation by Rahul Vishwakarma

DT-Solver: Automated Theorem Proving with Dynamic-Tree Sampling Guided by Proof-level Value Function

Haiming Wang[1*], Ye Yuan[2], Zhengying Liu[3] Jianhao Shen[2], Yichun Yin[3], Jing Xiong[1], Enze Xie[3], Han Shi[3], Yujun Li[3], Lin Li[3], Jian Yin[1†], Zhenguo Li[3], Xiaodan Liang[1,4†]

[1]Sun Yat-sen University, [2]Peking University, [3]Huawei Noah's Ark Lab, [4]MBZUAI

{wanghm39,xiongj69}@mail2.sysu.edu.cn, {yuanye_pku,jhshen}@pku.edu.cn,

{liuzhengying2,yinyichun,xie.enze,shi.han}@huawei.com

{liyujun9,lilin29,Li.Zhenguo}@huawei.com

issjyin@mail.sysu.edu.cn, xdliang328@gmail.com

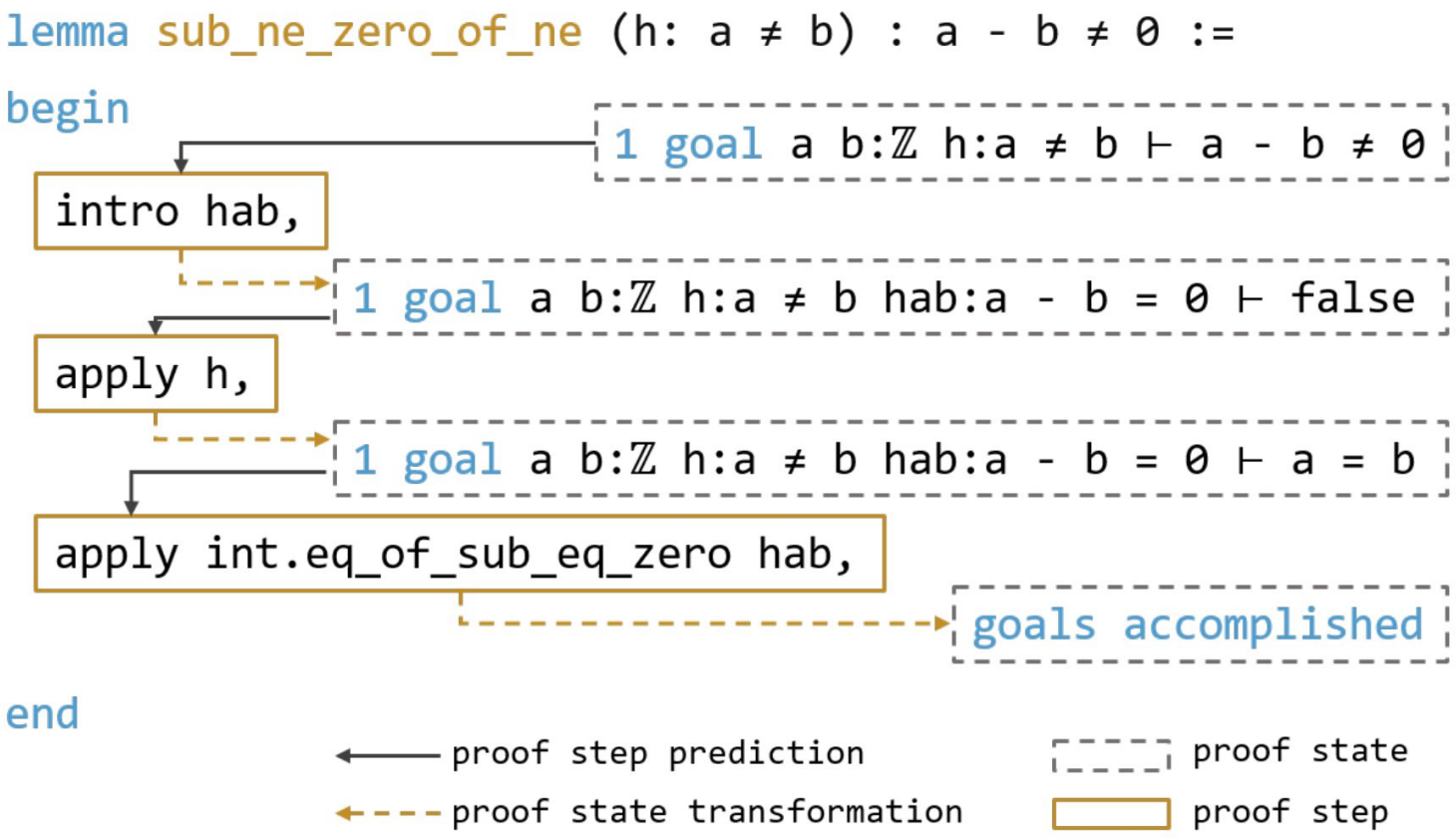
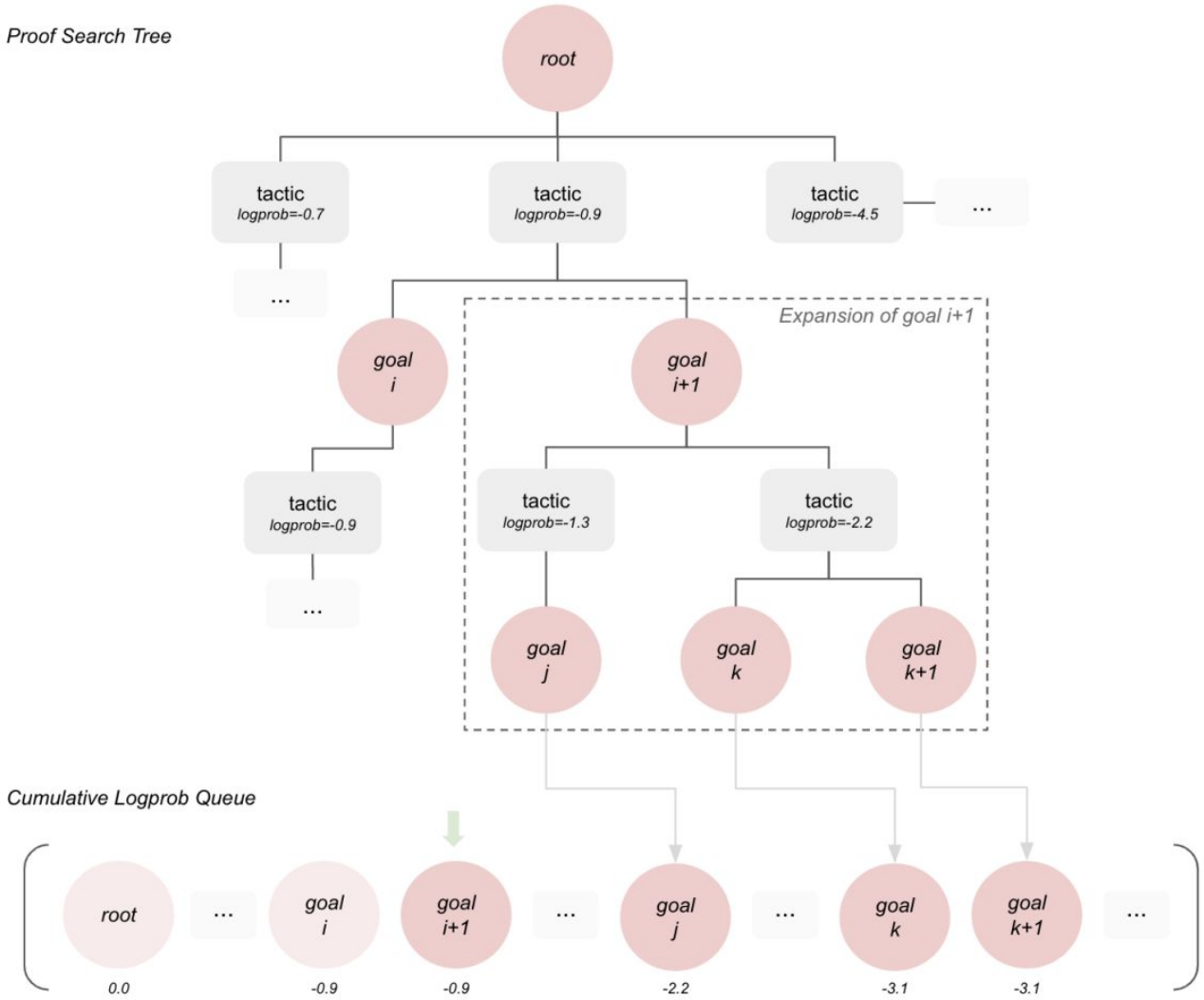


Illustration of a theorem and its proof in Lean

Drawbacks of GPT-f like approaches:

- High Computational Resources:
  a. GPT-f demands significant computational resources for training, with original GPT-f and HTPS consuming 2000 and 1000+ GPU days respectively, making it impractical for many researchers.
  a. Inefficient Inference: Inference with GPT models is also resource-intensive, posing challenges for real-time applications.
- Issues in Search Process:
  a. 'Empty Queue' State: GPT-f's search process can encounter an 'empty queue' state, where all proof step candidates generated by the model become inapplicable, leading to premature 'Failed' results and low pass rates.
  b. Inadequate Exploration: Previous approaches often treat all proving states uniformly, neglecting the need for more exploration in difficult states, and lack holistic evaluation and guidance for proof search trajectory.


Proof Search Tree
root
tactic
logprob=-0.7
tactic
logprob=-0.9
tactic
logprob=-4.5
...
...
Expansion of goal i+1
goal
i
goal
i+1
tactic
logprob=-0.9
tactic
logprob=-1.3
tactic
logprob=-2.2
...
goal
j
goal
k
goal
k+1
Cumulative Logprob Queue
root
...
goal
i
goal
i+1
...
goal
j
...
goal
k
goal
k+1
...
0.0
-0.9
-0.9
-2.2
-3.1
-3.1

Addressing Challenges with Dynamic-Tree Theorem Solver

- Dynamic-Tree Sampling:
  - Issue: Premature 'empty queue' failure in proof search.
  - Solution: Implement dynamic-tree sampling allowing nodes to be expanded multiple times, producing new child nodes using language model predictions.

- Proof-Level Value Function:
  - Issue: Previous methods (GPT-f and HTPS) use step-level value functions.
  - Solution: RoBERTa-based proof-level value function assesses entire proof trajectory, aiding the search for promising proof states.

Language Model-Guided Theorem Proving (Step 1)

- Recent approaches utilizes language models like GPT, to predict applicable proof steps based on current proof state.
- Language model trained with language modeling objective on sequences of the form: GOAL $(proof state) PROOFSTEP $(proof step).
- Language model learns to predict proof steps given a specific proof state.
- At test time, multiple new proof steps are sampled from the language model, expanding the proof search.
- GPT-f employs a best-first search algorithm to construct a complete proof for a given theorem.
- Iterates through selecting the best-scoring proof state from a priority queue, performing expansion using the language model, and adding new states to the queue.
- Each state's score calculated using a value function.
- Iterations continue and theorem proving terminates upon finding a proof or reaching computational limits.

Methodology of DT-Solver for Theorem Proving

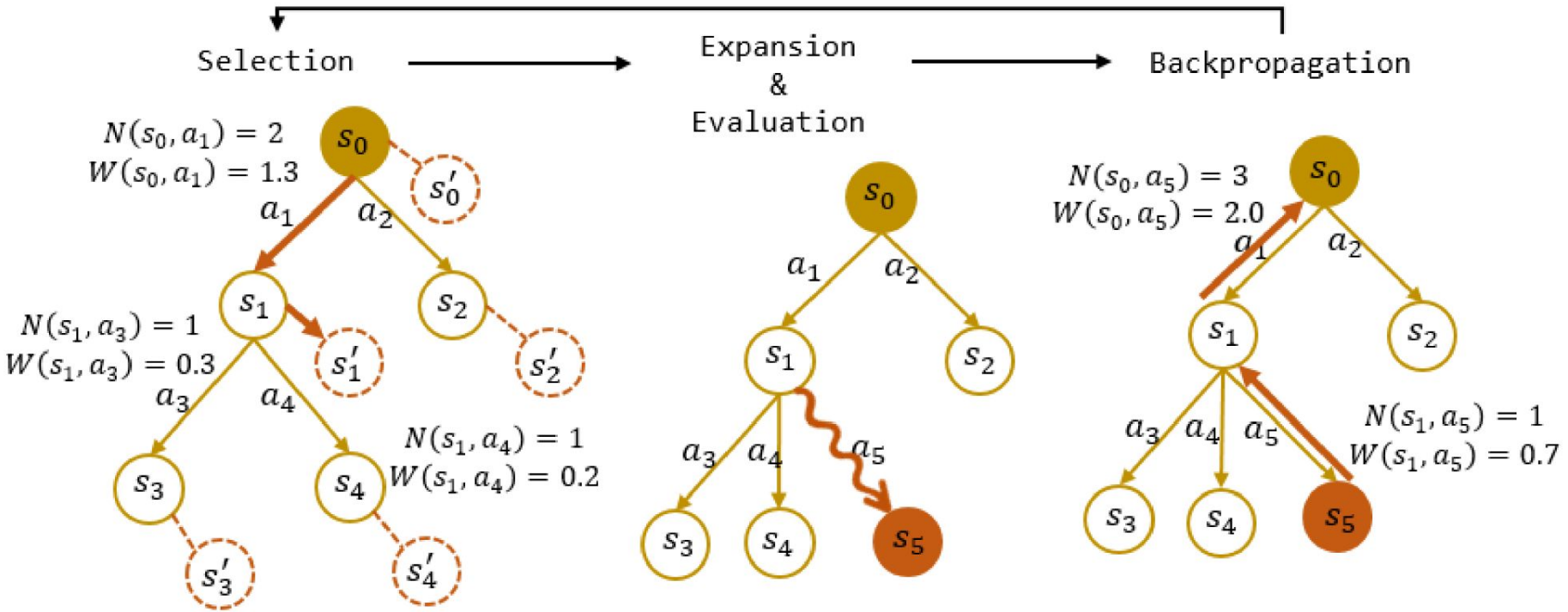
- DT-Solver begins with an initial theorem state and leverages the language model to suggest forward-moving actions.
- Empowers dynamic-tree sampling guided by a proof-level value function to explore and identify the complete proof path.
- **Step 1:** Training the Policy Model - DT-Solver trains a language model using supervised data from formal mathematics libraries. This model predicts proof steps based on proof states.
- **Step 2:** Dynamic-Tree Sampling - The trained policy model facilitates dynamic-tree sampling to generate steps and search for theorem proofs in the training set. A data collection procedure captures proof trajectories.
- **Step 3:** Training the Critic Model - DT-Solver trains the proof-level value function using collected data. The critic model distinguishes promising proof paths from less effective ones.
- During evaluation, DT-Solver employs the trained policy model and critic model.
- DT-Solver's multi-step approach enables efficient proof path exploration.

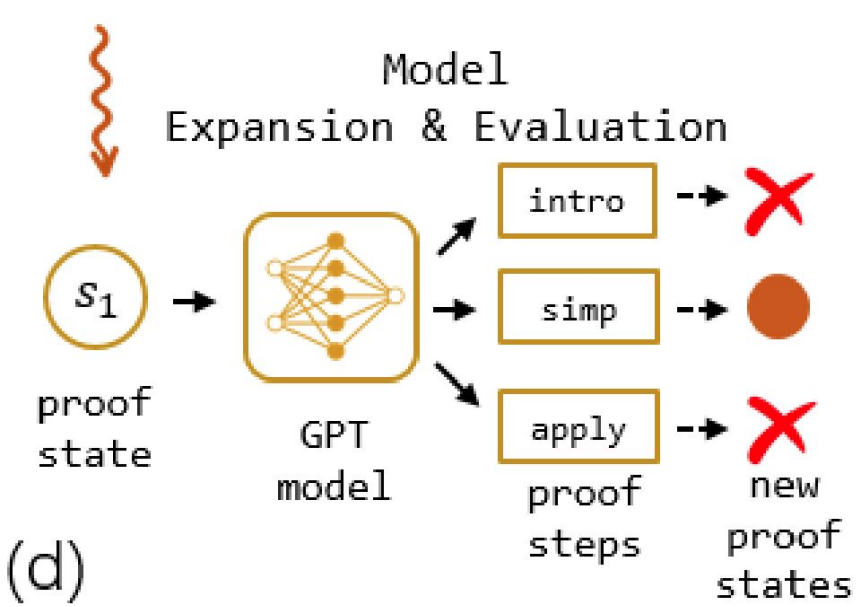
Dynamic-tree sampling (Step 2)

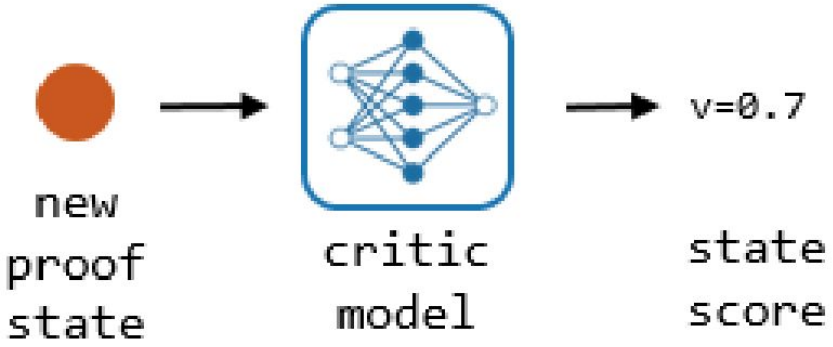
- The dynamic-tree sampling algorithm strives to maximize the efficiency of the search procedure under limited computational budgets.
- In practice, it controls the exploration of different proof states according to two criteria: state value and model confidence.
- The state value estimates whether the state is on the correct proof path. Only high-valued states deserve more exploration.
- The model confidence is calculated with the prior probabilities of the current state’s proof step.
- A state is considered to require more exploration if the policy model cannot produce confident steps toward the current state.

Dynamic-tree sampling (Step 2)

- To achieve the above objectives, authors build the dynamic-tree sampling based on the Monte-Carlo tree search algorithm.
- Dynamic tree sampling progressively adds new proof states to construct a proof tree.
- Specifically, each node in the proof tree represents a proof state denoted as $s_i \in \{s_0, s_1..., s_M\}$, where M is the total number of tree nodes.
- Each edge represents a proof step denoted as $a_j \in \{a_0, a_1..., a_{M-1}\}$, where $M - 1$ is the total number of edges.
- Similar to the classic Monte-Carlo tree search algorithm, every node-edge pair in the proof tree has a visit count denoted as $N(s_i, a_j)$ and a value count denoted as $W(s_i, a_j)$.
- Starting from the root node, dynamic tree sampling repeats the (i) Selection, (ii) Expansion & Evaluation and (iii) Backpropagation steps until the theorem's proof is found or the computational budget is exhausted.


Selection
Expansion
&
Evaluation
Backpropagation
$N(s_0, a_1) = 2$
$W(s_0, a_1) = 1.3$
$N(s_1, a_3) = 1$
$W(s_1, a_3) = 0.3$
$N(s_1, a_4) = 1$
$W(s_1, a_4) = 0.2$
$N(s_0, a_5) = 3$
$W(s_0, a_5) = 2.0$
$N(s_1, a_5) = 1$
$W(s_1, a_5) = 0.7$
s_0
s_1
s_2
s_3
s_4
s_5
s_0'
s_1'
s_2'
s_3'
s_4'
a_1
a_2
a_3
a_4
a_5


Model
Expansion & Evaluation
s_1
proof
state
GPT
model
intro
simp
apply
proof
steps
new
proof
states
(d)


v=0.7
new
proof
state
critic
model
state
score

Selection, Expansion & Evaluation and Backpropagation

- Dynamic-tree sampling computes the PUCT (Positional Upper Confidence for Trees) score for child nodes and the virtual child node.
- Select the highest-scoring child node and proceed as:
  - if a non-virtual child node is selected, dynamic-tree sampling continues the selection phrase from the selected child node.
  - If a virtual node is selected, the selection phase ends and proceeds to expansion & evaluation.

Selection

- Current node to perform selection is denoted as s_t, and its children as $s_c \in C(s_t)$ where $C(\cdot)$ is the function that returns all the children for a given node. The virtual child of s_t is denoted as s'_t
- The PUCT score for each child node $s_c \in C(s_t)$ is formulated as

$$\text{PUCT}_{s_c} = \frac{W(s_t, a_c)}{N(s_t, a_c)} + c \cdot p(a_c|s_t) \cdot \frac{\sqrt{N(s_t, \cdot)}}{N(s_t, a_c)}$$

- Where c is a constant balancing the exploration and exploitation trade-off, and $p(a_c|s_t)$ is the probability (estimated by the language model) of generating the proof step a_c given current state s_t.

Selection, **Expansion & Evaluation** and Backpropagation

- Dynamic-tree sampling performs expansion on the parent node of the selected virtual node.
- A fixed number (denoted as e) of proof steps are sampled from the trained policy model.
- The formal environment verifies proof steps and produces new proof states, which are then deduplicated and evaluated by the critic model, and finally added to the tree.

Selection, Expansion & Evaluation and **Backpropagation**

- Dynamic-tree sampling backpropagates new state scores to the root state by successively adding the score to the parent's value count W and accumulating visit count N.
- The backpropagation follows the classical Monte-Carlo tree search.
- We denote the function that returns the node's parent as $P(\cdot)$.
- The value counts $W(P(s_c), a_c)$ and visit counts $N(P(s_c), a_c)$ are both initialized as 0 for newly add child node.
- Given the newly added leaf state s_c and its estimated score v_c, dynamic-tree sampling repeats the following steps until the root node is reached:
  - Accumulate the visit count $N(P(s_c), a_c)$ += 1, and accumulate the value count $W(P(s_c), a_c)$ += v_c
  - Traverse back the tree by changing the current node to the parent $s_c \Leftarrow P(s_c)$.

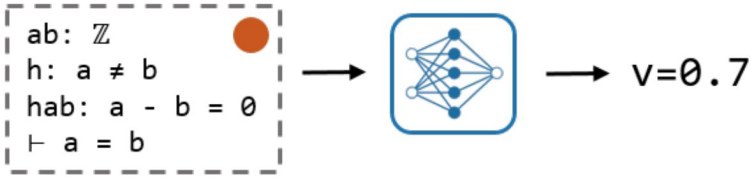
Data collection

- To construct a proof-level value function capable of identifying promising states from astray ones, DT-Solver collects supervised training data by performing dynamic-tree sampling in training set theorems.
- Specifically, DT-Solver collects trajectories data in the form of ([(s_0, a_0), (s_1, a_1), ..., (s_l, a_l)], y), where s_0 is the root state and s_l is the leaf state.
- The label y = 1 if the trajectory correctly proves the theorem and y = 0 otherwise.

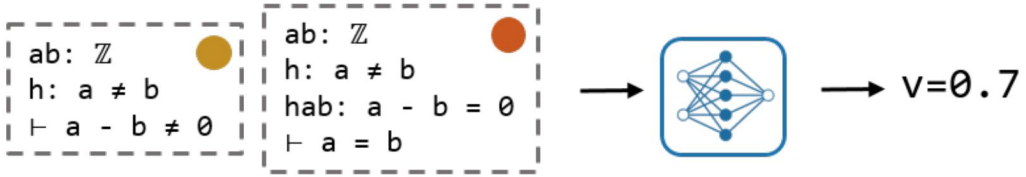
Proof-level value function

- The performance and efficiency of the dynamic tree sampling algorithm depend heavily on correctly evaluated state values.
- This paper proposes four types of proof-level value functions.
- Current-state only value function construct supervised data formatted as (s_l, y) from the collected trajectory data.
- Root-state-and-current-state value function constructs sentence pair data formatted as ([s_0, s_l], y) from the collected trajectory.
- Conversely, previous-state-to current-state uses more recent state information and formats the training data as ([s_{l-1}, s_l], y).
- Entire trajectory value function concatenates the entire proof path as follows: **GOAL $($s_0$) PROOFSTEP $($a_0$) </s></s> GOAL $($s_1$) PROOFSTEP $($a_1$) …** where the </s></s> separate different state action pair.

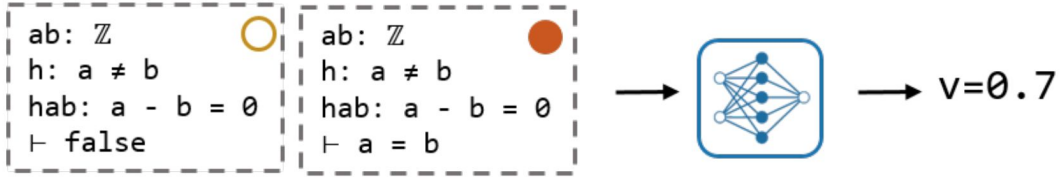
Current state only:



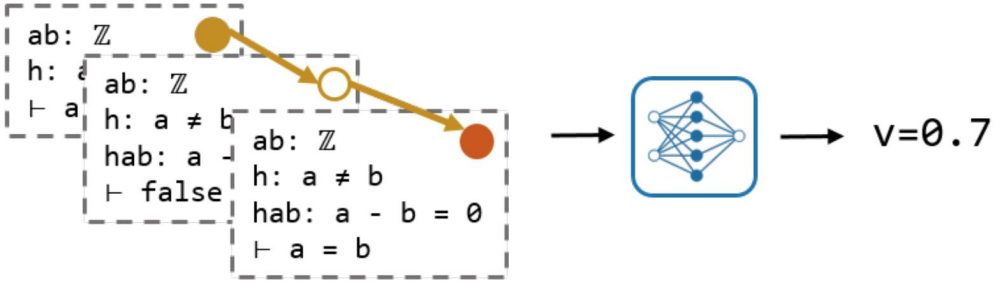
Root state and current state:



Previous state to current state:



Entire trajectory:



Four types of proof-level value functions

Thank You