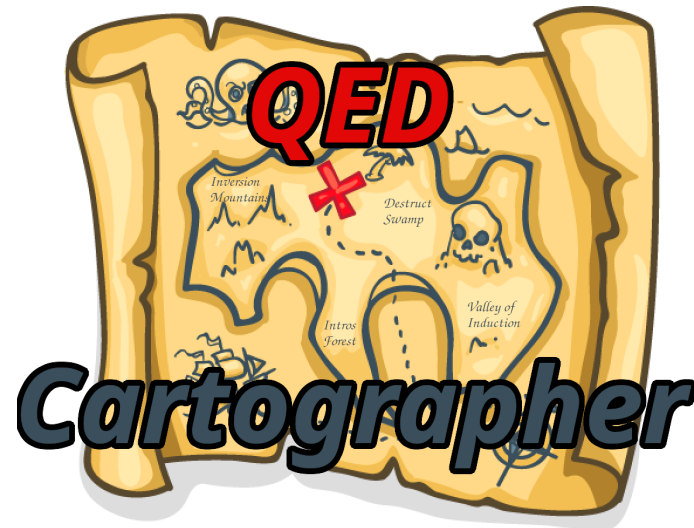


Alex Sanchez-Stern, Abhishek Varghese, Zhanna Kaufman,
Dylan Zhang, Talia Ringer, Yuriy Brun



Automating **Formal Verification** with
Reward-Free Reinforcement Learning

Formal Verification

writing proofs about programs!

more reliability
than testing!



labor
intensive

requires specialized
expertise

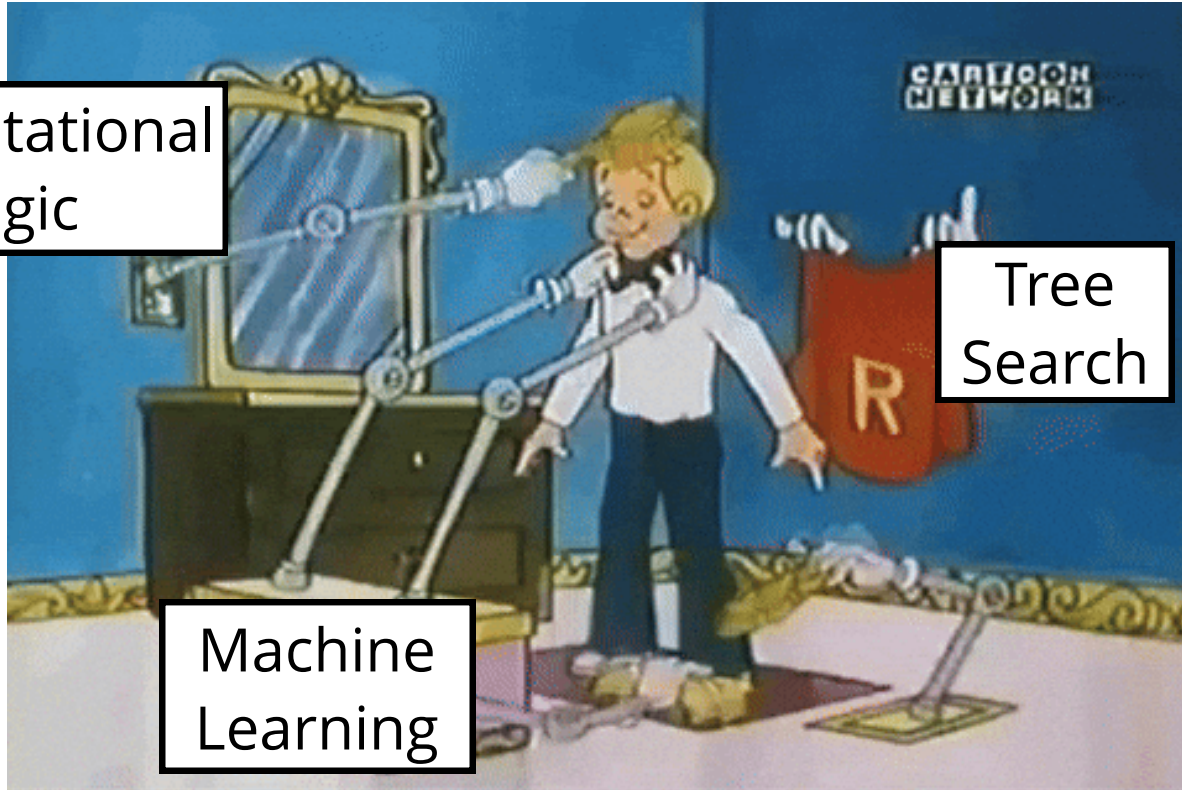


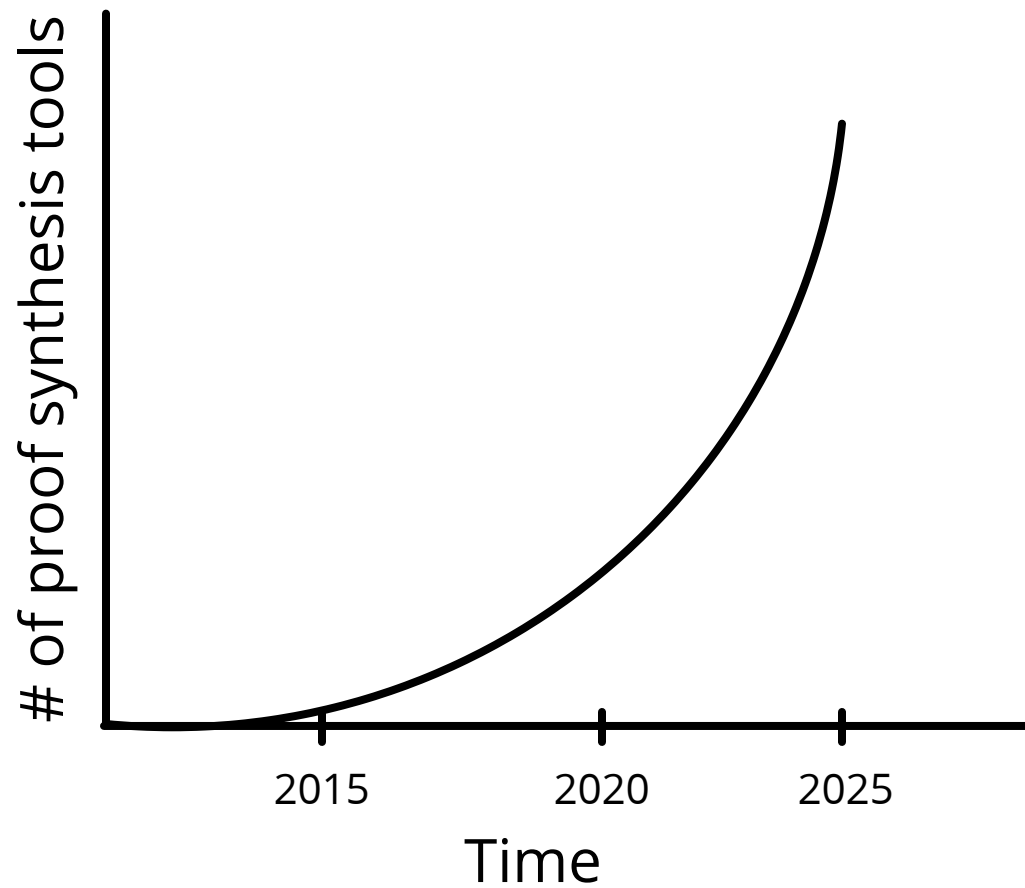
Automating Formal Verification

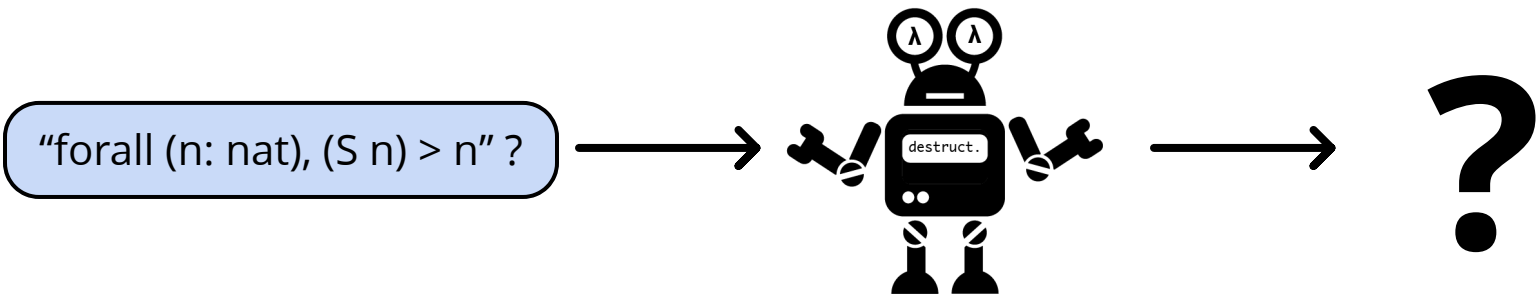
Computational
Logic

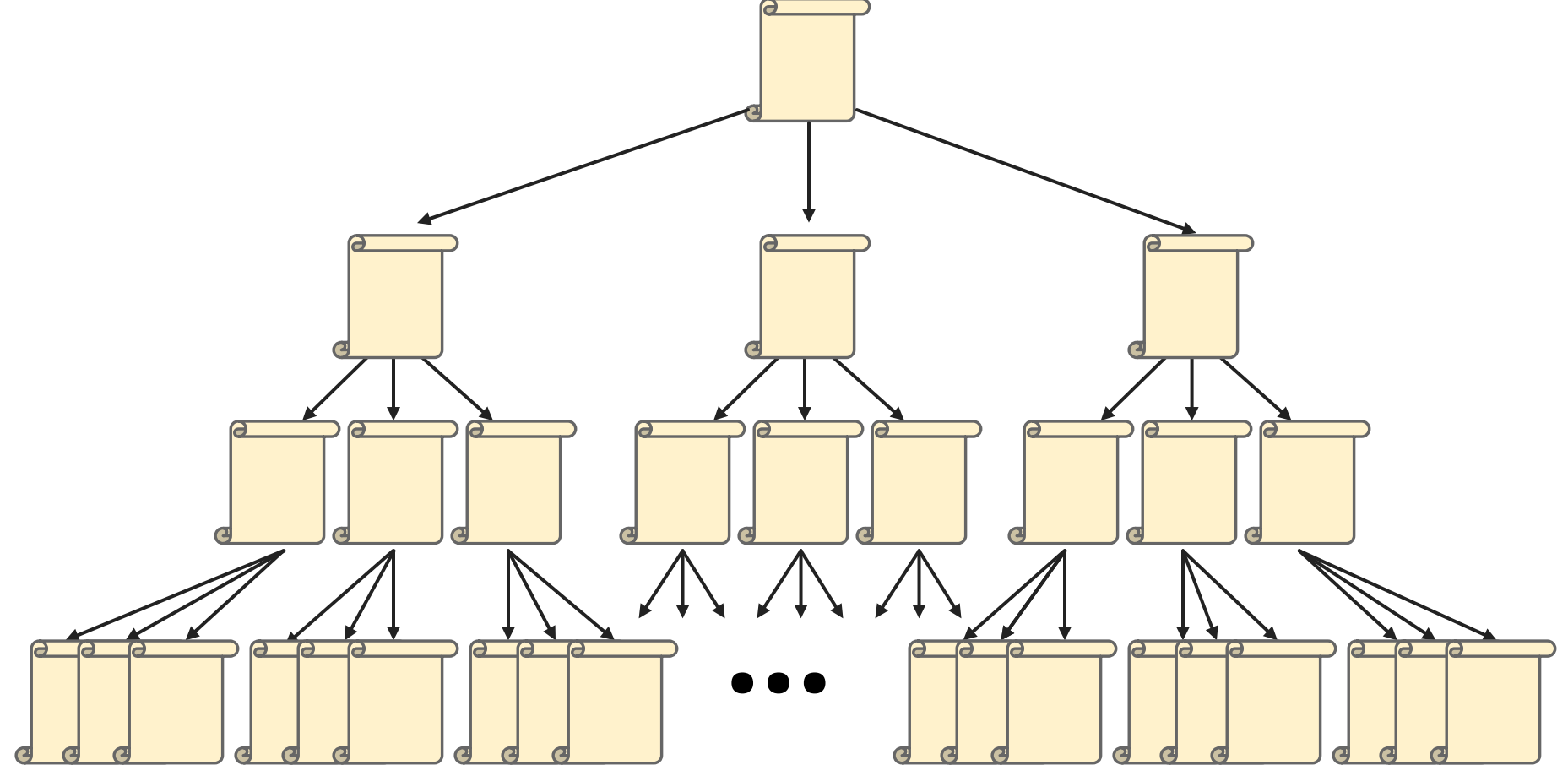
Tree
Search

Machine
Learning







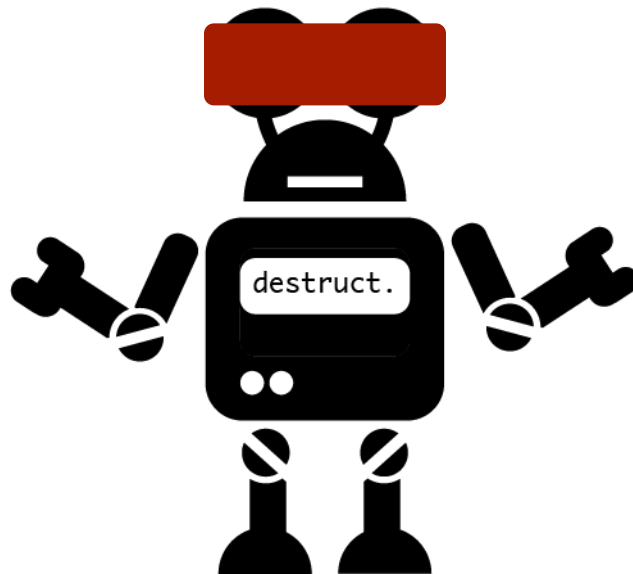


Unbounded Search Space

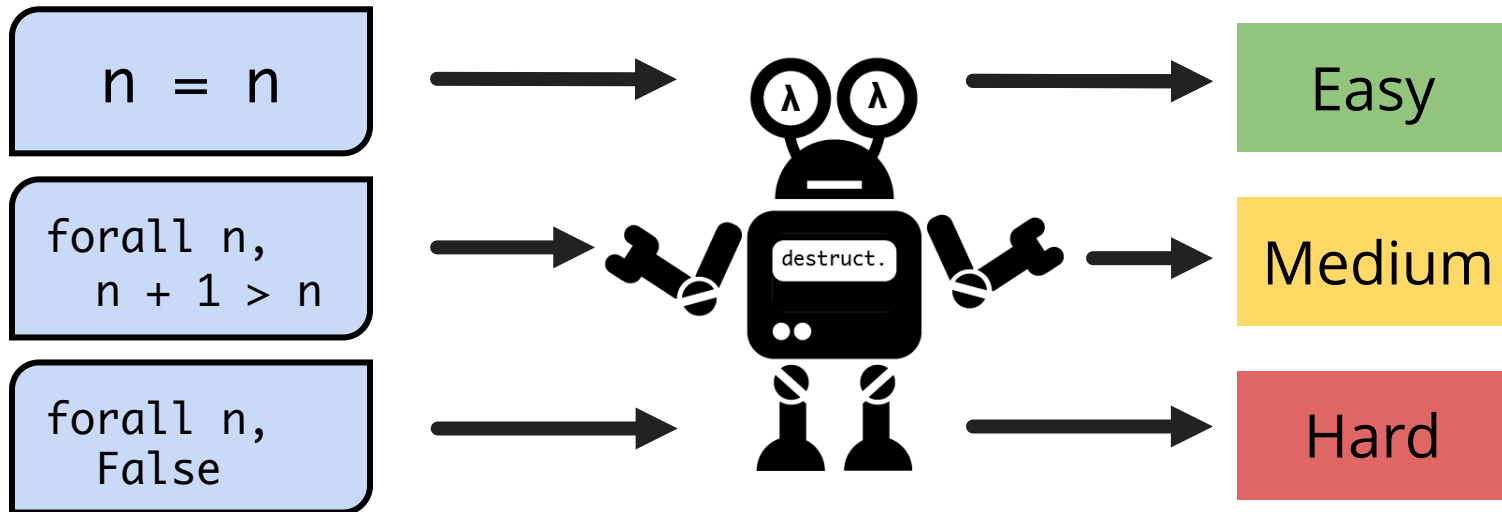
There are some techniques that can
help us prune the search tree

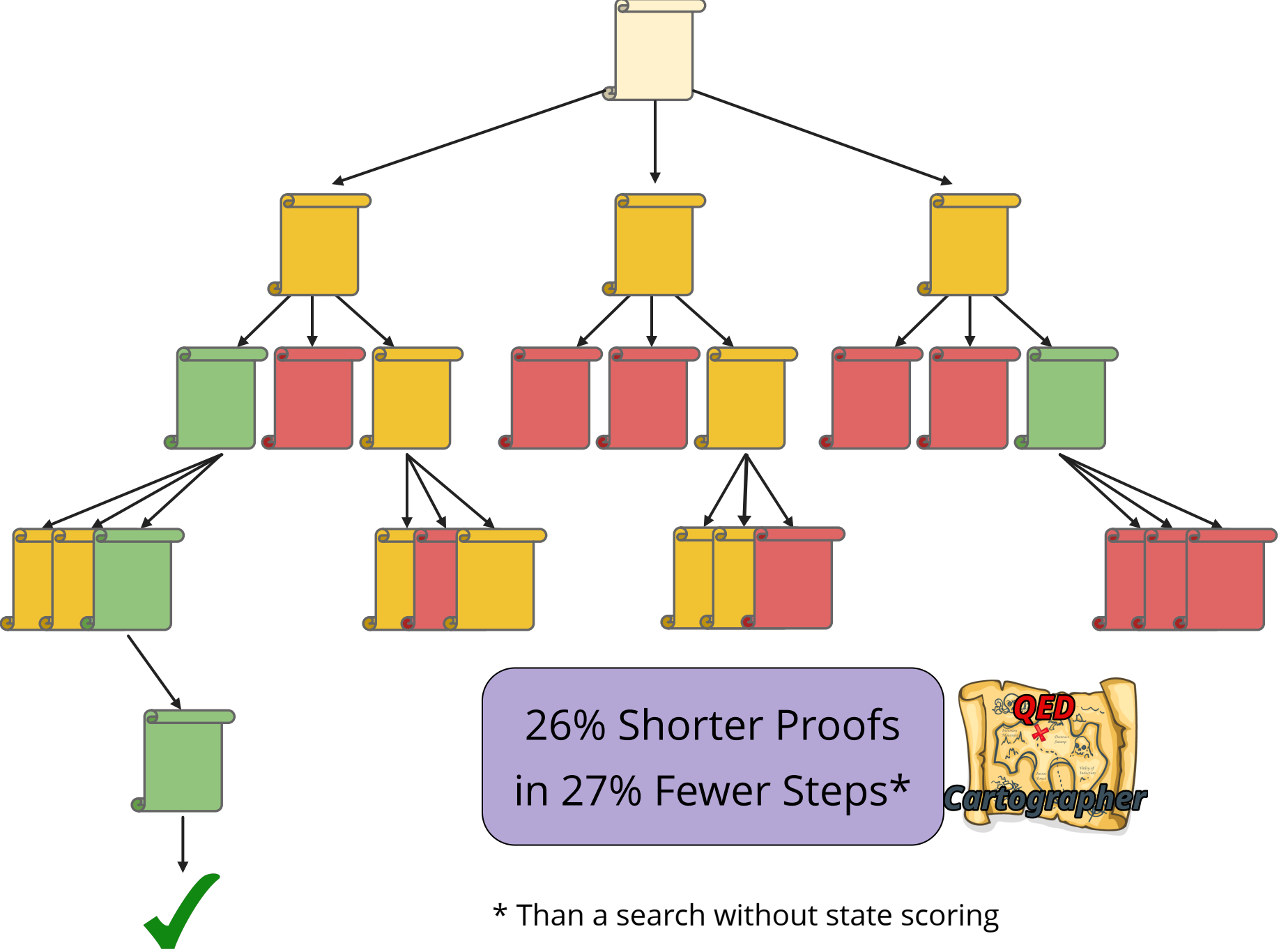


It's hard to explore when you don't know
where you are!



We can search more
efficiently if we can evaluate
proof states

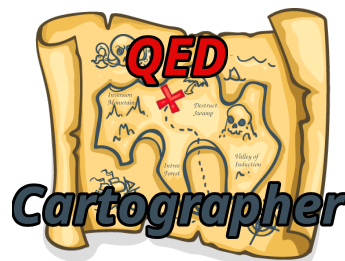
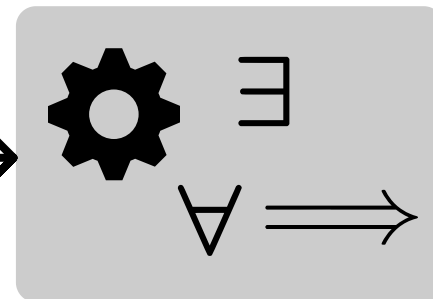
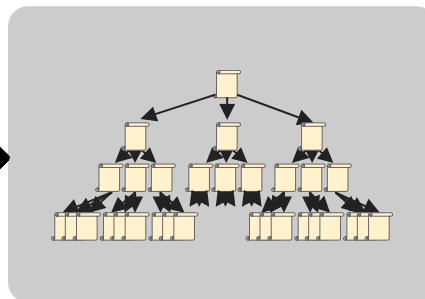
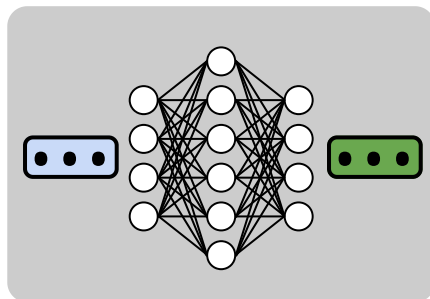




Predictor

Search

Theorem Prover



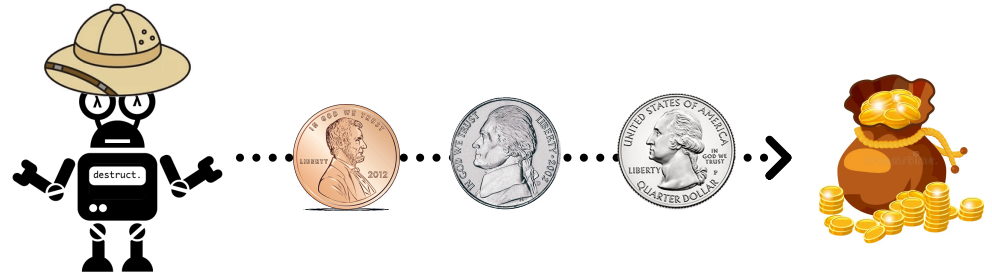
"Reward-free Reinforcement Learning"



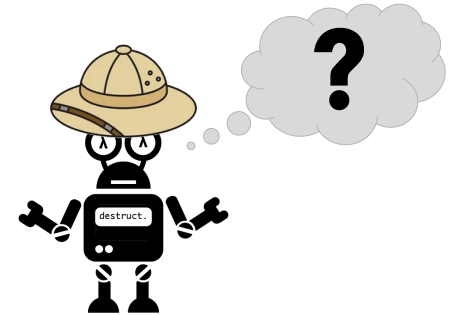
In particular, V-learning

This Talk

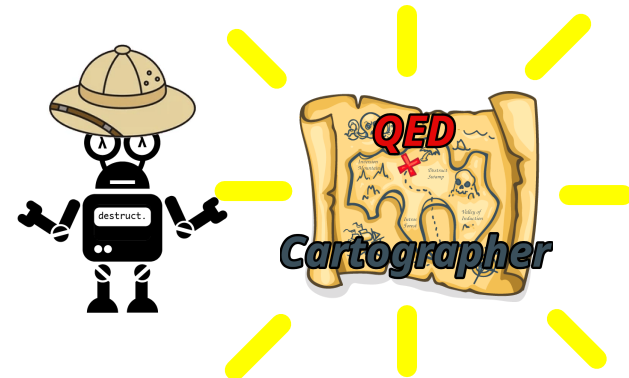
V-Learning



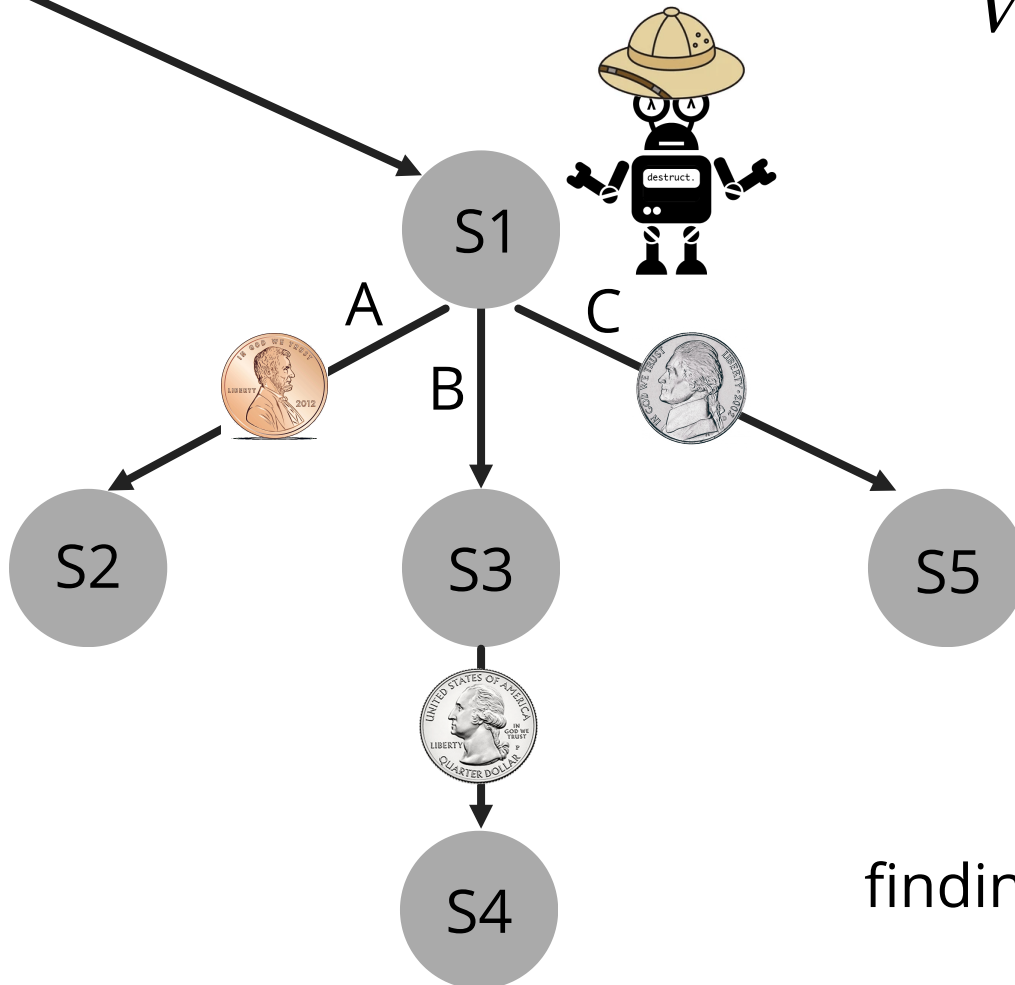
Limitations in Proofs



Adapting to Proofs



Classic V-Learning



$$V(S1) = \max($$

$$\begin{aligned} & \text{penny} + \gamma V(S2), \\ & \gamma(\text{quarter} + \gamma V(S4)), \\ & \text{nickel} + \gamma V(S5)) \end{aligned}$$

γ = time discount

finding rewards sooner is better!

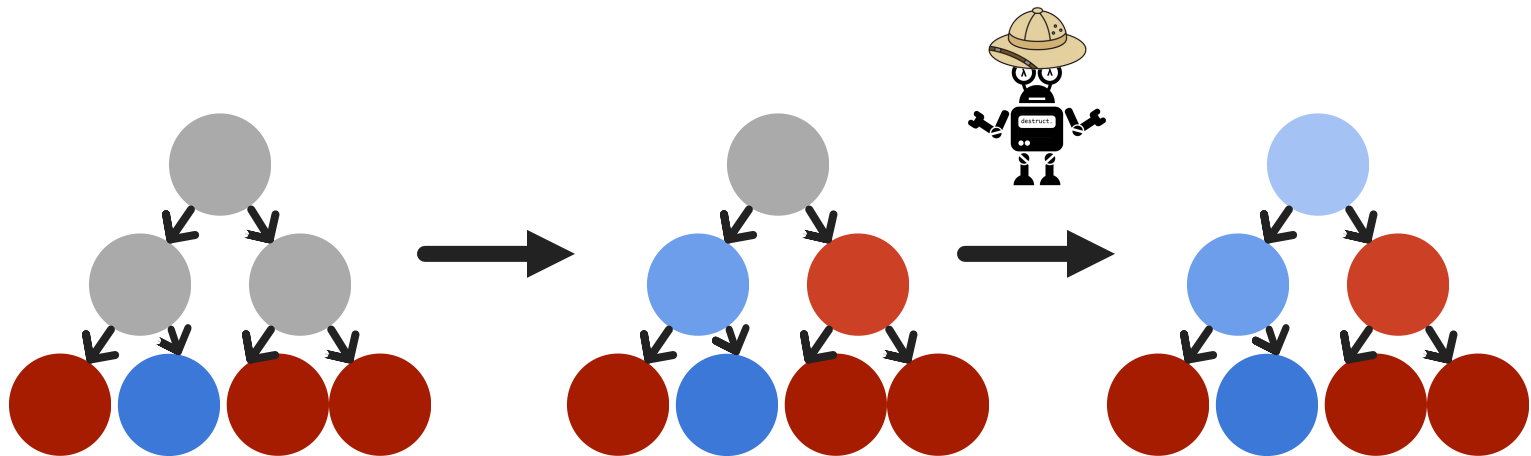
$$V(S1) = \max(\text{penny} + \gamma V(S2), \gamma (\text{quarter}) + \gamma V(S4), \text{nickel} + \gamma V(S5))$$

Generalizes to



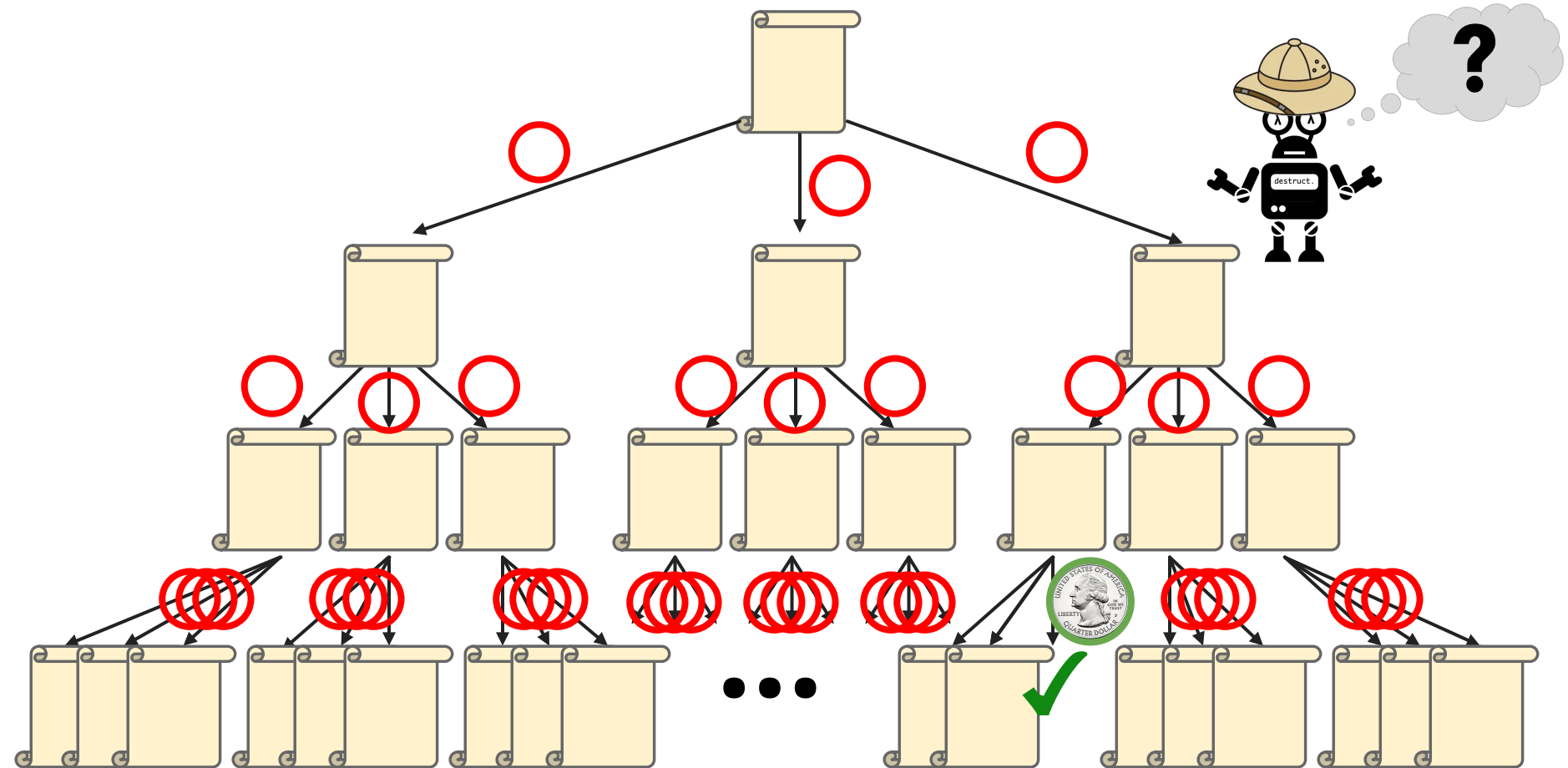
$$V(S) = \max_{a \in \text{actions}(S)} (R(S, a) + \gamma V(\text{next-state}(S, a)))$$

V-Learning in Practice: Iterative Updates



V-Learning in Proofs:

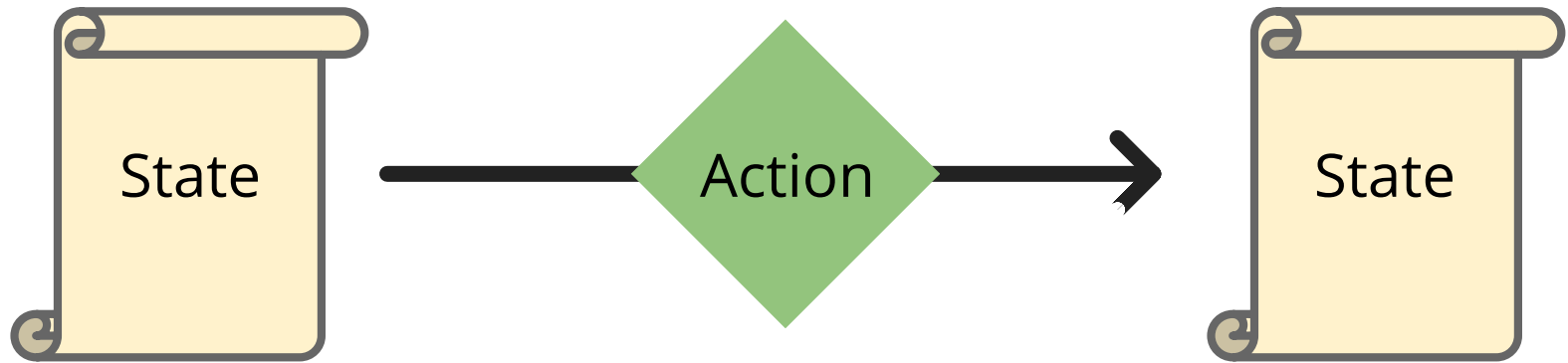
The Sparse Reward Problem

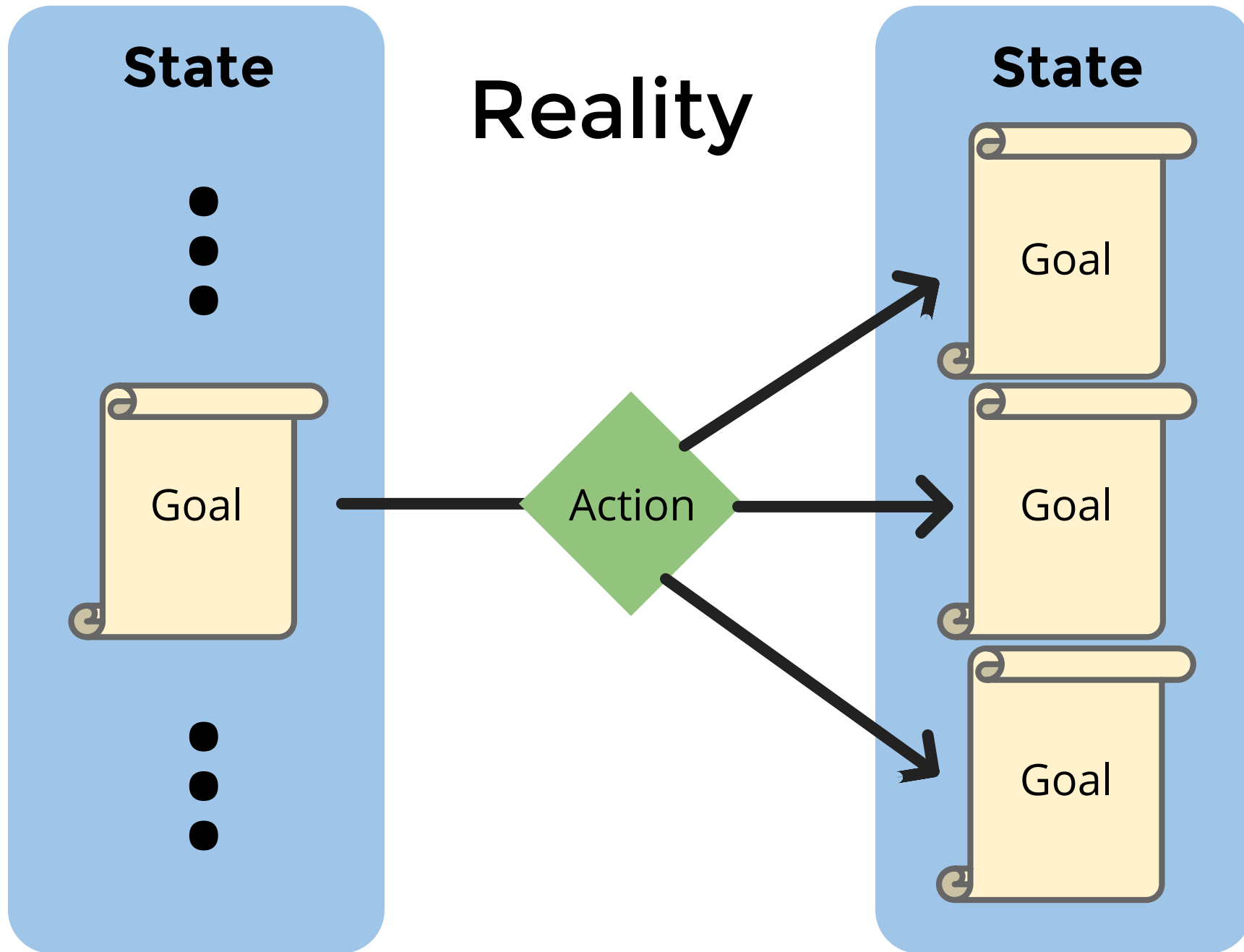


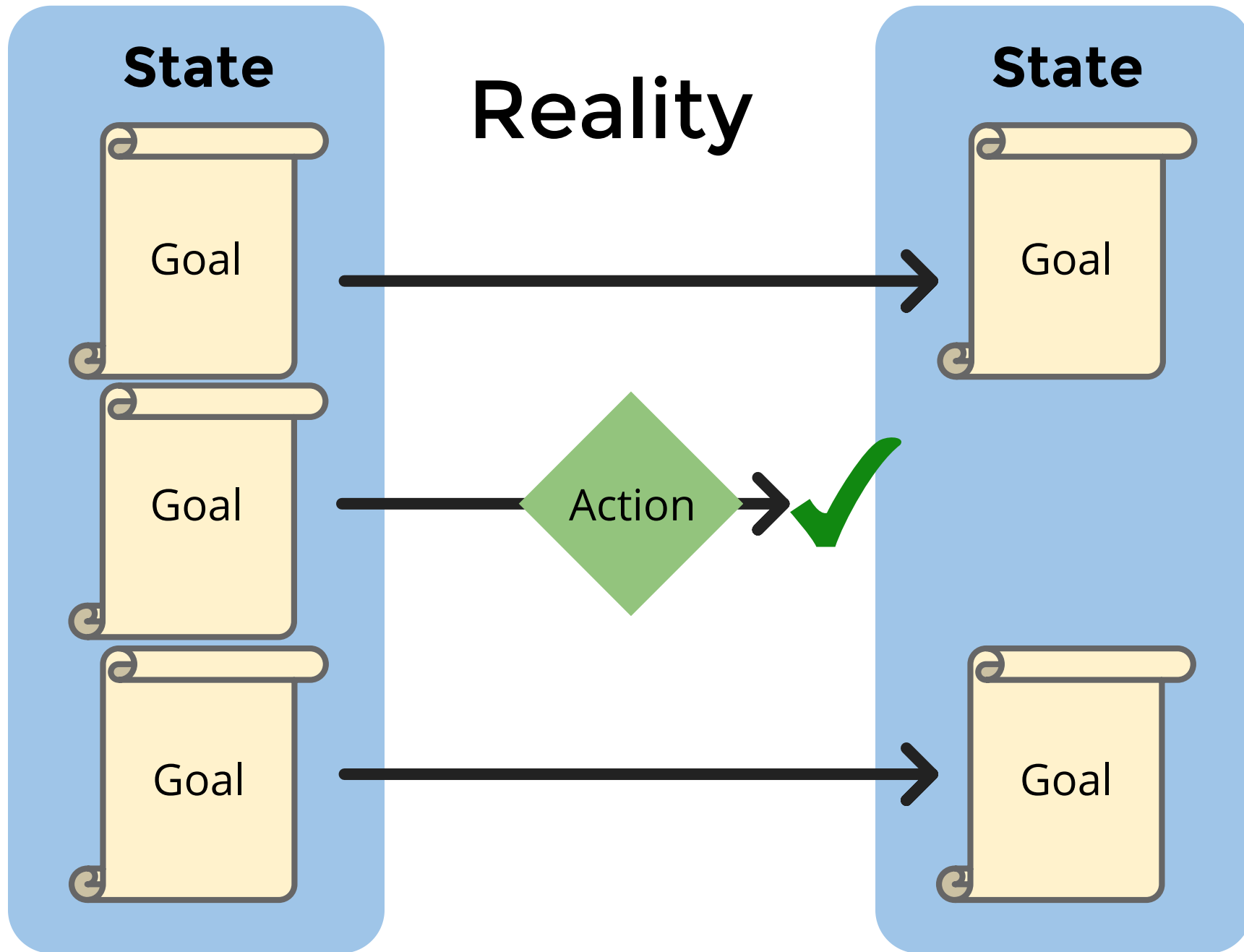
Insight:

Proofs have Useful Structure

Abstraction

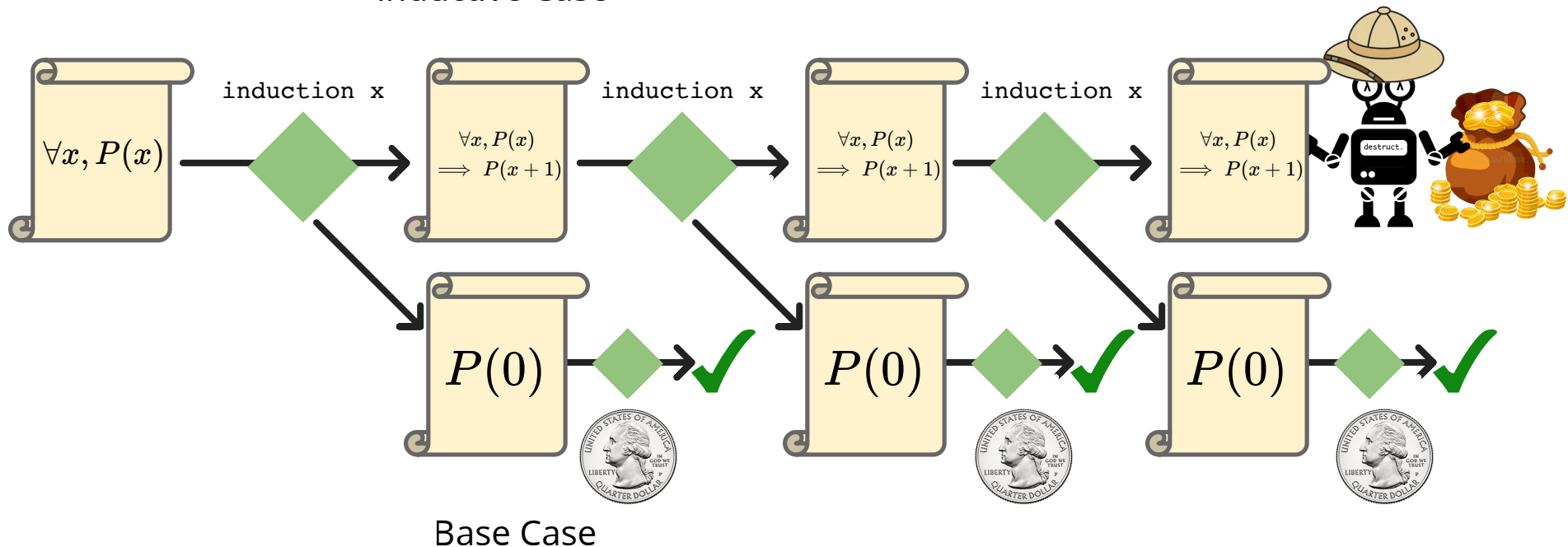






Adding extra rewards can lead to bad behavior

Inductive Case



Reward-free doesn't have this problem!

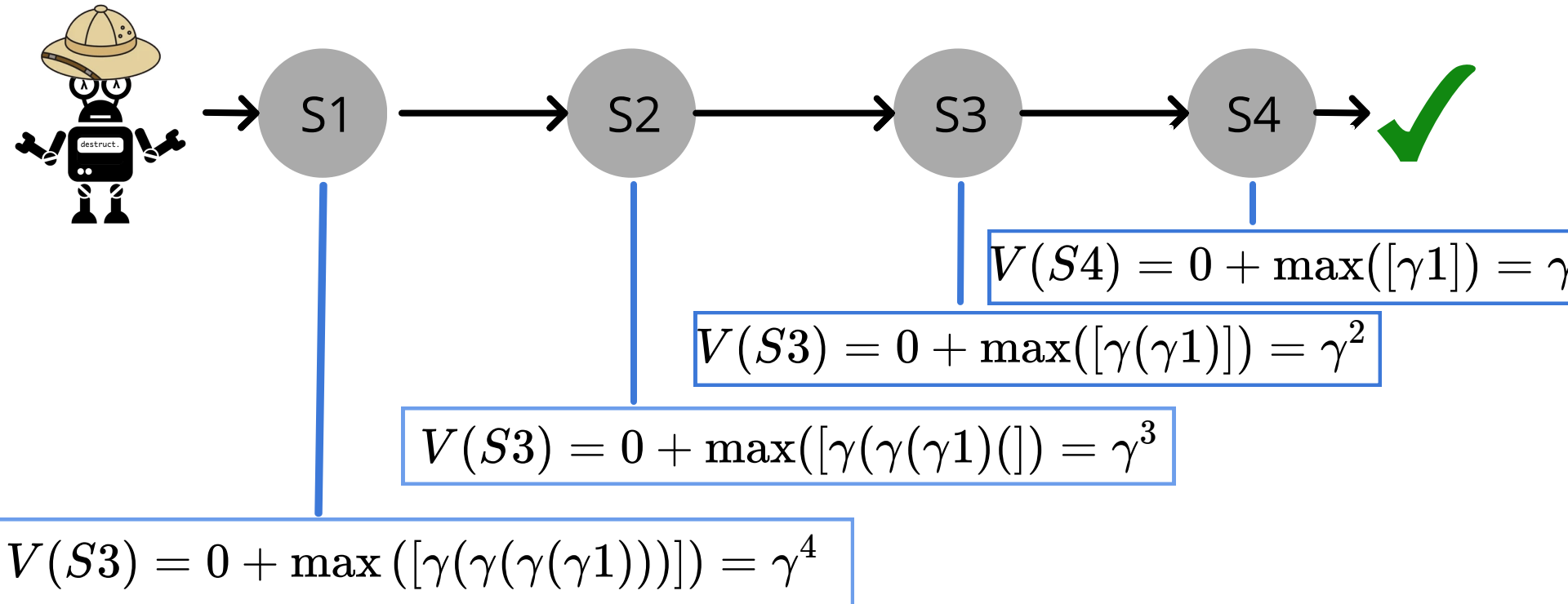
What We Need

A new update equation that accounts for
the branching structure of proofs

Assumptions:

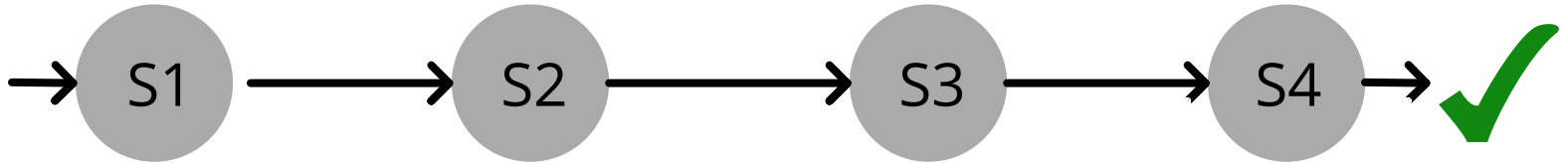
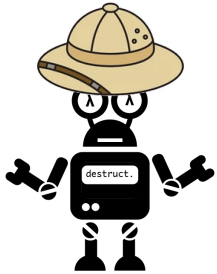
- The state of a completed proof has value 1

$$V(\checkmark) = 1$$

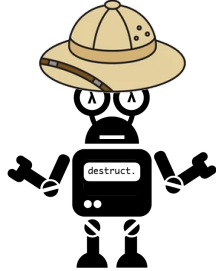


Assumption: The state of a completed proof
has value 1

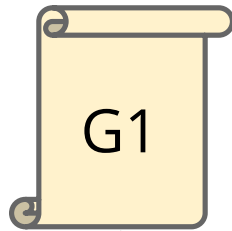
$$V(\checkmark) = 1$$



$$V(S) = \gamma^{(\text{number of steps left})}$$



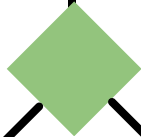
$$V(G1) = ???$$



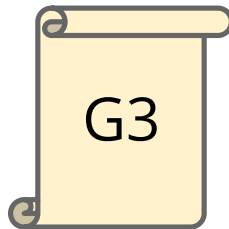
m = Steps to complete proof from G2

n = Steps to complete proof from G3

Steps to complete proof from G1 = $m + n + 1$



$$V(G1) = \gamma * V(G2) * V(G3)$$



Update Equation for Branch-Structured Proofs

$$V(G) = \boxed{\max_{a \in \text{actions}(G)}} \left(\gamma \prod_{G' \in \text{next-states}(G,a)} V(G') \right)$$

Update Equation for Branch-Structured Proofs

$$V(G) = \max_{a \in \text{actions}(G)} \left(\gamma \prod_{G' \in \text{next-states}(G,a)} V(G') \right)$$

Update Equation for Branch-Structured Proofs

$$V(G) = \max_{a \in \text{actions}(G)} \left(\gamma \boxed{\prod_{G' \in \text{next-states}(G,a)} V(G')} \right)$$

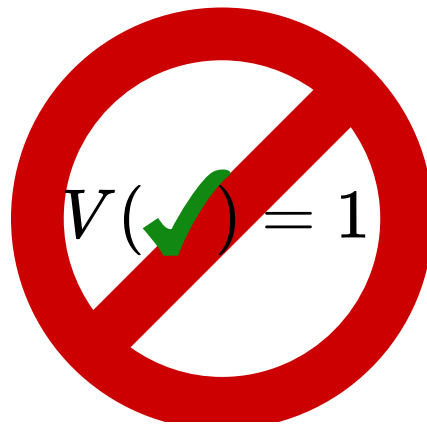
Update Equation for Branch-Structured Proofs

$$V(G) = \max_{a \in \text{actions}(G)} \left(\gamma \prod_{G' \in \text{next-states}(G,a)} V(G') \right)$$

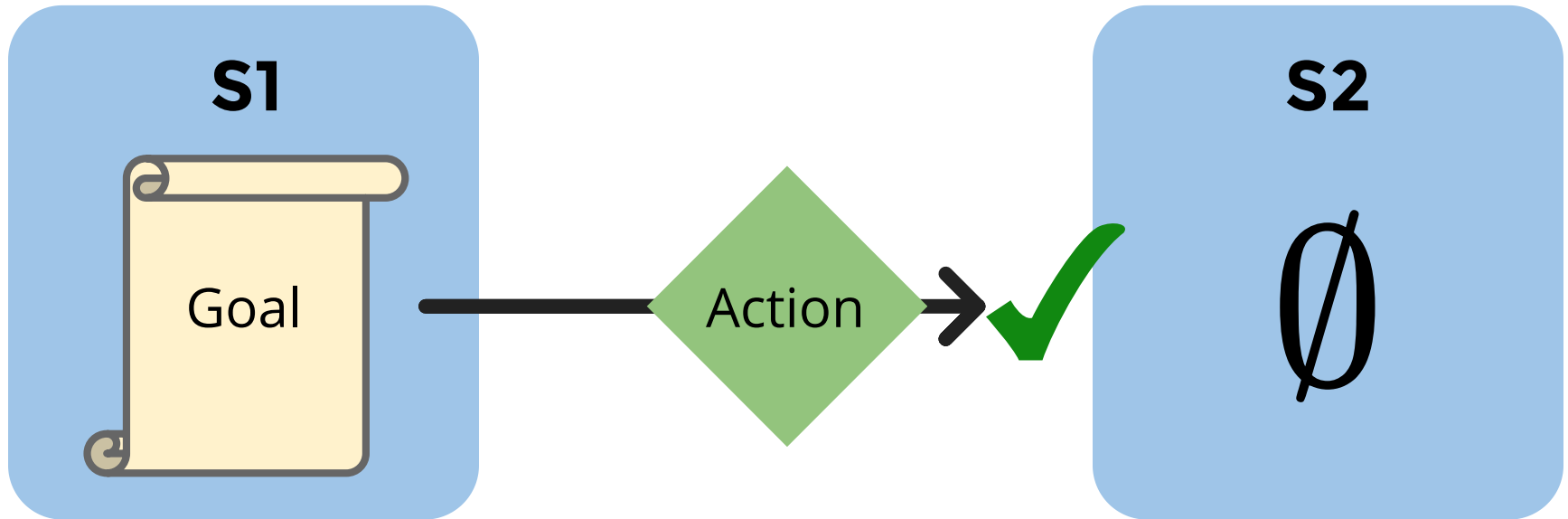
Update Equation for Branch-Structured Proofs

$$V(G) = \max_{a \in \text{actions}(G)} \left(\gamma \prod_{G' \in \text{next-states}(G, a)} V(G') \right)$$

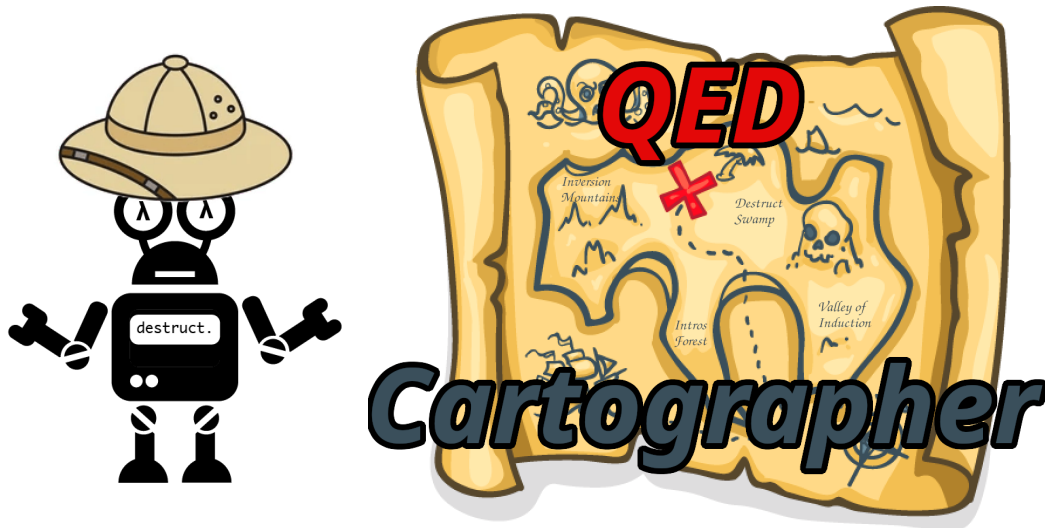
Where's the Reward?



Don't need one!



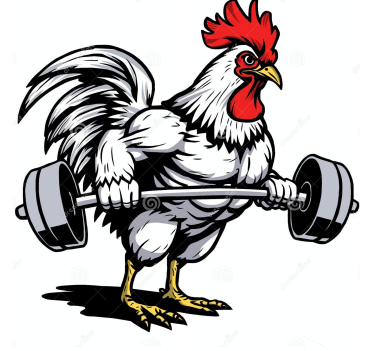
$$V(G) = \max_{a \in \text{actions}(G)} \left(\gamma \prod_{G' \in \text{next-states}(G, a)} V(G') \right)$$
$$\gamma \prod_{G' \in \emptyset} V(G')$$
$$\gamma(1)$$



Automating Formal Verification with
Reward-Free Reinforcement Learning

26% Shorter Proofs
in 27% Fewer Steps

Benchmark: CoqGym



124 Coq Projects

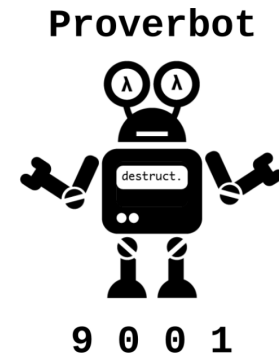
68,501 Theorems

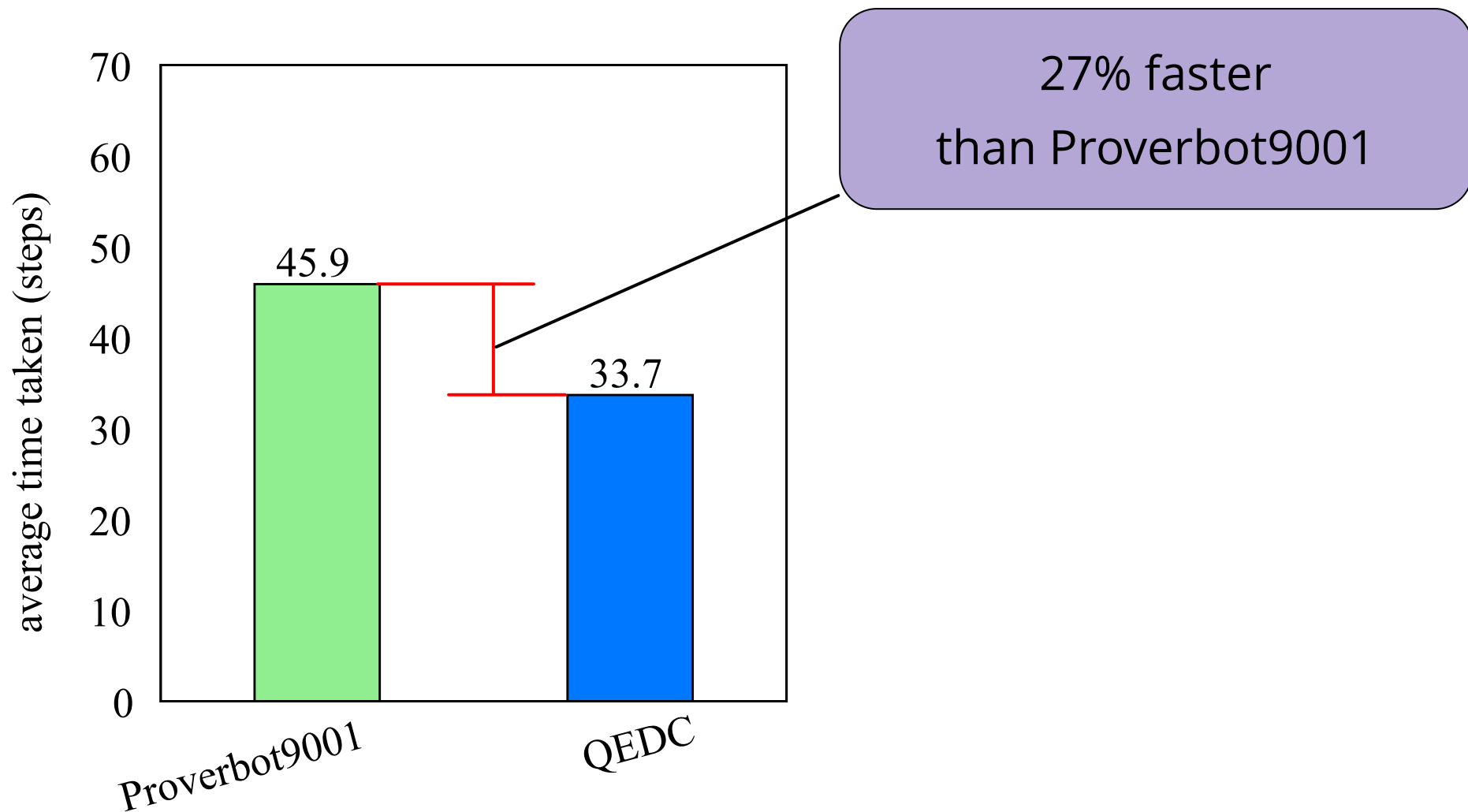
85/15 train-test split

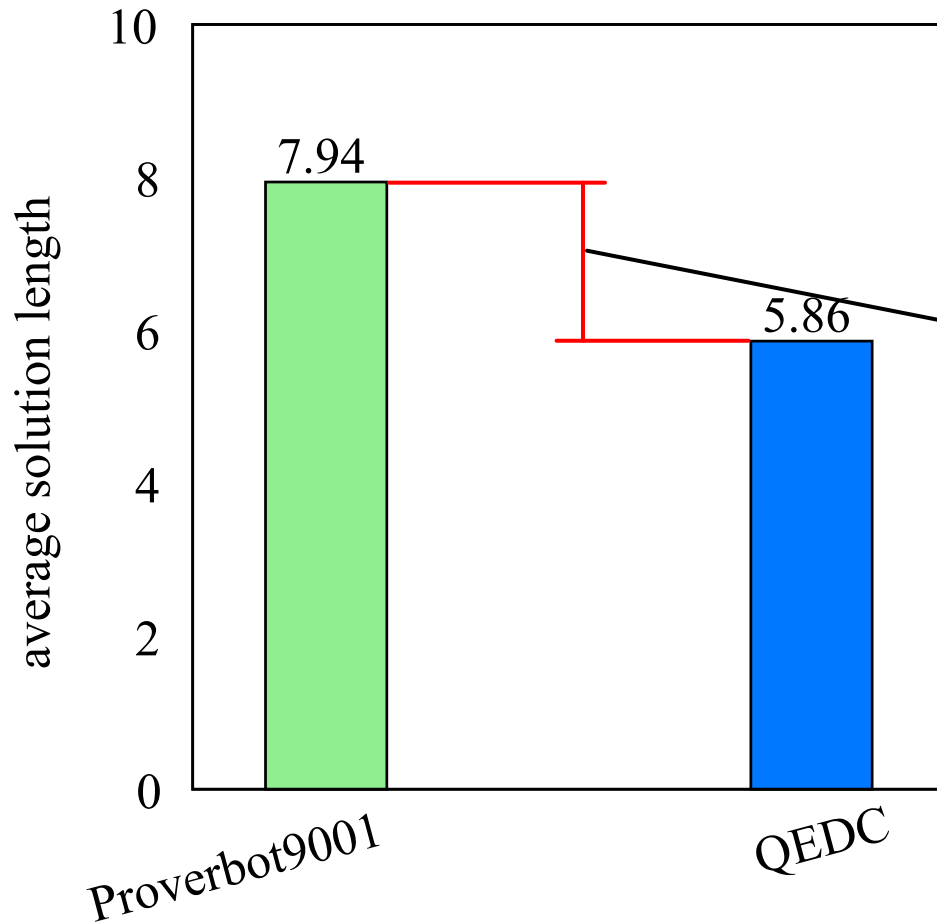
Baseline: Proverbot9001 (updated)

QEDCartographer, except without state
scoring-based search

Uses a variant of depth-first search instead



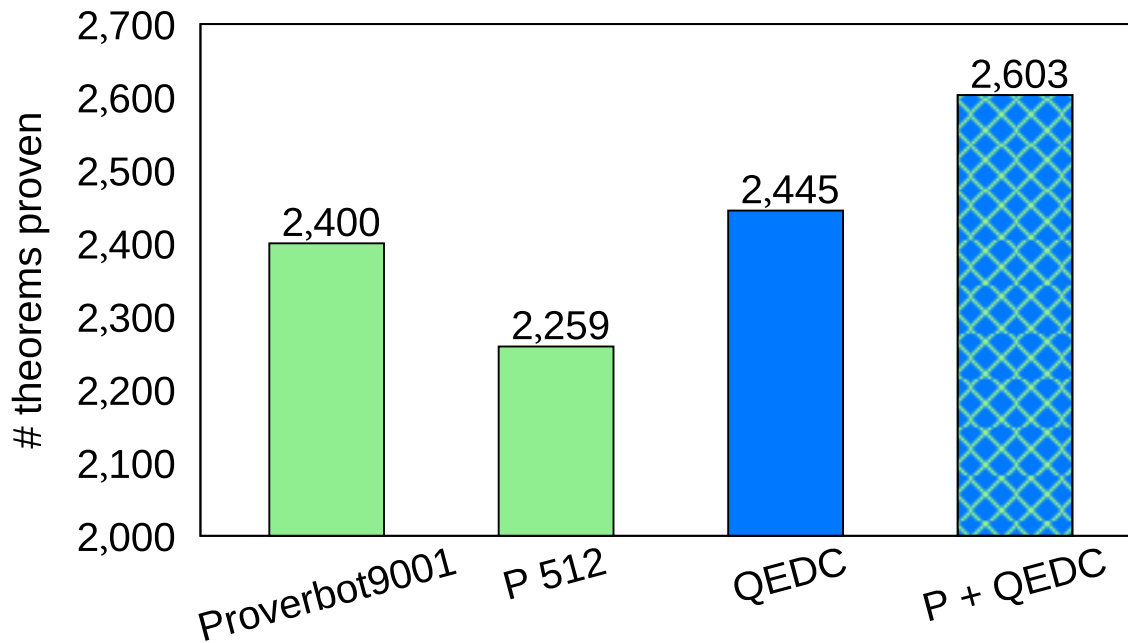




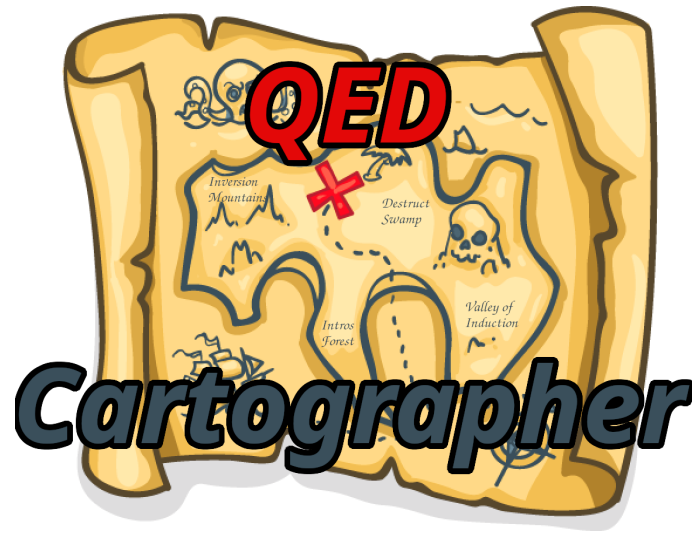
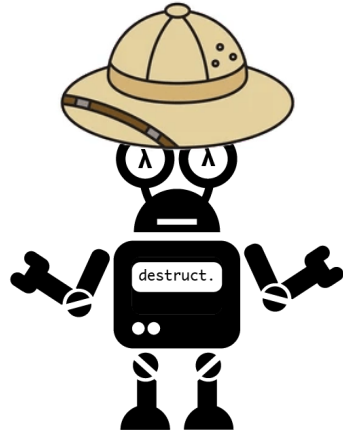
26% Shorter Proofs
than Proverbot9001

Shorter proofs are:

- More readable
- More maintainable
- Less expensive to check



Proves slightly more theorems, and proves complementary theorems



Automating Formal Verification with Reward-Free Reinforcement Learning

Uses a new V-value equation for branching goal structure

Makes producing verified-correct code easier and faster

Preprint available at alexsanchezstern.com