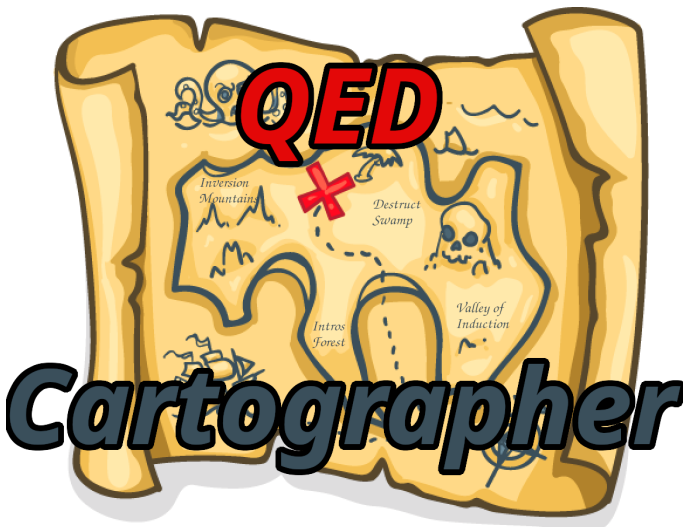
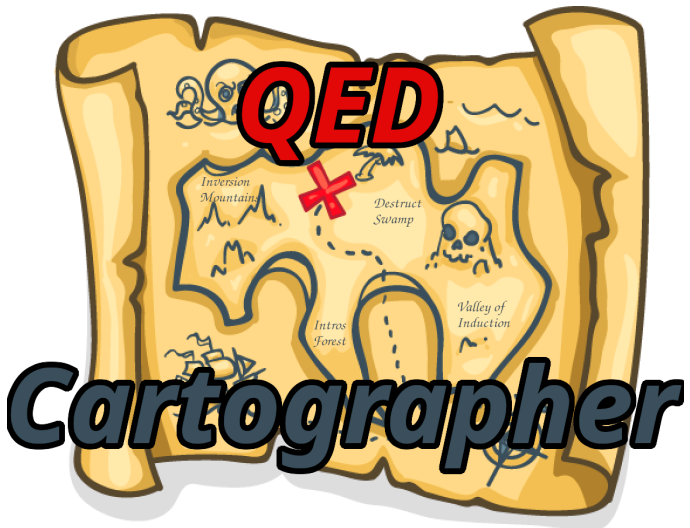


Alex Sanchez-Stern, Abhishek Varghese, Zhanna Kaufman,
Dylan Zhang, Talia Ringer, Yuriy Brun



Automating Formal Verification with Reward-Free Reinforcement Learning

Formal Verification

writing proofs about programs!

Formal Verification

writing proofs about programs!

more reliability
than testing!

Formal Verification

writing proofs about programs!

more reliability than testing!

labor intensive

Formal Verification

writing proofs about programs!

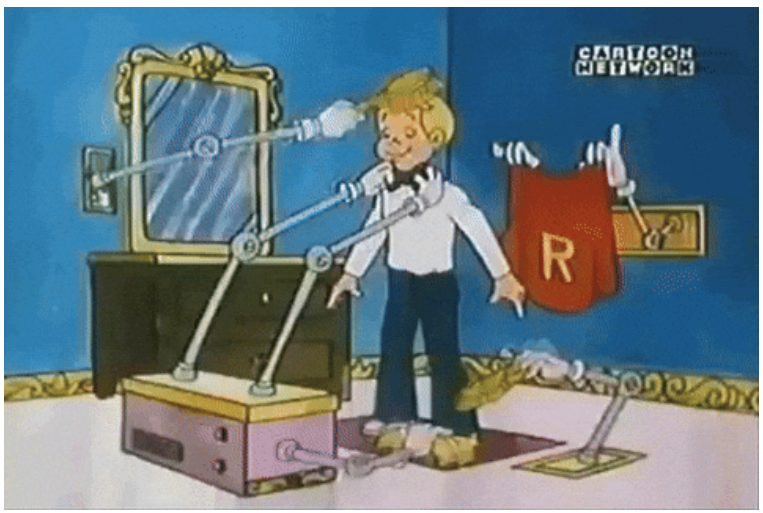
more reliability than testing!

labor intensive

requires specialized expertise

Automating Formal Verification

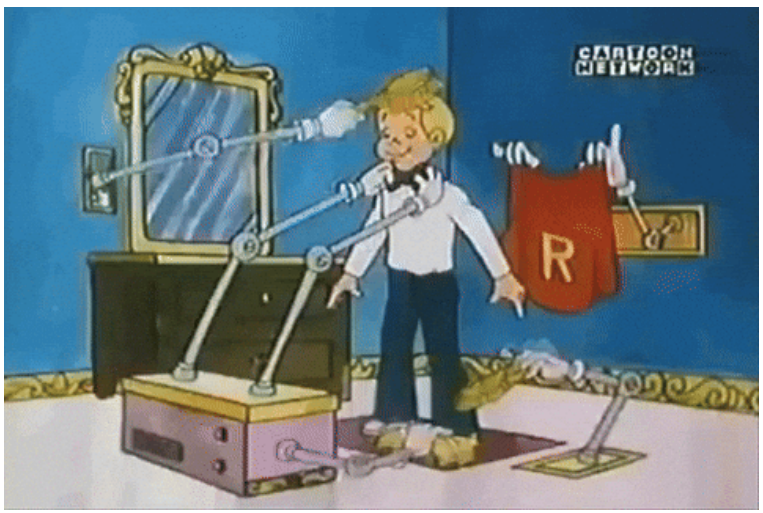
Proof Synthesis!



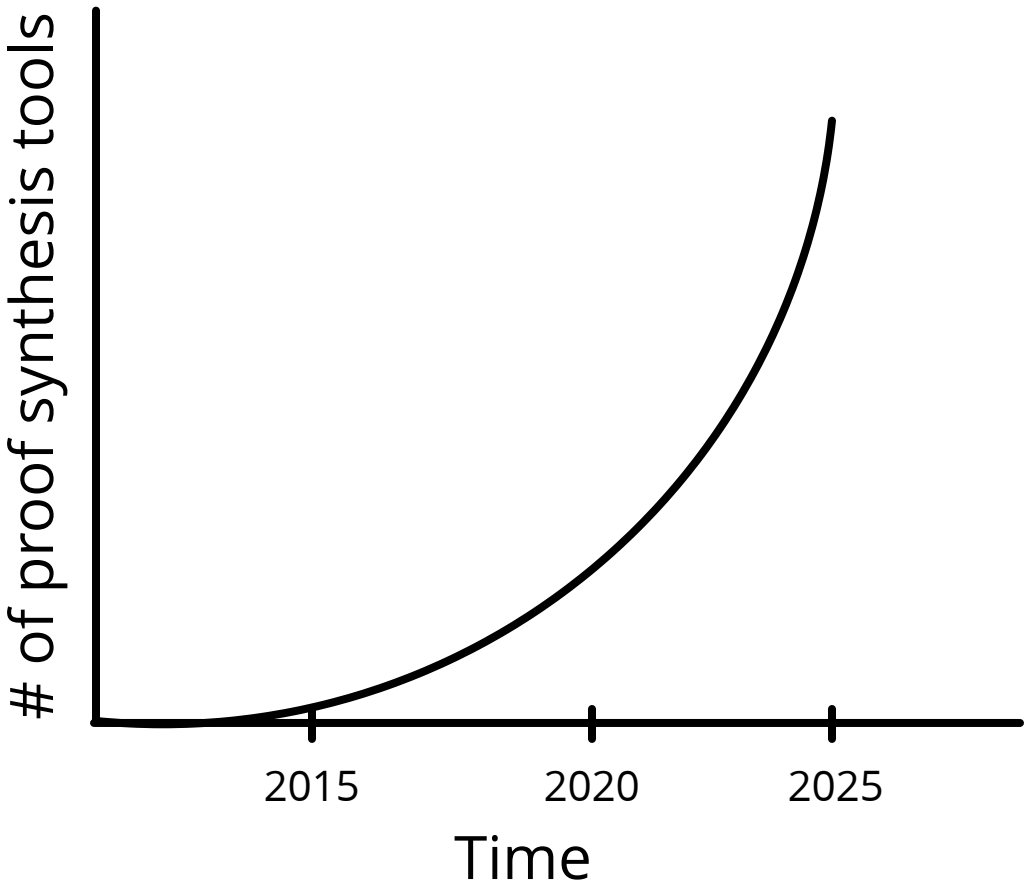
Labor saving!

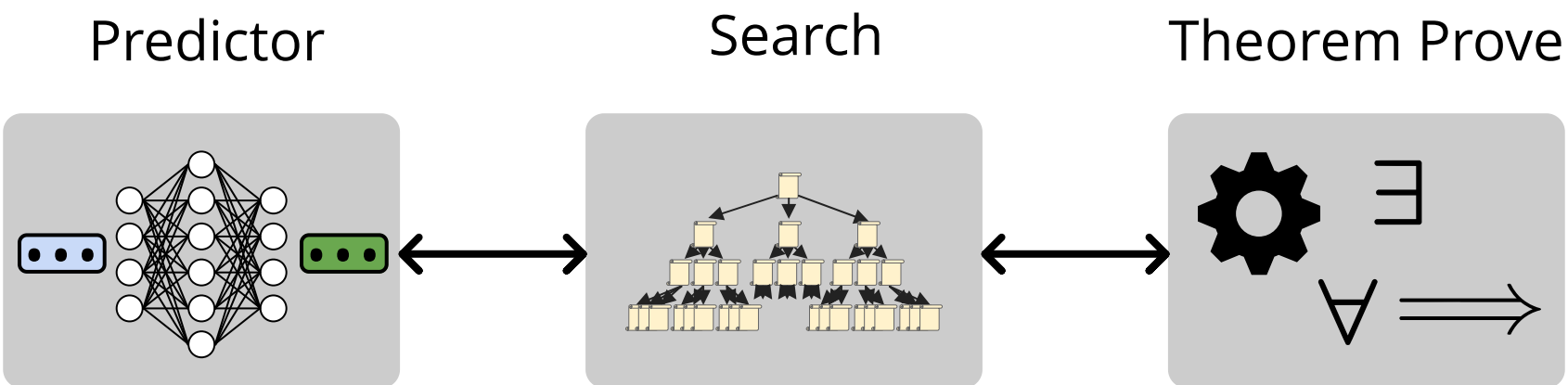
Automating Formal Verification

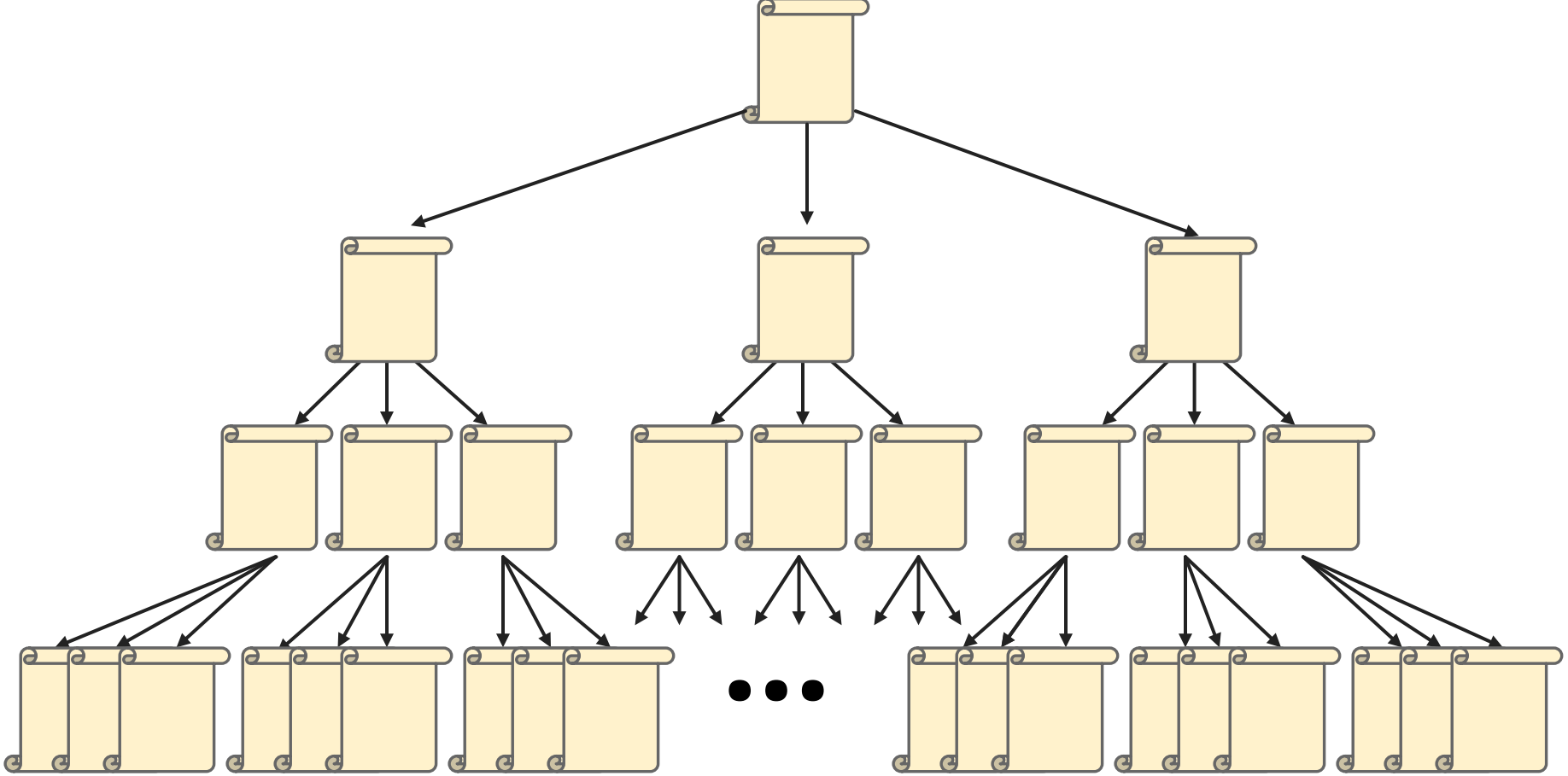
Proof Synthesis!

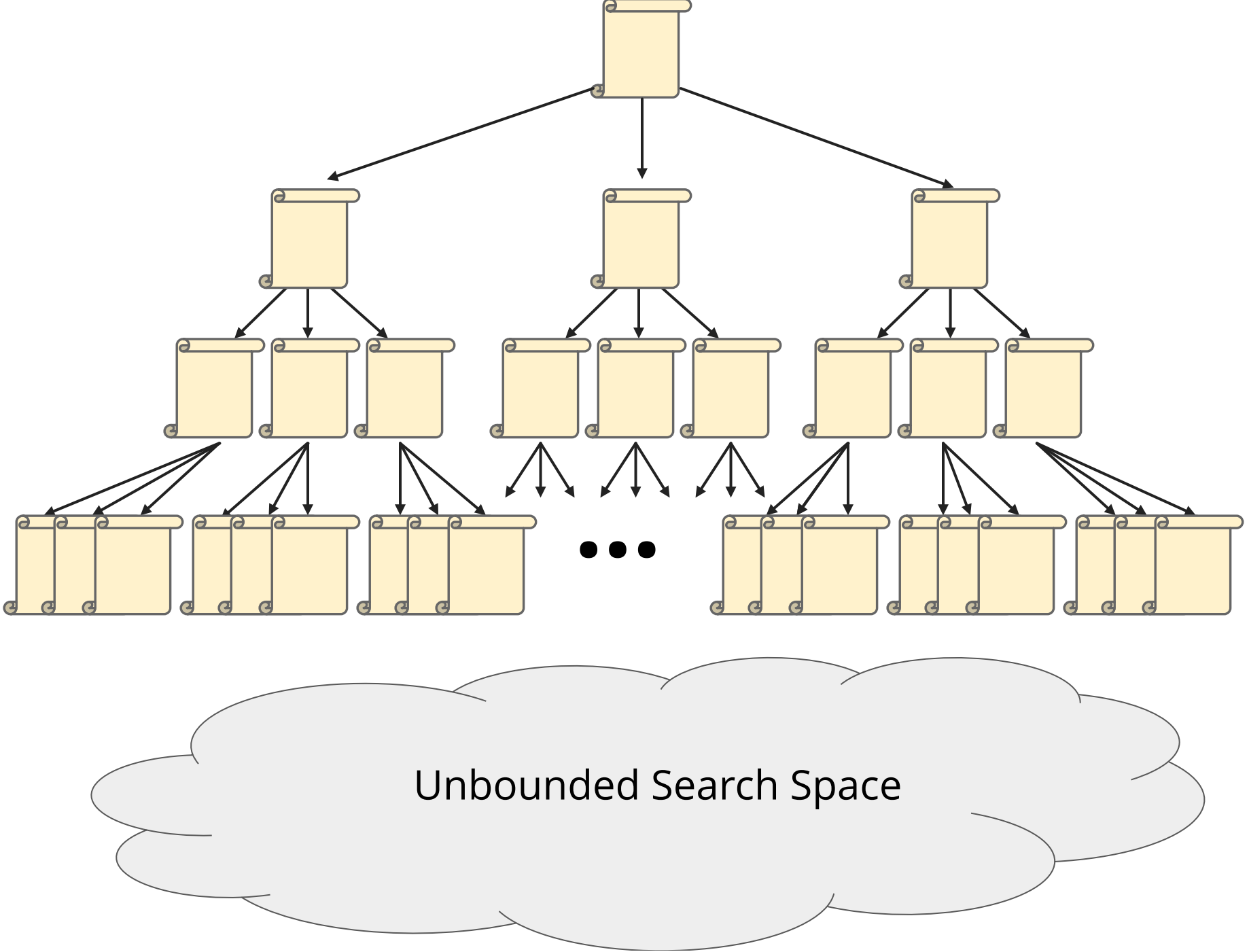


Labor saving!




Predictor
Search
Theorem Prover
∃
∀
⟹

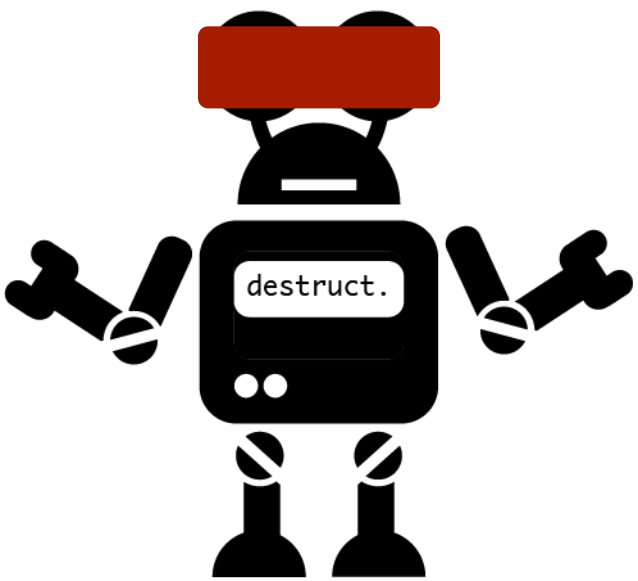



Unbounded Search Space

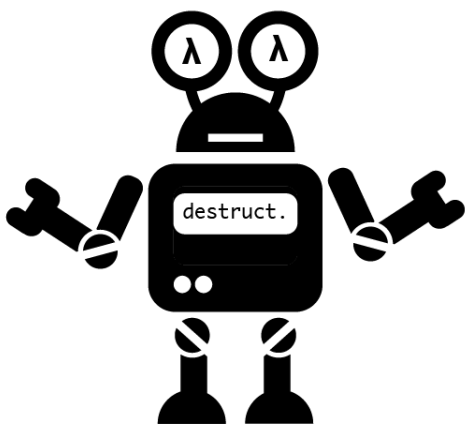
There are some techniques that can help us prune the search tree

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

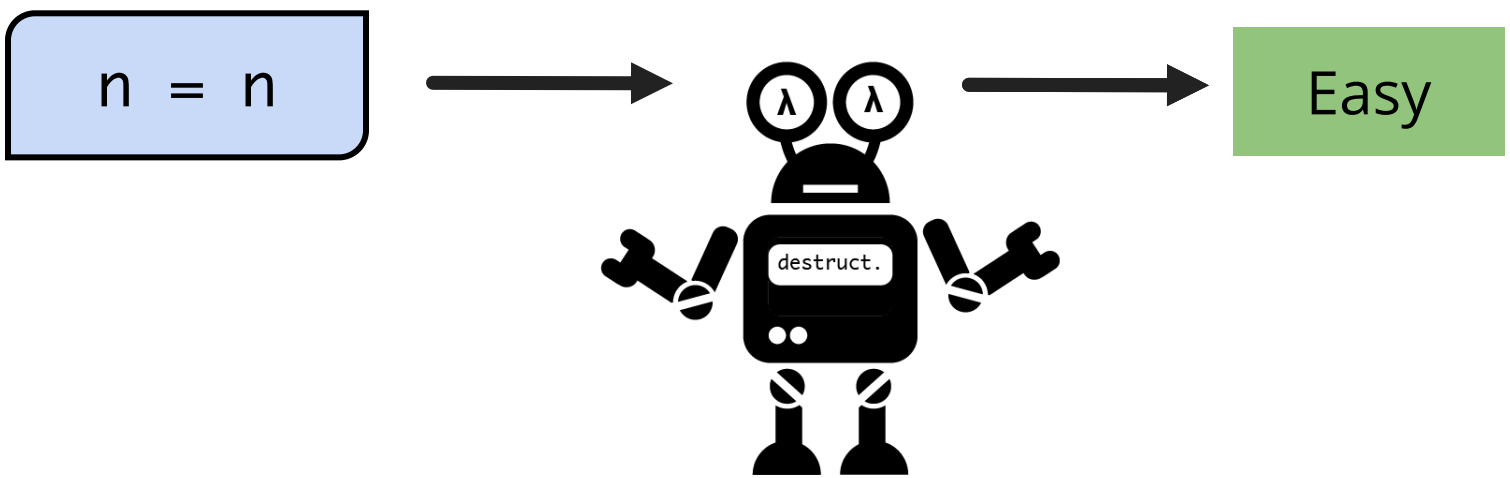
It's hard to explore when you don't know how close you are!



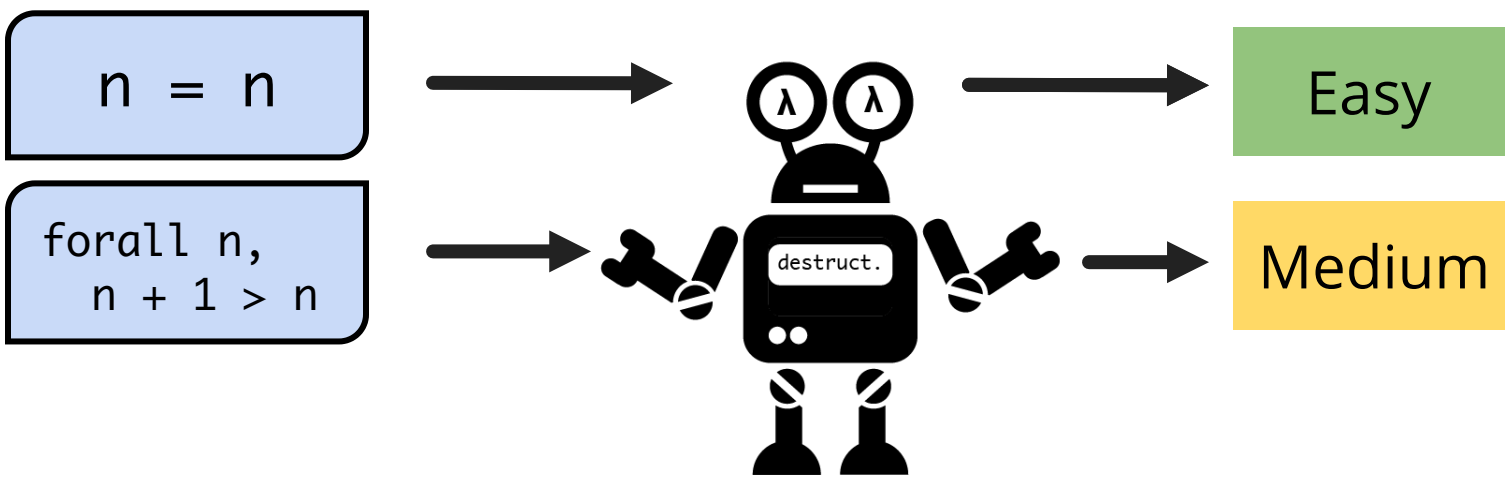
We can search more efficiently if we can evaluate proof states



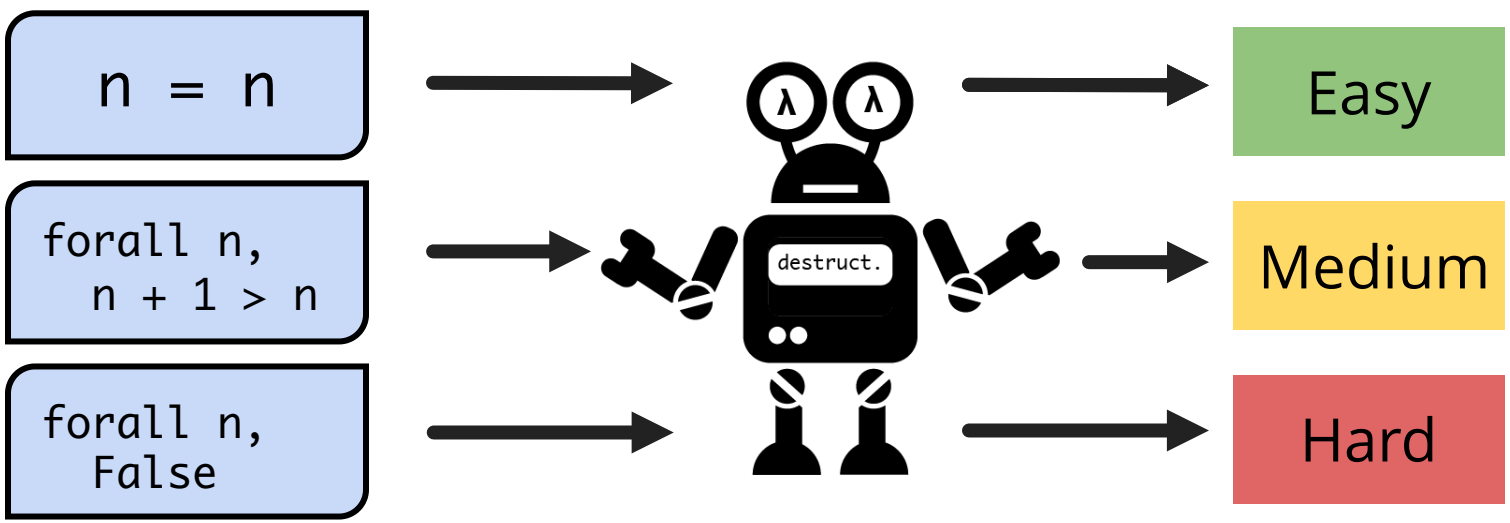
We can search more efficiently if we can evaluate proof states

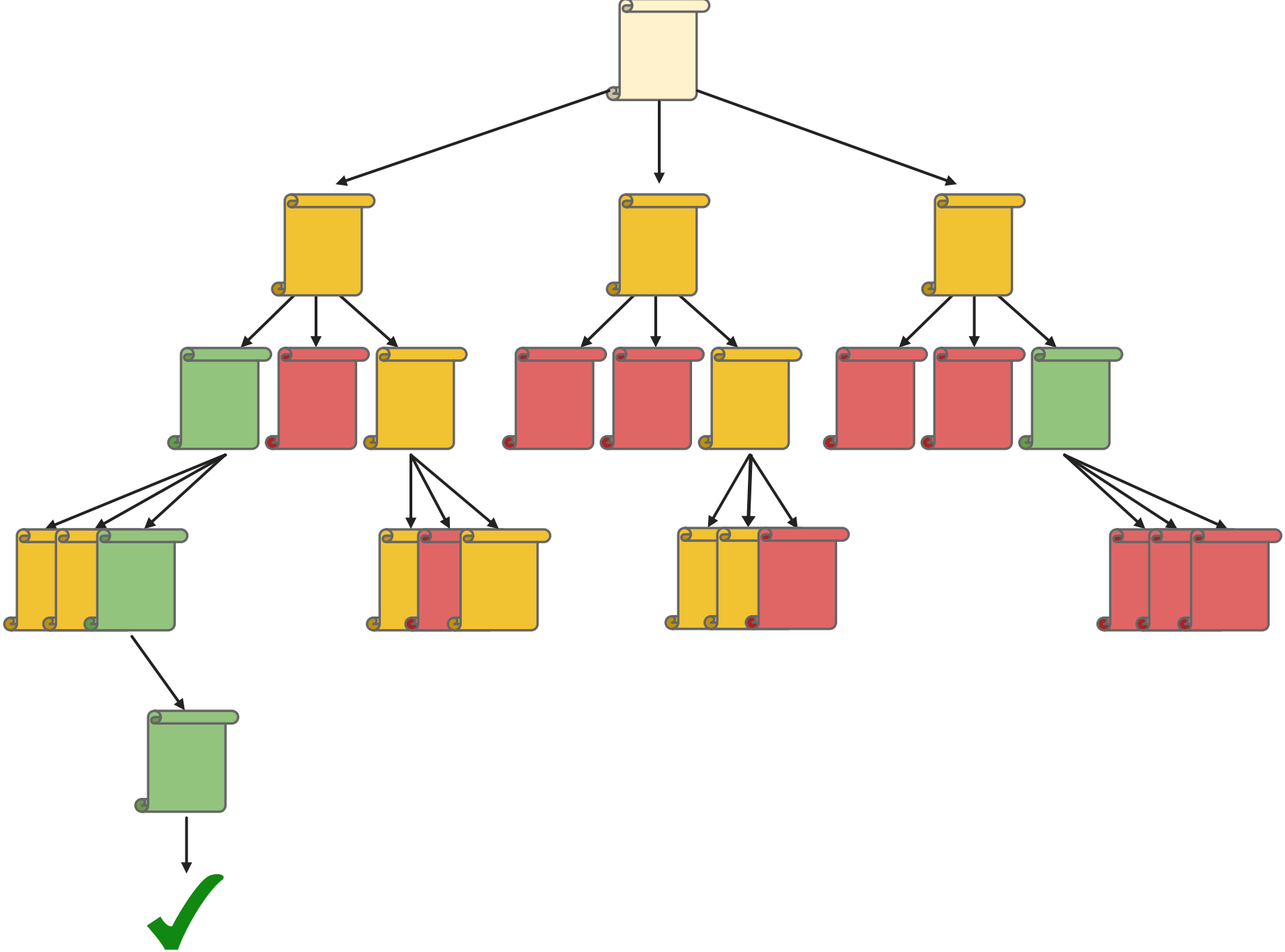


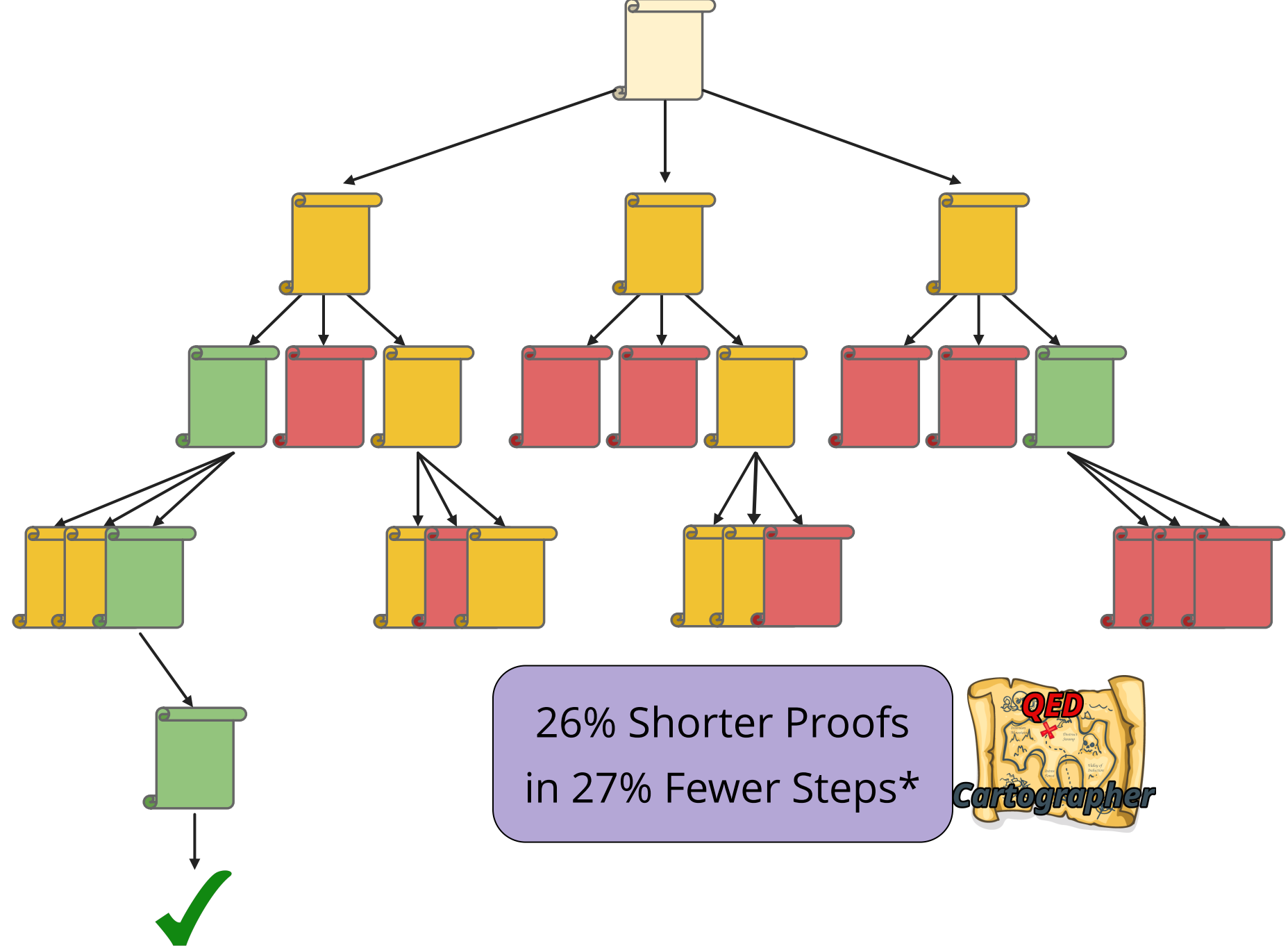
We can search more efficiently if we can evaluate proof states

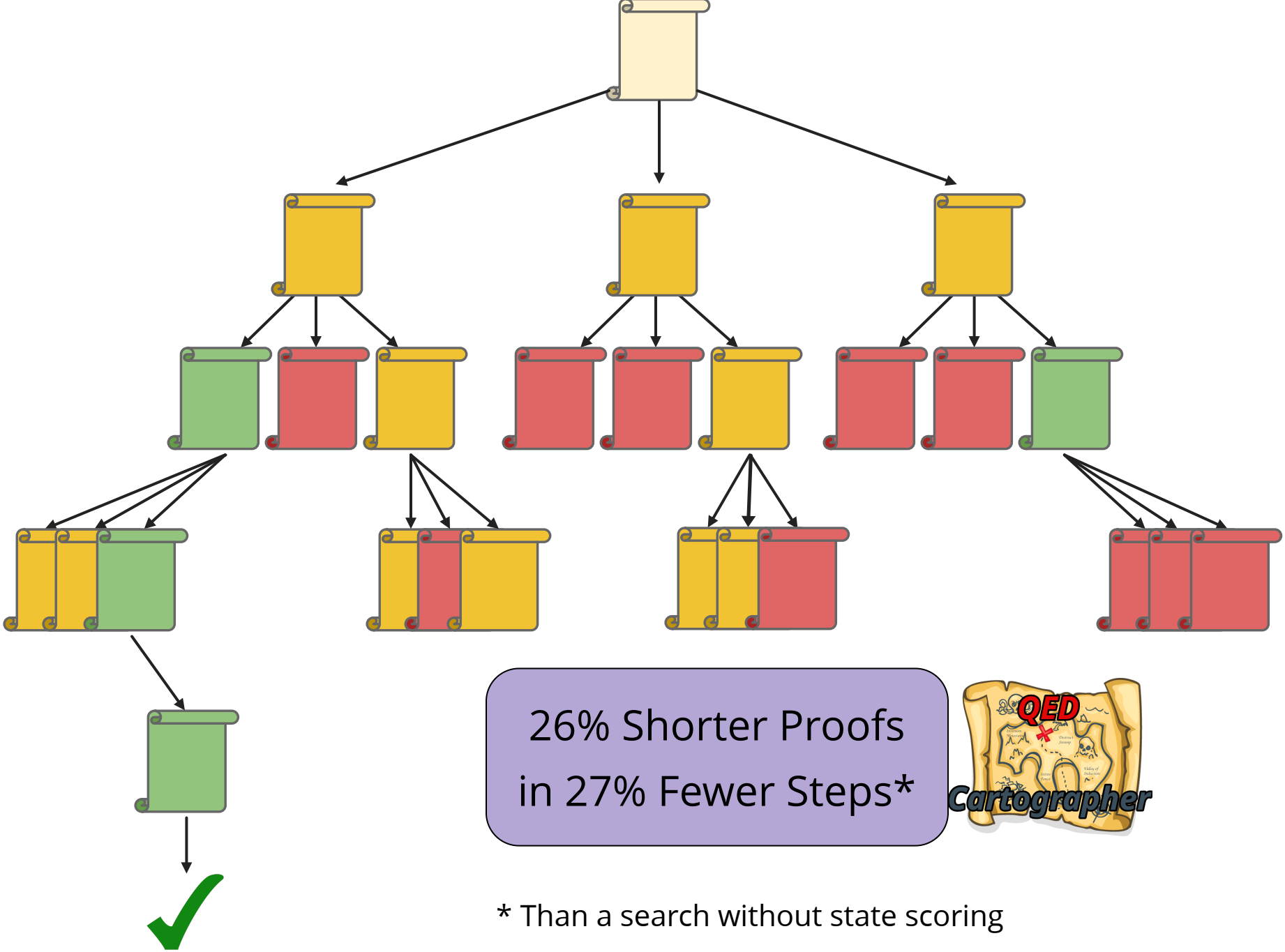


We can search more efficiently if we can evaluate proof states



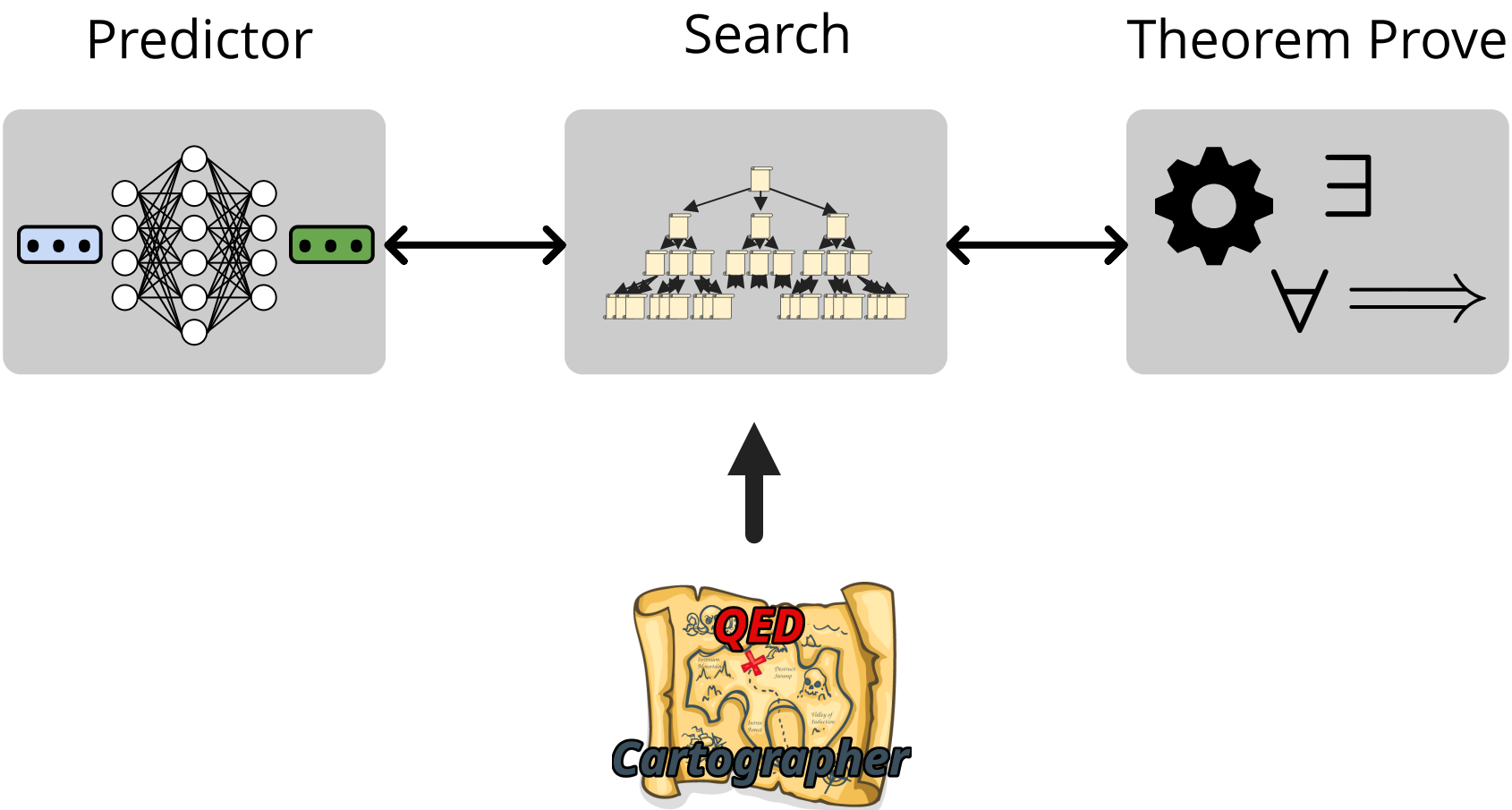



26% Shorter Proofs
in 27% Fewer Steps*
QED
Cartographer



26% Shorter Proofs
in 27% Fewer Steps*

* Than a search without state scoring


Predictor
Search
Theorem Prover
∃
∀
⟹
QED
Cartographer

"Reward-free Reinforcement Learning"

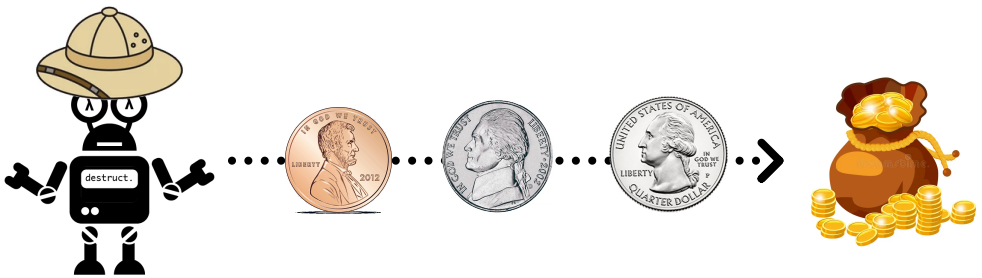
"Reward-free Reinforcement Learning"

In particular, V-learning

This Talk

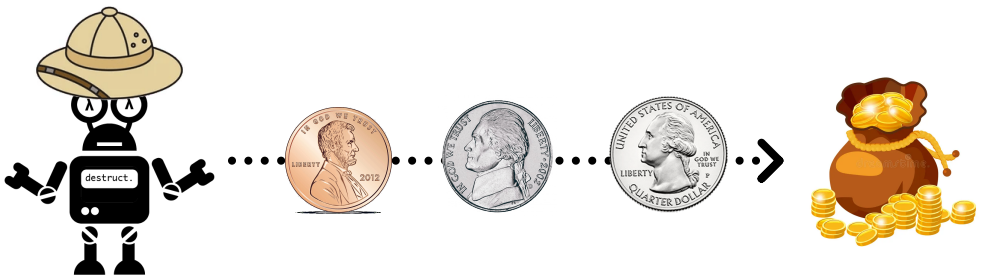
This Talk

V-Learning

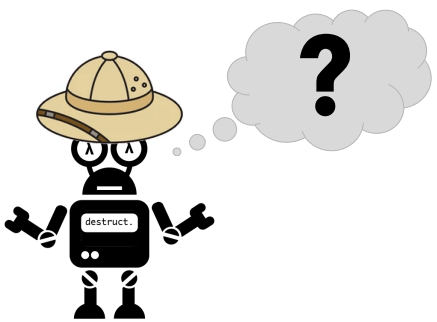


This Talk

V-Learning

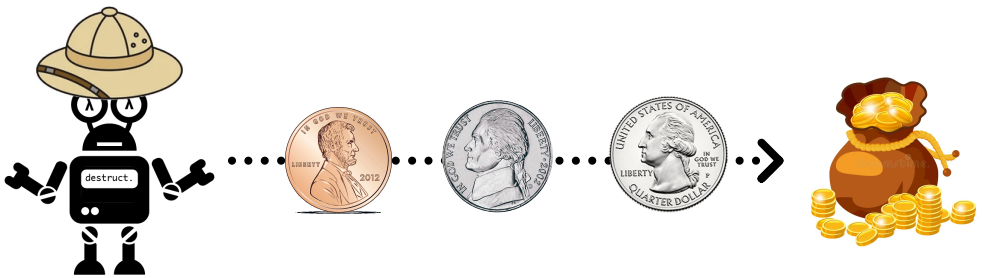


Limitations in Proofs

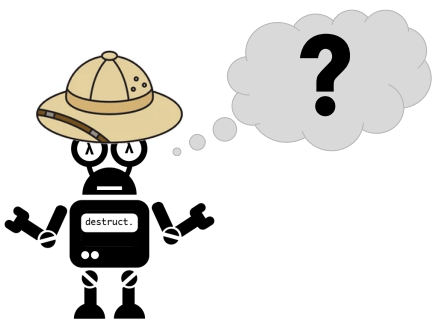


This Talk

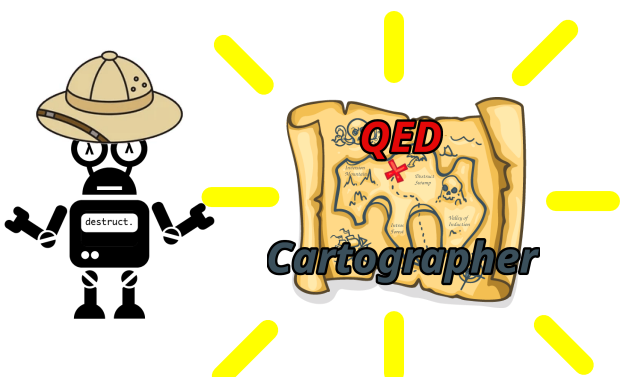
V-Learning



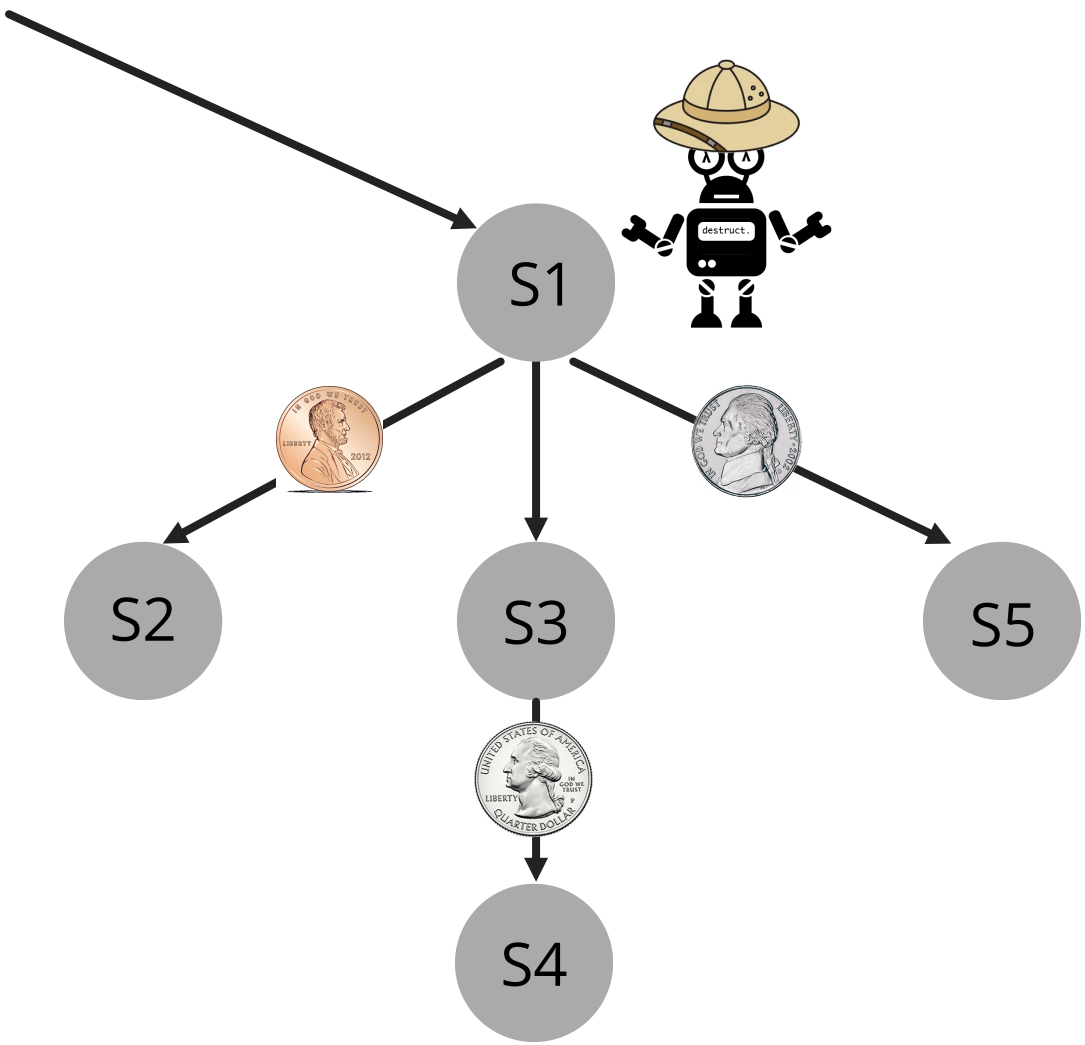
Limitations in Proofs



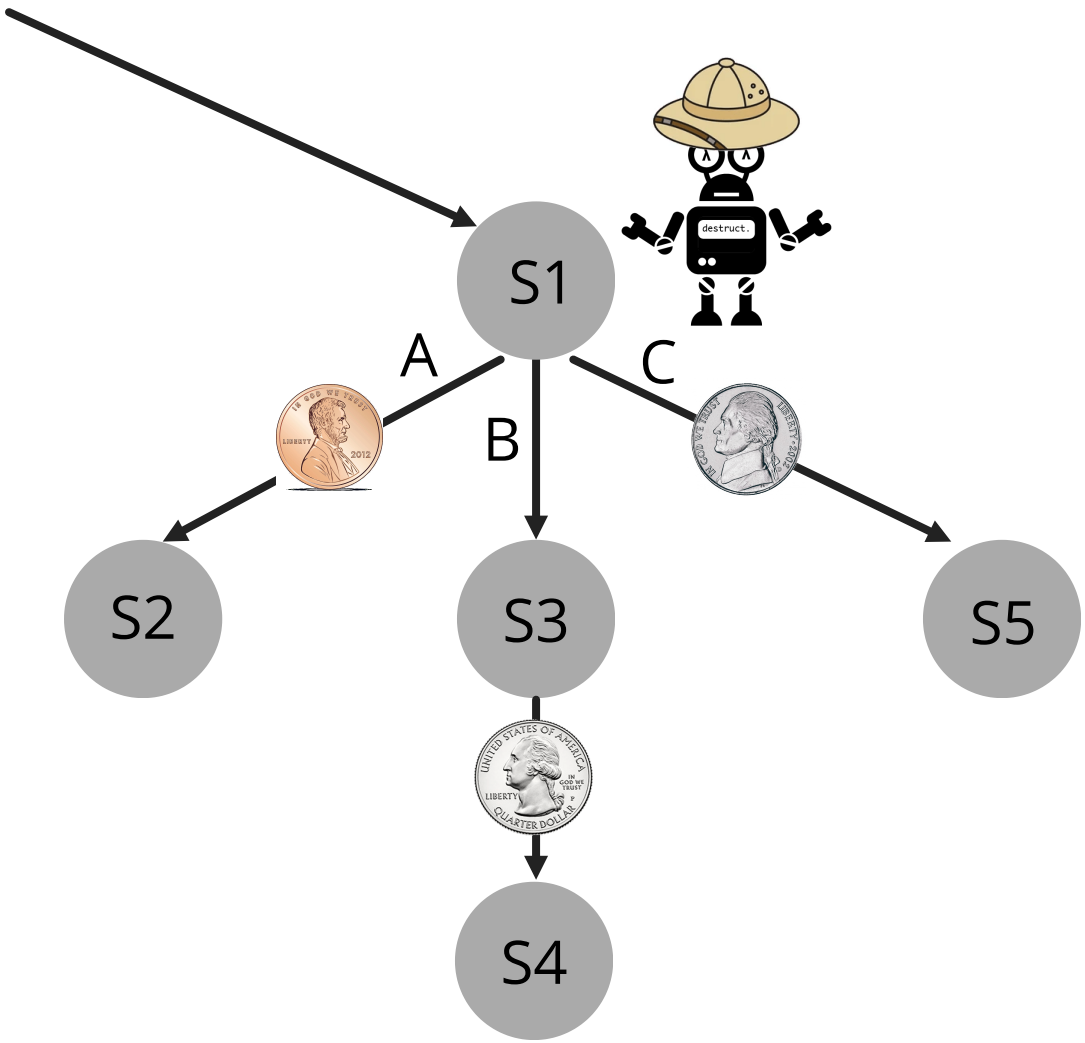
Adapting to Proofs



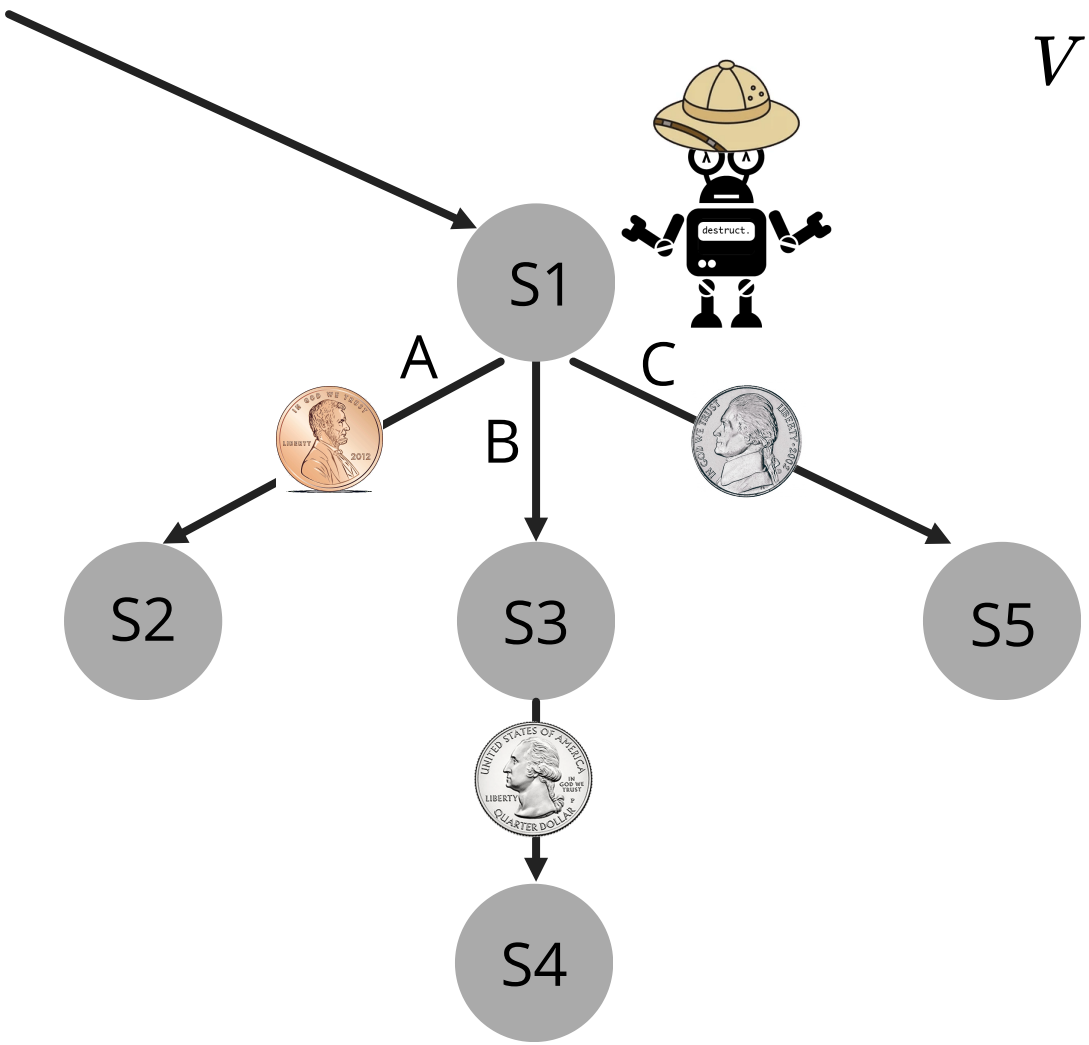
Classic V-Learning



Classic V-Learning



Classic V-Learning



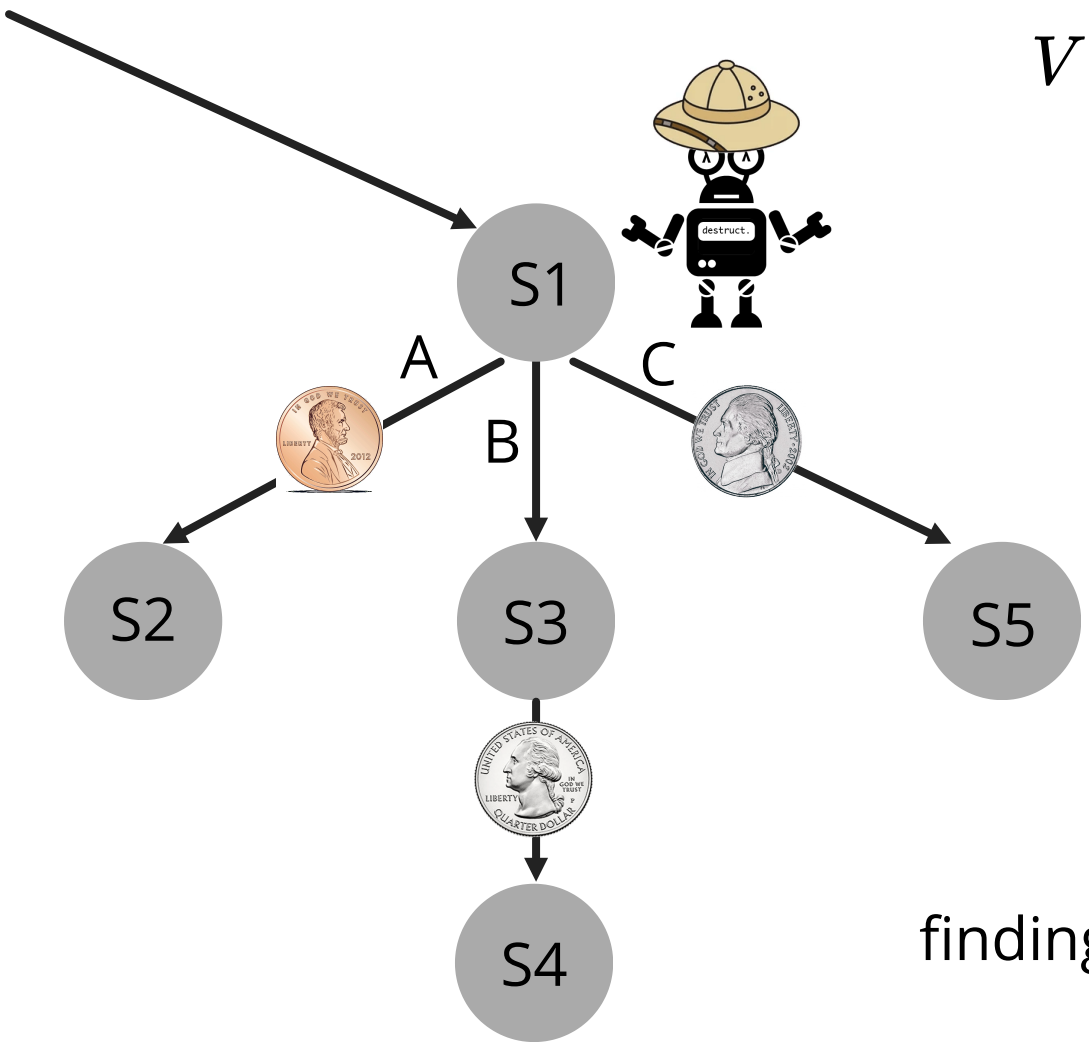
$$V(S1) = \max($$

 $+ \quad V(S2),$

$($  $+ \quad V(S4)),$

 $+ \quad V(S5))$

Classic V-Learning



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

 $+\gamma V(S2)$,

$\gamma($  $+\gamma V(S4))$,

 $+\gamma V(S5))$

γ = time discount

finding rewards sooner is better!

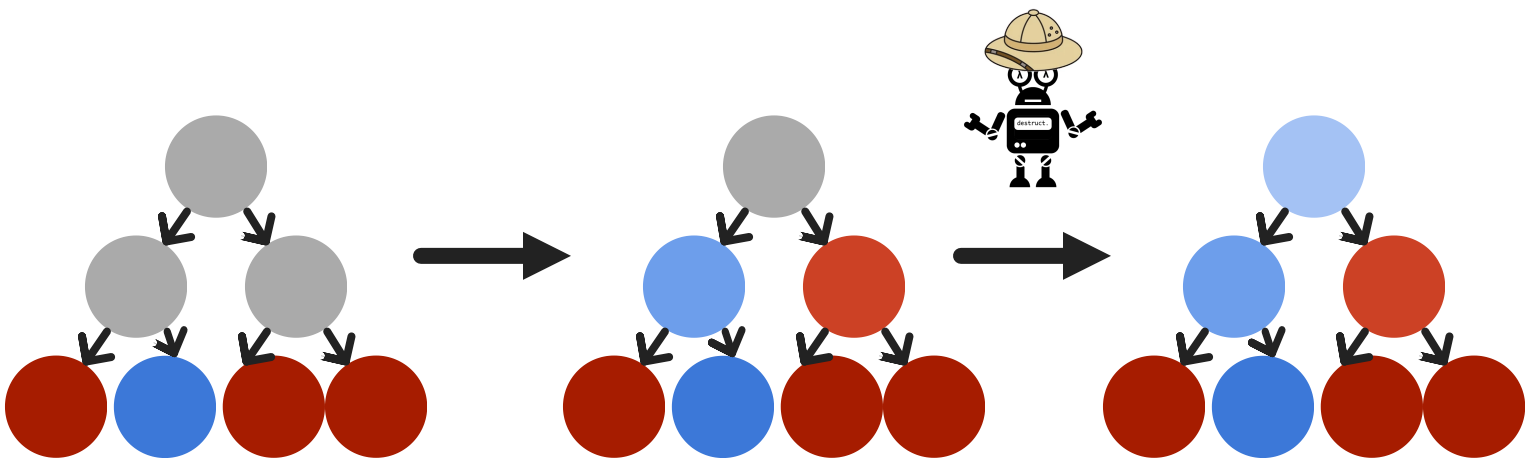
$$V(S1) = \mathbf{max}(\text{  } + \gamma V(S2),\ \gamma\left(\text{  } + \gamma V(S4)\right), \text{  } + \gamma V(S5))$$

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

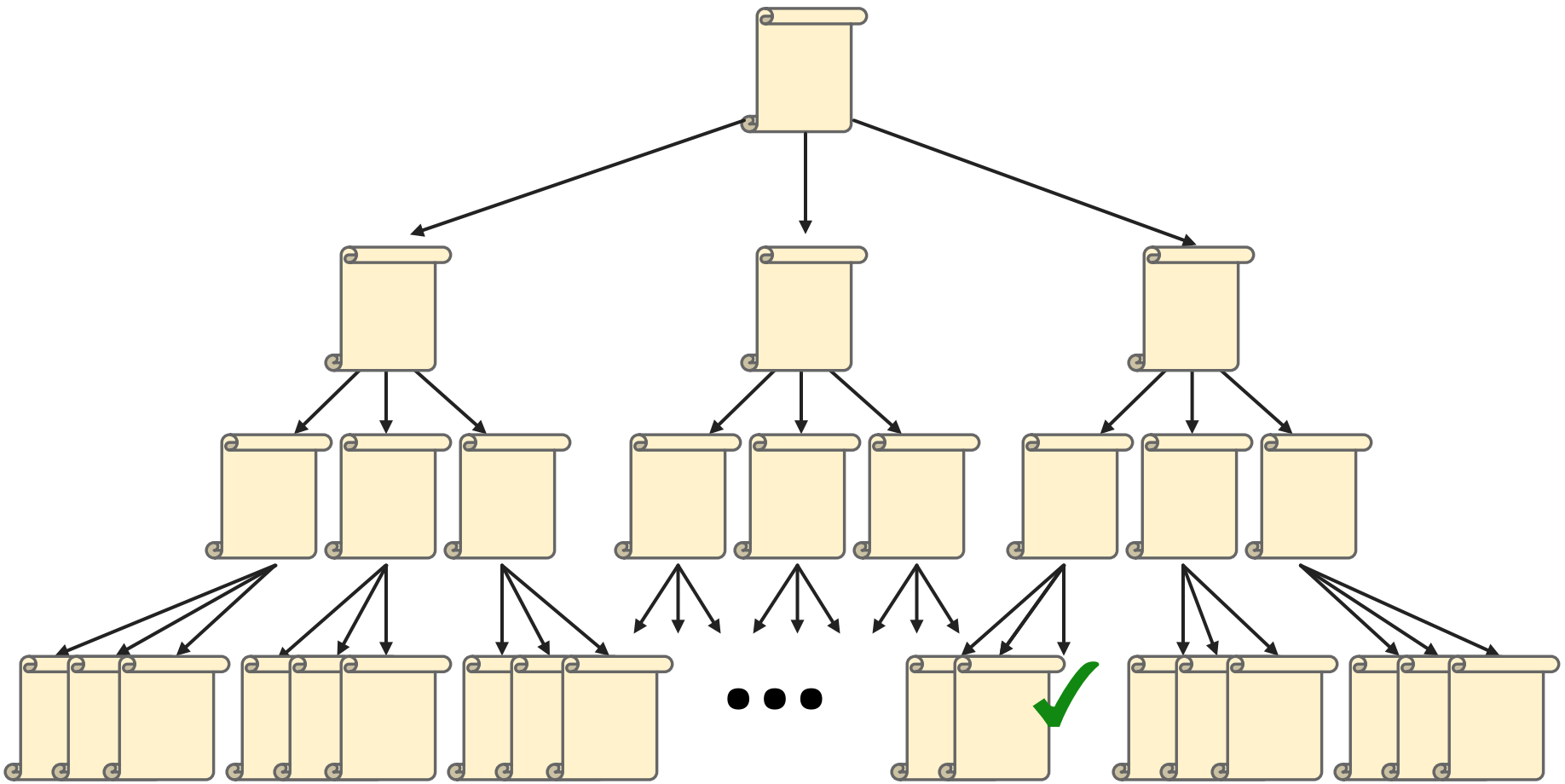
Generalizes to

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

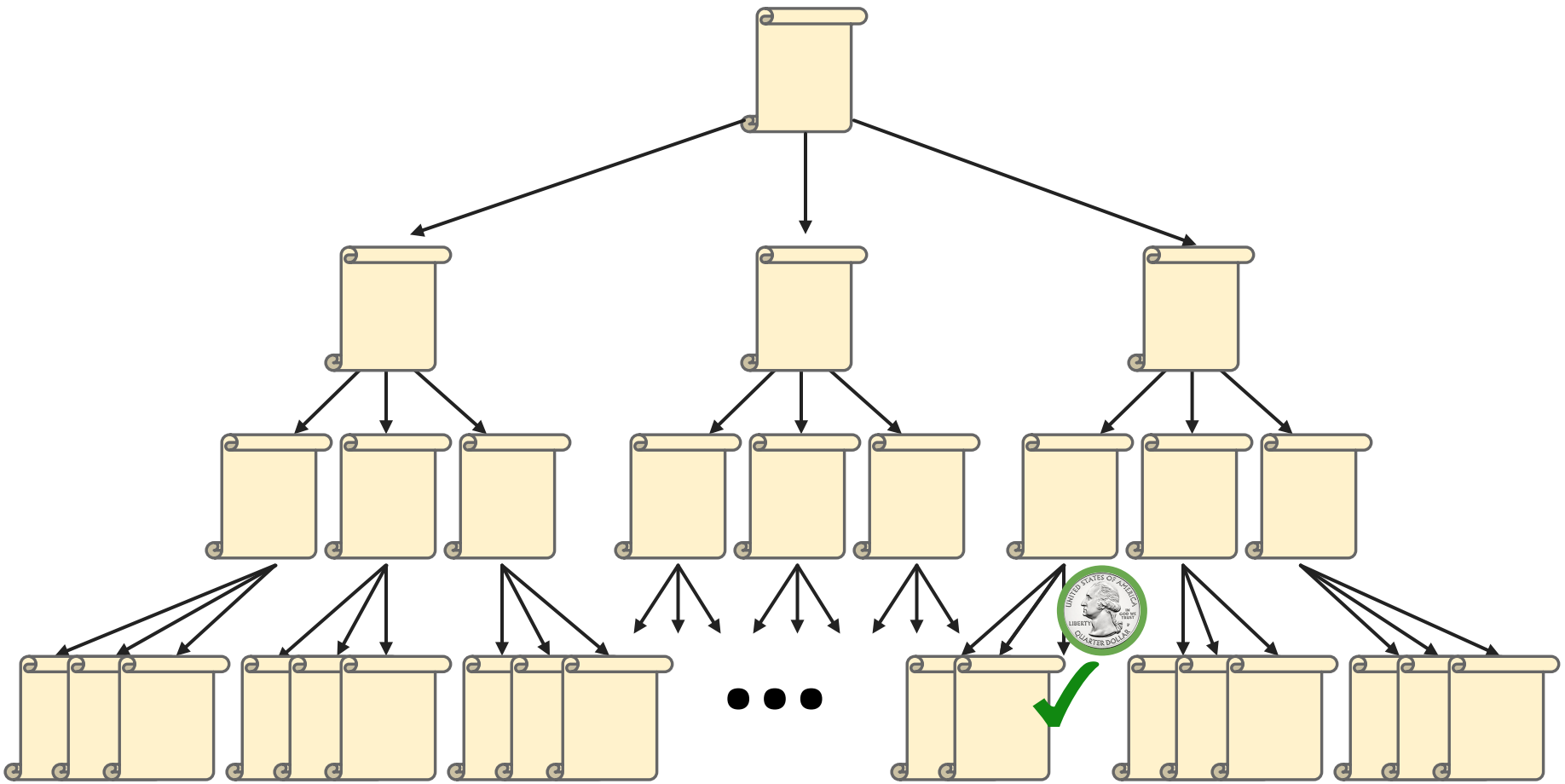
V-Learning in Practice: Iterative Updates



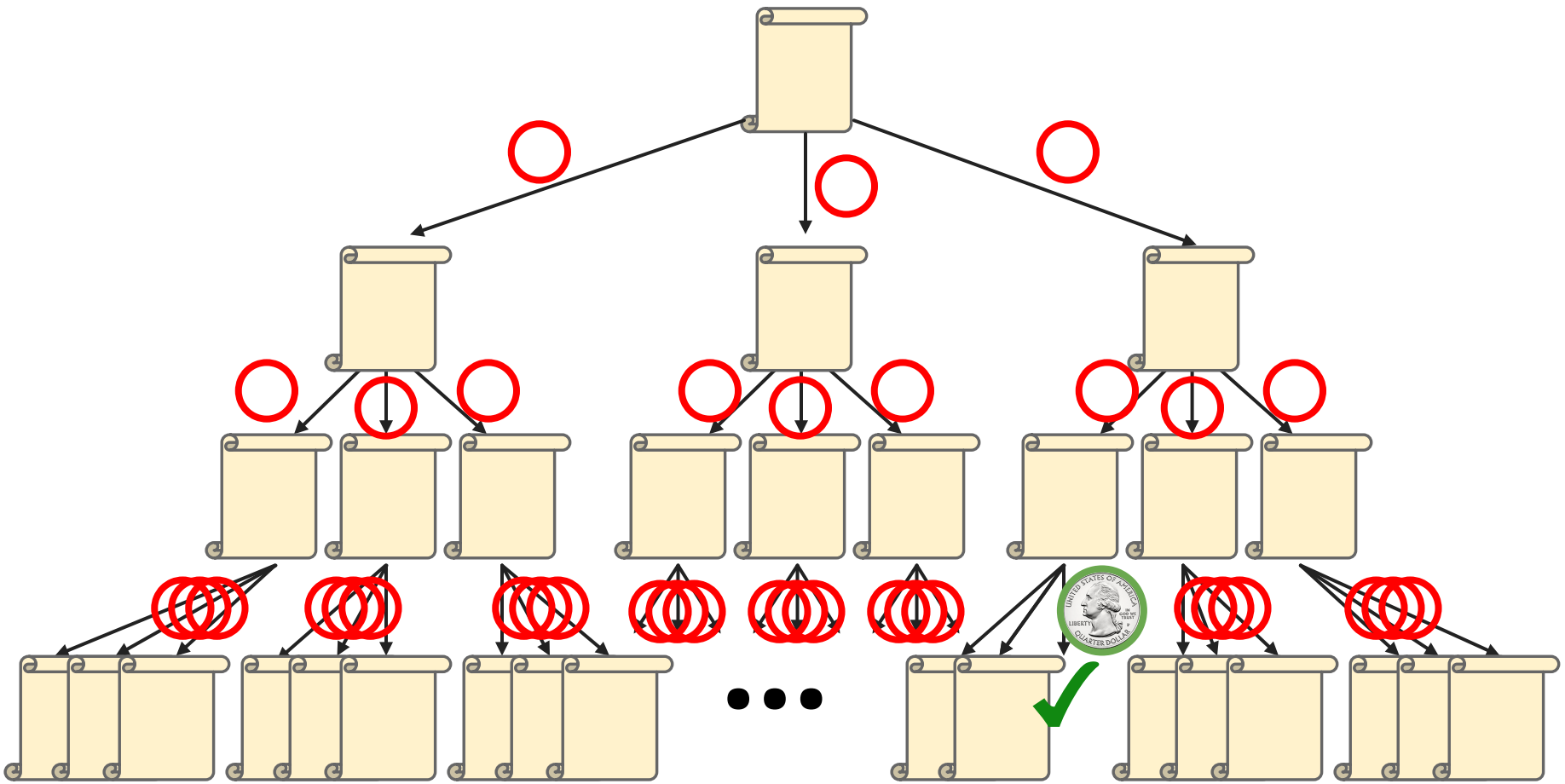
V-Learning in Proofs:
The Sparse Reward Problem



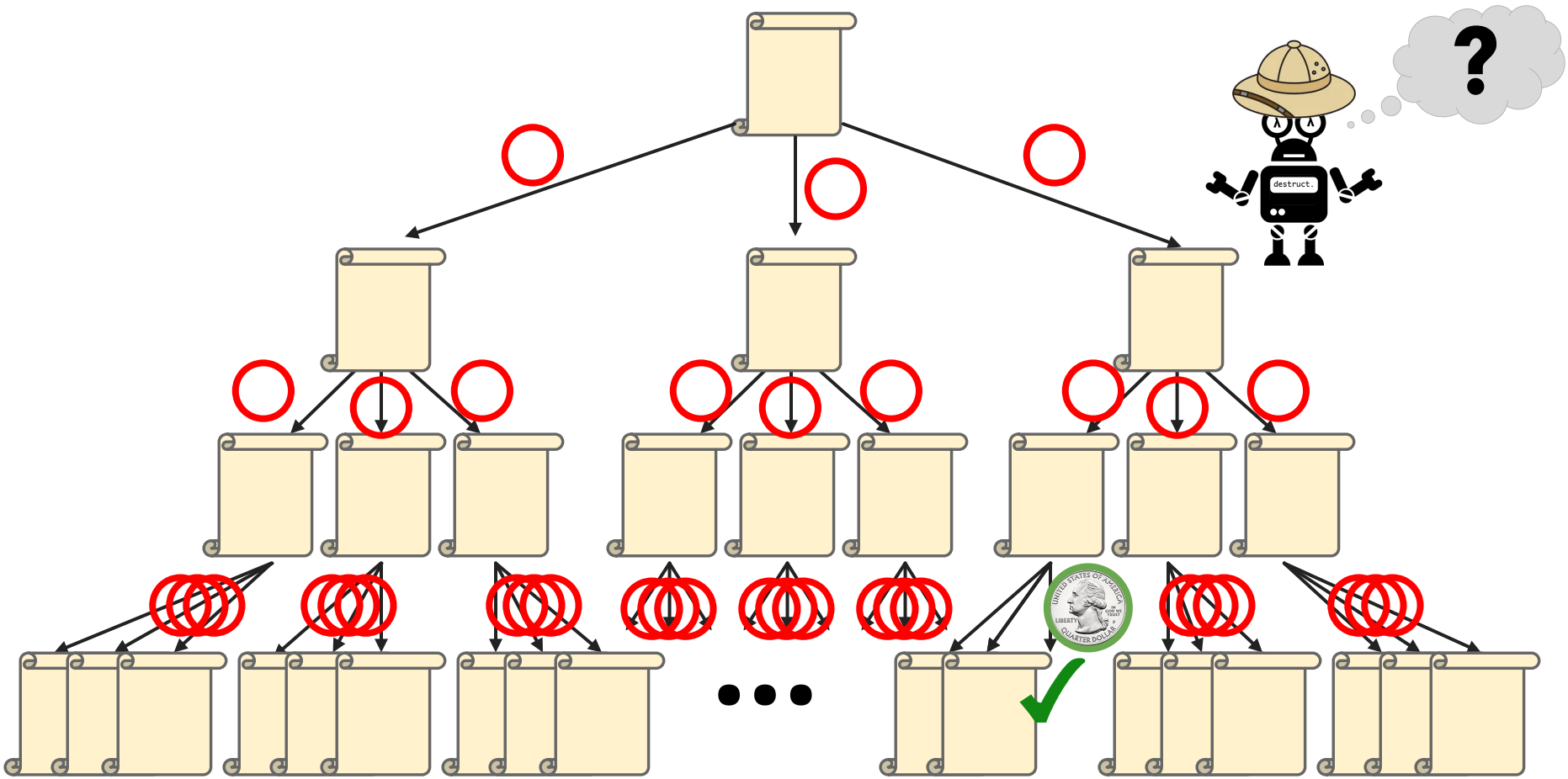
V-Learning in Proofs: The Sparse Reward Problem



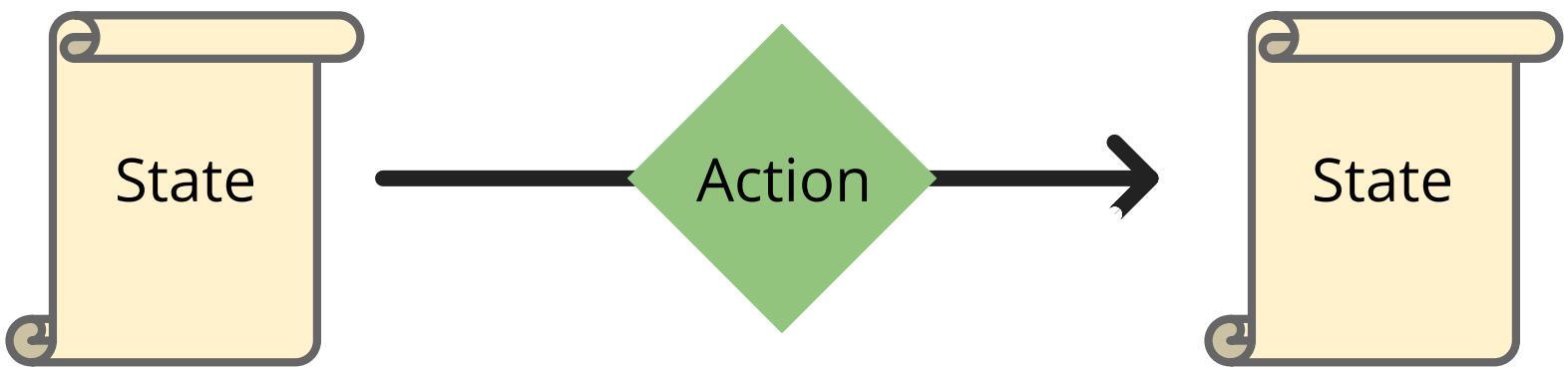
V-Learning in Proofs: The Sparse Reward Problem

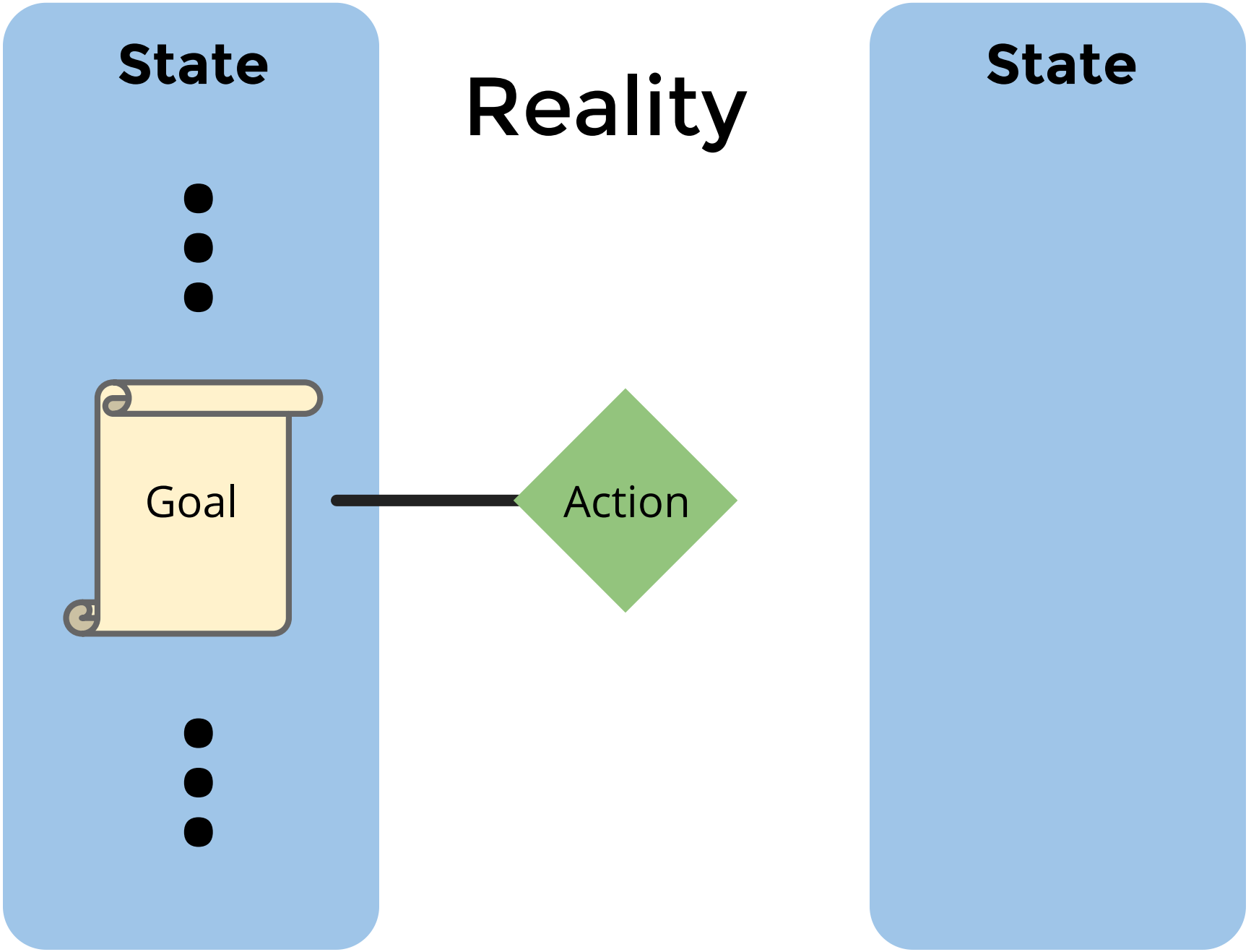


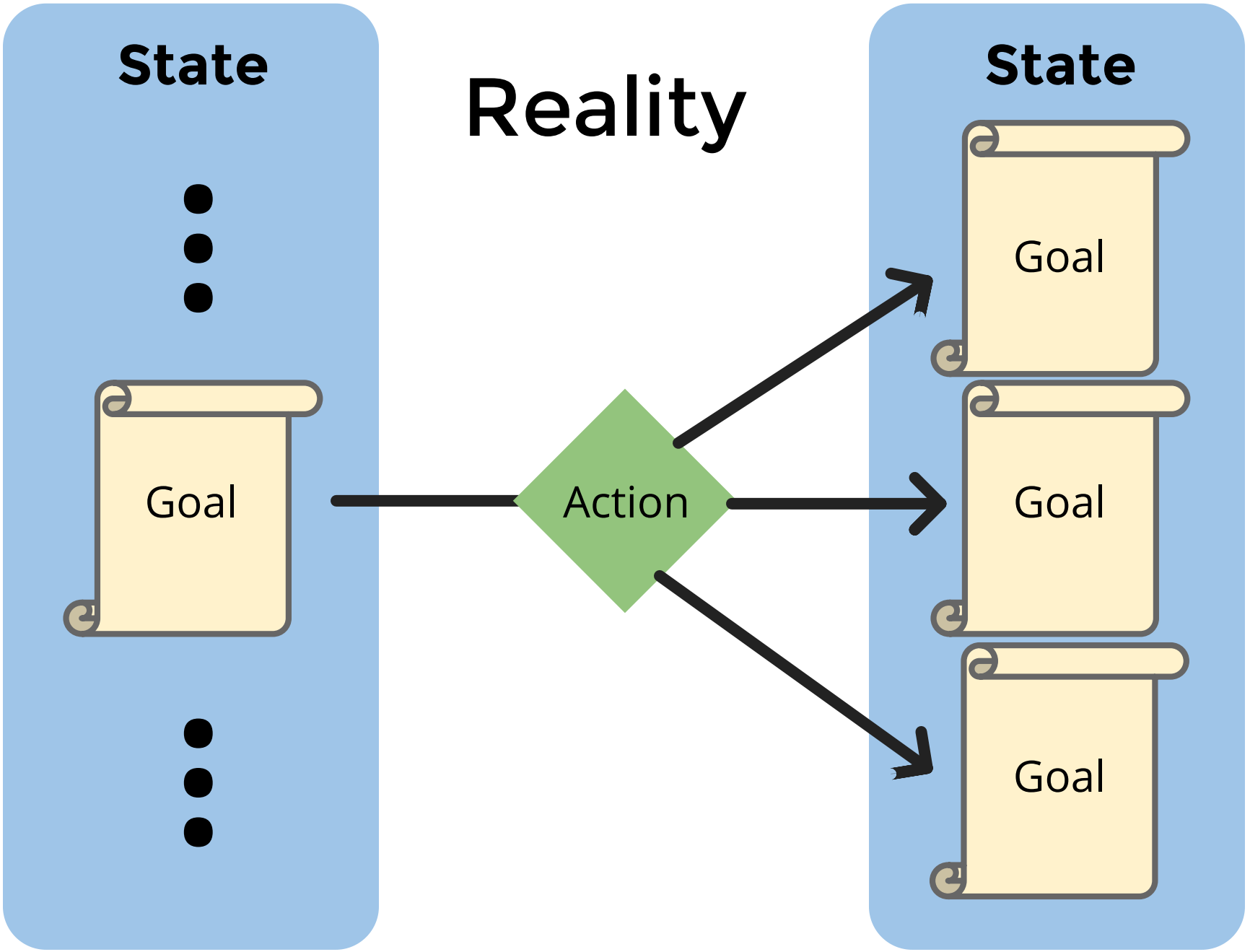
V-Learning in Proofs: The Sparse Reward Problem

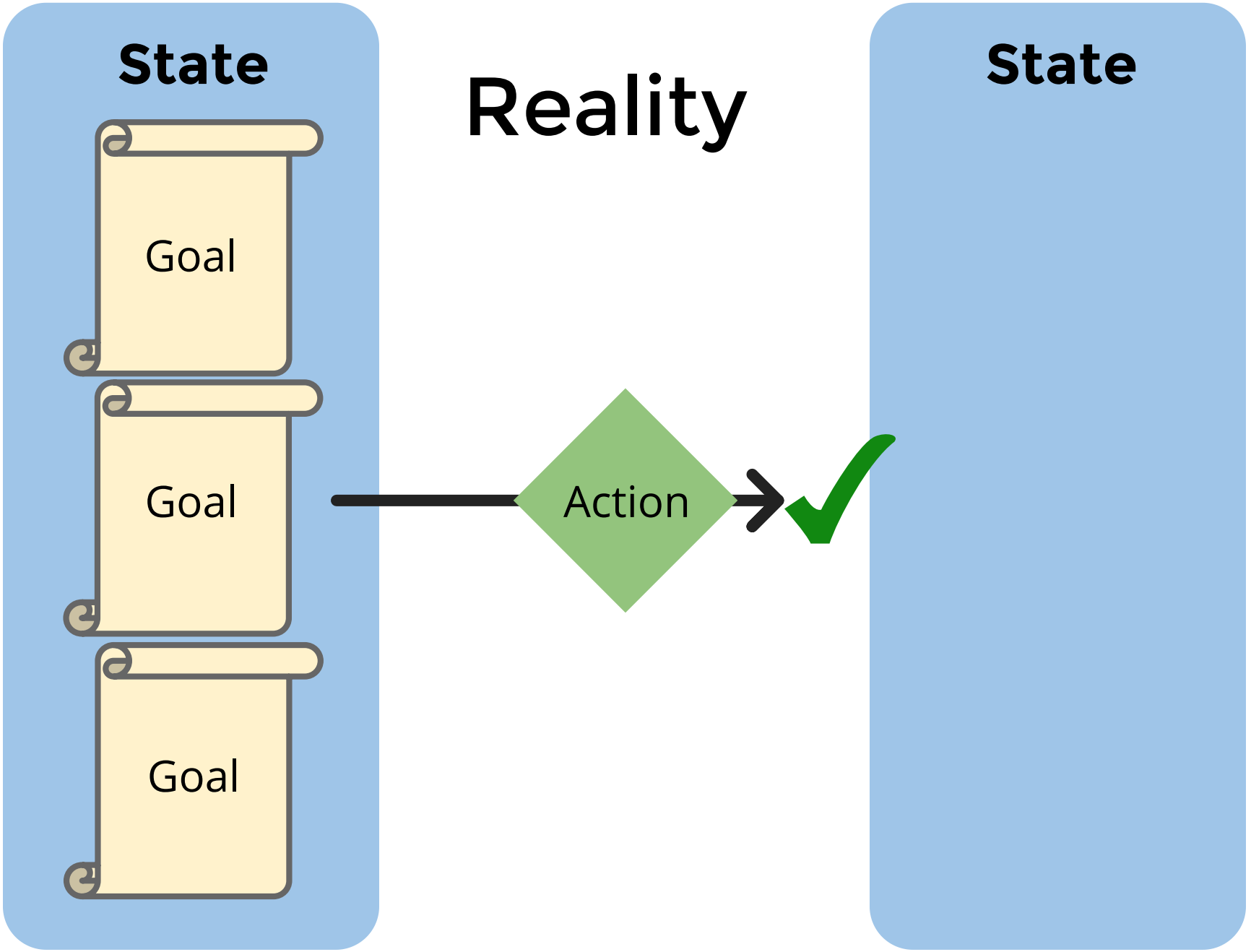


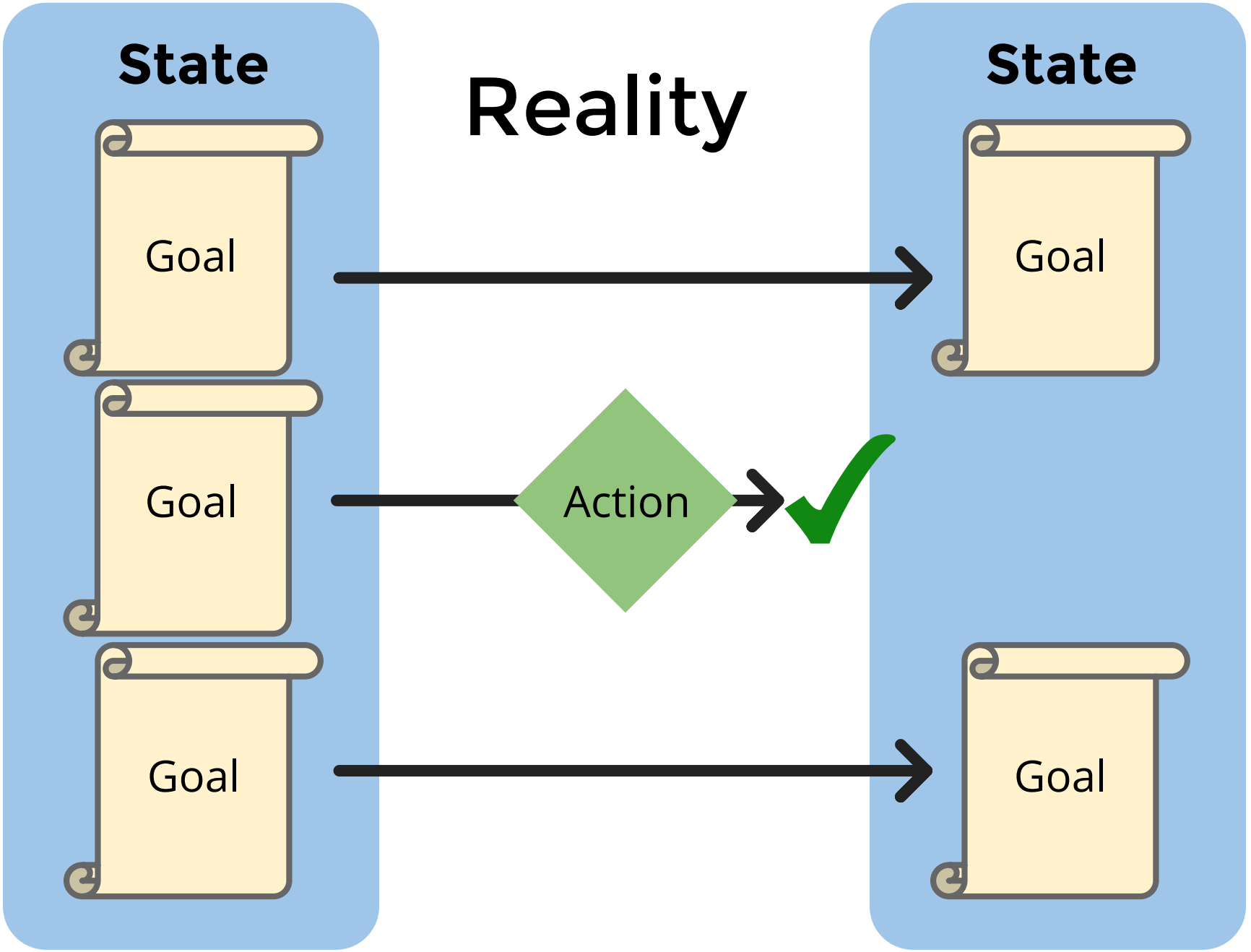
Abstraction




State
Reality
State
Goal
Action

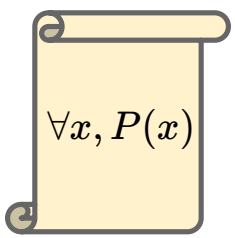

State
Reality
State
Goal
Action
Goal
Goal
Goal


State
Reality
State
Goal
Goal
Goal
Action

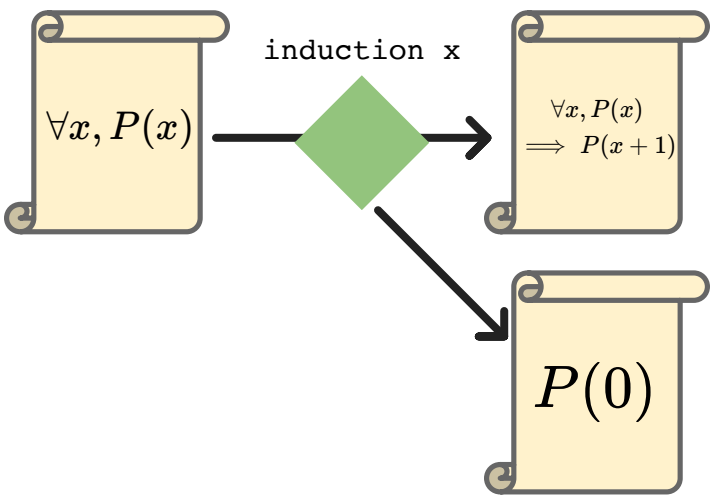

State
Reality
State
Goal
Goal
Action
Goal
Goal
Goal

Adding extra rewards can lead to bad behavior

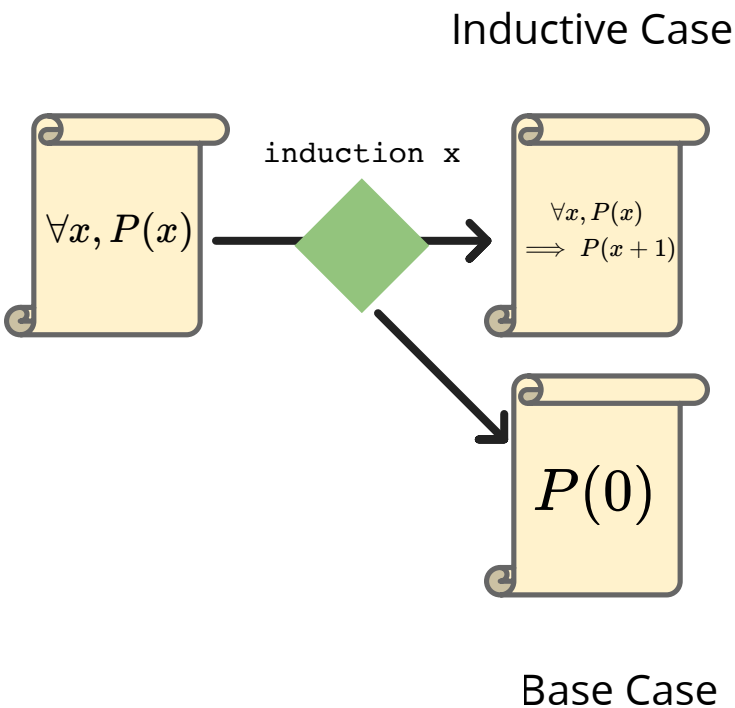
Adding extra rewards can lead to bad behavior



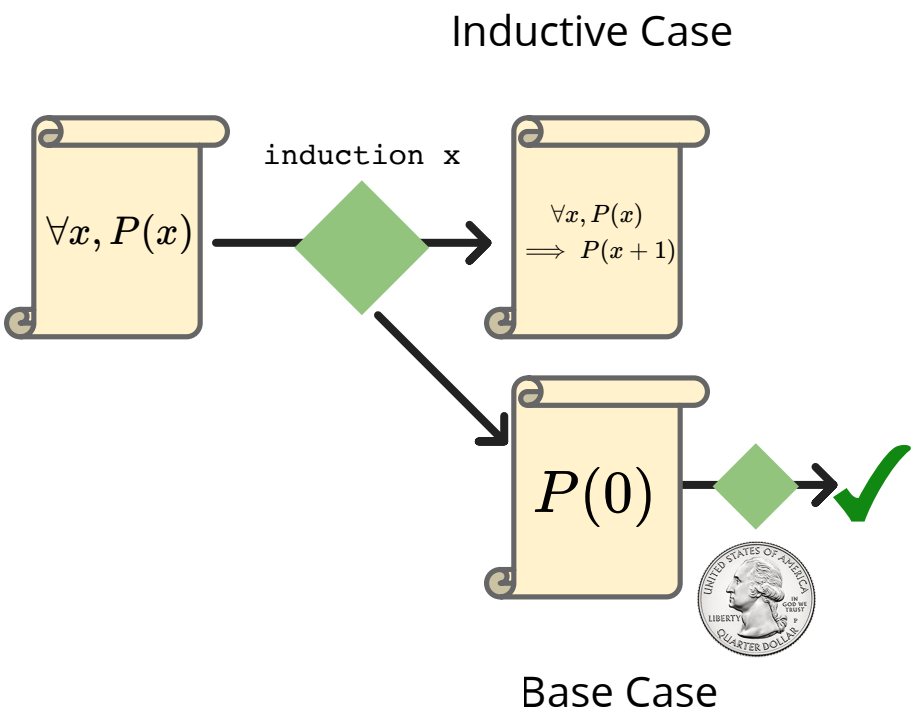
Adding extra rewards can lead to bad behavior



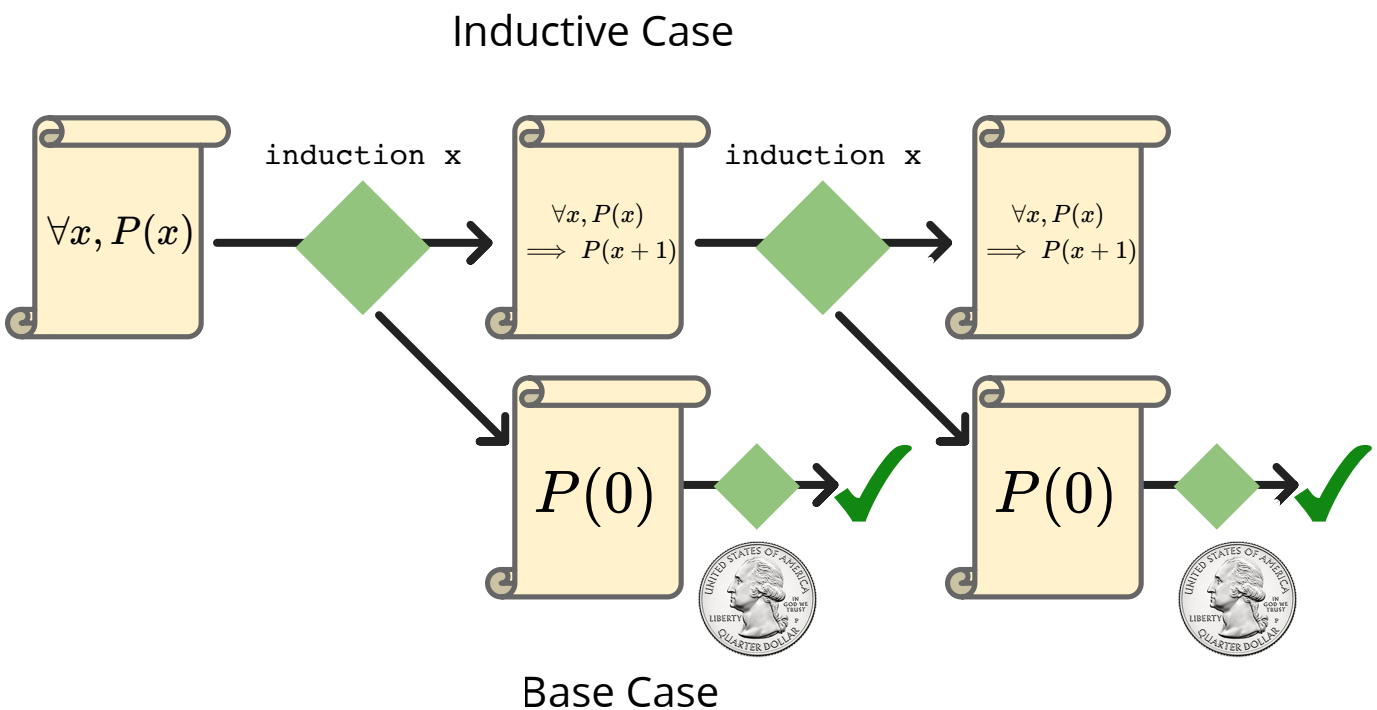
Adding extra rewards can lead to bad behavior



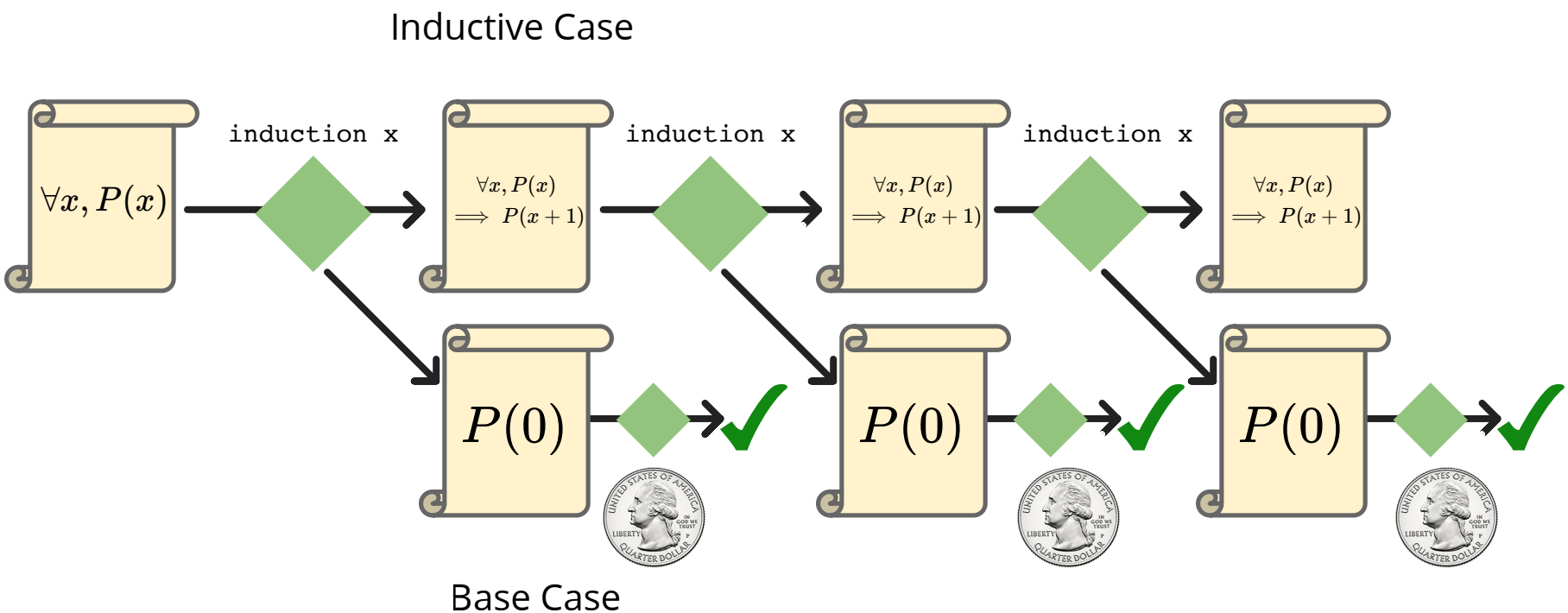
Adding extra rewards can lead to bad behavior



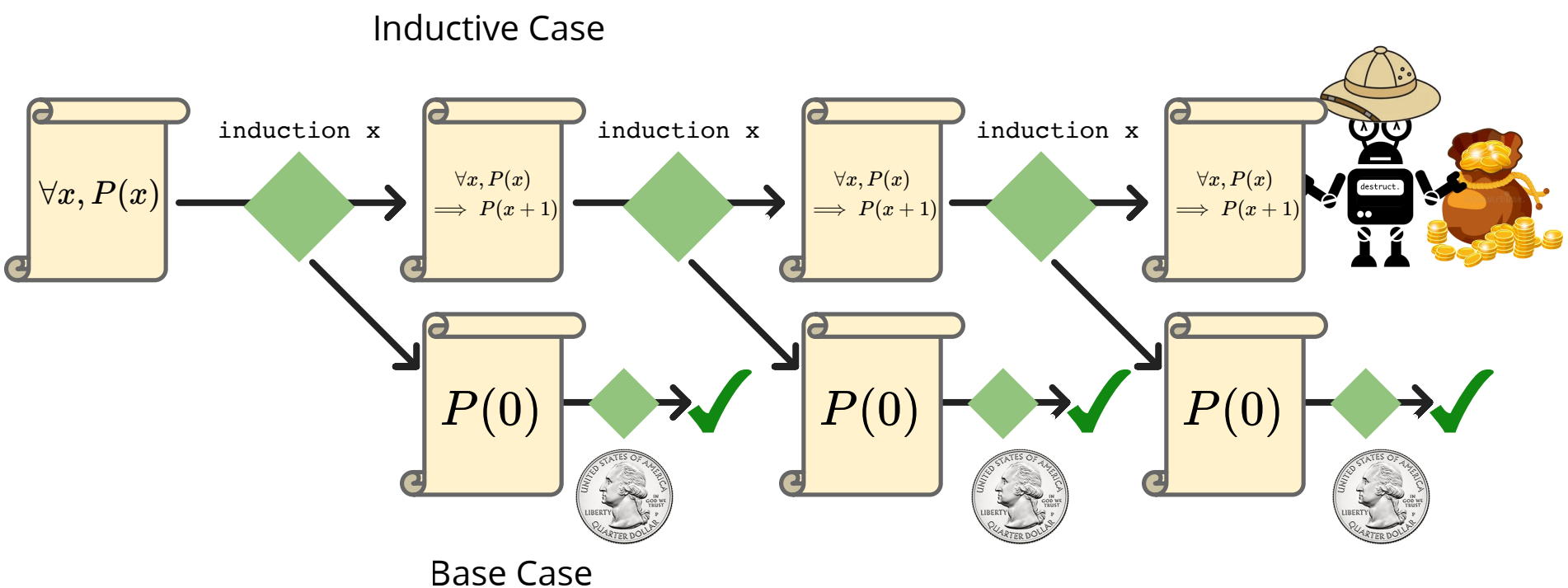
Adding extra rewards can lead to bad behavior



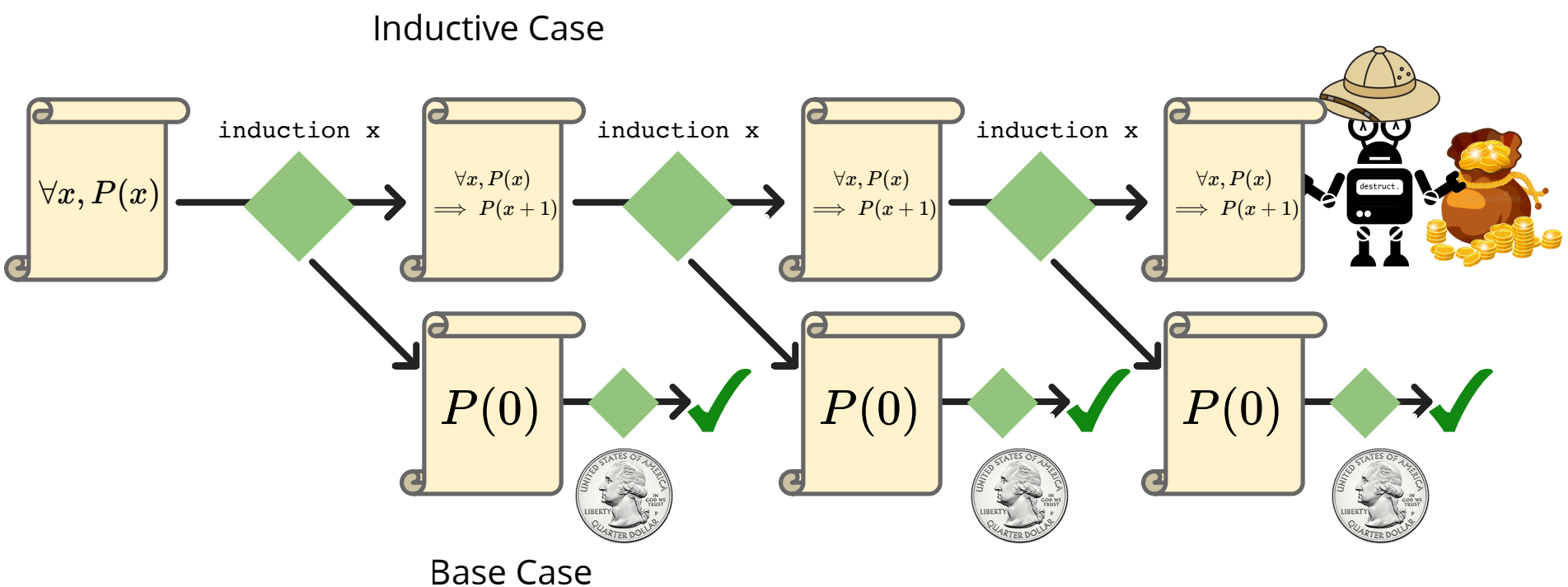
Adding extra rewards can lead to bad behavior



Adding extra rewards can lead to bad behavior



Adding extra rewards can lead to bad behavior



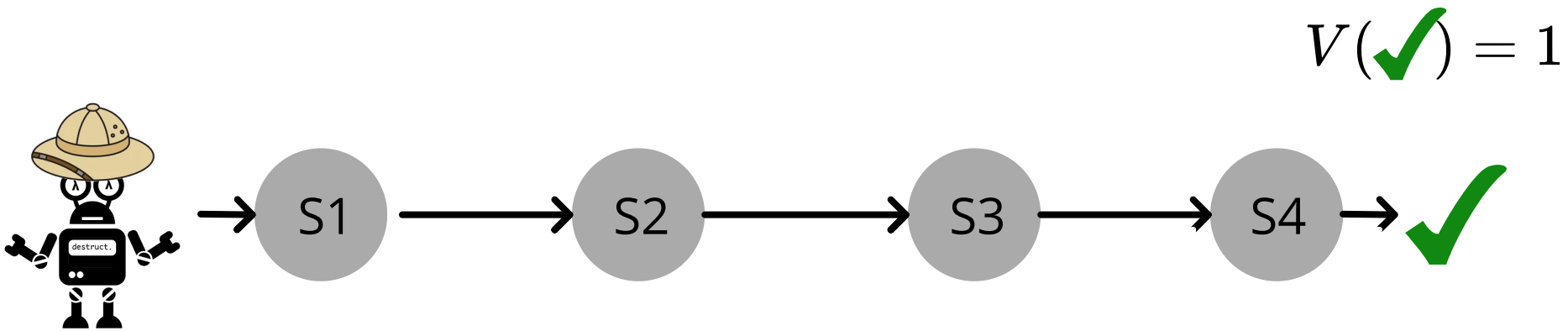
Reward-free doesn't have this problem!

What We Need

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

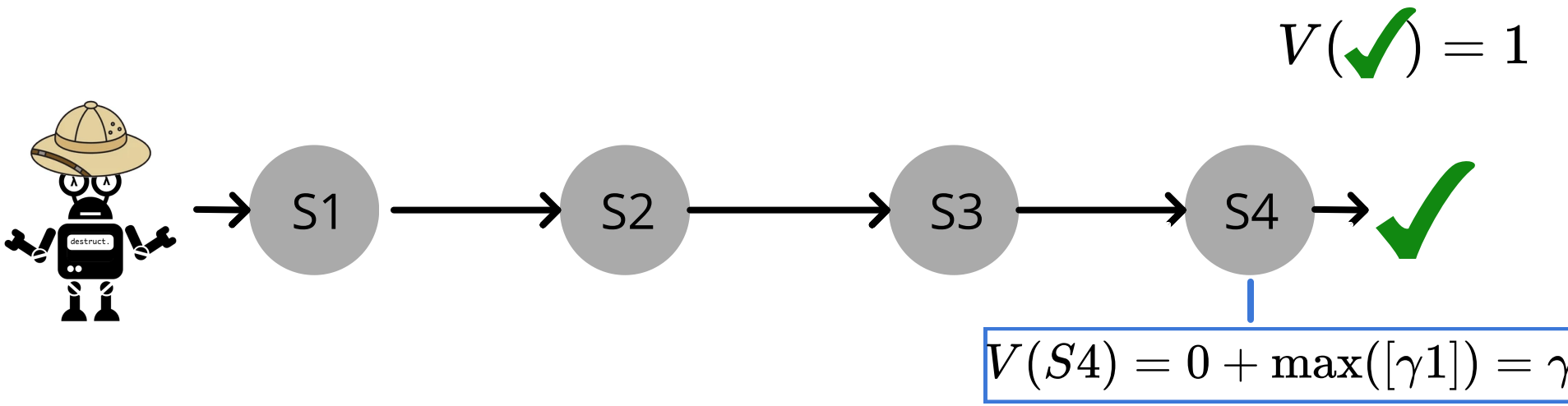
Assumptions:

- The state of a completed proof has value 1



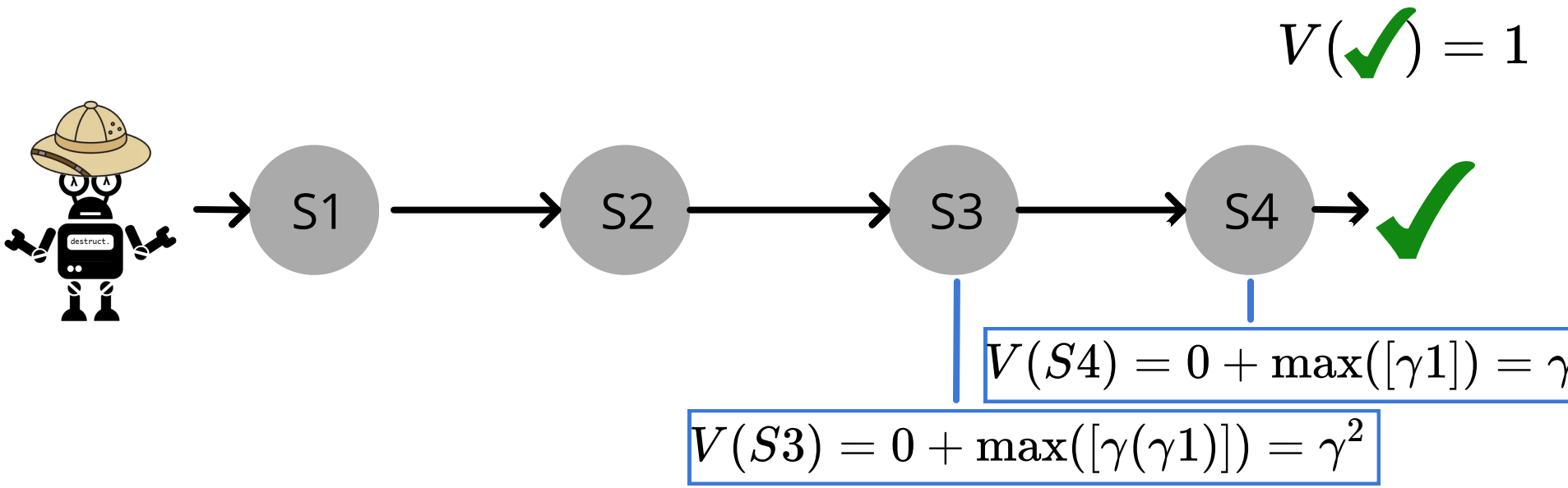
Assumptions:

- The state of a completed proof has value 1



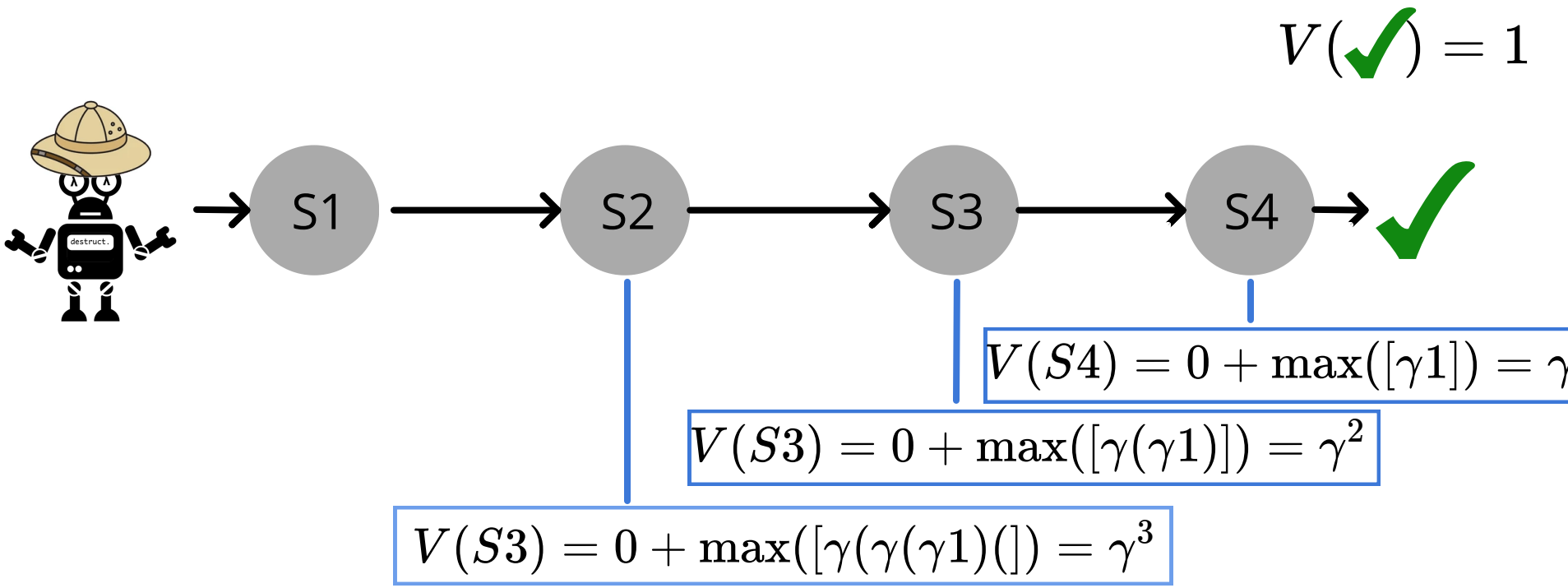
Assumptions:

- The state of a completed proof has value 1



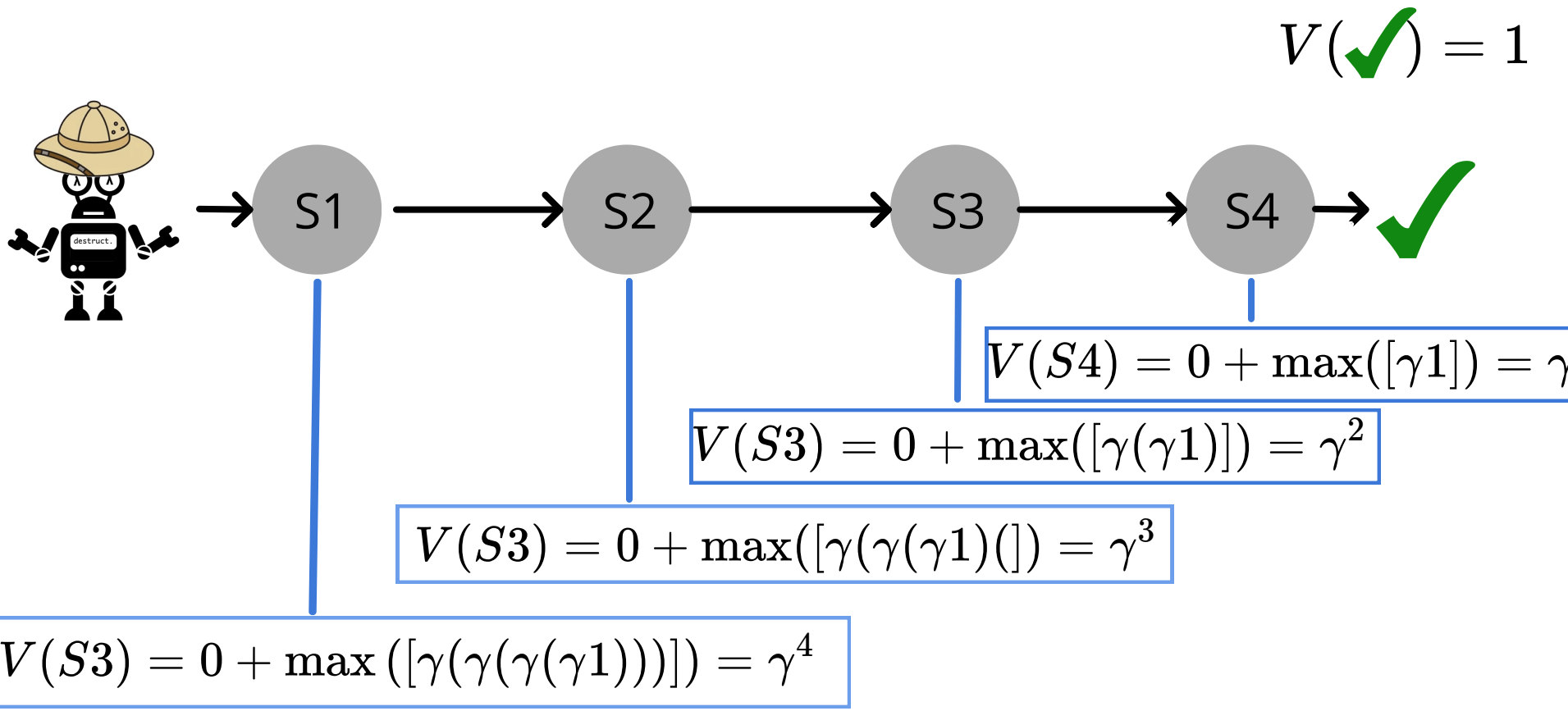
Assumptions:

- The state of a completed proof has value 1



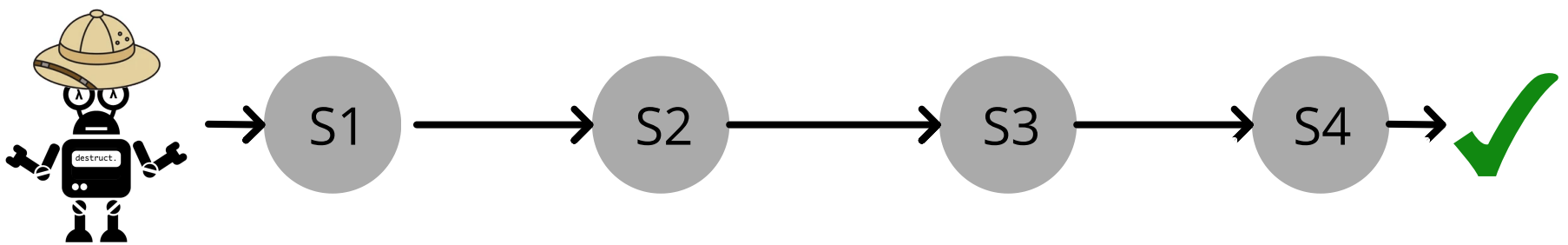
Assumptions:

- The state of a completed proof has value 1

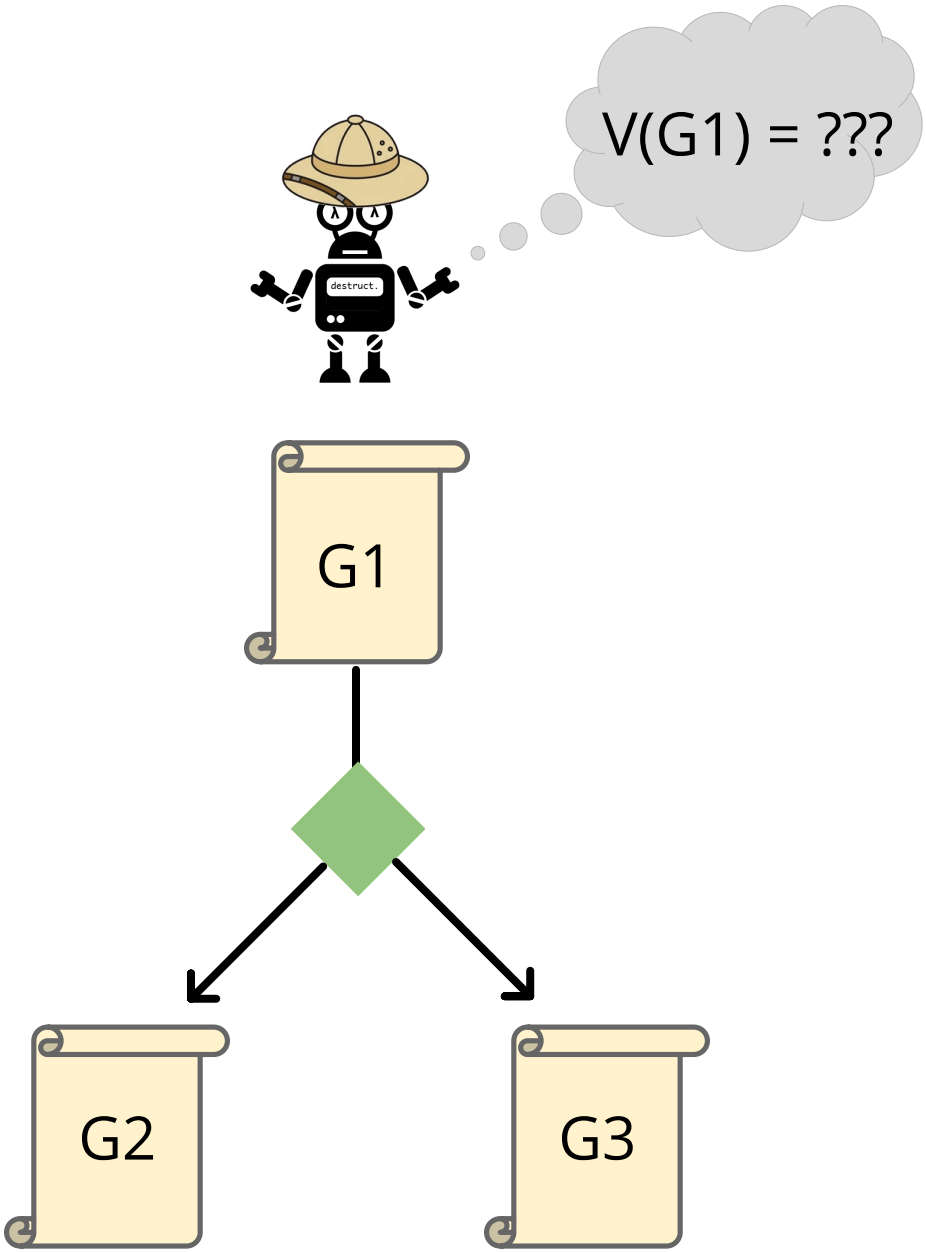


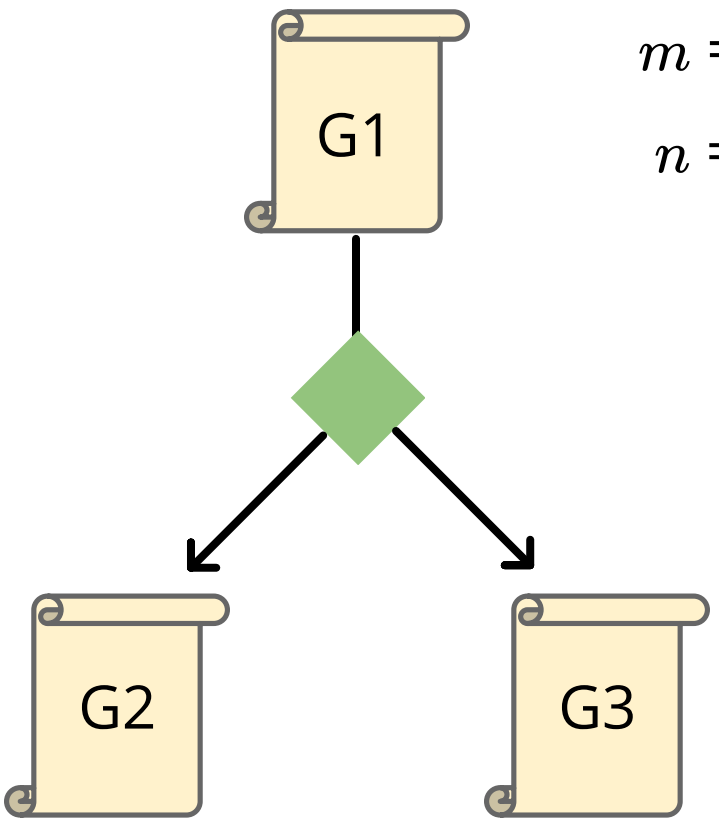
Assumption: The state of a completed proof has value 1

$V(\checkmark) = 1$



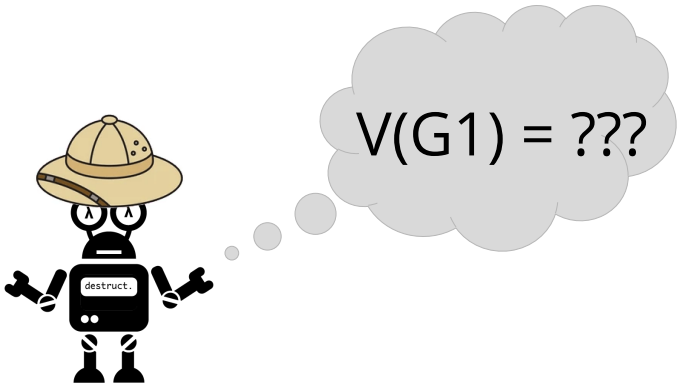
$$V(S) = \gamma^{(\text{number of steps left})}$$

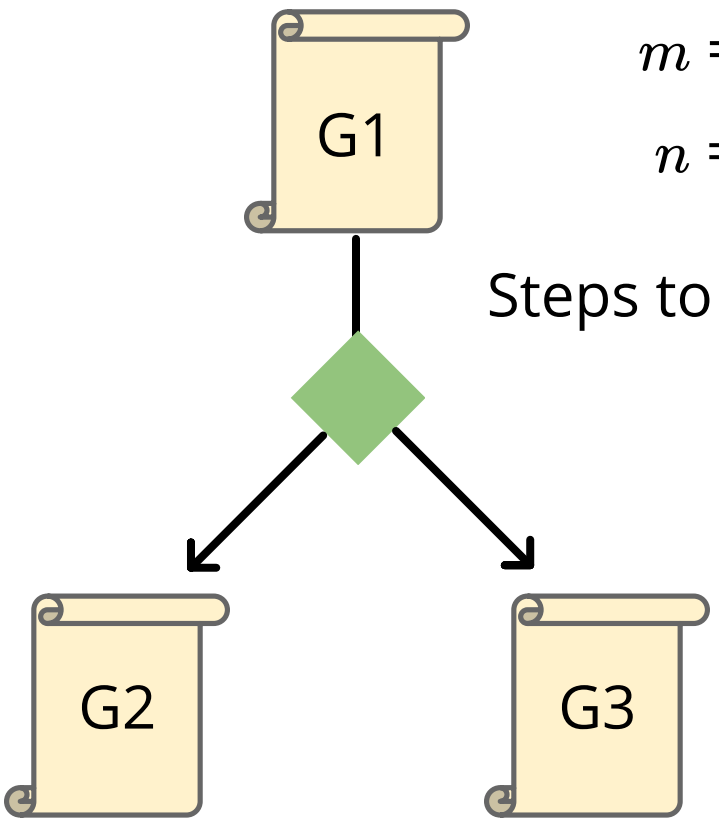

V(G1) = ???
destruct.
G1
G2
G3



m = Steps to complete proof from G2

n = Steps to complete proof from G3

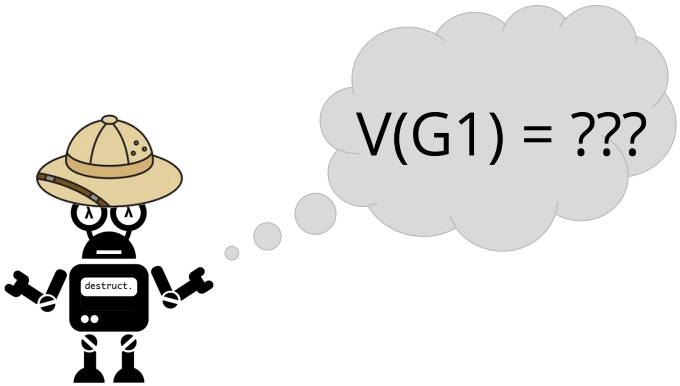


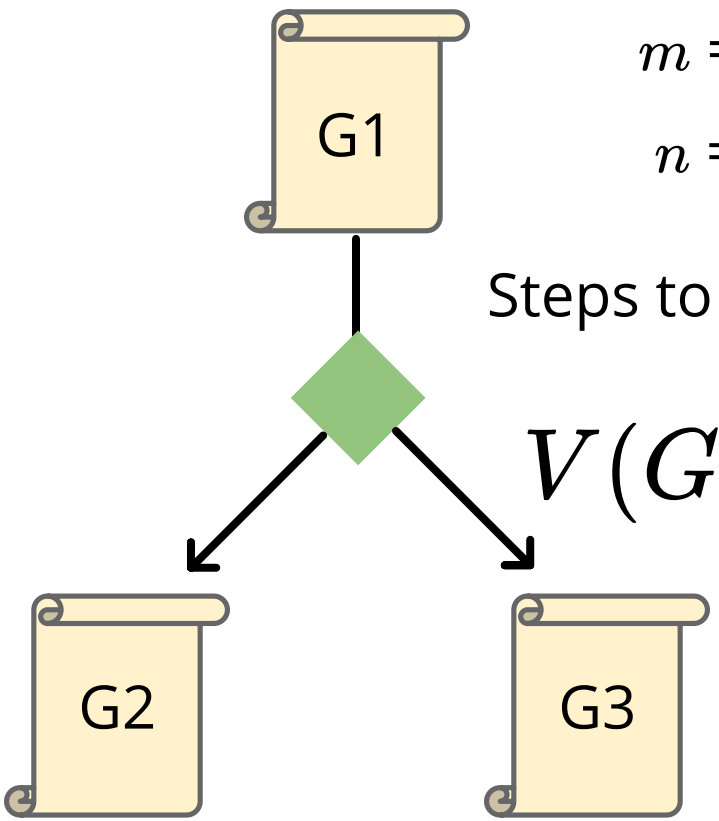


m = Steps to complete proof from G2

n = Steps to complete proof from G3

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





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

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

Update Equation for Branch-Structured Proofs

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

Update Equation for Branch-Structured Proofs

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

Update Equation for Branch-Structured Proofs

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

Update Equation for Branch-Structured Proofs

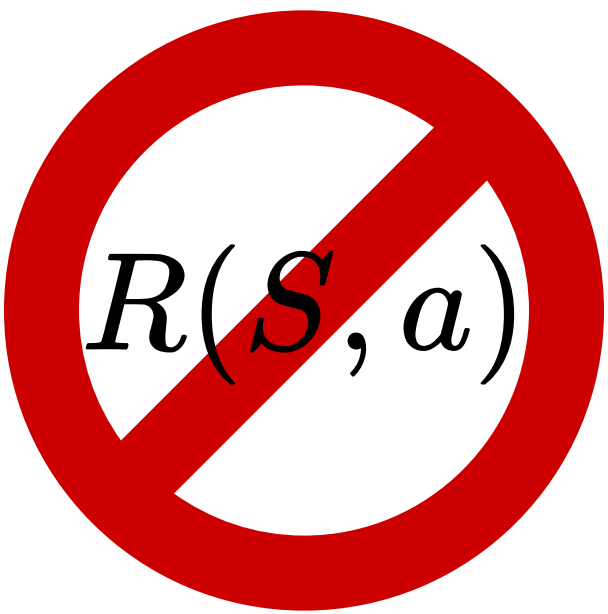
$$V(G) = \max_{a \in \text{actions}(G)} \left(\boxed{\gamma} \prod_{G' \in \text{next-state}(G,a)} V(G') \right)$$

Update Equation for Branch-Structured Proofs

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

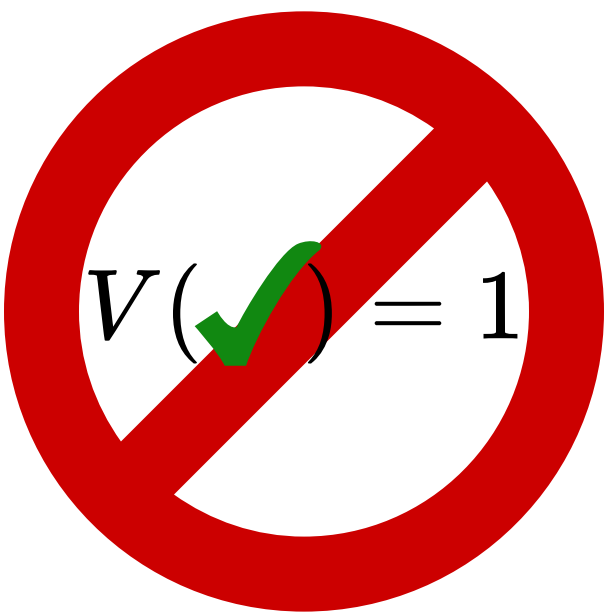
Update Equation for Branch-Structured Proofs

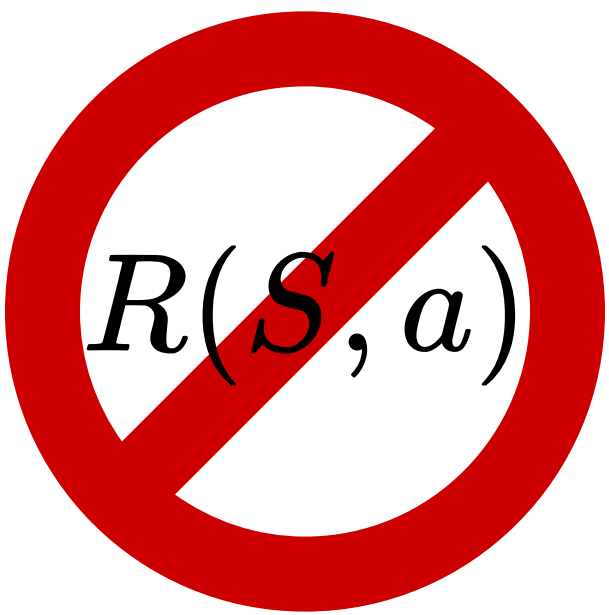
$$V(G) = \max_{a \in \text{actions}(G)} \left(\gamma \prod_{G' \in \text{next-states}(G,a)} V(G') \right)$$



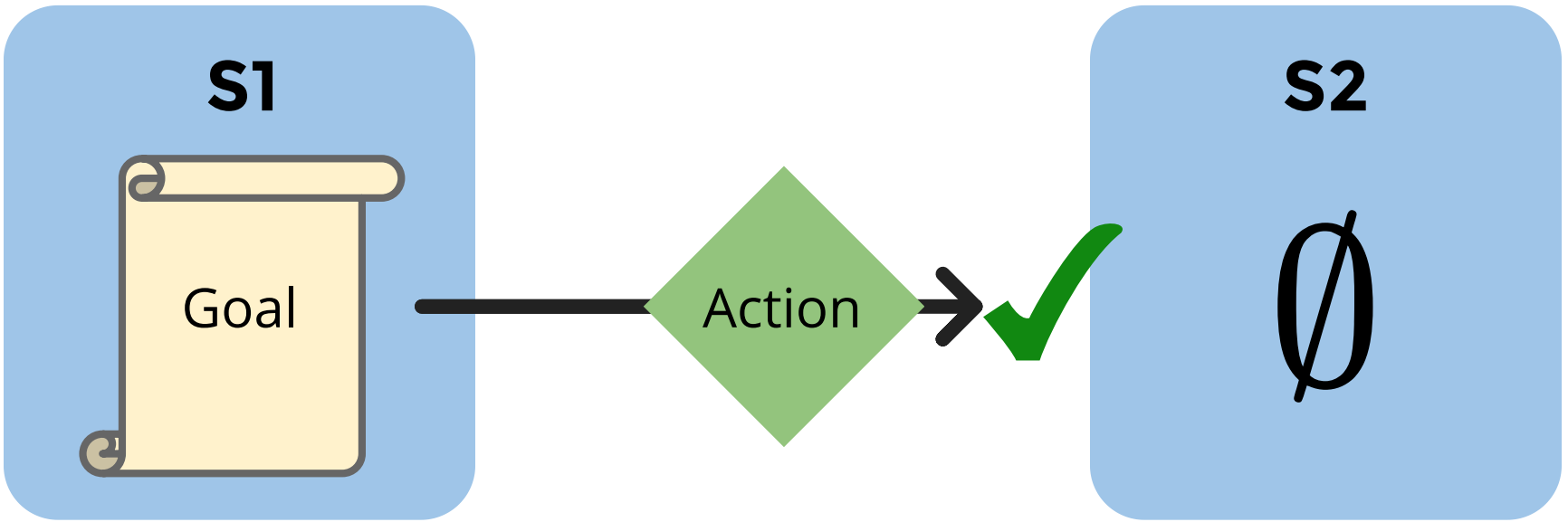
Update Equation for Branch-Structured Proofs

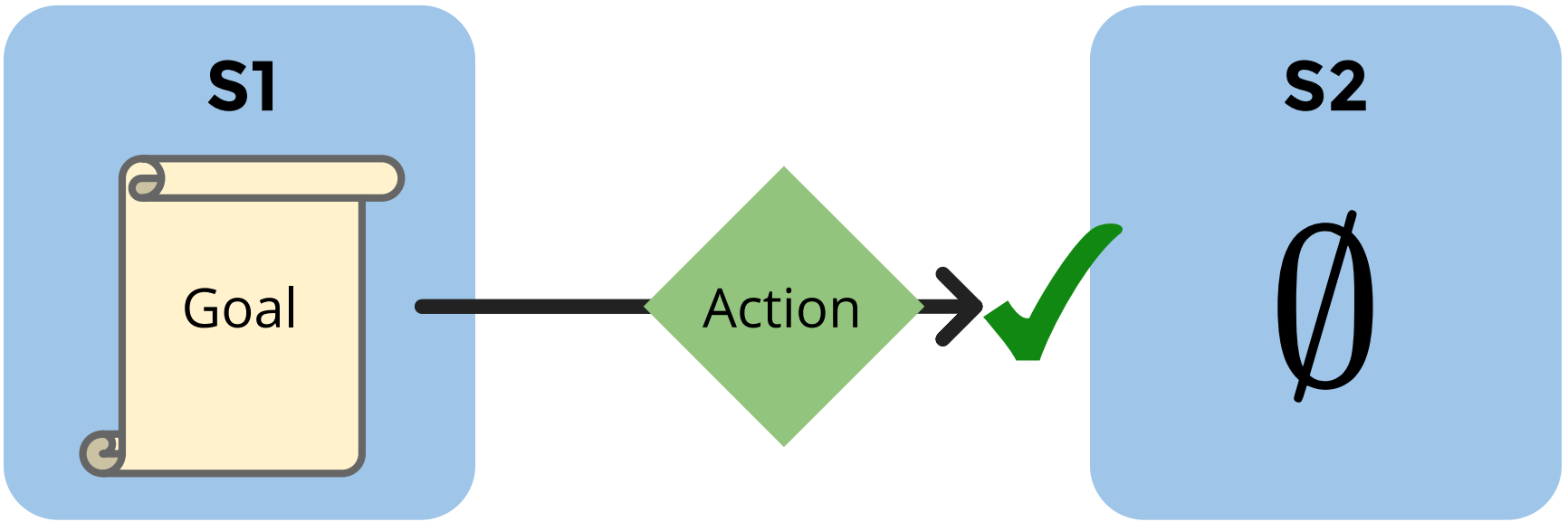
$$V(G) = \max_{a \in \text{actions}(G)} \left(\gamma \prod_{G' \in \text{next-states}(G,a)} V(G') \right)$$



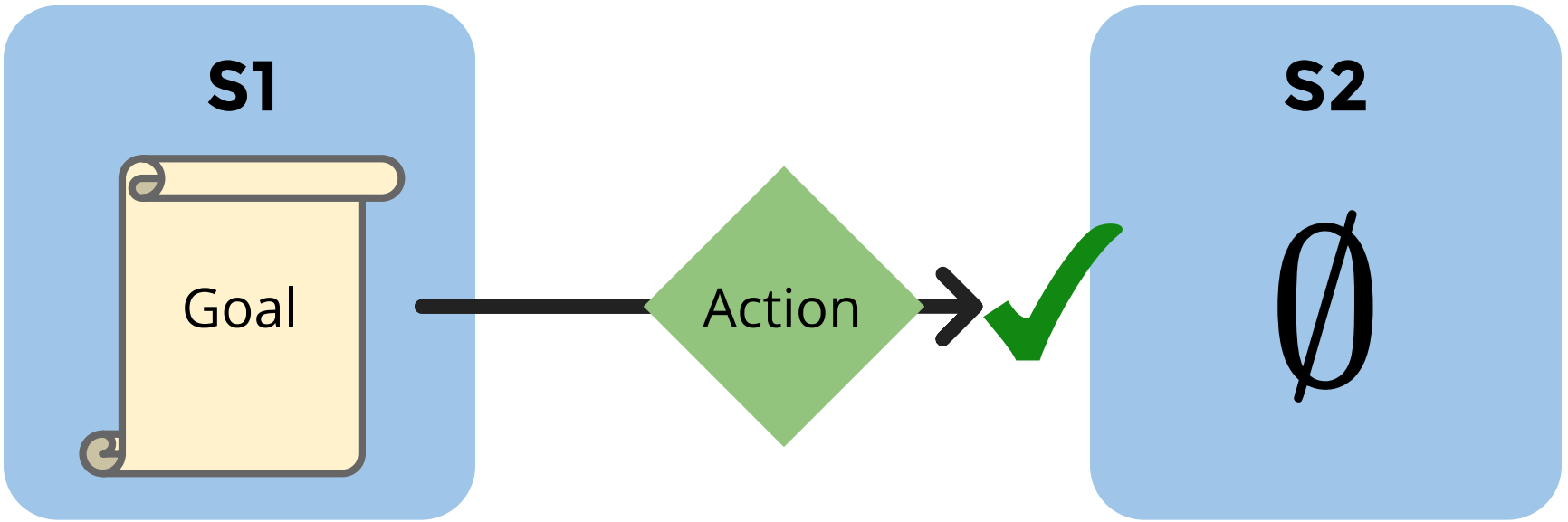



S1
Goal


S1
Goal
Action
S2
∅

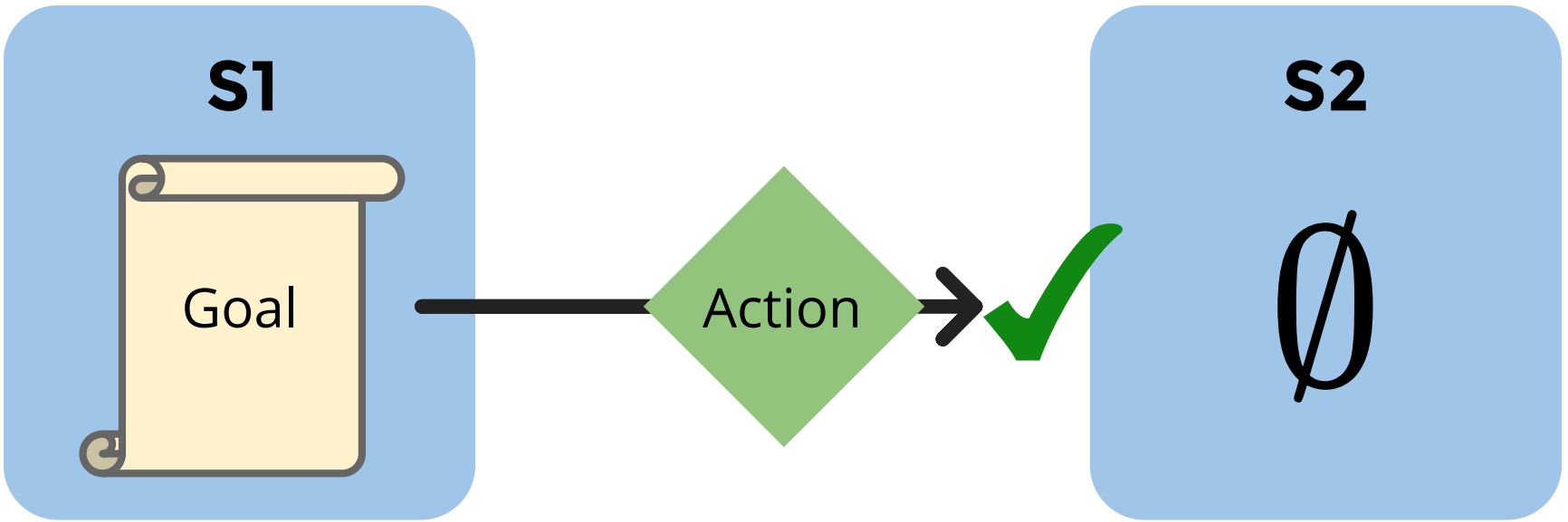


$$V(G) = \max_{a \in \text{actions}(G)} \left(\gamma \prod_{G' \in \text{next-states}(G,a)} V(G') \right)$$



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

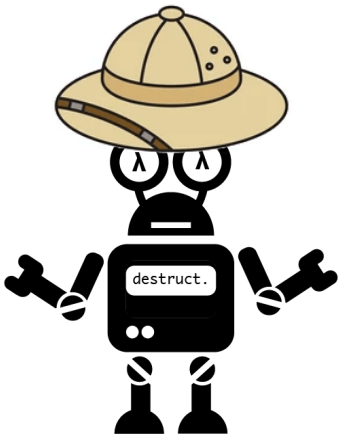
$$\gamma \prod_{G' \in \emptyset} V(G')$$

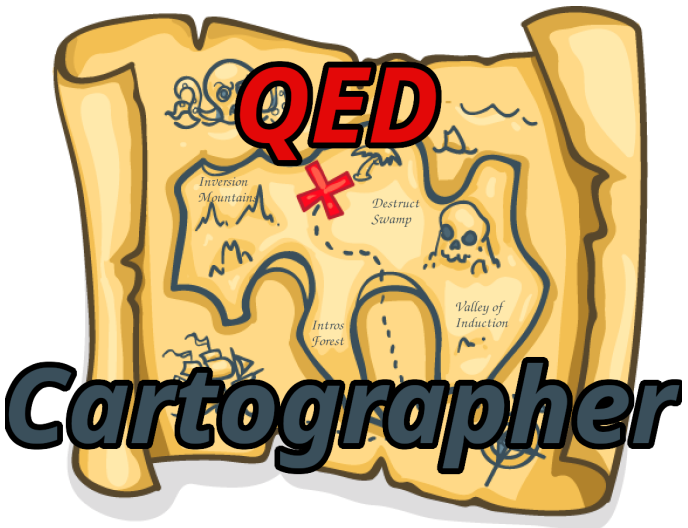


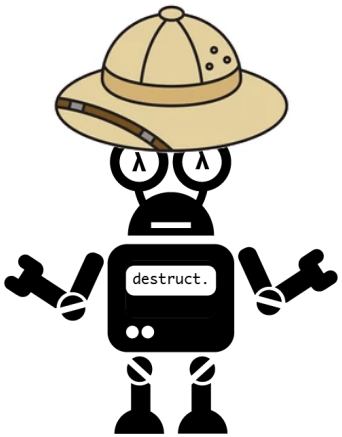
$$V(G) = \max_{a \in \text{actions}(G)} \left(\gamma \prod_{G' \in \text{next-states}(G,a)} V(G') \right)$$

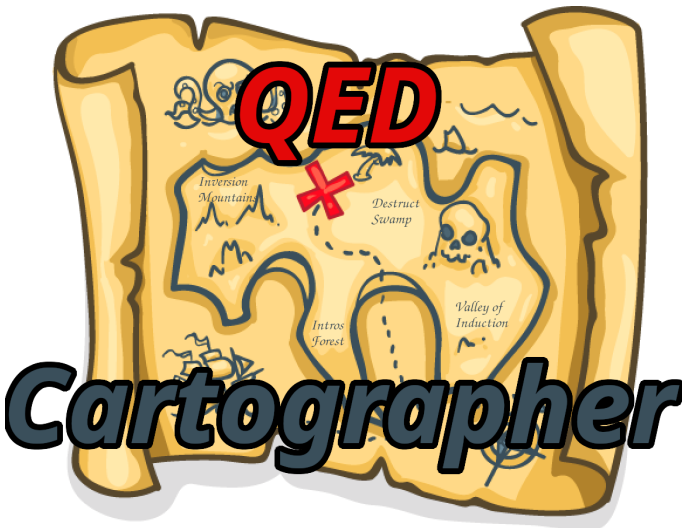
$$\gamma \prod_{G' \in \emptyset} V(G')$$

$$\gamma(1)$$


destruct.


QED
Inversion Mountain
Destruct Swamp
Intros Forest
Valley of Induction
Cartographer

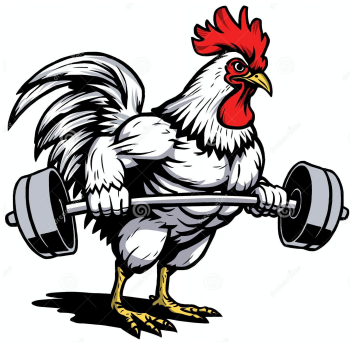




26% Shorter Proofs

in 27% Fewer Steps

Benchmark: CoqGym



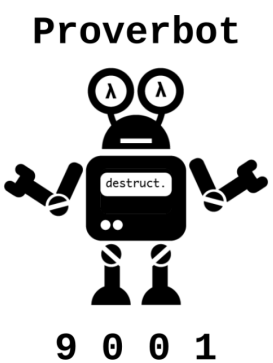
124 Coq Projects

68,501 Theorems

85/15 train-test split

Baseline: Proverbot9001 (updated)

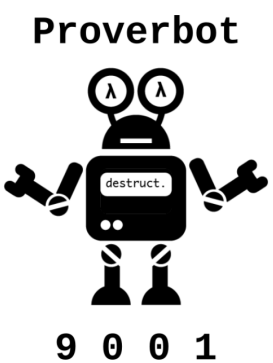
QEDCartographer, except without state scoring-based search

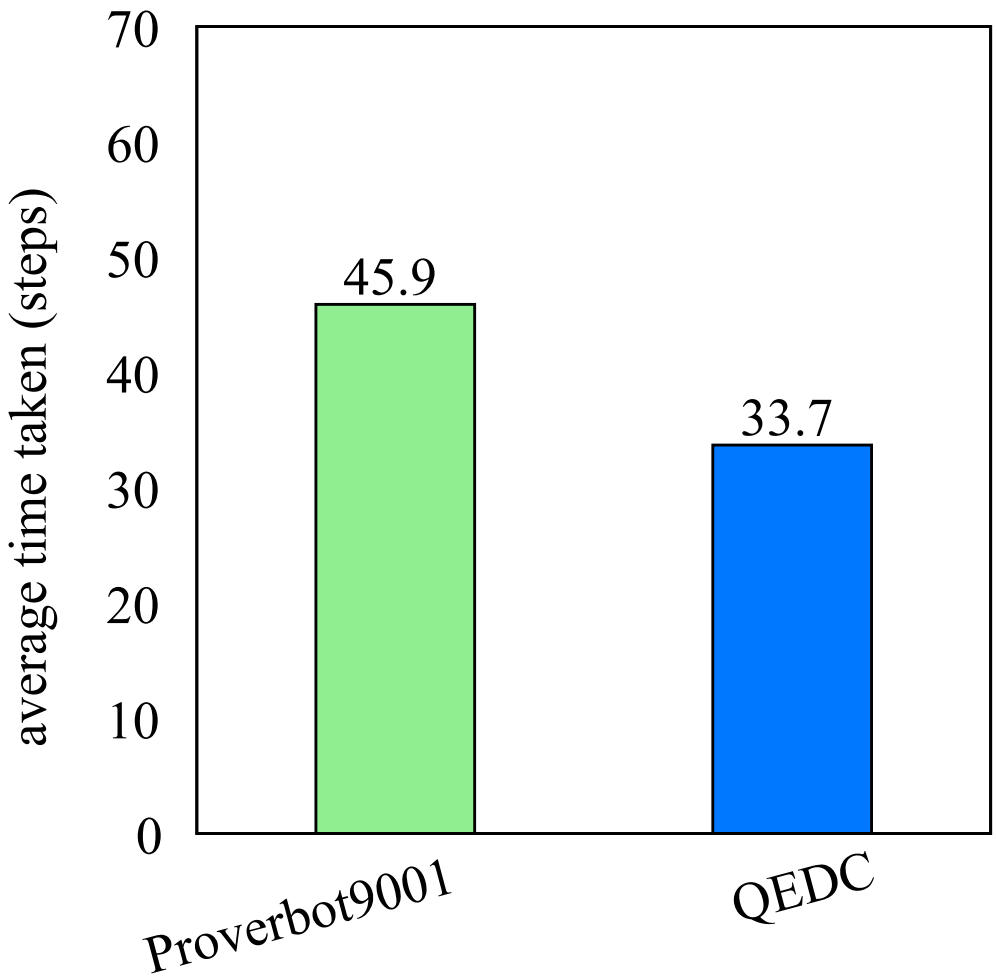


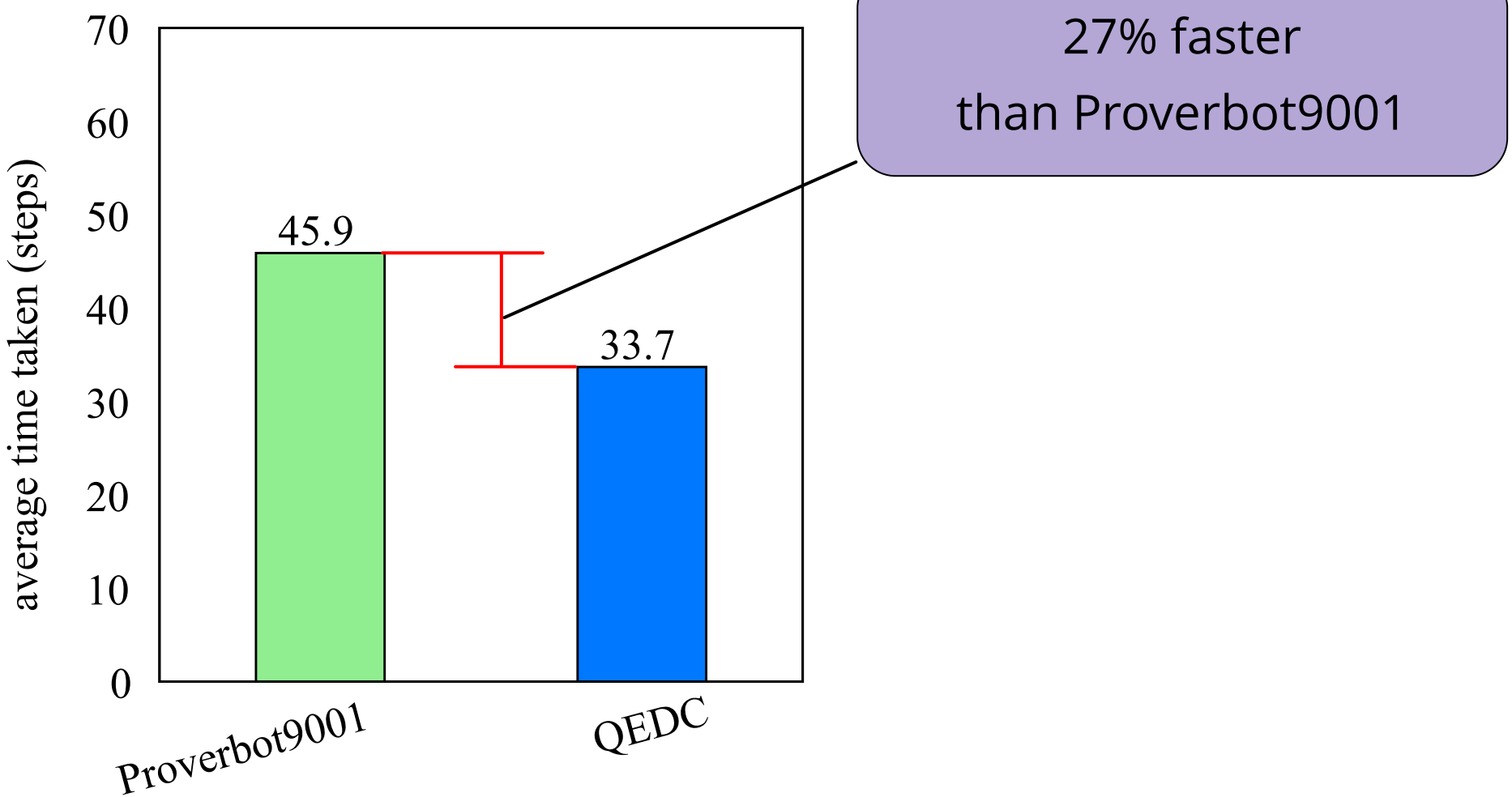
Baseline: Proverbot9001 (updated)

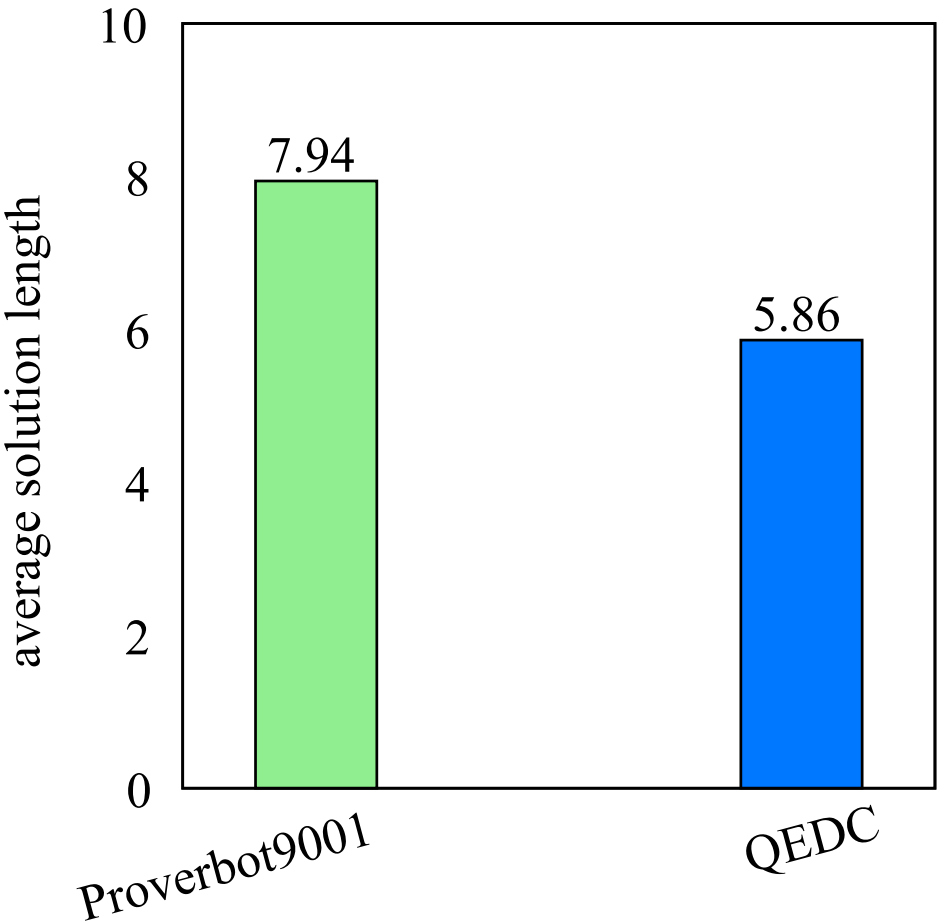
QEDCartographer, except without state scoring-based search

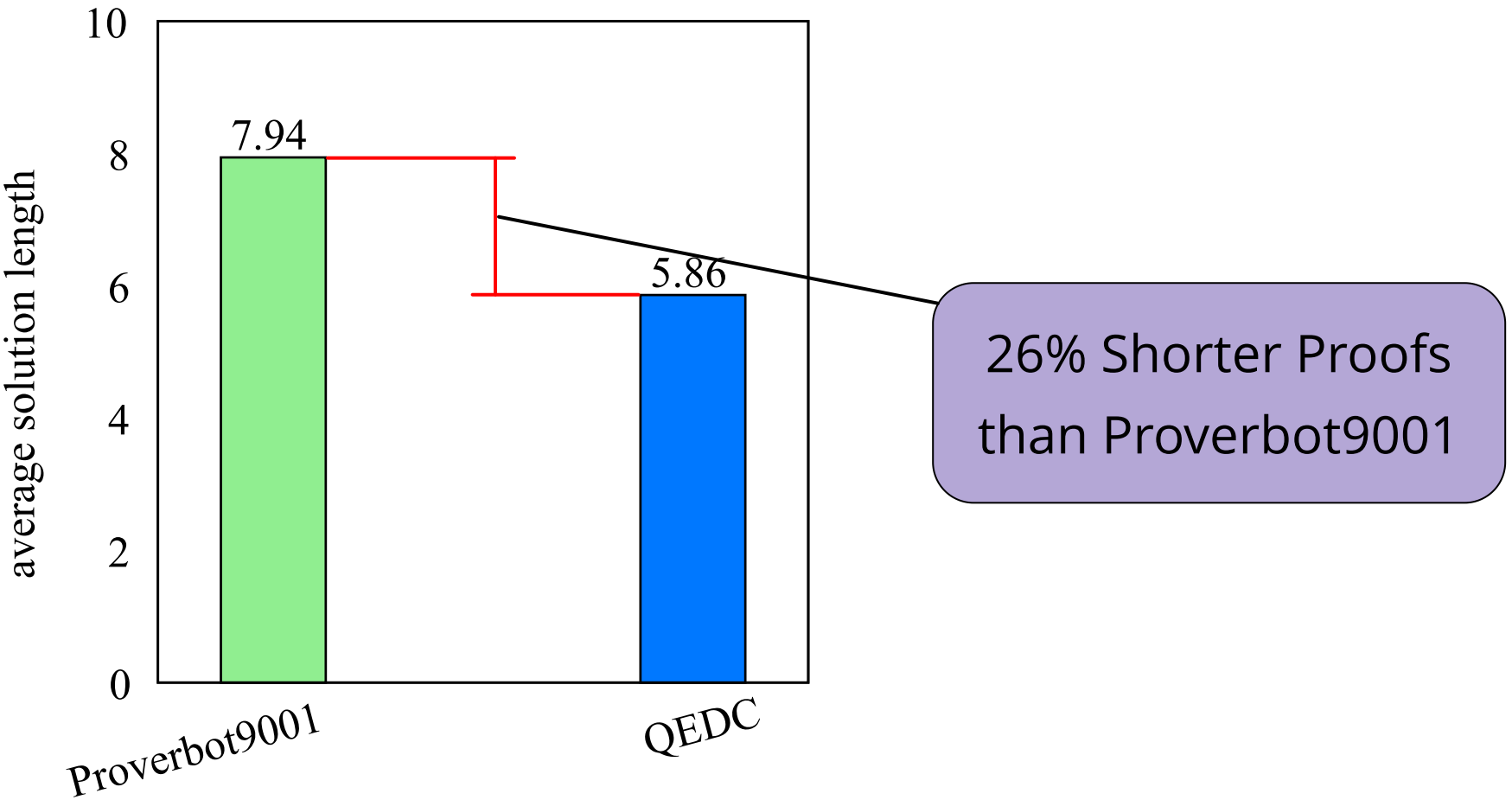
Uses a variant of depth-first search instead

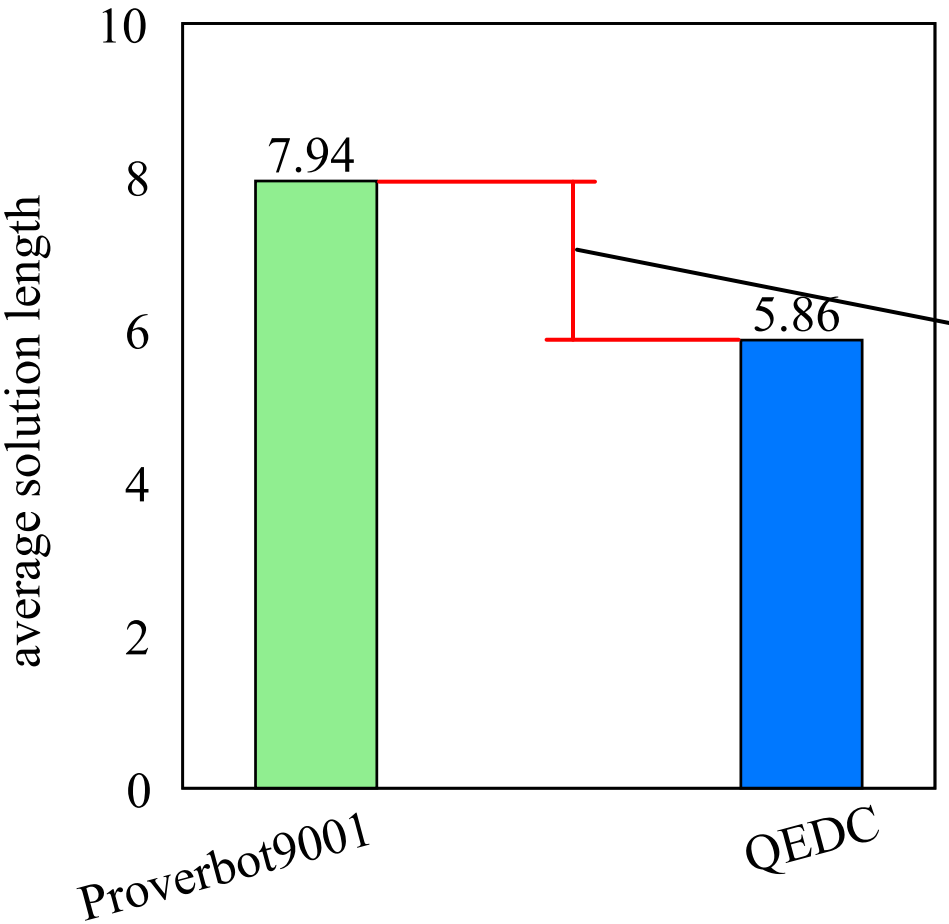



70
60
50
40
30
20
10
0
average time taken (steps)
45.9
33.7
Proverbot9001
QEDC


70
60
50
40
30
20
10
0
average time taken (steps)
45.9
33.7
Proverbot9001
QEDC
27% faster
than Proverbot9001


10
8
6
4
2
0
average solution length
7.94
5.86
Proverbot9001
QEDC

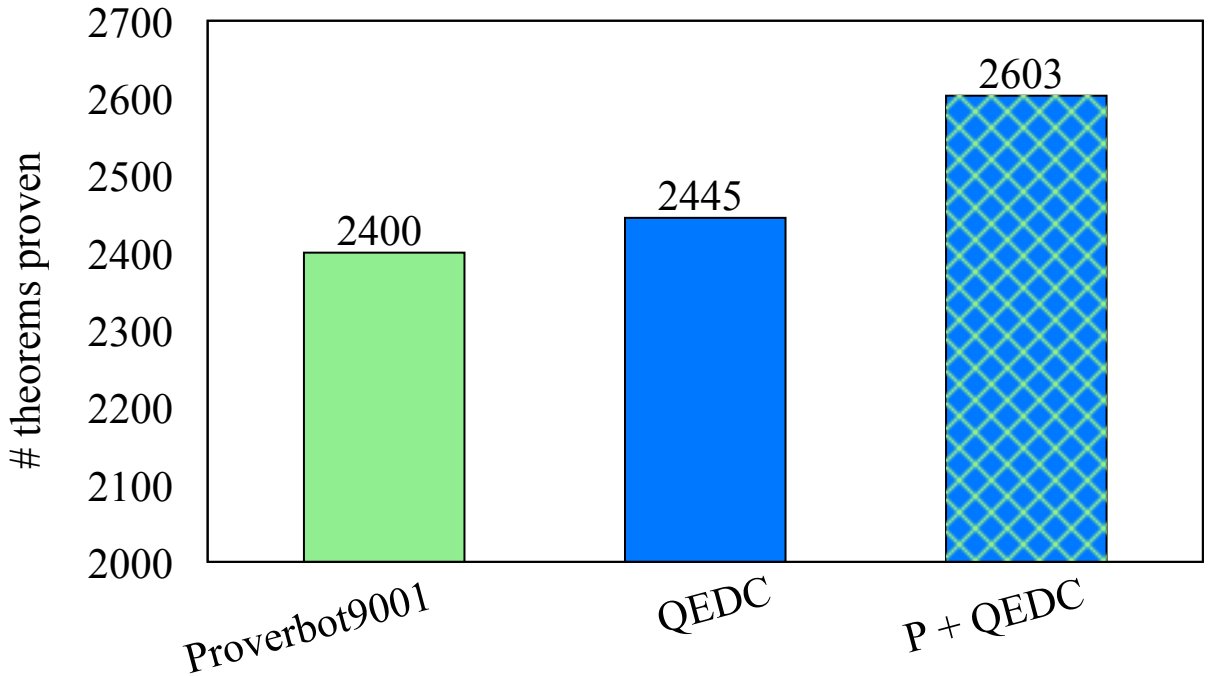

10
8
6
4
2
0
average solution length
7.94
5.86
Proverbot9001
QEDC
26% Shorter Proofs
than Proverbot9001

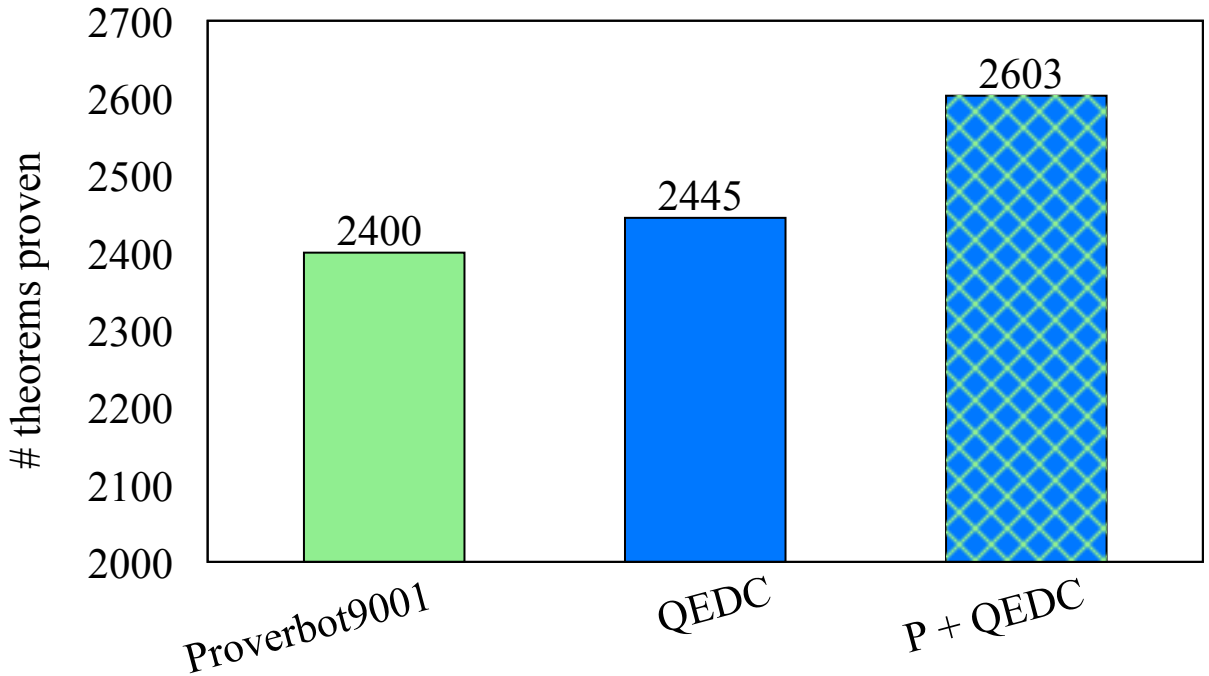


26% Shorter Proofs than Proverbot9001

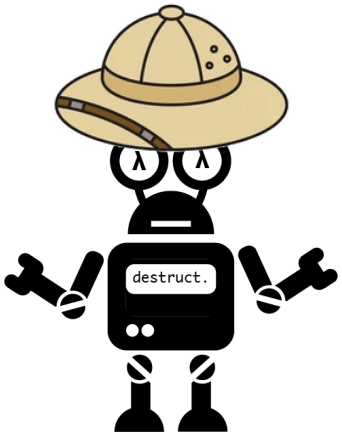
Shorter proofs are:

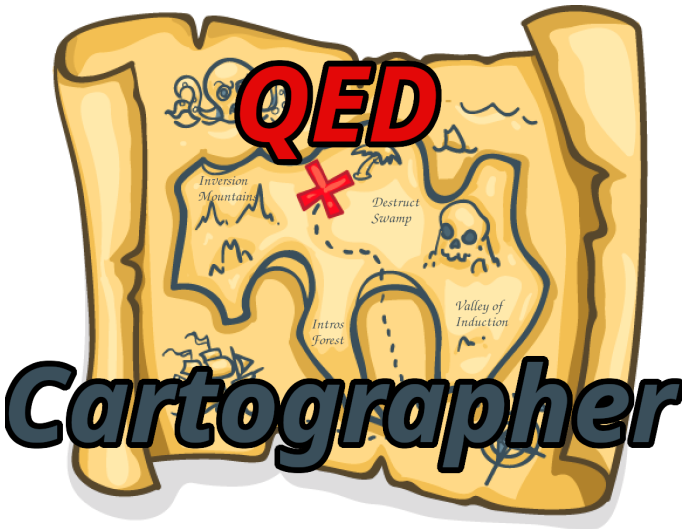
- More readable
- More maintainable
- Less expensive to check


theorems proven
2700
2600
2500
2400
2300
2200
2100
2000
2400
2445
2603
Proverbot9001
QEDC
P + QEDC



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