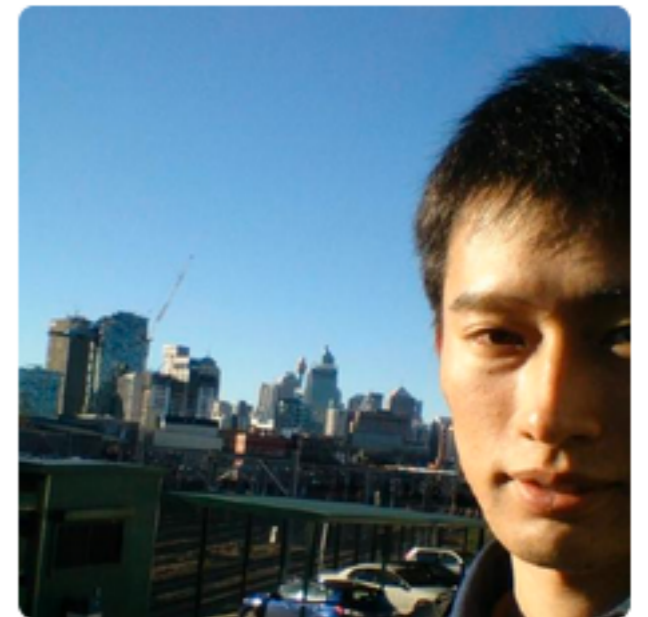


# Automating Induction for Isabelle/HOL with DSLs

*This work was supported by the project AI&Reasoning (reg. no. CZ.02.1.01/0.0/0.0/15\_003/0000466).*




Yutaka Nagashima  
University of Innsbruck  
Czech Technical University



**Yutaka Ng**

yutakang

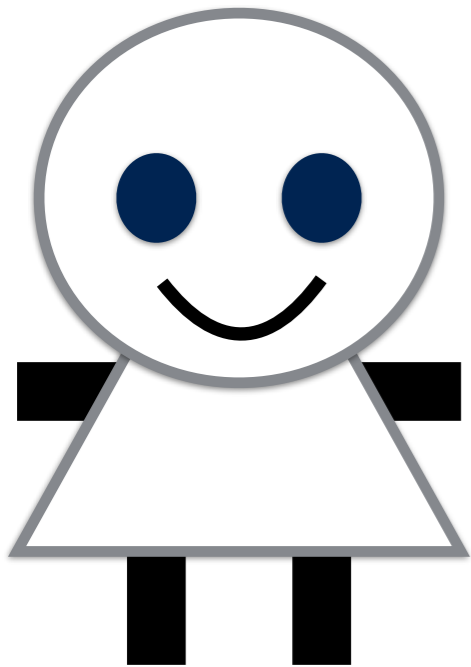
Block or report user

 CVUT, CTU, CIIRC



**CZECH INSTITUTE  
OF INFORMATICS  
ROBOTICS AND  
CYBERNETICS  
CTU IN PRAGUE**

# Interactive theorem proving with Isabelle/HOL



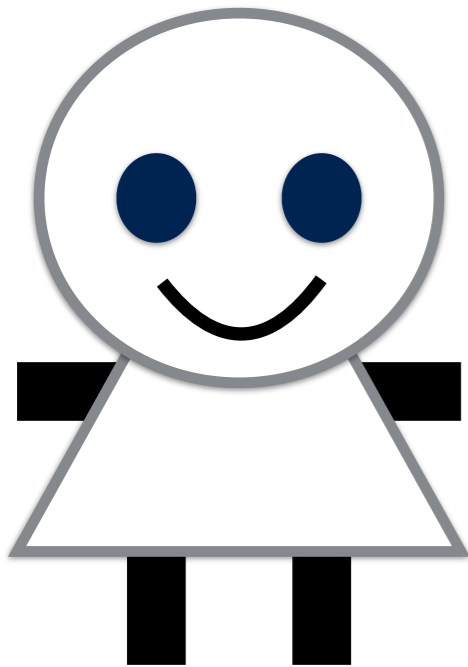
# Interactive theorem proving with

## Isabelle/HOL

proof goal

context

tactic / proof method



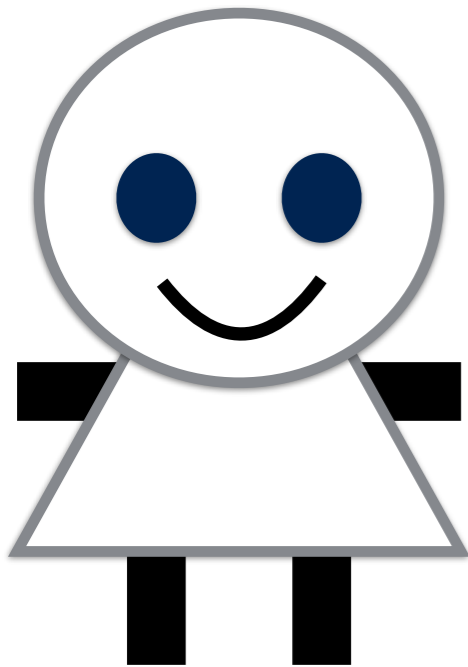
# Interactive theorem proving with

## Isabelle/HOL

proof goal

context

tactic / proof method



error-message

subgoals

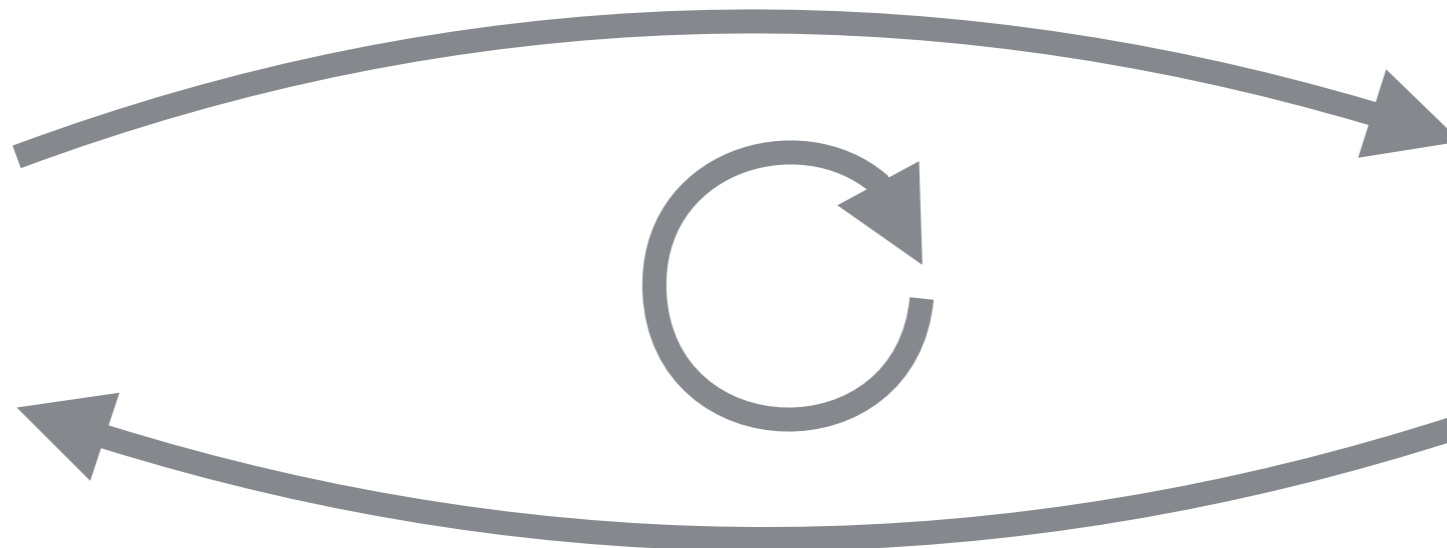
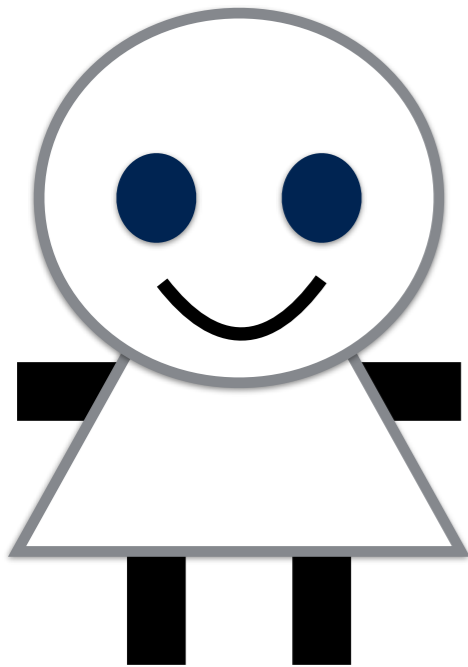
# Interactive theorem proving with

## Isabelle/HOL

proof goal

context

tactic / proof method



error-message

subgoals

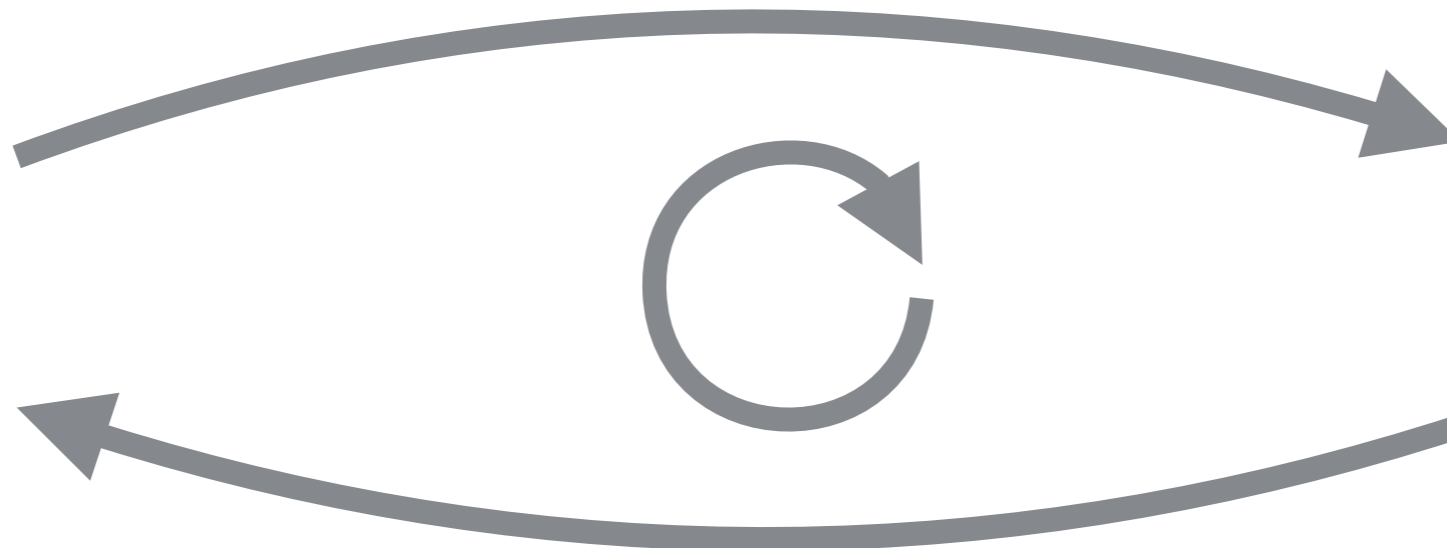
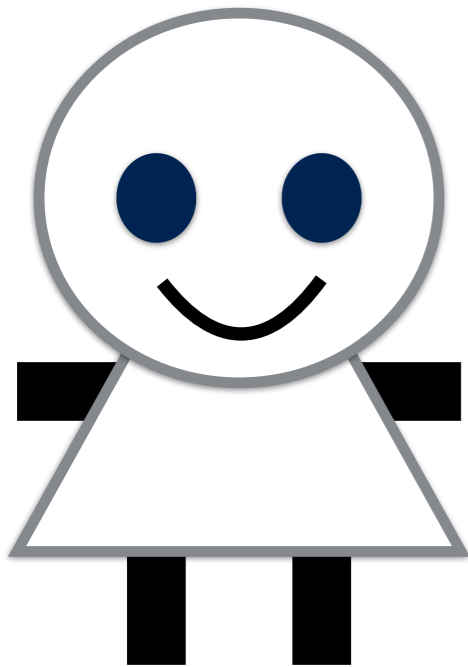
# Interactive theorem proving with

## Isabelle/HOL

proof goal

context

tactic / proof method



error-message

subgoals

no sub-goal!

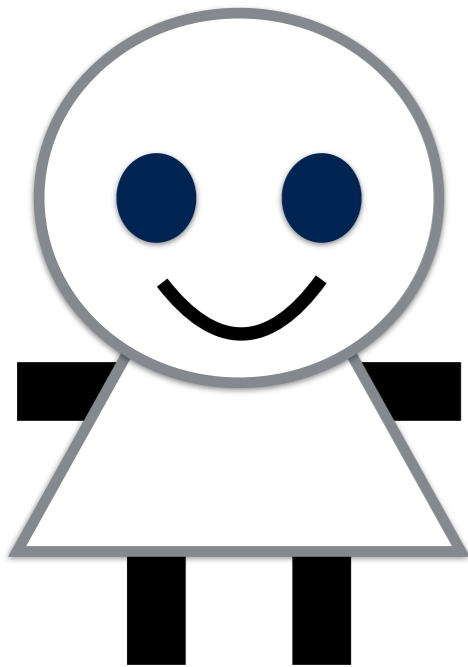
# Interactive theorem proving with

## Isabelle/HOL

proof goal

context

tactic / proof method



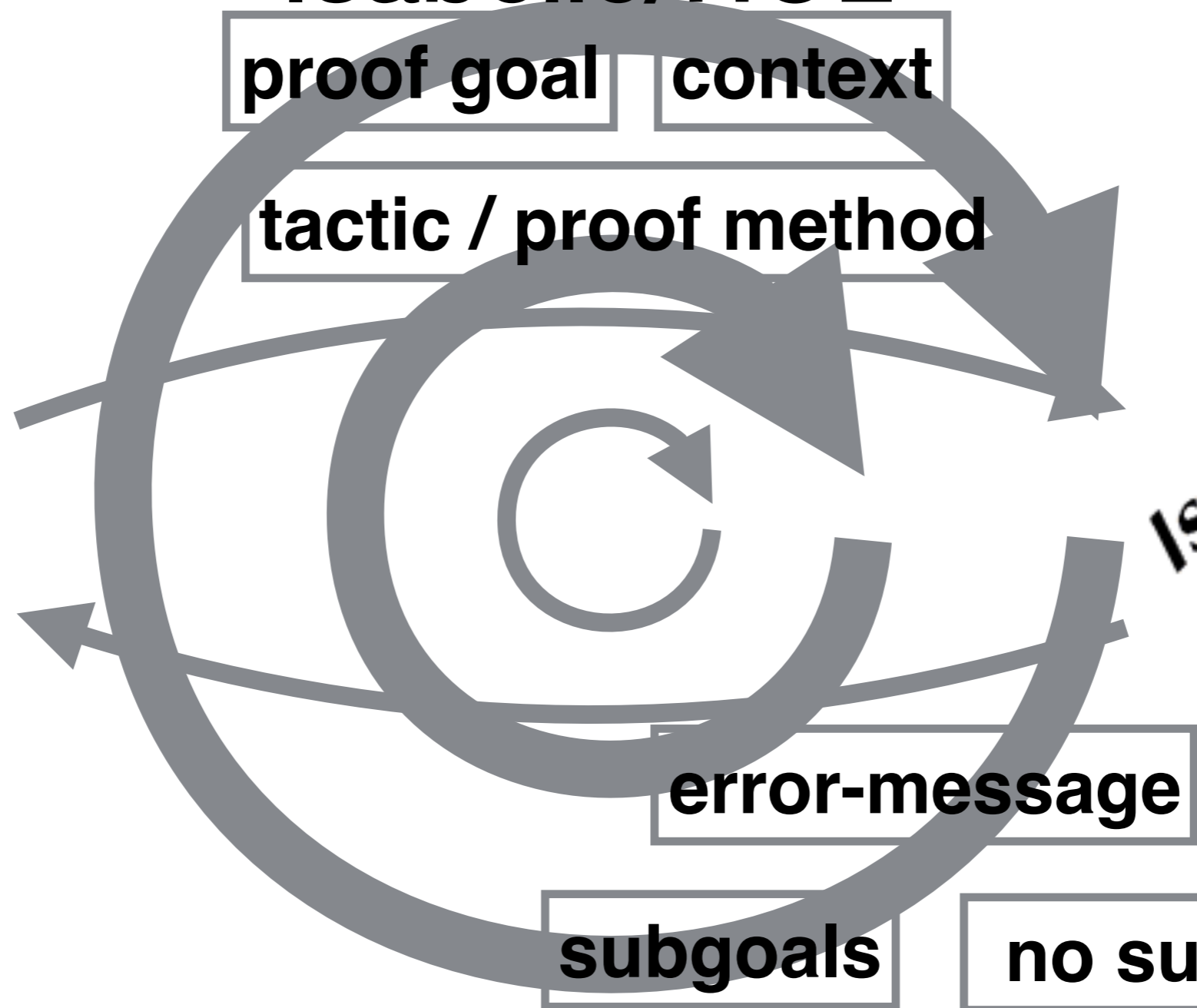
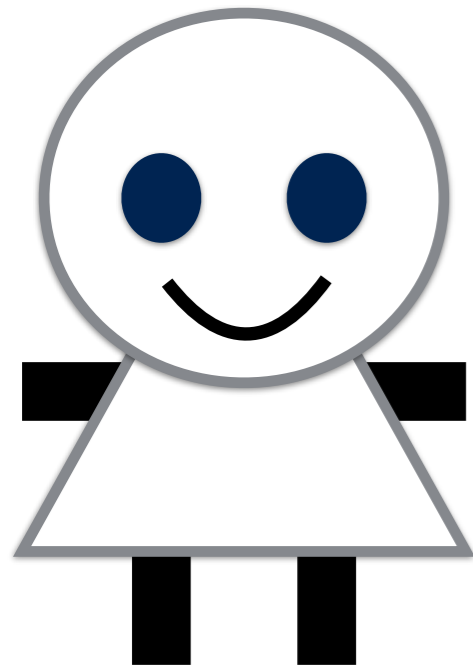
error-message

subgoals

no sub-goal!

# Interactive theorem proving with

## Isabelle/HOL



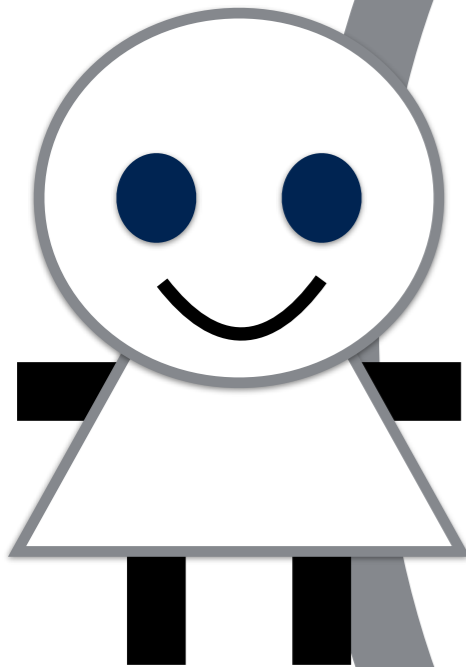


# Interactive theorem proving with

## Isabelle/HOL

proof goal | context

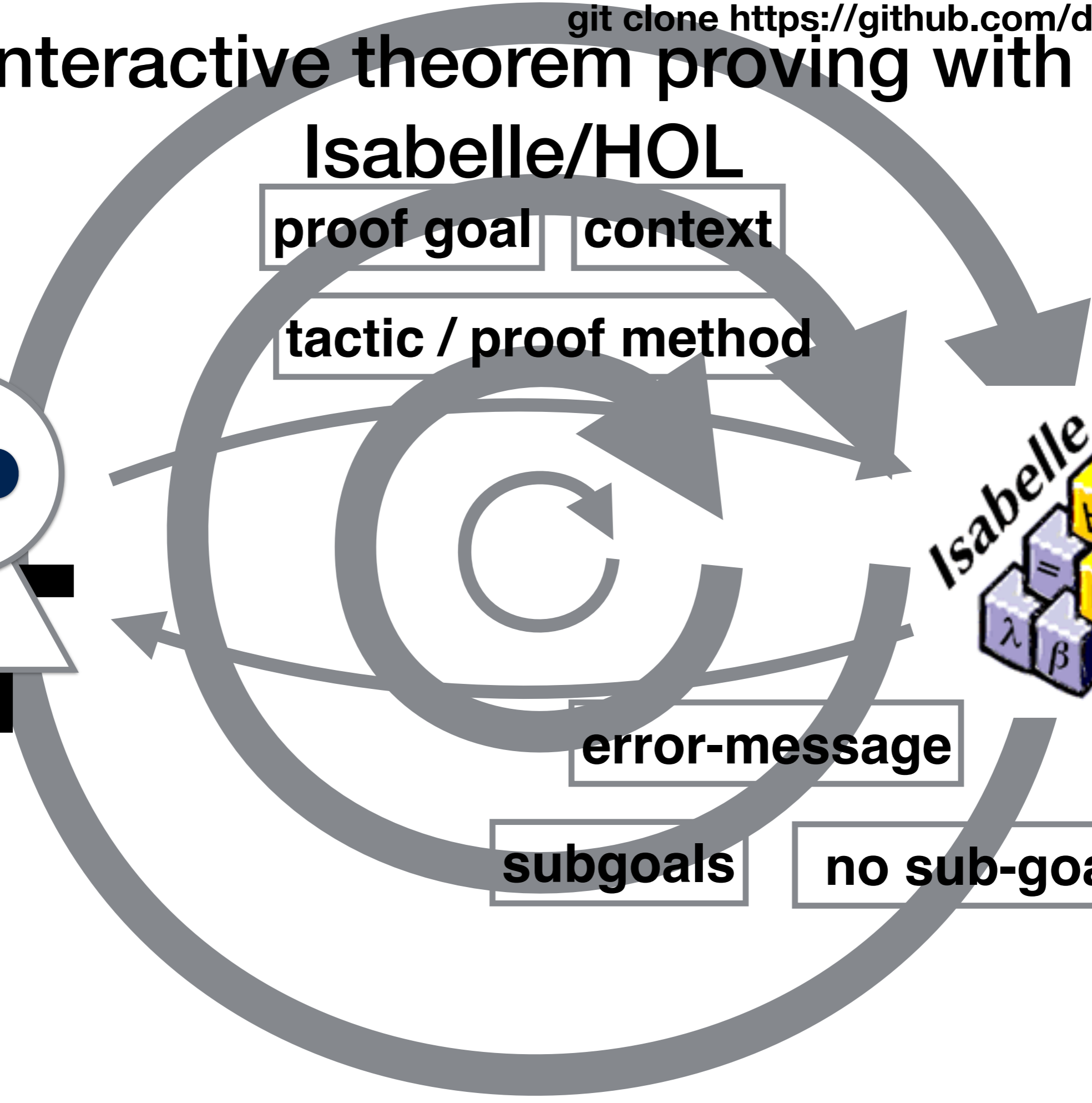
tactic / proof method



error-message

subgoals

no sub-goal!

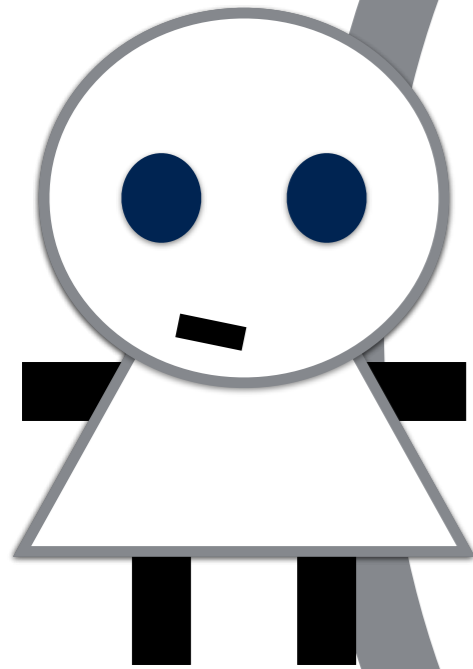


# Interactive theorem proving with

## Isabelle/HOL

proof goal | context

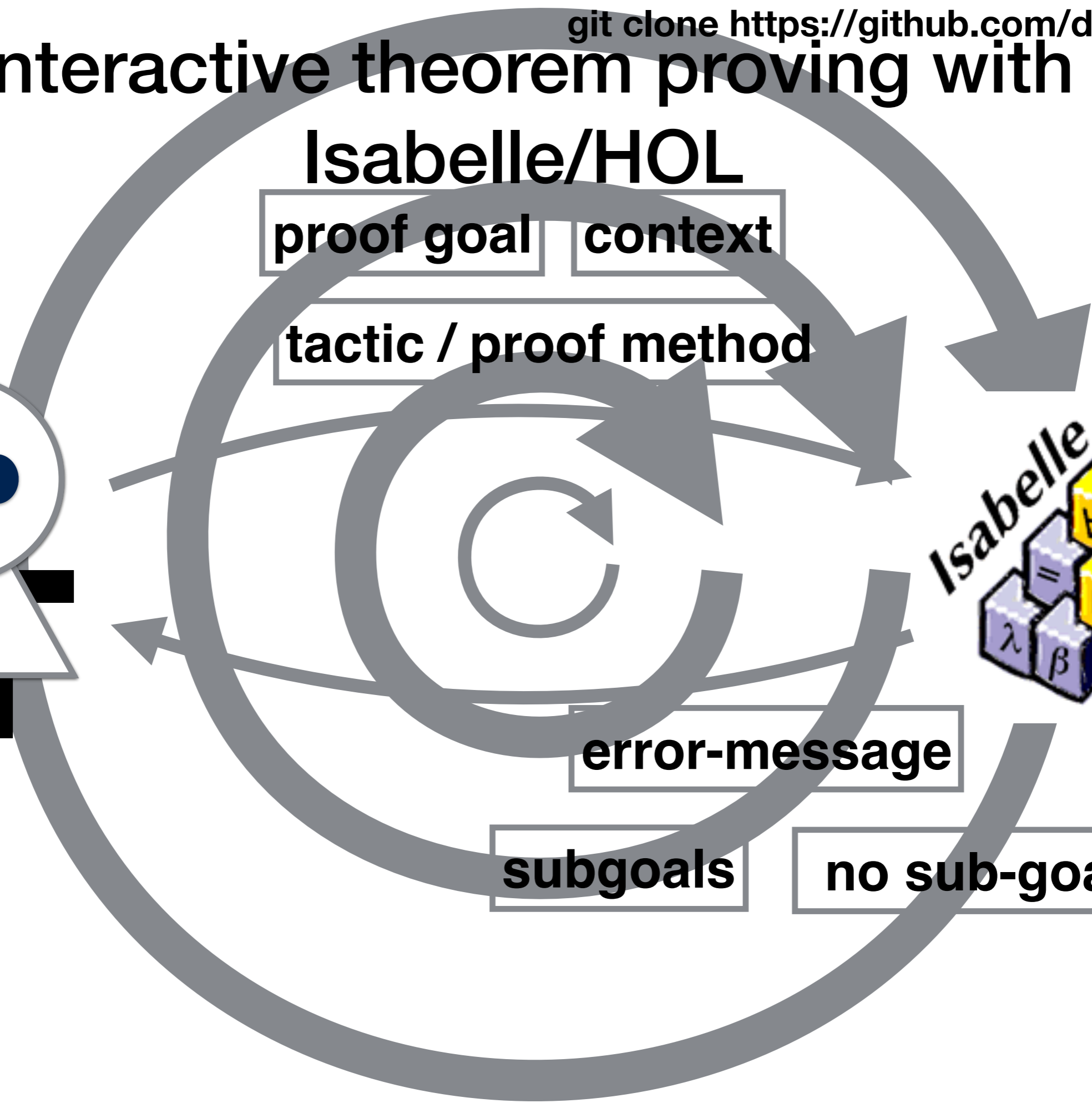
tactic / proof method



error-message

subgoals

no sub-goal!



# Interactive theorem proving with

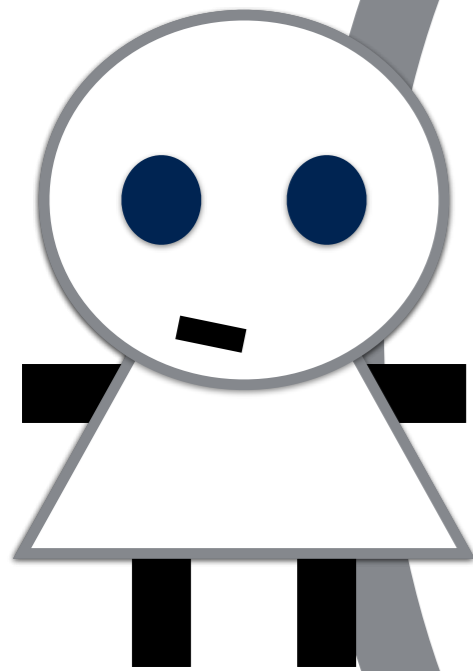
## Isabelle/HOL

proof goal | context

tactic / proof method

error-message

proof-goal!



It's blatantly clear  
You stupid machine, that what  
I tell you is true  
(Michael Norrish)

The screenshot shows a theorem prover interface with a central code editor. The title bar reads "Isabelle2019/HOL - Demo.thy". The code editor contains the following text:

```
18 fun sep:: "'a ⇒ 'a list ⇒ 'a list" where
19   "sep a [] = []" |
20   "sep a [x] = [x]" |
21   "sep a (x#y#zs) = x # a # sep a (y#zs)"
22
23 strategy DInd = Thens [Dynamic (Induct), Auto, IsSolved]
24
25 lemma "map f (sep x xs) = sep (f x) (map f xs)"
26 find_proof DInd
27 try_hard
```

Below the code editor, there are two checked checkboxes: "Proof state" and "Auto update", followed by an "Update" button and a "Search:" label. A dropdown menu is currently set to "100%". At the bottom, there are tabs for "Output", "Query", "Sledgehammer", and "Symbols". On the left side, there are vertical labels "Documentation", "File Browser", and "Theories". On the right side, there are vertical labels "Sidekick", "State", and "Theories".

The screenshot shows the Isabelle2019 HOL - Demo.thy editor interface. The main window displays the following code:

```
18 fun sep:: "'a ⇒ 'a list ⇒ 'a list" where
19   "sep a [] = []" |
20   "sep a [x] = [x]" |
21   "sep a (x#y#zs) = x # a # sep a (y#zs)"
22
23 strategy DInd = Thens [Dynamic (Induct), Auto, IsSolved]
24
25 lemma "map f (sep x xs) = sep (f x) (map f xs)"
26 find_proof DInd
27 try_hard
```

Below the code editor, there are checkboxes for "Proof state" and "Auto update", both of which are checked. An "Update" button is also present. To the right, there is a search field and a zoom level dropdown set to "100%".

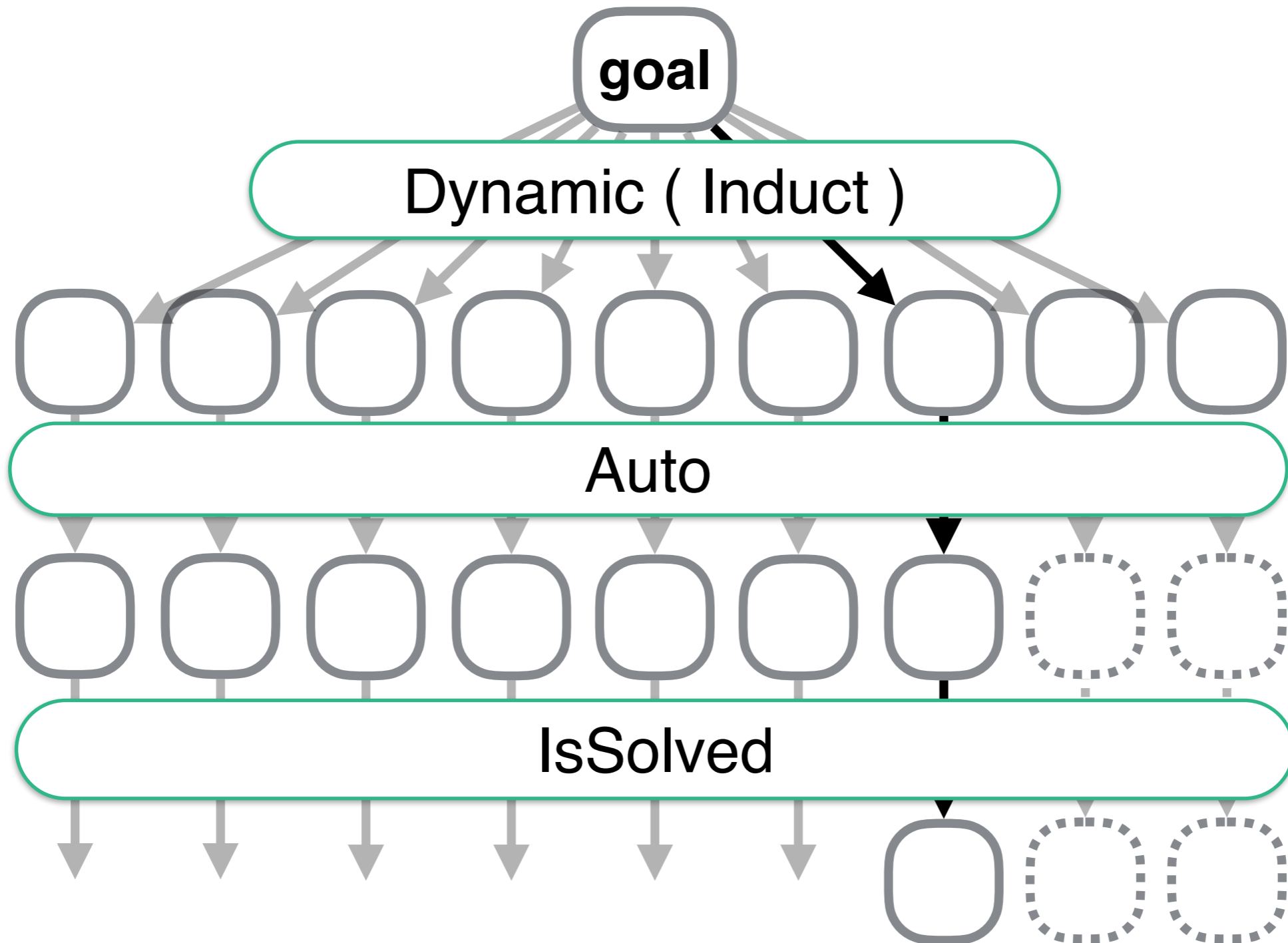
At the bottom of the interface, there is an output window showing the following text:

```
Number of lines of commands: 3
Number of lines of commands: 3
apply (induct xs rule: Demo.sep.induct)
apply auto
done
```

The interface includes a toolbar at the top with various icons for file operations, a sidebar on the left with "Documentation" and "File Browser" tabs, and a sidebar on the right with "Sidekick", "State", and "Theories" tabs. The bottom status bar contains "Output", "Query", "Sledgehammer", and "Symbols" tabs.

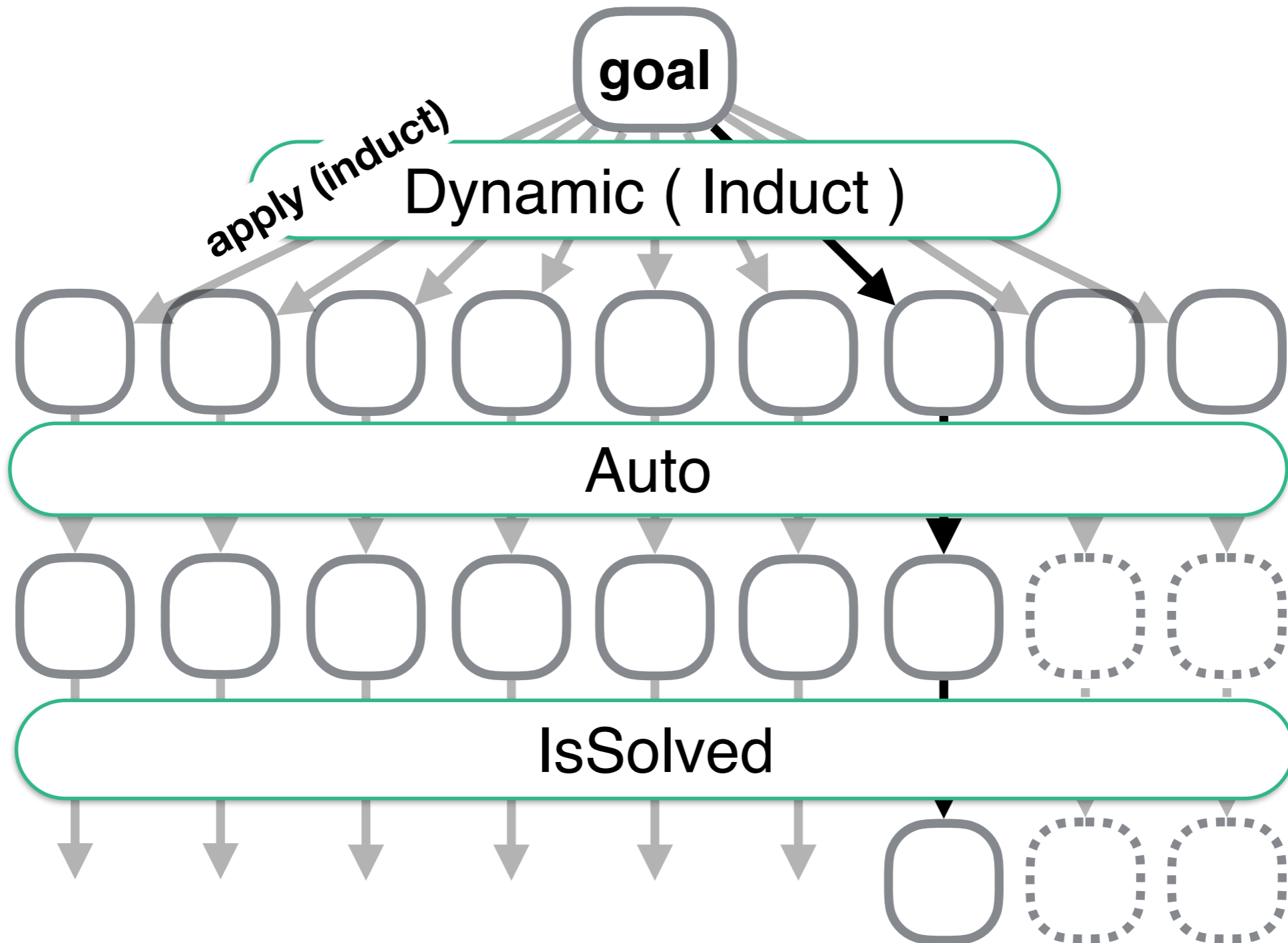
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



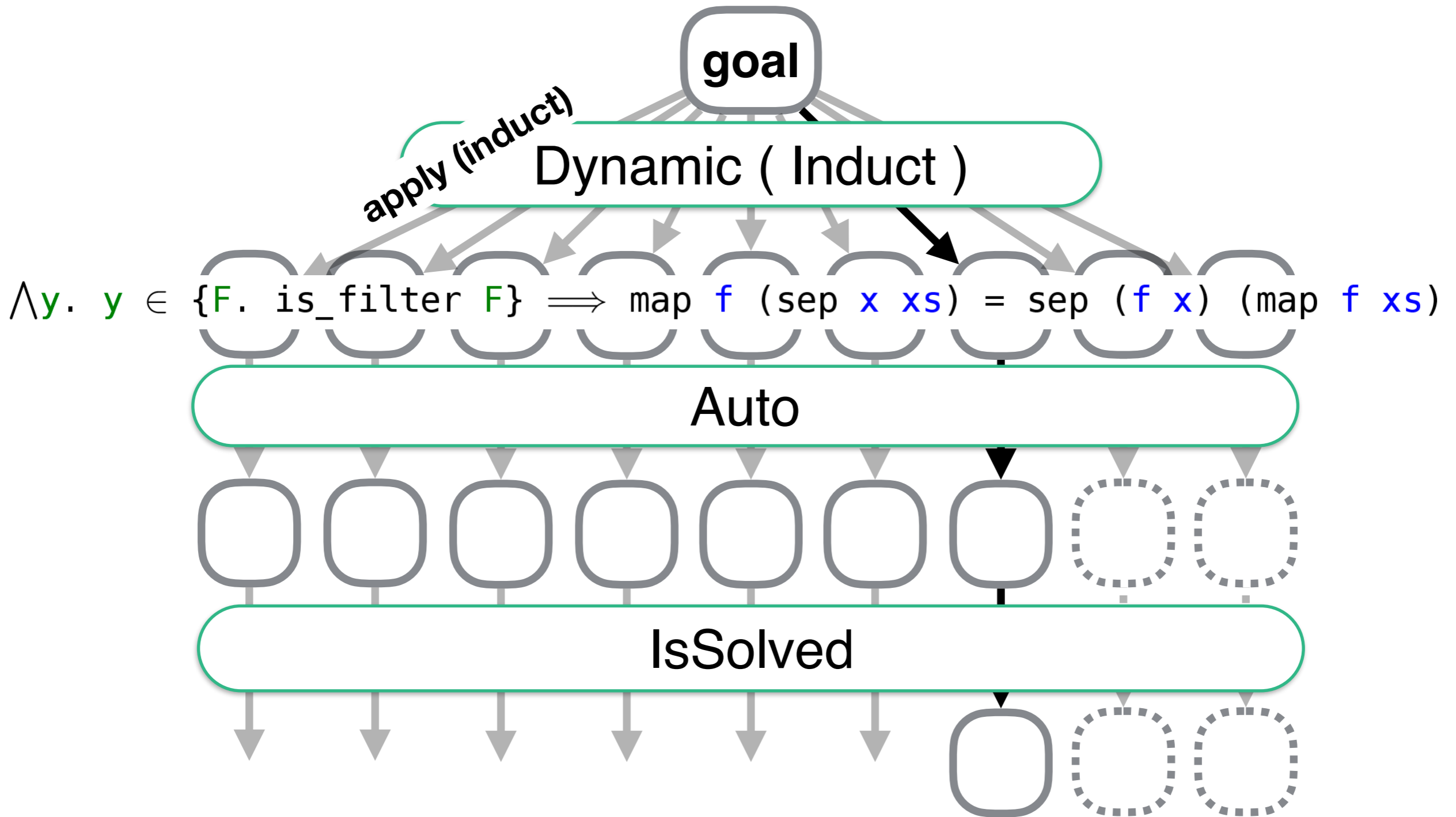
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

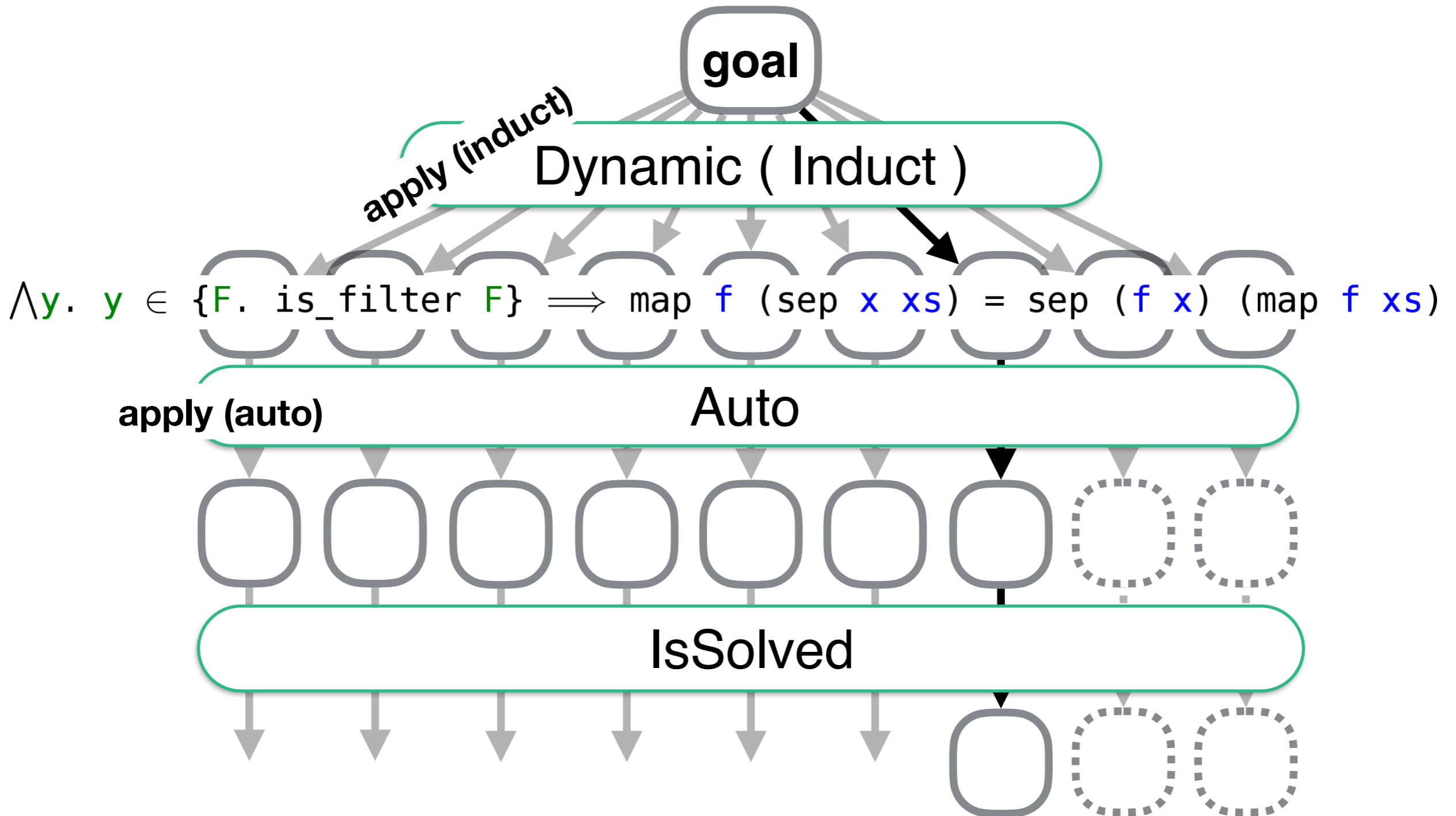
find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)





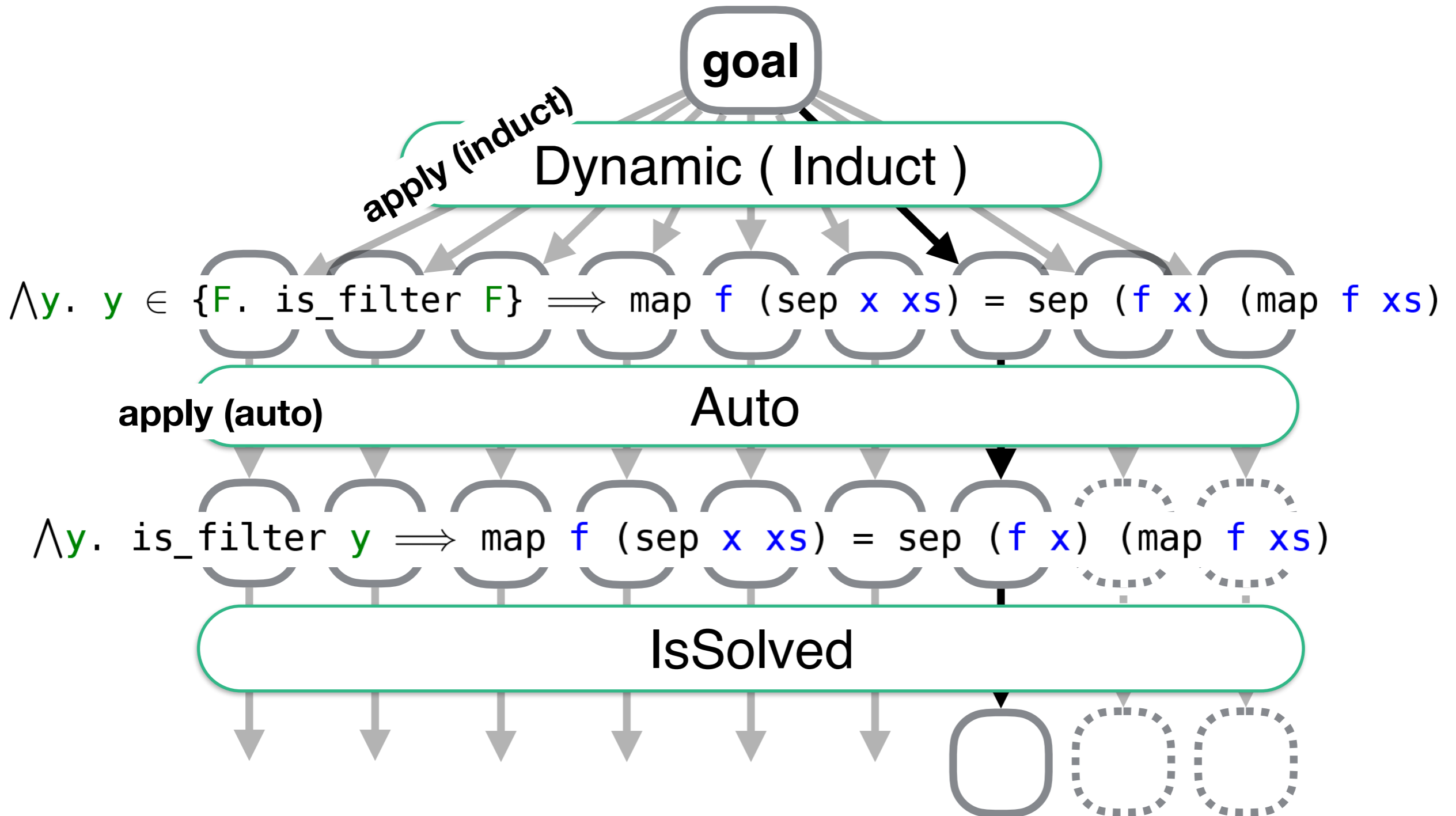
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



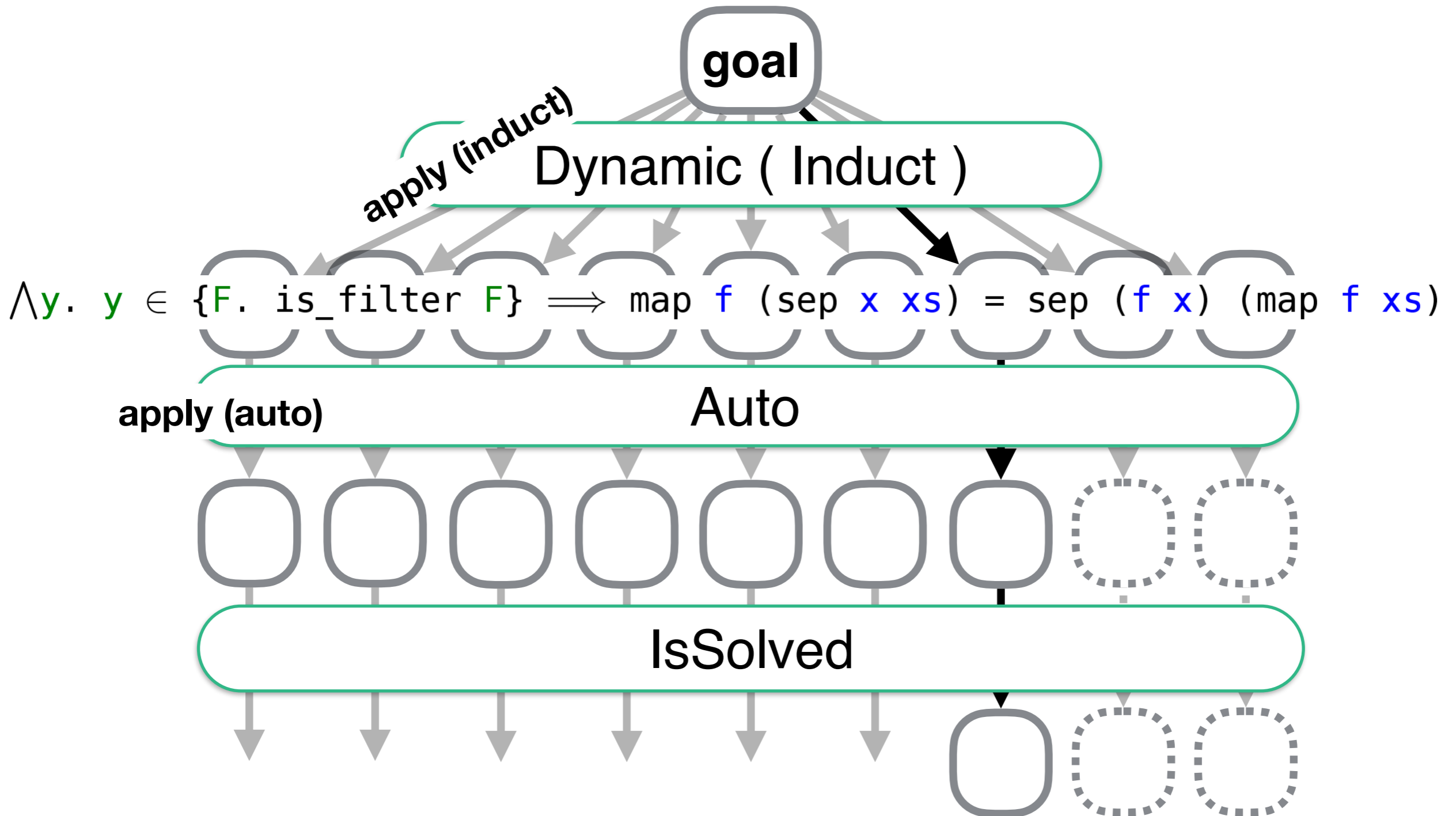
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



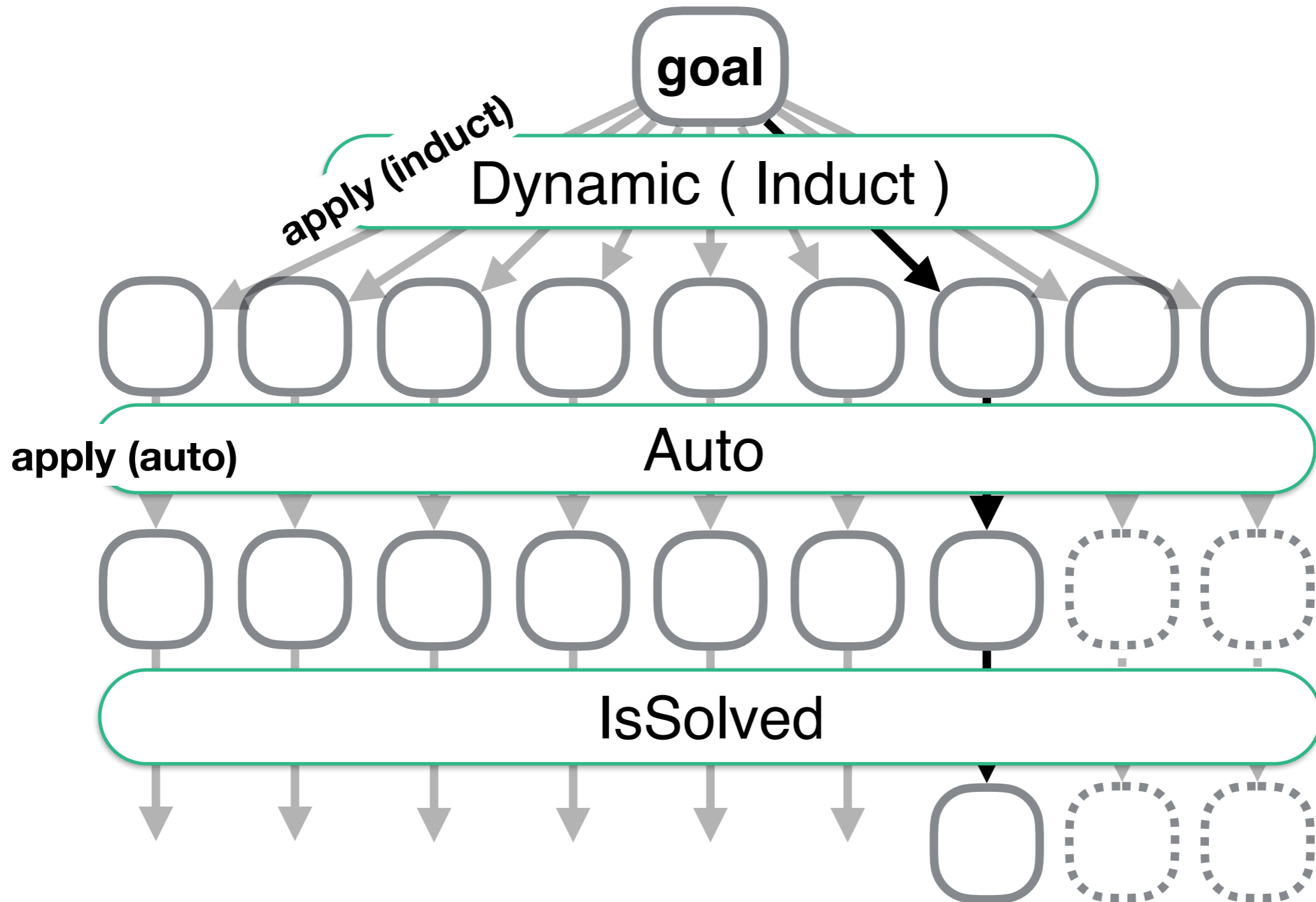
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



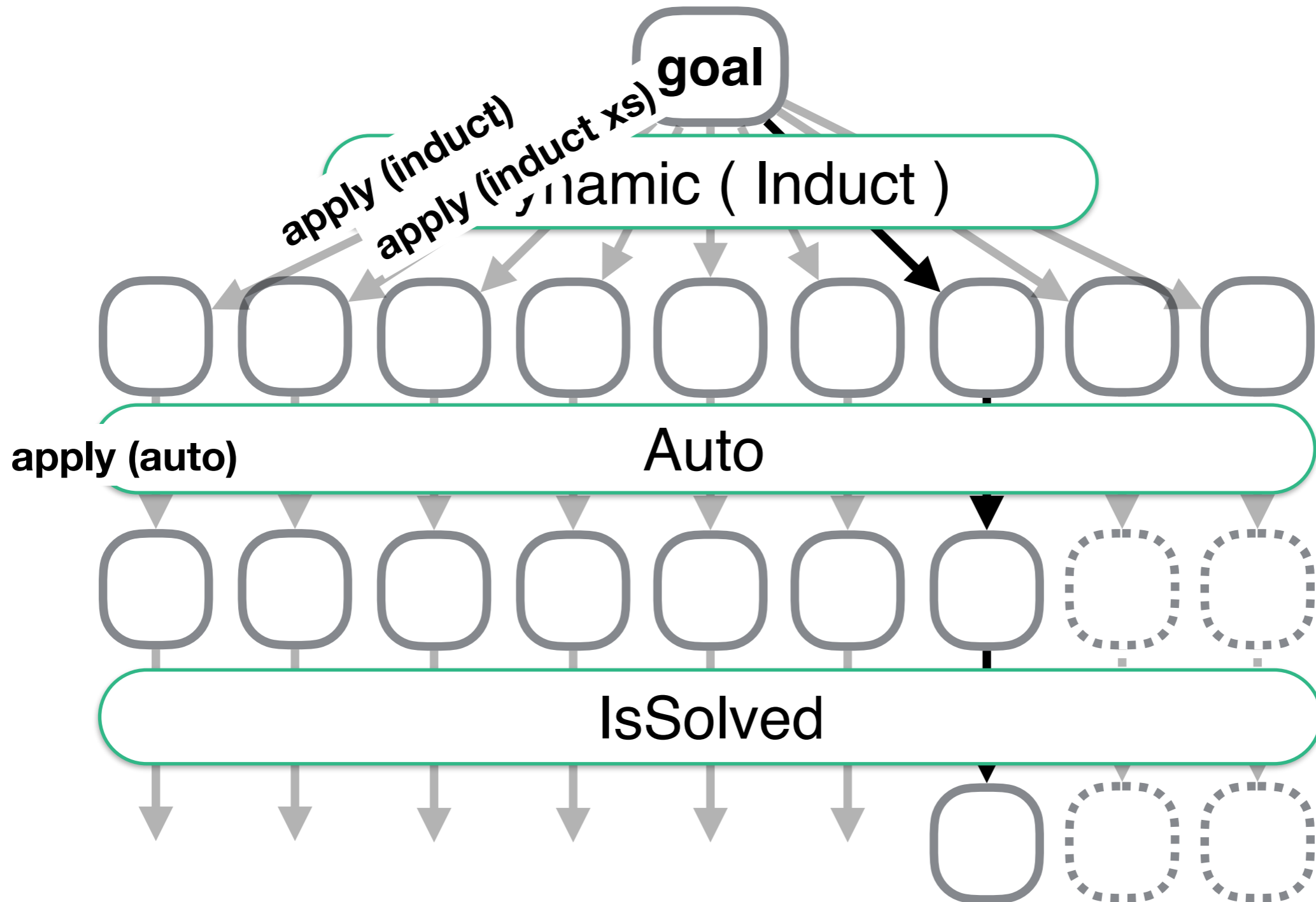
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



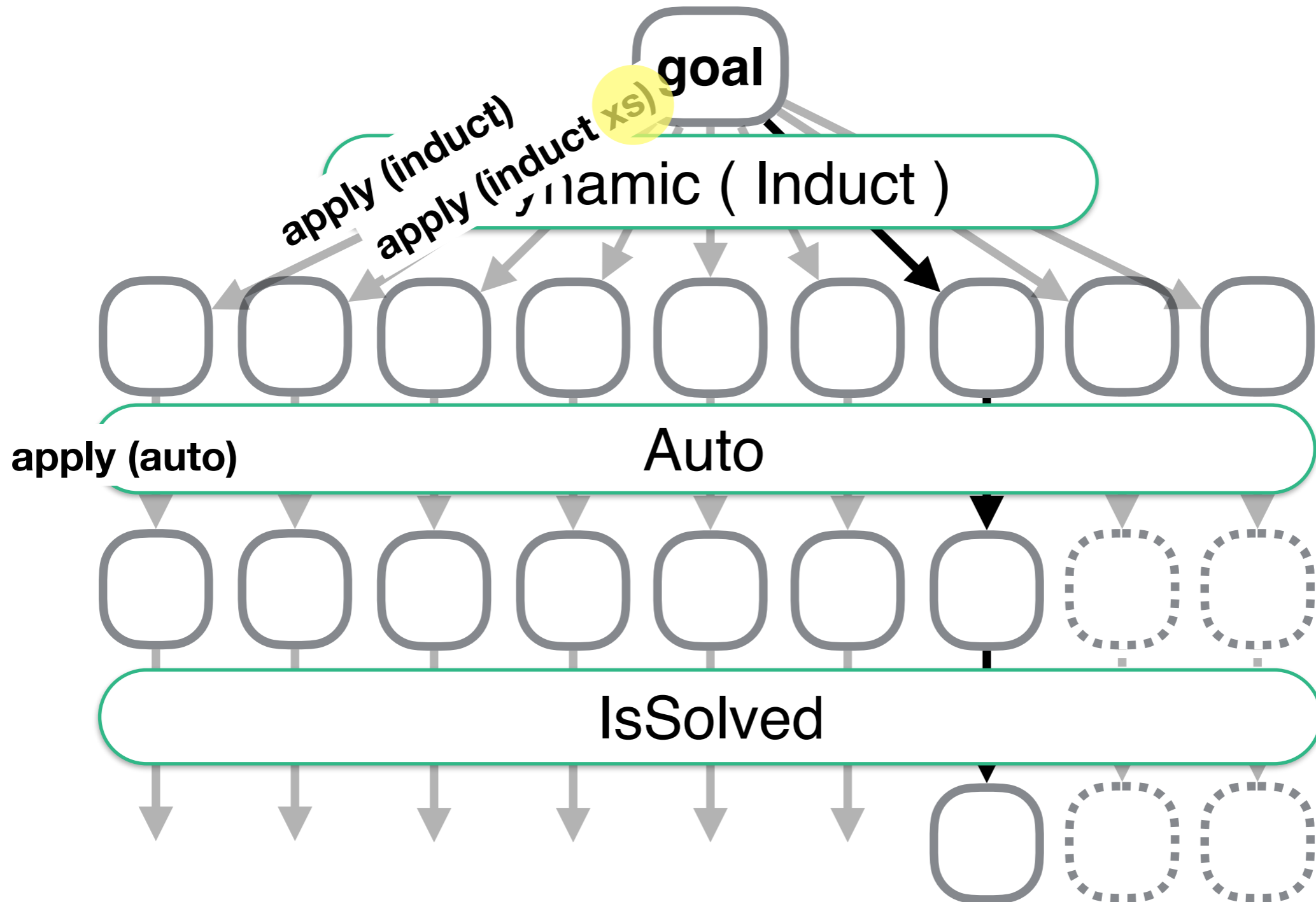
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



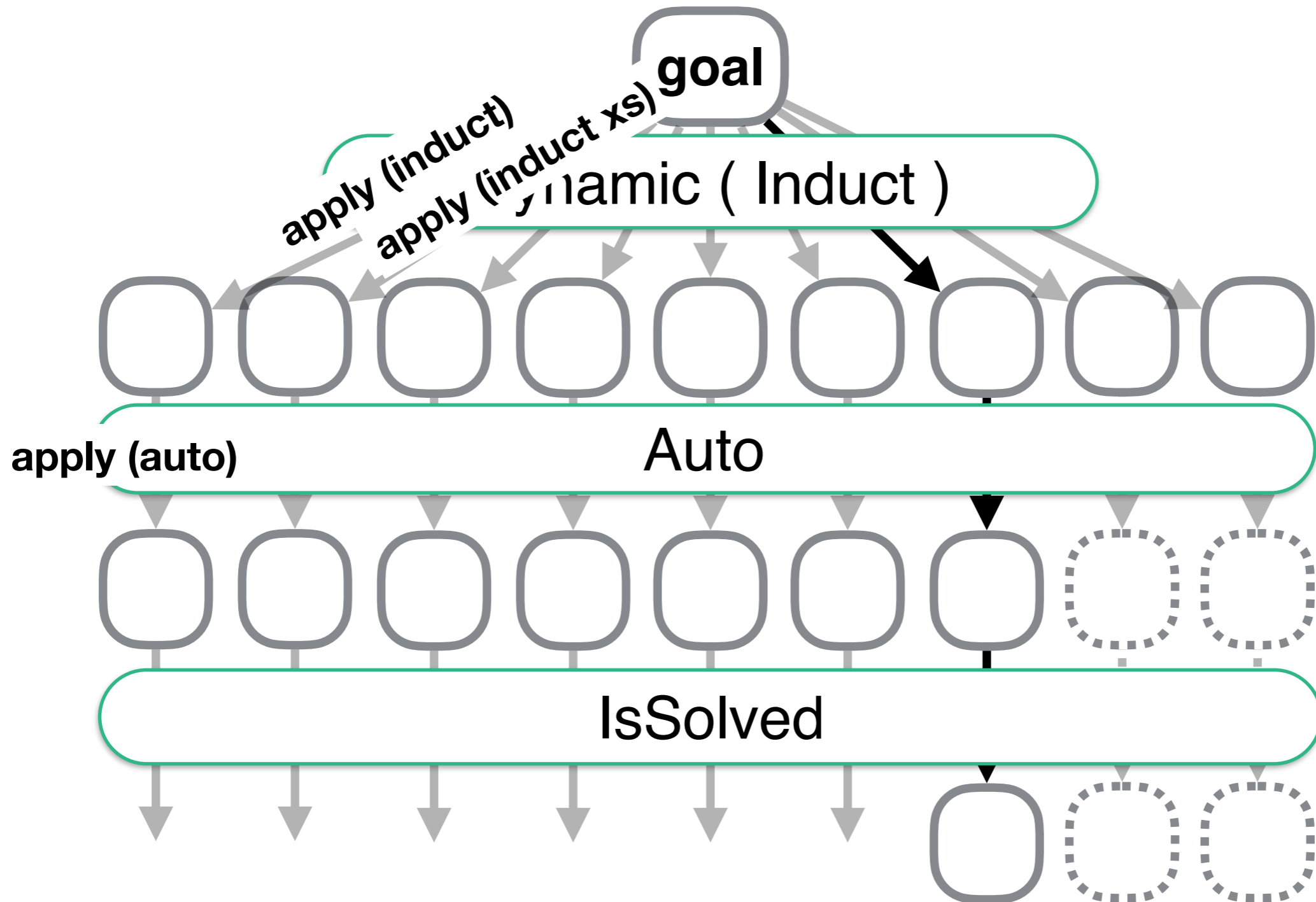
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



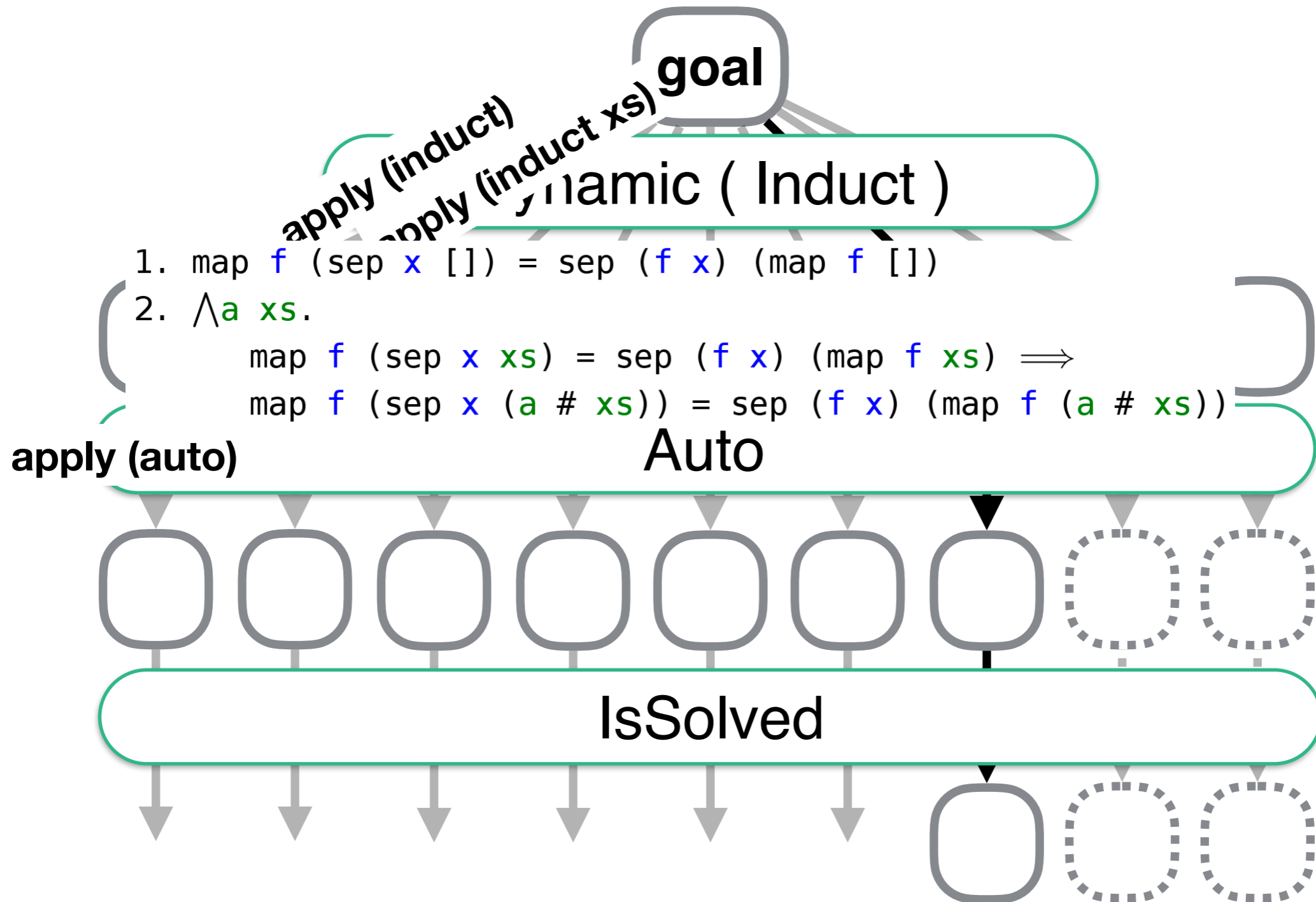
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

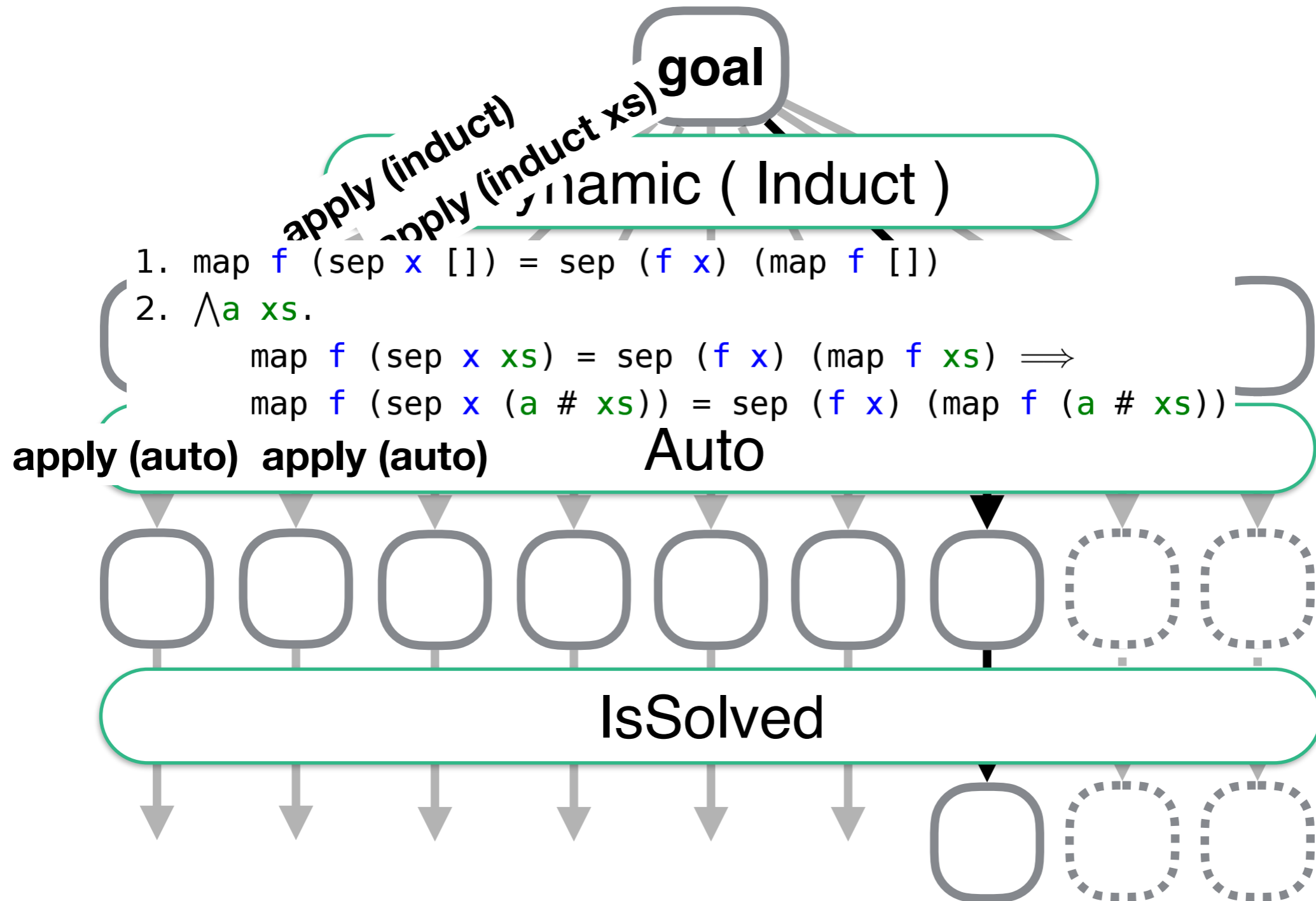
**find\_proof** DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)





**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

**find\_proof** DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)

goal

apply (induct)

apply (induct xs)

Dynamic (Induct)

1. map f (sep x []) = sep (f x) (map f [])

2.  $\wedge a$  xs.

map f (sep x xs) = sep (f x) (map f xs)  $\implies$

map f (sep x (a # xs)) = sep (f x) (map f (a # xs))

apply (auto) apply (auto)

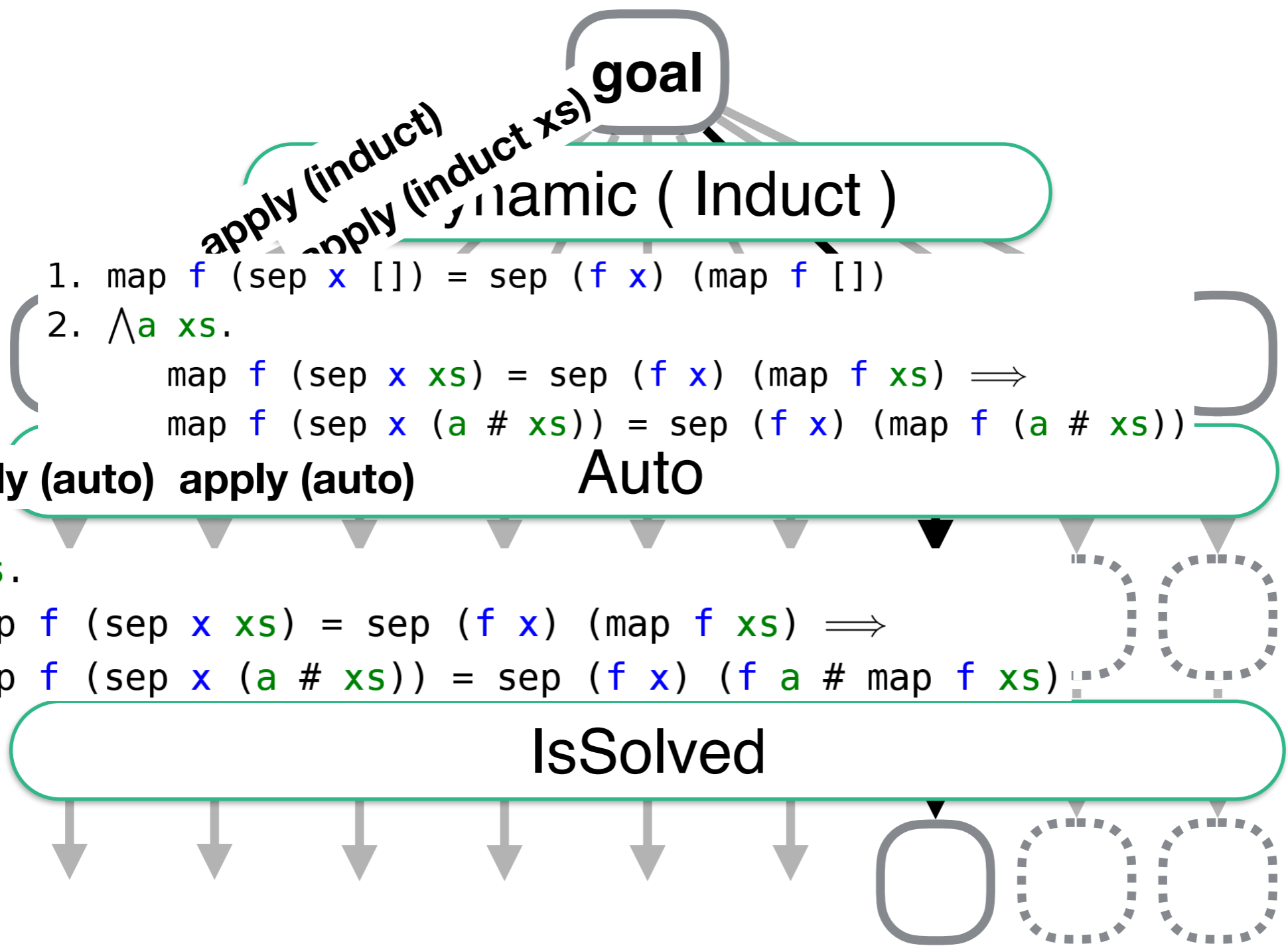
Auto

1.  $\wedge a$  xs.

map f (sep x xs) = sep (f x) (map f xs)  $\implies$

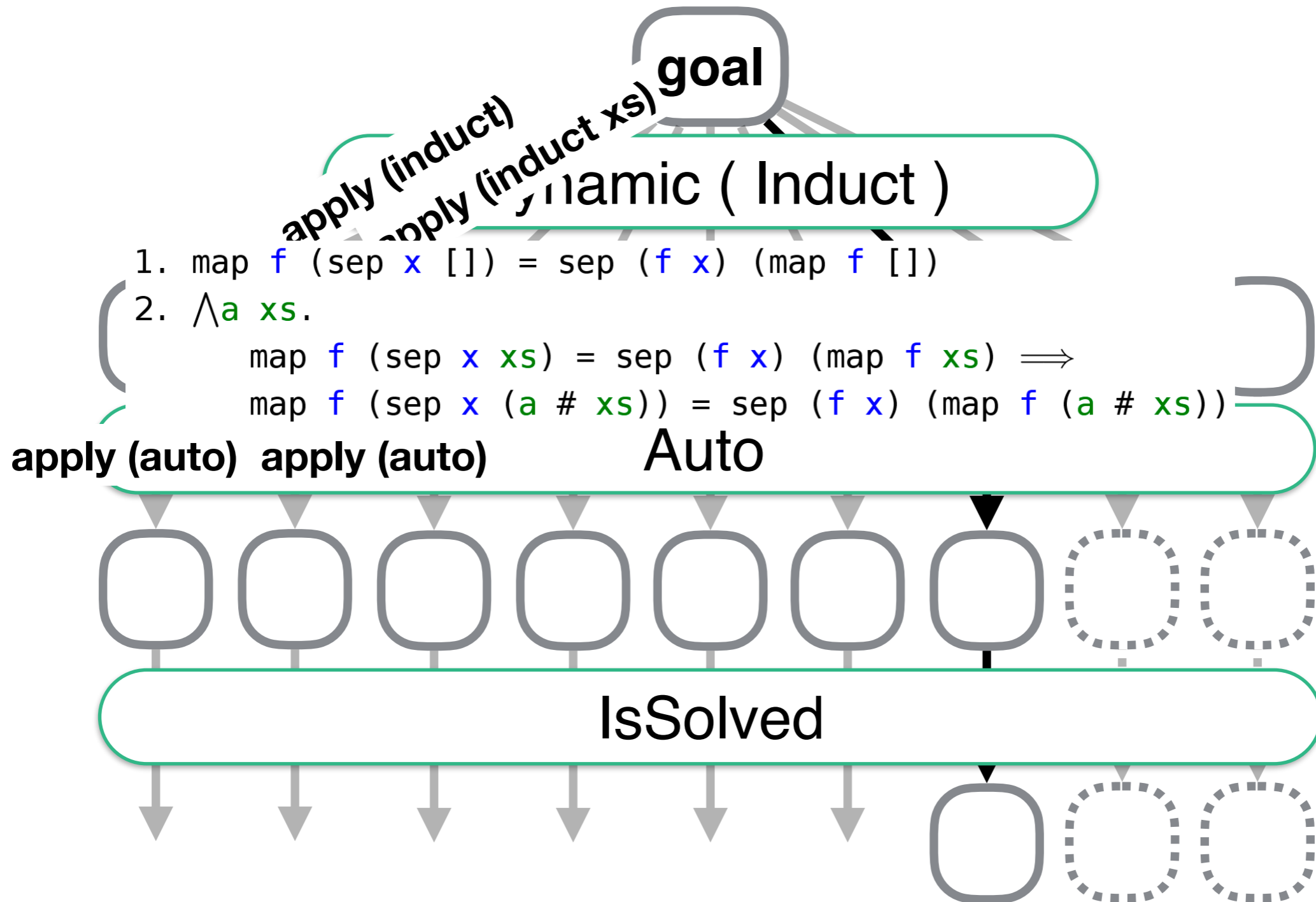
map f (sep x (a # xs)) = sep (f x) (f a # map f xs)

IsSolved



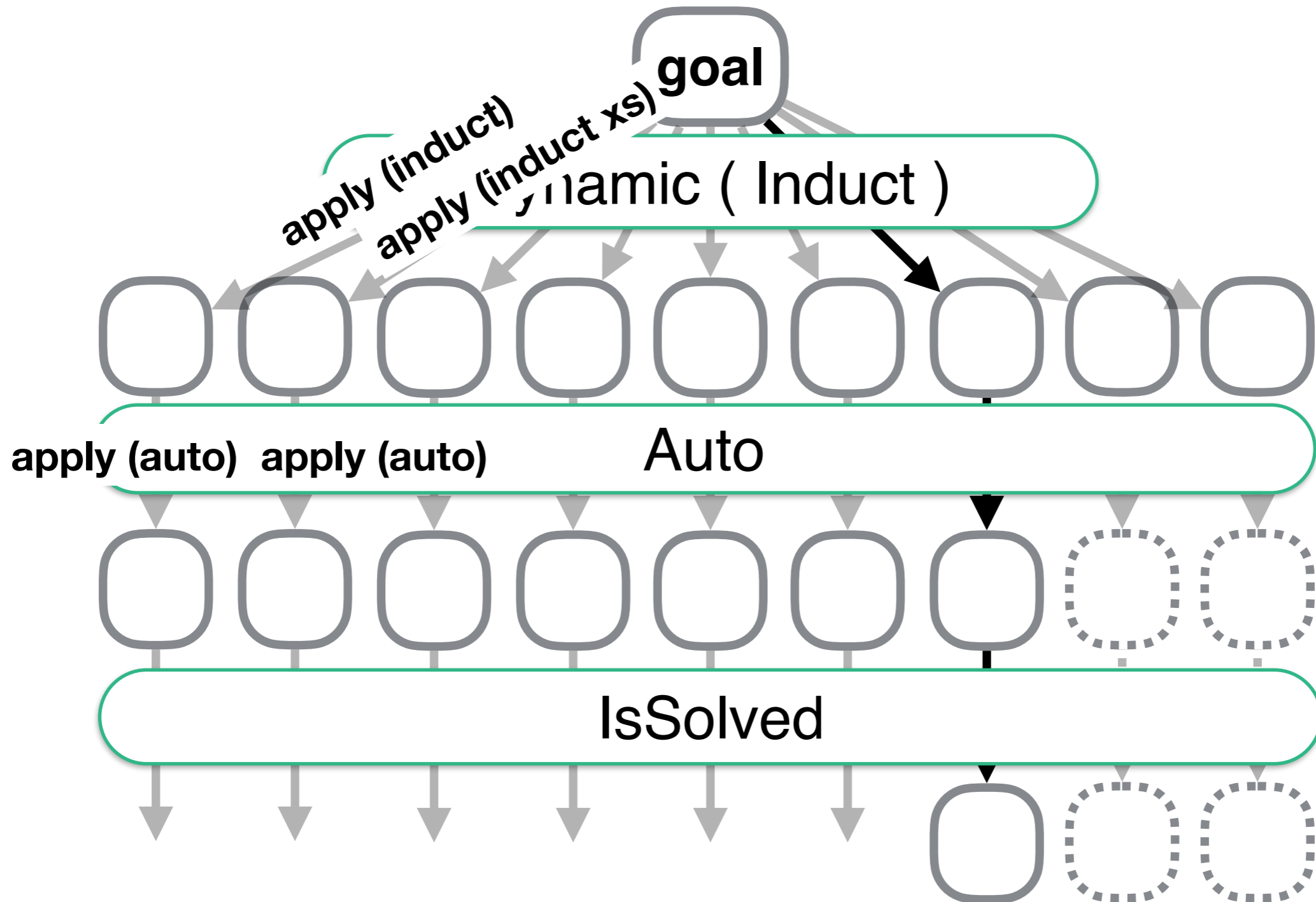
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

**find\_proof** DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



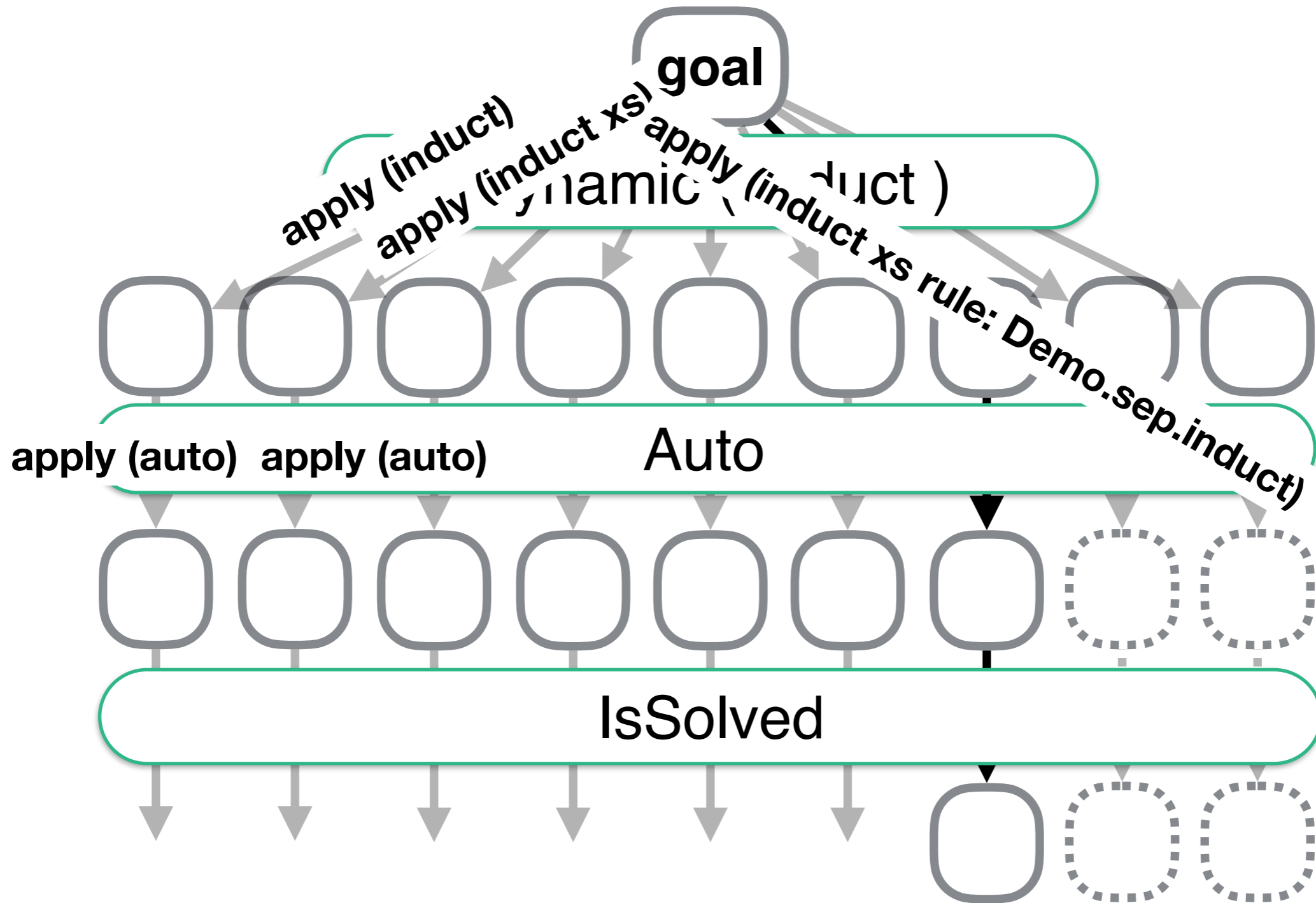
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



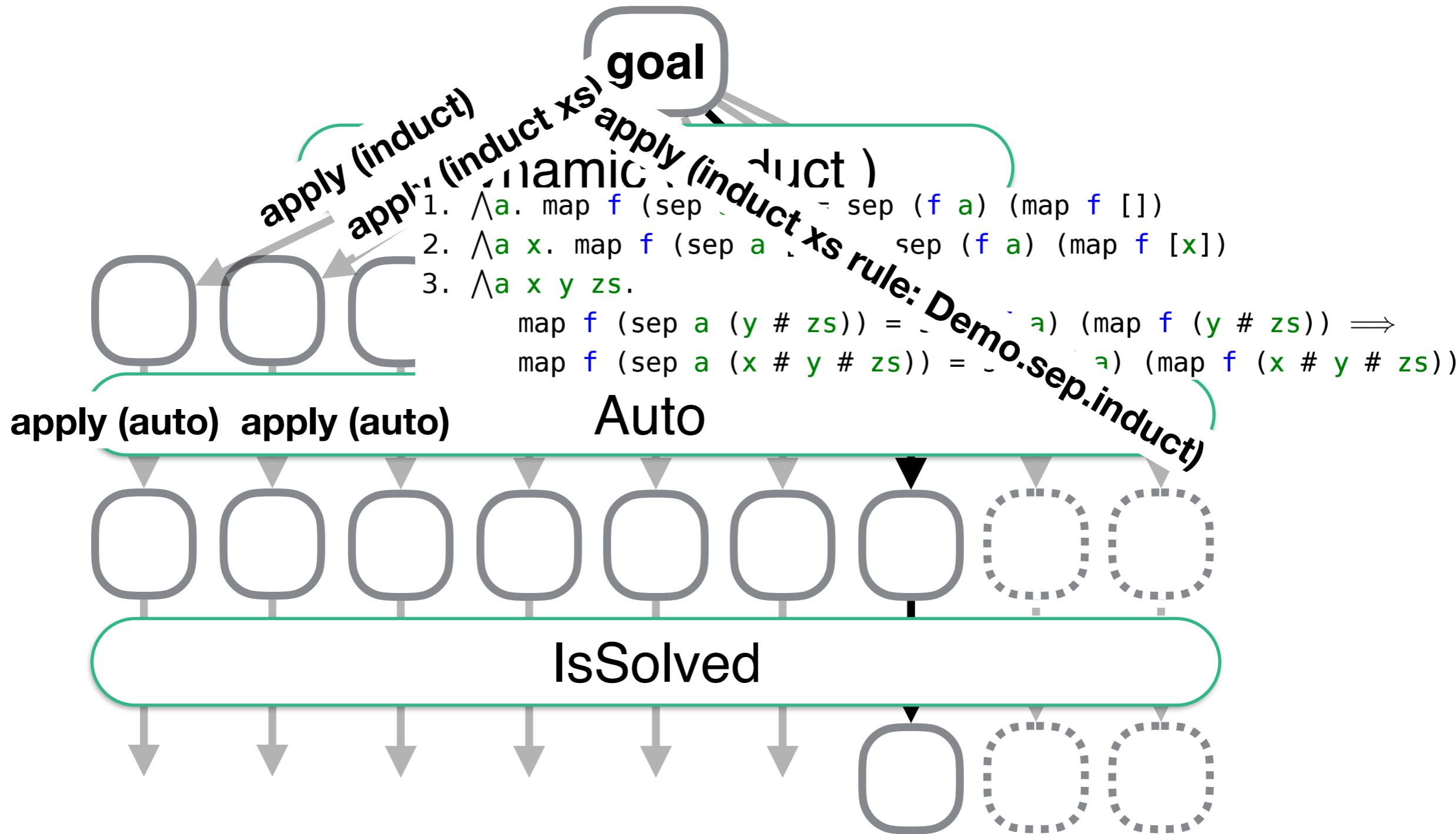
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



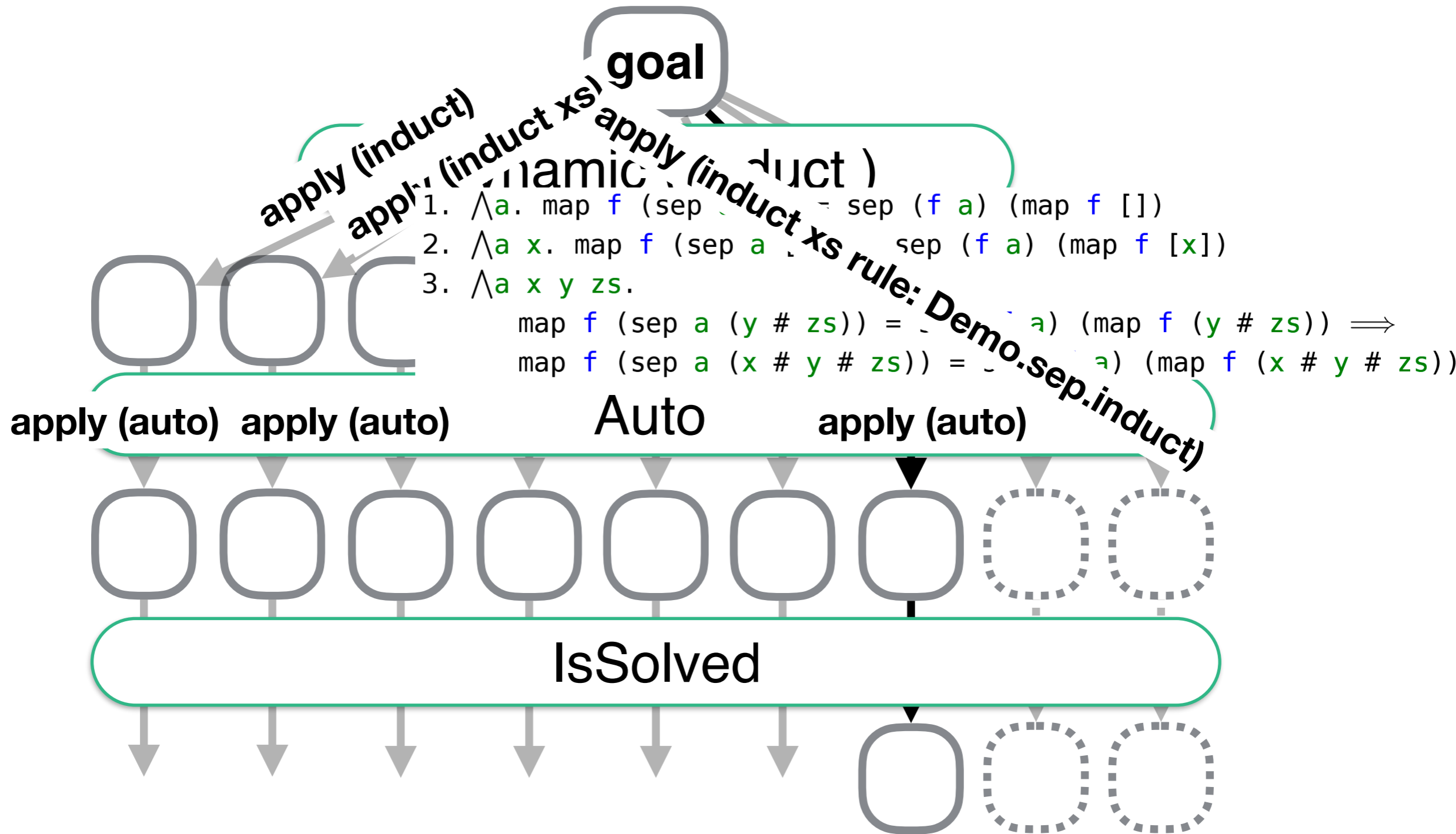
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

**find\_proof** DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



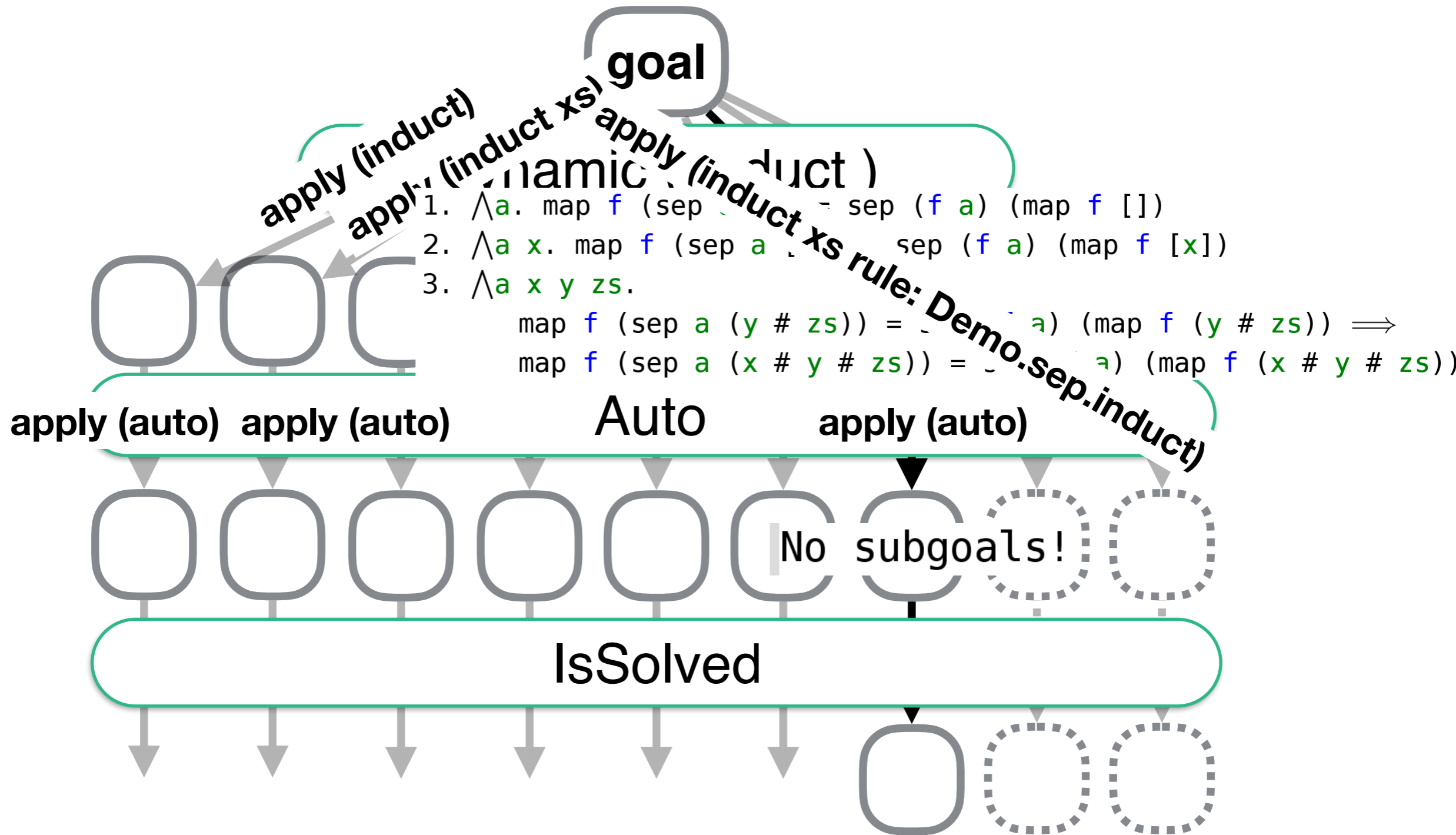
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

**find\_proof** DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

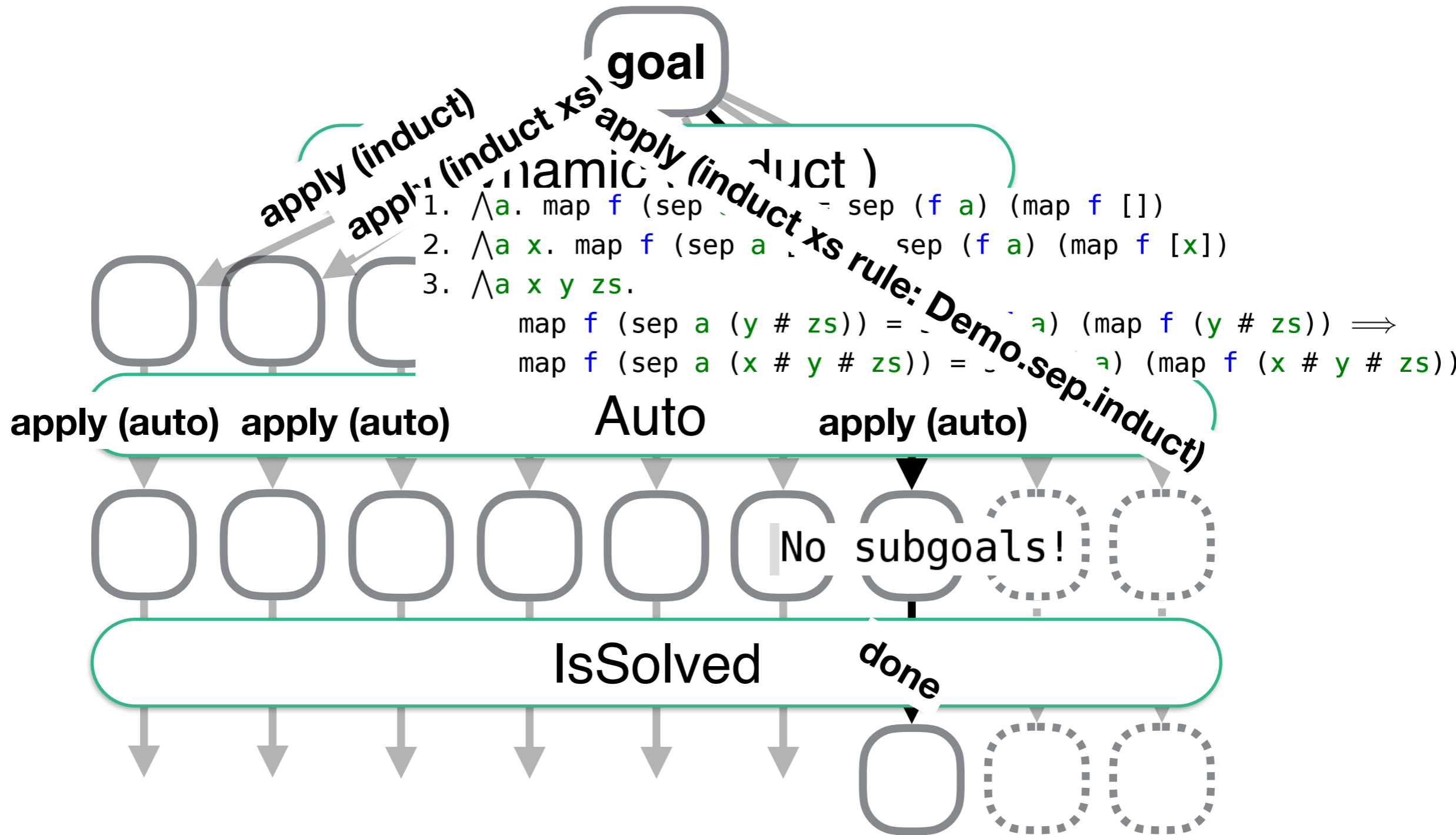
**find\_proof** DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)





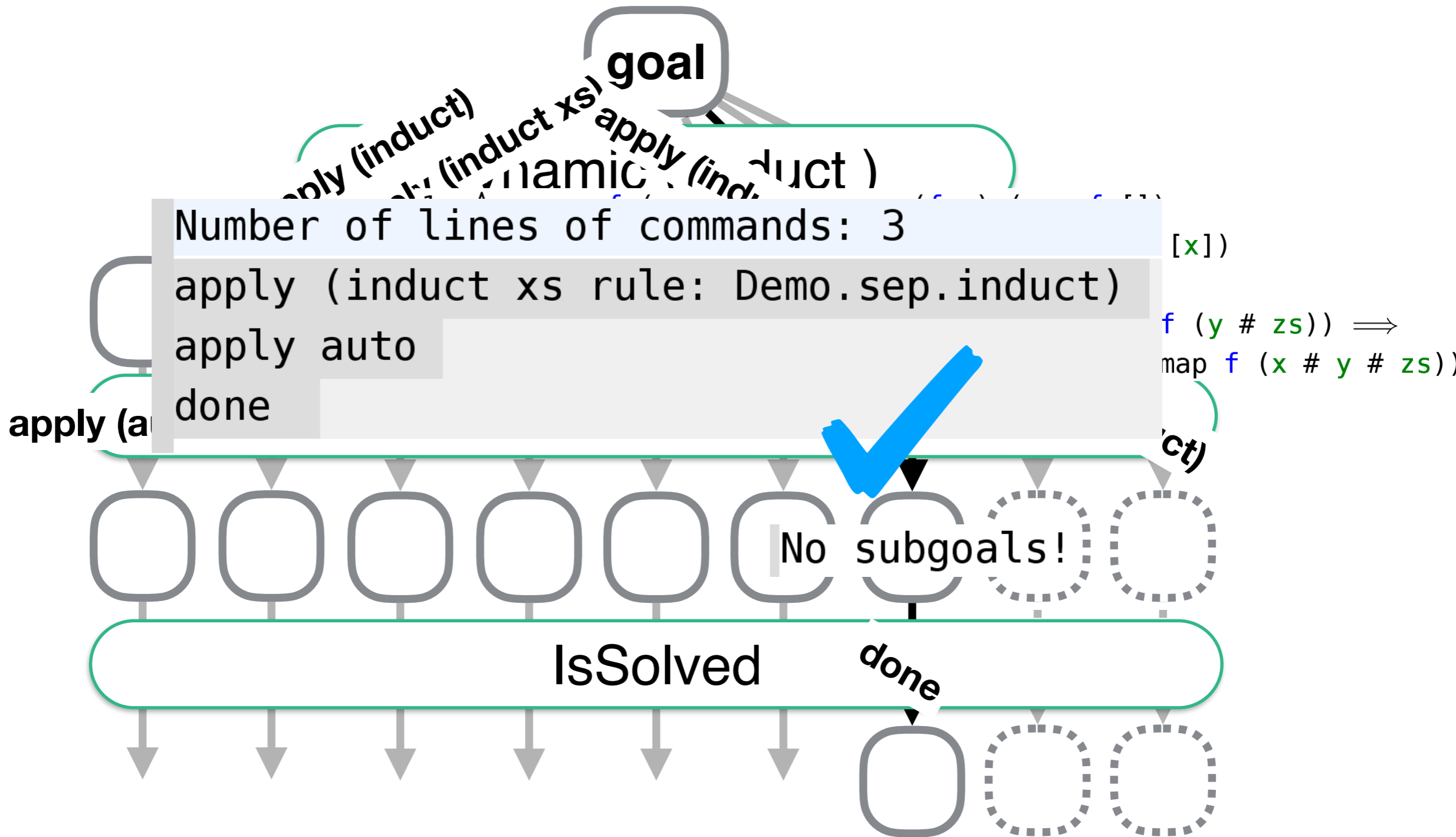
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

**find\_proof** DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



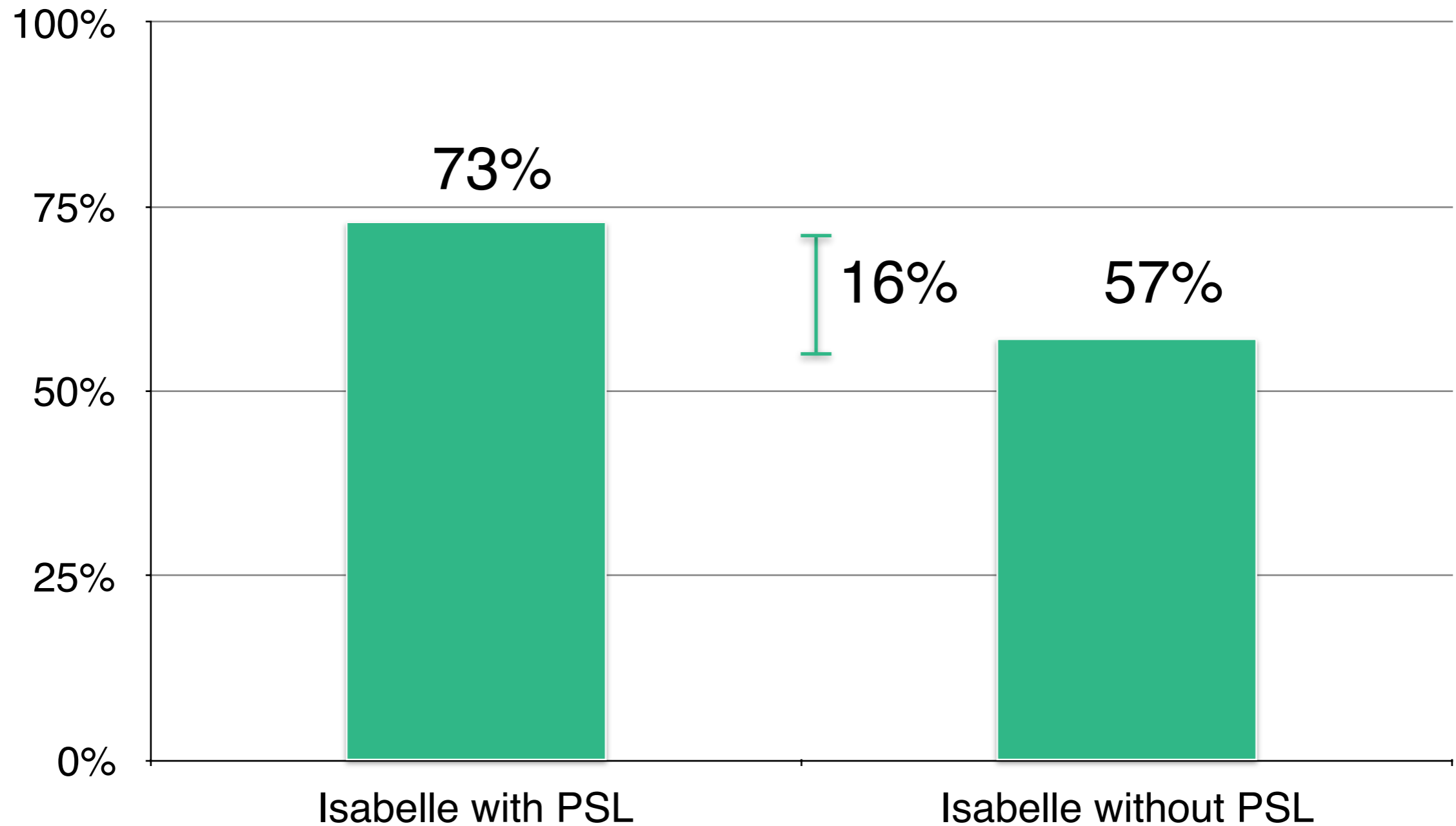
**Lemma** "map f (sep x xs) = sep (f x) (map f xs)"

find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)



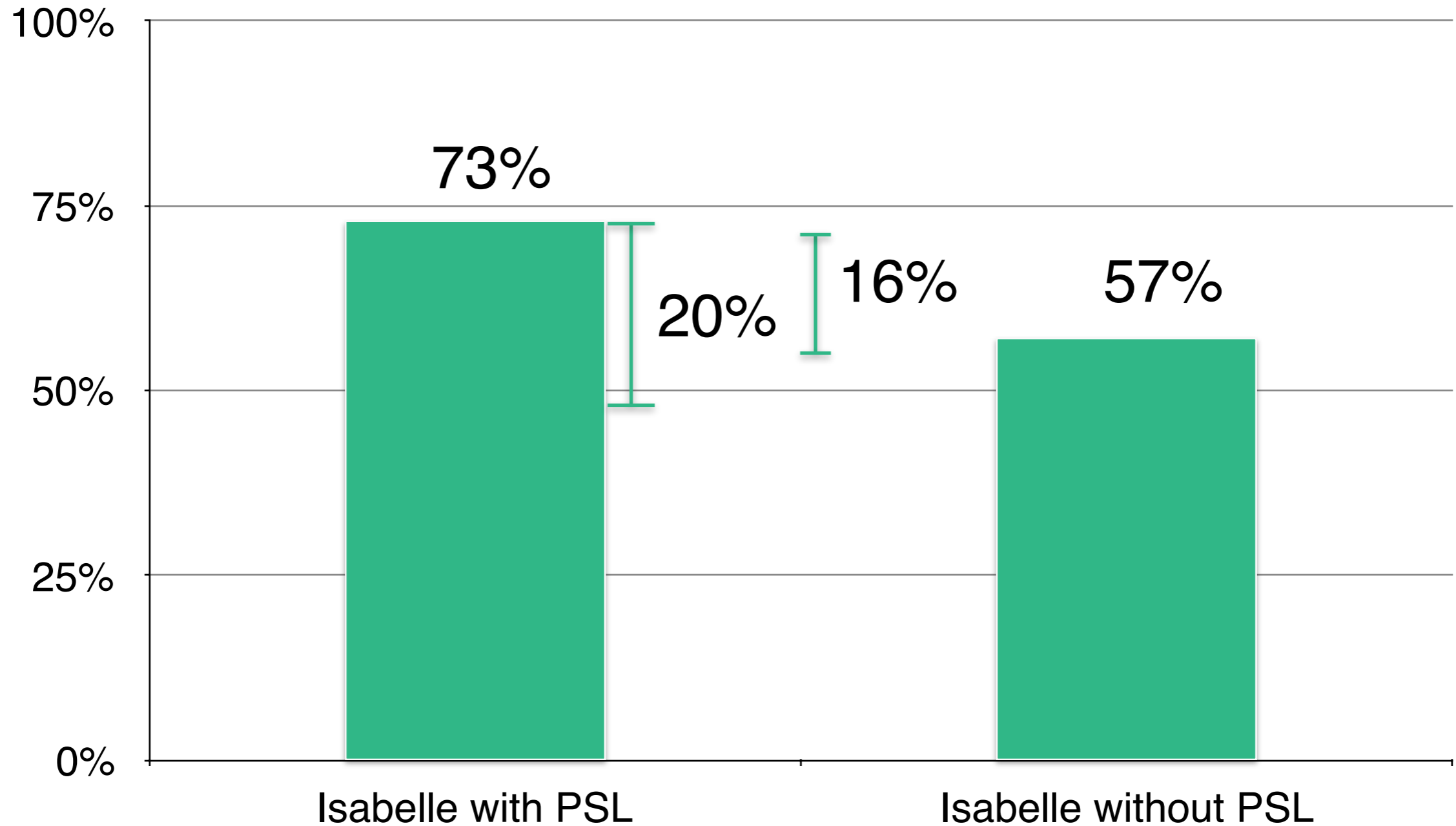
# Evaluation

The percentage of automatically proved obligations out of 1526 proof obligations (timeout = 300s)



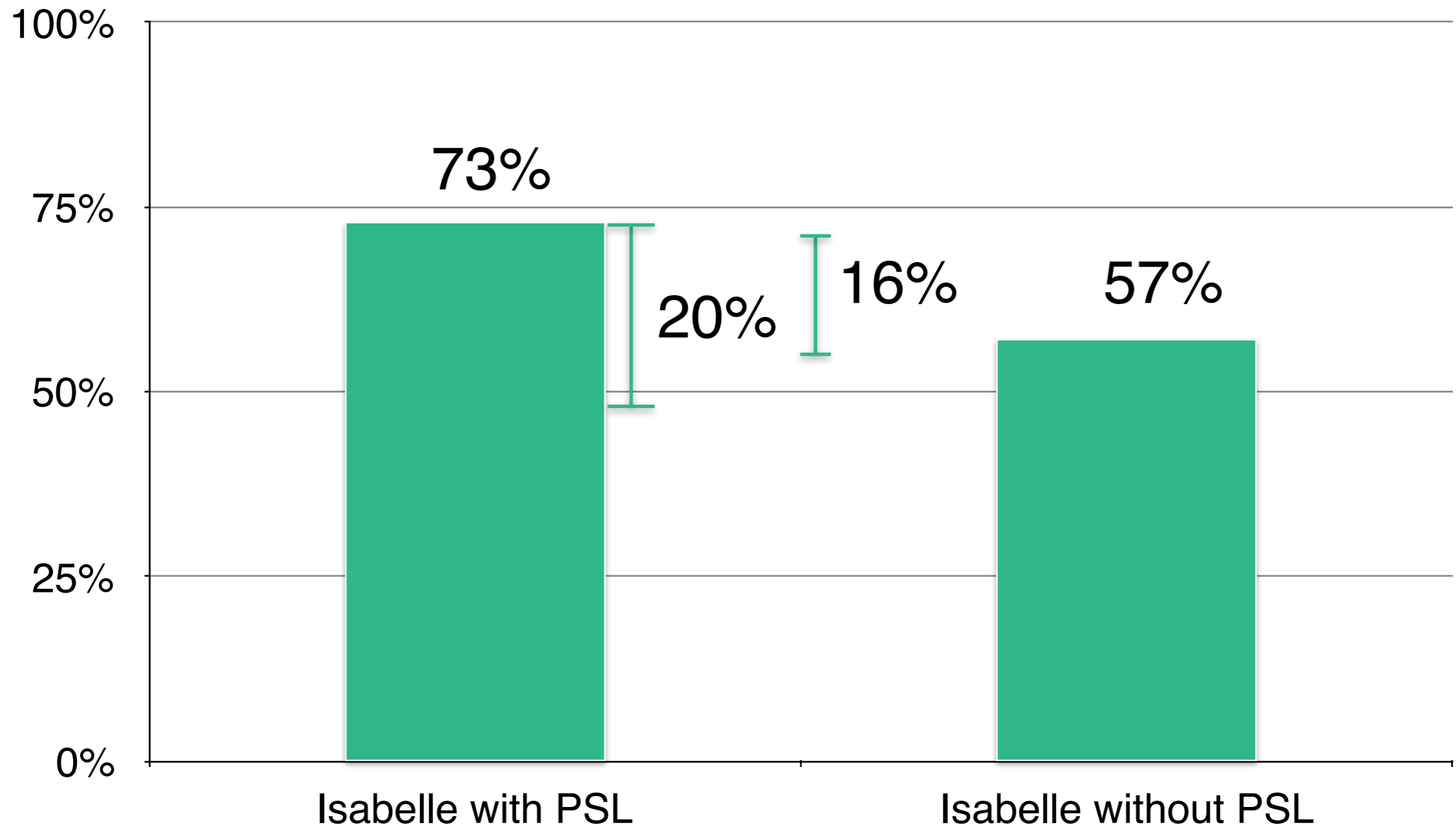
# Evaluation

The percentage of automatically proved obligations out of 1526 proof obligations (timeout = 300s)



# Evaluation

The percentage of automatically proved obligations out of 1526 proof obligations (timeout = 300s)



**5 minutes!**