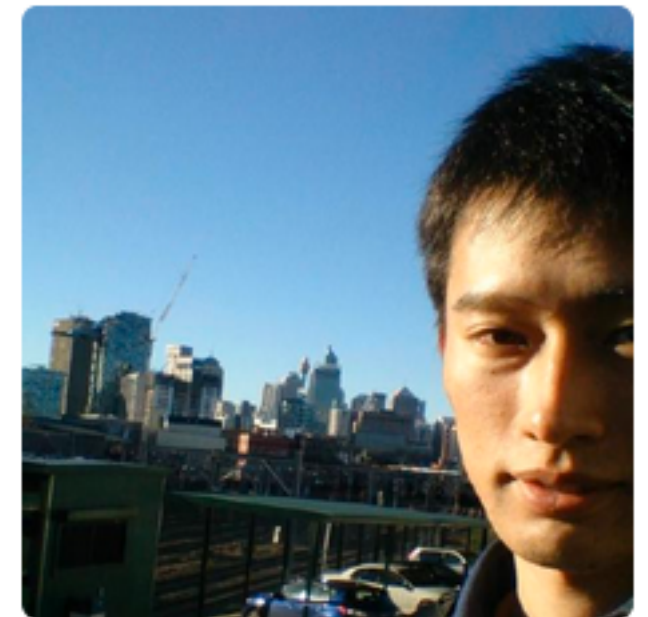


# AI for theorem proving in Isabelle/HOL

*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**

# **M**athematics

**Number Theory**

**Analysis**

**Algebra**

**Geometry**

**Probability Theory**

**etc.**

# **P**hysics

**Acoustics**

**Astrophysics**

**Electromagnetism**

**Molecular Physics**

**Quantum Physics**

**etc.**

# **I**nformatics

**Language**

**Algorithms**

**Data Structures**

**Architecture**

**Software Engineering**

**Formal Method**

**Computational Logic**

# **M**athematics

**Number Theory**

**Analysis**

**Algebra**

**Geometry**

**Probability Theory**

**etc.**

# **P**hysics

**Acoustics**

**Astrophysics**

**Electromagnetism**

**Molecular Physics**

**Quantum Physics**

**etc.**

# **I**nformatics

**Language**

**Algorithms**

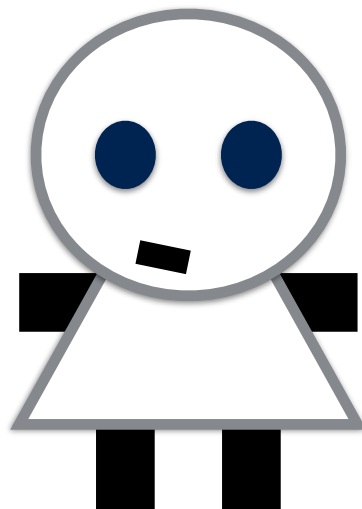
**Data Structures**

**Architecture**

**Software Engineering**

**Formal Method**

**Computational Logic**



# **M**athematics

**Number Theory**

**Analysis**

**Algebra**

**Geometry**

**Probability Theory**

**etc.**

# **P**hysics

**Acoustics**

**Astrophysics**

**Electromagnetism**

**Molecular Physics**

**Quantum Physics**

**etc.**

# **I**nformatics

**Language**

**Algorithms**

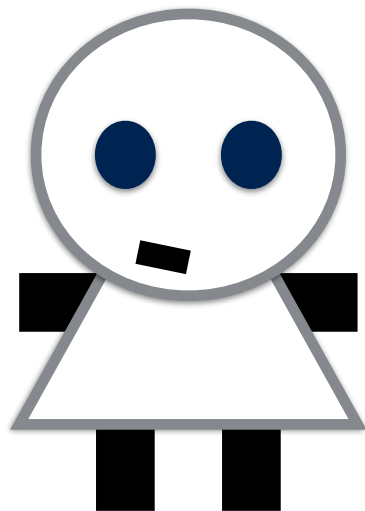
**Data Structures**

**Architecture**

**Software Engineering**

**Formal Method**

**Computational Logic**



**A tiny field inside  
Informatics. Who cares?**

## Physics

**Acoustics**  
**Astrophysics**  
**Electromagnetism**  
**Molecular Physics**  
**Quantum Physics**  
**etc.**

## Informatics

**Language**  
**Algorithms**  
**Data Structures**  
**Architecture**  
**Software Engineering**  
**Formal Method**  
Computational Logic

## Physics

**Acoustics**  
**Astrophysics**  
**Electromagnetism**  
**Molecular Physics**  
**Quantum Physics**  
**etc.**

## Informatics

**Language**  
**Algorithms**  
**Data Structures**  
**Architecture**  
**Software Engineering**  
**Formal Method**  
Computational Logic

**Mathematics** The Language of Science.

**Analysis Algebra Geometry Probability Theory**

## Physics

Acoustics  
Astrophysics  
Electromagnetism  
Molecular Physics  
Quantum Physics  
etc.

## Informatics

Language  
Algorithms  
Data Structures  
Architecture  
Software Engineering  
**Formal Method**  
Computational Logic

**Mathematics** The Language of Science.

Analysis Algebra Geometry Probability Theory

**Logic: the foundation of Mathematics.**

## Physics

## Informatics

**Chemistry**

**Electronics**

**etc.**

**Acoustics**

**Astrophysics**

**Electromagnetism**

**Molecular Physics**

**Quantum Physics**

**etc.**

**Language**

**Algorithms**

**Data Structures**

**Architecture**

**Software Engineering**

**Formal Method**

**Computational Logic**

**Mathematics The Language of Science.**

**Analysis Algebra Geometry Probability Theory**

**Logic: the foundation of Mathematics.**



## Physics

## Informatics

**Chemistry**

**Electronics**

**etc.**

**Acoustics**

**Astrophysics**

**Electromagnetism**

**Molecular Physics**

**Quantum Physics**

**etc.**

**Language**

**Algorithms**

**Data Structures**

**Architecture**

**Software Engineering**

**Formal Method**

**Computational Logic**

**Mathematics The Language of Science.**

**Analysis Algebra Geometry Probability Theory**

**Logic: the foundation of Mathematics.**

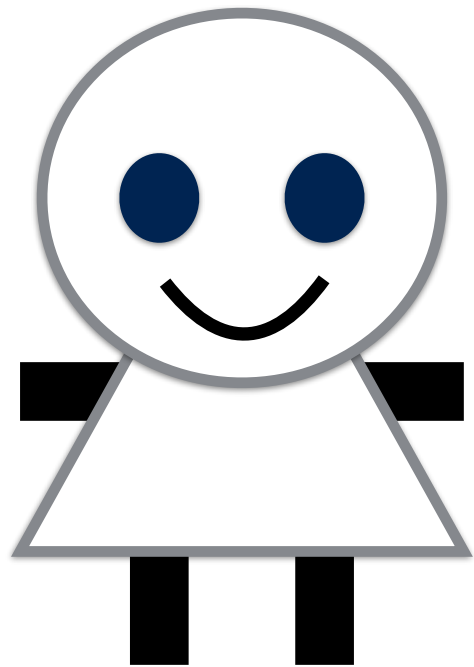
**Automate Logic using AI to Accelerate Science!**



<https://twitter.com/YutakangE>

git clone <https://github.com/data61/PSL>

# Interactive theorem proving with Isabelle/HOL



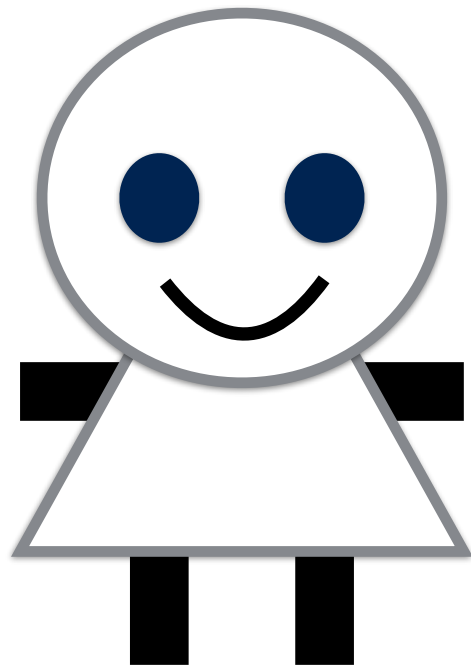
# Interactive theorem proving with

## Isabelle/HOL

proof goal

context

tactic / proof method



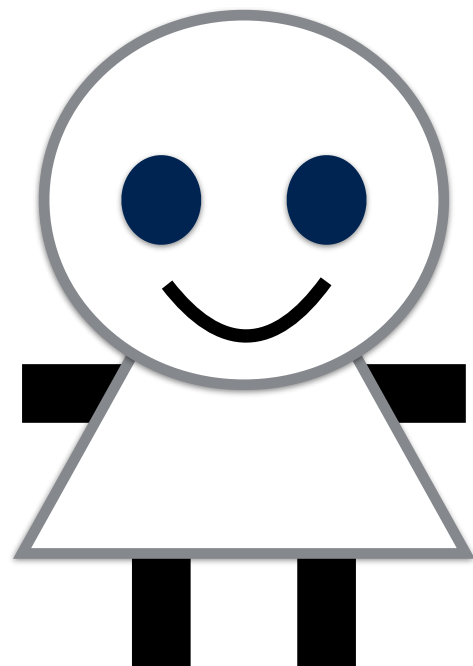
# Interactive theorem proving with

## Isabelle/HOL

proof goal

context

tactic / proof method



error-message

subgoals

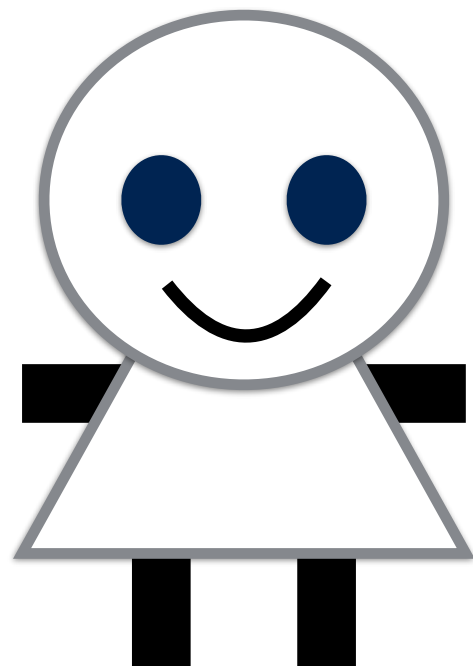
# Interactive theorem proving with

## Isabelle/HOL

proof goal

context

tactic / proof method



error-message

subgoals

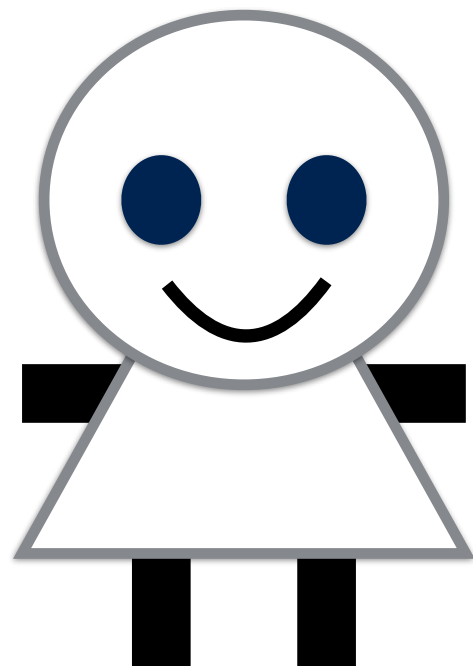
# Interactive theorem proving with

## Isabelle/HOL

proof goal

context

tactic / proof method



error-message

subgoals

no sub-goal!

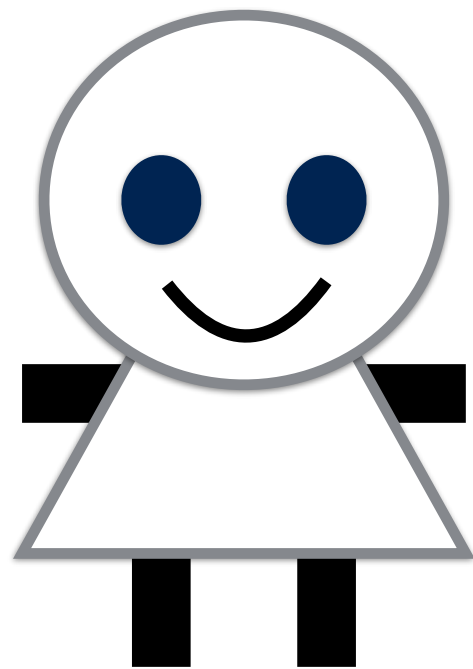
# Interactive theorem proving with

## 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

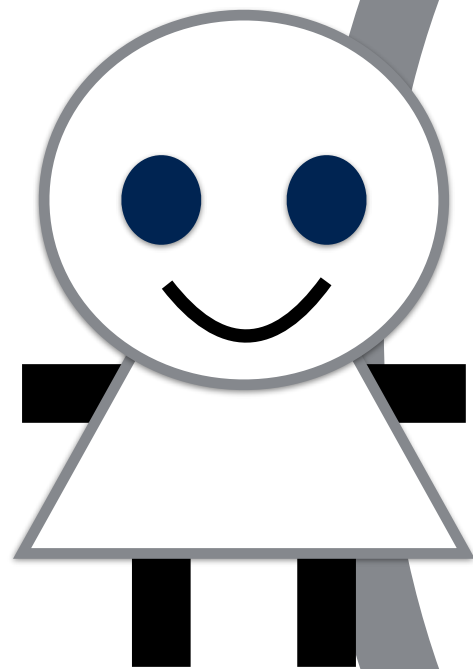
proof goal    context

tactic / proof method

error-message

subgoals

no sub-goal!



# Interactive theorem proving with

Isabelle/HOL

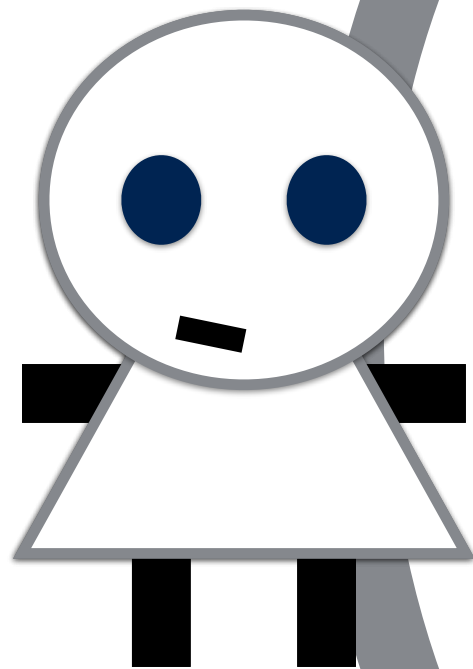
proof goal    context

tactic / proof method

error-message

subgoals

no sub-goal!



# Interactive theorem proving with

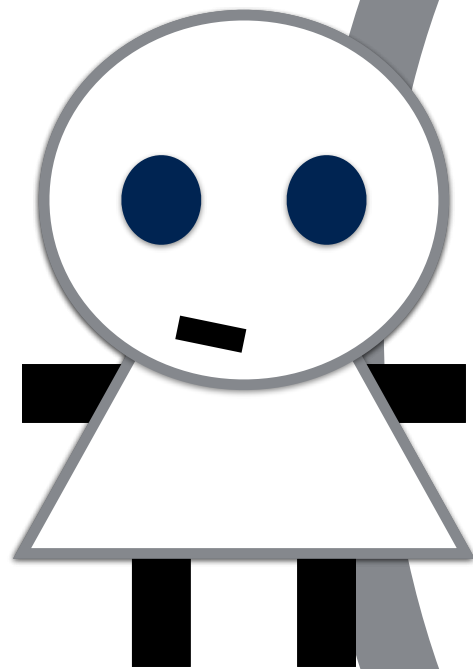
## Isabelle/HOL

proof goal    context

tactic / proof method

error-message

o-goal!



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

The screenshot shows the Isabelle2019 HOL - Demo.thy editor interface. The main window displays a proof script for a list separation function. The script is as follows:

```
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
```

The script defines a function `sep` that separates a list of elements from a list of lists. It uses a `strategy` `DInd` and a `lemma` to prove that `map f (sep x xs) = sep (f x) (map f xs)`. The `find_proof` and `try_hard` commands are used to attempt a proof.

The interface includes a toolbar at the top with various icons for file operations, editing, and viewing. The left sidebar contains a "File Browser" and "Documentation" pane. The right sidebar contains a "Sidekick" pane with "State" and "Theories" tabs. At the bottom, there are checkboxes for "Proof state" and "Auto update", an "Update" button, and a "Search:" field. The bottom status bar shows "Output", "Query", "Sledgehammer", and "Symbols" tabs.

Isabelle2019/HOL - Demo.thy

□ Demo.thy (~/Workplace/MiLkMald2/PSL/Innsbruck/)

```
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
```

File Browser Documentation Sidekick State Theories

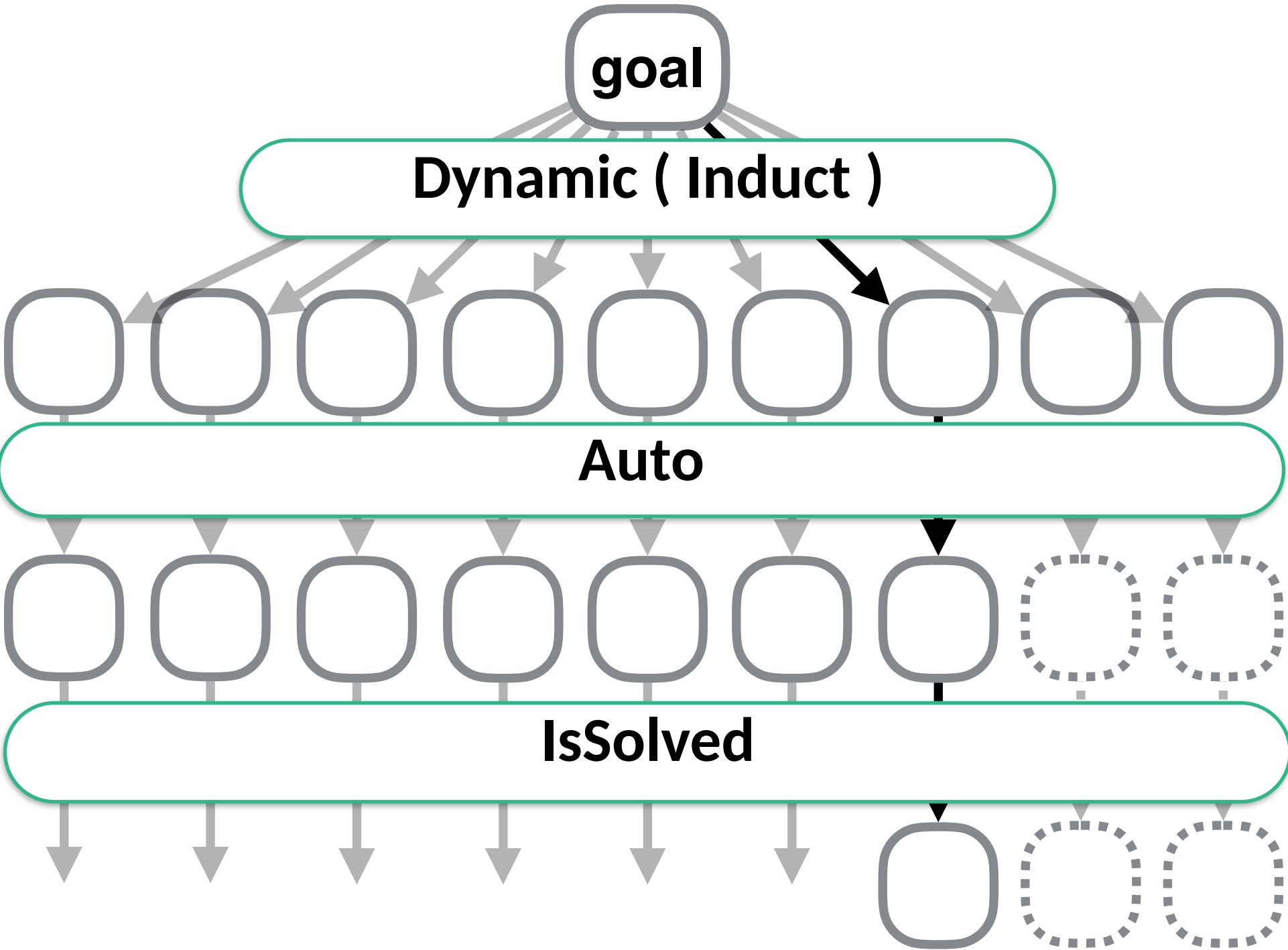
✓ Proof state ✓ Auto update Update Search:

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

Output Query Sledgehammer Symbols

```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

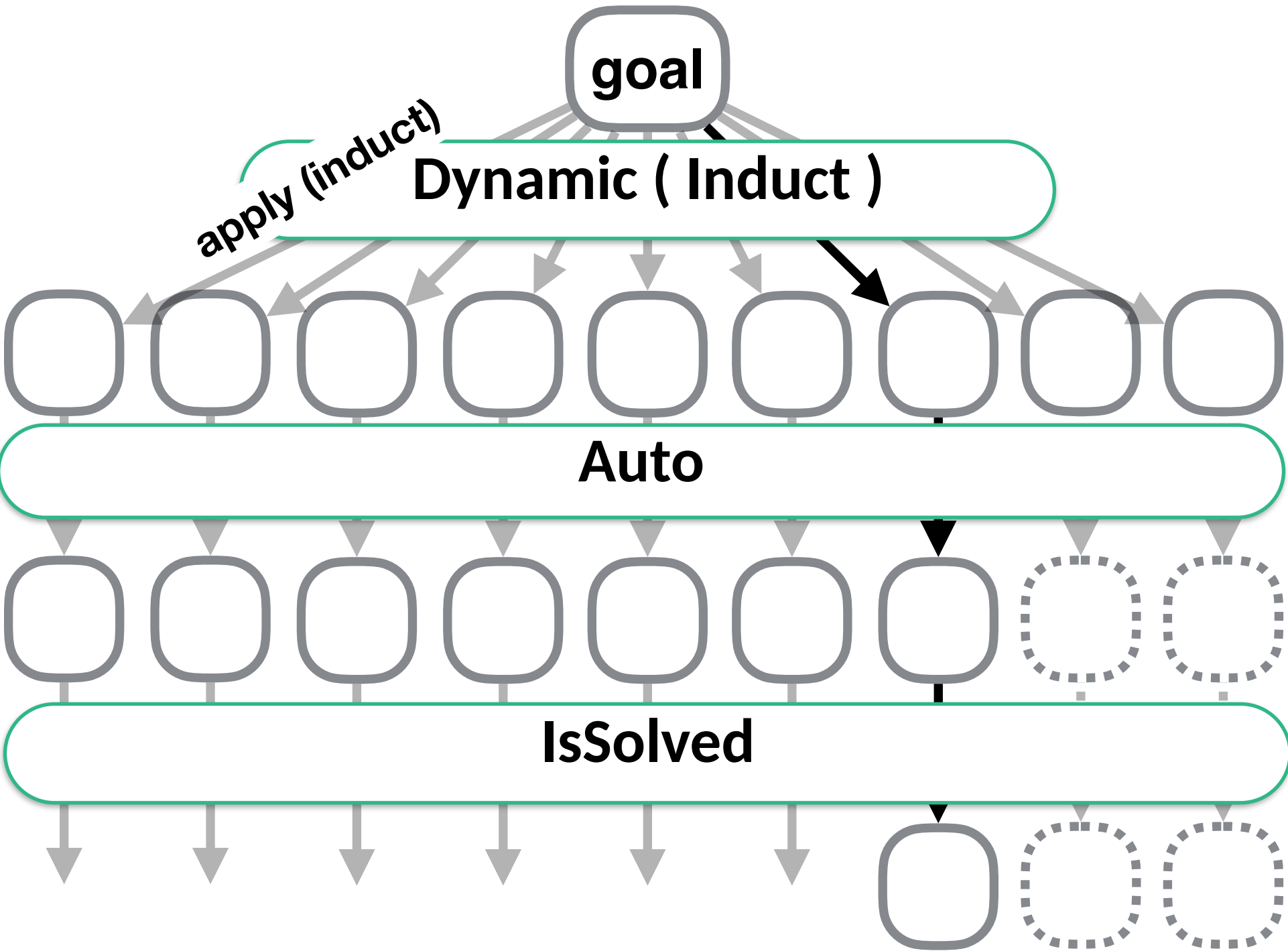
```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```





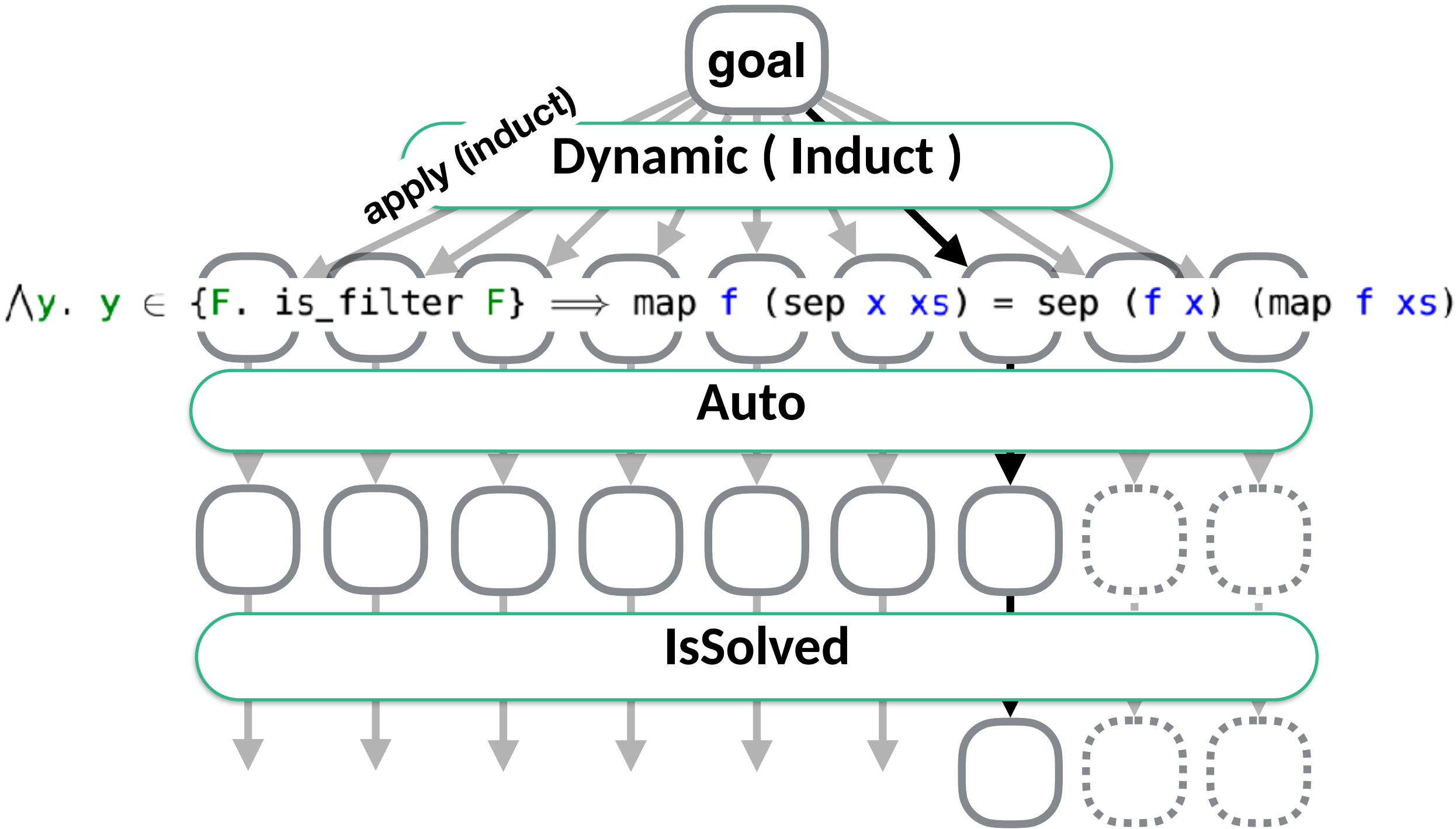
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

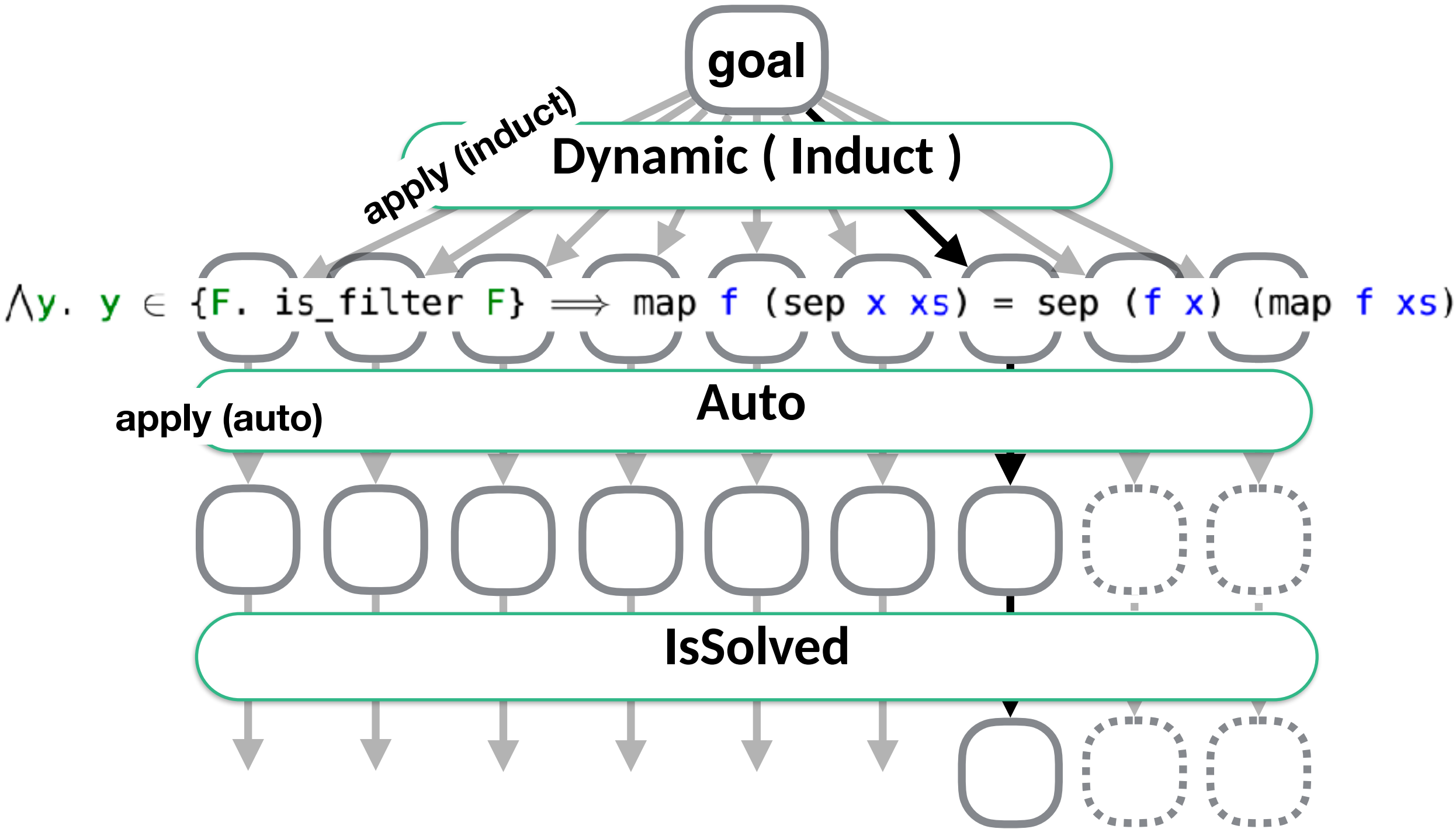
```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```





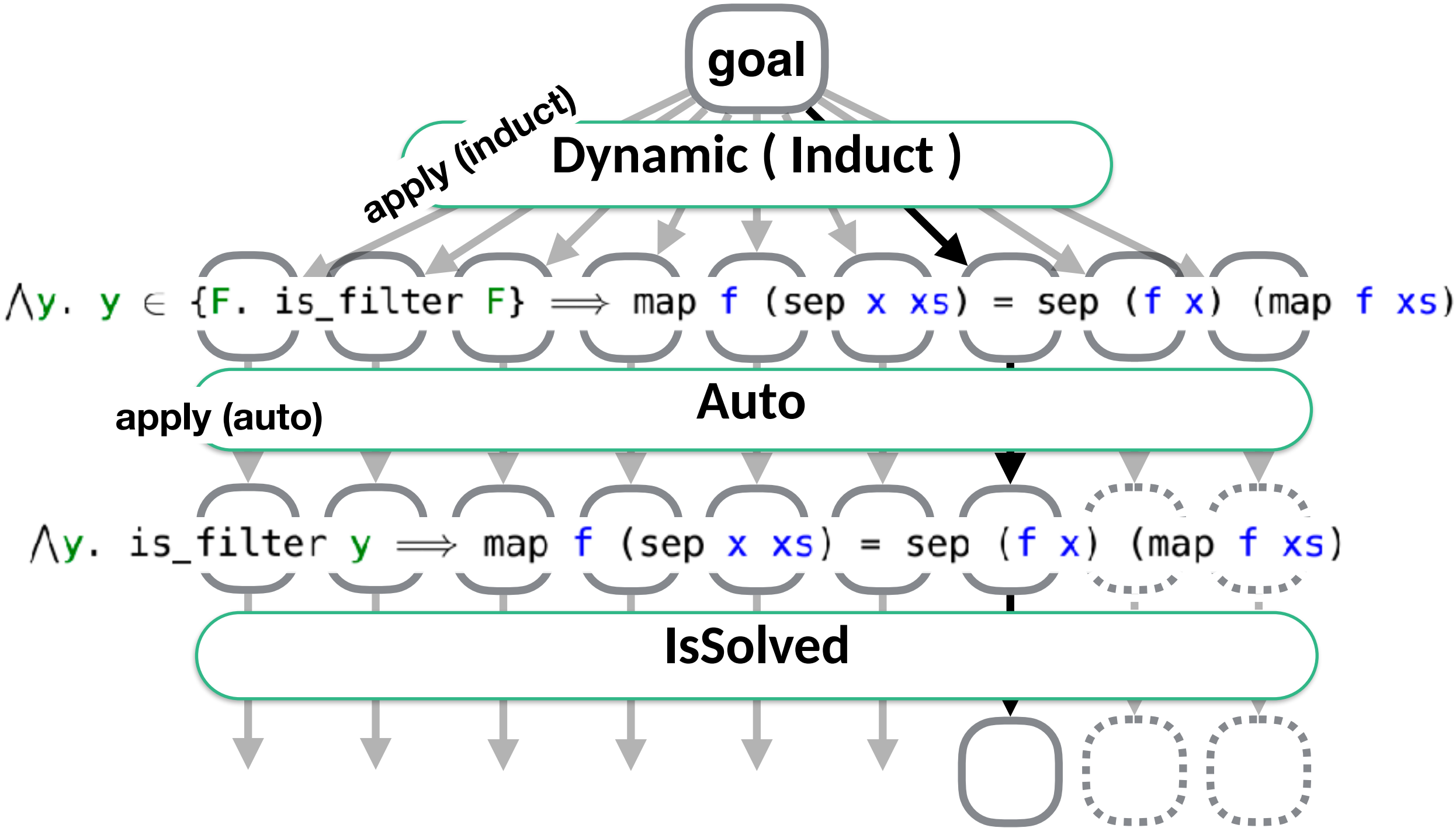
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



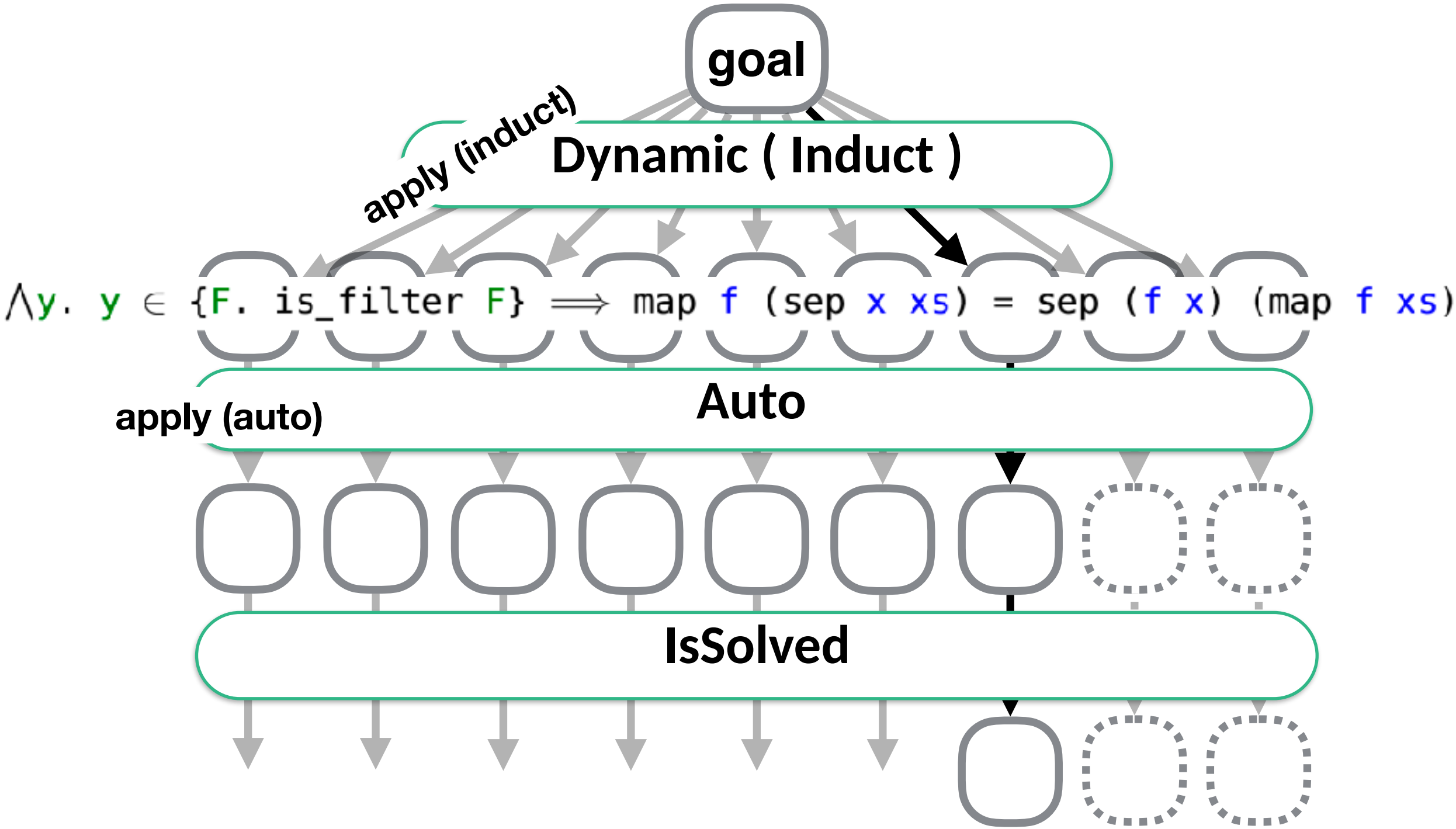
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



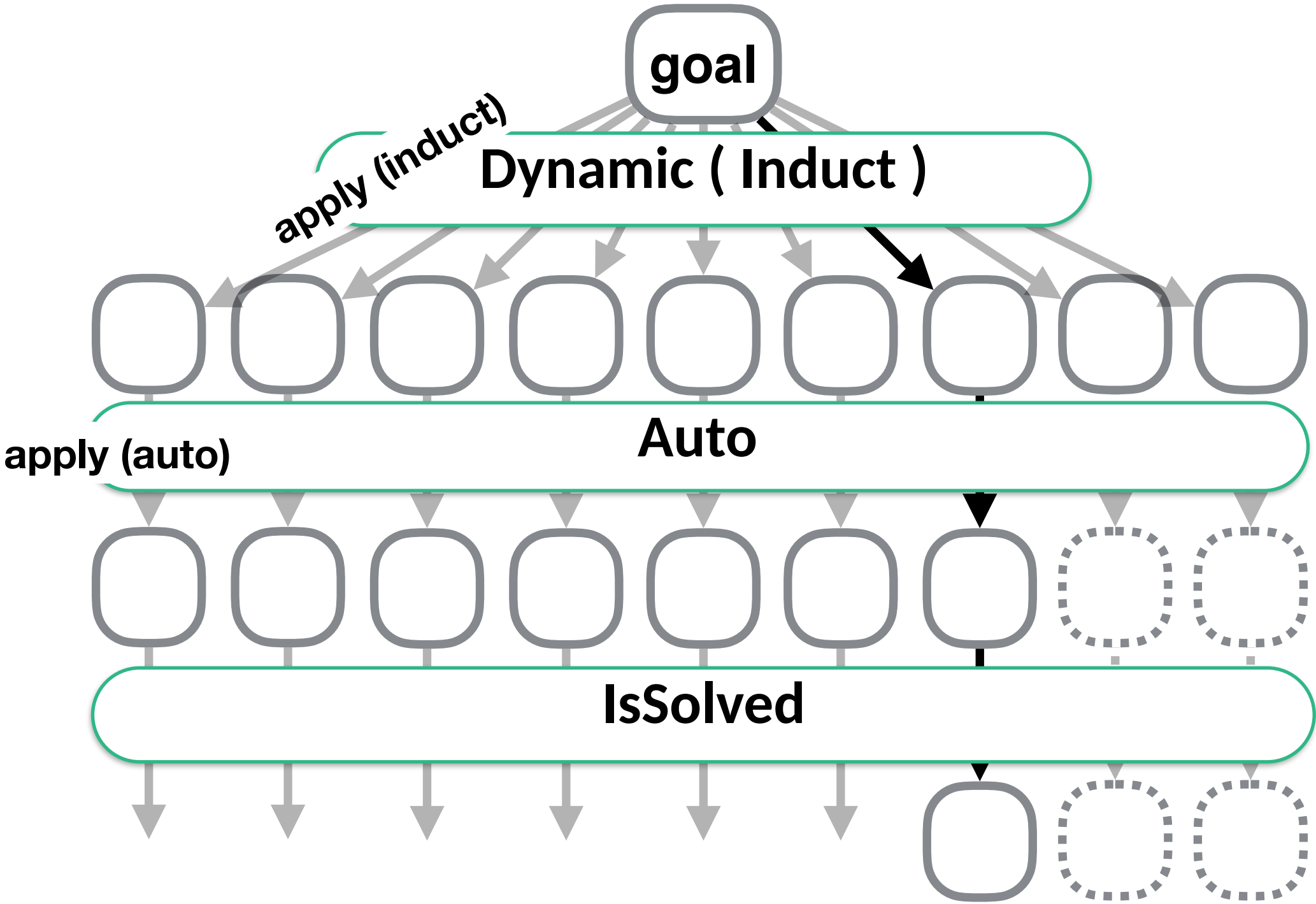
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



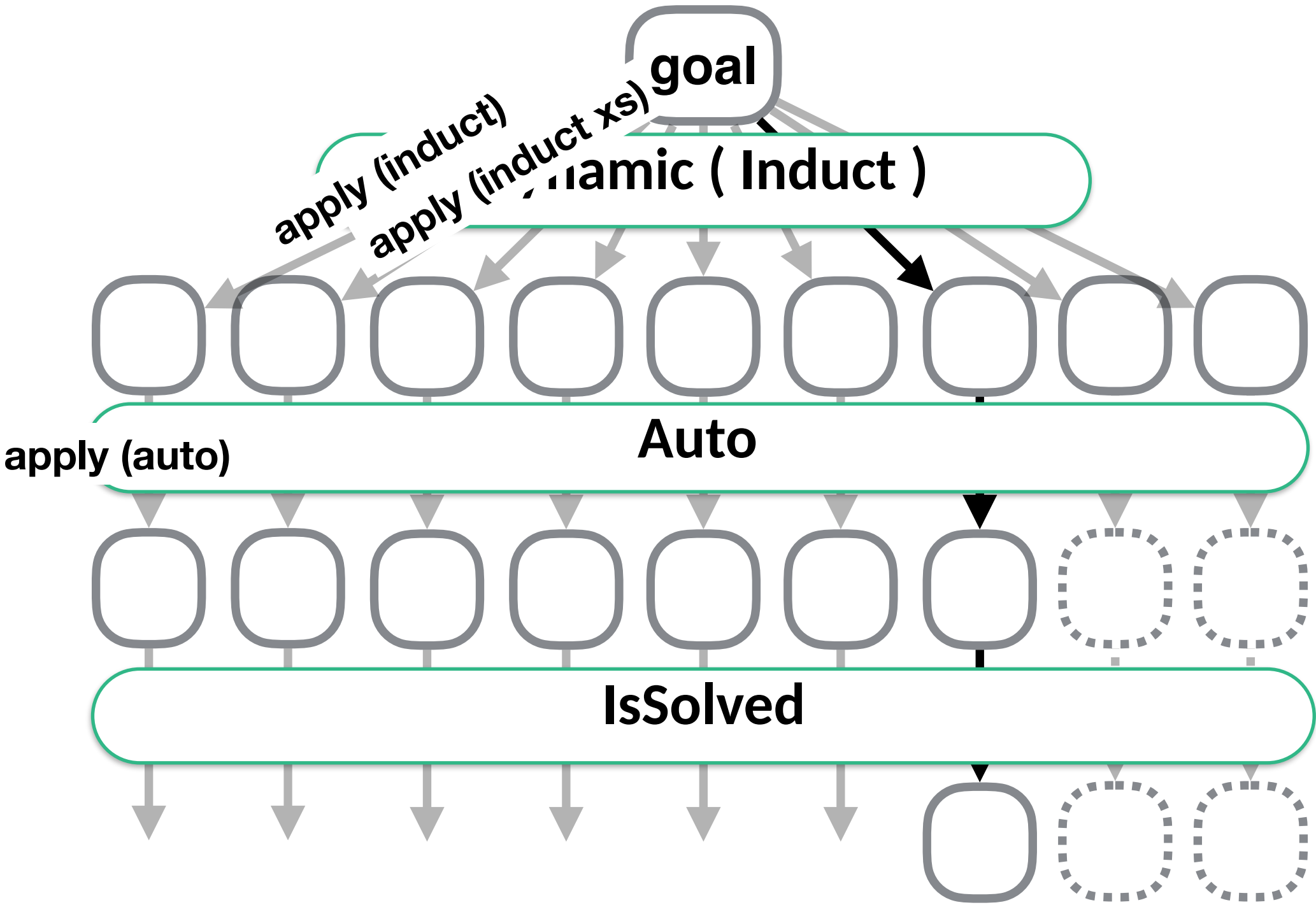
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



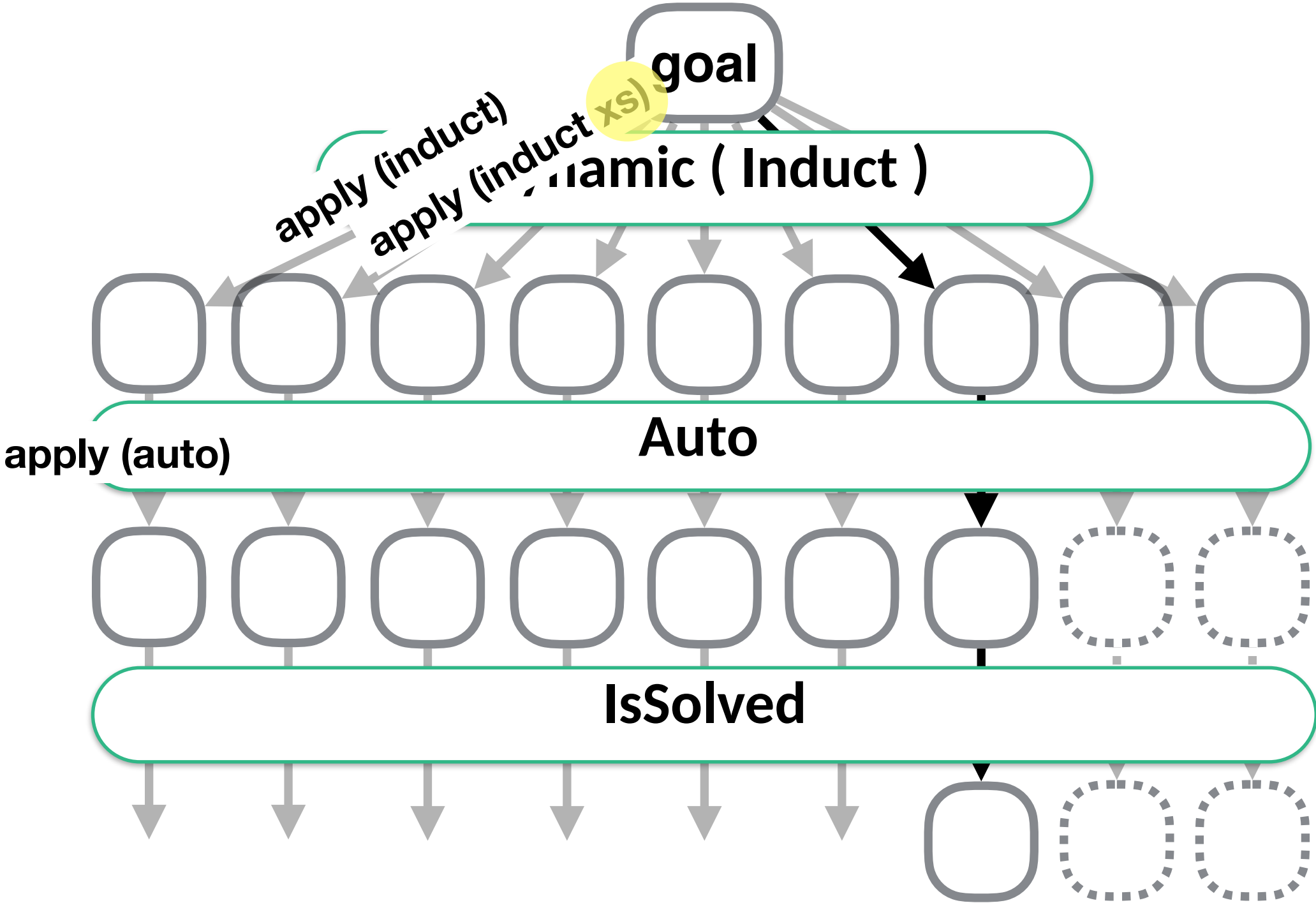
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



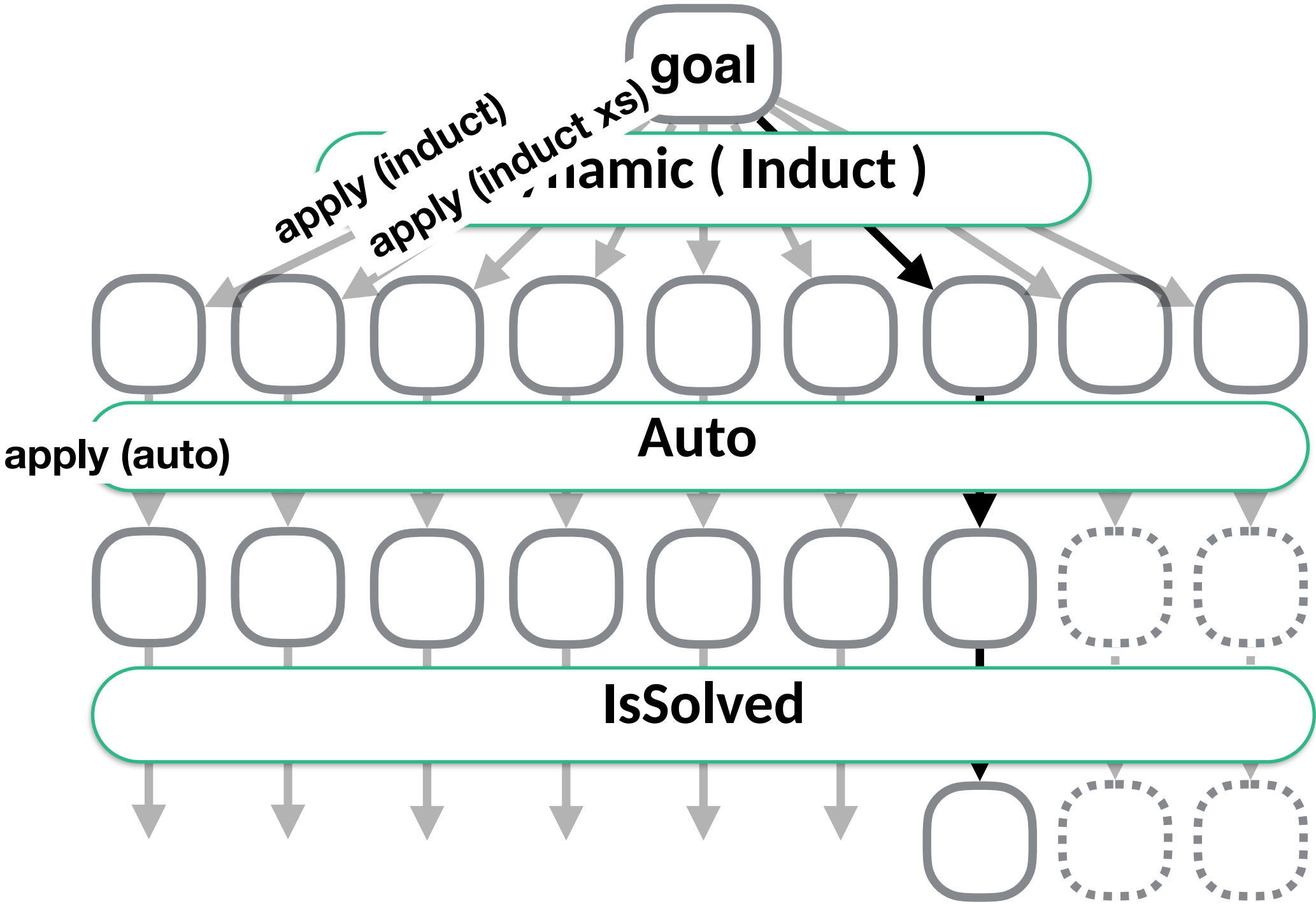
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



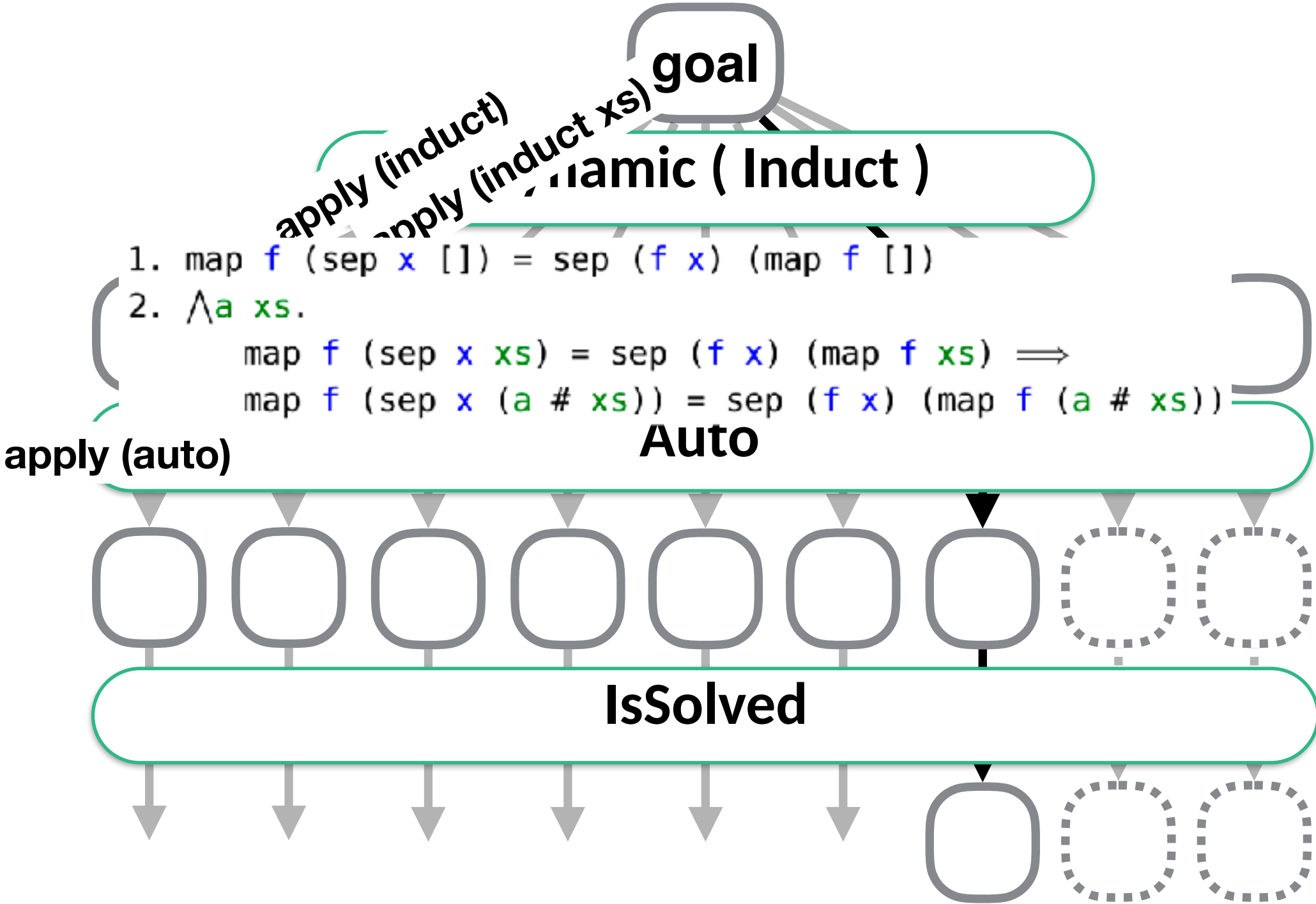
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

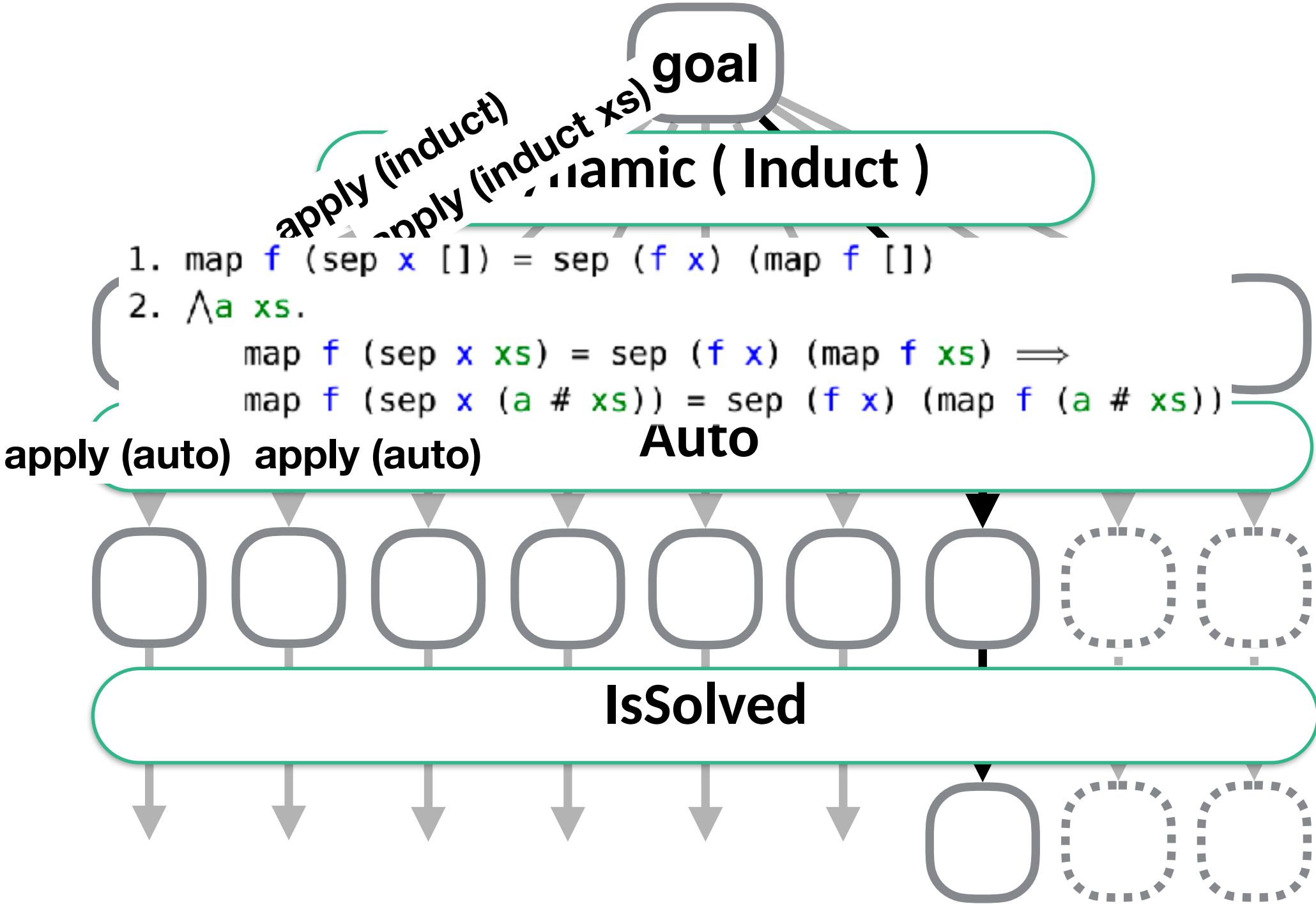
```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```





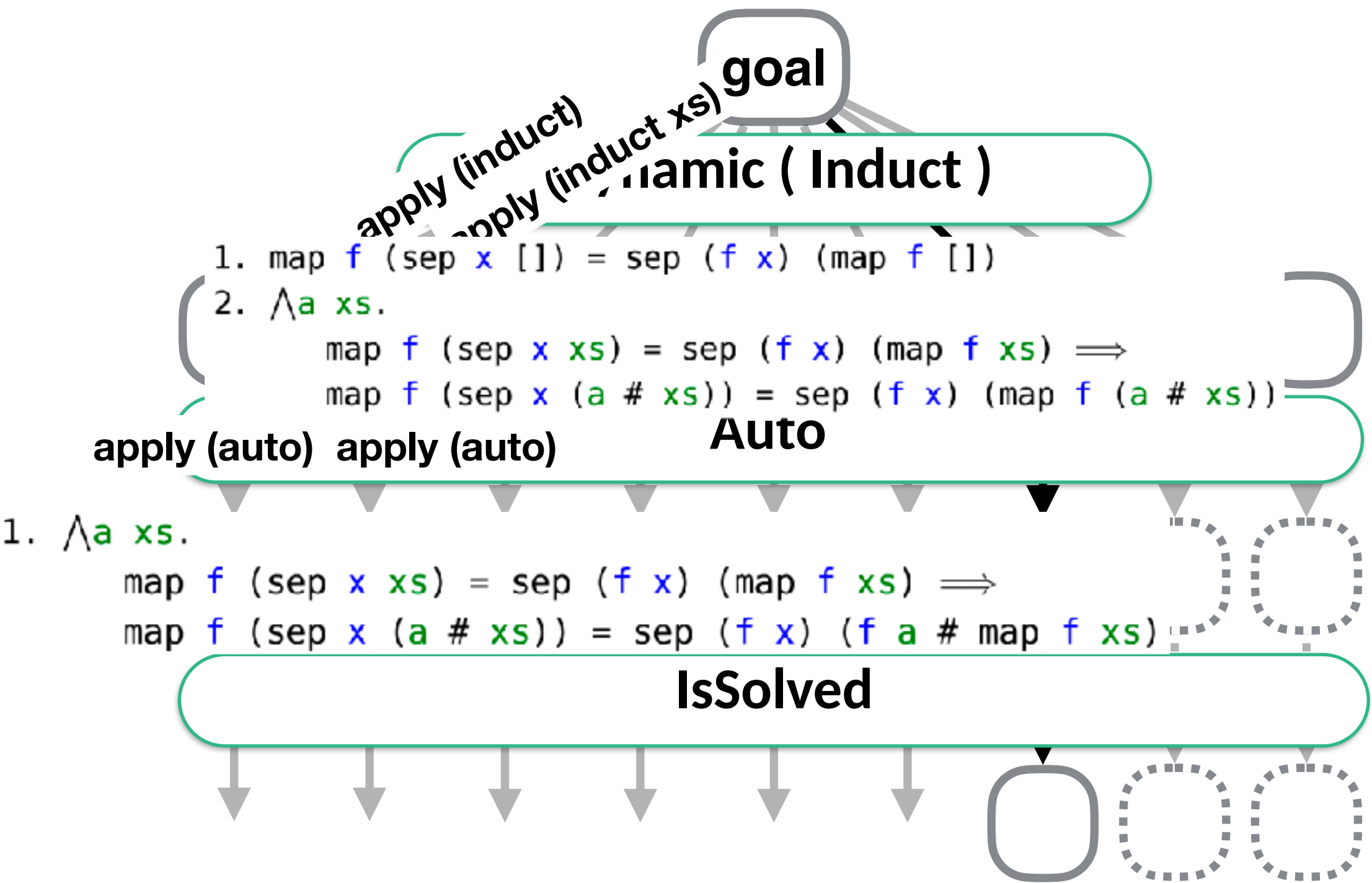
**lemma** "map f (sep x xs) = sep (f x) (map f xs)"

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



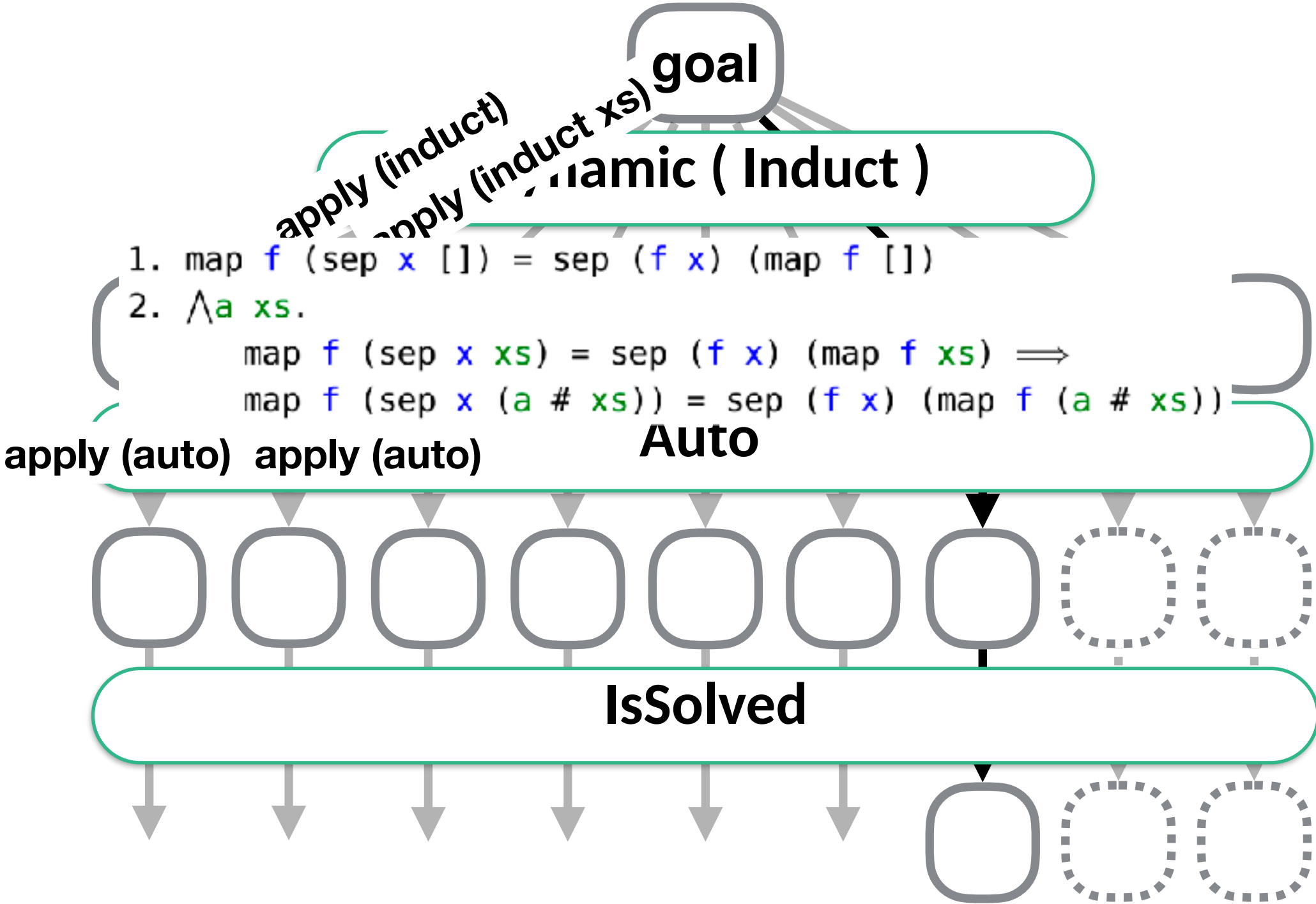
**lemma** "map f (sep x xs) = sep (f x) (map f xs)"

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



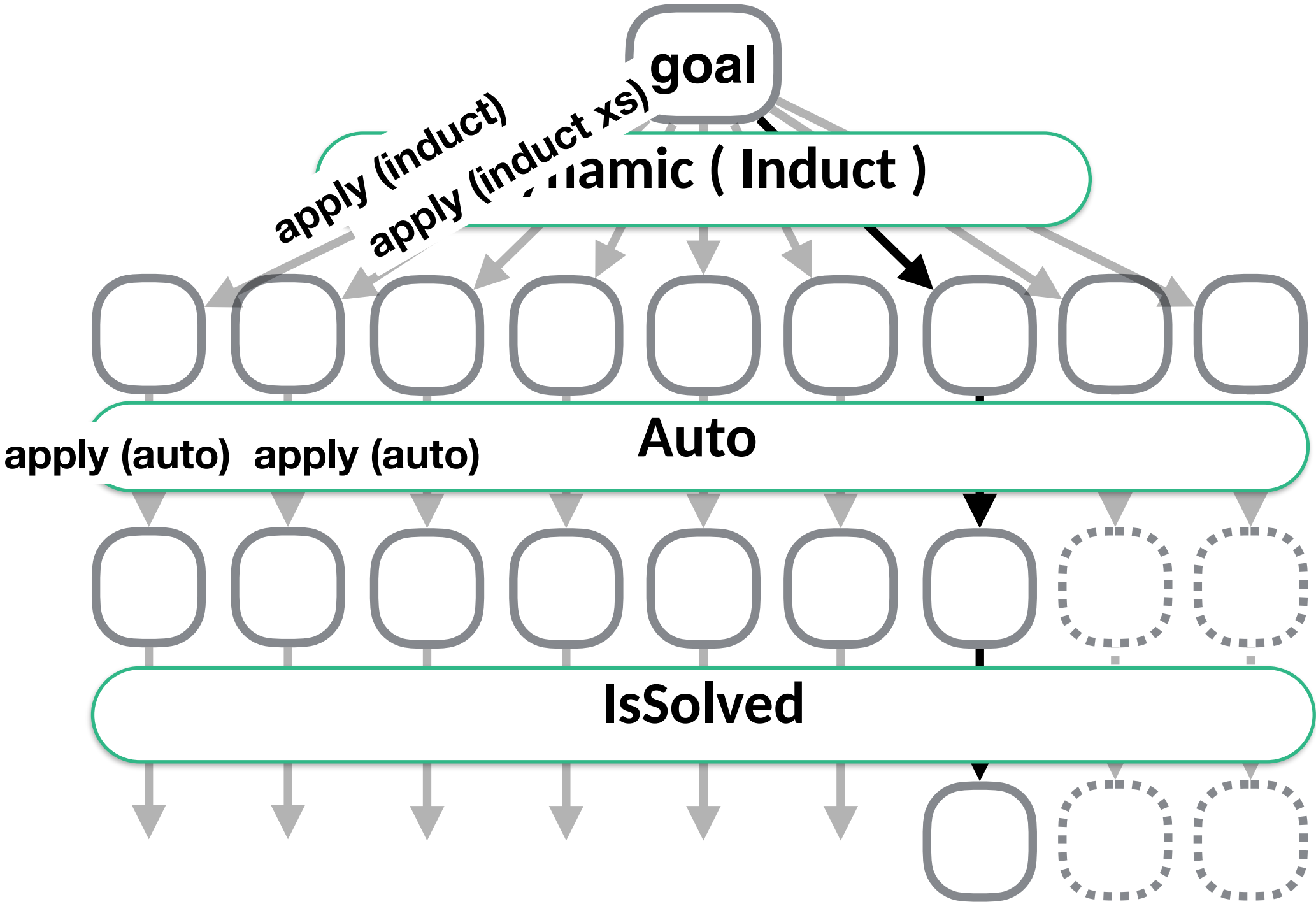
**lemma** "map f (sep x xs) = sep (f x) (map f xs)"

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



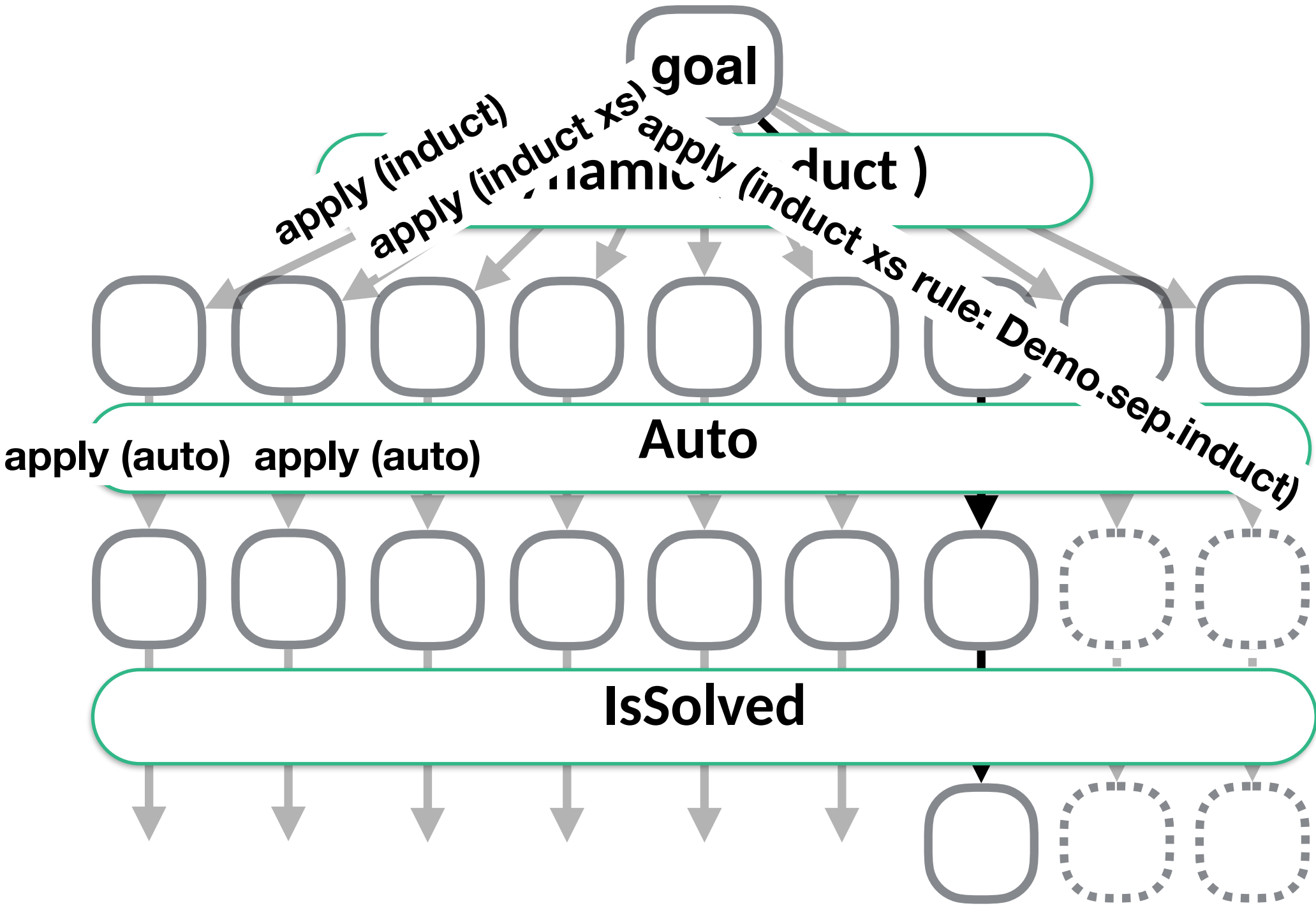
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



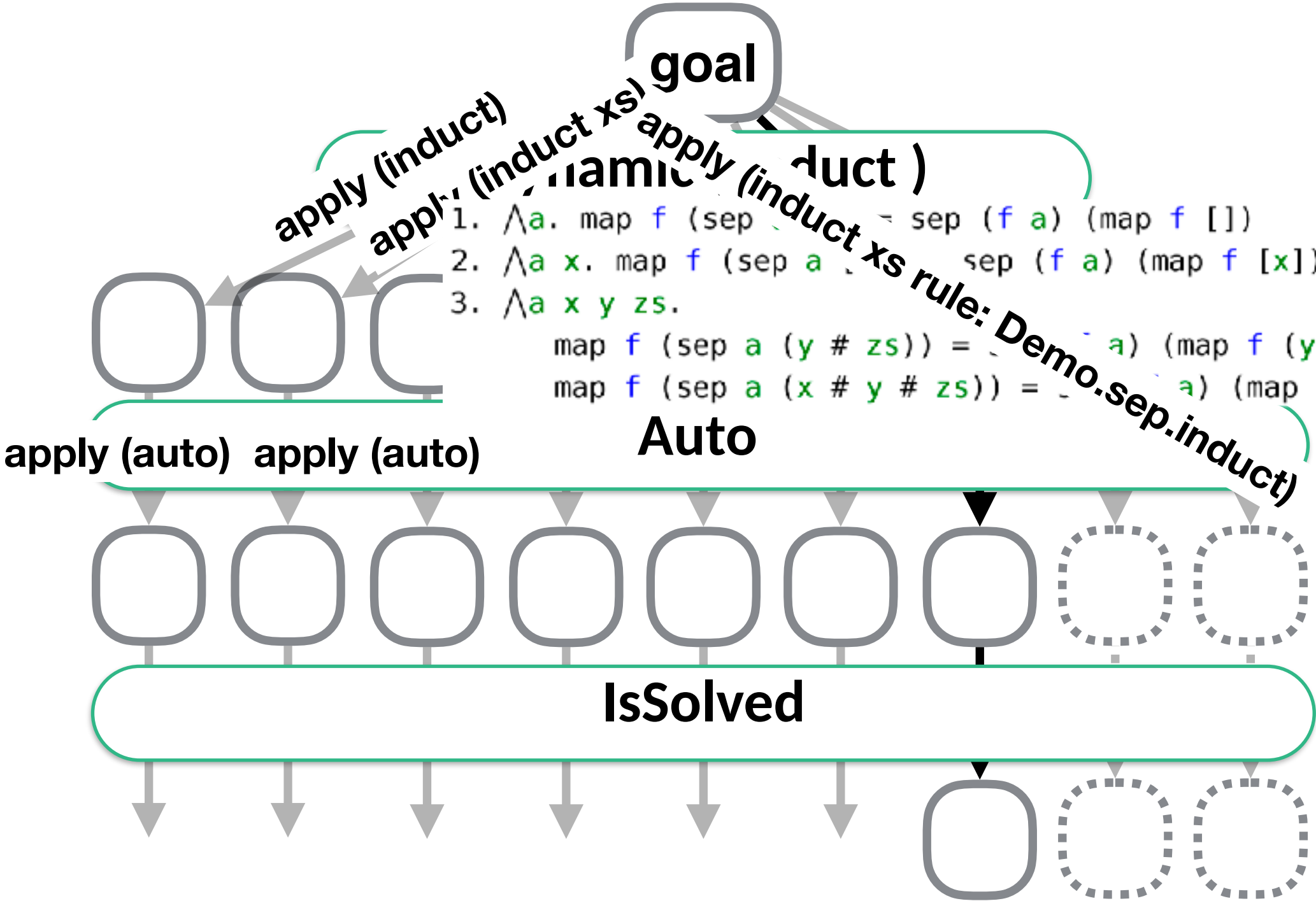
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



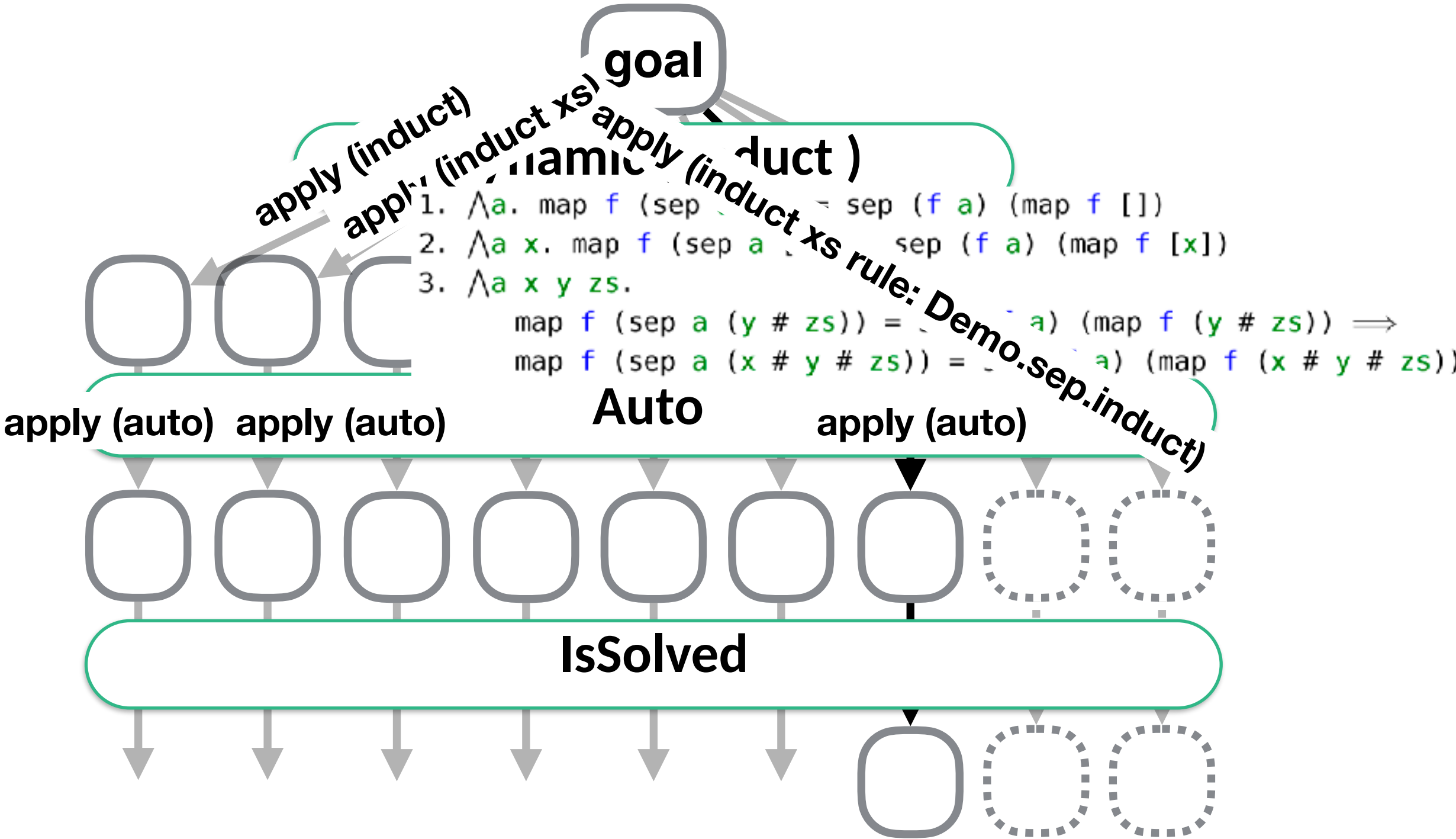
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



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

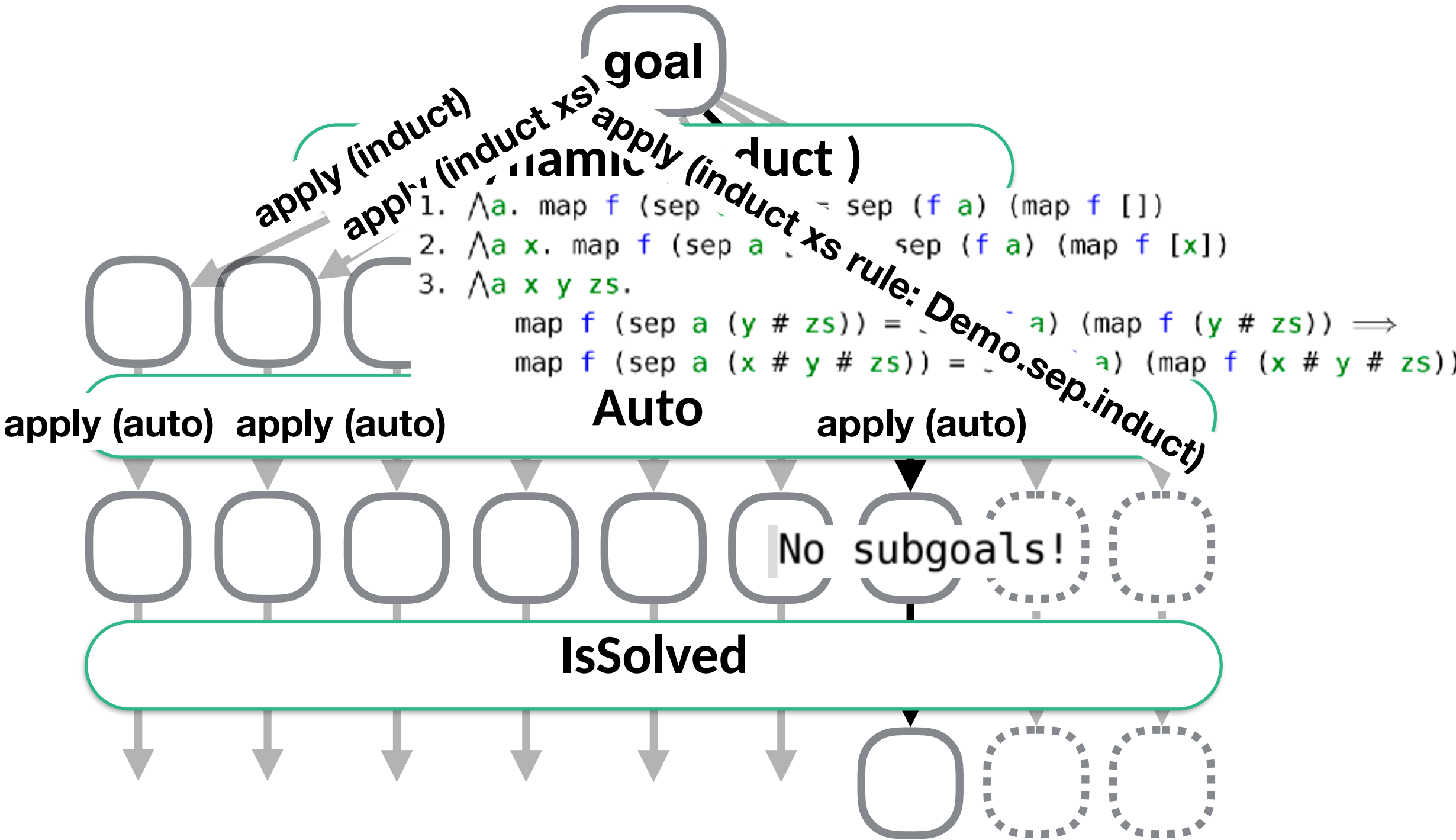
find\_proof DInd(\*= Thens [Dynamic (Induct), Auto, IsSolved]\*)





```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

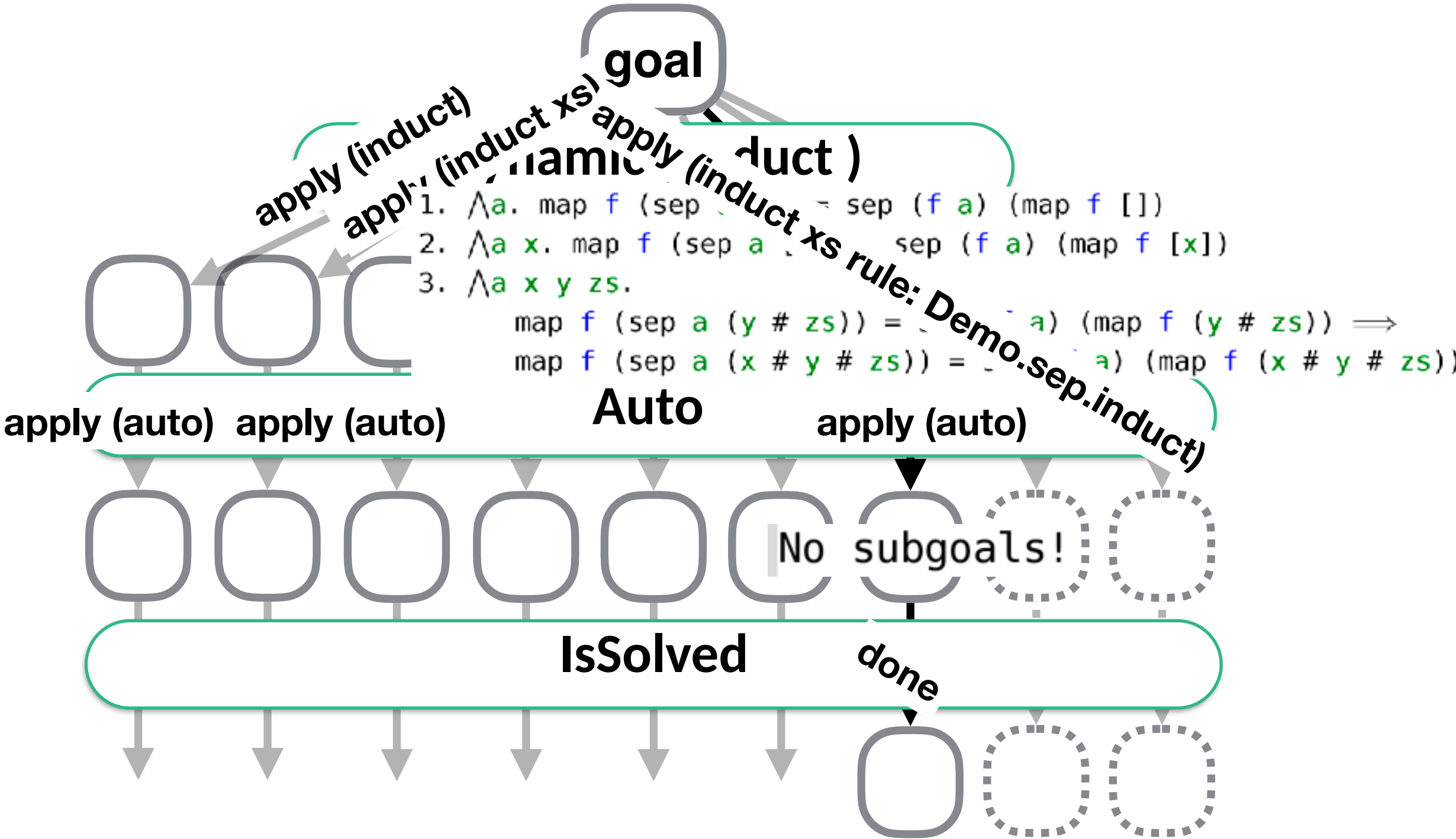
```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```





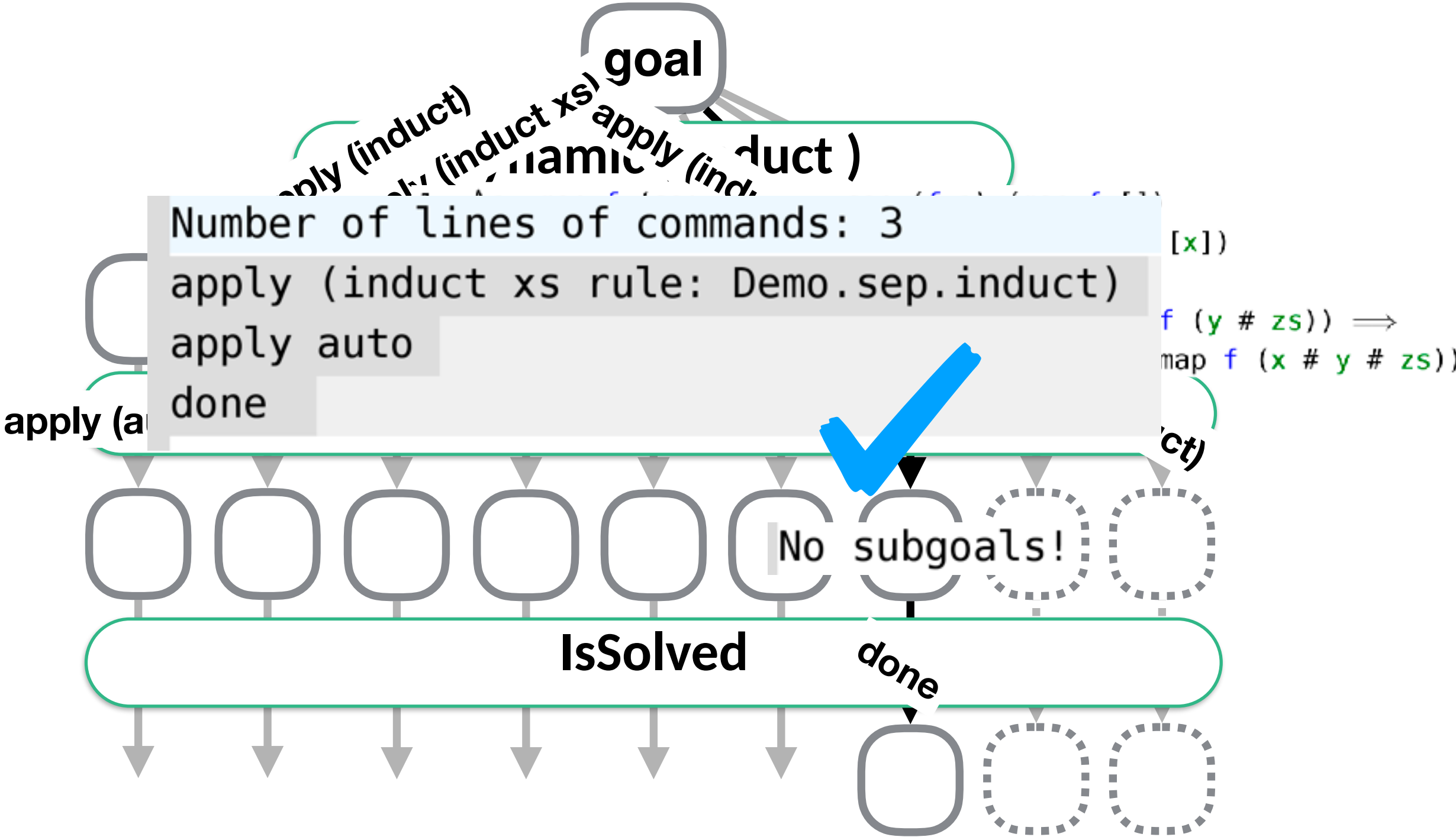
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

```
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]\*)



# Try\_Hard: the default strategy

```
strategy Basic =
```

```
Ors [
```

```
  Auto_Solve,
```

```
  Blast_Solve,
```

```
  FF_Solve,
```

```
  Thens [IntroClasses, Auto_Solve],
```

```
  Thens [Transfer, Auto_Solve],
```

```
  Thens [Normalization, IsSolved],
```

```
  Thens [DInduct, Auto_Solve],
```

```
  Thens [Hammer, IsSolved],
```

```
  Thens [DCases, Auto_Solve],
```

```
  Thens [DCoinduction, Auto_Solve],
```

```
  Thens [Auto, RepeatN(Hammer), IsSolved],
```

```
  Thens [DAuto, IsSolved]]
```

```
strategy Try_Hard =
```

```
Ors [Thens [Subgoal, Basic],
```

```
      Thens [DInductTac, Auto_Solve],
```

```
      Thens [DCaseTac, Auto_Solve],
```

```
      Thens [Subgoal, Advanced],
```

```
      Thens [DCaseTac, Solve_Many],
```

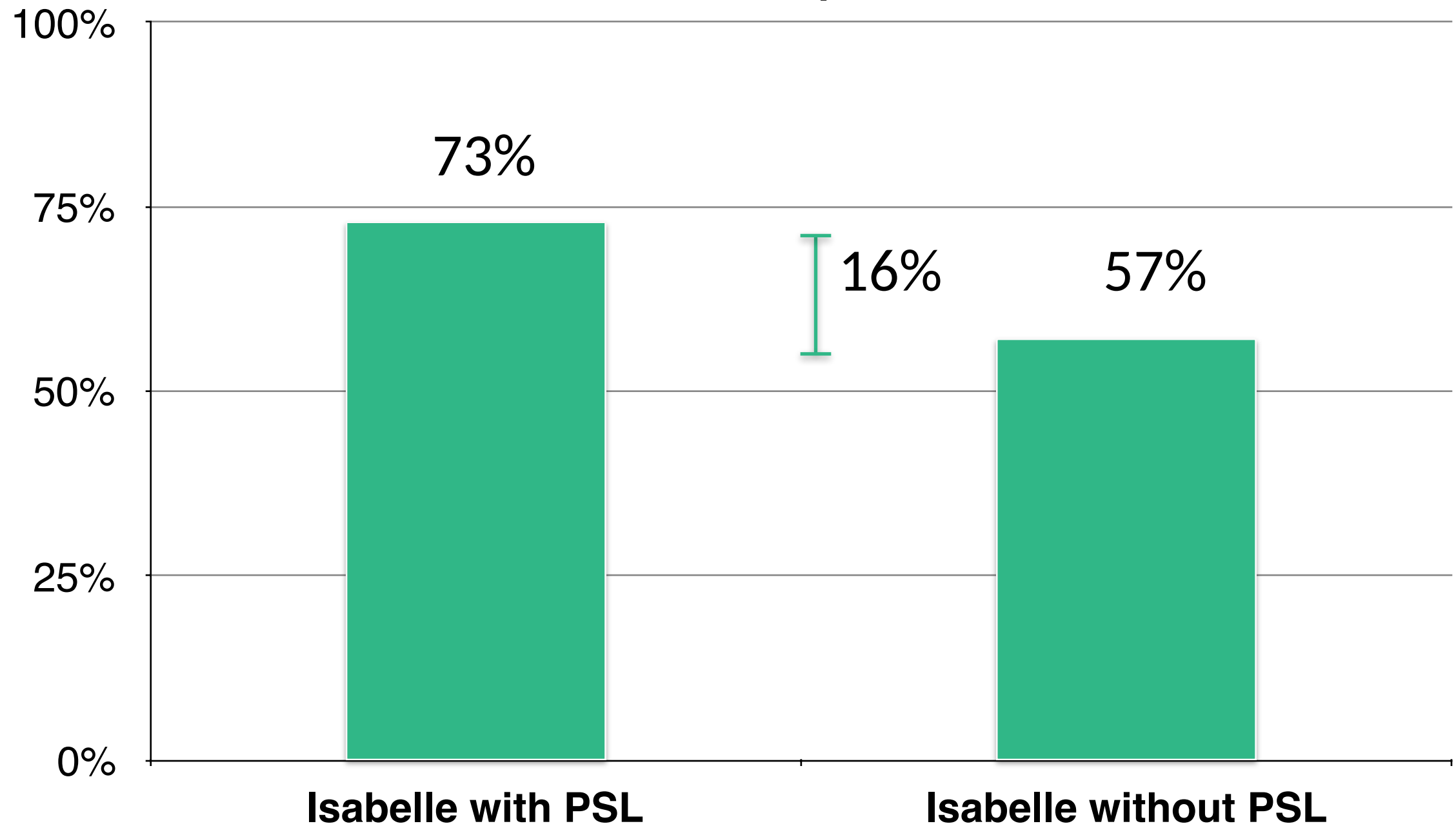
```
      Thens [DInductTac, Solve_Many] ]
```

# Evaluation

but the search space explodes



**PaMpeR: Proof Method Recommendation**

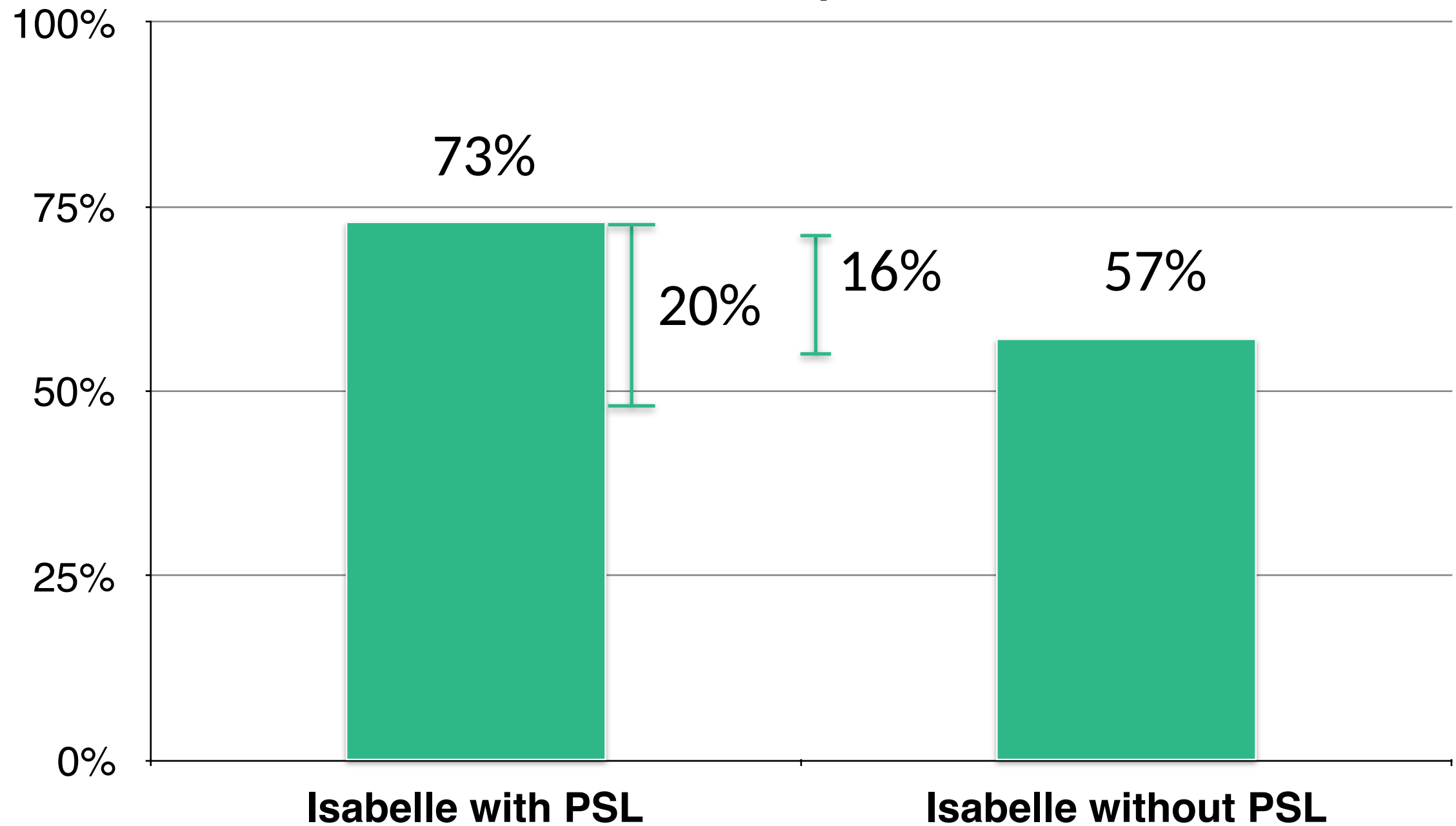


# Evaluation

but the search space explodes



**PaMpeR: Proof Method Recommendation**



preparation phase

**How does  
PaMpeR work?**

recommendation phase

preparation phase

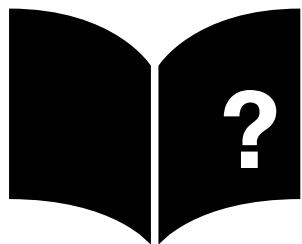
large proof corpora



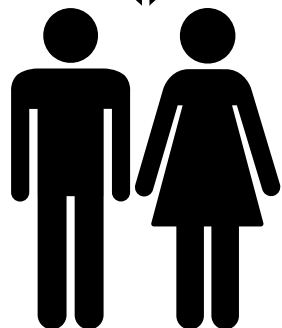
AFP and standard library

**How does  
PaMpeR work?**

recommendation phase



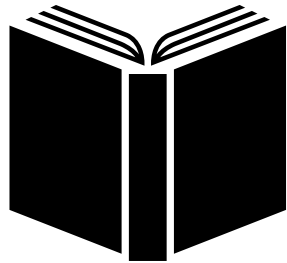
proof  
state



proof  
engineer

preparation phase

large proof corpora



AFP and standard library



STATISTICS

Archive of Formal Proofs (<https://www.isa-afp.org>)

## Statistics

Number of Articles: 468

Number of Authors: 313

Number of lemmas: ~128,900

Lines of Code: ~2,170,300

### Most used AFP articles:

	Name	Used by ? articles
1.	<a href="#">Collections</a>	15
2.	<a href="#">List-Index</a>	14
3.	<a href="#">Coinductive</a>	12

Home

About

Submission

Updating  
Entries

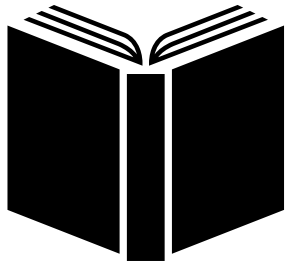
Using Entries

Search



preparation phase

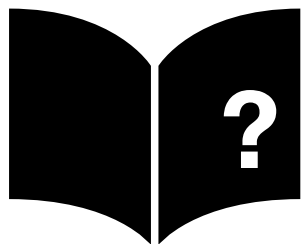
large proof corpora



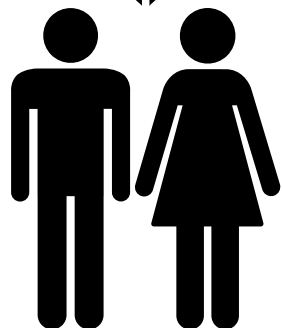
AFP and standard library

**How does  
PaMpeR work?**

recommendation phase



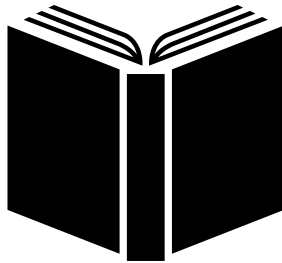
proof  
state



proof  
engineer

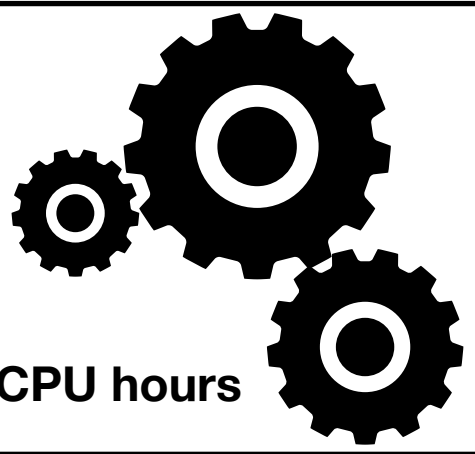
## preparation phase

large proof corpora



AFP and standard library

full feature extractor



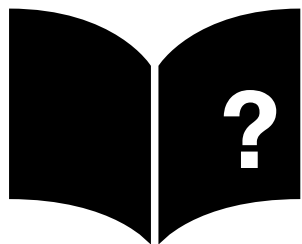
6021 CPU hours

108 assertions

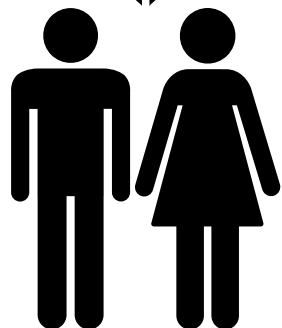


**How does  
PaMpeR work?**

## recommendation phase



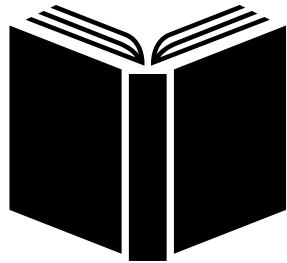
proof  
state



proof  
engineer

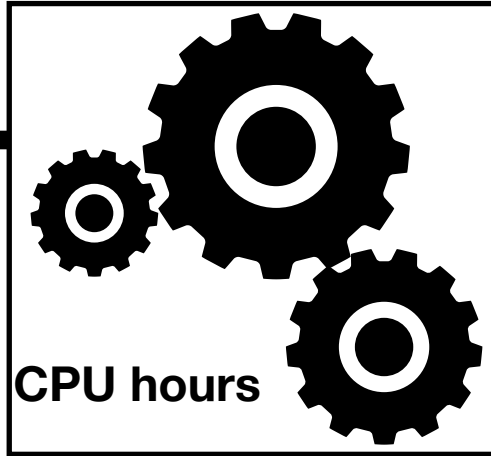
preparation phase

large proof corpora



AFP and standard library

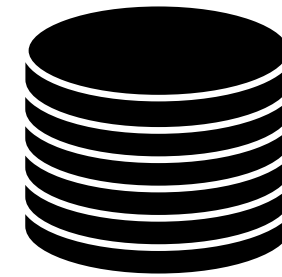
full feature extractor



6021 CPU hours

108 assertions

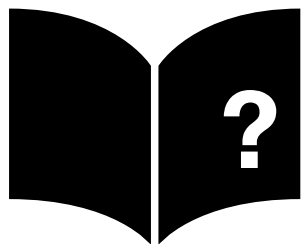
database ( 425334 data points )



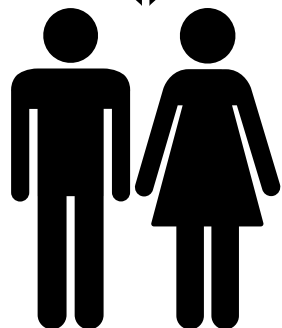
:: ( tactic\_name, [ bool ] )

How does  
PaMpeR work?

recommendation phase



proof  
state



proof  
engineer

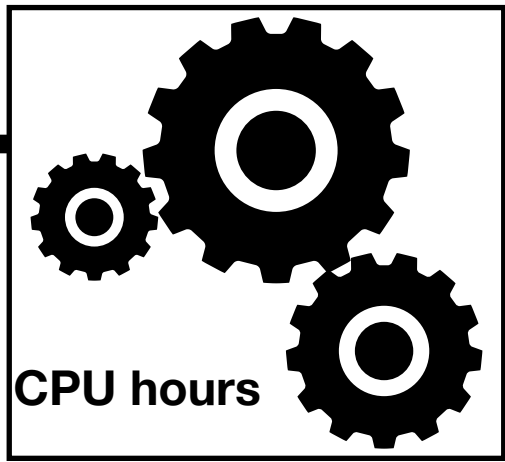
preparation phase

large proof corpora



AFP and standard library

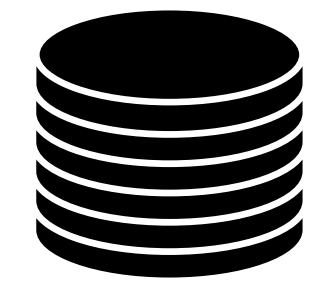
full feature extractor



6021 CPU hours

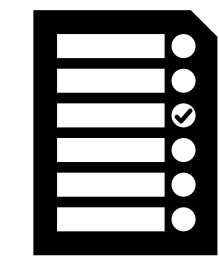
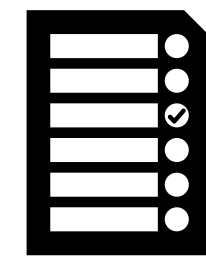
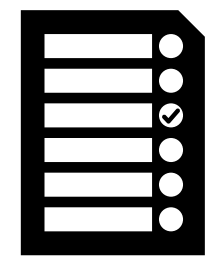
108 assertions

database ( 425334 data points )

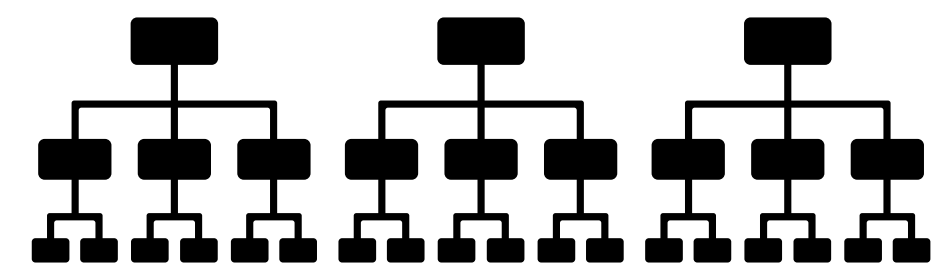


:: ( tactic\_name, [ bool ] )

preprocess

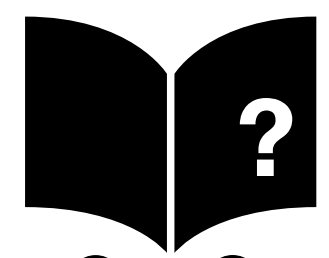


decision tree construction

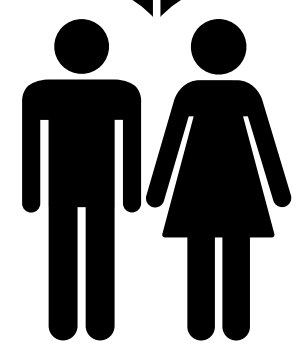


How does  
PaMpeR work?

recommendation phase



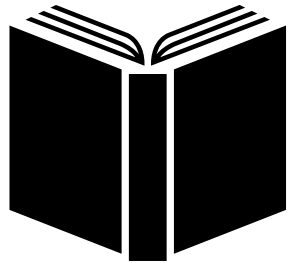
proof  
state



proof  
engineer

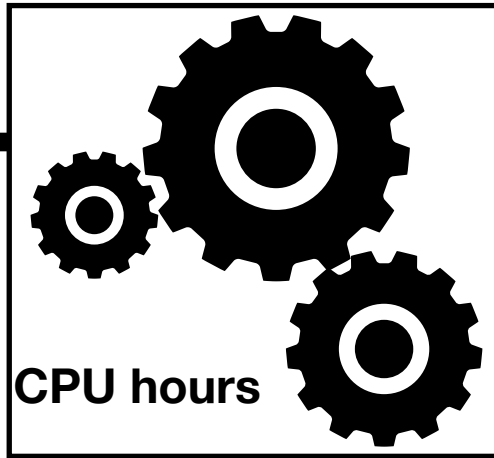
preparation phase

large proof corpora



AFP and standard library

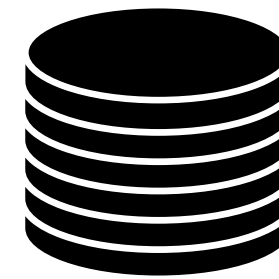
full feature extractor



6021 CPU hours

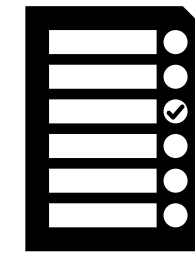
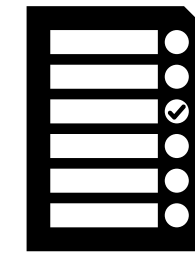
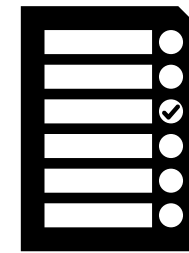
108 assertions

database ( 425334 data points )

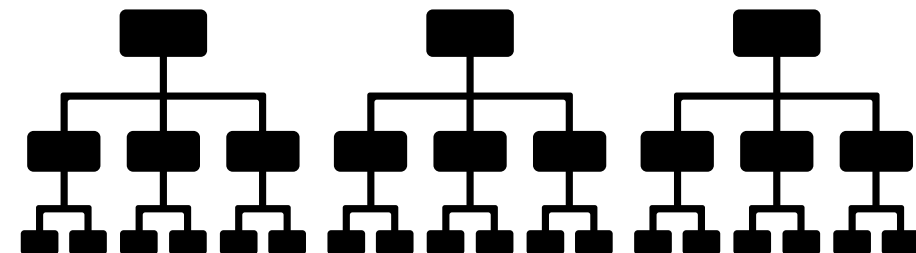


:: ( tactic\_name, [ bool ] )

preprocess



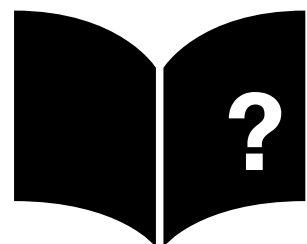
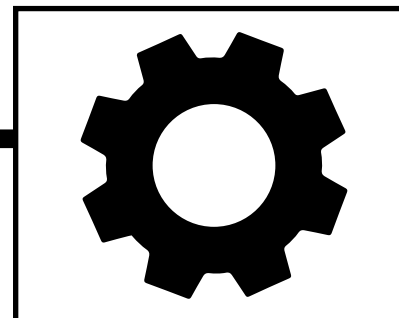
decision tree construction



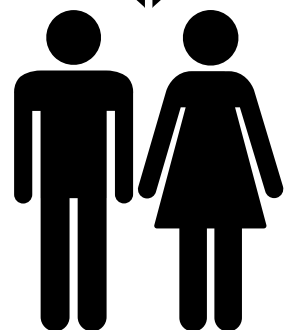
How does  
PaMpeR work?

recommendation phase

fast feature extractor



proof  
state



proof  
engineer

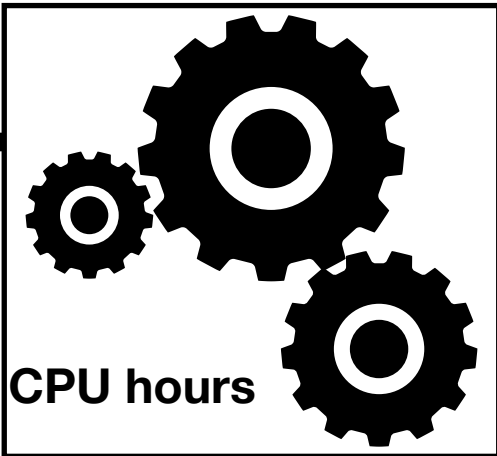
**preparation phase**

large proof corpora



AFP and standard library

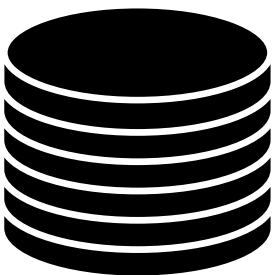
full feature extractor



6021 CPU hours

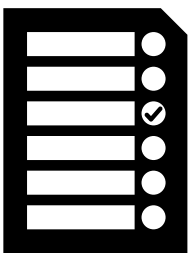
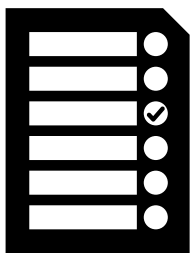
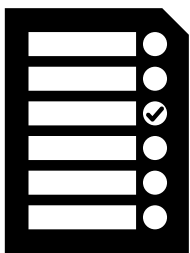
108 assertions

database ( 425334 data points )

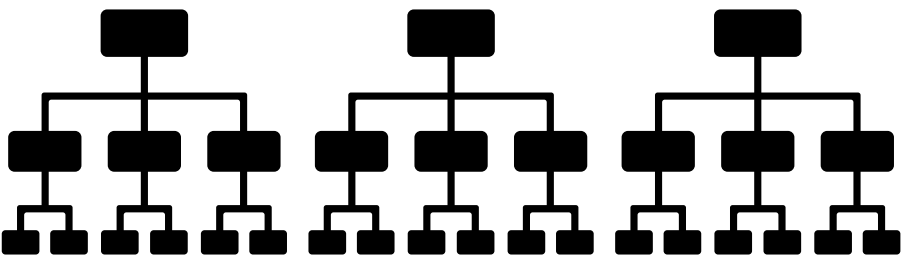


:: ( tactic\_name, [ bool ] )

preprocess



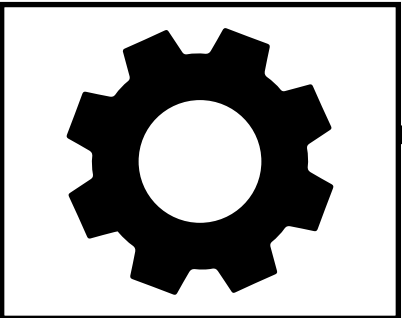
decision tree construction



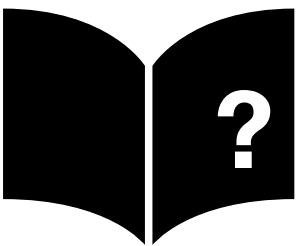
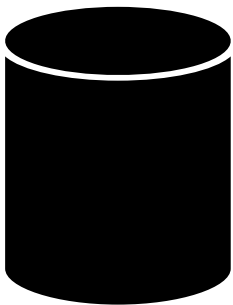
**How does  
PaMpeR work?**

**recommendation phase**

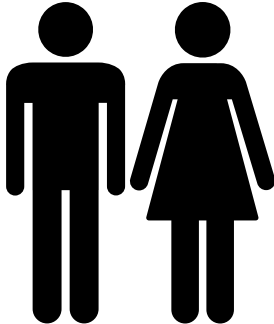
fast feature extractor



feature vector



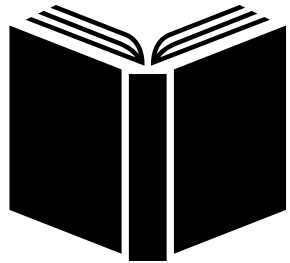
proof  
state



proof  
engineer

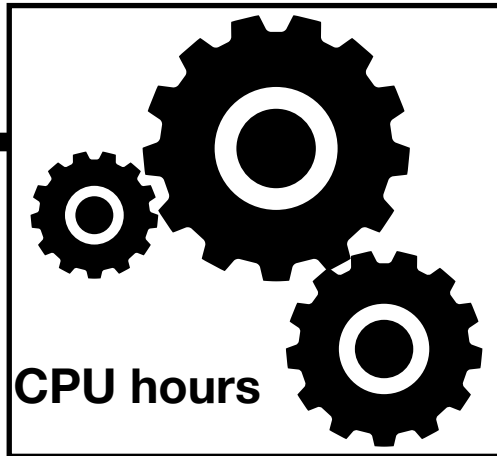
preparation phase

large proof corpora



AFP and standard library

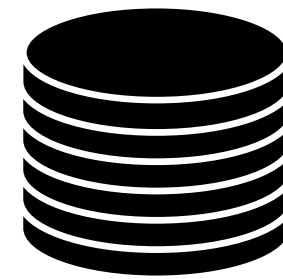
full feature extractor



6021 CPU hours

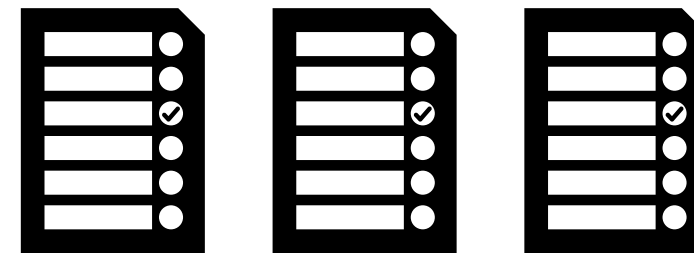
108 assertions

database ( 425334 data points )

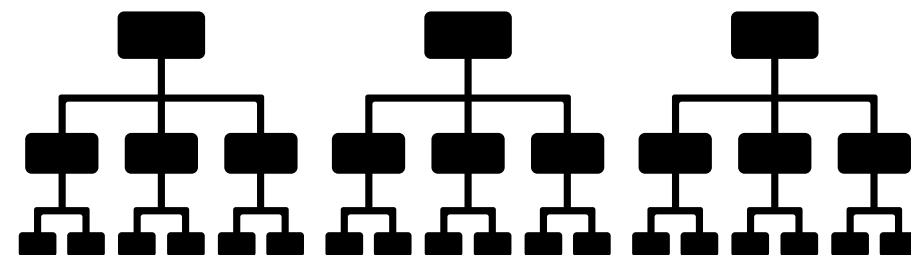


:: ( tactic\_name, [ bool ] )

preprocess



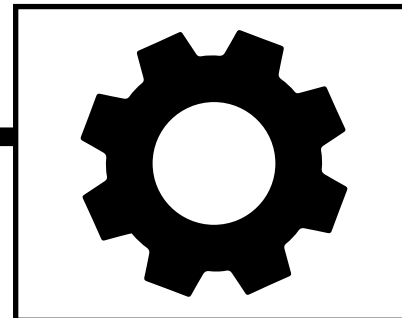
decision tree construction



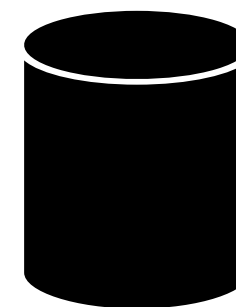
How does  
PaMpeR work?

recommendation phase

fast feature extractor

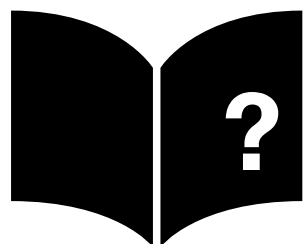


feature vector

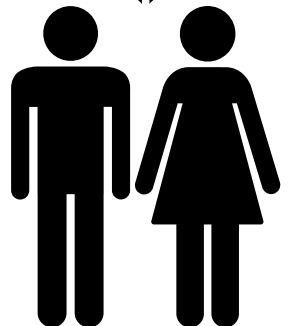


lookup

proof  
state



proof  
engineer



proof method  
recommendation



# Summary

PSL can find how to apply induction for easy problems.

**CADE2017** ([https://link.springer.com/10.1007/978-3-319-63046-5\\_32](https://link.springer.com/10.1007/978-3-319-63046-5_32))

PaMpeR recommends which proof methods to use.

**ASE2018** (<https://dx.doi.org/10.1145/3238147.3238210>)

