https://twitter.com/YutakangE

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

universität innsbruck

CZECH INSTITUTE OF INFORMATICS ROBOTICS AND CYBERNETICS CTU IN PRAGUE

Yutaka Ng
yutakang

Block or report user

CVUT, CTU, CIIRC

https://twitter.com/YutakangE

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

universität innsbruck

Yutaka Ng
yutakang

Block or report user

CVUT, CTU, CIIRC

CZECH INSTITUTE
OF INFORMATICS
ROBOTICS AND
CYBERNETICS
CTU IN PRAGUE

Why theorem proving?

https://twitter.com/YutakangE

Why theorem proving?

To build trustworthy software (Complete Formal Verification)!

Why theorem proving?

To build trustworthy software (Complete Formal Verification)!

1. **Specify what we want.**

https://twitter.com/YutakangE

Why theorem proving?

To build trustworthy software (Complete Formal Verification)!

1. Specify what we want.

2. Implement what we want.

Why theorem proving?

To build trustworthy software (Complete Formal Verification)!

1. **Specify what we want.**
2. **Implement what we want.**
3. **Prove the implementation satisfies the specification.**

https://twitter.com/YutakangE

Why theorem proving?

To build trustworthy software (Complete Formal Verification)!

1. Specify what we want.

2. Implement what we want.

3. Prove the implementation satisfies the specification.

Example

https://twitter.com/YutakangE

Why theorem proving?

To build trustworthy software (Complete Formal Verification)!

1. Specify what we want.
2. Implement what we want.
3. Prove the implementation satisfies the specification.

Example

developer proof assistant / ITP implementation

Gewin Klein et. al

+

Isabelle/HOL

=

seL4 Security. Performance. Proof.

verified micro kernel,
seL4

https://twitter.com/YutakangE

Why theorem proving?

To build trustworthy software (Complete Formal Verification)!

1. Specify what we want.
2. Implement what we want.
3. Prove the implementation satisfies the specification.

Example

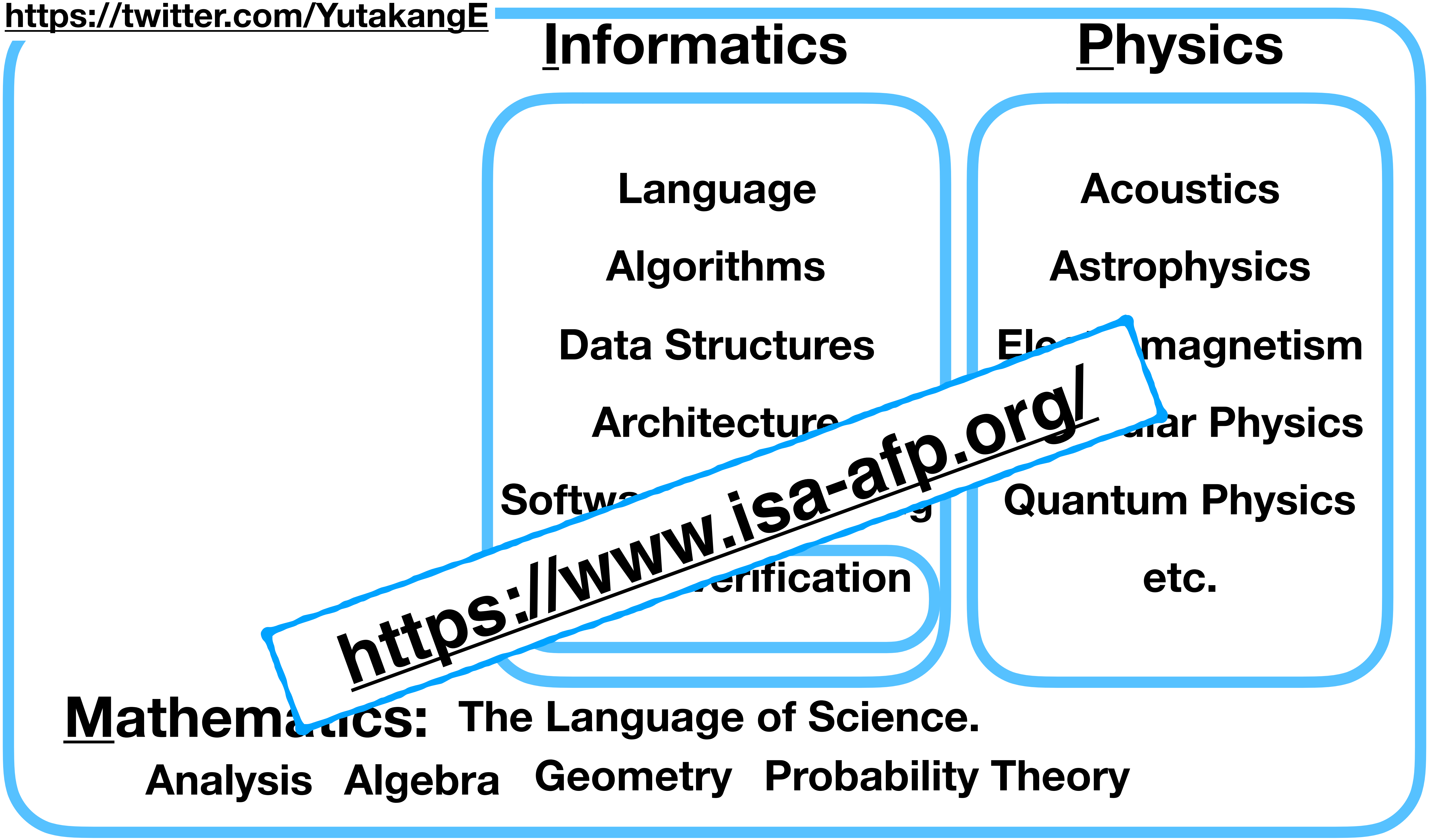
developer proof assistant / ITP implementation

+ =

Security. Performance. Proof.

Gewin Klein et. al Isabelle/HOL

verified micro kernel,
seL4



Informatics

Language

Algorithms

Data Structures

Architecture

Softw g

erification

Physics

Acoustics

Astrophysics

Ele magnetism

lar Physics

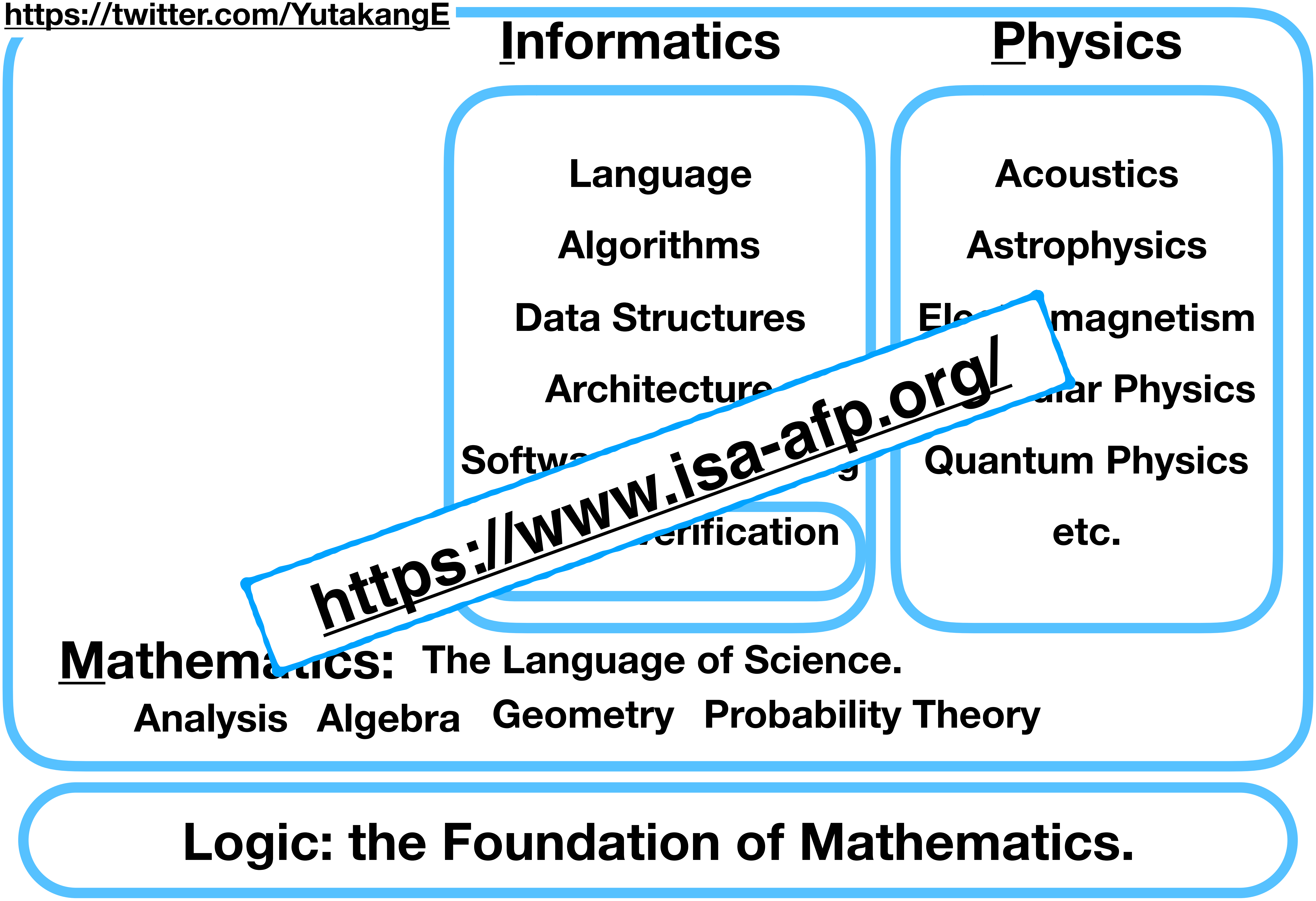
Quantum Physics

etc.

https://www.isa-afp.org/

Mathematics: The Language of Science.

Analysis Algebra Geometry Probability Theory



Informatics

Language

Algorithms

Data Structures

Architecture

Softw g

erification

Physics

Acoustics

Astrophysics

Ele magnetism

ular Physics

Quantum Physics

etc.

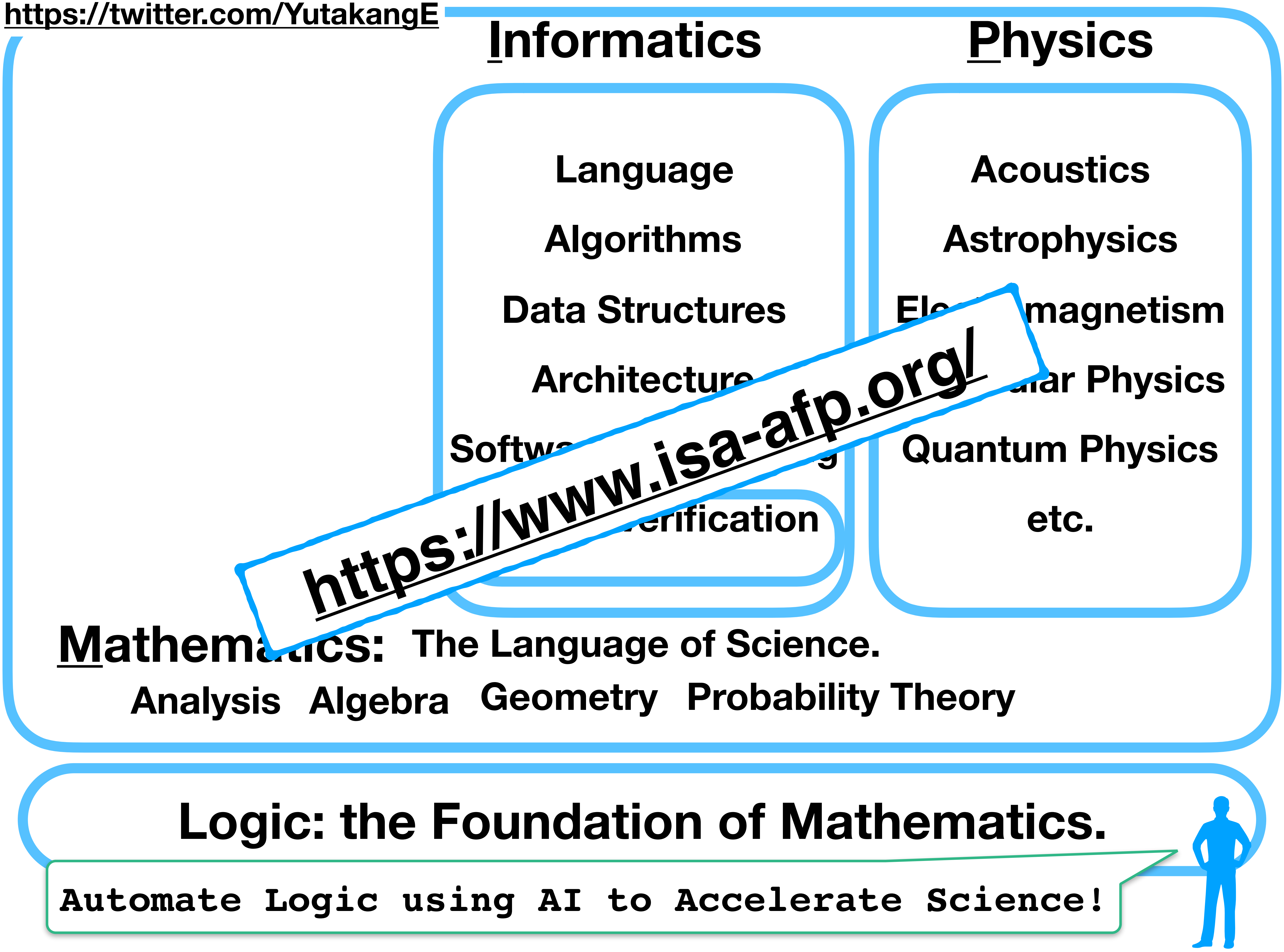
https://www.isa-afp.org/

Mathema tics: The Language of Science.

Analysis Algebra Geometry Probability Theory

Logic: the Foundation of Mathematics.

https://twitter.com/YutakangE



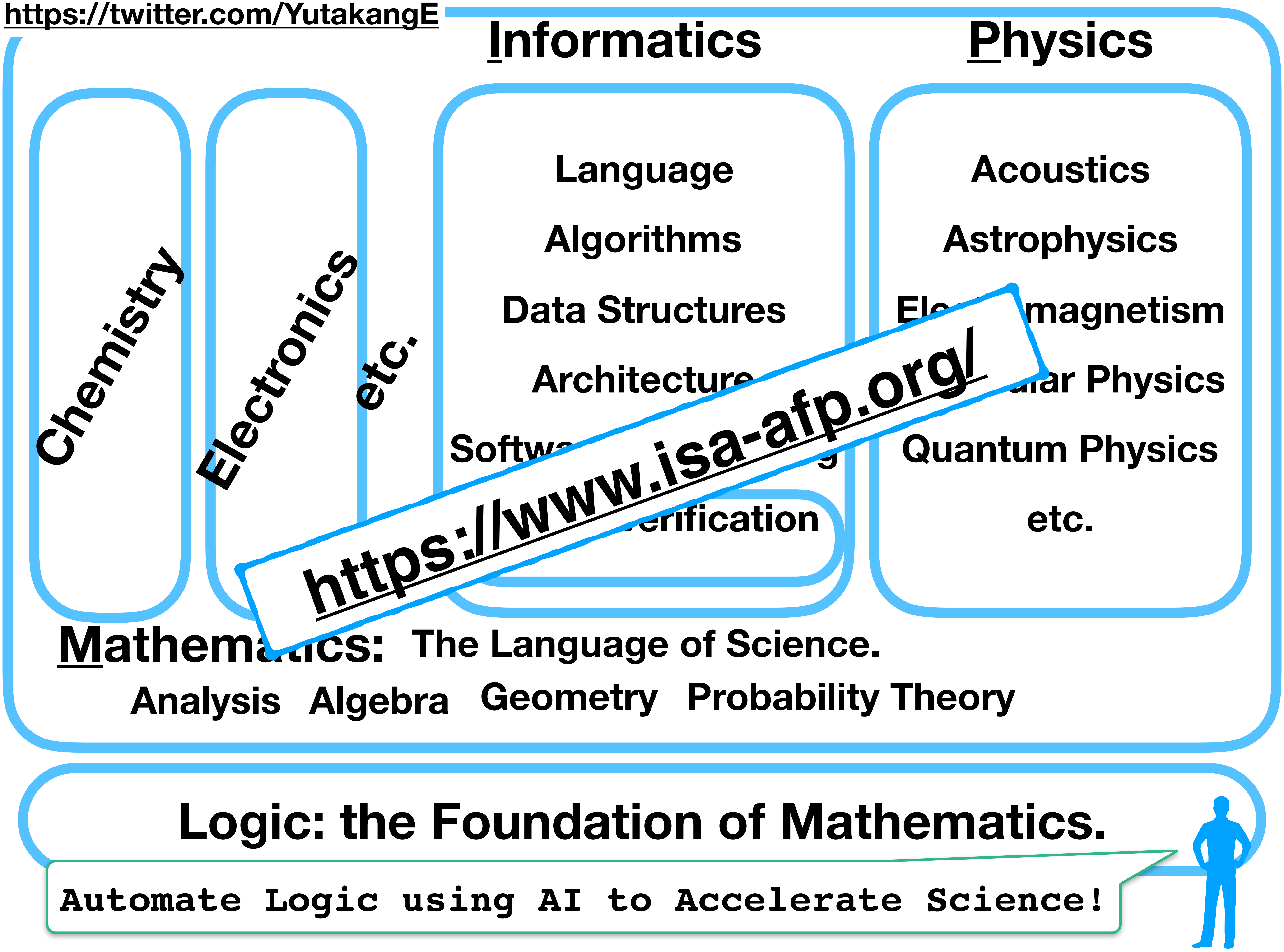
Mathematics: The Language of Science.

Analysis Algebra Geometry Probability Theory

Logic: the Foundation of Mathematics.

`Automate Logic using AI to Accelerate Science!`

https://twitter.com/YutakangE



Informatics

Physics

Chemistry

Electronics

etc.

Language

Algorithms

Data Structures

Architecture

Softw g

erification

Acoustics

Astrophysics

Ele magnetism

ular Physics

Quantum Physics

etc.

https://www.isa-afp.org/

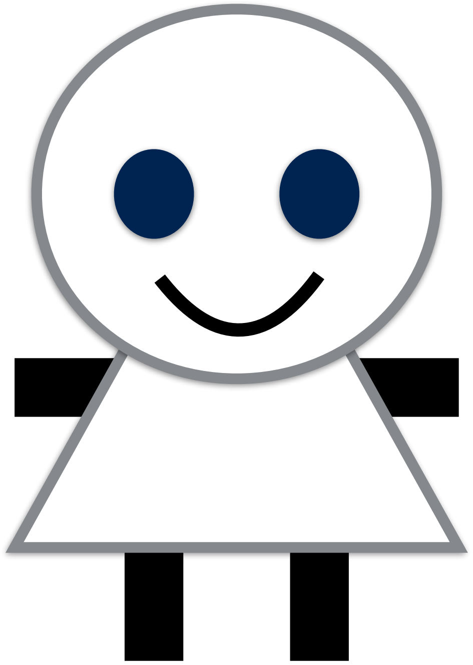
Mathematics: The Language of Science.

Analysis Algebra Geometry Probability Theory

Logic: the Foundation of Mathematics.

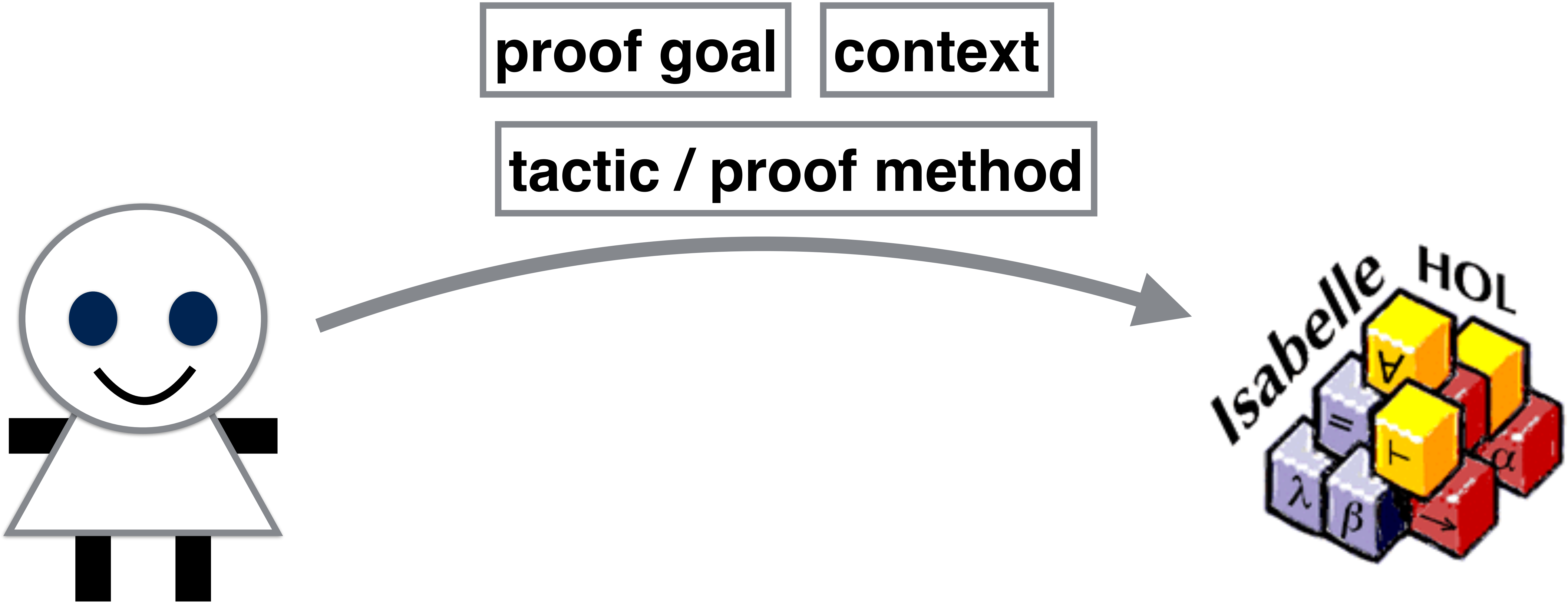
Automate Logic using AI to Accelerate Science!

Interactive theorem proving with Isabelle/HOL



https://twitter.com/YutakangE

Interactive theorem proving with Isabelle/HOL



proof goal

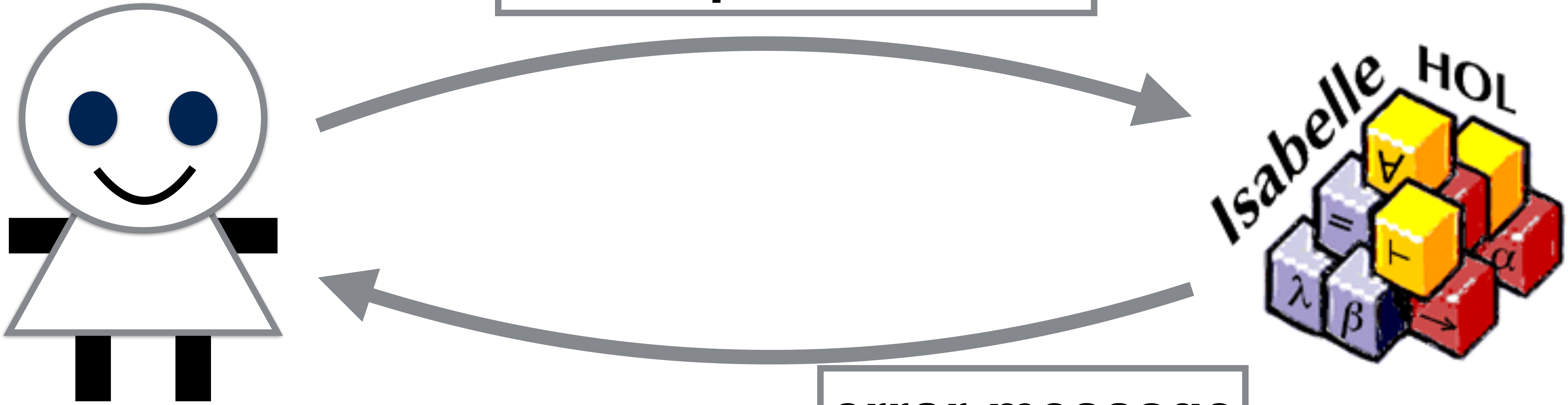
context

tactic / proof method

Interactive theorem proving with Isabelle/HOL

proof goal **context**

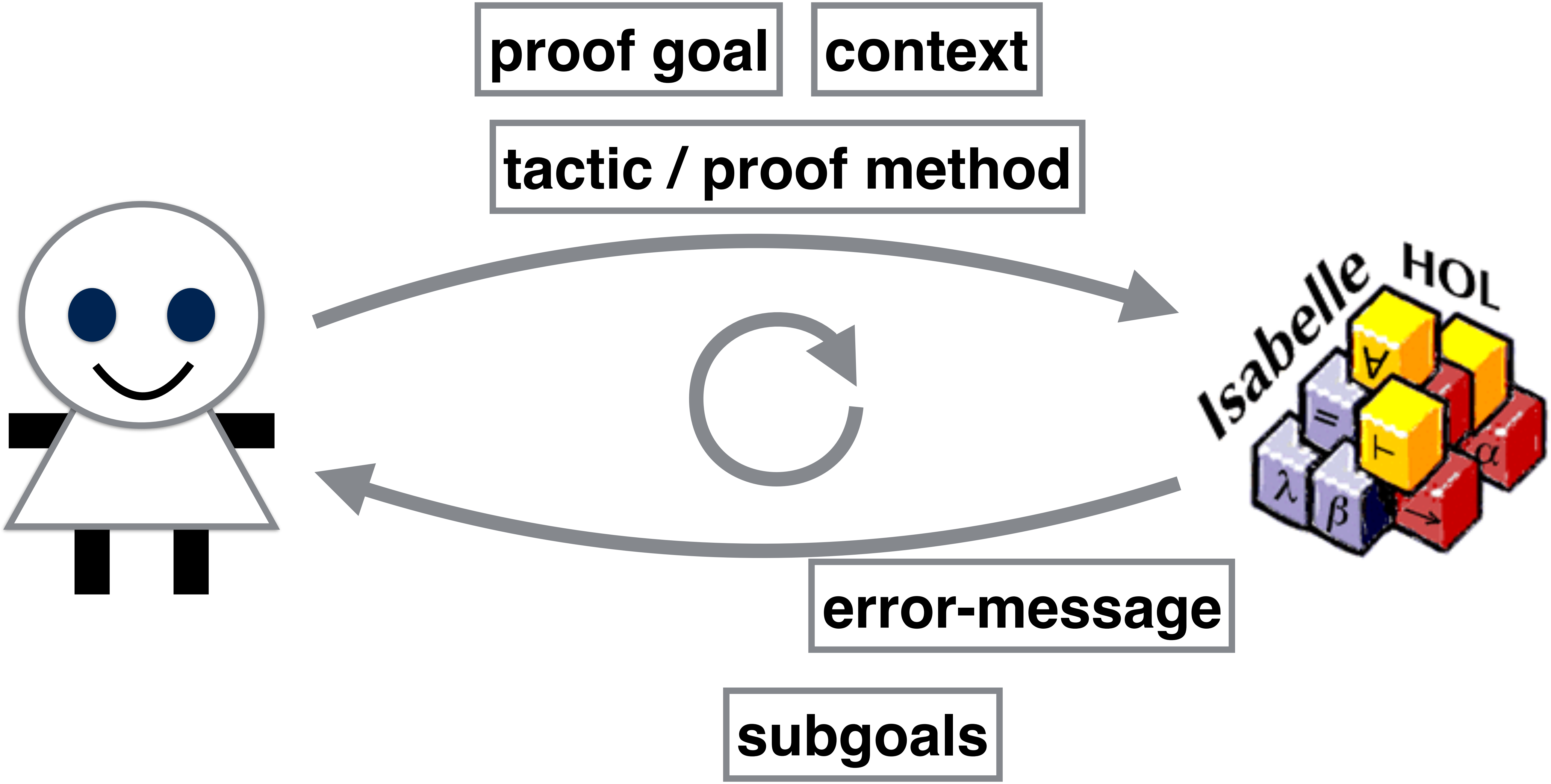
tactic / proof method



error-message

subgoals

Interactive theorem proving with Isabelle/HOL



proof goal

context

tactic / proof method

error-message

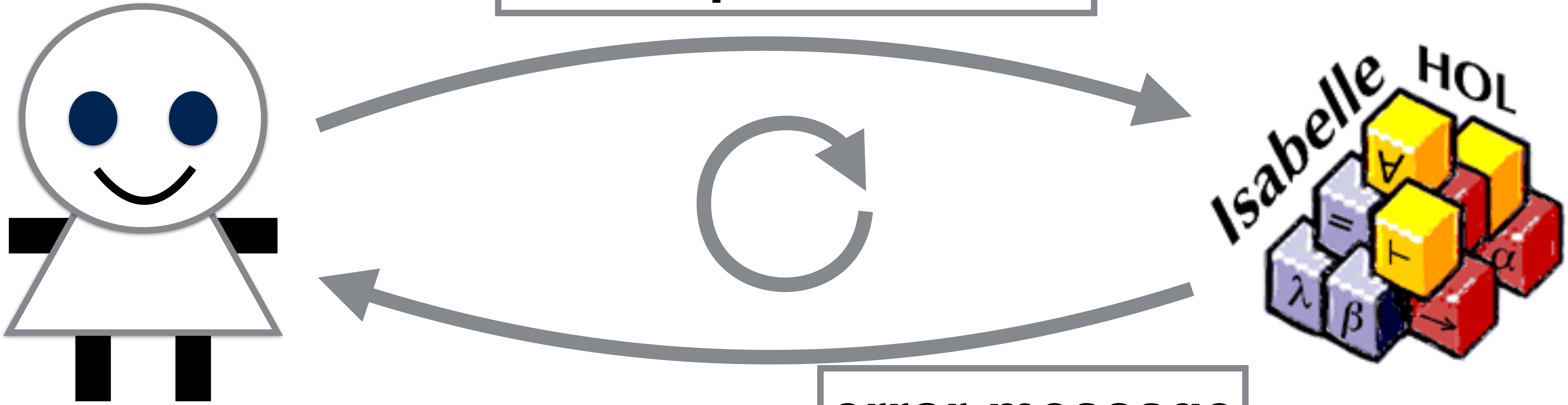
subgoals

Interactive theorem proving with Isabelle/HOL

proof goal

context

tactic / proof method

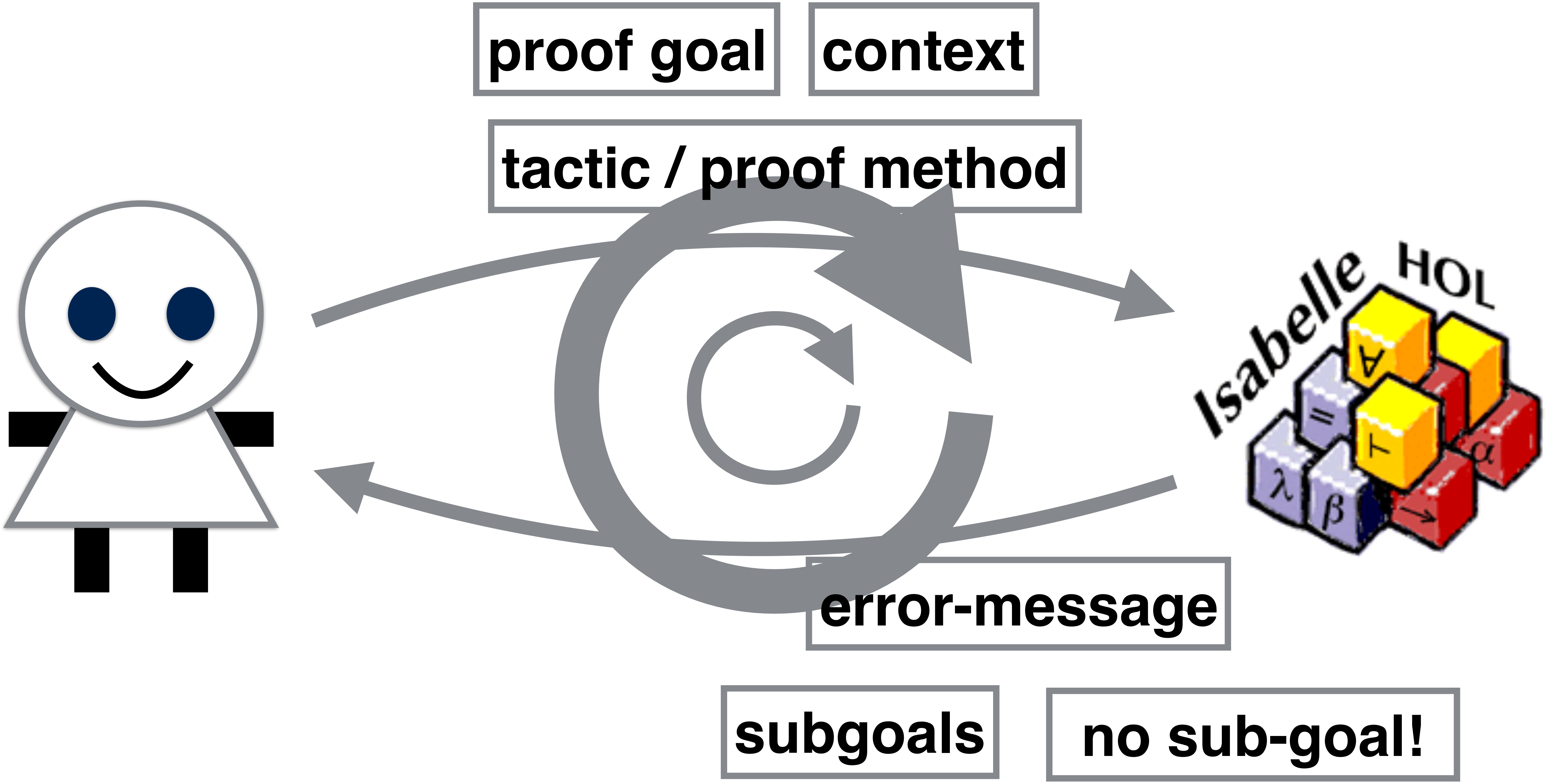


error-message

subgoals

no sub-goal!

Interactive theorem proving with Isabelle/HOL



proof goal

context

tactic / proof method

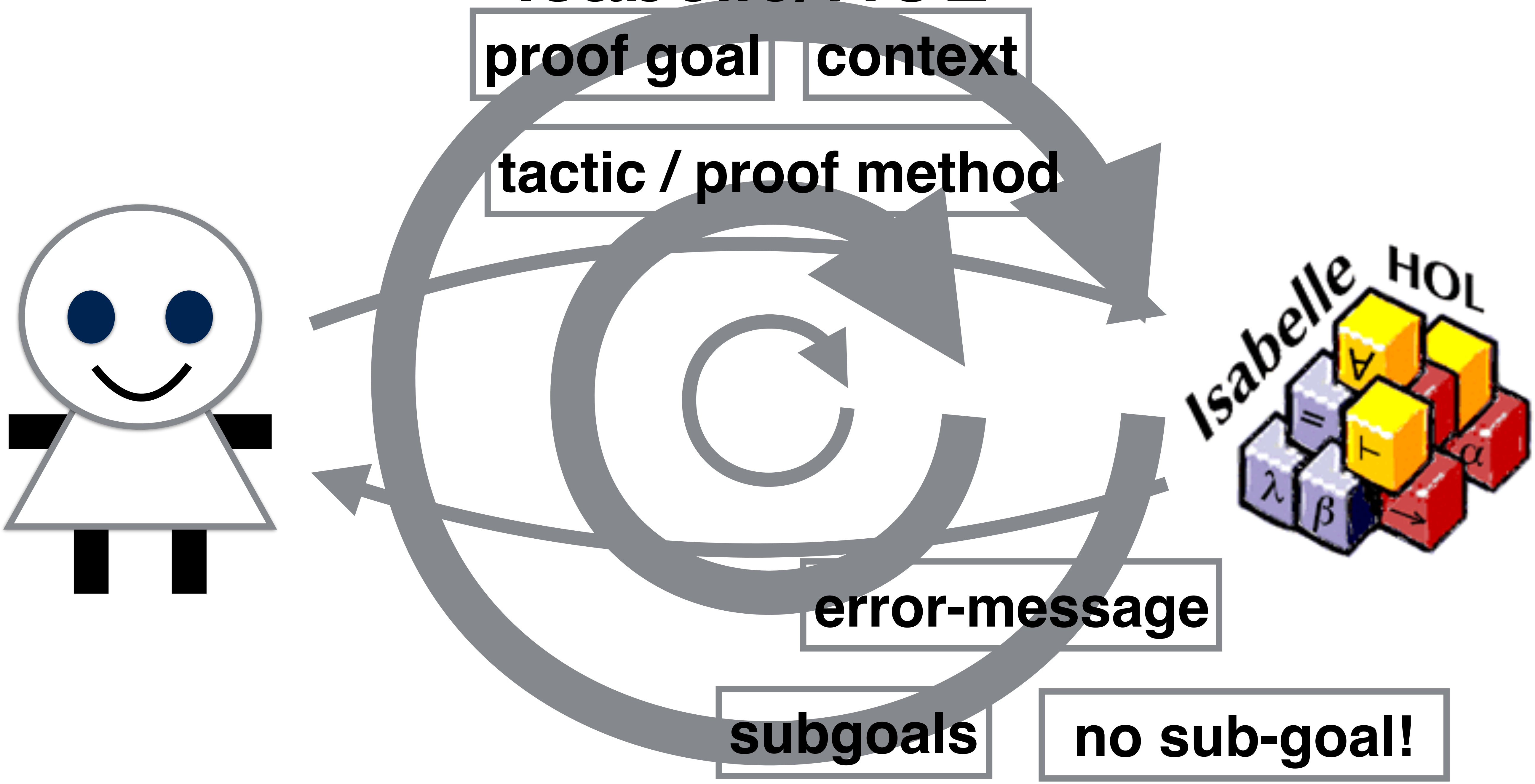
error-message

subgoals

no sub-goal!

https://twitter.com/YutakangE

Interactive theorem proving with Isabelle/HOL



proof goal

context

tactic / proof method

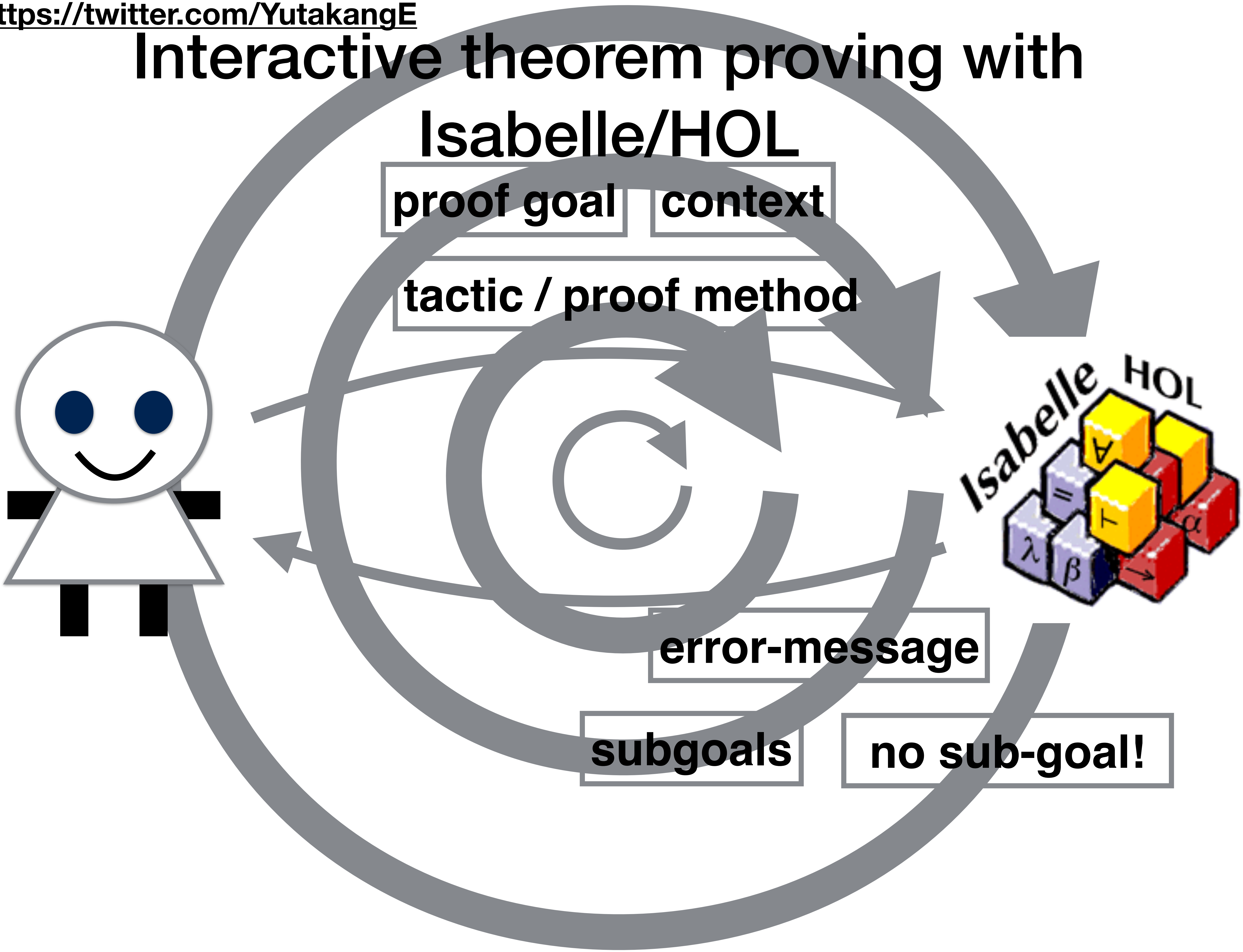
error-message

subgoals

no sub-goal!

https://twitter.com/YutakangE

Interactive theorem proving with Isabelle/HOL



proof goal

context

tactic / proof method

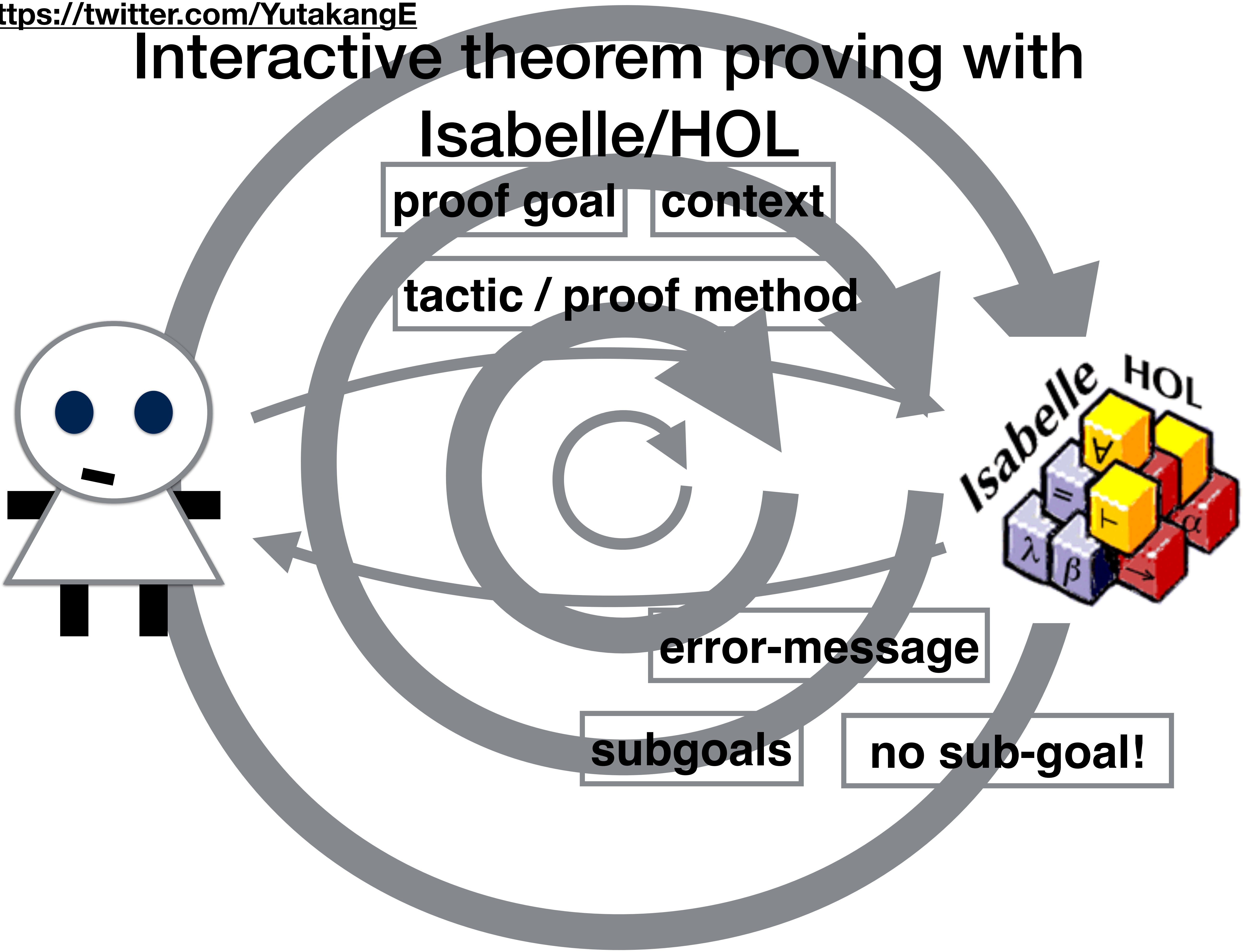
error-message

subgoals

no sub-goal!

https://twitter.com/YutakangE

Interactive theorem proving with Isabelle/HOL



proof goal

context

tactic / proof method

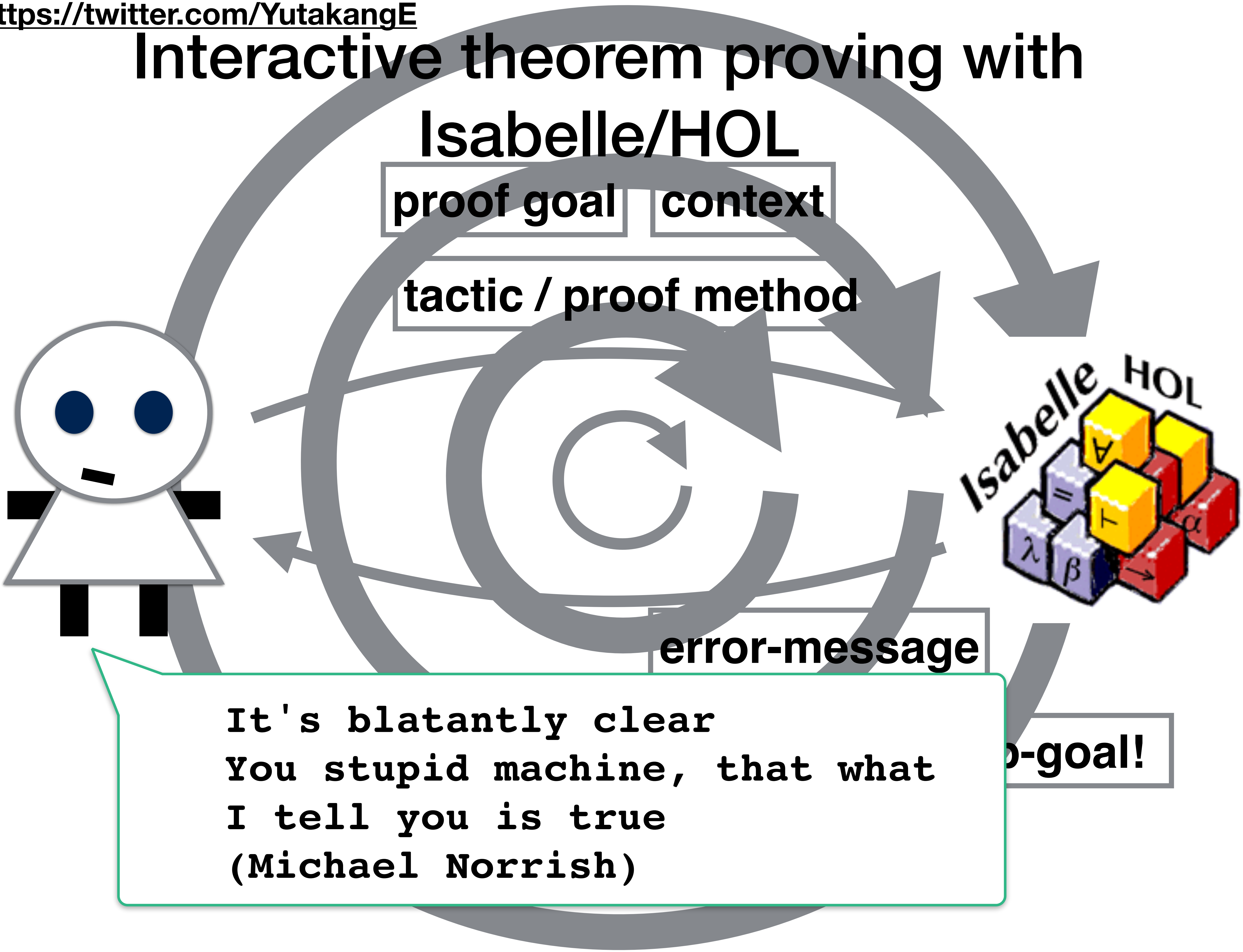
error-message

subgoals

no sub-goal!

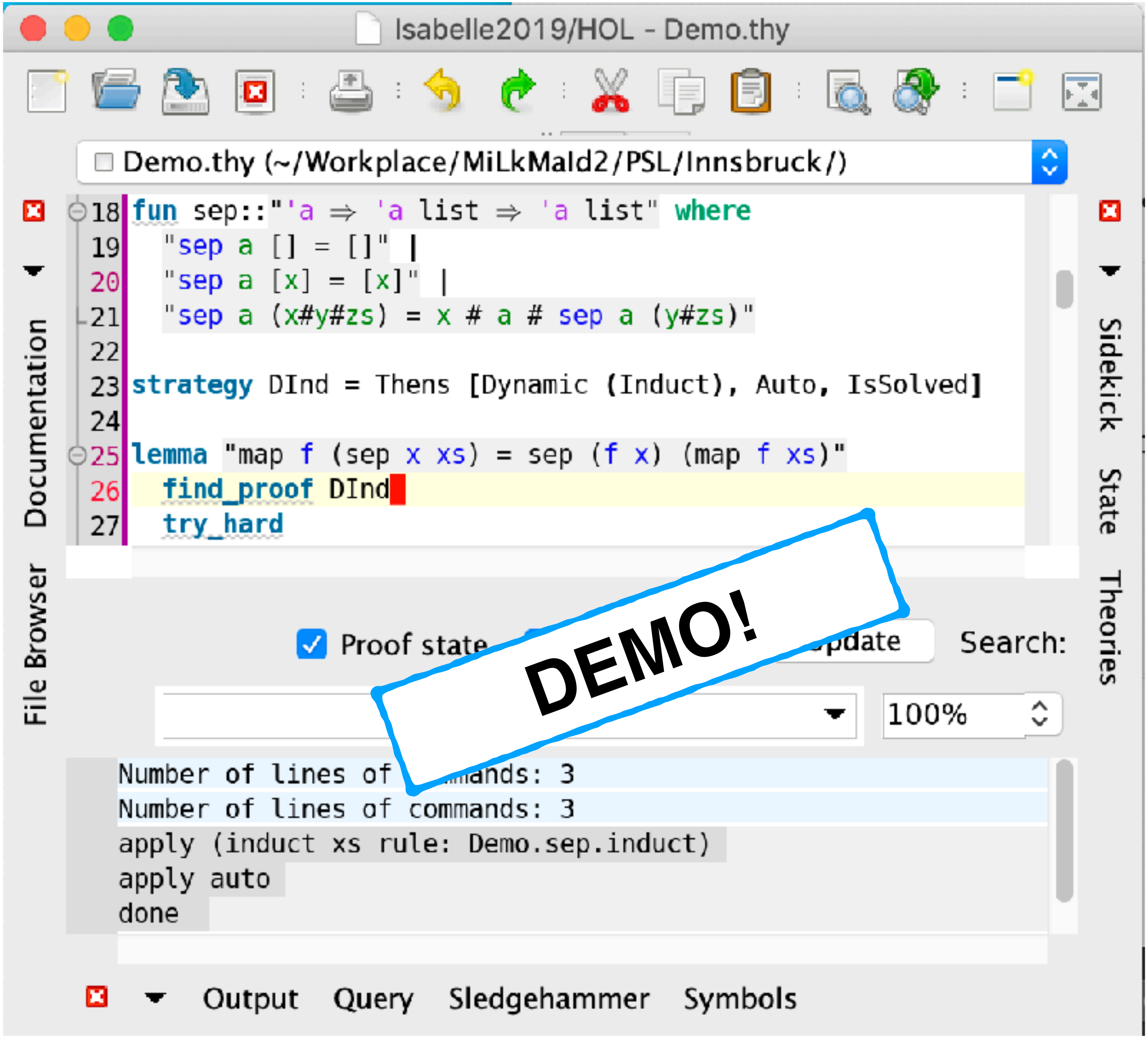
https://twitter.com/YutakangE

Interactive theorem proving with Isabelle/HOL



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

https://twitter.com/YutakangE

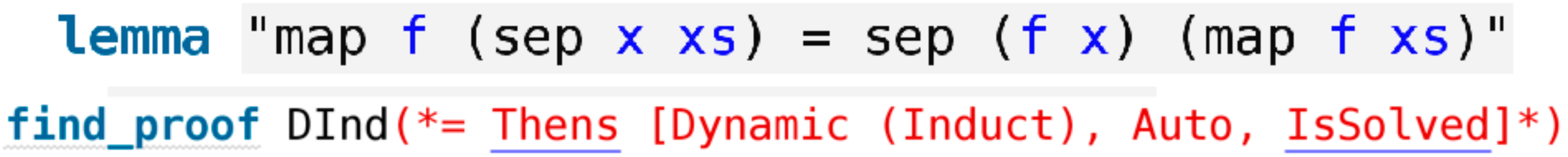


```
fun sep::"'a ⇒ 'a list ⇒ 'a list" where
  "sep a [] = []" |
  "sep a [x] = [x]" |
  "sep a (x#y#zs) = x # a # sep a (y#zs)"

strategy DInd = Thens [Dynamic (Induct), Auto, IsSolved]

lemma "map f (sep x xs) = sep (f x) (map f xs)"
  find_proof DInd
  try_hard
```

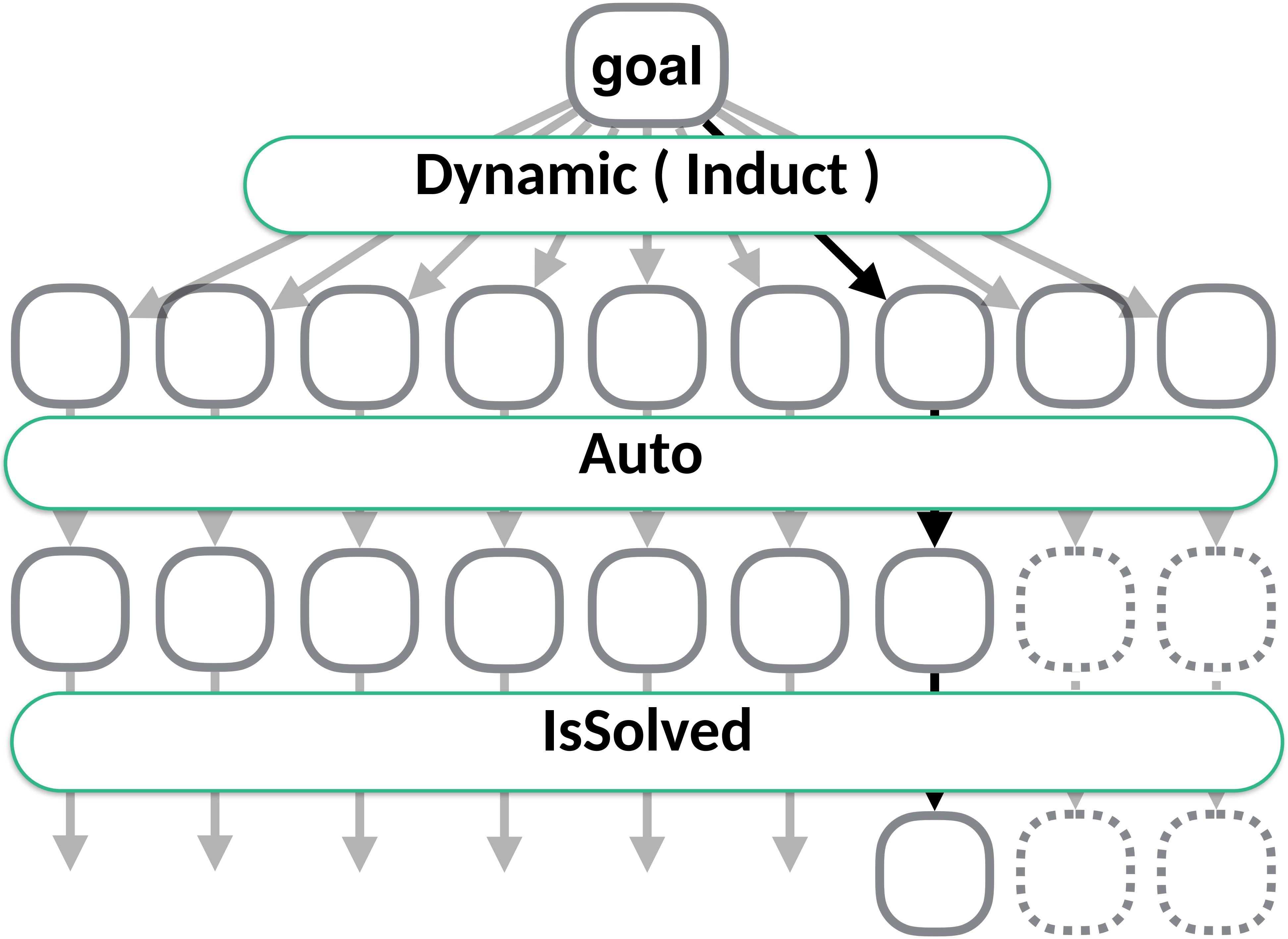
```
Number of lines of commands: 3
Number of lines of commands: 3
apply (induct xs rule: Demo.sep.induct)
apply auto
done
```

DEMO!

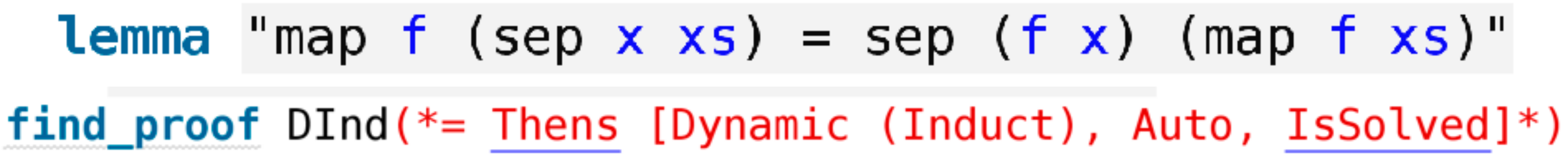
https://twitter.com/YutakangE



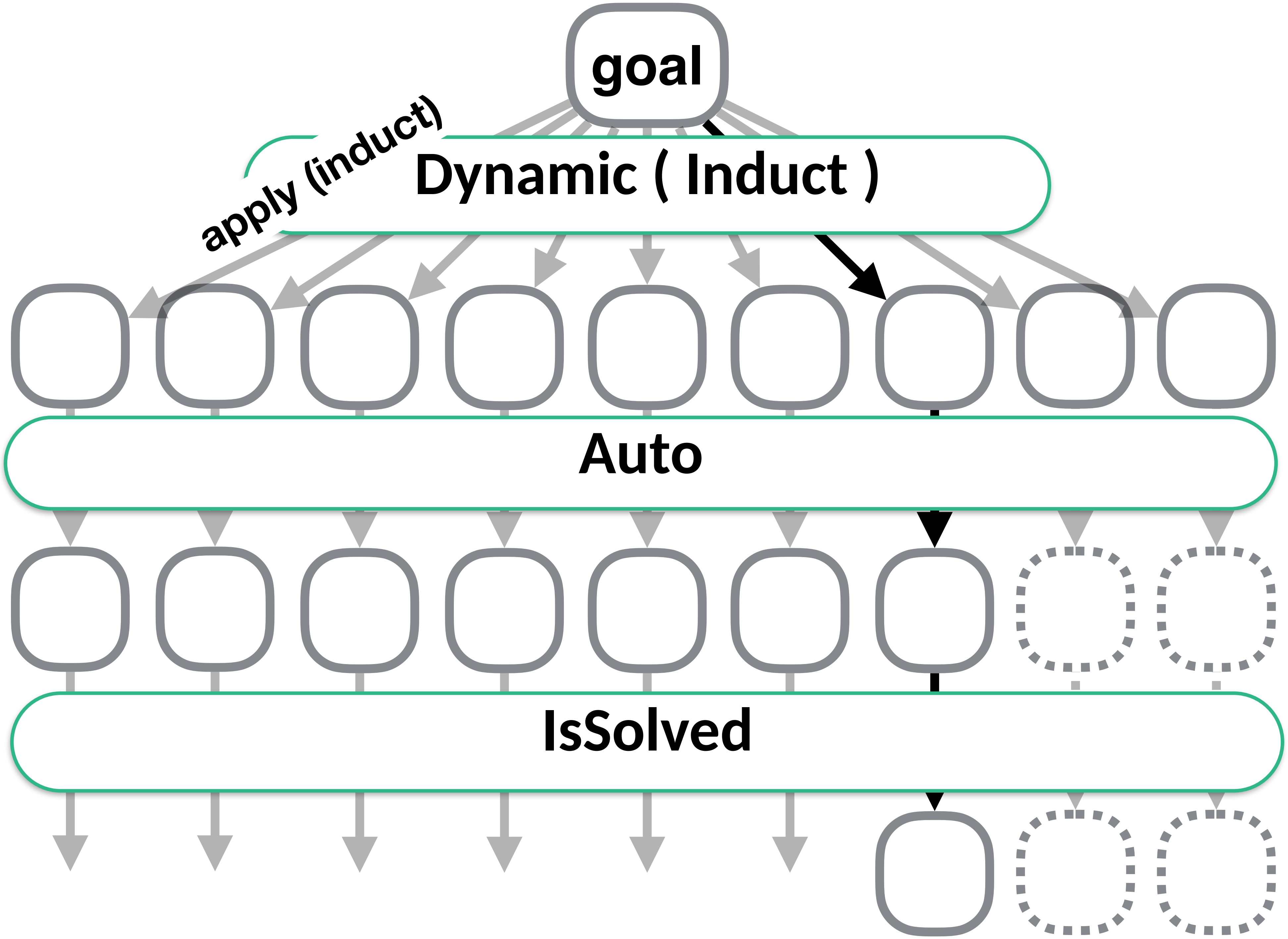
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



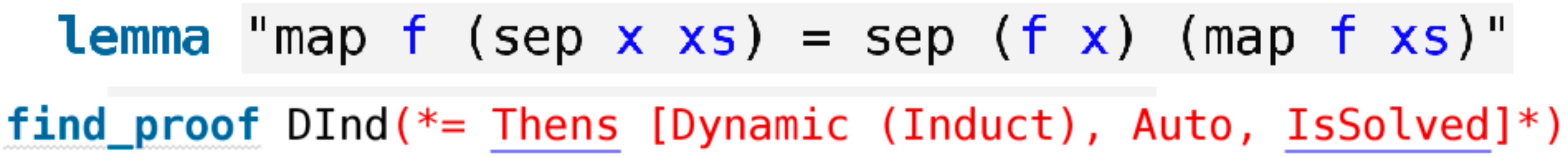
https://twitter.com/YutakangE



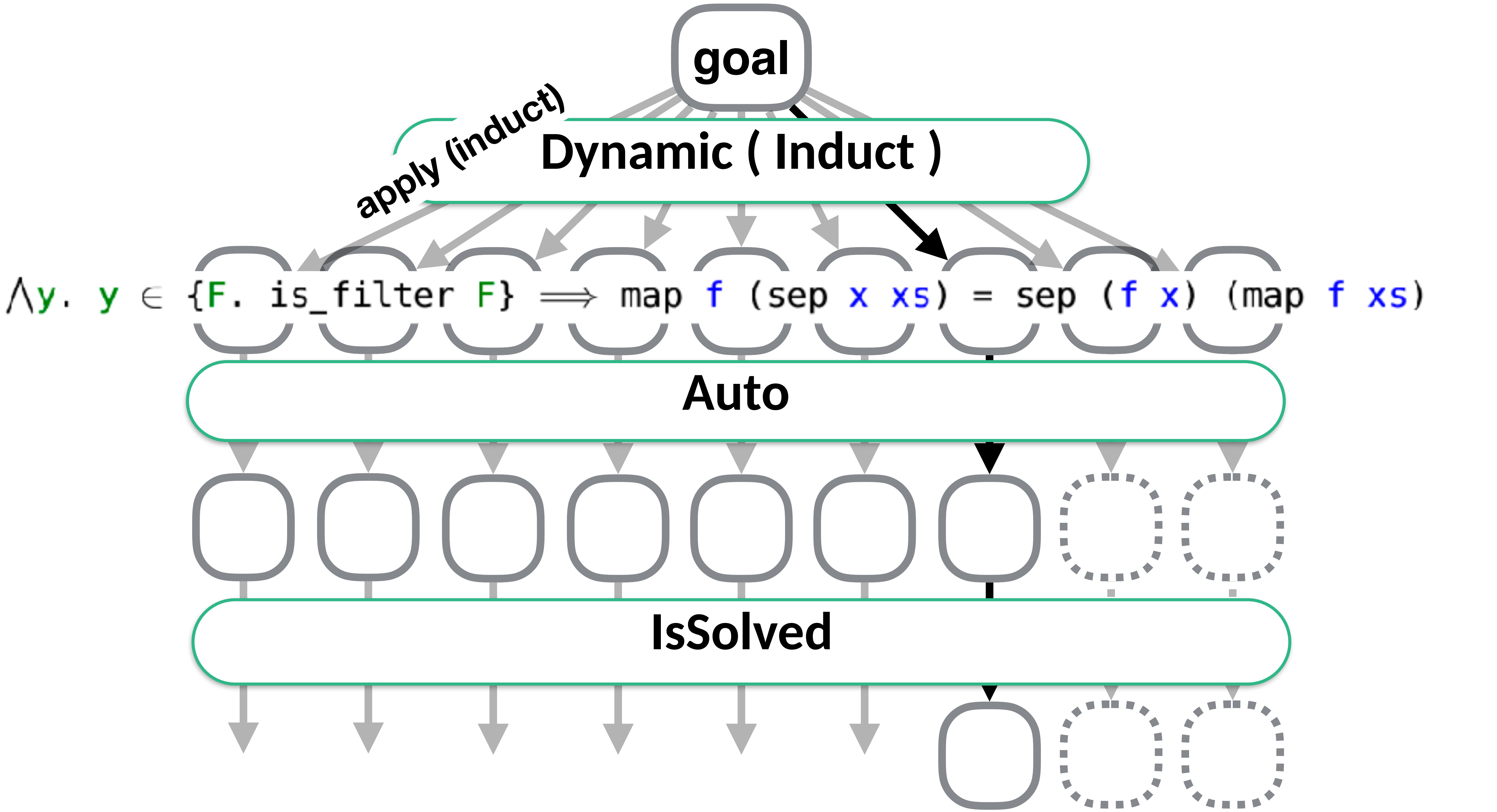
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



https://twitter.com/YutakangE

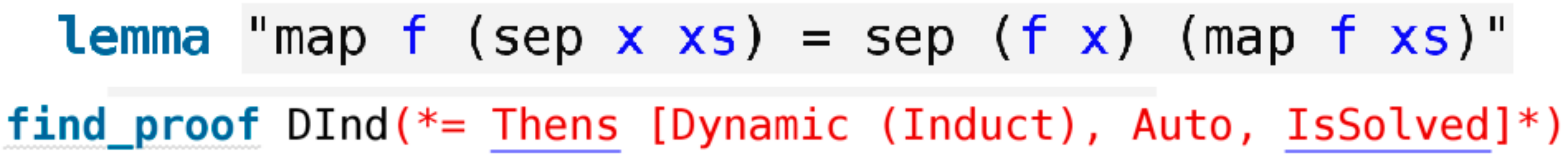


```
lemma "map f (sep x xs) = sep (f x) (map f xs)"

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

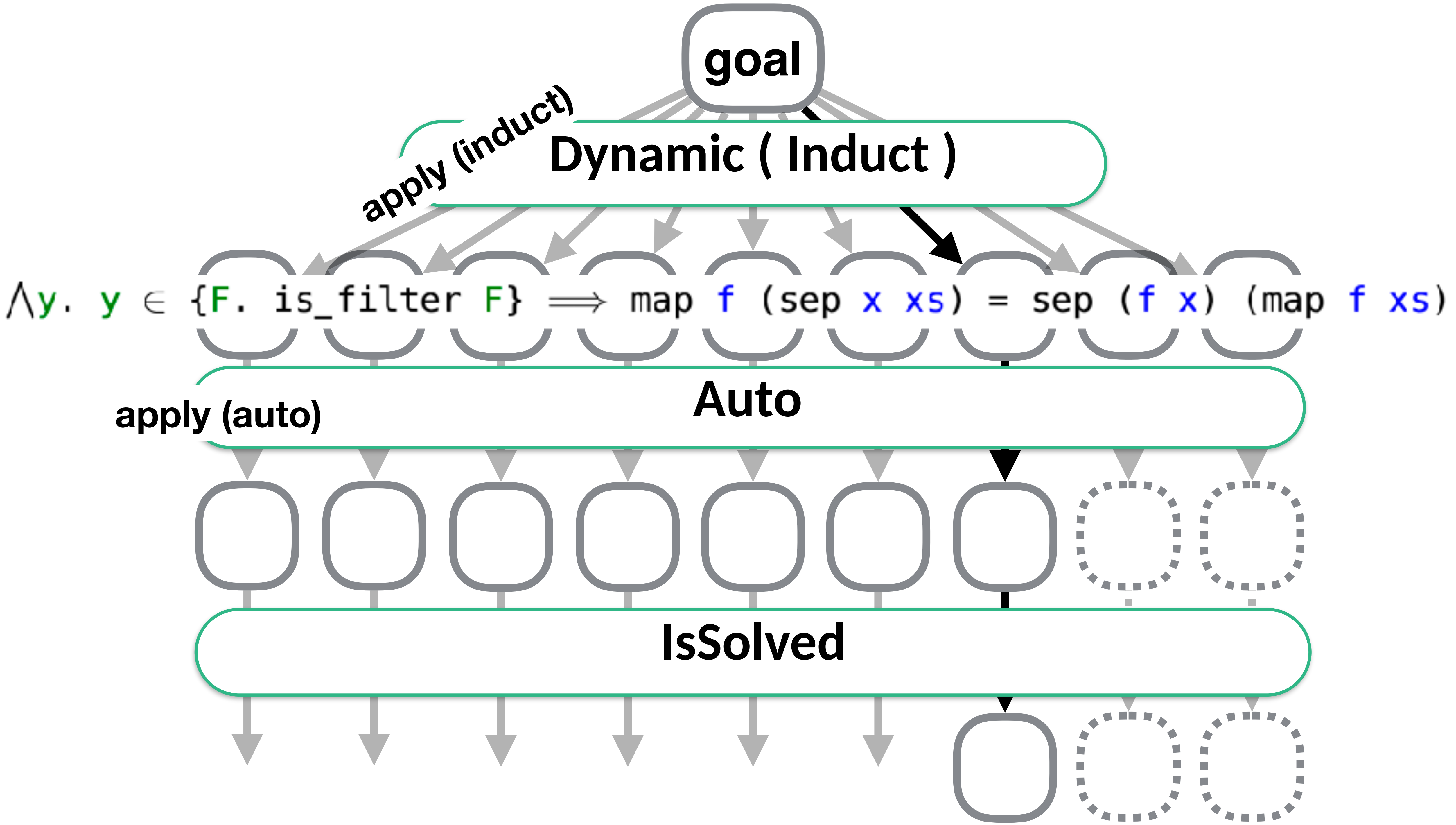


```
⋀y. y ∈ {F. is_filter F} ⟹ map f (sep x xs) = sep (f x) (map f xs)
```

https://twitter.com/YutakangE



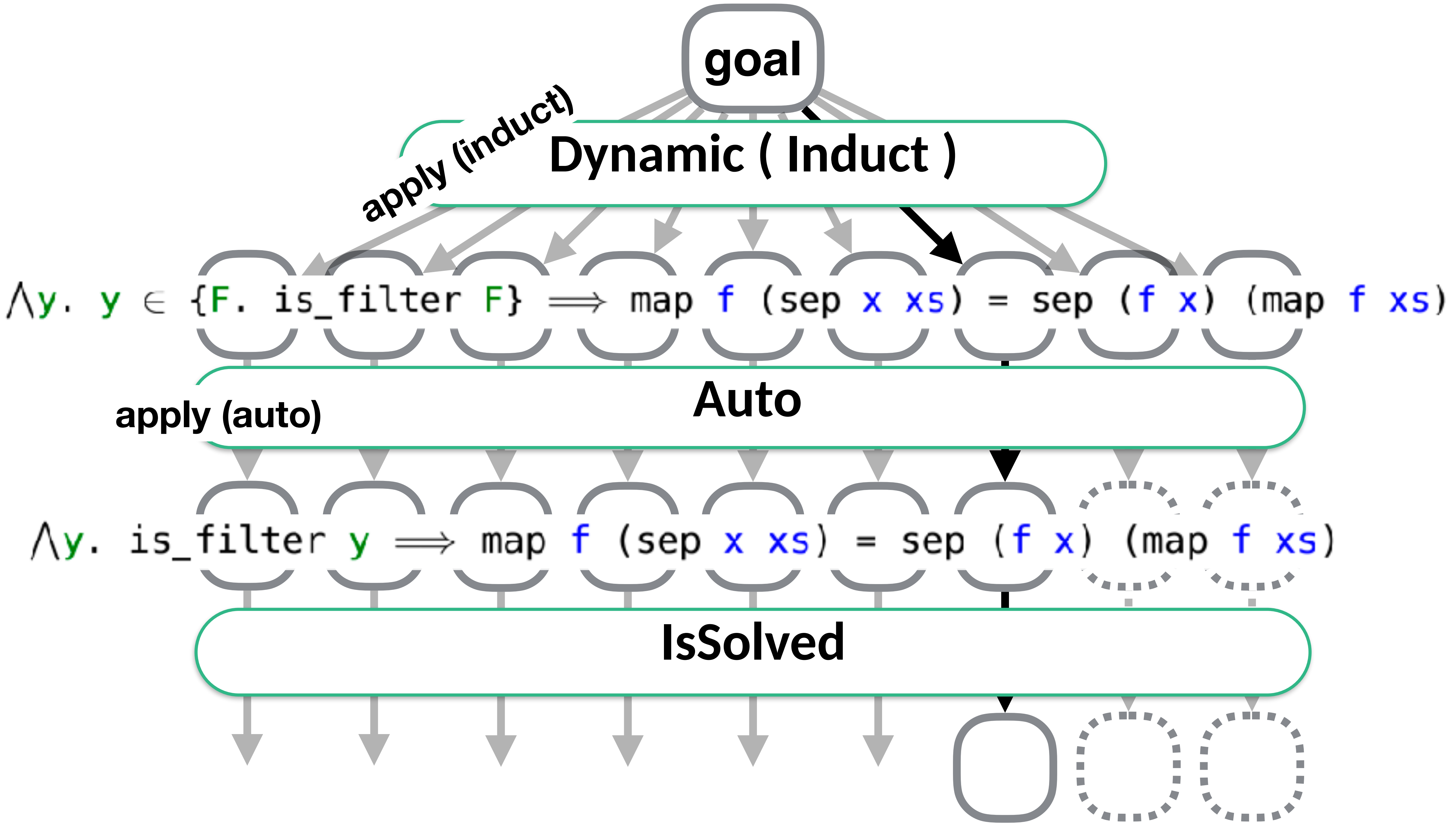
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"

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



⋀y. y ∈ {F. is_filter F} ⟹ map f (sep x xs) = sep (f x) (map f xs)

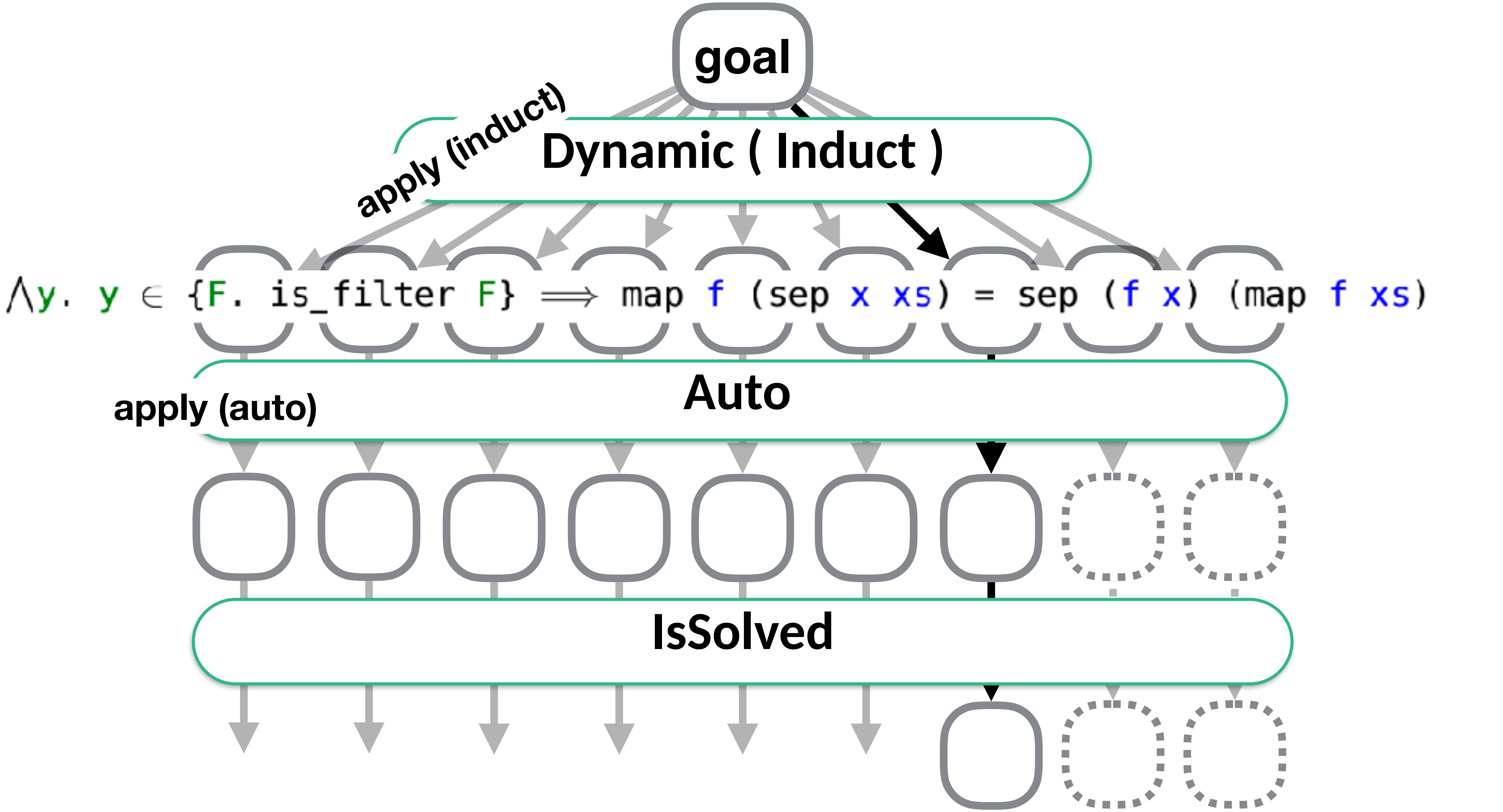
https://twitter.com/YutakangE

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

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


goal
apply (induct)
Dynamic (Induct)
⋀y. y ∈ {F. is_filter F} ⟹ map f (sep x xs) = sep (f x) (map f xs)
apply (auto)
Auto
⋀y. is_filter y ⟹ map f (sep x xs) = sep (f x) (map f xs)
IsSolved

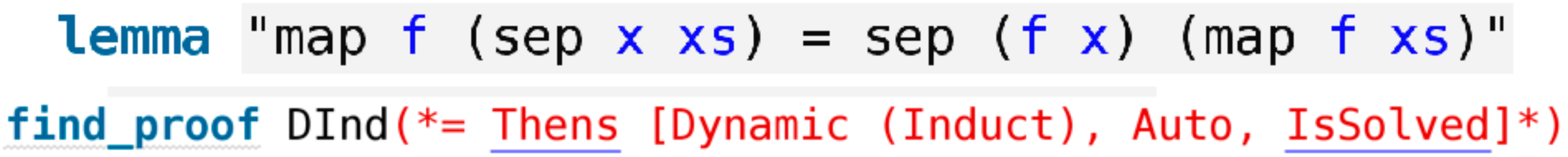
https://twitter.com/YutakangE

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

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

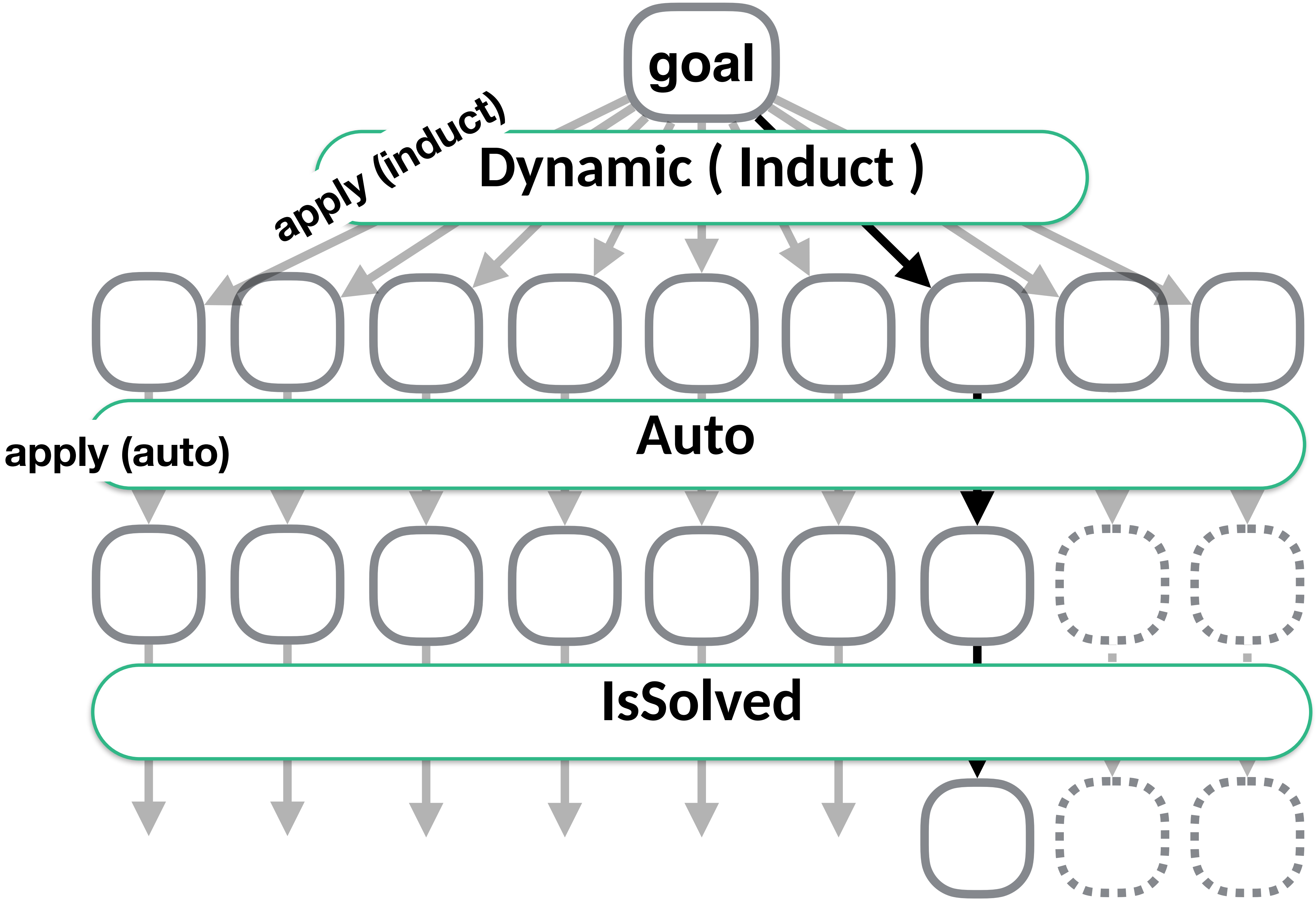


⋀y. y ∈ {F. is_filter F} ⟹ map f (sep x xs) = sep (f x) (map f xs)

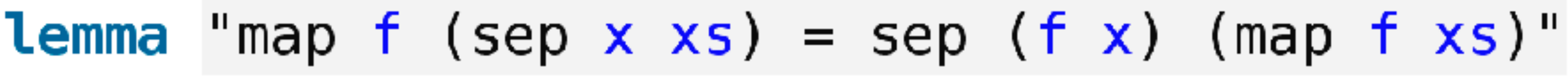
https://twitter.com/YutakangE



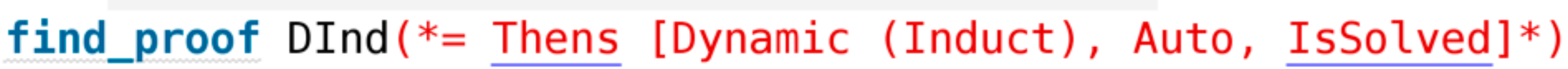
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



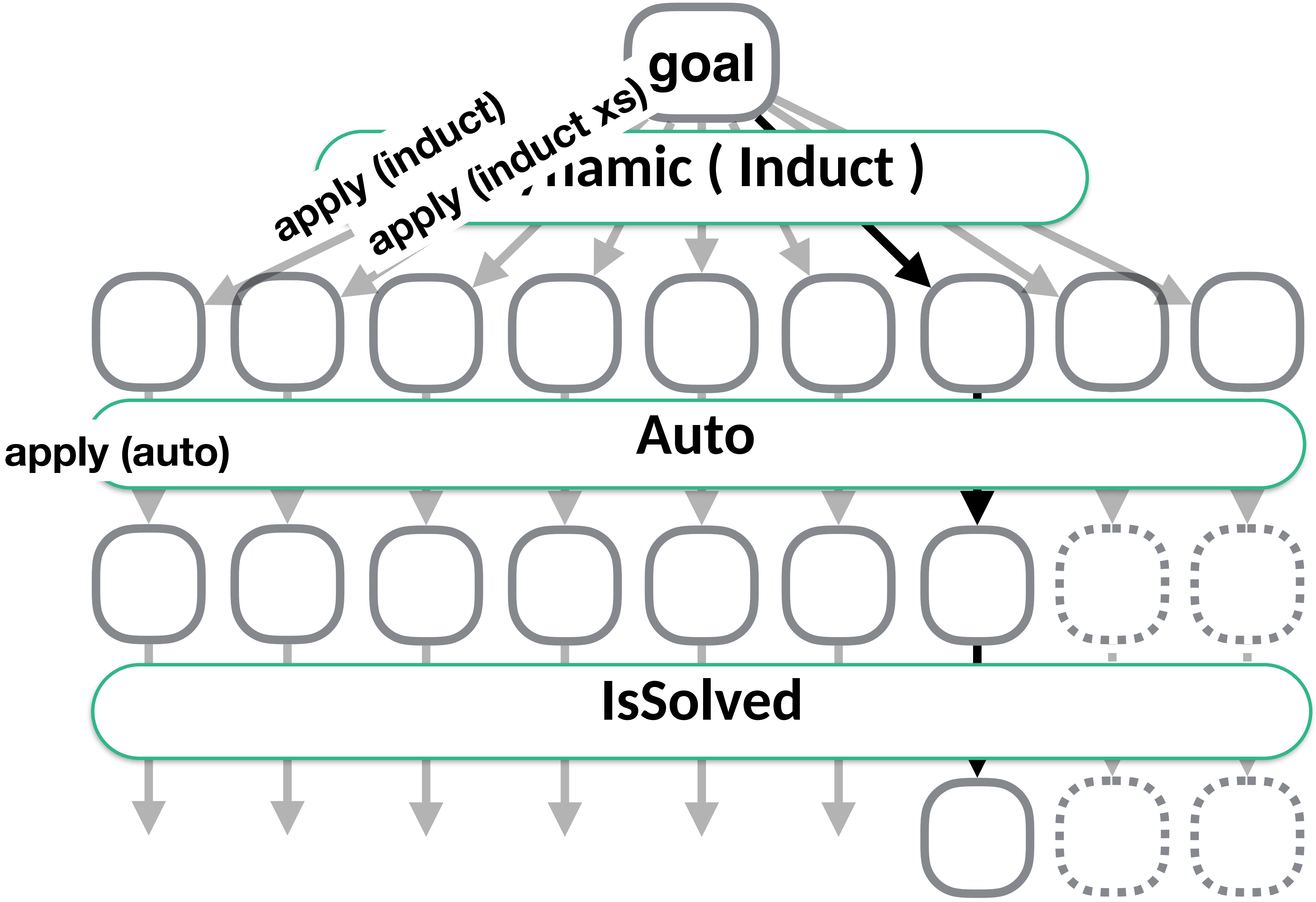
https://twitter.com/YutakangE



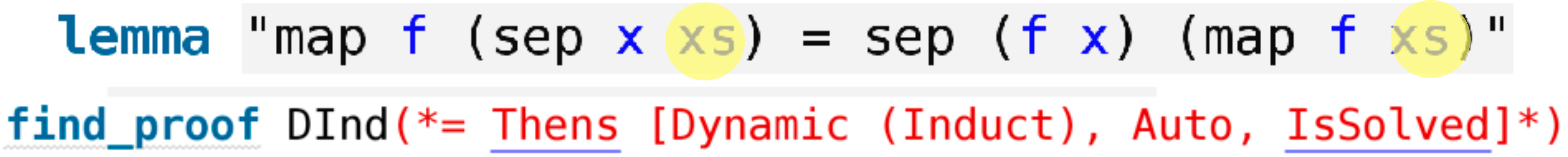
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```



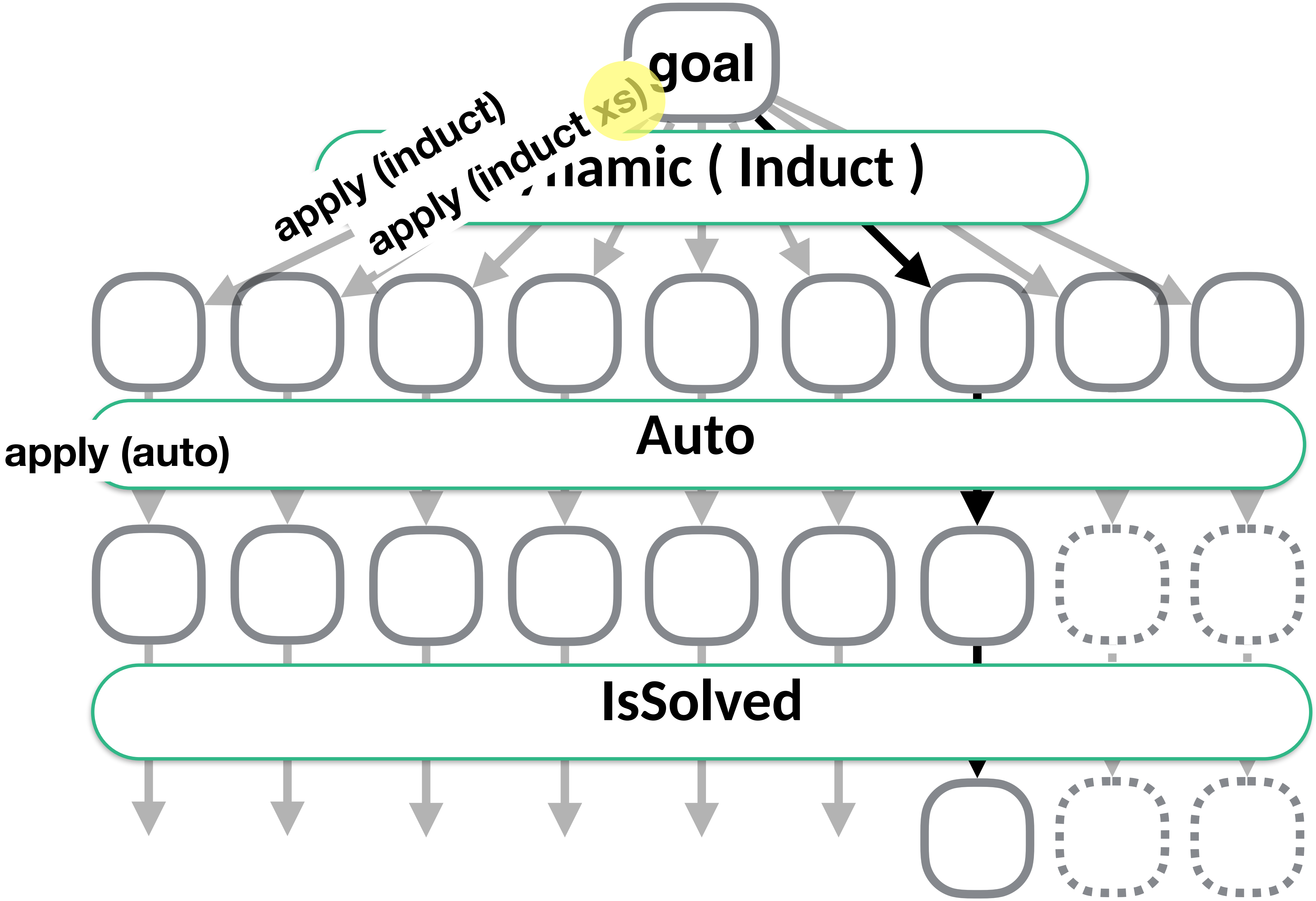
```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



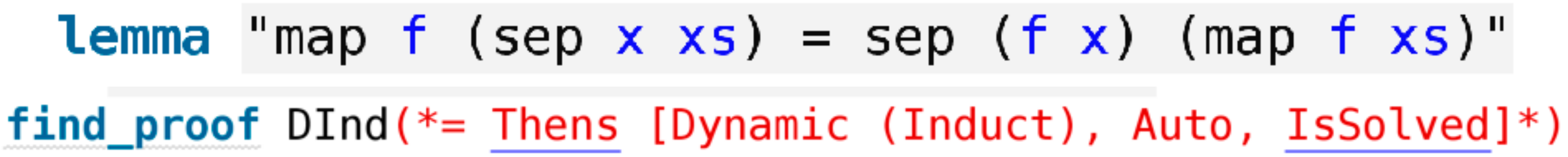
https://twitter.com/YutakangE



```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```

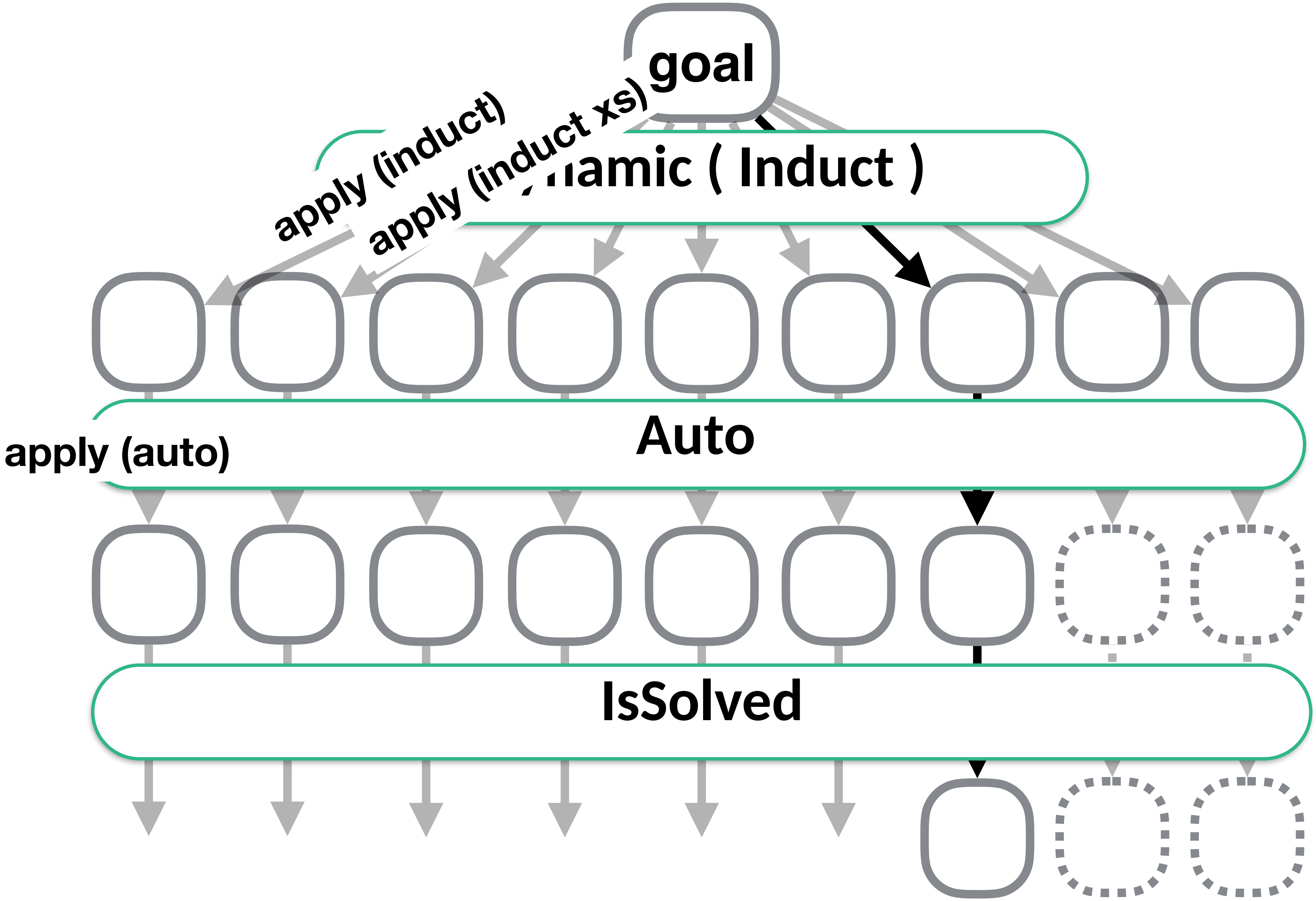


https://twitter.com/YutakangE

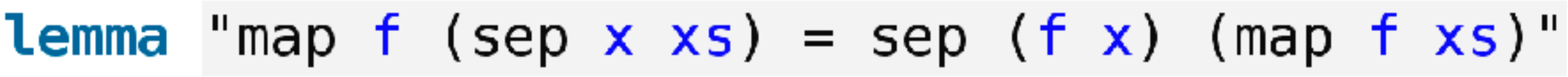


```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

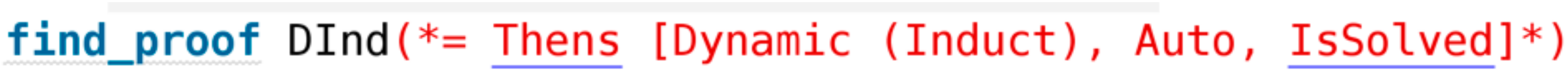
```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



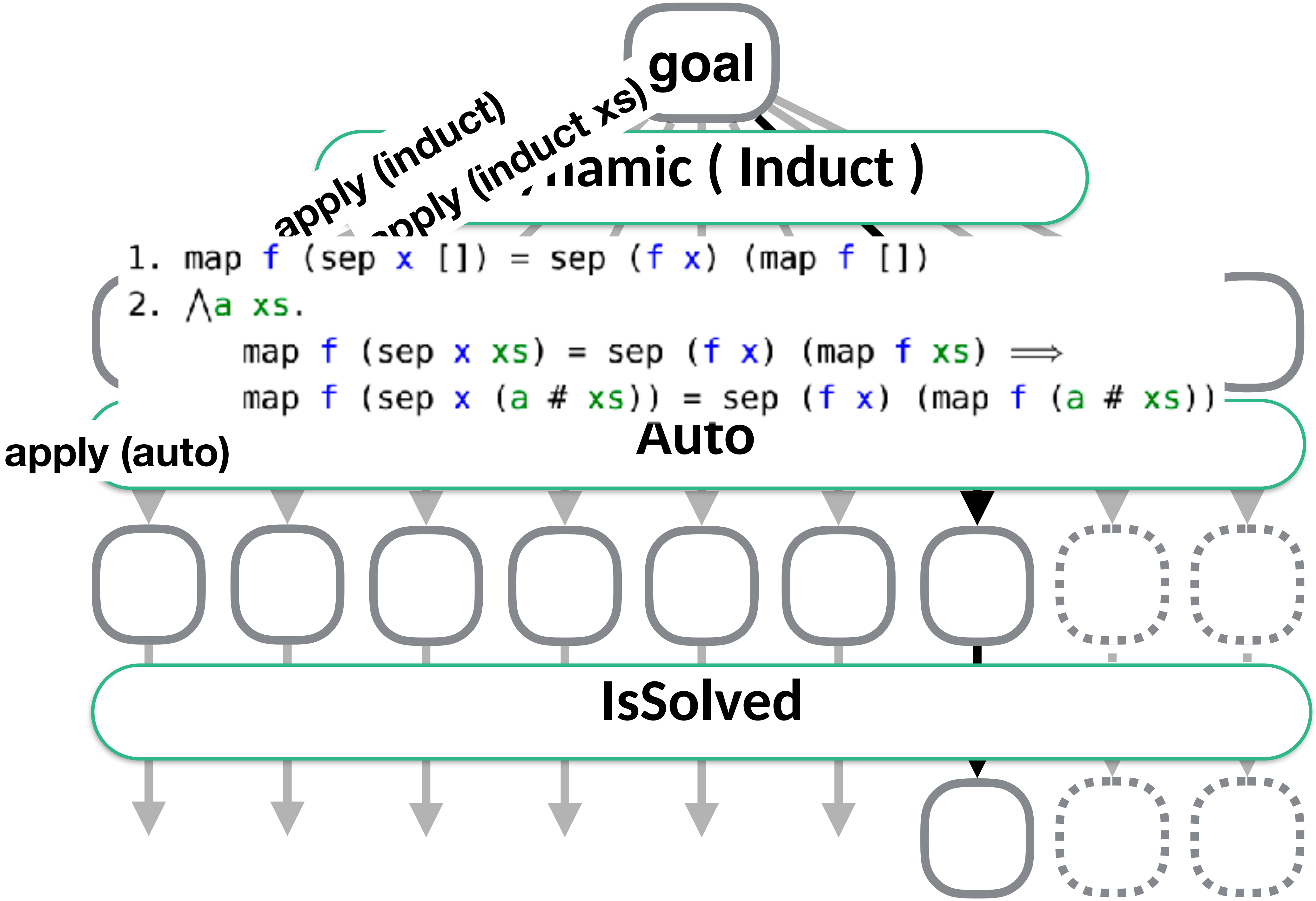
https://twitter.com/YutakangE



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

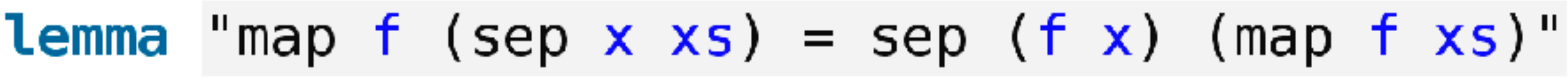


```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```

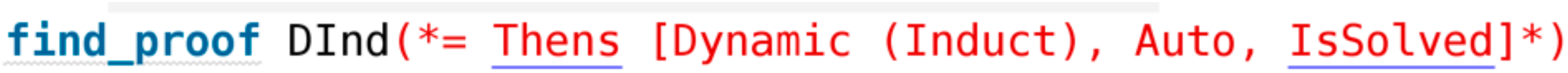


```
1. map f (sep x []) = sep (f x) (map f [])
2. ⋀a xs.
      map f (sep x xs) = sep (f x) (map f xs) ⟹
      map f (sep x (a # xs)) = sep (f x) (map f (a # xs))
```

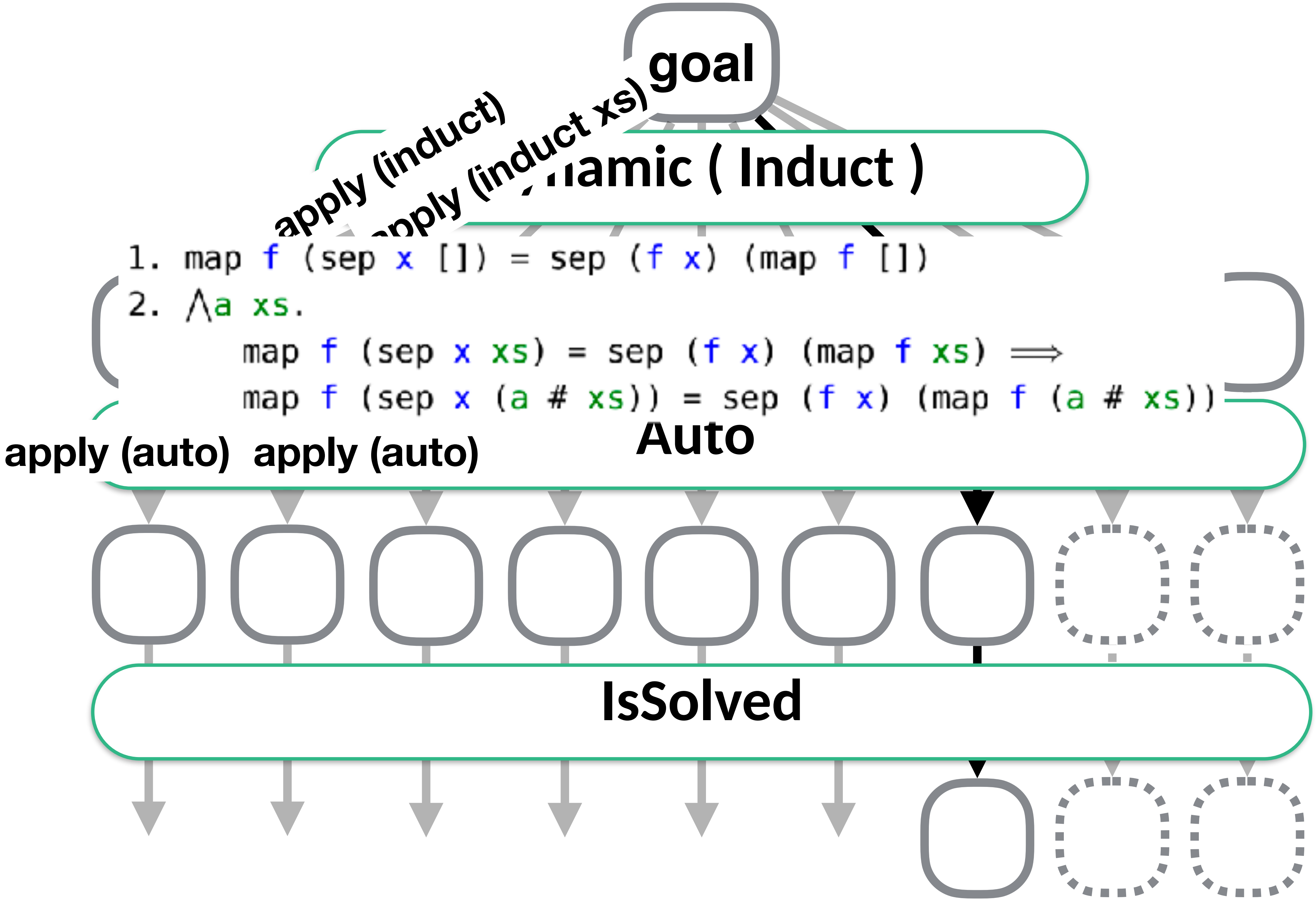
https://twitter.com/YutakangE



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

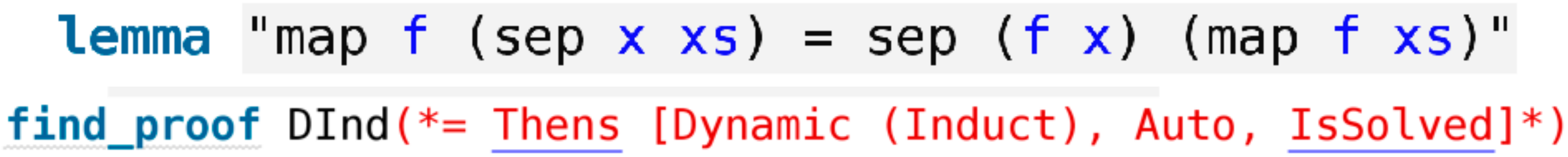


```
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```

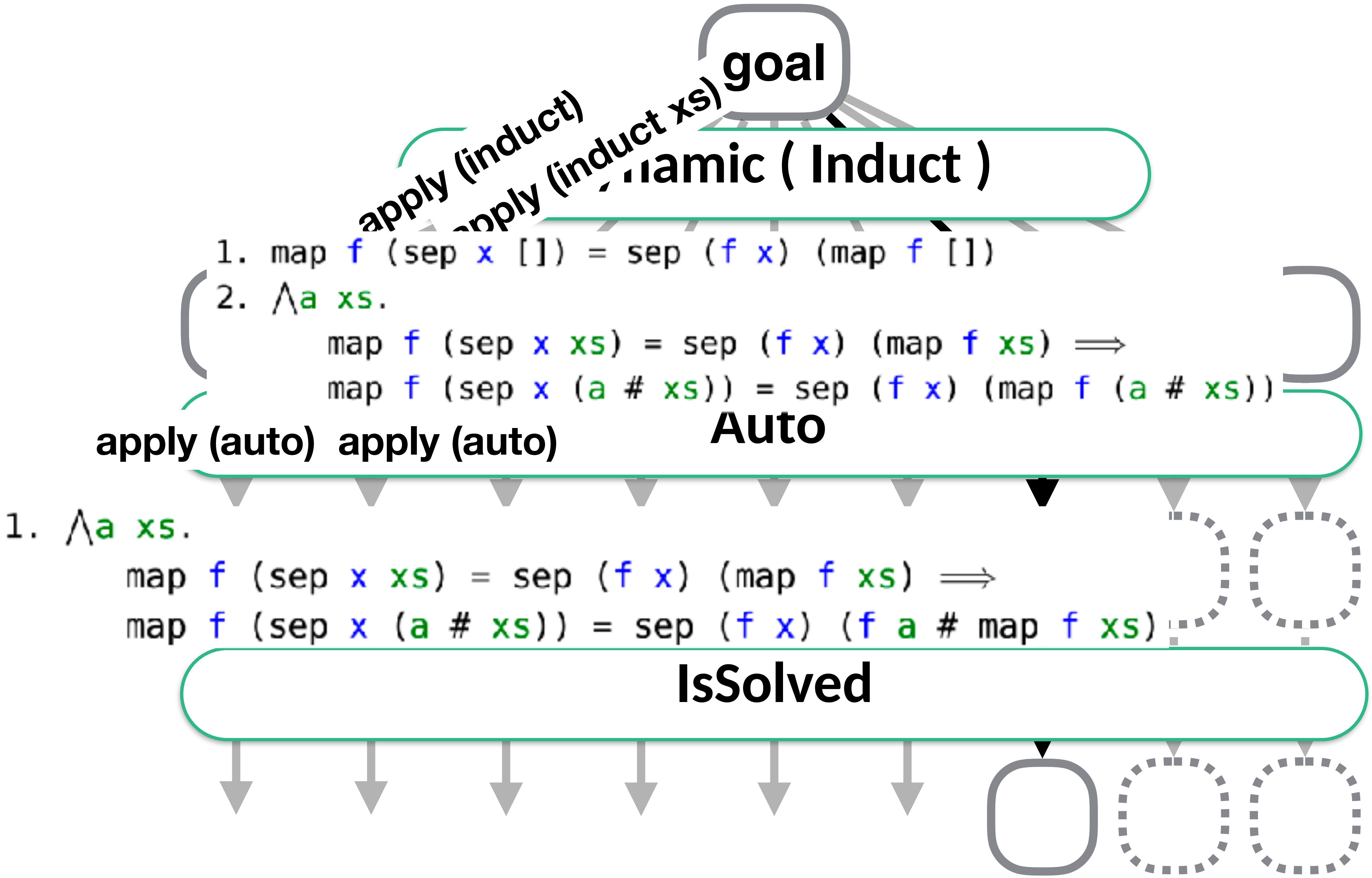


```
1. map f (sep x []) = sep (f x) (map f [])
2. ⋀a xs.
       map f (sep x xs) = sep (f x) (map f xs) ⟹
       map f (sep x (a # xs)) = sep (f x) (map f (a # xs))
```

https://twitter.com/YutakangE



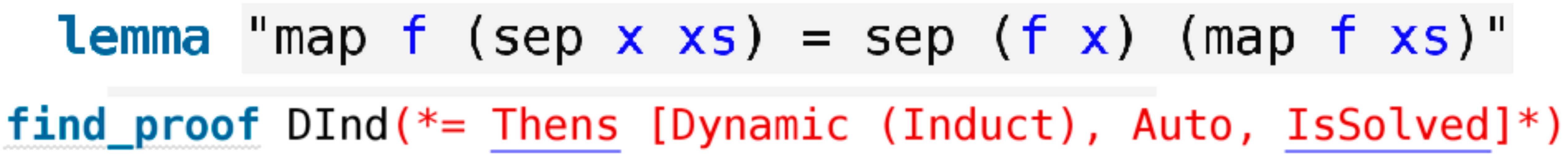
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



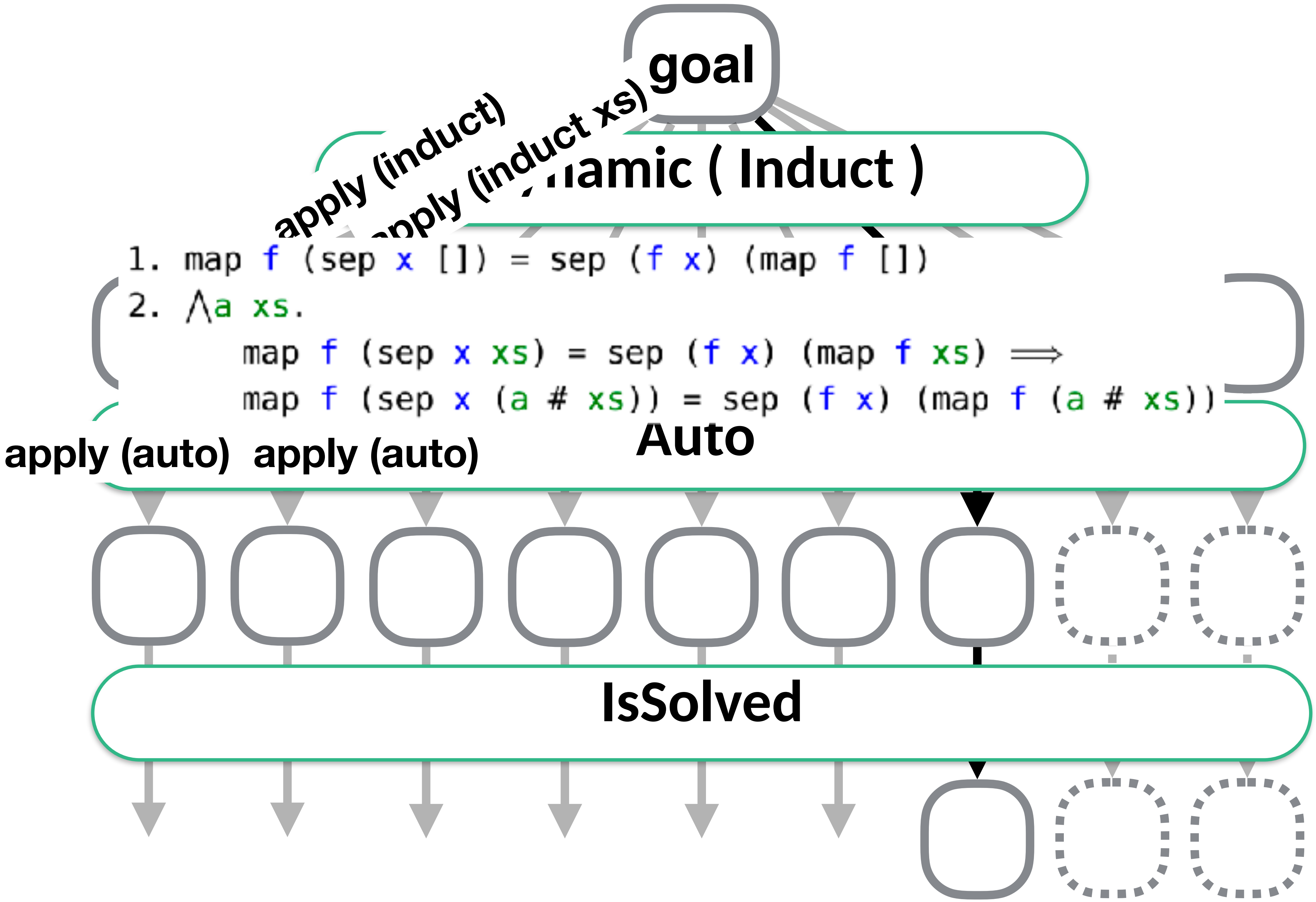
```
1. map f (sep x []) = sep (f x) (map f [])
2. ⋀a xs.
      map f (sep x xs) = sep (f x) (map f xs) ⟹
      map f (sep x (a # xs)) = sep (f x) (map f (a # xs))
```

```
1. ⋀a xs.
      map f (sep x xs) = sep (f x) (map f xs) ⟹
      map f (sep x (a # xs)) = sep (f x) (f a # map f xs)
```

https://twitter.com/YutakangE

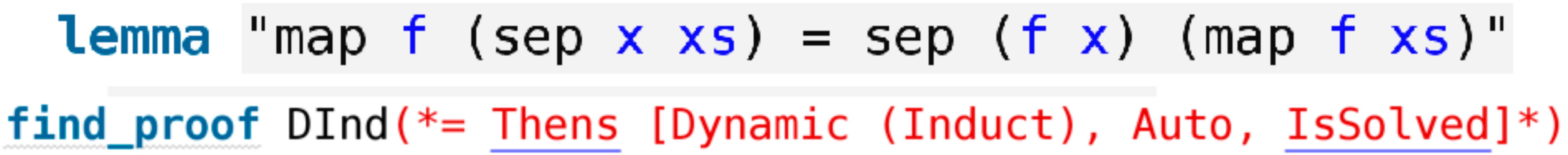


```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```

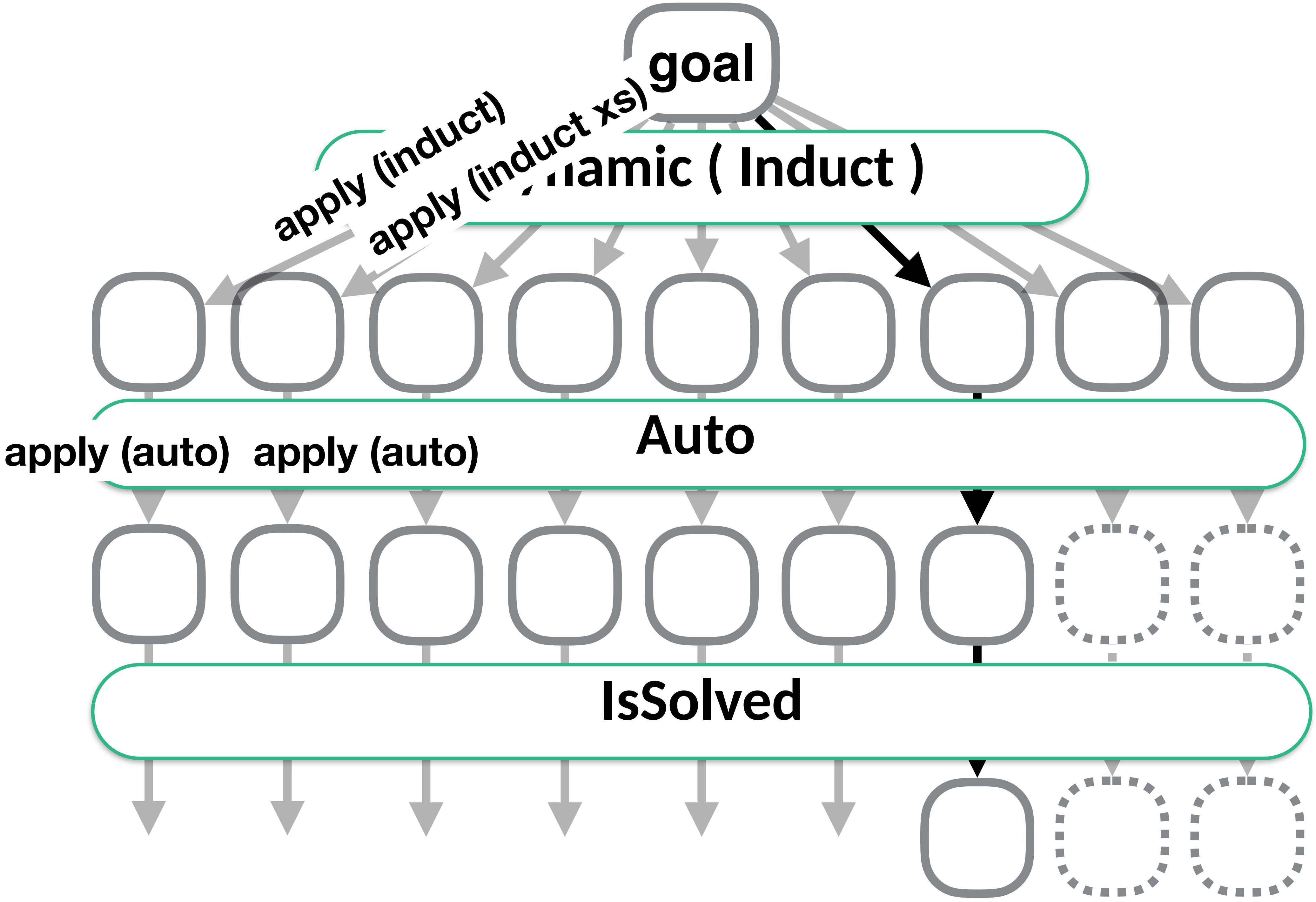


```
1. map f (sep x []) = sep (f x) (map f [])
2. ⋀a xs.
      map f (sep x xs) = sep (f x) (map f xs) ⟹
      map f (sep x (a # xs)) = sep (f x) (map f (a # xs))
```

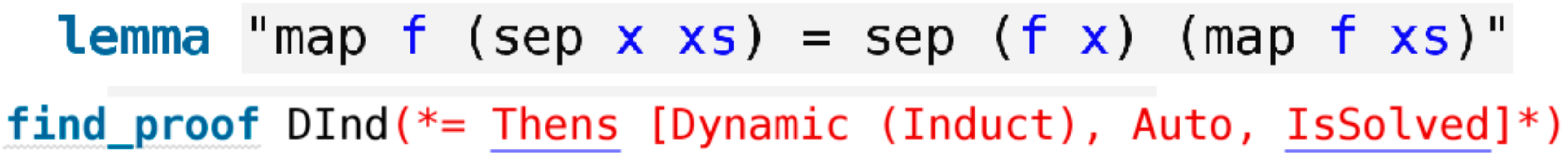
https://twitter.com/YutakangE



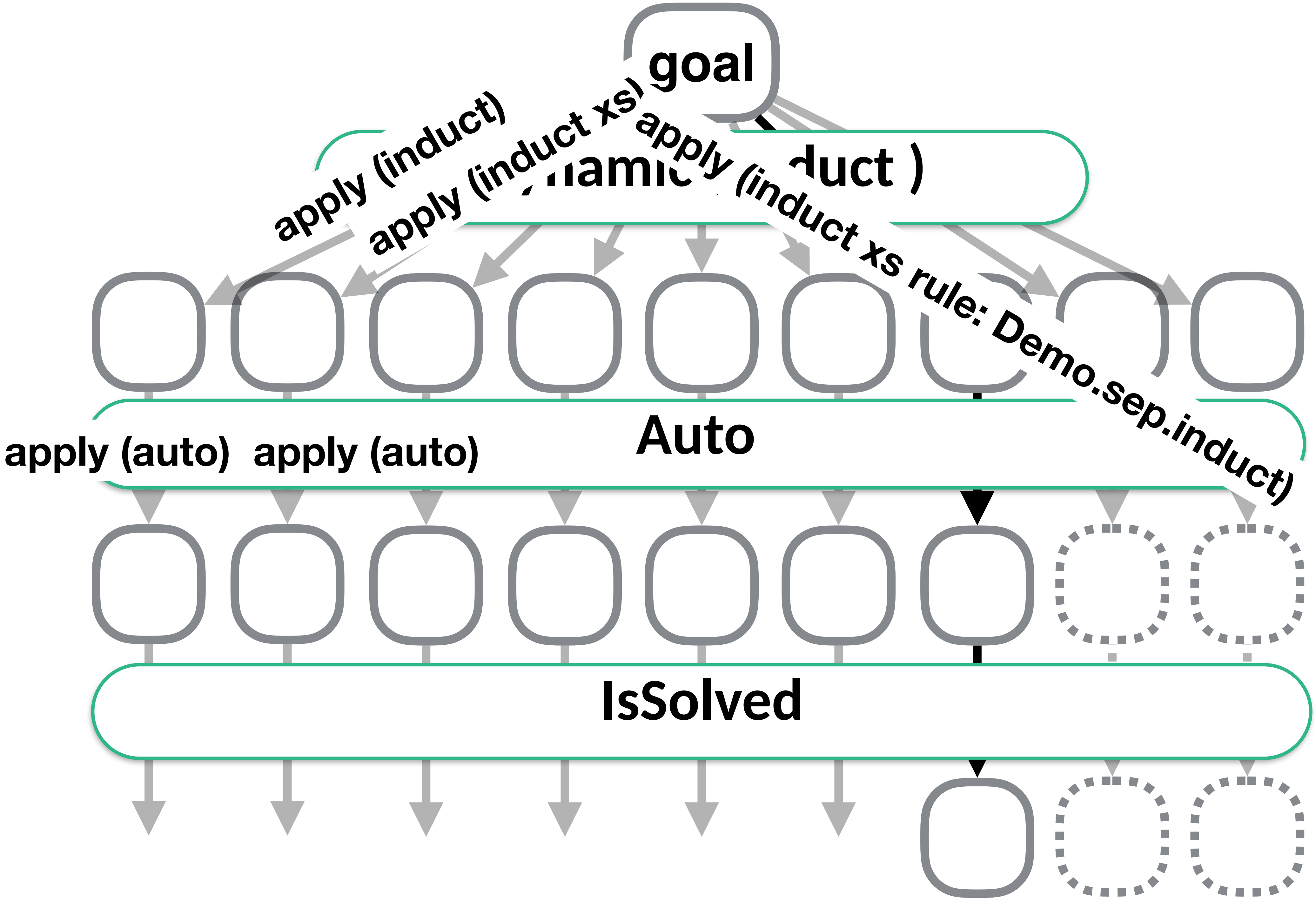
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"

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



https://twitter.com/YutakangE

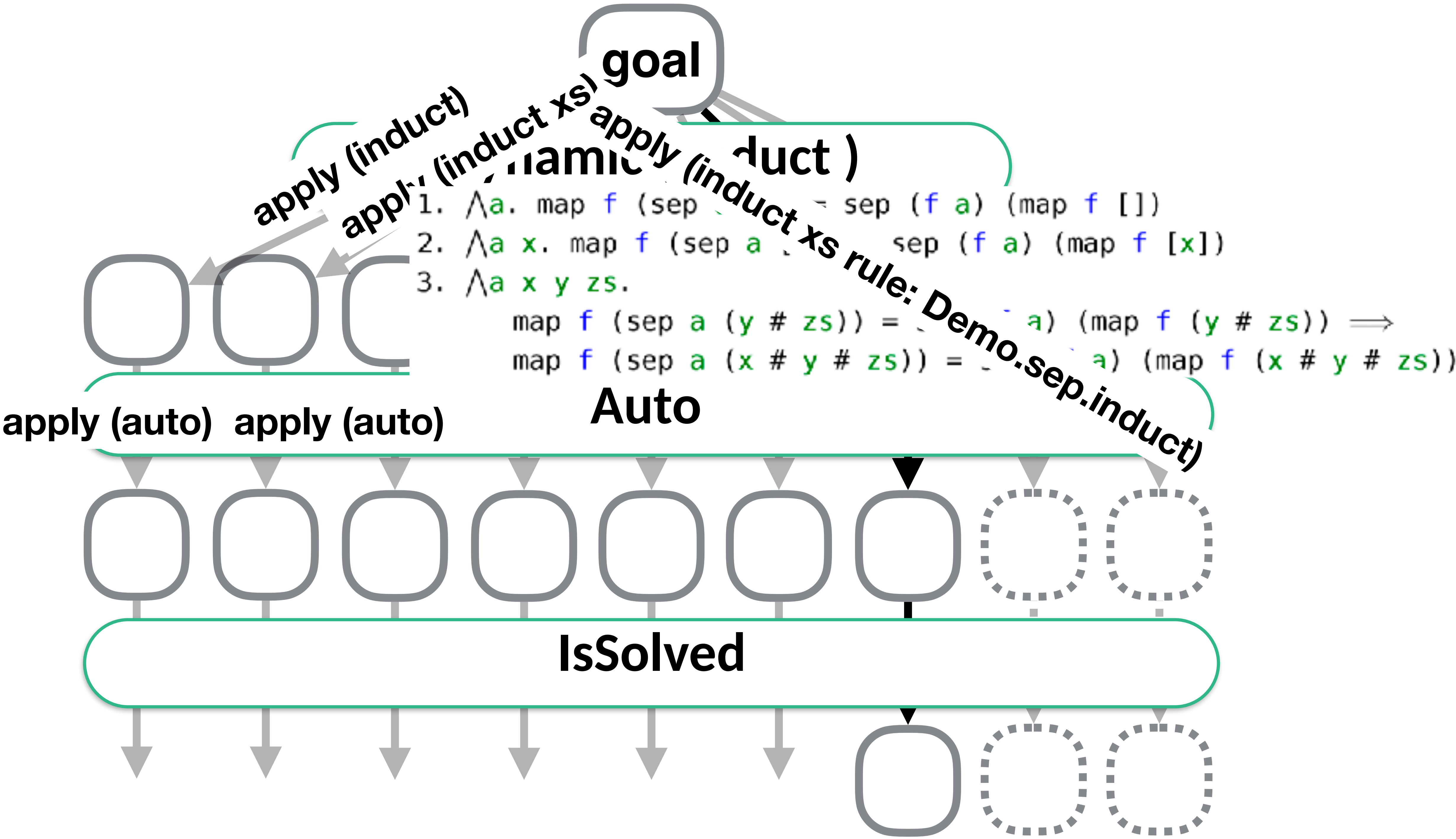


```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



https://twitter.com/YutakangE

```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```

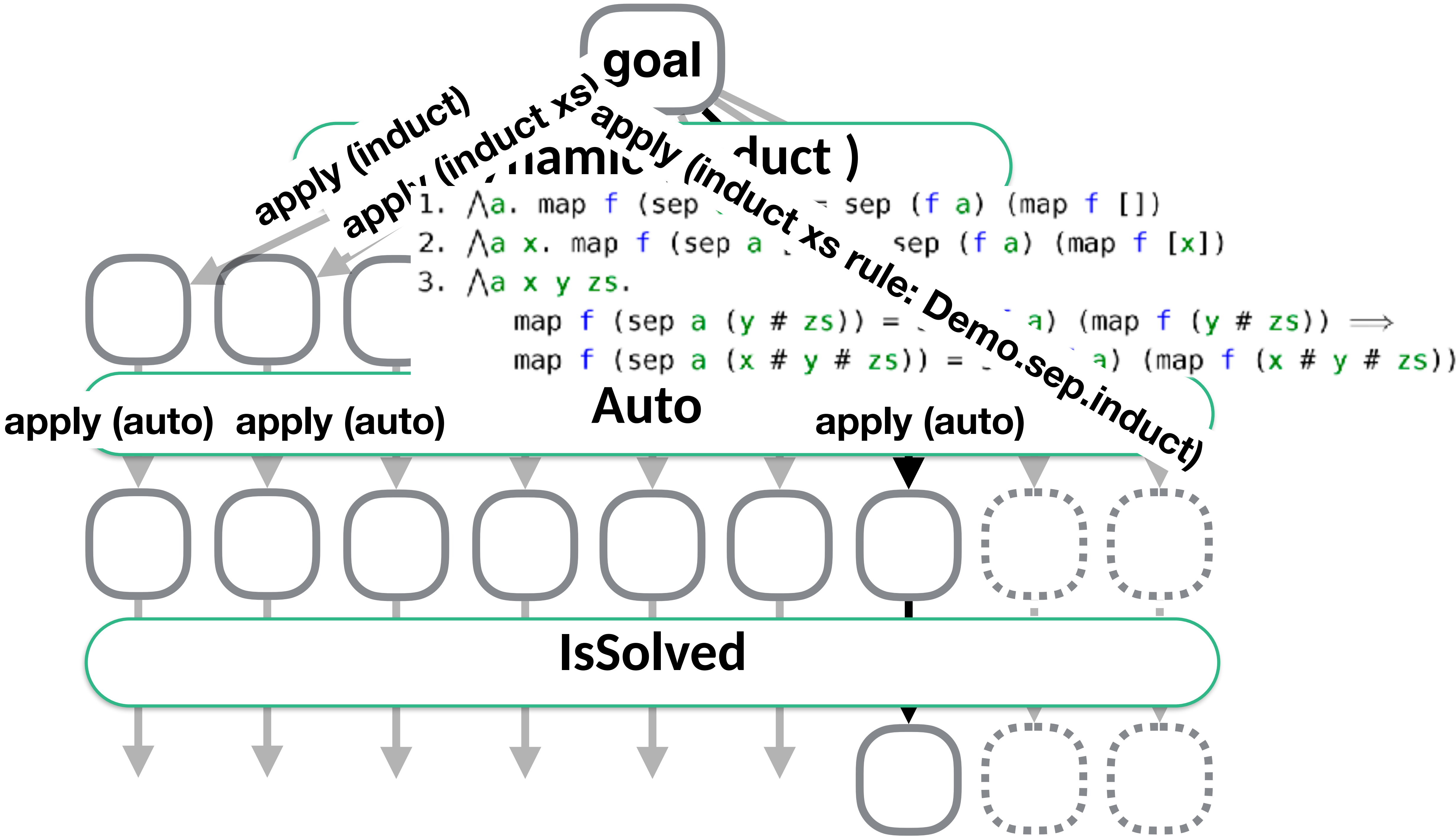


```
1. ⋀a. map f (sep              sep (f a) (map f [])
2. ⋀a x. map f (sep a         sep (f a) (map f [x])
3. ⋀a x y zs.
      map f (sep a (y # zs)) =        a) (map f (y # zs)) ⟹
      map f (sep a (x # y # zs)) =        a) (map f (x # y # zs))
```

https://twitter.com/YutakangE

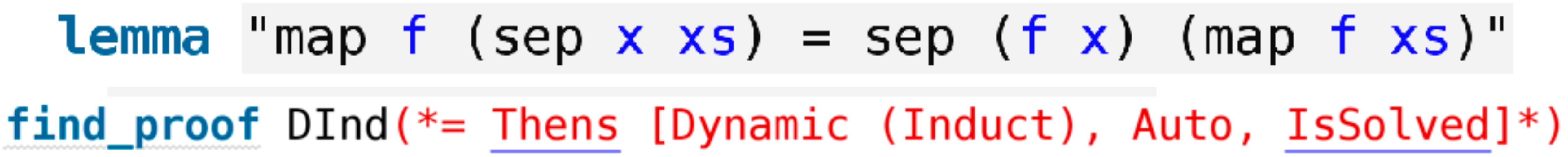
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

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

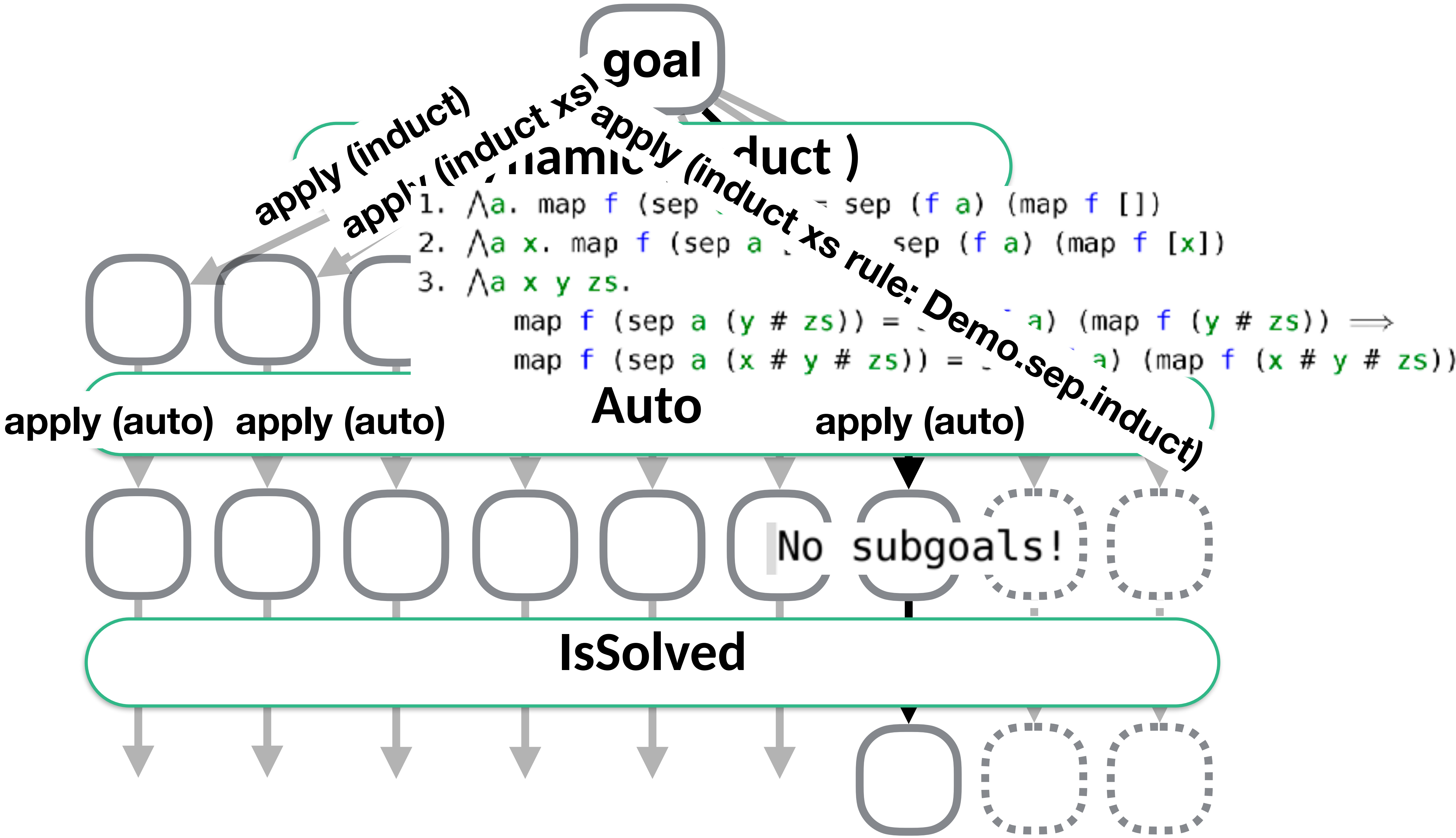


```
1. ⋀a. map f (sep                = sep (f a) (map f [])
2. ⋀a x. map f (sep a               sep (f a) (map f [x])
3. ⋀a x y zs.
      map f (sep a (y # zs)) =               a) (map f (y # zs)) ⟹
      map f (sep a (x # y # zs)) =               a) (map f (x # y # zs))
```

https://twitter.com/YutakangE



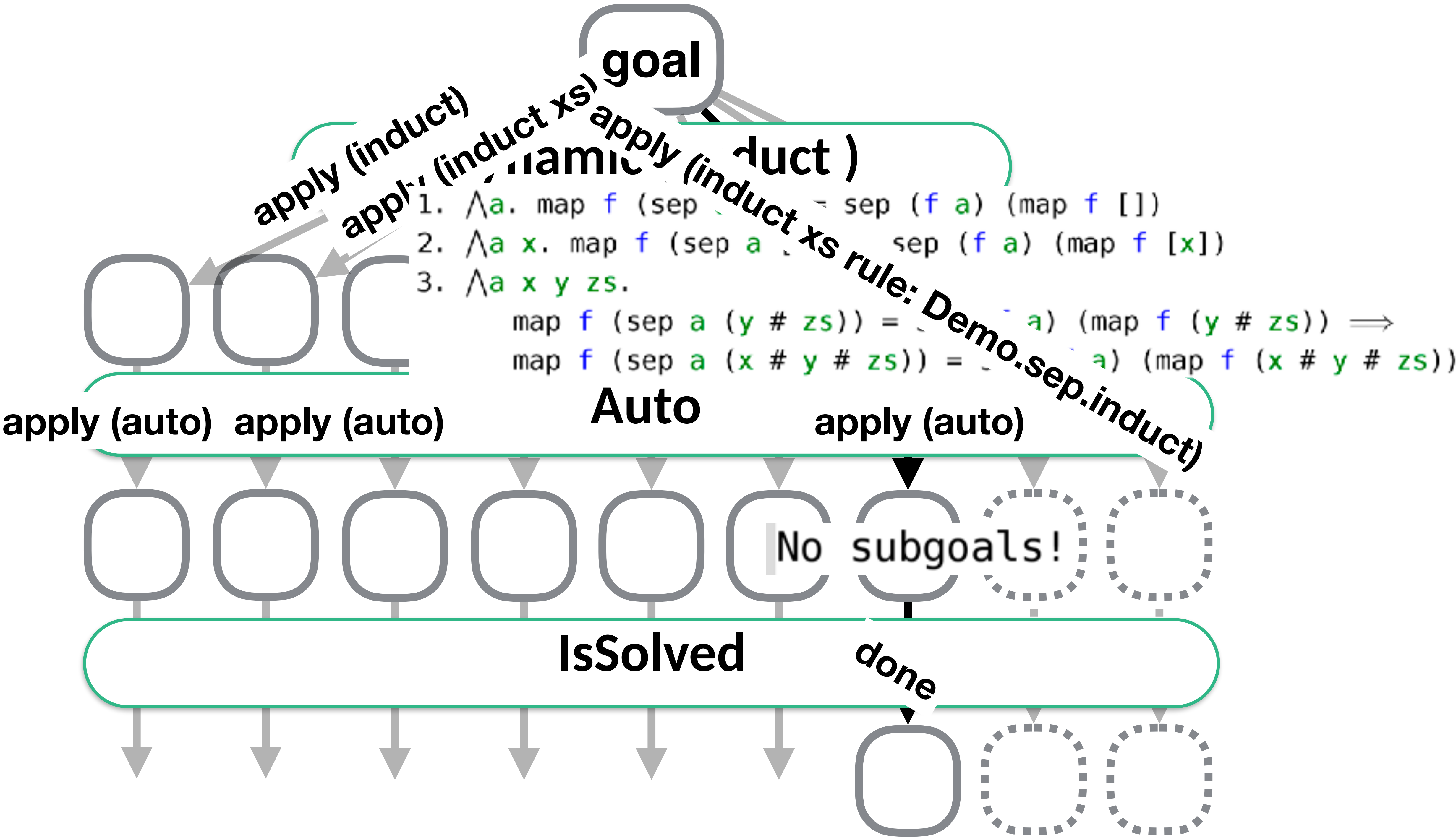
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



```
1. ⋀a. map f (sep   = sep (f a) (map f [])
2. ⋀a x. map f (sep a   sep (f a) (map f [x])
3. ⋀a x y zs.
      map f (sep a (y # zs)) =   a) (map f (y # zs)) ⟹
      map f (sep a (x # y # zs)) =   a) (map f (x # y # zs))
```

https://twitter.com/YutakangE

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

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

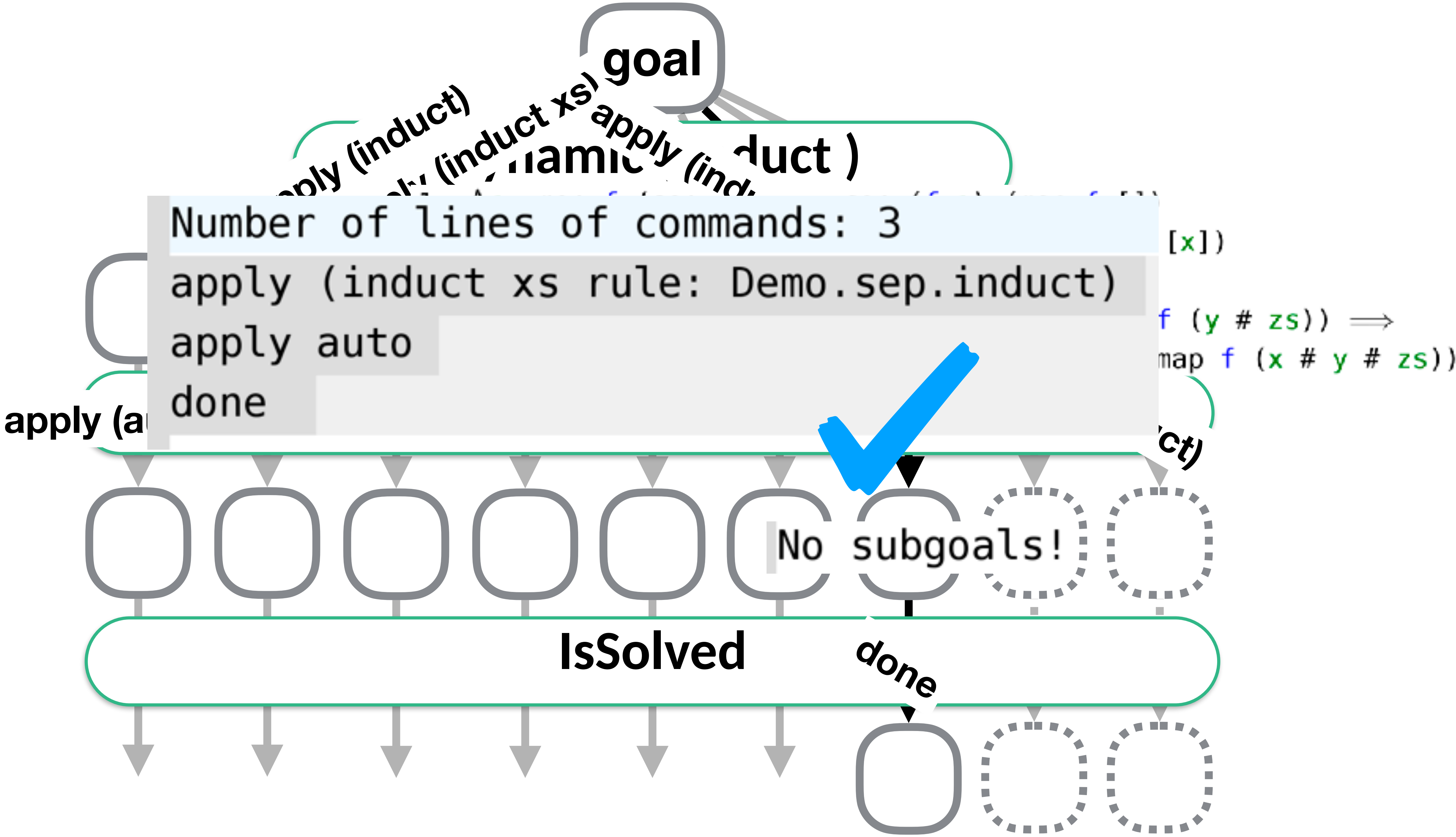


```
1. ⋀a. map f (sep          sep (f a) (map f [])
2. ⋀a x. map f (sep a     sep (f a) (map f [x])
3. ⋀a x y zs.
      map f (sep a (y # zs)) =          a) (map f (y # zs)) ⟹
      map f (sep a (x # y # zs)) =          a) (map f (x # y # zs))
```

```
No subgoals!
```

https://twitter.com/YutakangE

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

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



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

Try_Hard: the default strategy

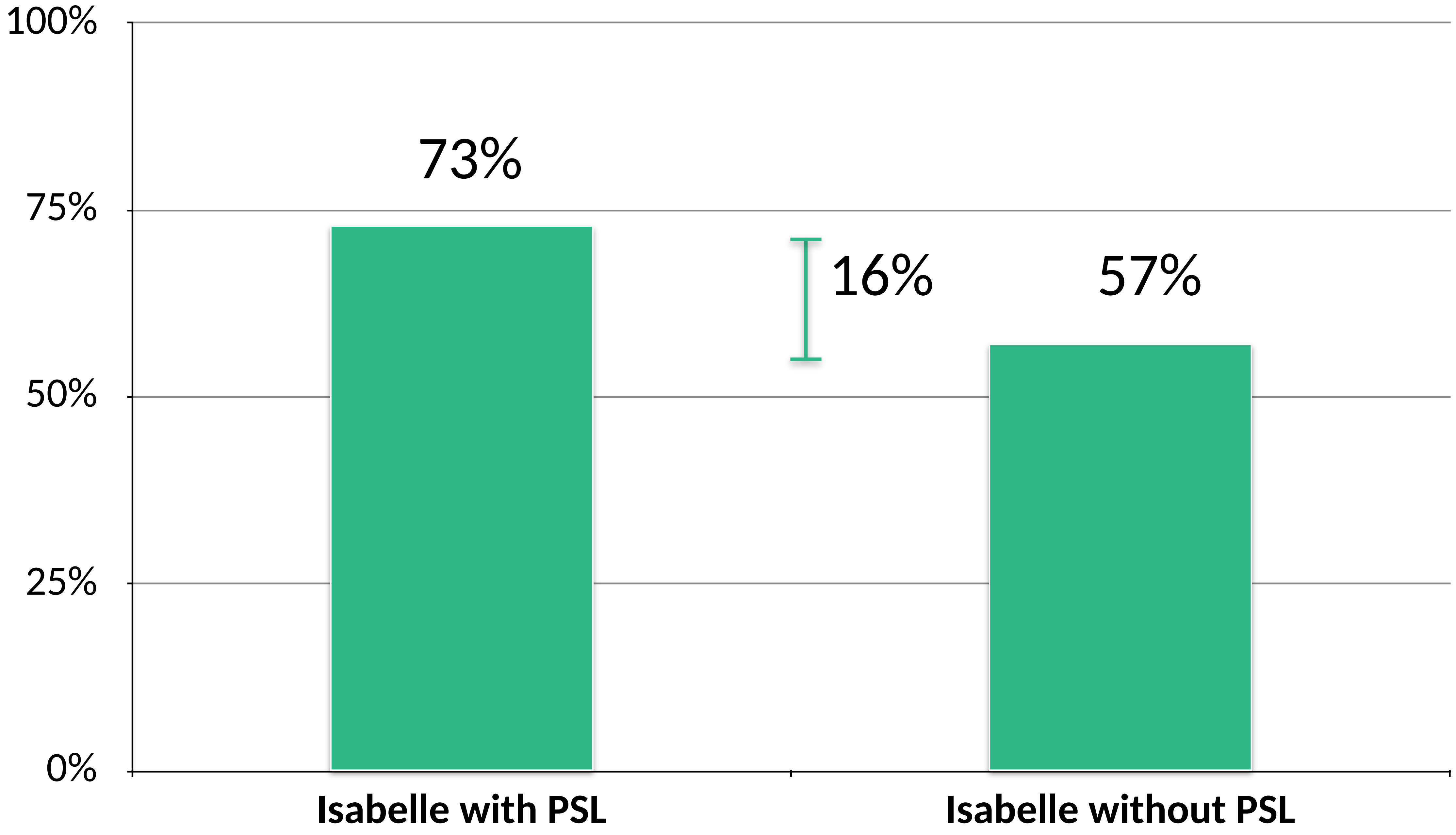
```
strategy Try_Hard =
Ors [Thens [Subgoal, Basic],
      Thens [DInductTac, Auto_Solve],
      Thens [DCaseTac, Auto_Solve],
      Thens [Subgoal, Advanced],
      Thens [DCaseTac, Solve_Many],
      Thens [DInductTac, Solve_Many] ]
```

```
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]]
```

https://twitter.com/YutakangE

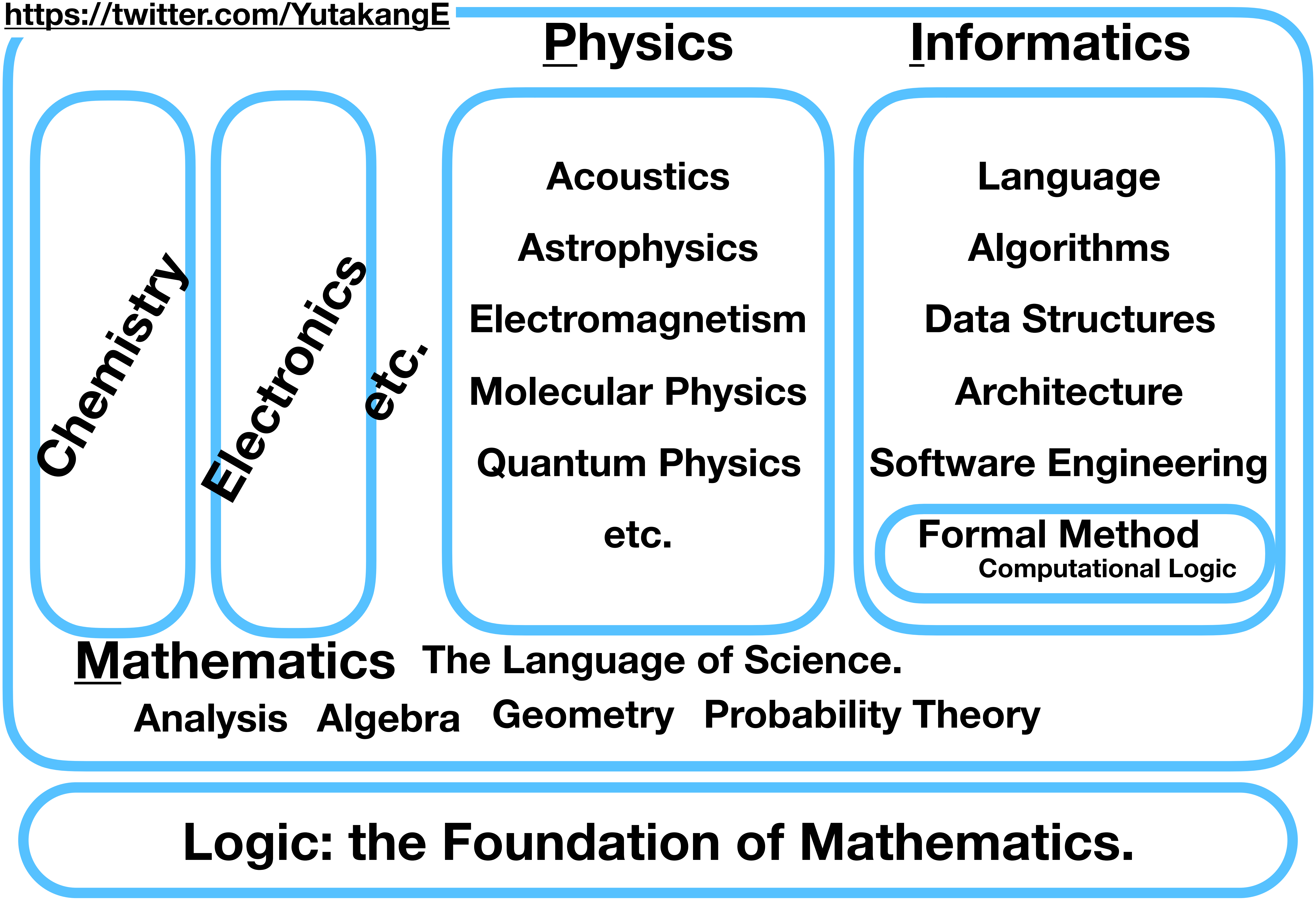
Evaluation

The percentage of automatically proved obligations out of 1526 proof obligations (timeout = 300s) from the AFP and courseworks.



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

https://twitter.com/YutakangE



Physics

Informatics

Chemistry

Electronics

etc.

Acoustics

Astrophysics

Electromagnetism

Molecular Physics

Quantum Physics

etc.

Language

Algorithms

Data Structures

Architecture

Software Engineering

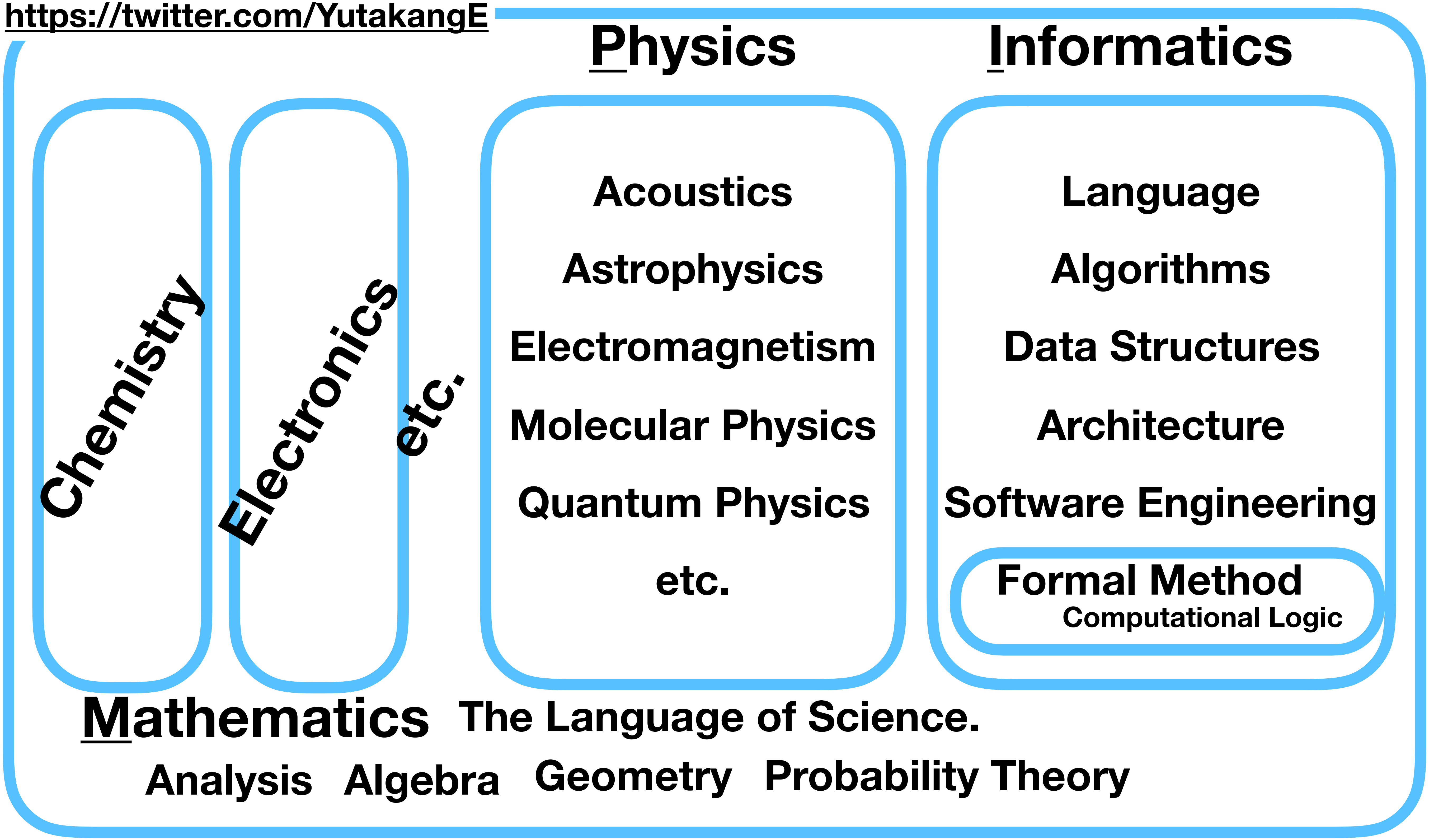
Formal Method
Computational Logic

Mathematics The Language of Science.

Analysis Algebra Geometry Probability Theory

Logic: the Foundation of Mathematics.

https://twitter.com/YutakangE



Chemistry

Electronics

etc.

Physics

Acoustics

Astrophysics

Electromagnetism

Molecular Physics

Quantum Physics

etc.

Informatics

Language

Algorithms

Data Structures

Architecture

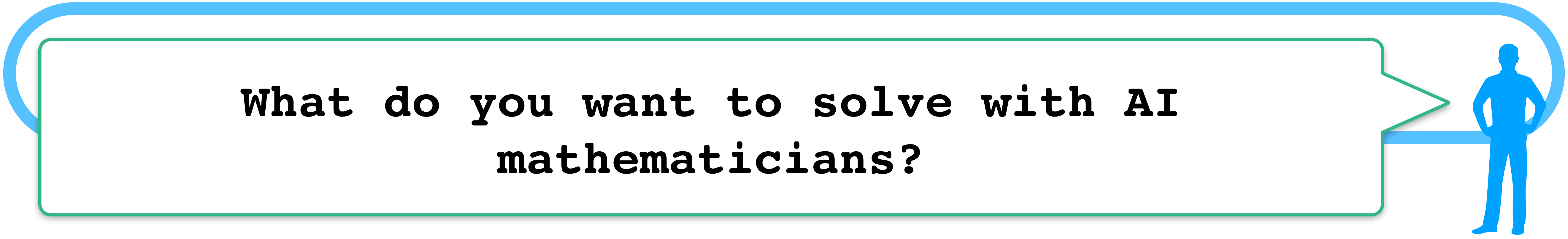
Software Engineering

Formal Method

Computational Logic

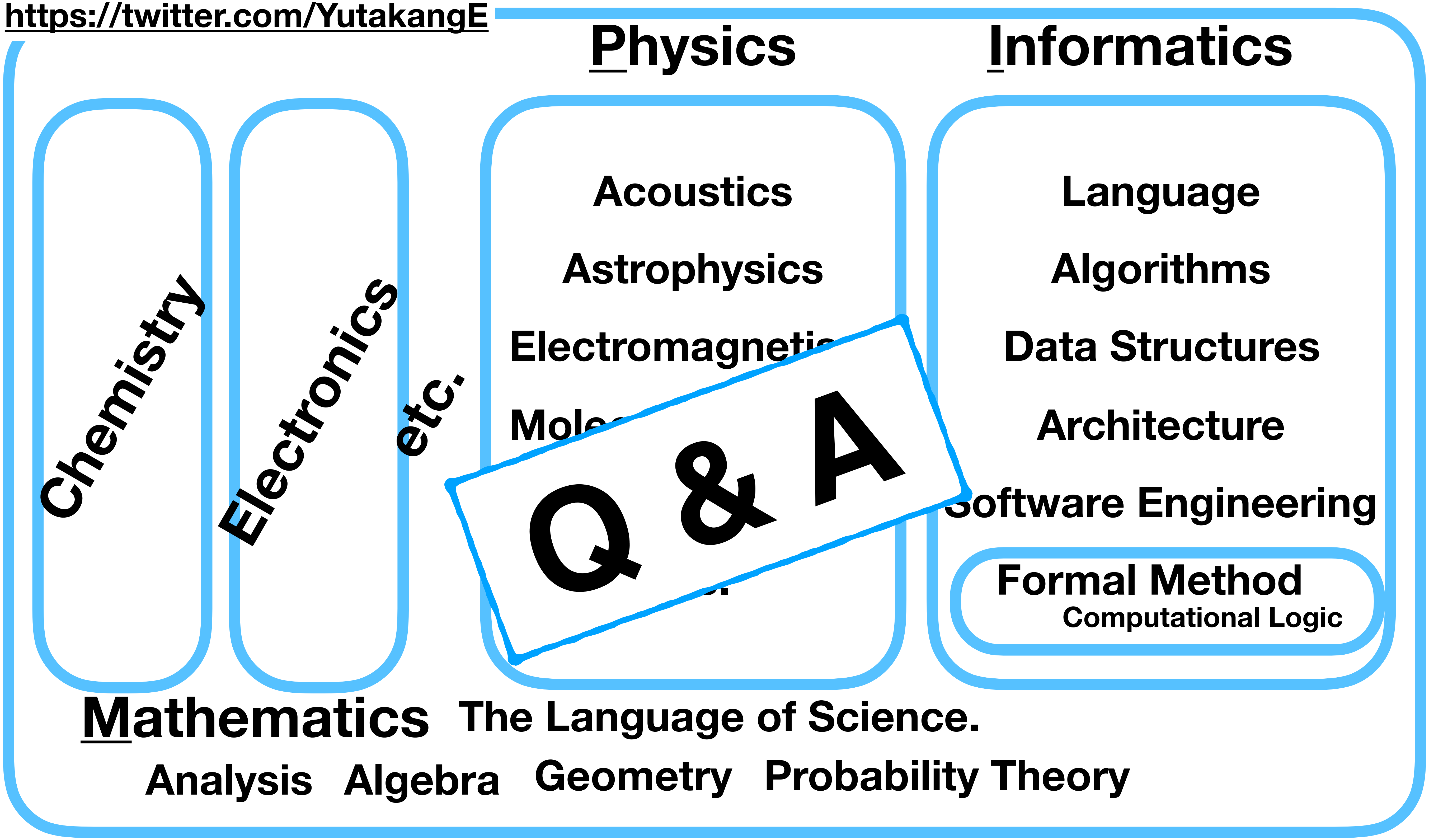
Mathematics The Language of Science.

Analysis Algebra Geometry Probability Theory



`What do you want to solve with AI mathematicians?`

https://twitter.com/YutakangE



Physics

Informatics

Chemistry

Electronics

etc.

Acoustics

Astrophysics

Electromagneti

Mole

Language

Algorithms

Data Structures

Architecture

Software Engineering

Formal Method

Computational Logic

Q & A

Mathematics The Language of Science.

Analysis Algebra Geometry Probability Theory

What do you want to solve with AI mathematicians?

preparation phase

How does PaMpeR work?

recommendation phase

preparation phase

large proof corpora

AFP and standard library

How does PaMpeR work?

recommendation phase

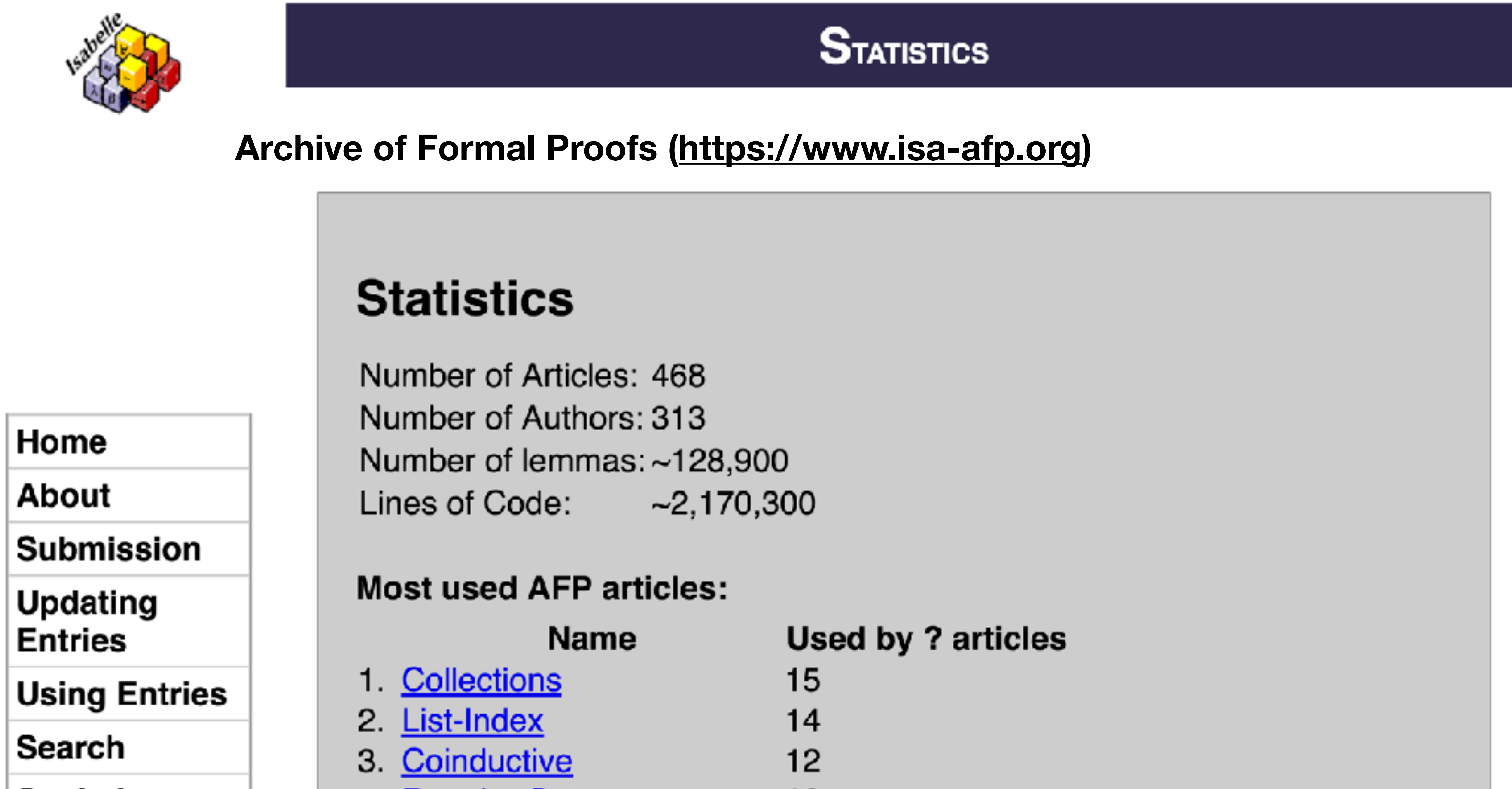
proof state

proof engineer

preparation phase

large proof corpora

AFP and standard library



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.	Collections	15
2.	List-Index	14
3.	Coinductive	12

preparation phase

large proof corpora

AFP and standard library

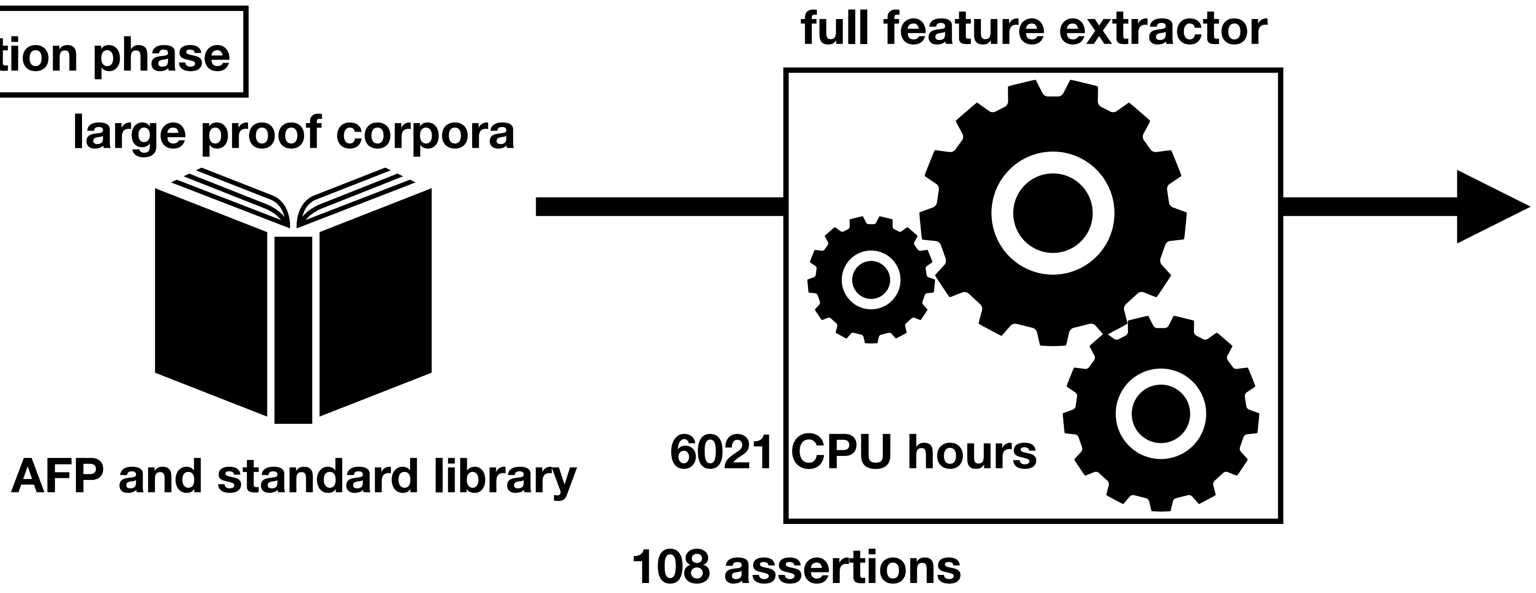
How does PaMpeR work?

recommendation phase

proof state

proof engineer

preparation phase



large proof corpora

AFP and standard library

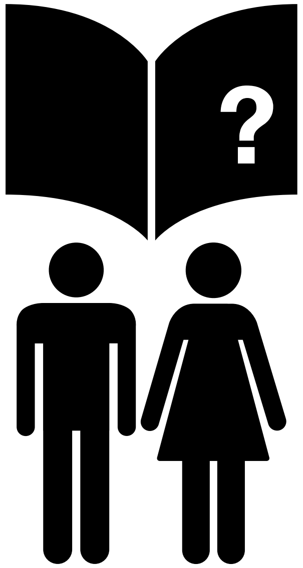
full feature extractor

6021 CPU hours

108 assertions

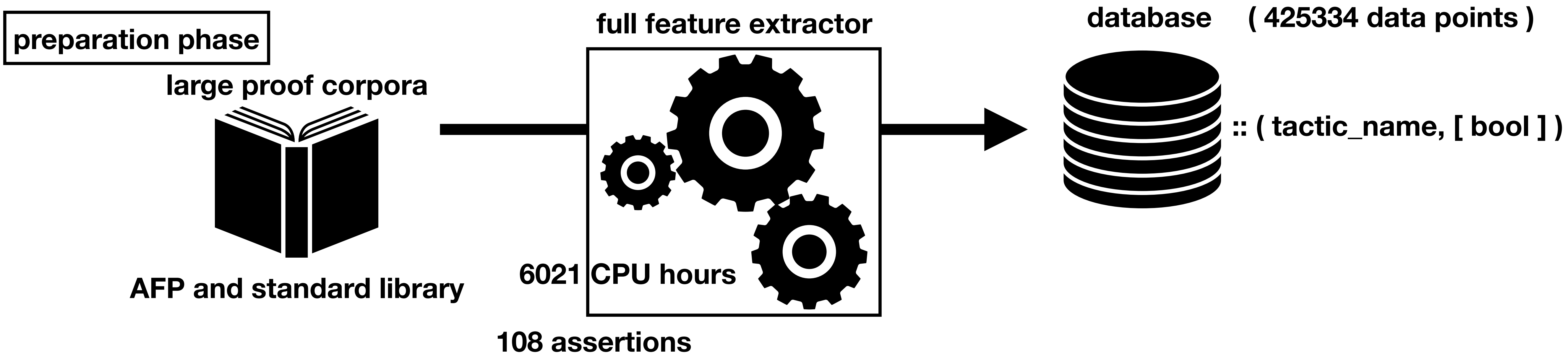
How does PaMpeR work?

recommendation phase



proof state

proof engineer



preparation phase

large proof corpora

AFP and standard library

full feature extractor

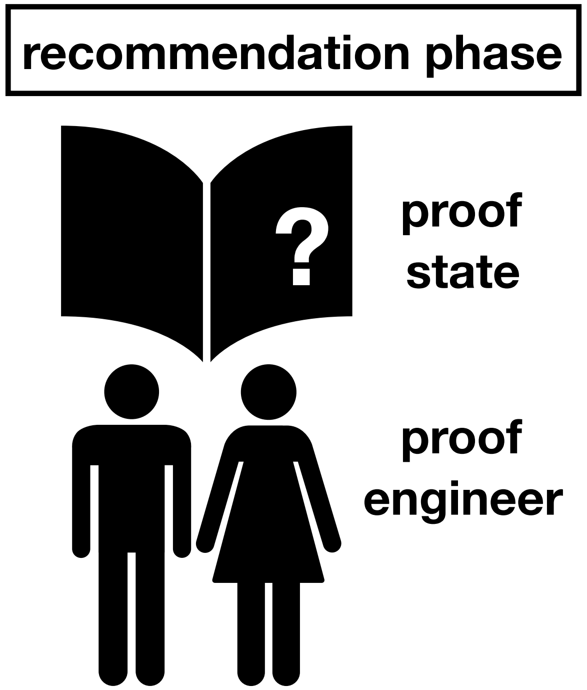
6021 CPU hours

108 assertions

database (425334 data points)

:: (tactic_name, [bool])

How does PaMpeR work?

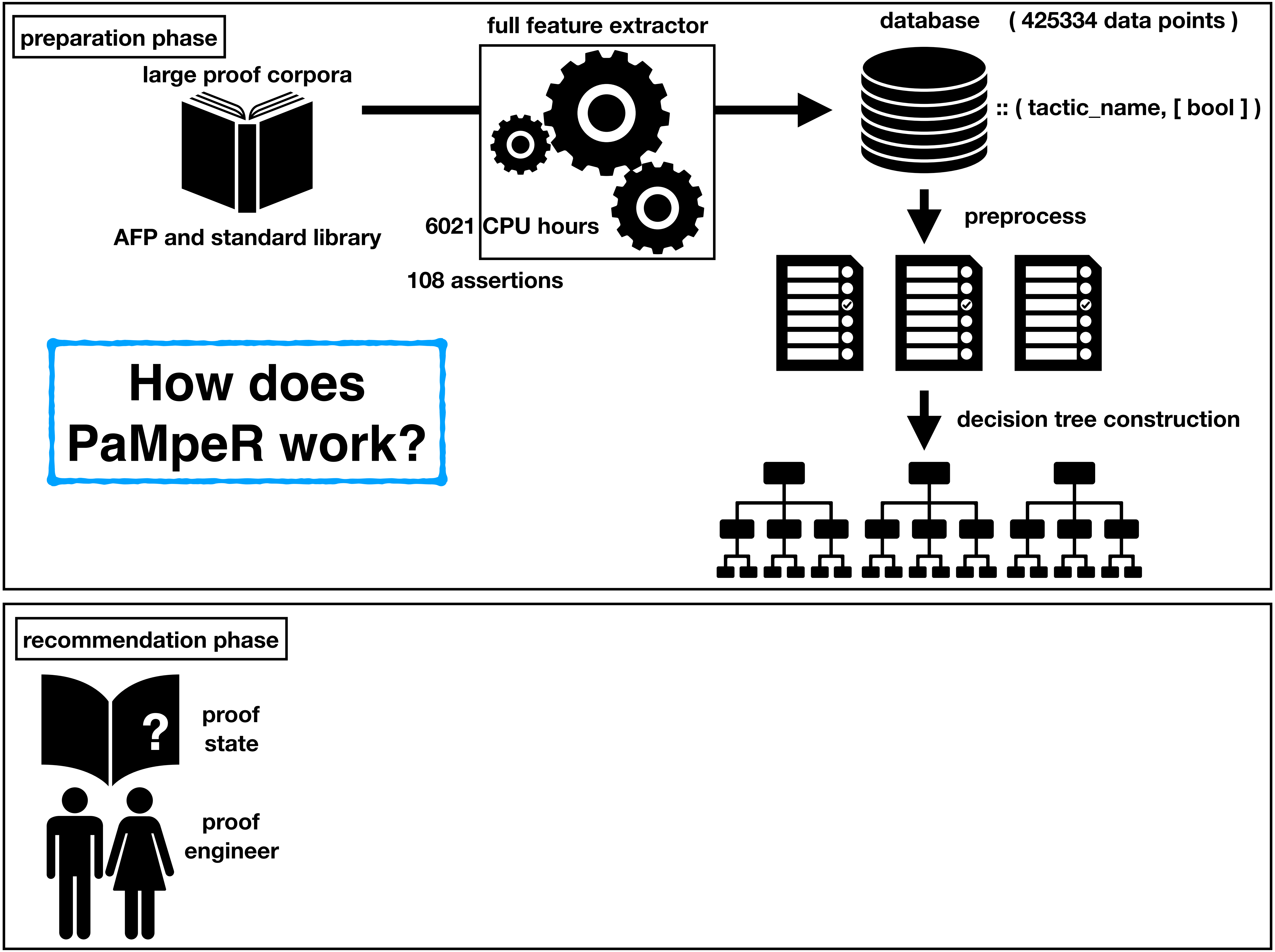


recommendation phase

proof state

proof engineer

How does PaMpeR work?



preparation phase

large proof corpora

AFP and standard library

full feature extractor

6021 CPU hours

108 assertions

database (425334 data points)

:: (tactic_name, [bool])

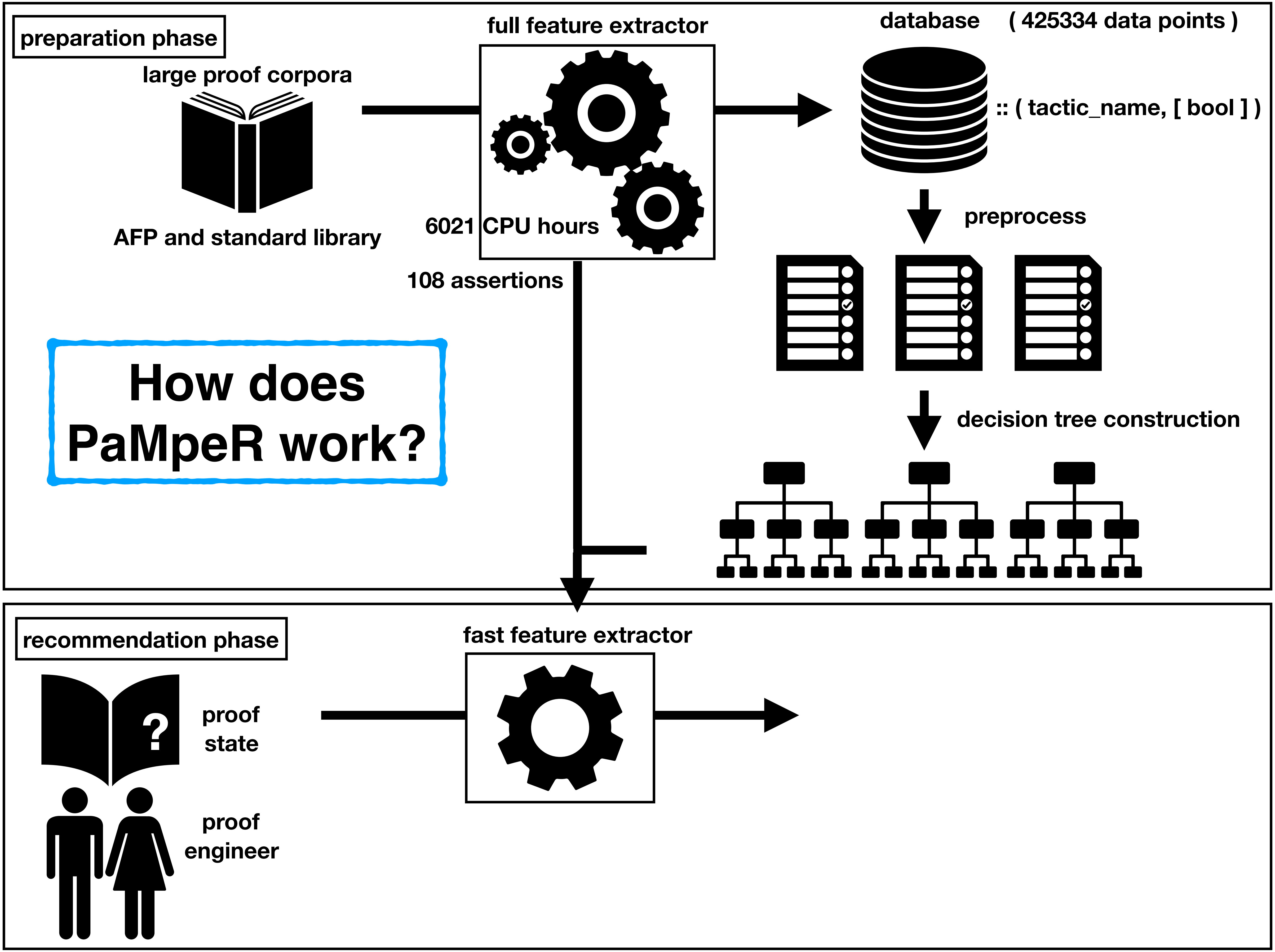
preprocess

decision tree construction

recommendation phase

proof state

proof engineer



How does PaMpeR work?

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

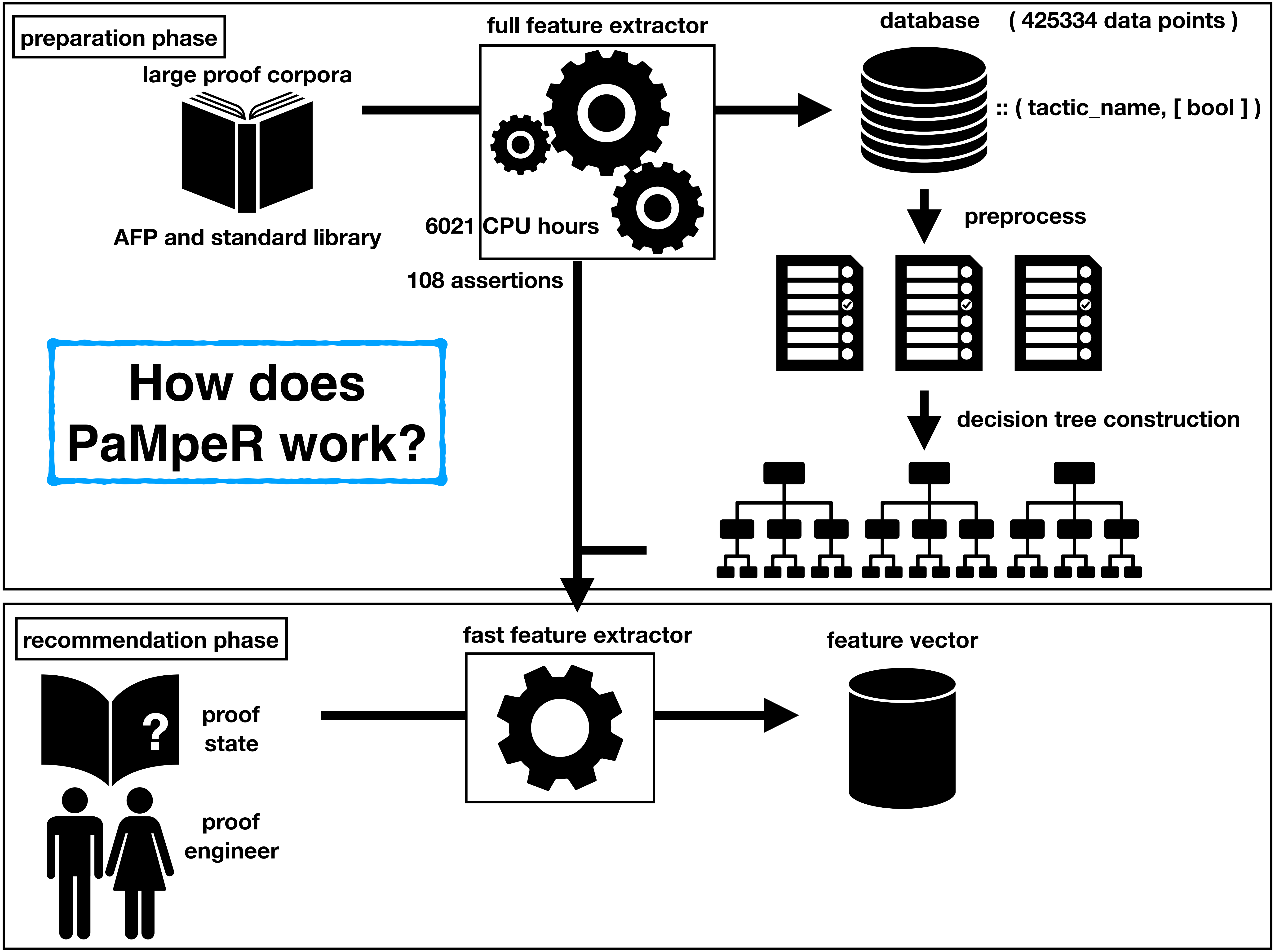
recommendation phase

proof state

proof engineer

fast feature extractor

How does PaMpeR work?



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

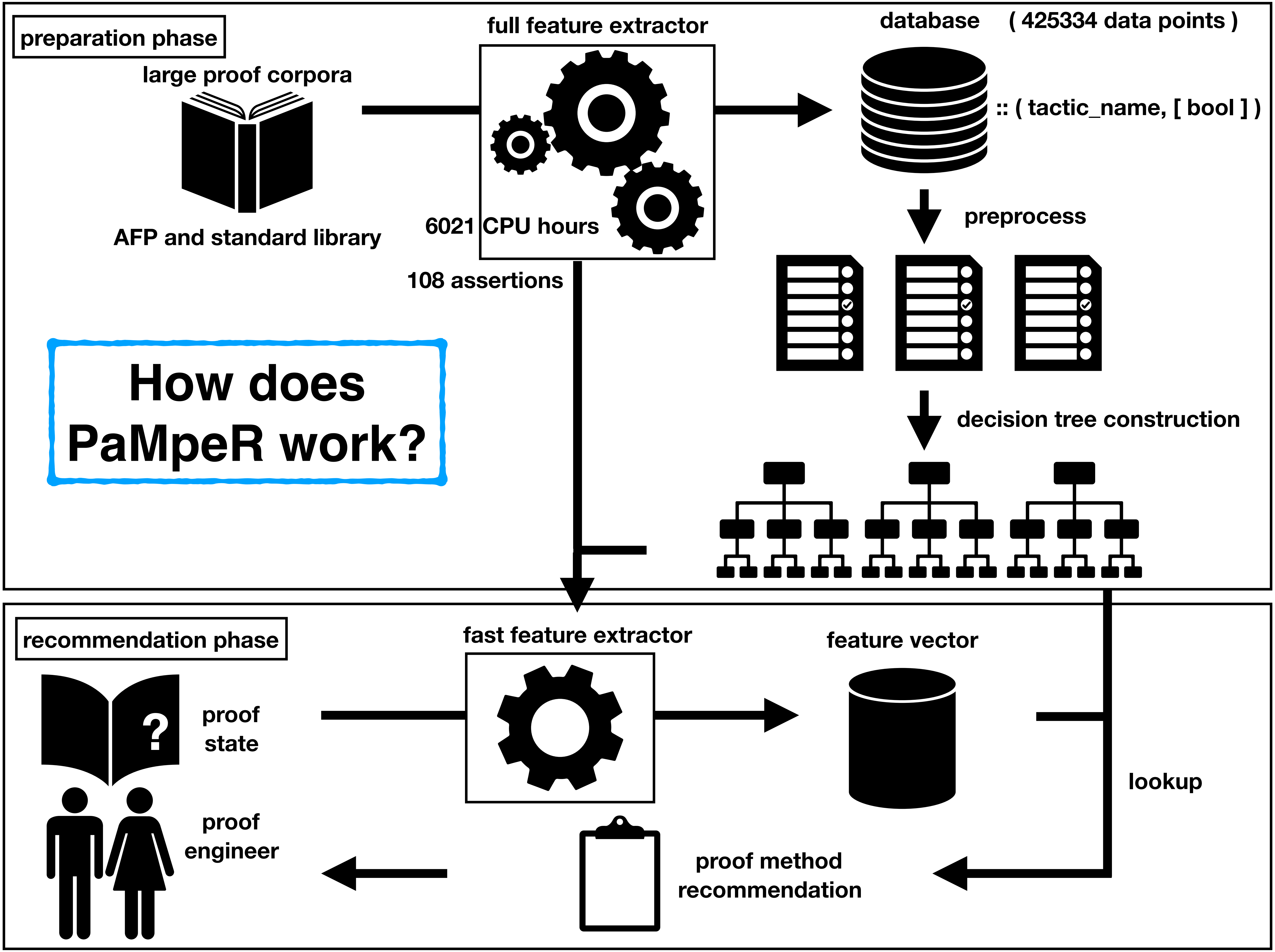
recommendation phase

proof state

proof engineer

fast feature extractor

feature vector



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

fast feature extractor

feature vector

lookup

proof method recommendation

Mathematics

Number Theory

Analysis

Algebra

Geometry

Probability Theory

etc.

Informatics

Language

Algorithms

Data Structures

Architecture

Software Engineering

Formal Verification

theorem proving

Physics

Acoustics

Astrophysics

Electromagnetism

Molecular Physics

Quantum Physics

etc.

Mathematics

Number Theory

Analysis

Algebra

Geometry

Probability Theory

etc.

Informatics

Language

Algorithms

Data Structures

Architecture

Software Engineering

Formal Verification

theorem proving

Physics

Acoustics

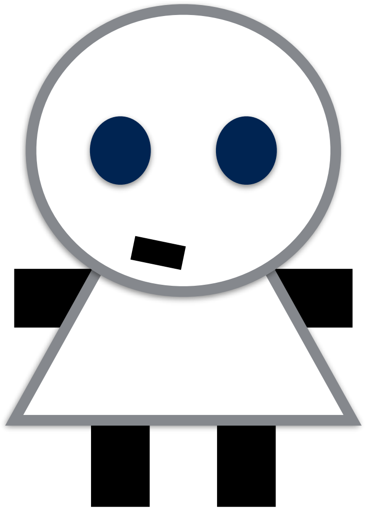
Astrophysics

Electromagnetism

Molecular Physics

Quantum Physics

etc.



A tiny field inside Informatics. Who cares?

Summary

PSL can find how to apply induction for easy problems.

CADE2017 (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)