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

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

universität innsbruck

CZECH INSTITUTE OF INFORMATICS ROBOTICS AND CYBERNETICS CTU IN PRAGUE

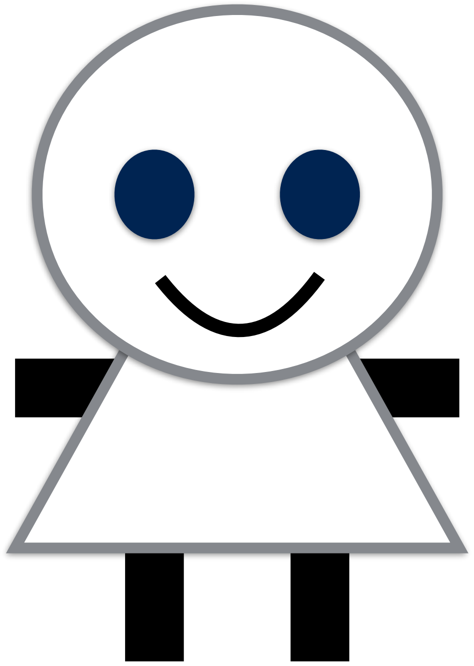
Yutaka Ng
yutakang

Block or report user

CVUT, CTU, CIIRC

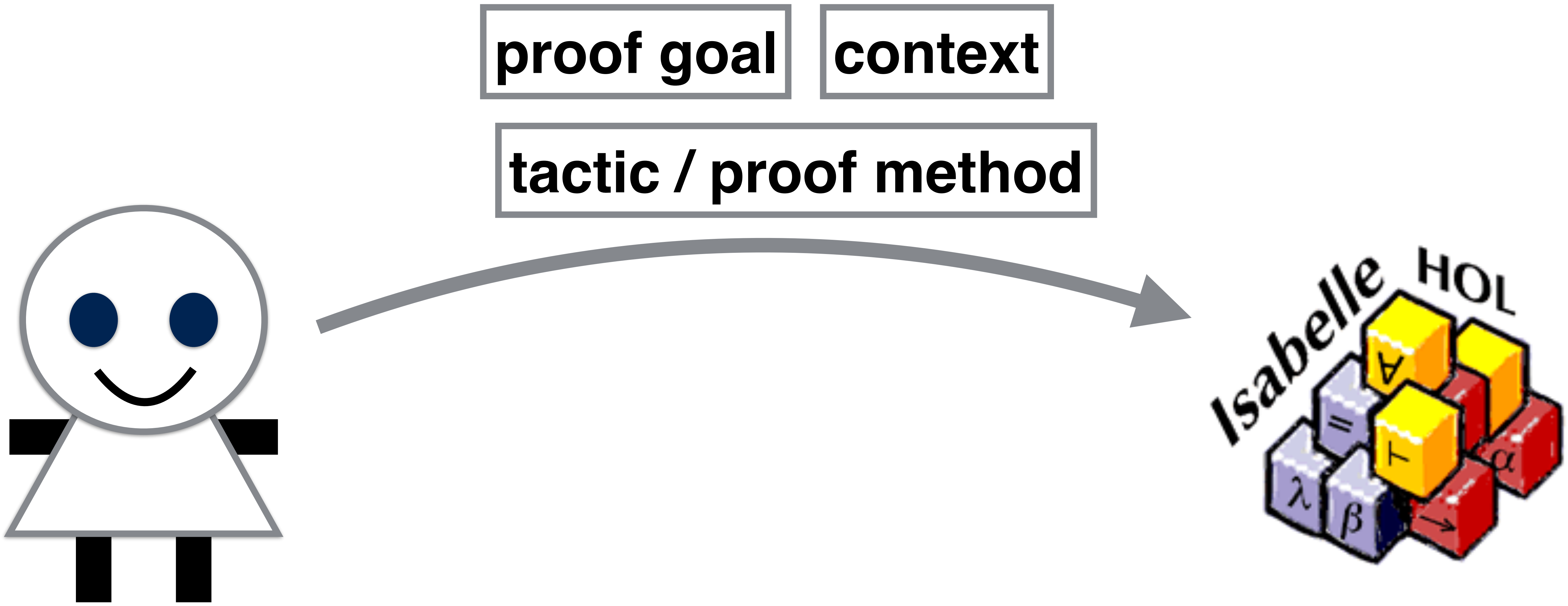
```
git clone https://github.com/data61/PSL
```

Interactive theorem proving with Isabelle/HOL



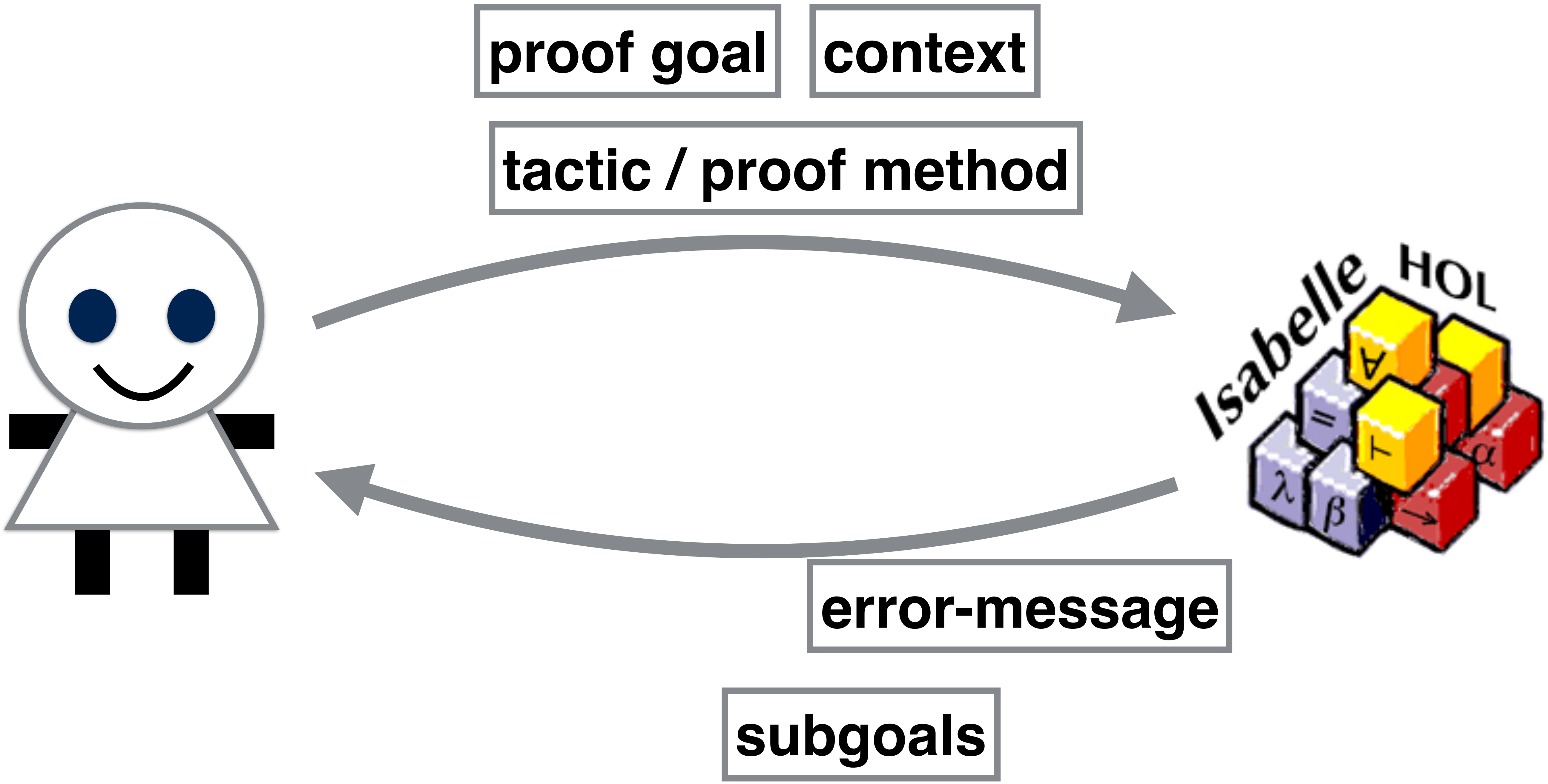
git clone https://github.com/data61/PSL

Interactive theorem proving with Isabelle/HOL



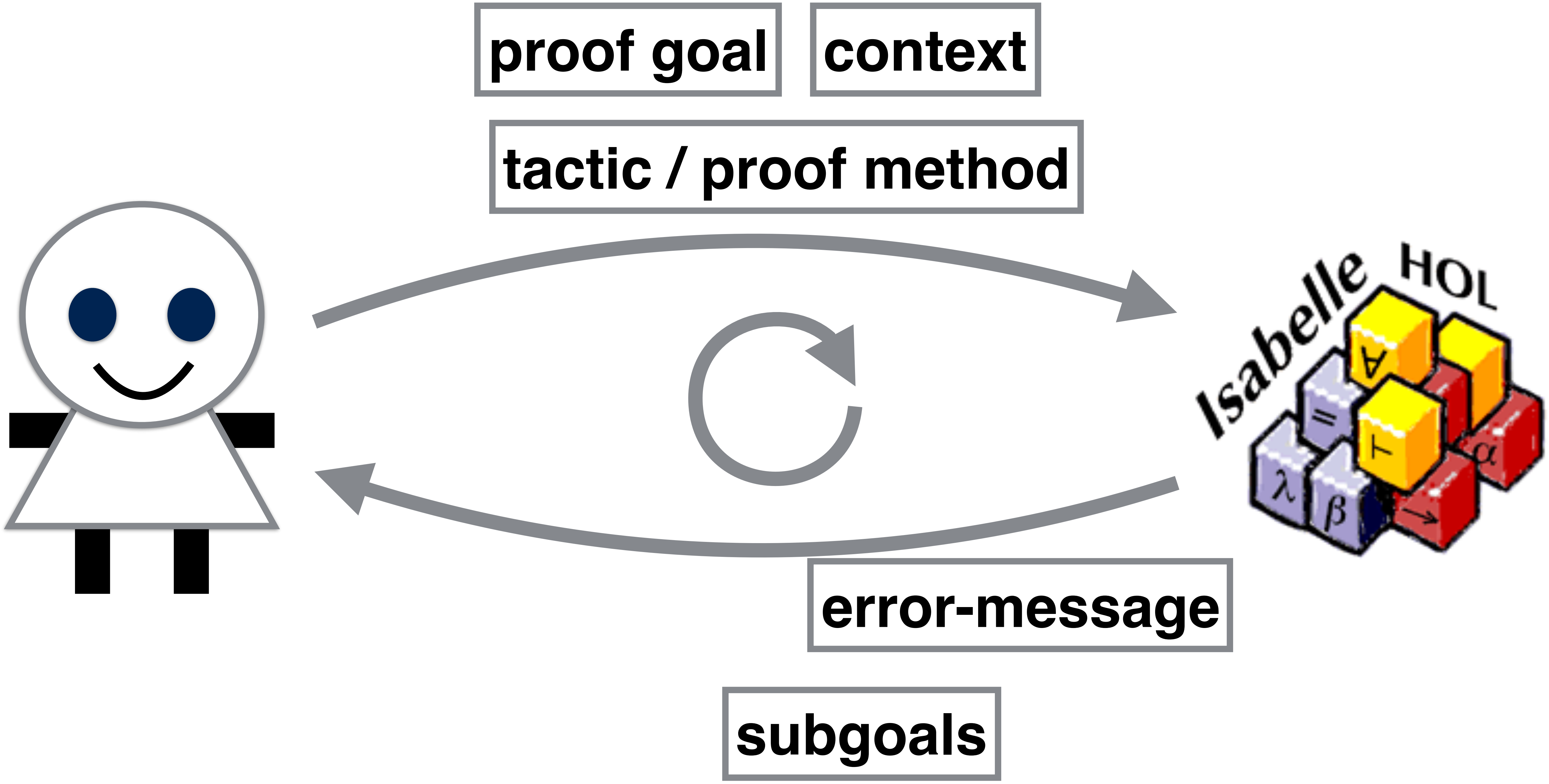
```
git clone https://github.com/data61/PSL
```

Interactive theorem proving with Isabelle/HOL



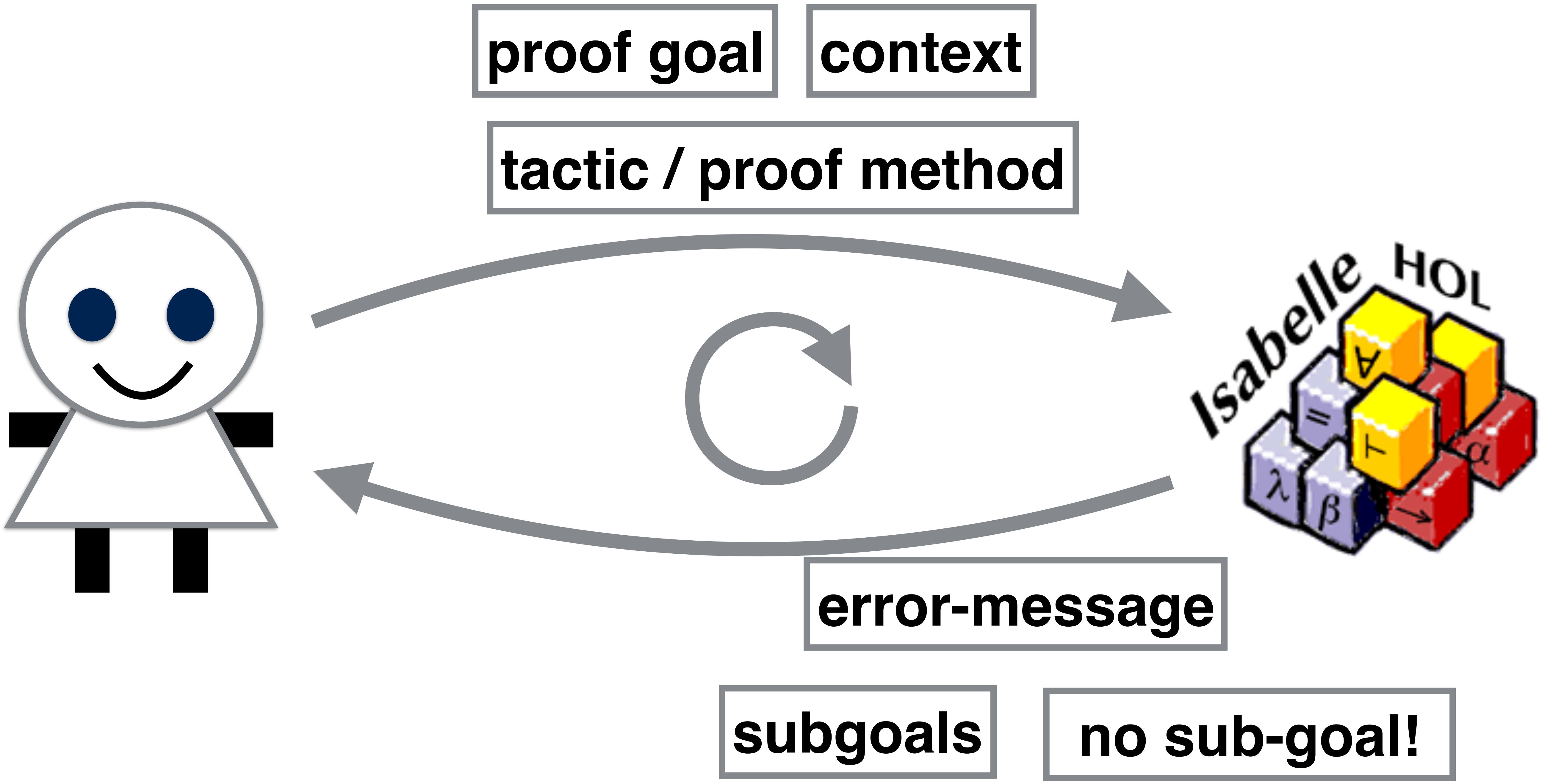
```
git clone https://github.com/data61/PSL
```

Interactive theorem proving with Isabelle/HOL



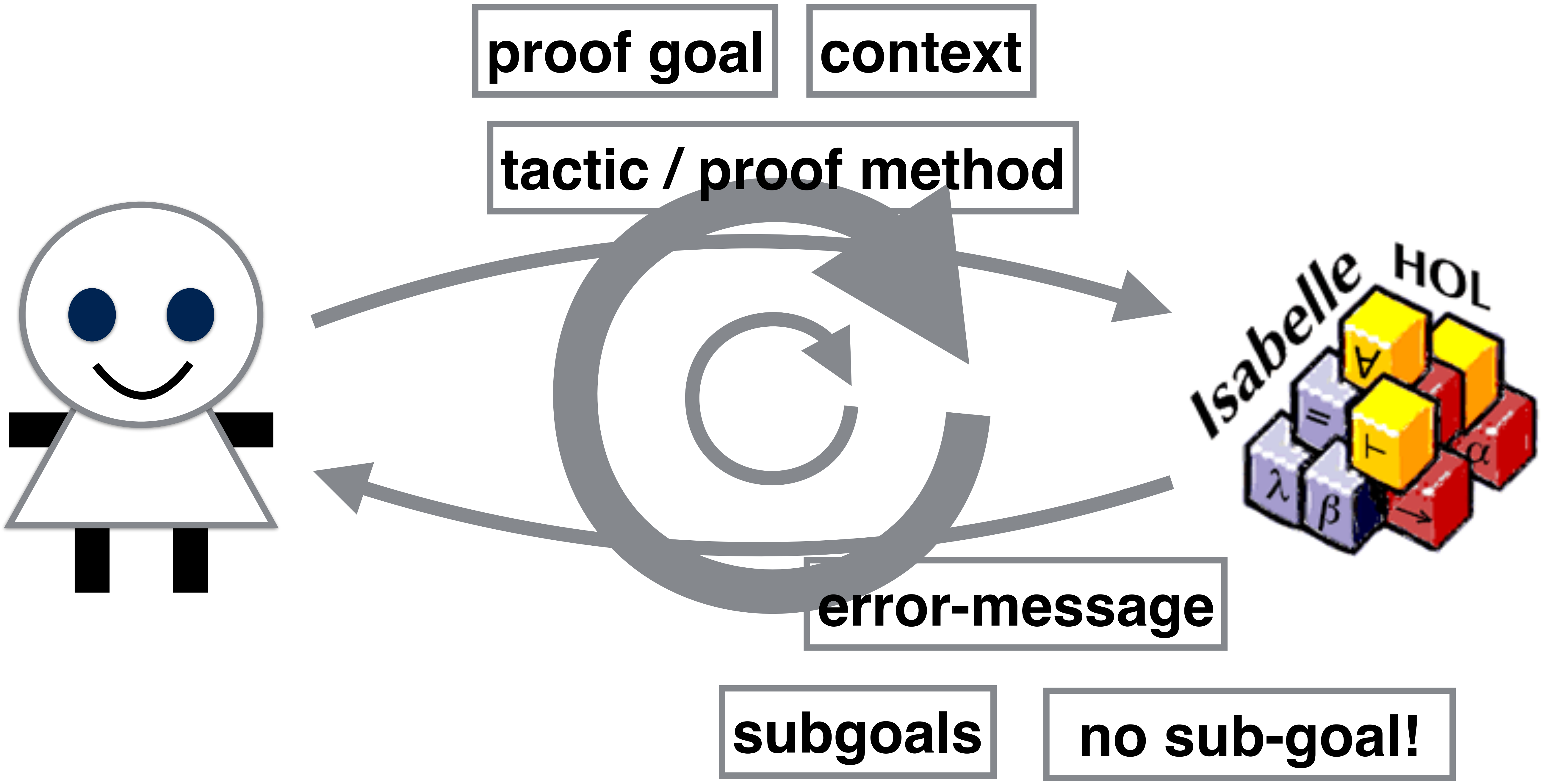
git clone https://github.com/data61/PSL

Interactive theorem proving with Isabelle/HOL



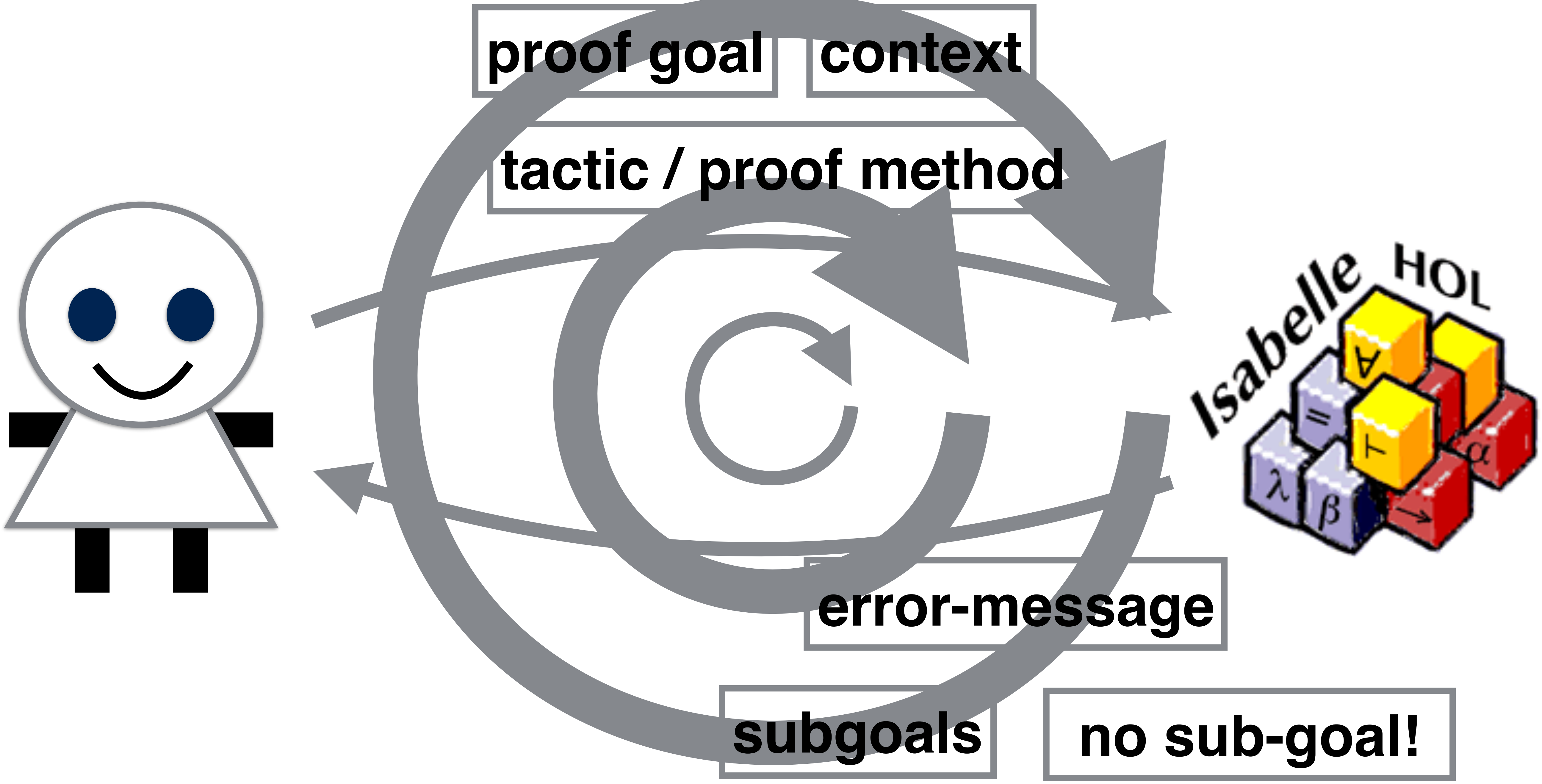
git clone https://github.com/data61/PSL

Interactive theorem proving with Isabelle/HOL



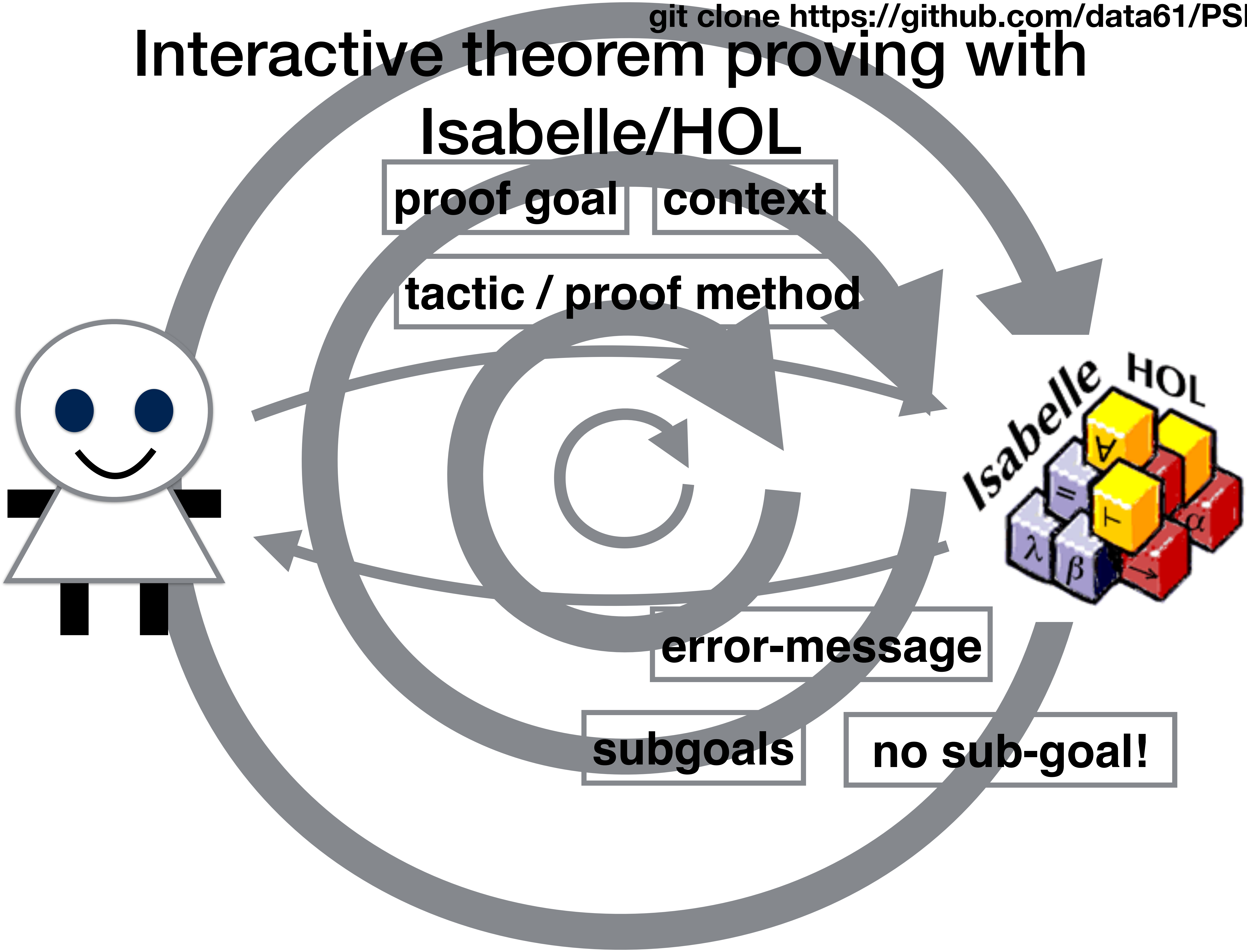
git clone https://github.com/data61/PSL

Interactive theorem proving with Isabelle/HOL



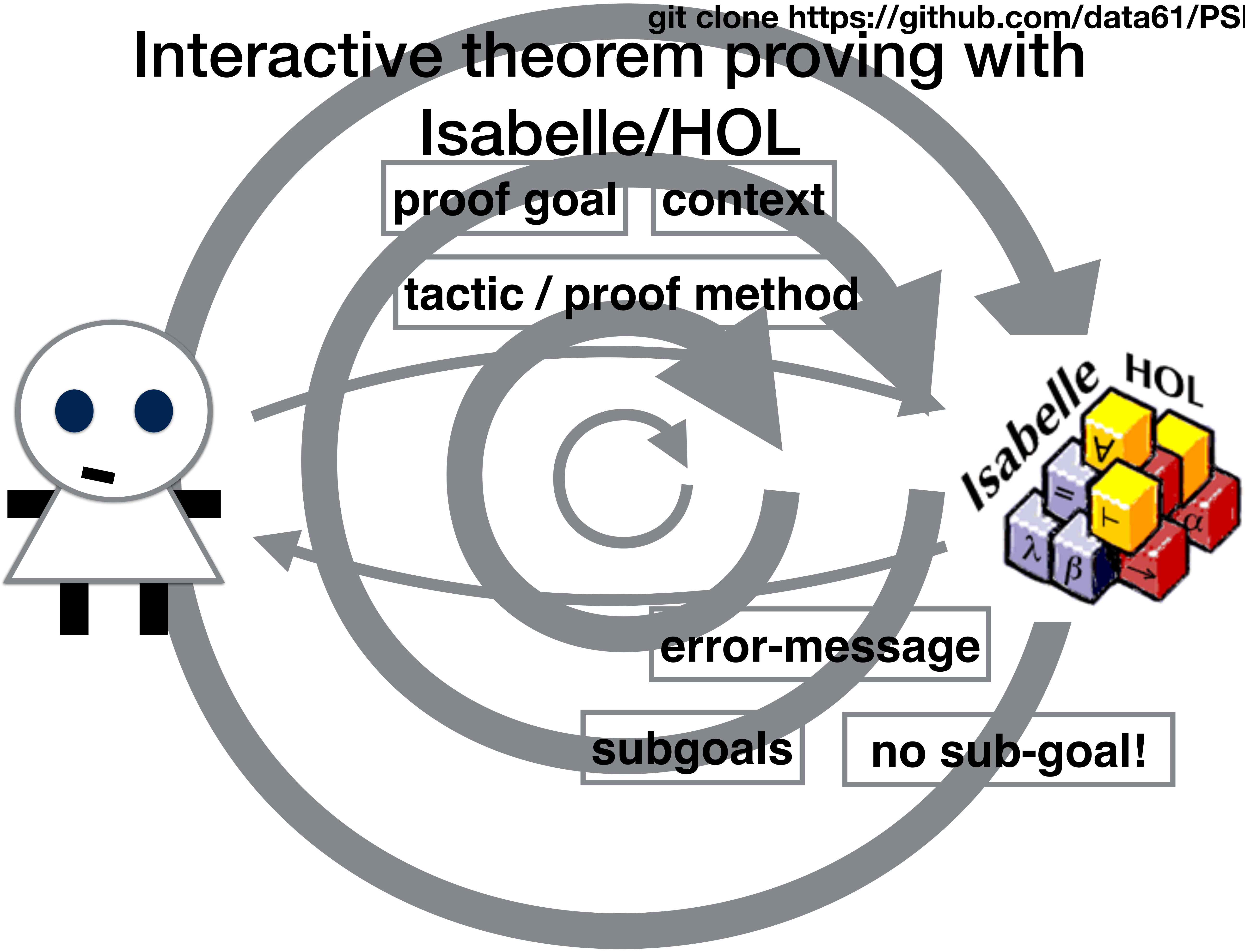
git clone https://github.com/data61/PSL

Interactive theorem proving with Isabelle/HOL



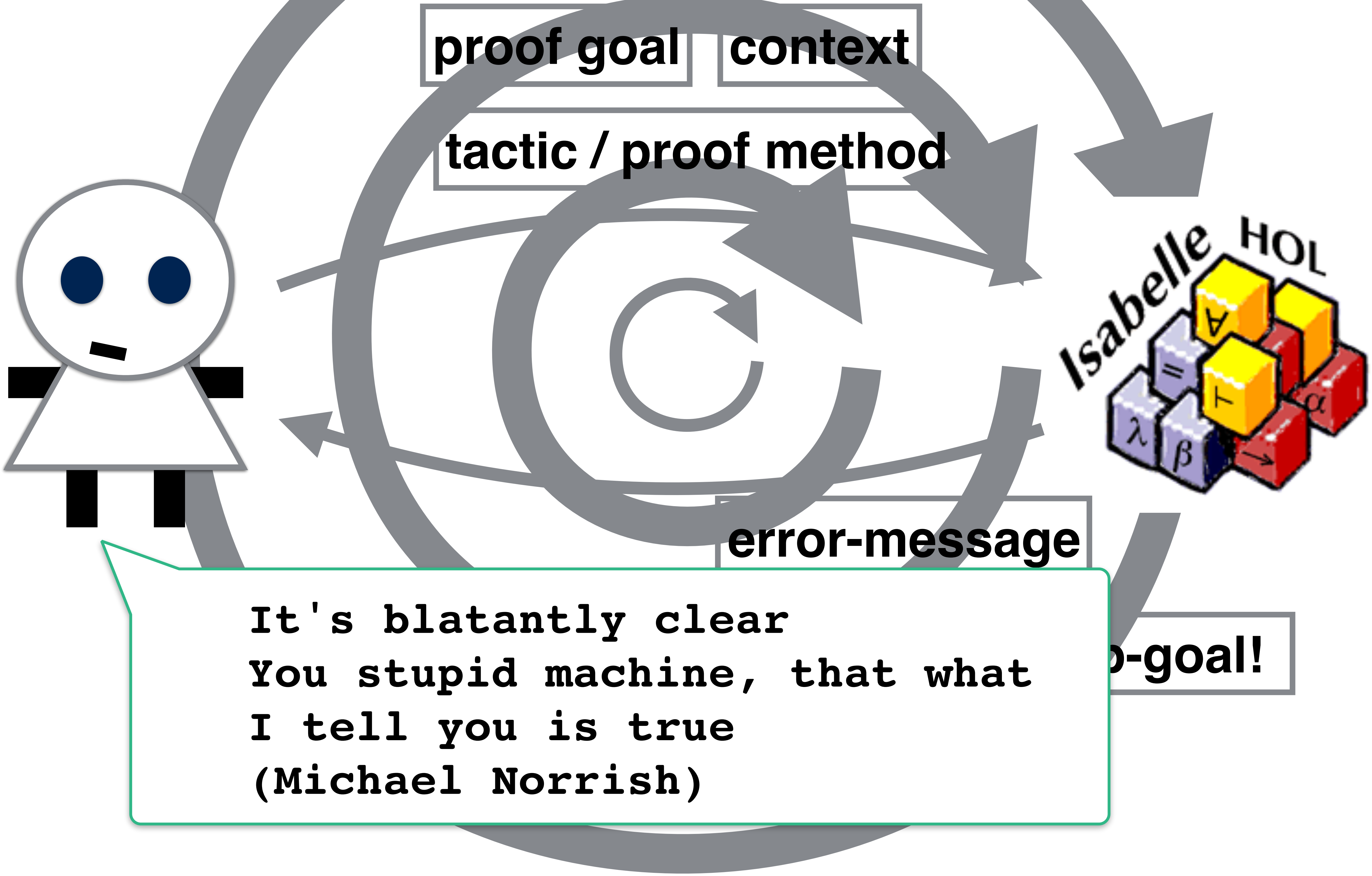
git clone https://github.com/data61/PSL

Interactive theorem proving with Isabelle/HOL

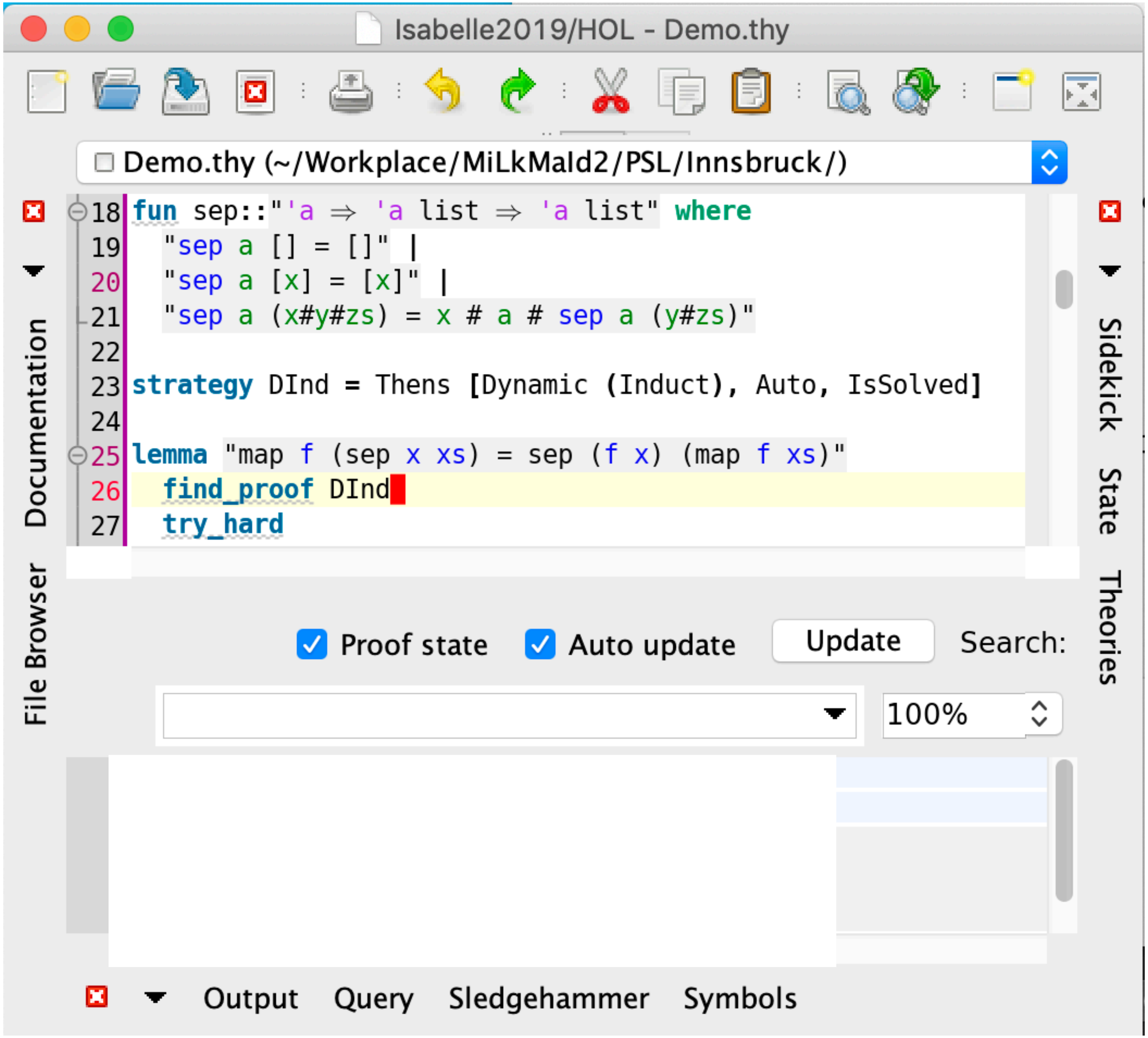


git clone https://github.com/data61/PSL

Interactive theorem proving with Isabelle/HOL

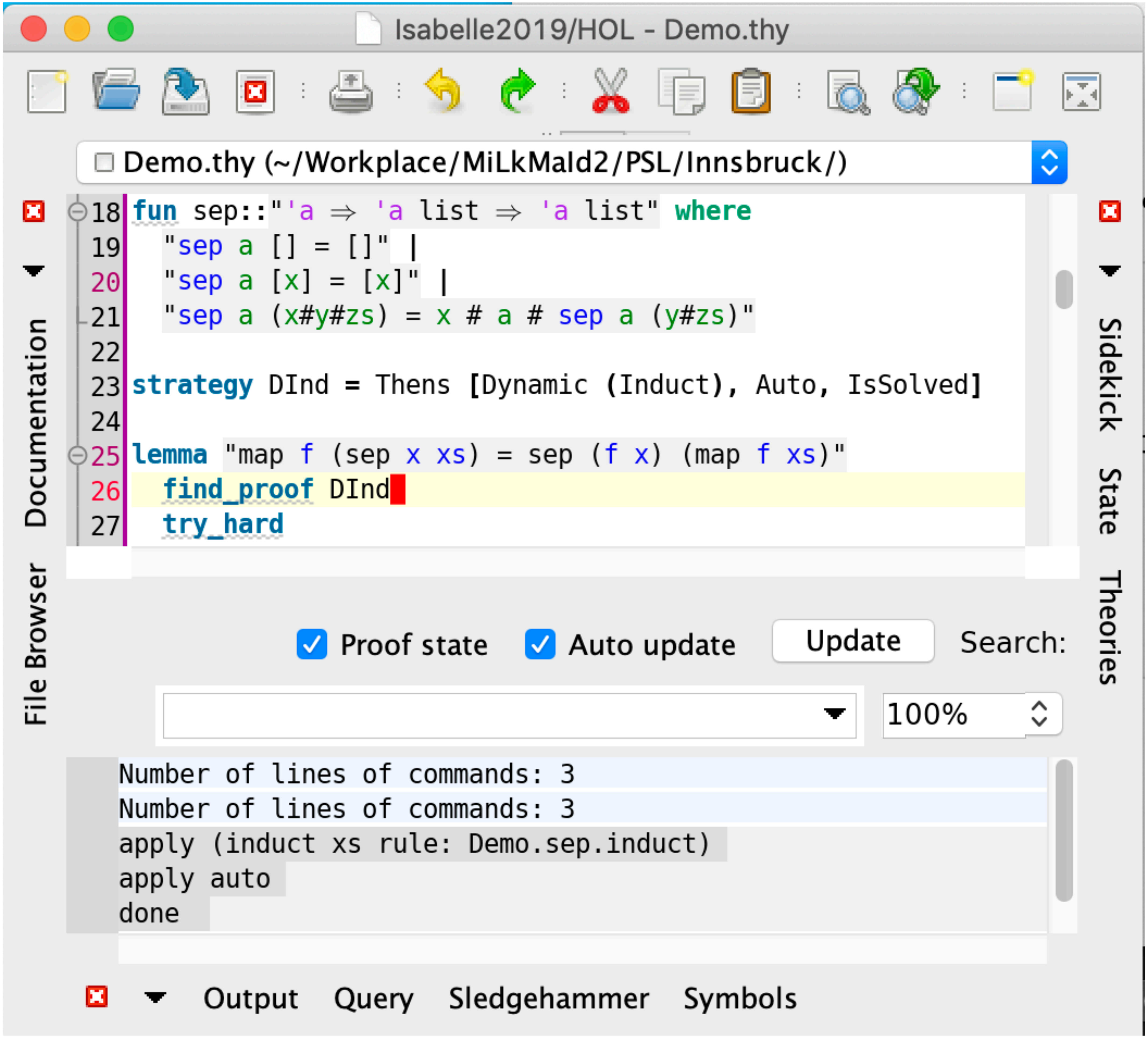

proof goal
context
tactic / proof method
Isabelle HOL
error-message
-goal!
It's blatantly clear You stupid machine, that what I tell you is true (Michael Norrish)

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



```
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
```

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

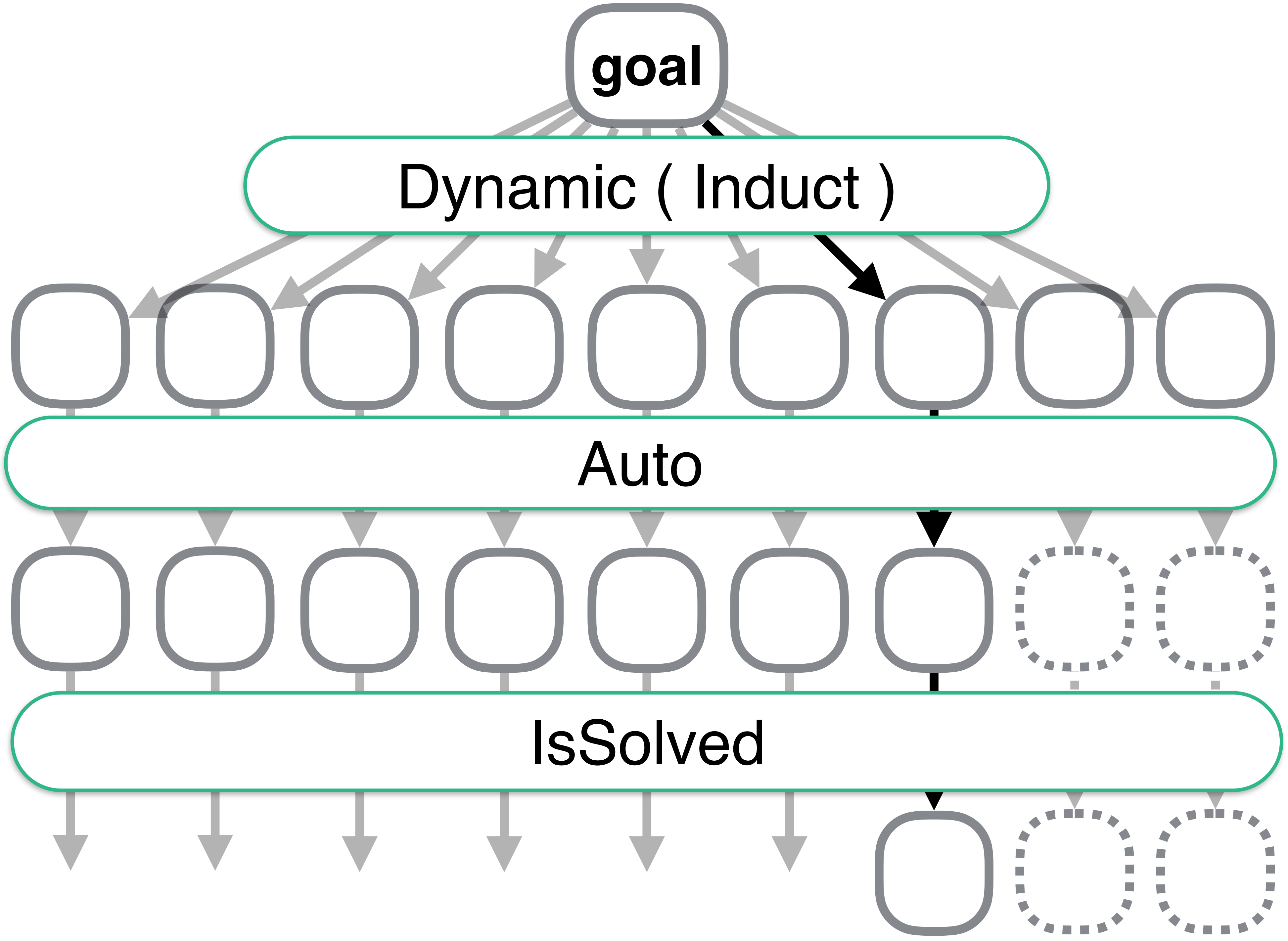


```
fun sep::"'a ⇒ 'a list ⇒ 'a list" where
  "sep a [] = []" |
  "sep a [x] = [x]" |
  "sep a (x#y#zs) = x # a # sep a (y#zs)"

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
```

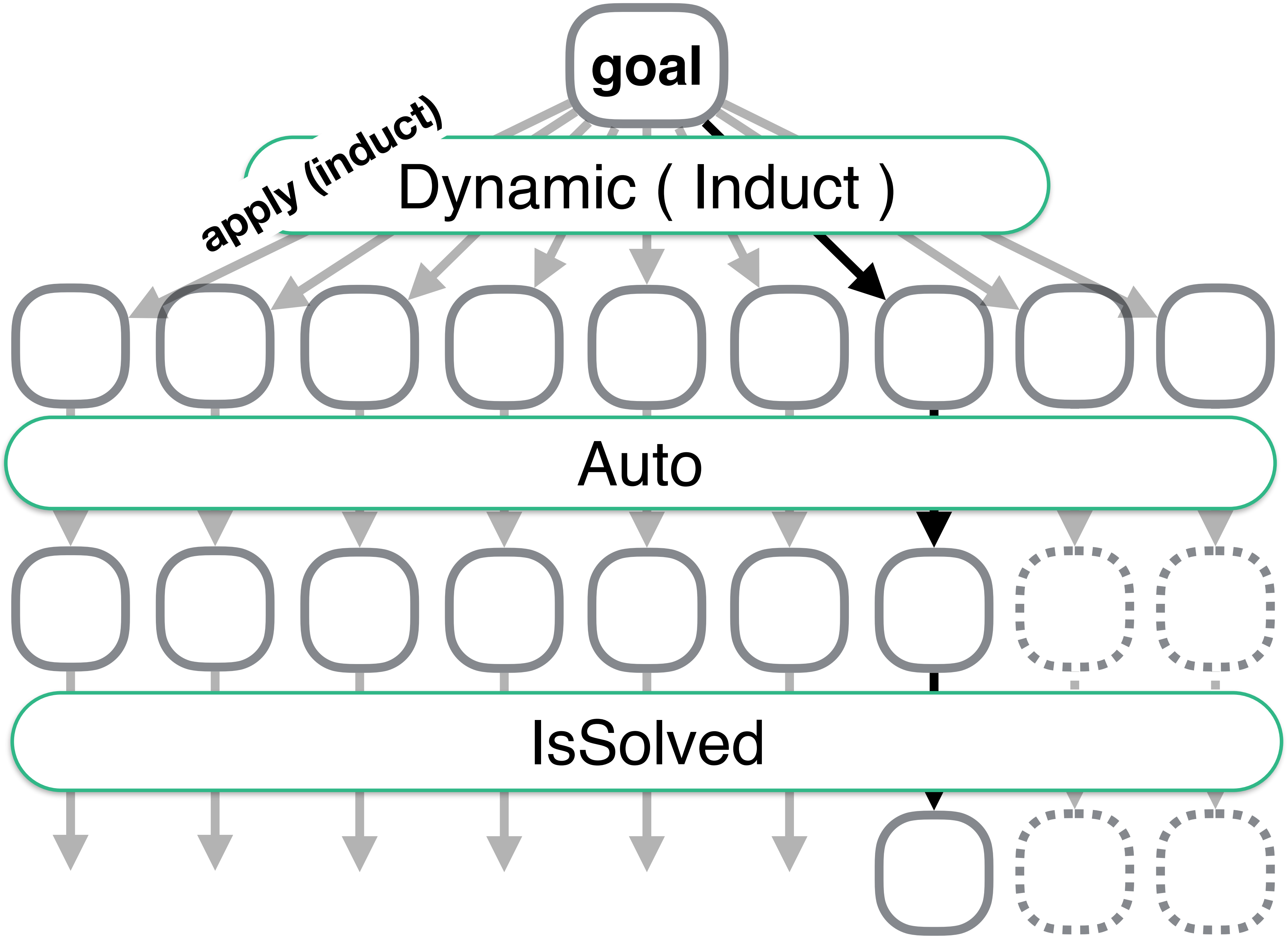
git clone https://github.com/data61/PSL

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



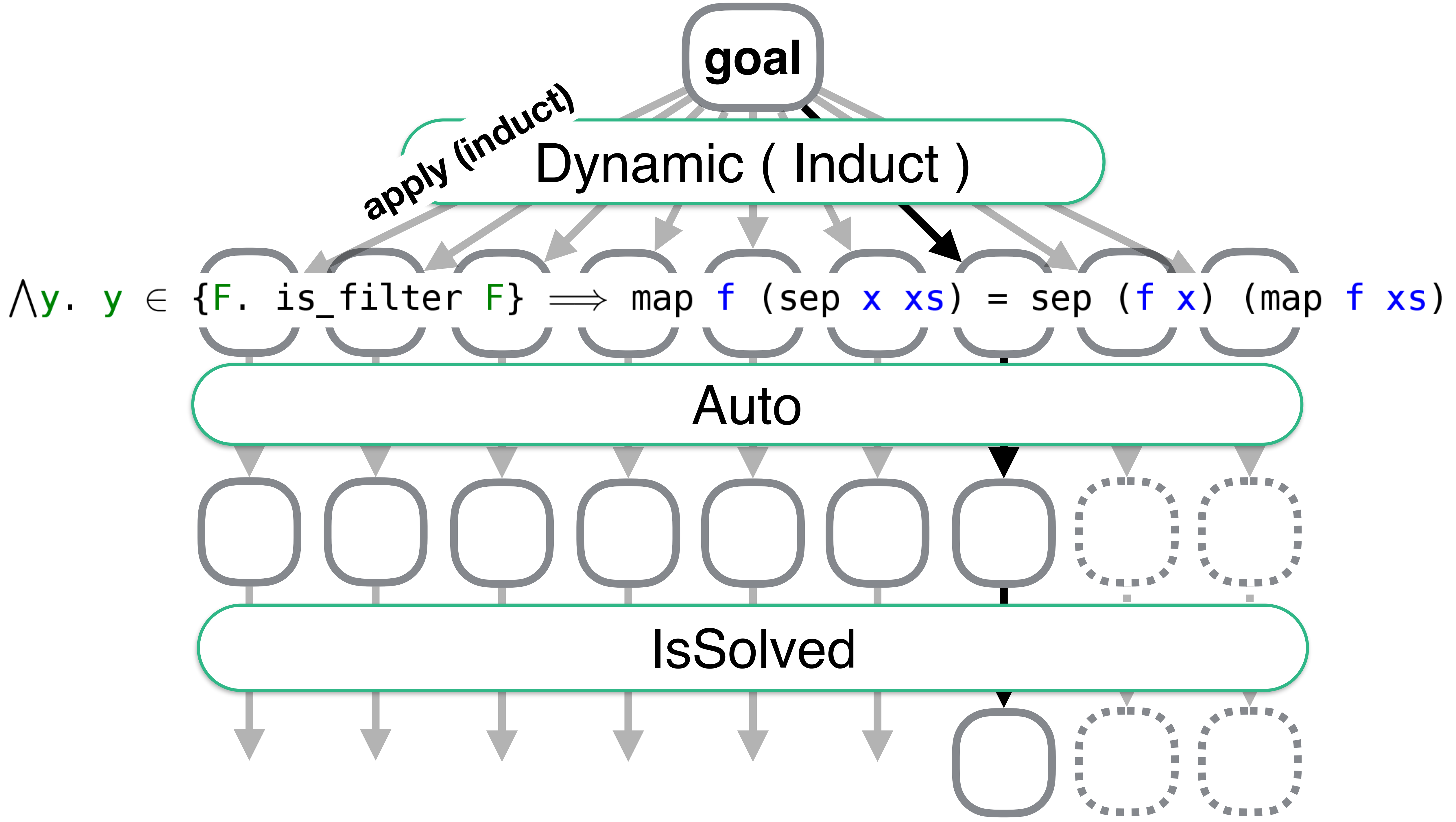
git clone https://github.com/data61/PSL

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



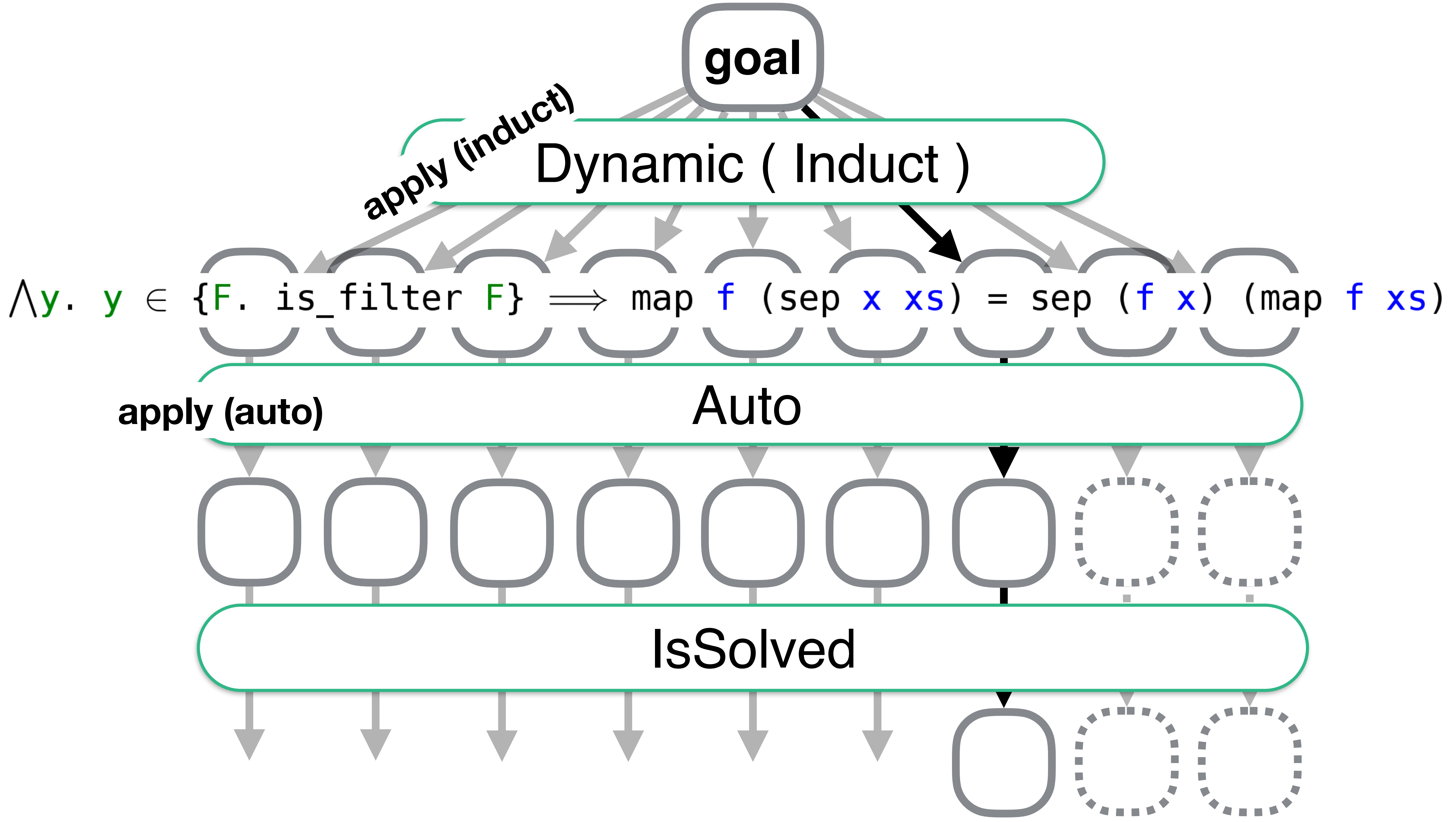
git clone https://github.com/data61/PSL

```
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)
Auto
IsSolved

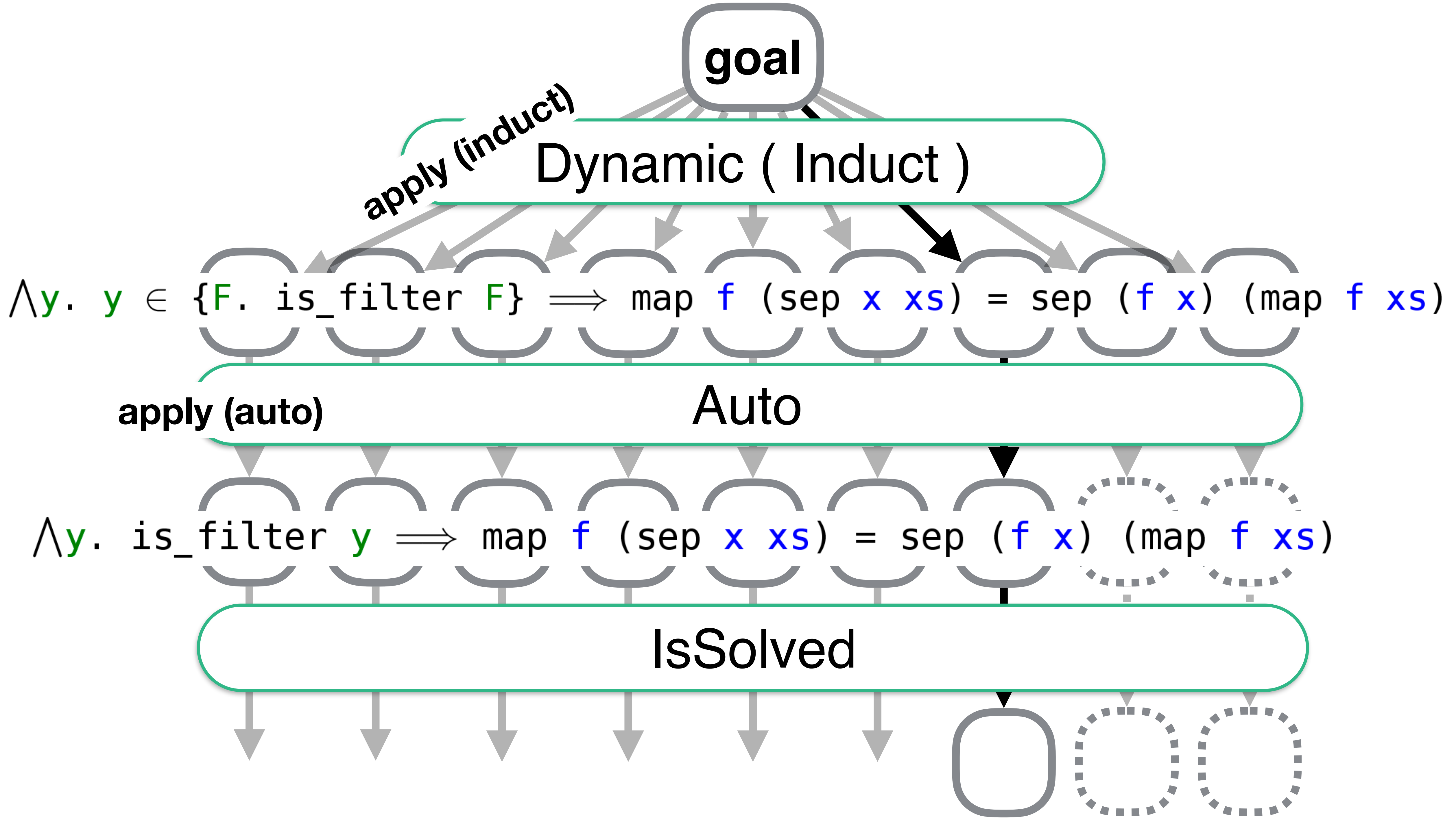
git clone https://github.com/data61/PSL

```
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
IsSolved

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

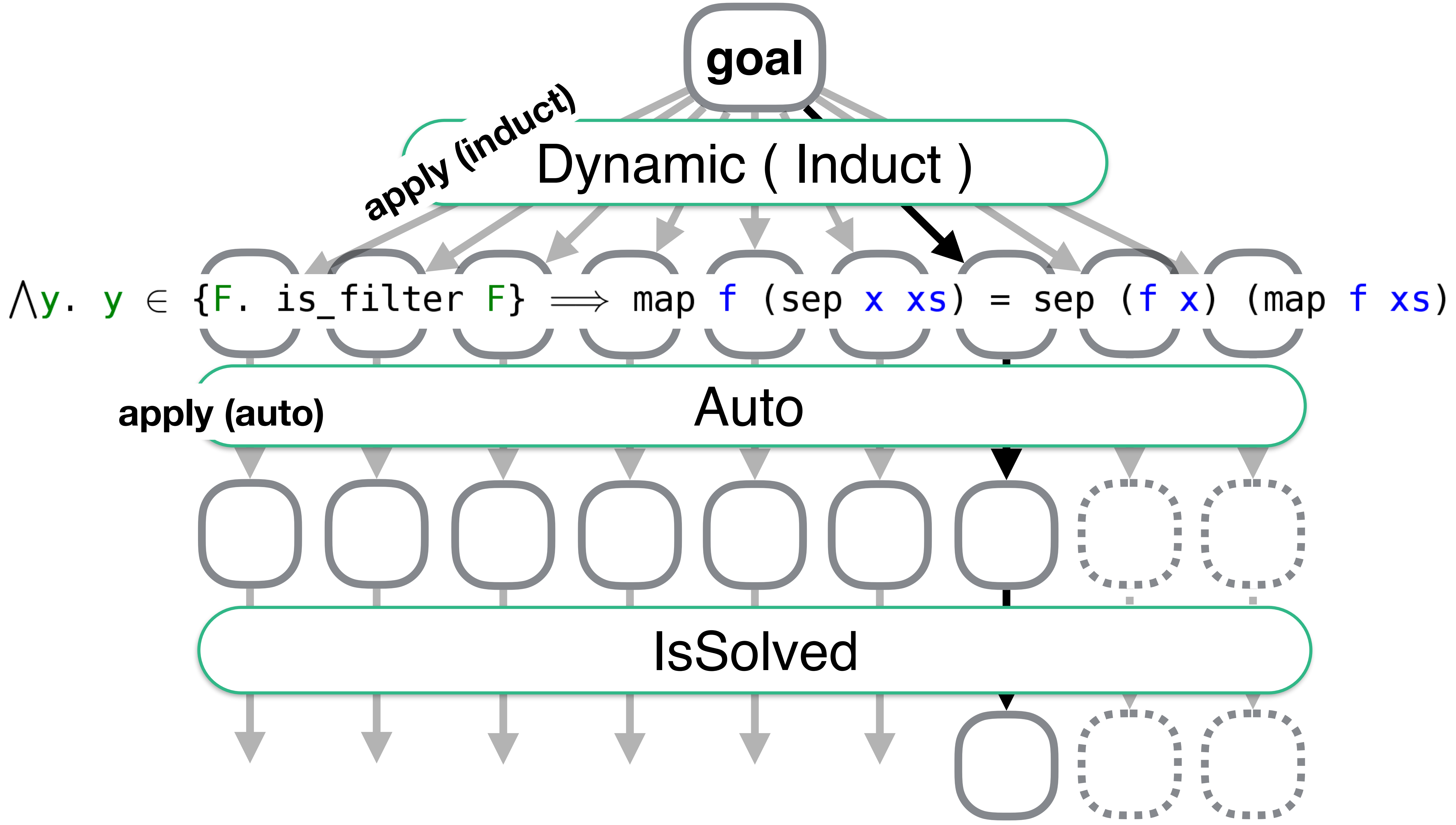
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"

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

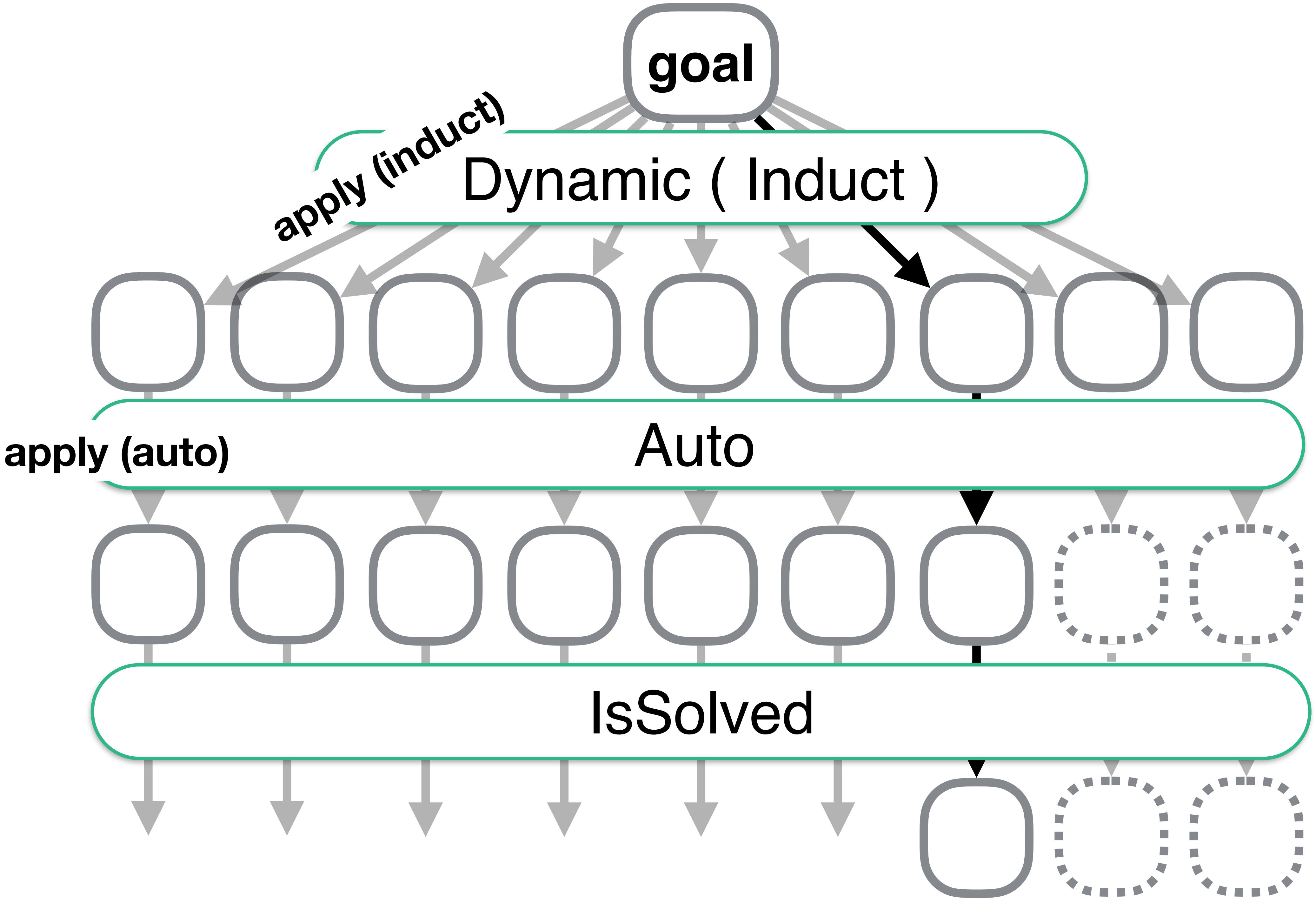
git clone https://github.com/data61/PSL

```
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
IsSolved

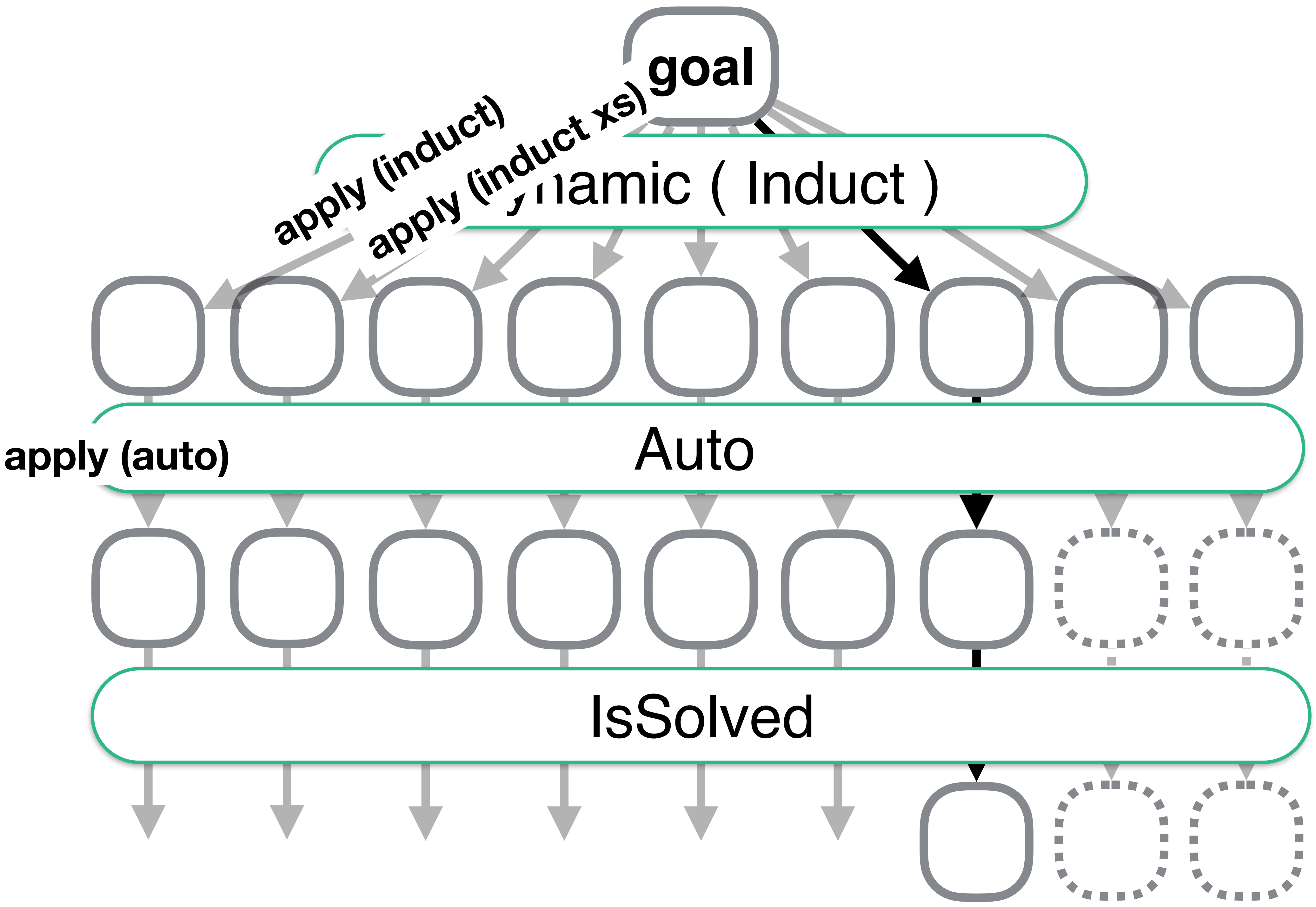
git clone https://github.com/data61/PSL

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



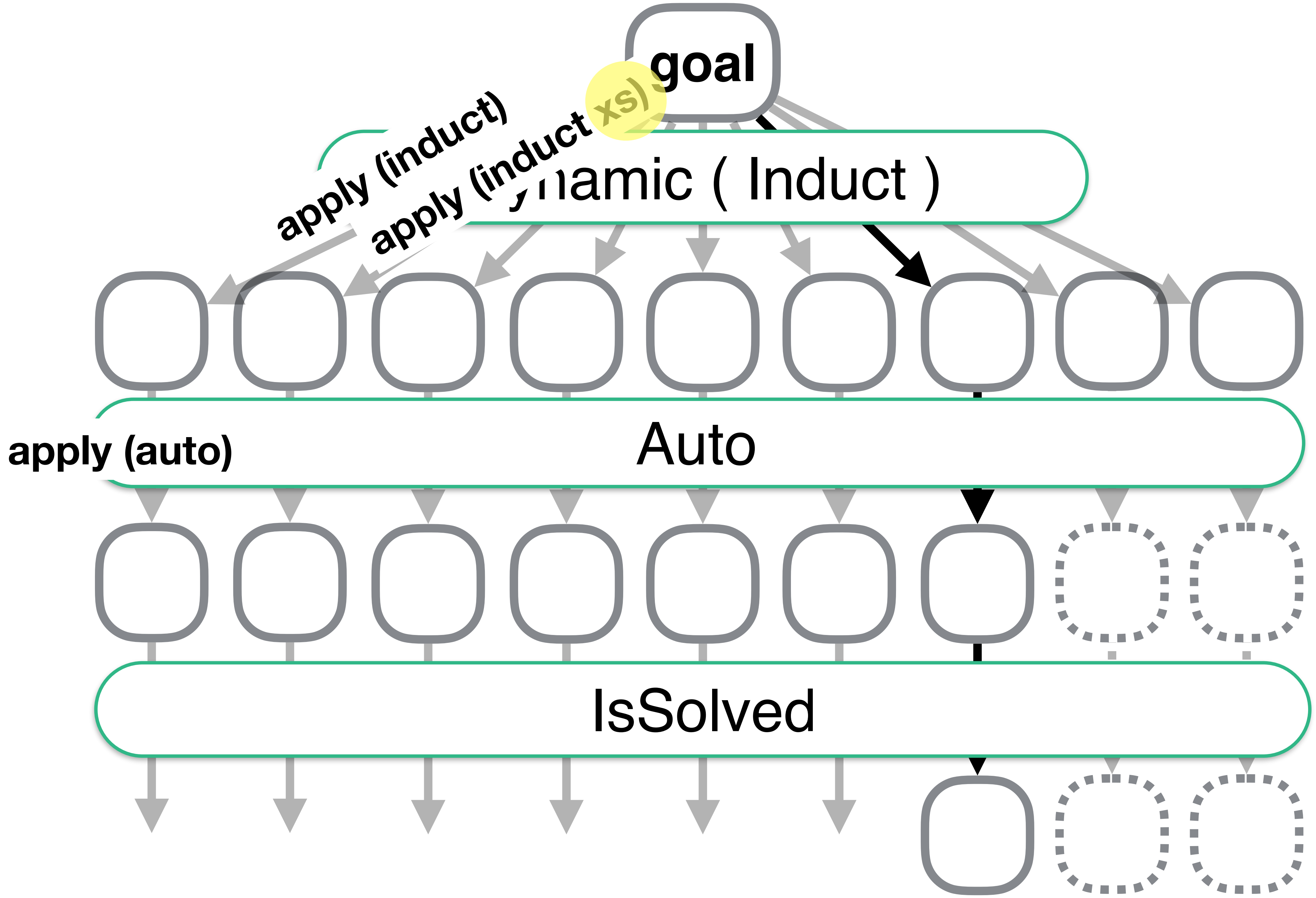
git clone https://github.com/data61/PSL

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

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



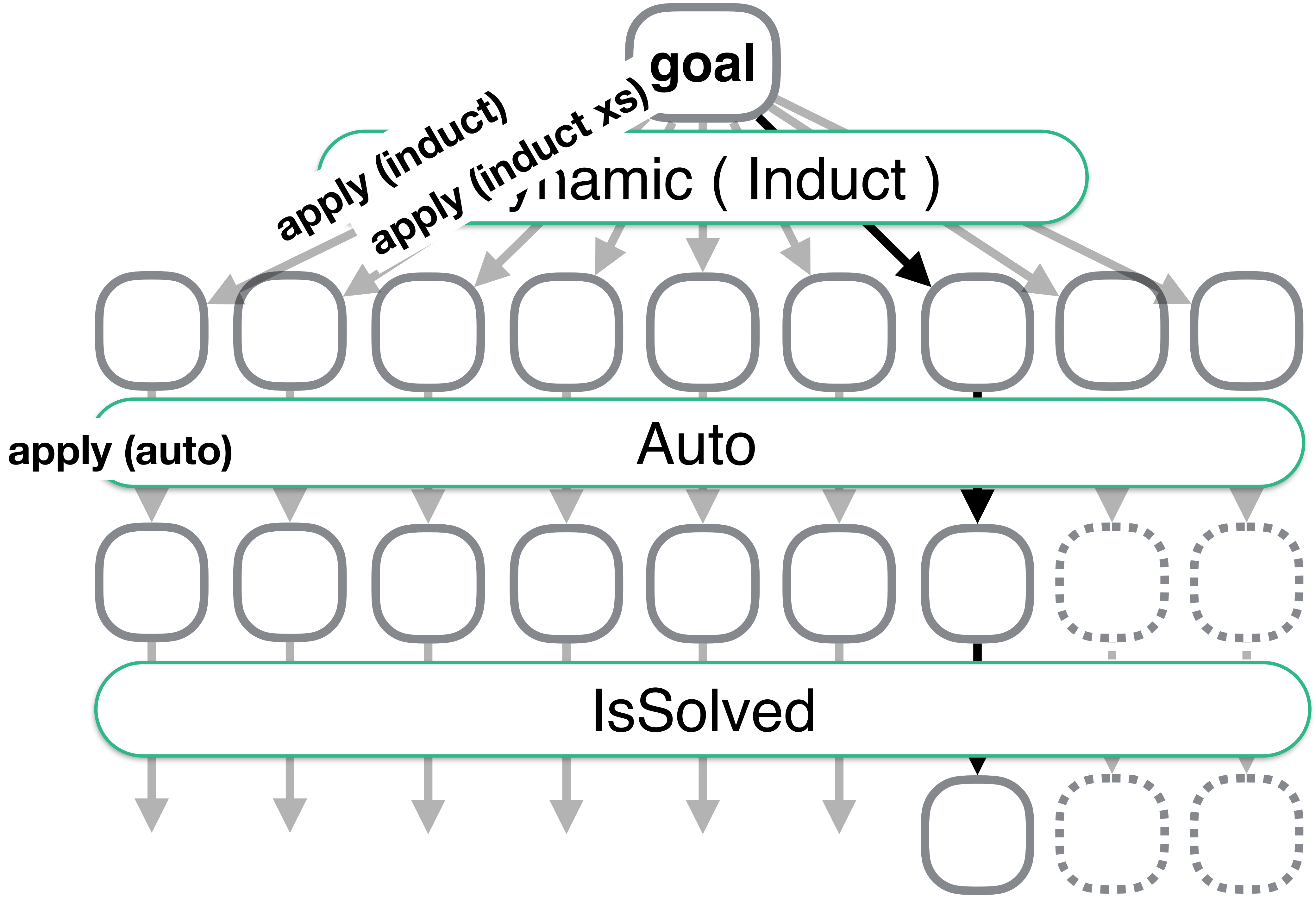
git clone https://github.com/data61/PSL

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



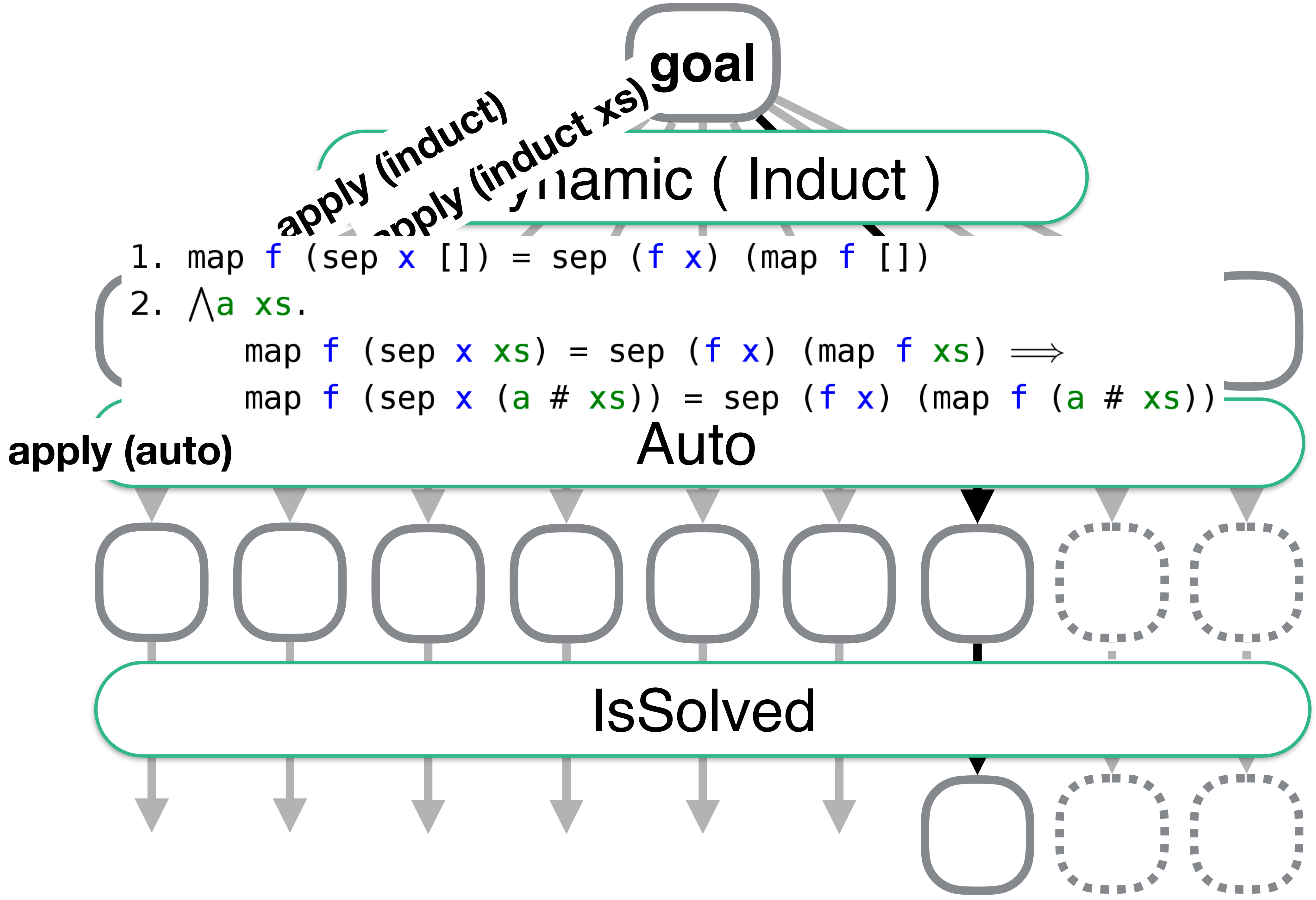
git clone https://github.com/data61/PSL

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



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

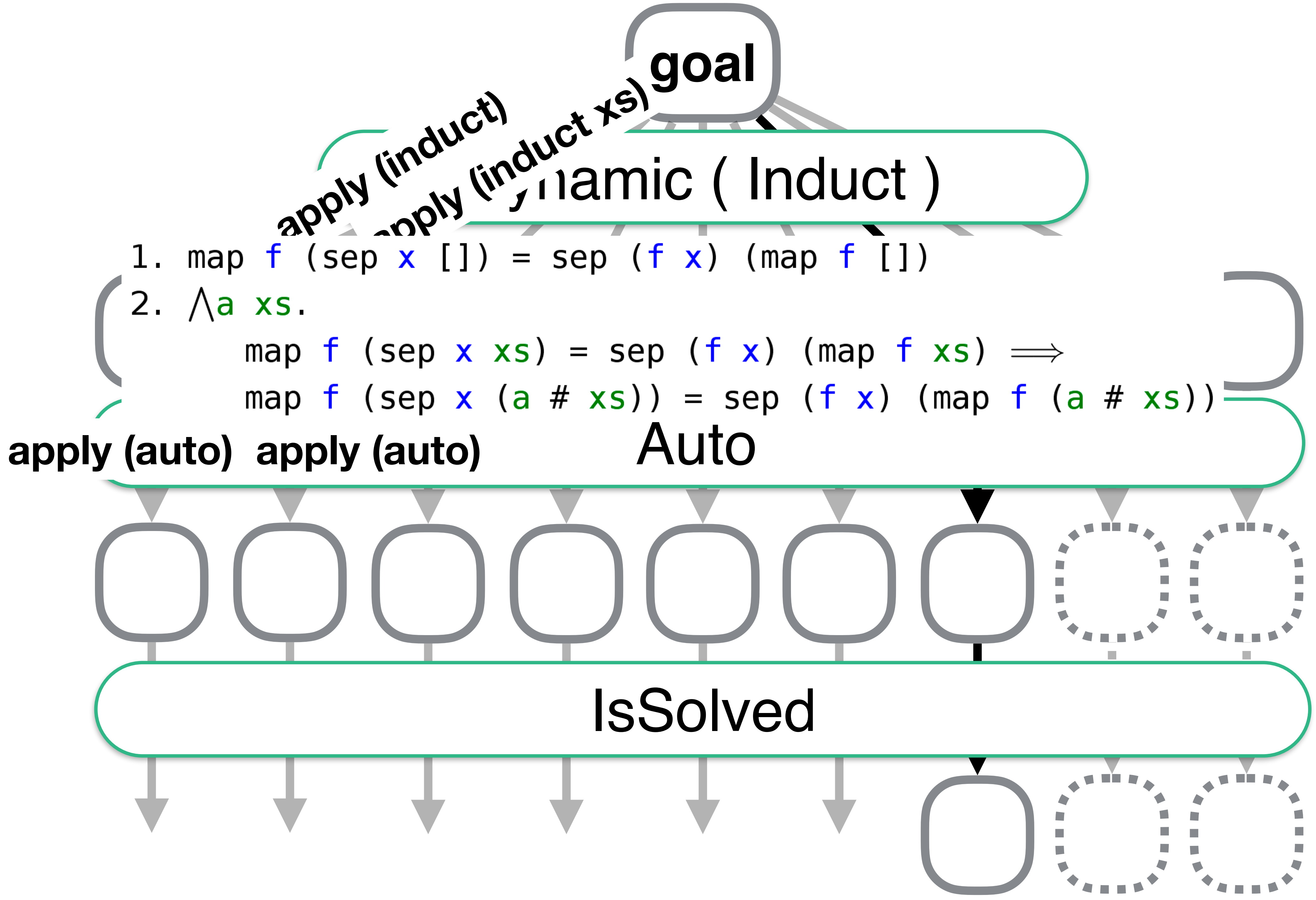
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



```
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))
```

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

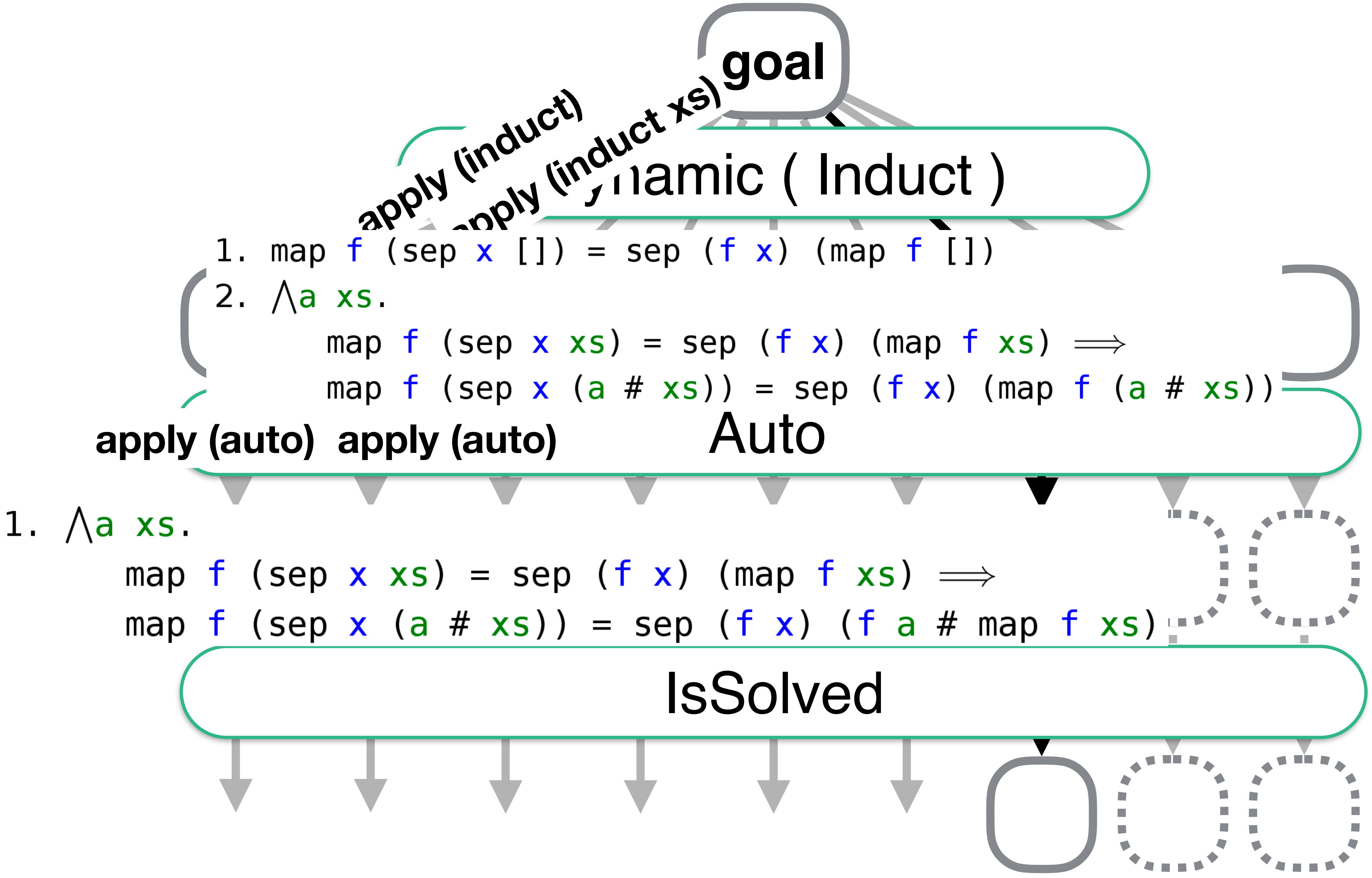
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



```
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))
```

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

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

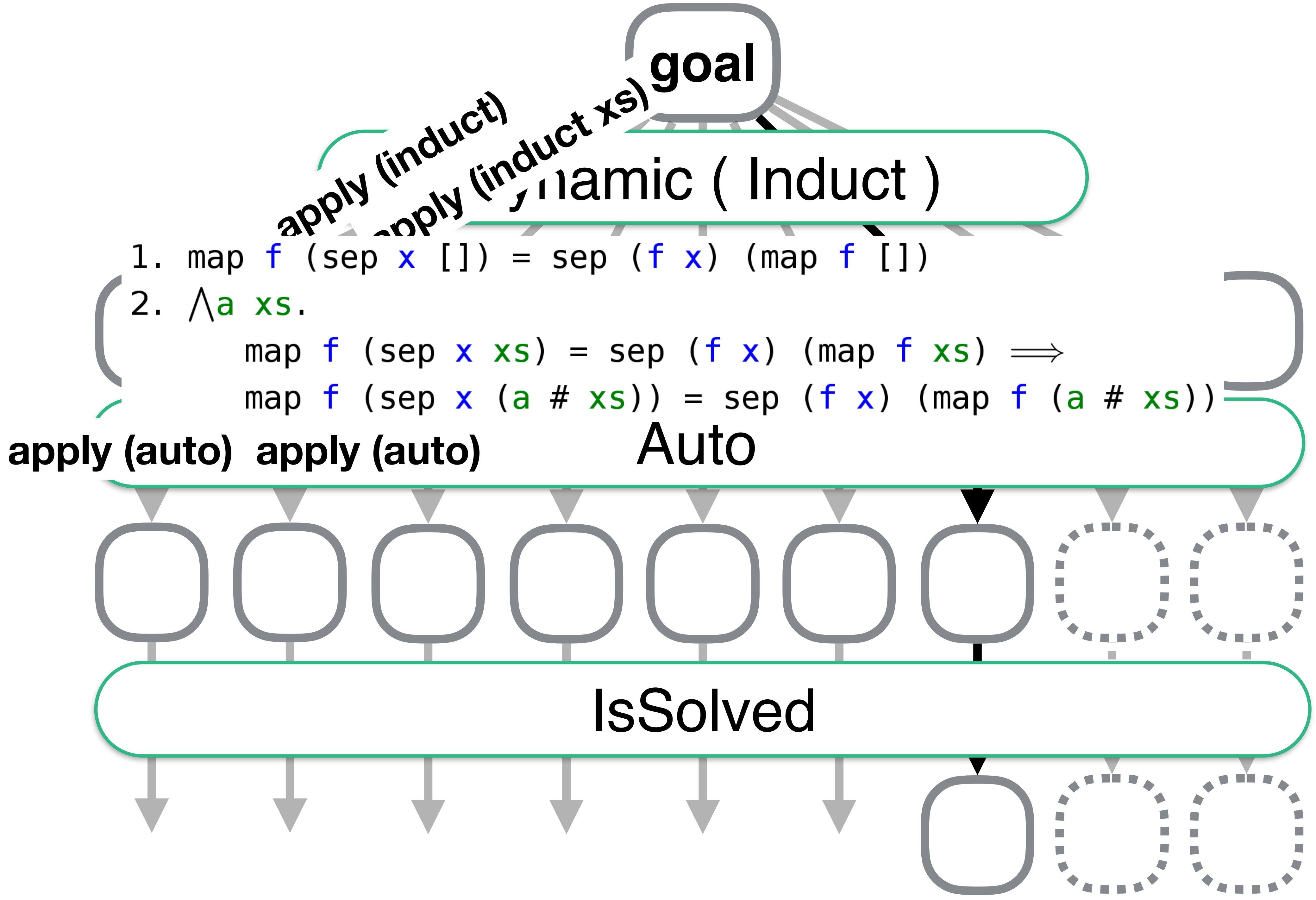


```
1. map f (sep x []) = sep (f x) (map f [])
2. ⋀a xs.
      map f (sep x xs) = sep (f x) (map f xs) ⟹
      map f (sep x (a # xs)) = sep (f x) (map f (a # xs))
```

```
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)
```

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

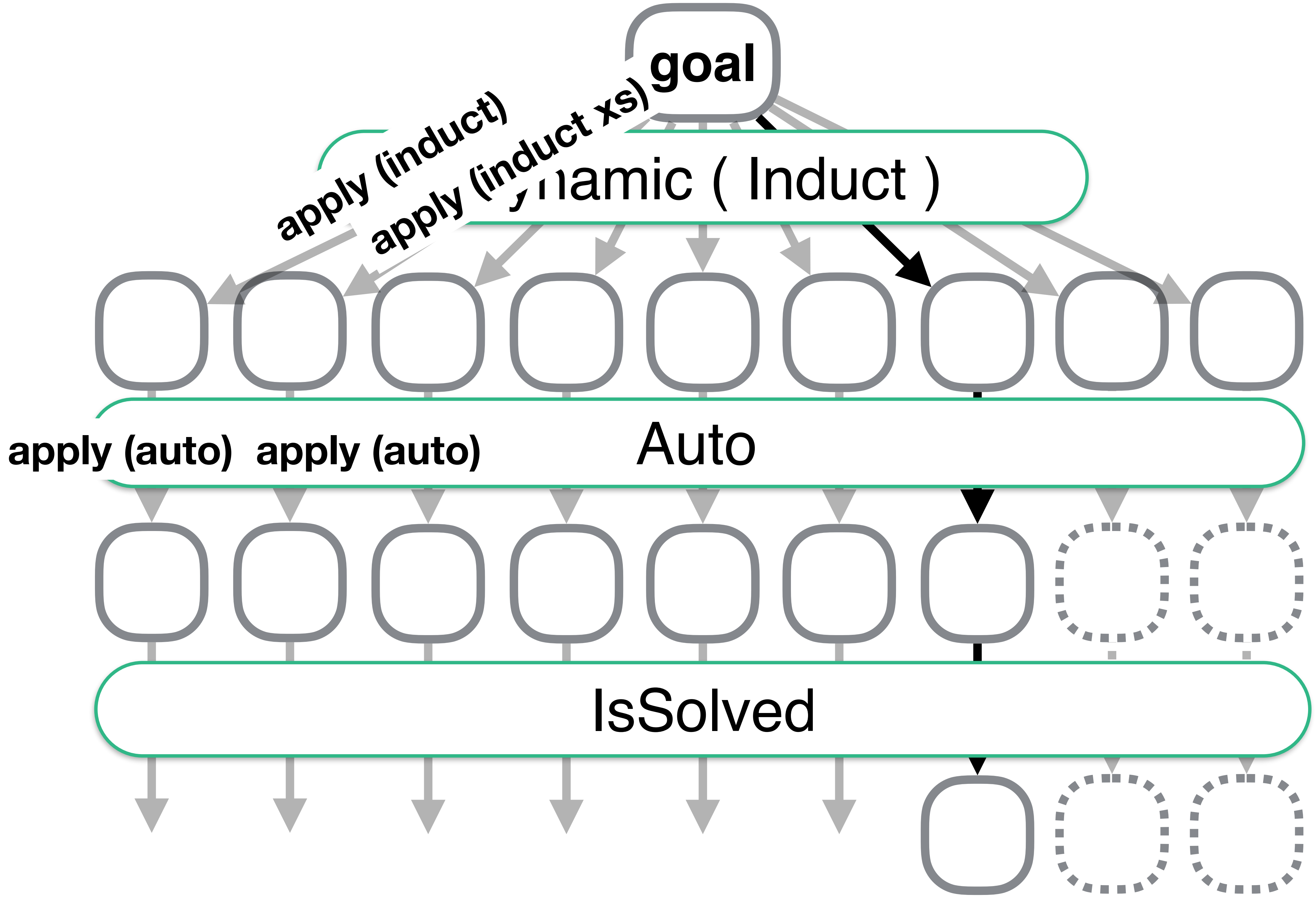
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



```
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))
```

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

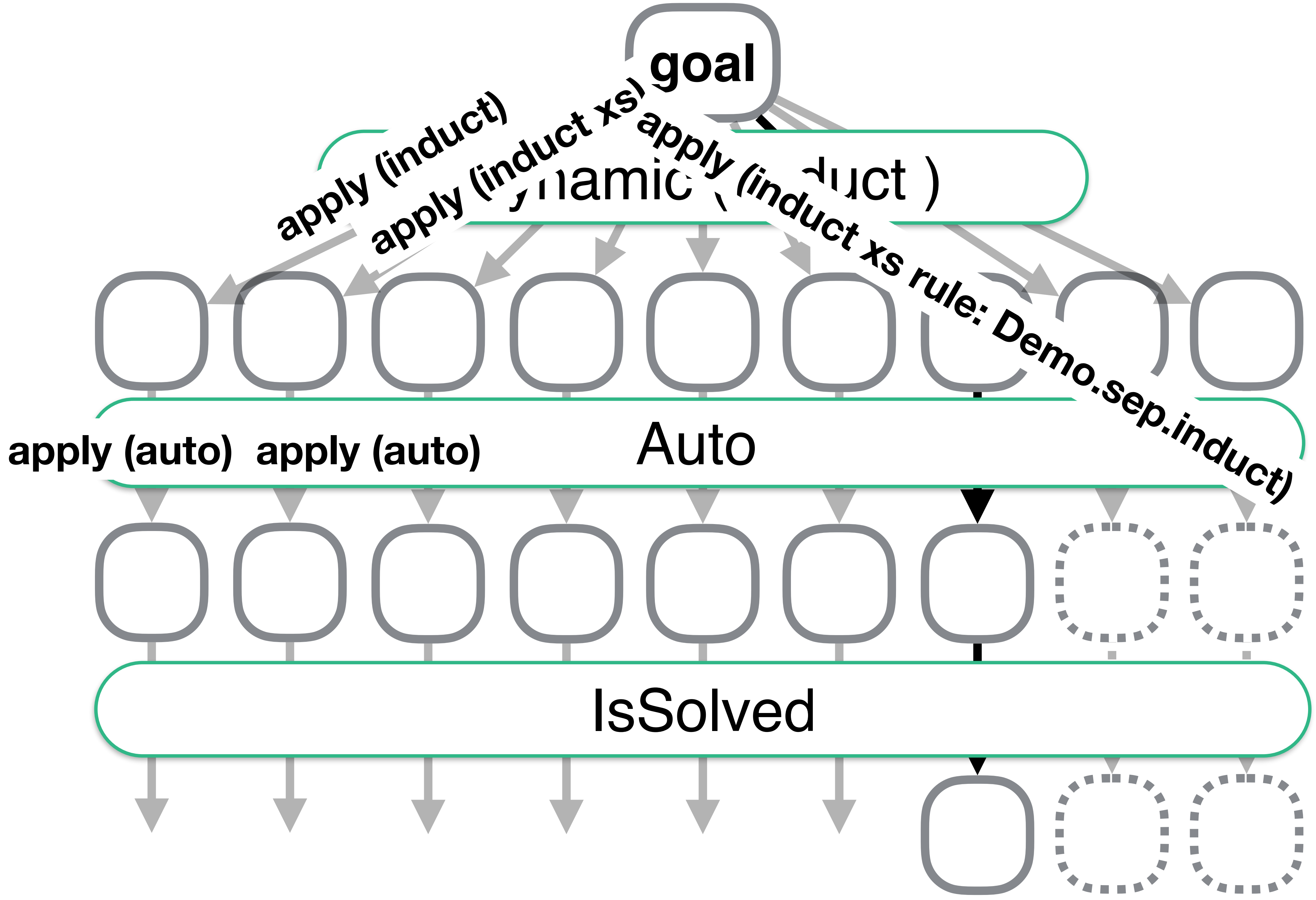
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



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

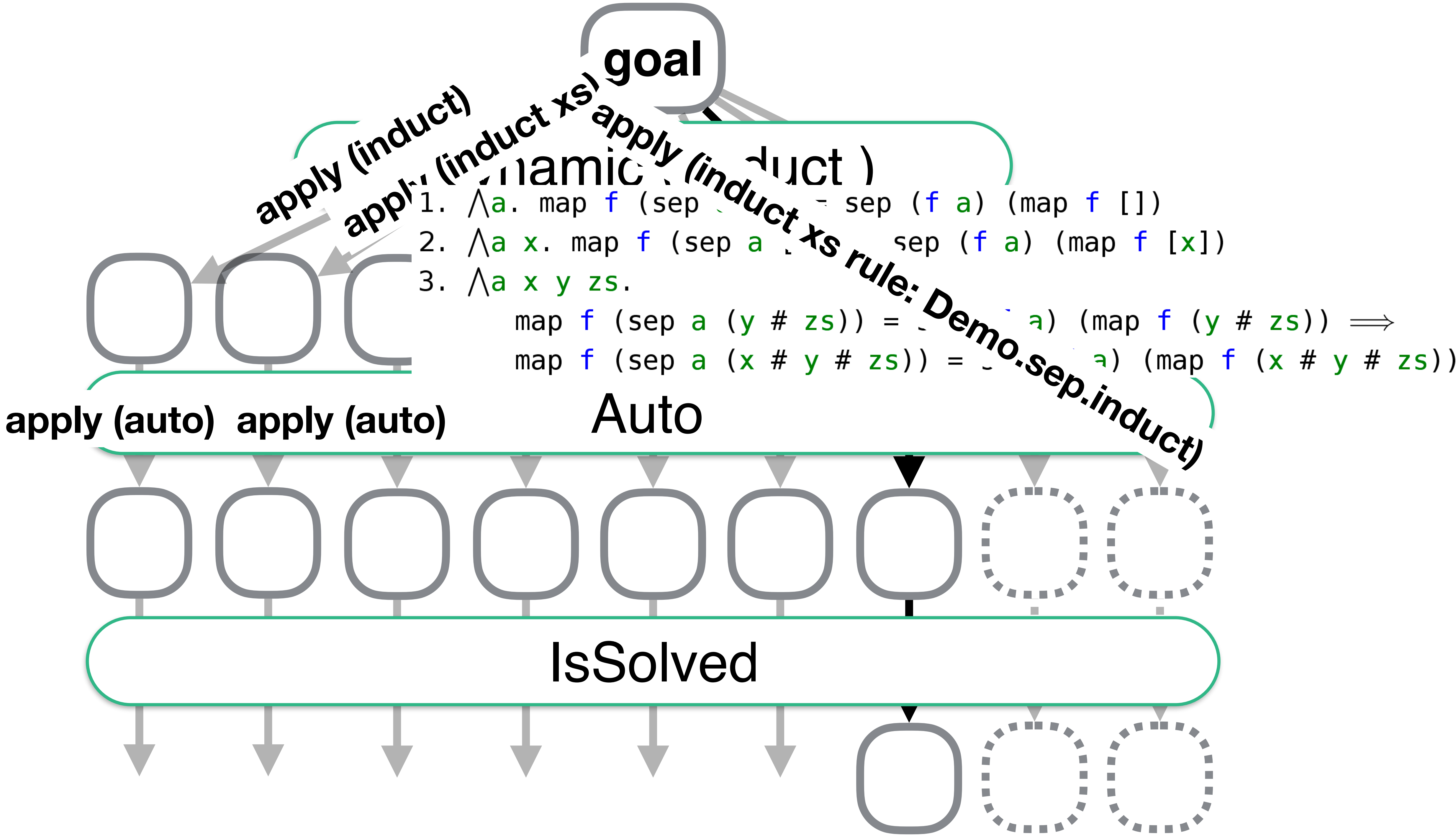
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
```

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



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

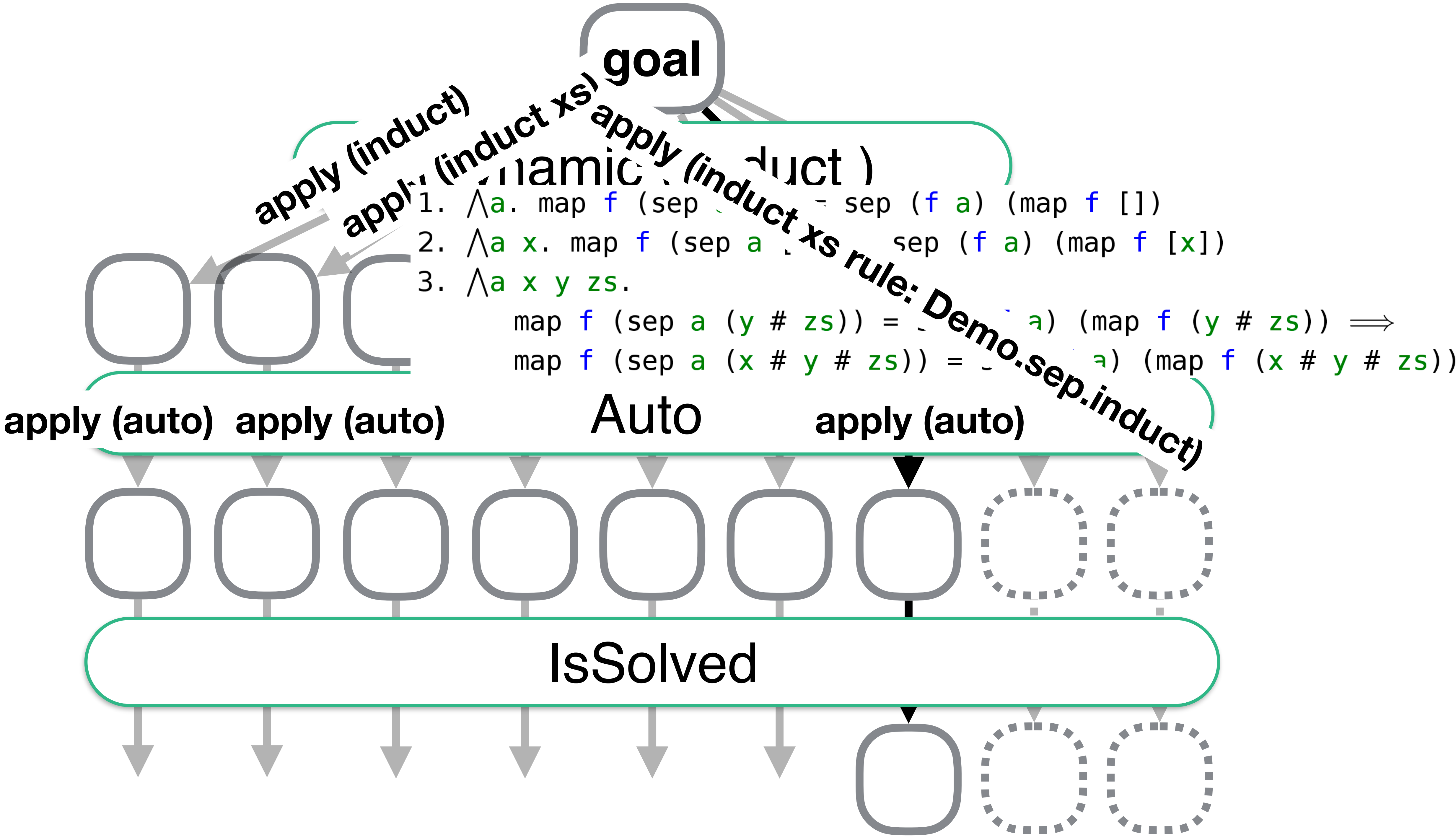
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



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

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

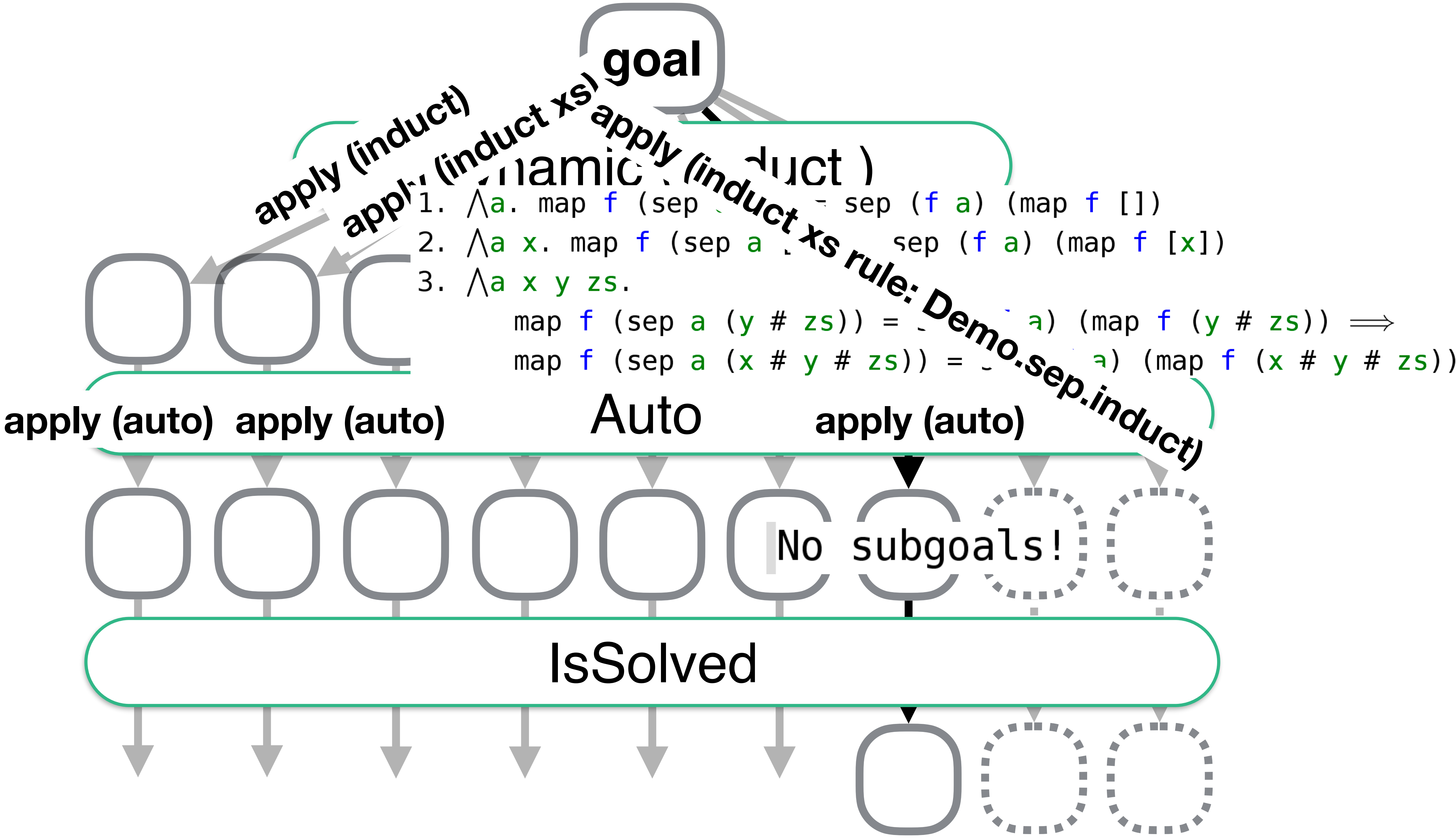
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



```
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))
```

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

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

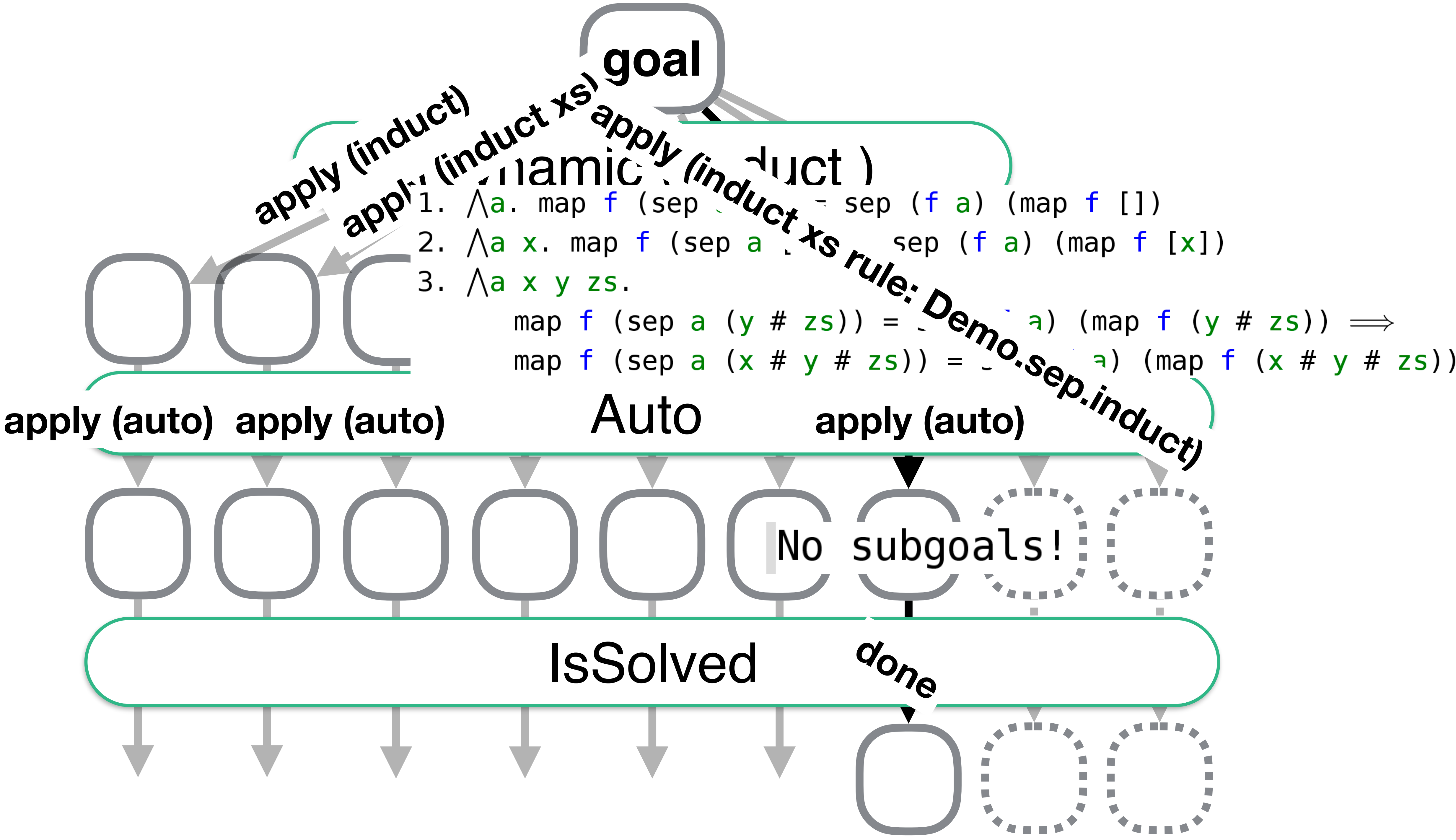


```
1. ⋀a. map f (sep ... = sep (f a) (map f [])
2. ⋀a x. map f (sep a ... sep (f a) (map f [x])
3. ⋀a x y zs.
      map f (sep a (y # zs)) = ... a) (map f (y # zs)) ⟹
      map f (sep a (x # y # zs)) = ... a) (map f (x # y # zs))
```

```
No subgoals!
```

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

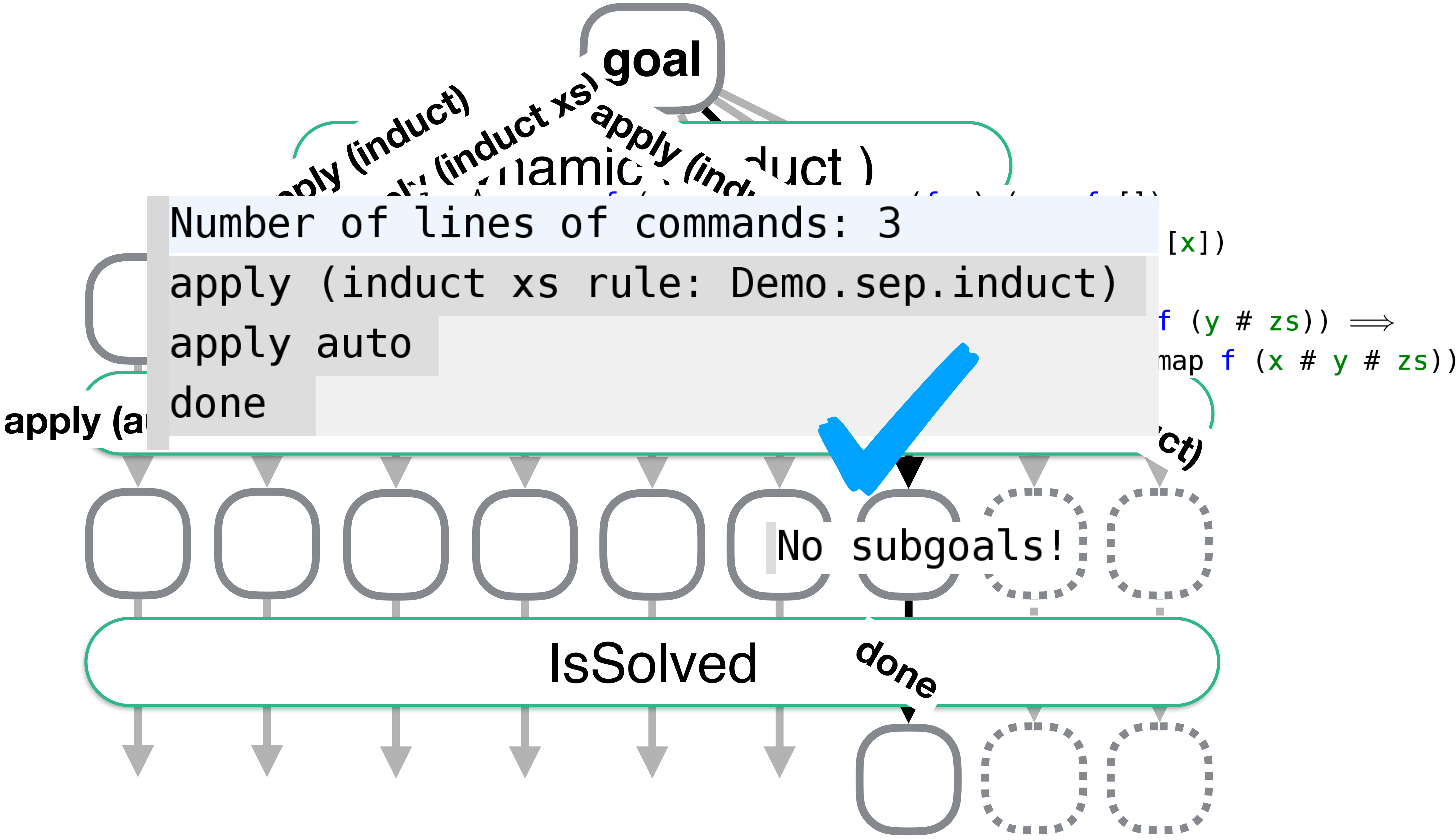
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"
find_proof DInd(*= Thens [Dynamic (Induct), Auto, IsSolved]*)
```



```
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))
```

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

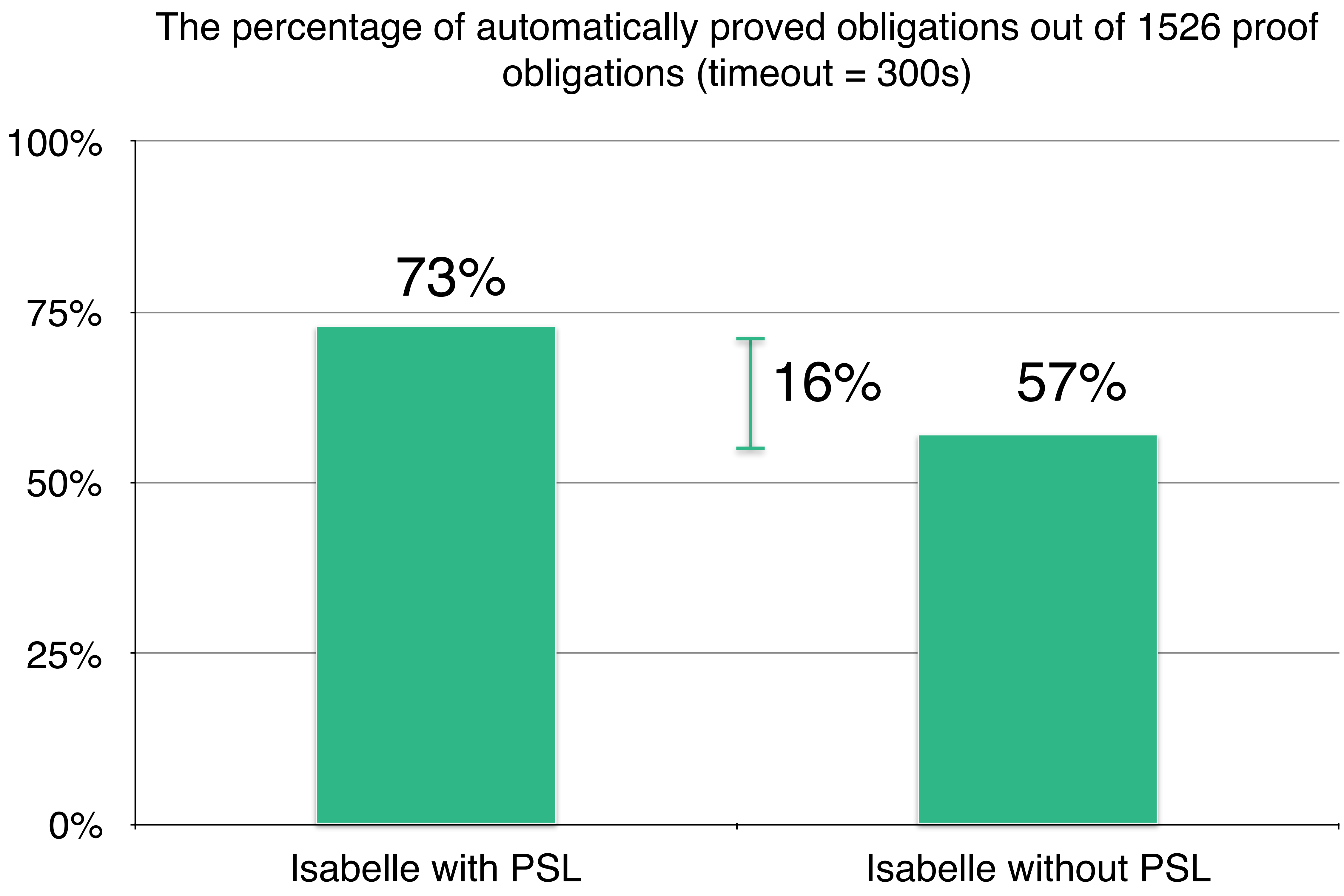
```
lemma "map f (sep x xs) = sep (f x) (map f xs)"

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



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

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

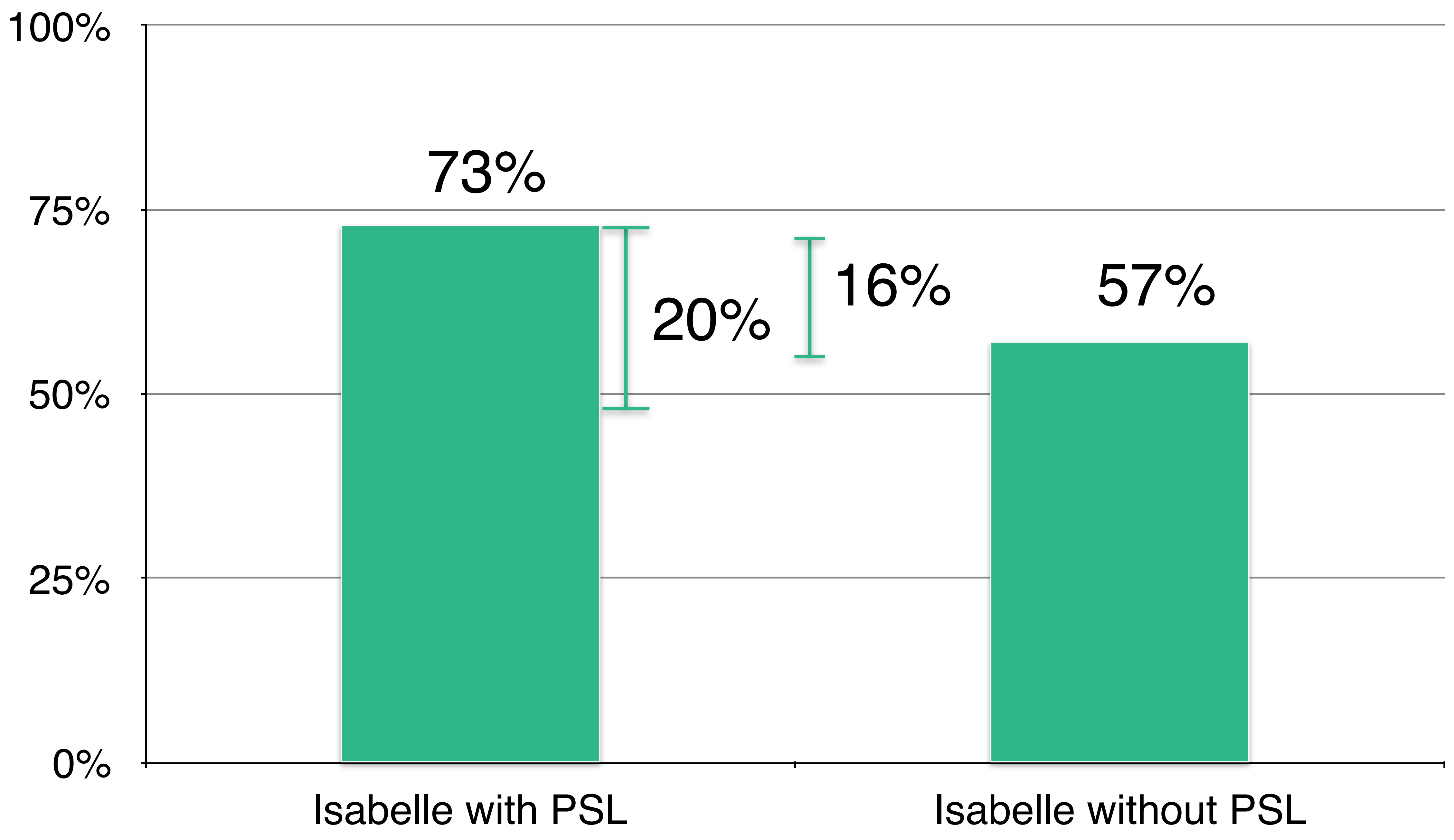
Evaluation



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

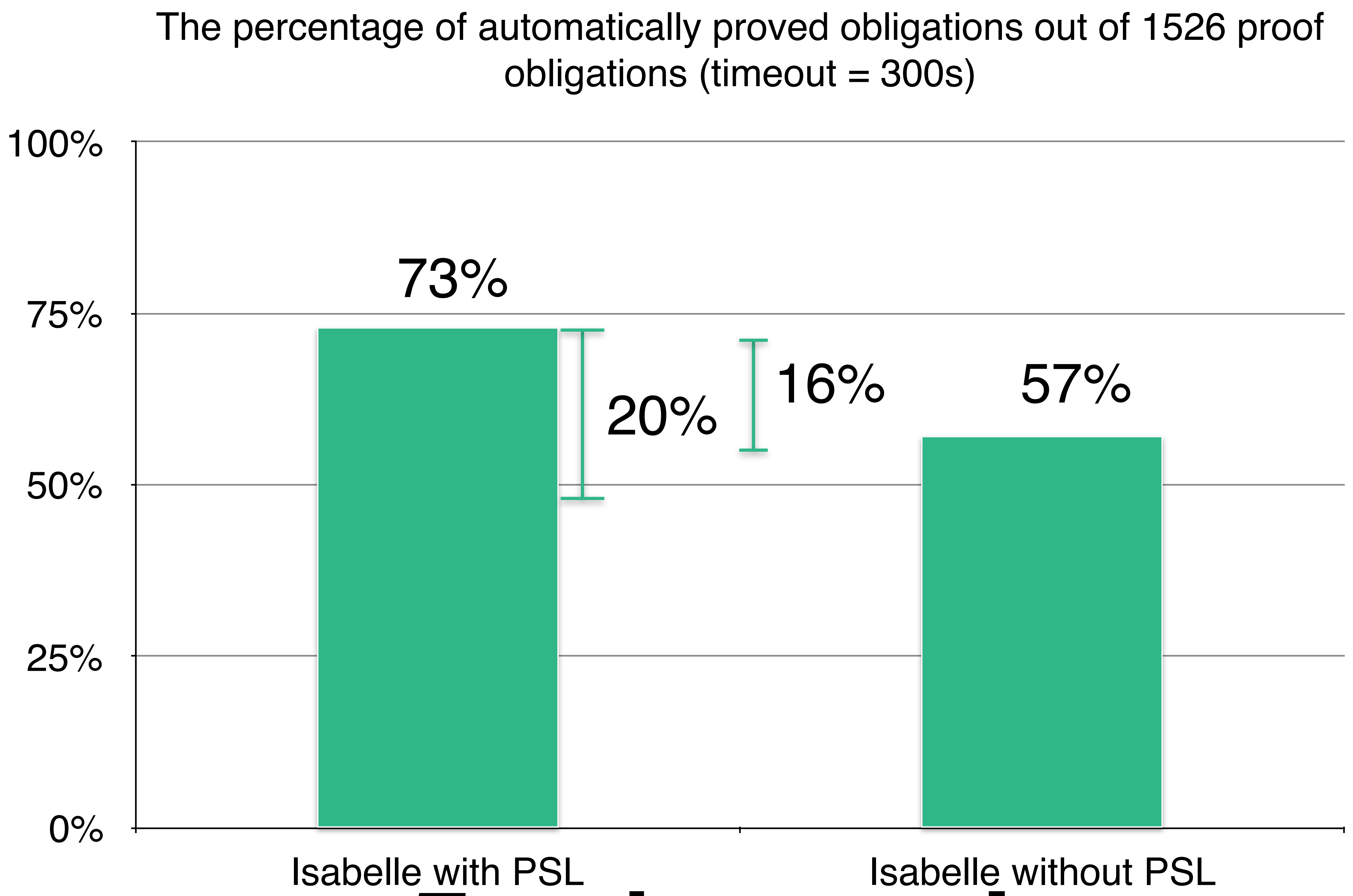
Evaluation

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



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

Evaluation



5 minutes!