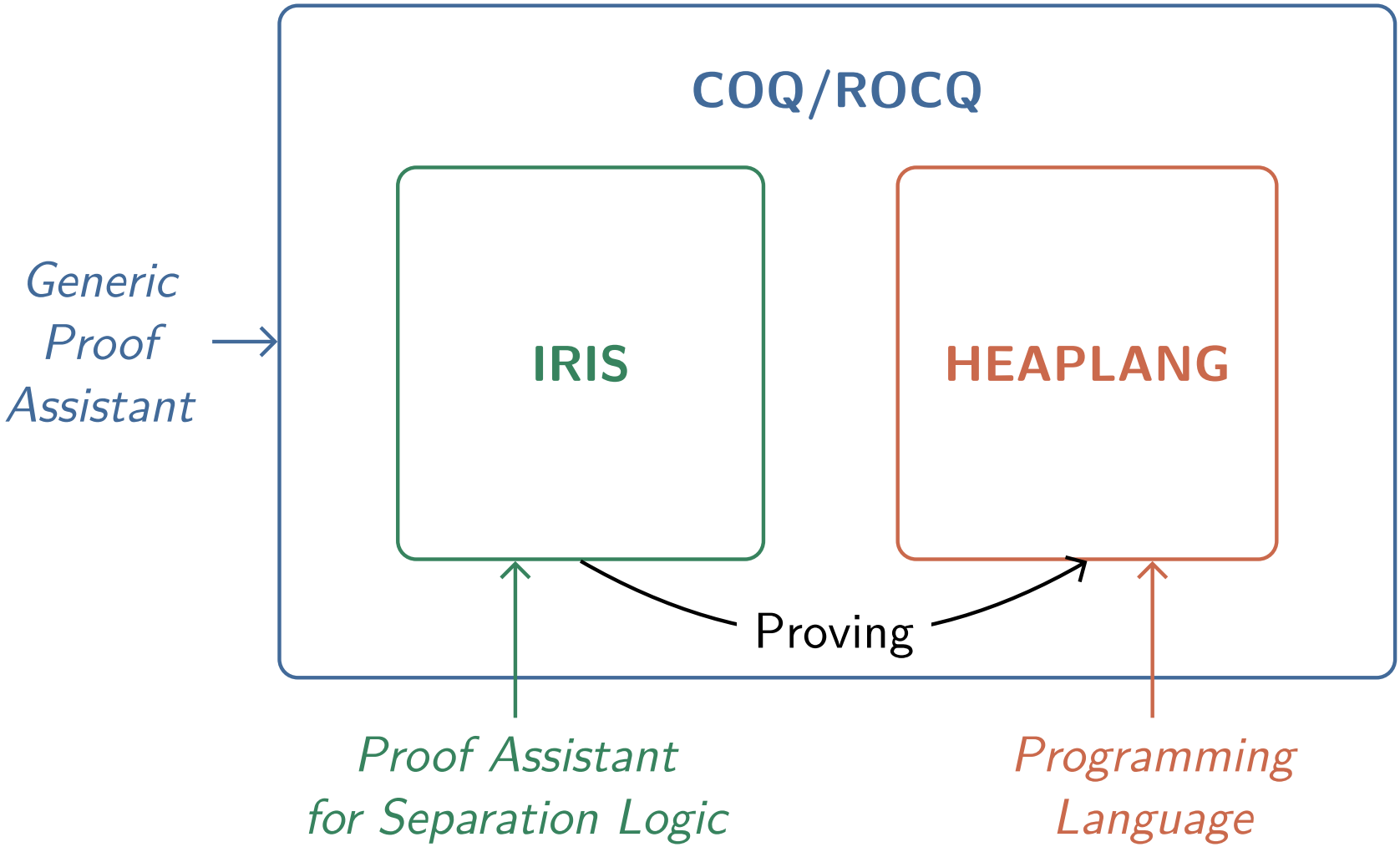
VeriSLO IP Final Presentation

Thibaut Blanc Amaury Mazoyer Juliette Ponsonnet Hugo Salou

École Normale Supérieure de Lyon, IP Project

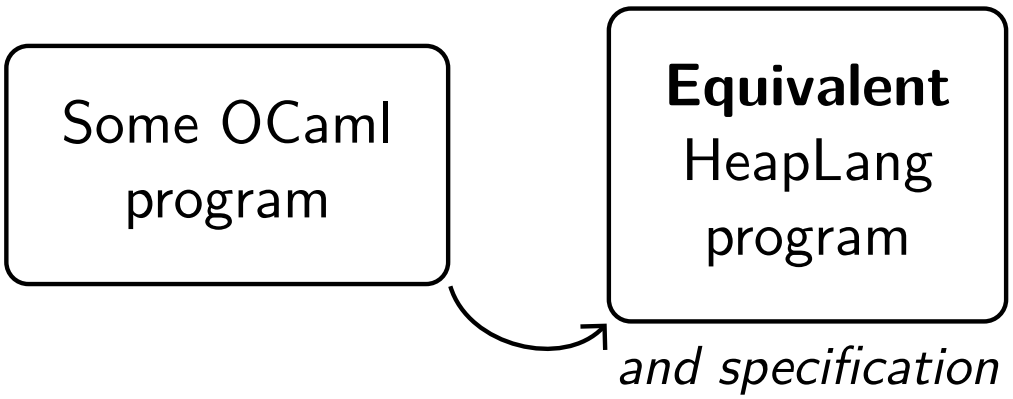
March, 17th 2025

Rocq and Iris

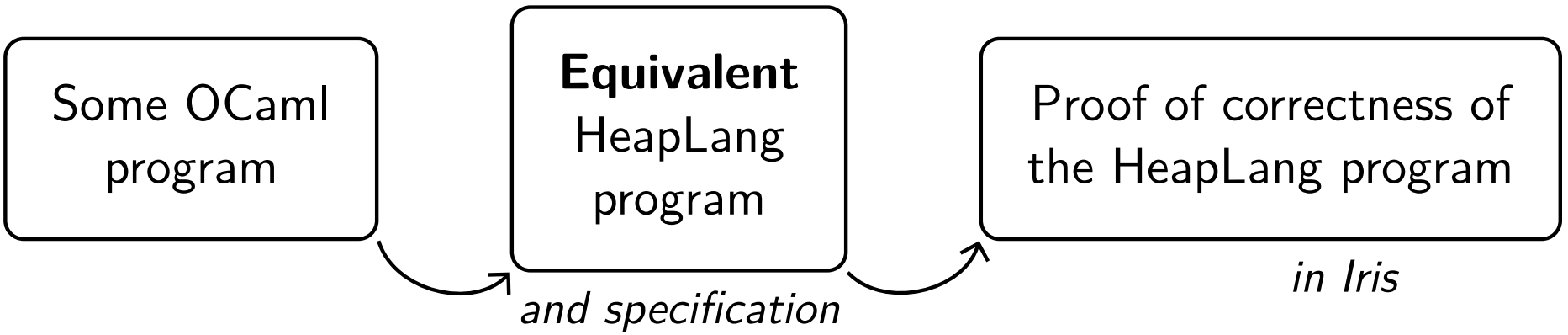


The “normal” process

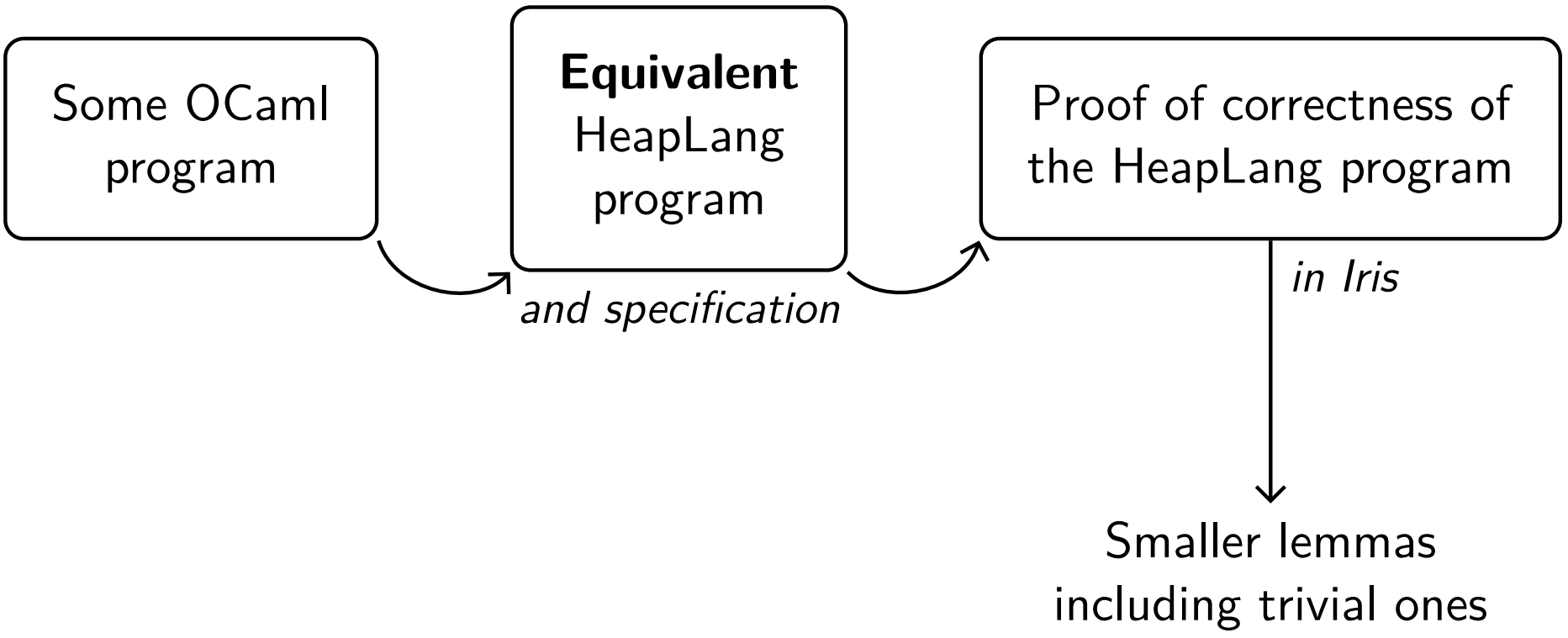
The “normal” process



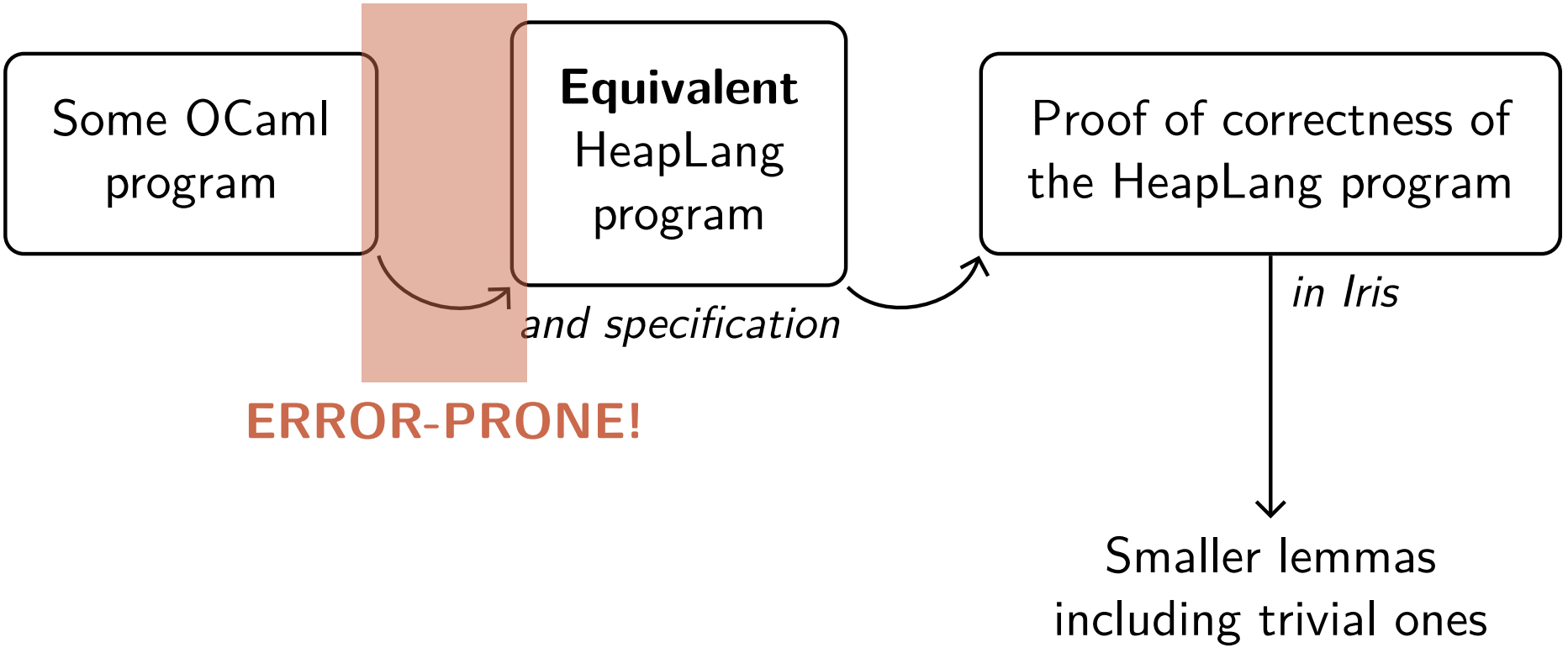
The “normal” process



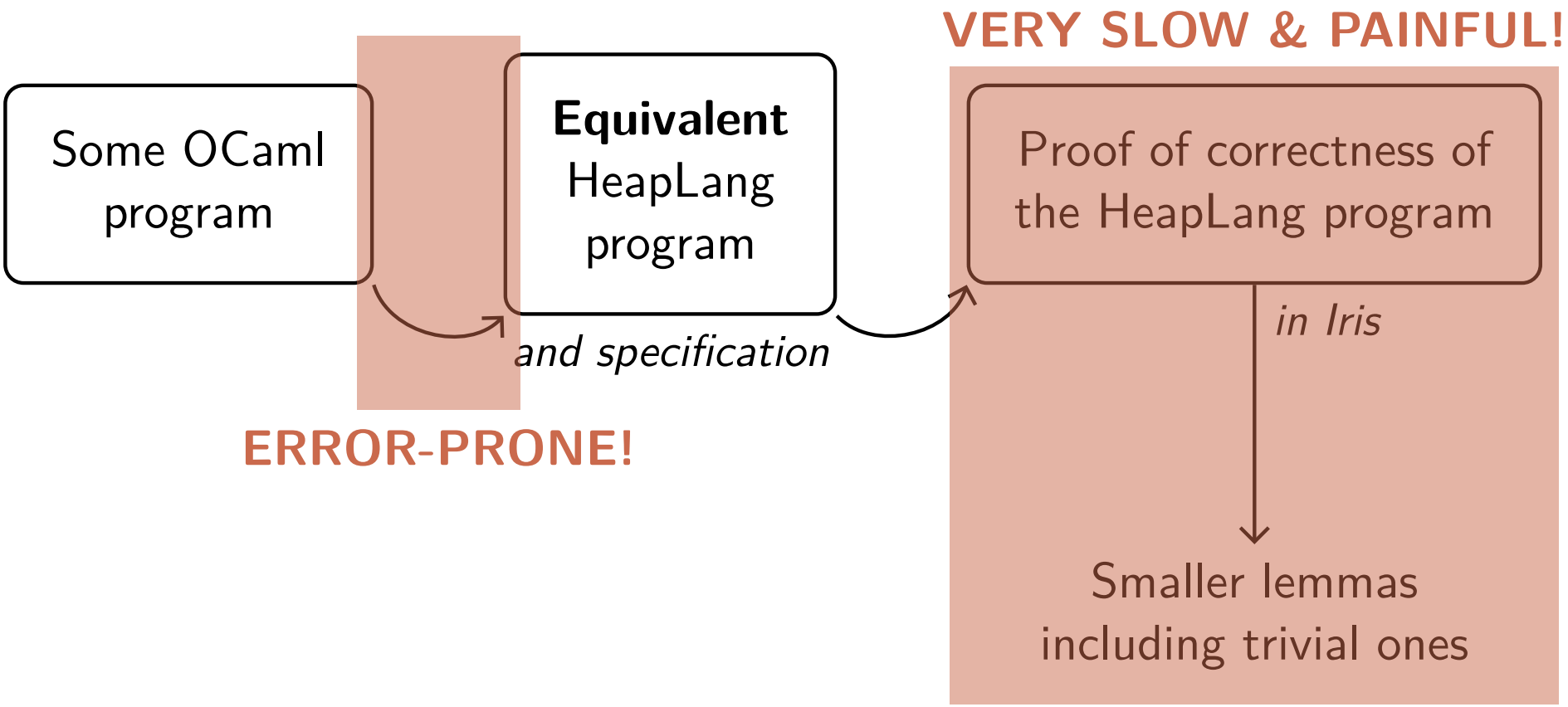
The “normal” process



The “normal” process



The “normal” process



Outline

Two example programs

We are able to *automatically* prove programs.

```
let result =
    let x = ref 1 in
    if !x <= 2
        then x := 3;
    !x
    [@post "!x = #3"]
```

Two example programs

We are able to *automatically* prove programs.

```
let result =
    let x = ref 1 in
    if !x <= 2
        then x := 3;
    !x
    [@post "!x = #3"]
```

```
[@@@vernac {|
Definition even (n : Z) : Prop :=
  exists k : Z, n = (2 * k)%Z.
|}]

let result =
  let x = ref 0 in
  while !x <= 10 do
    x := !x + 2;
  done
  [@invariant "even !x"]
  [@post "!x = #12"]
```

Two example programs

We are able to *automatically* prove programs.

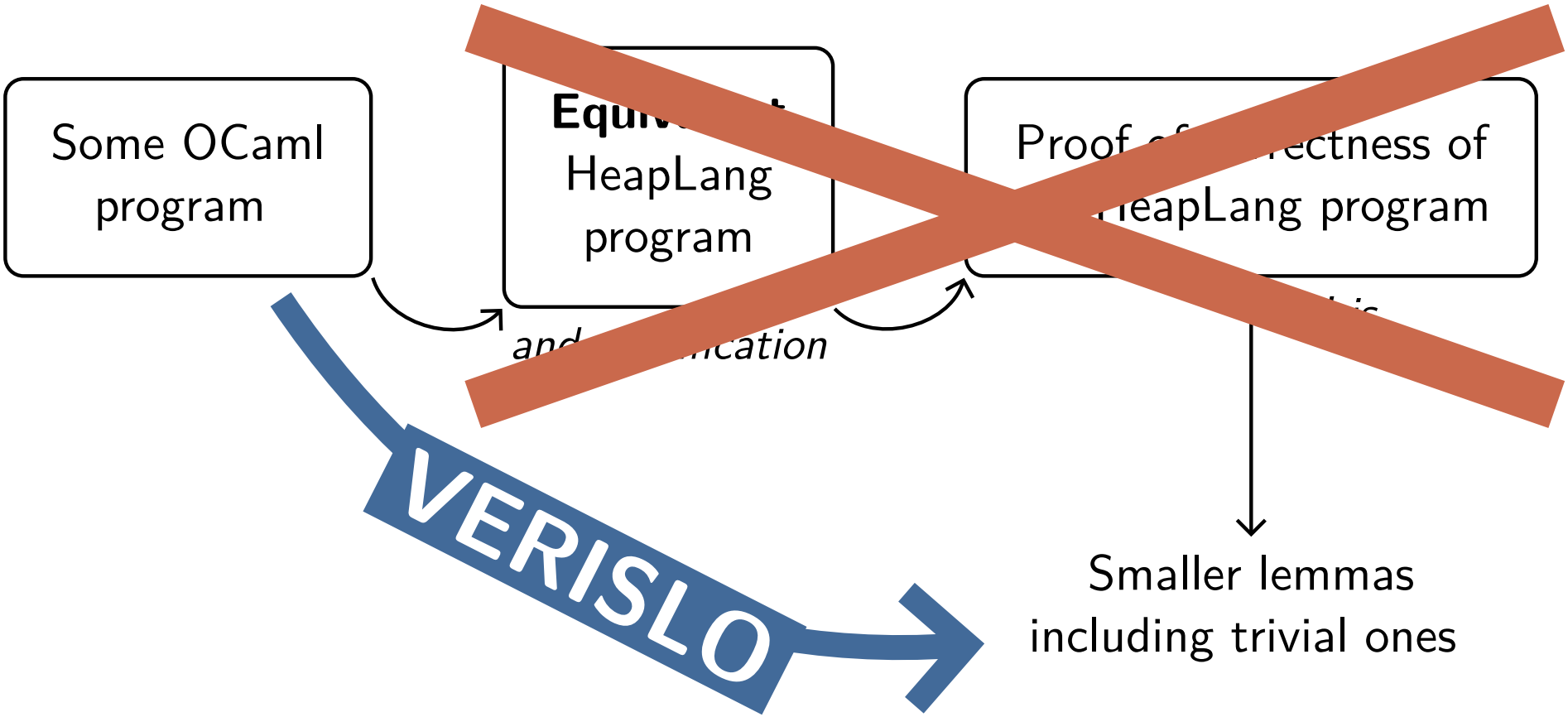
```
let result =
    let x = ref 1 in
    if !x <= 2
        then x := 3;
    !x
    [@post "!x = #3"]
```

```
[@@@vernac {|
Definition even (n : Z) : Prop :=
  exists k : Z, n = (2 * k)%Z.
|}]

let result =
  let x = ref 0 in
  while !x <= 10 do
    x := !x + 2;
  done
  [@invariant "even !x"]
  [@post "!x = #12"]
```

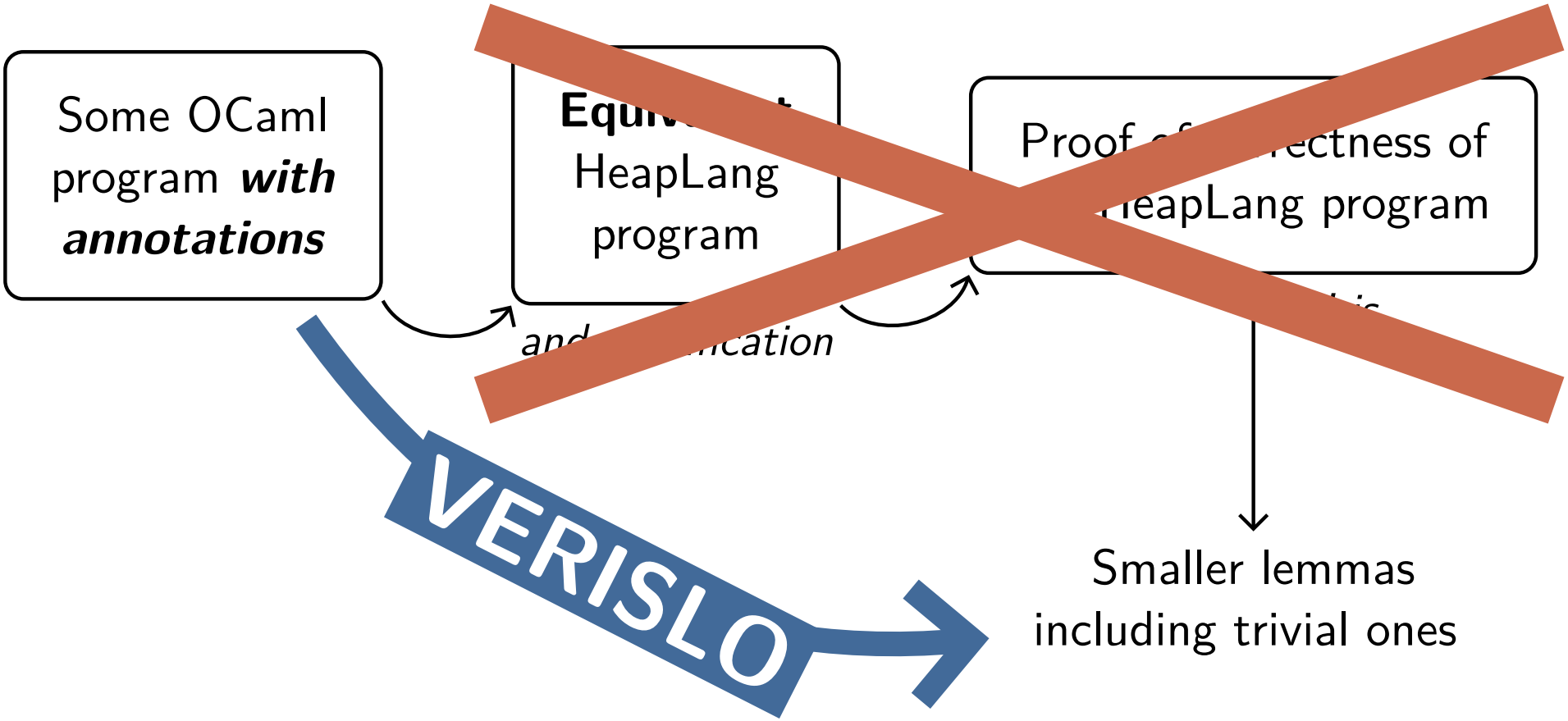
We have some ***syntactic sugar***:

$$\texttt{even !x} \quad \rightsquigarrow \quad \exists v : \texttt{val}, \quad \texttt{x} \mapsto v \quad * \quad \texttt{even } v.$$

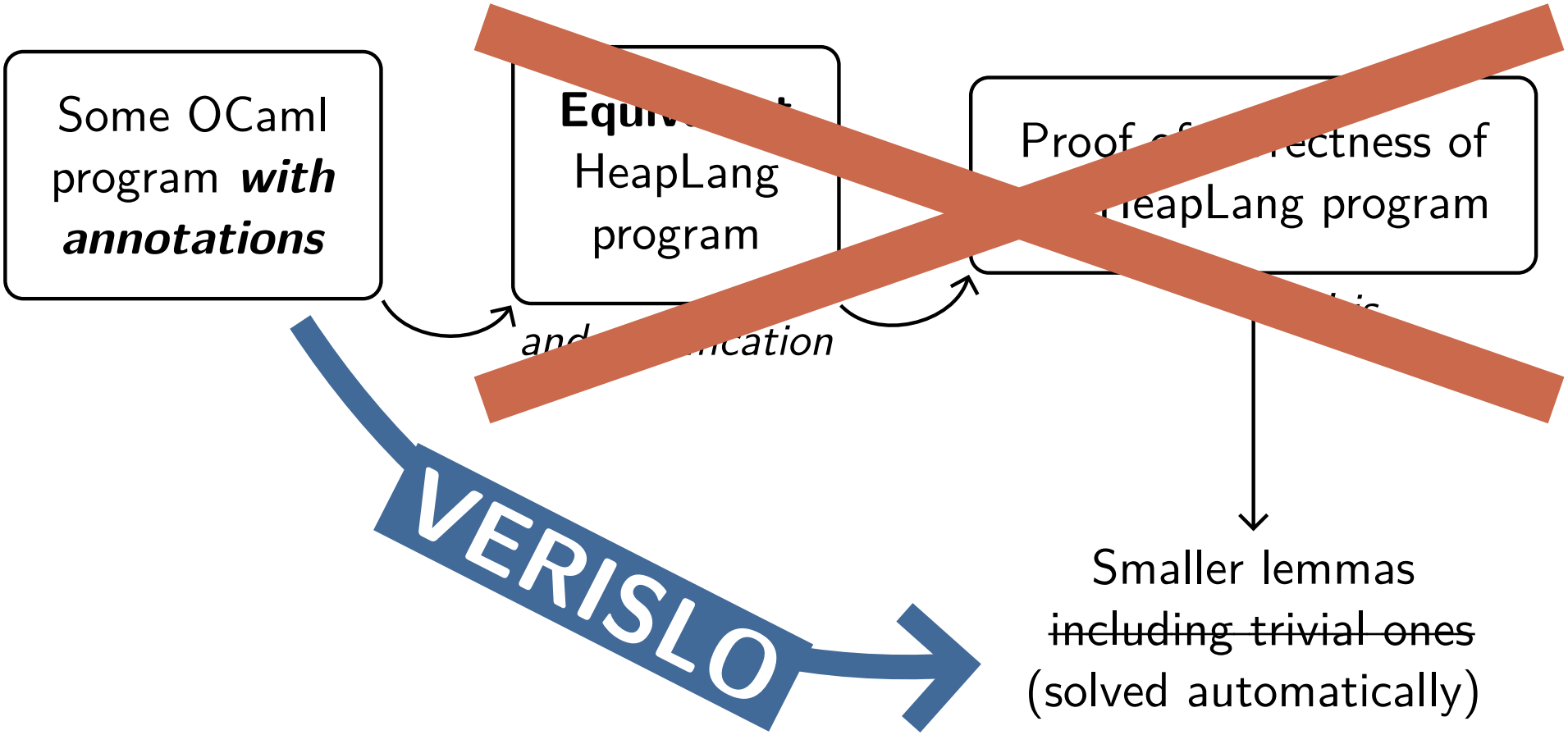
The VeriSLO process



The VeriSLO process



The VeriSLO process

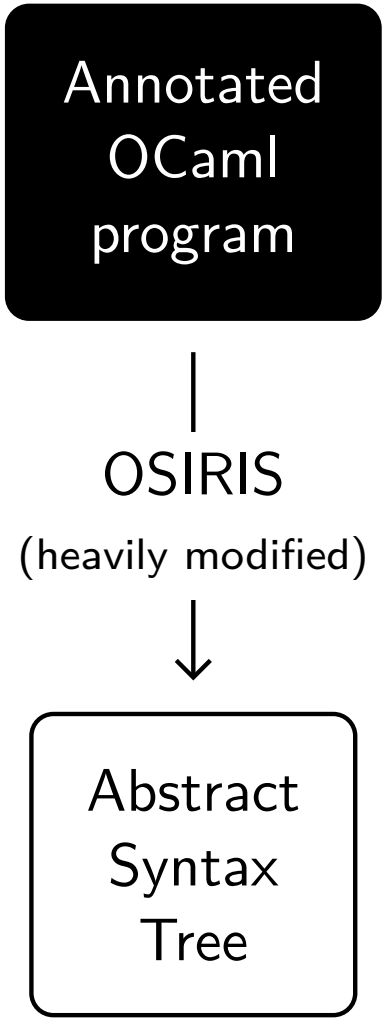


Outline

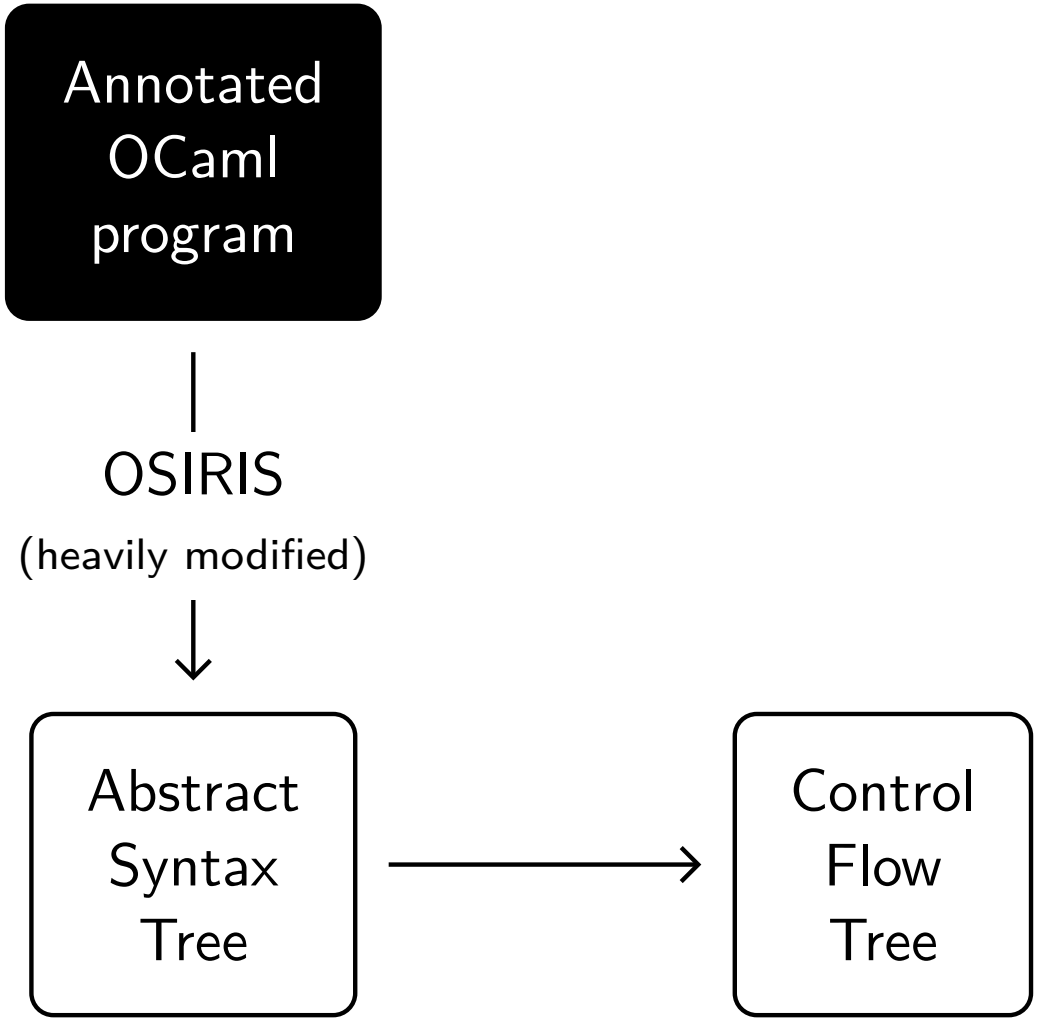
Architecture of VeriSLO: A bird's eye view

Annotated OCaml program

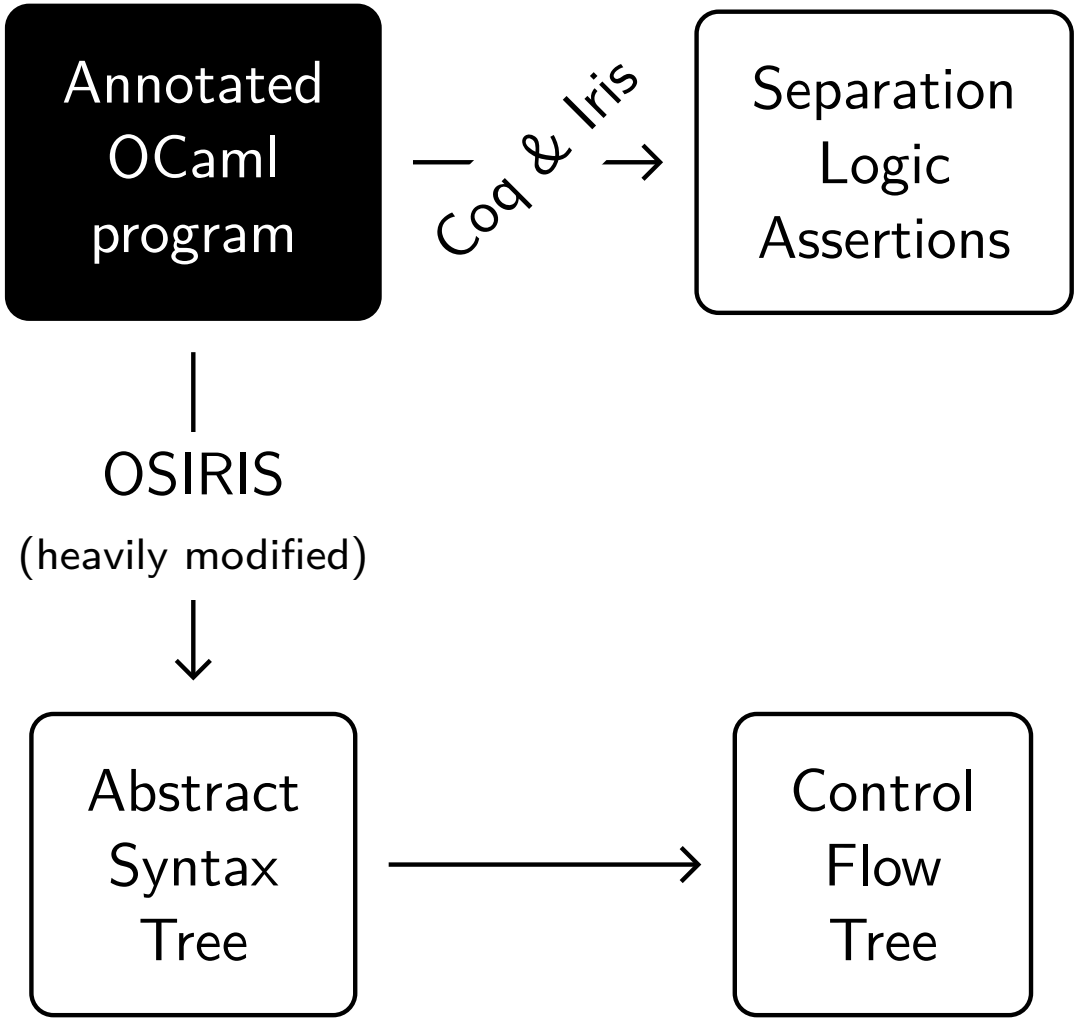
Architecture of VeriSLO: A bird's eye view



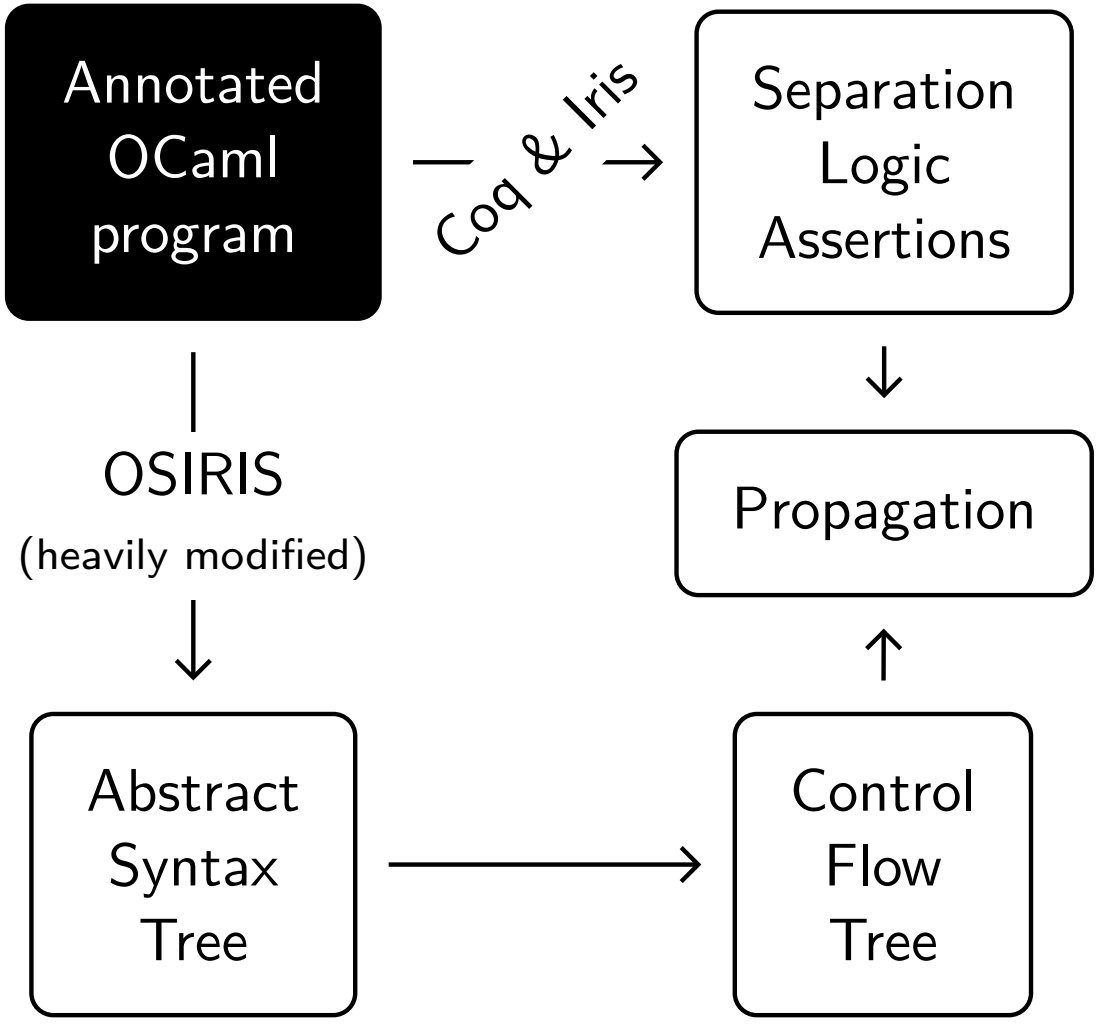
Architecture of VeriSLO: A bird's eye view



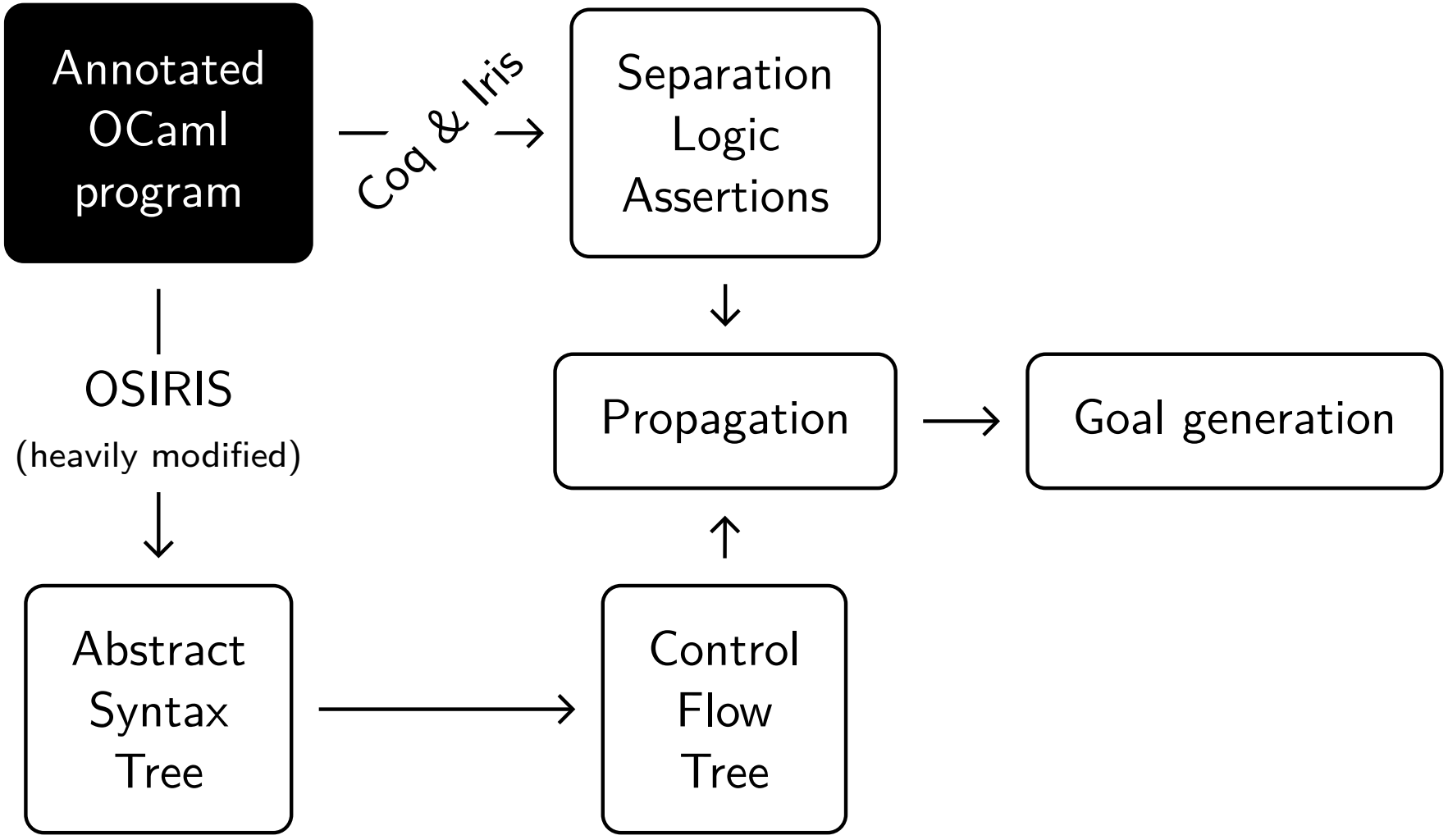
Architecture of VeriSLO: A bird's eye view



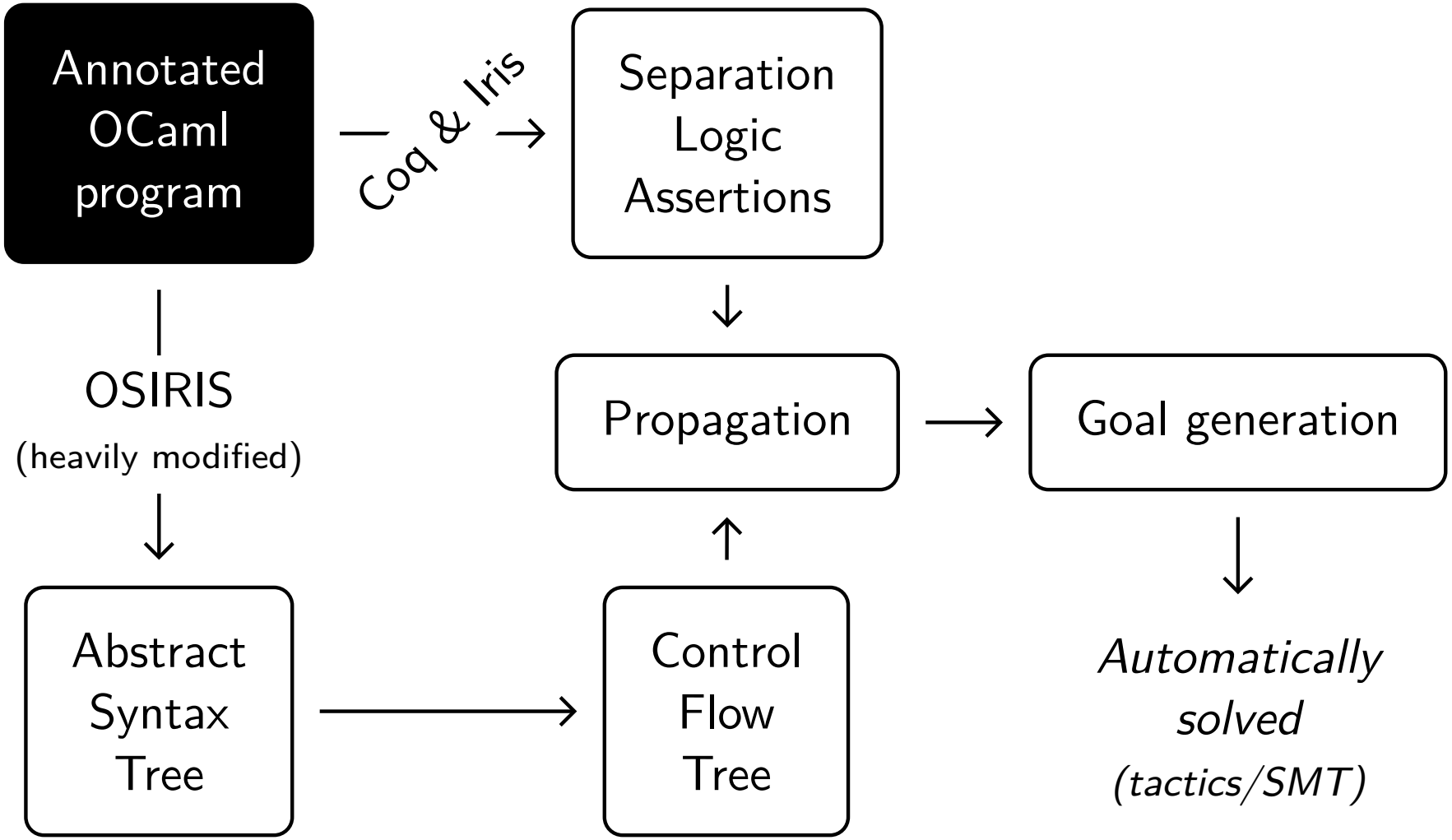
Architecture of VeriSLO: A bird's eye view



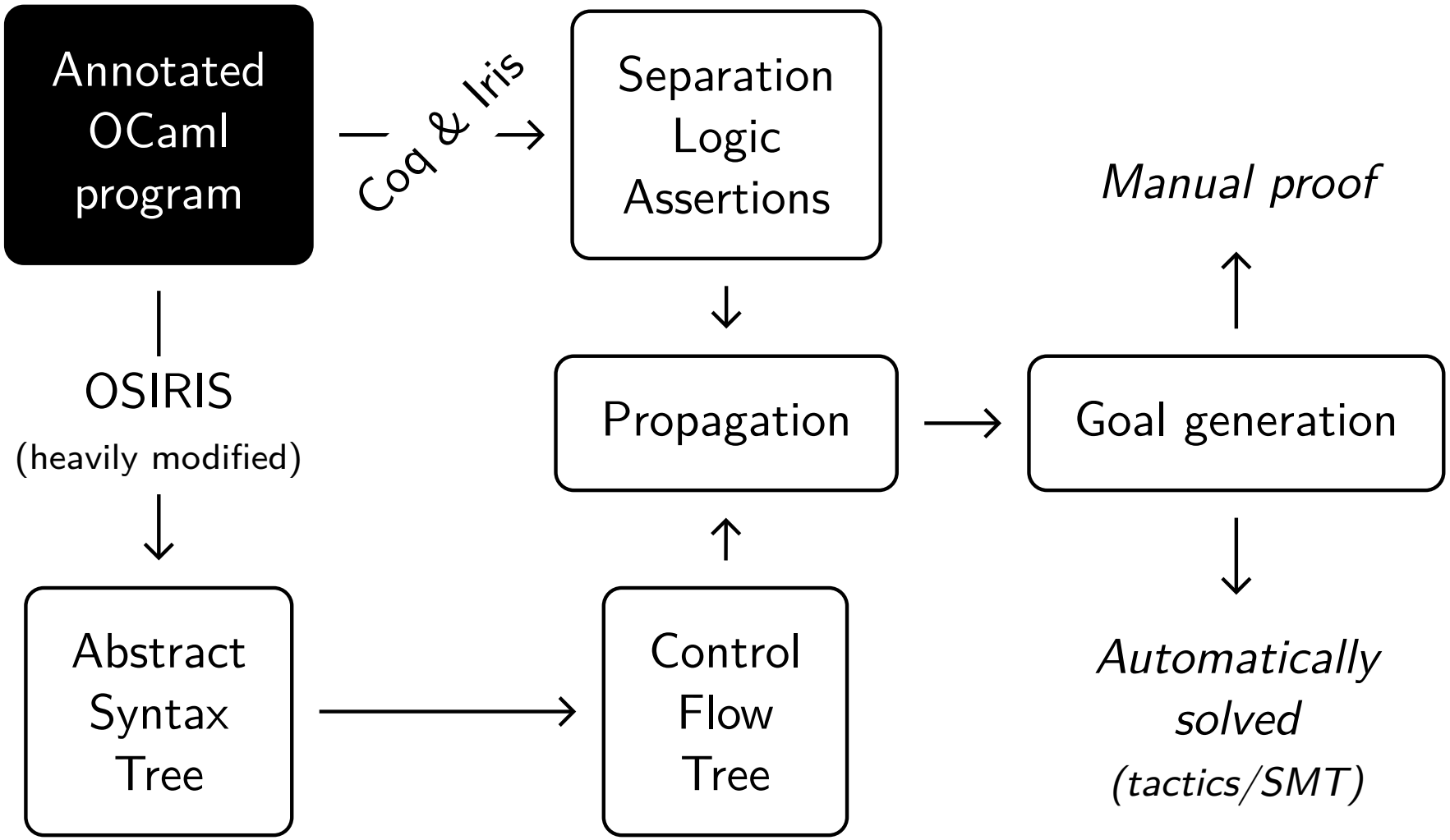
Architecture of VeriSLO: A bird's eye view



Architecture of VeriSLO: A bird's eye view

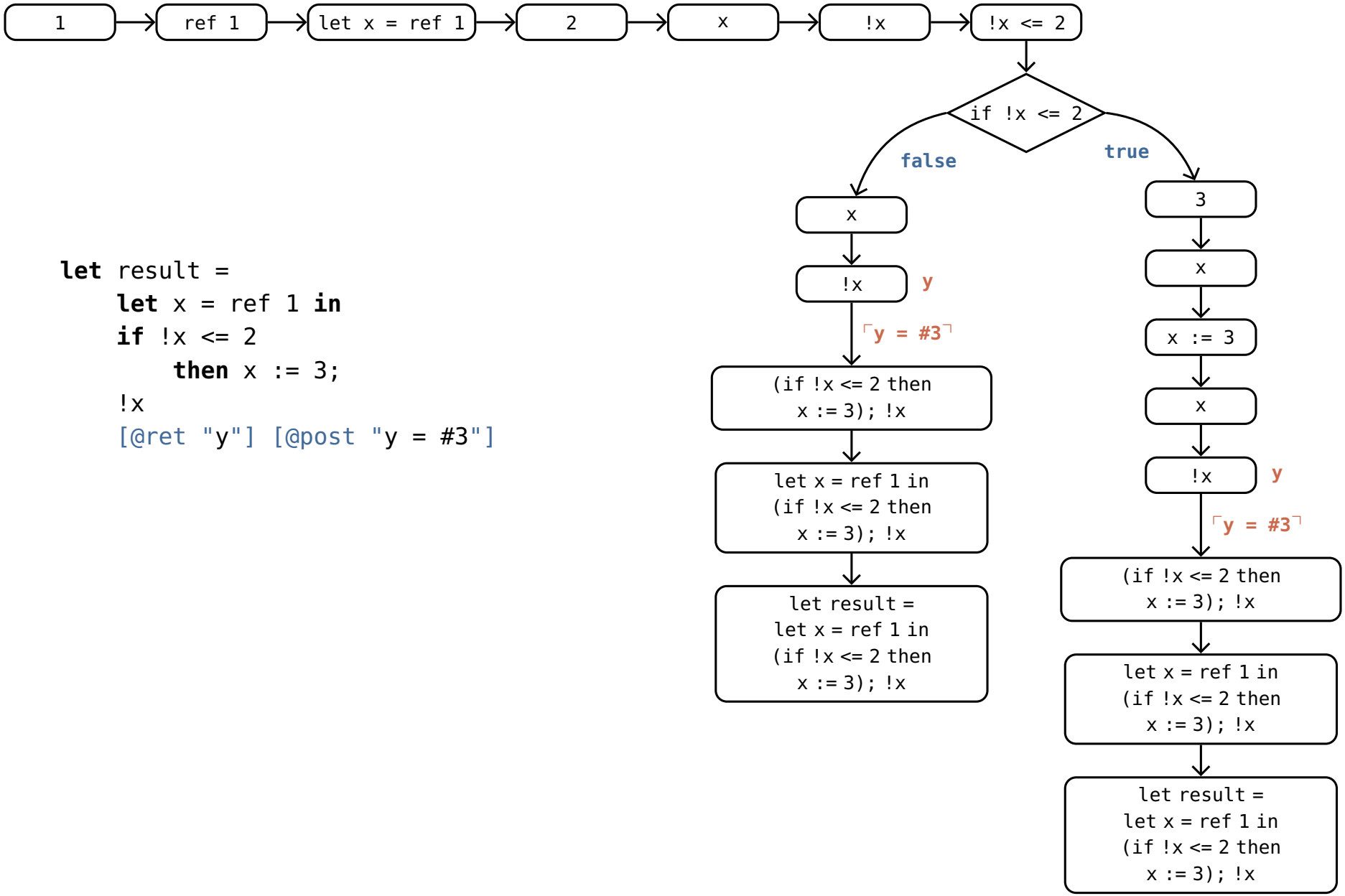


Architecture of VeriSLO: A bird's eye view



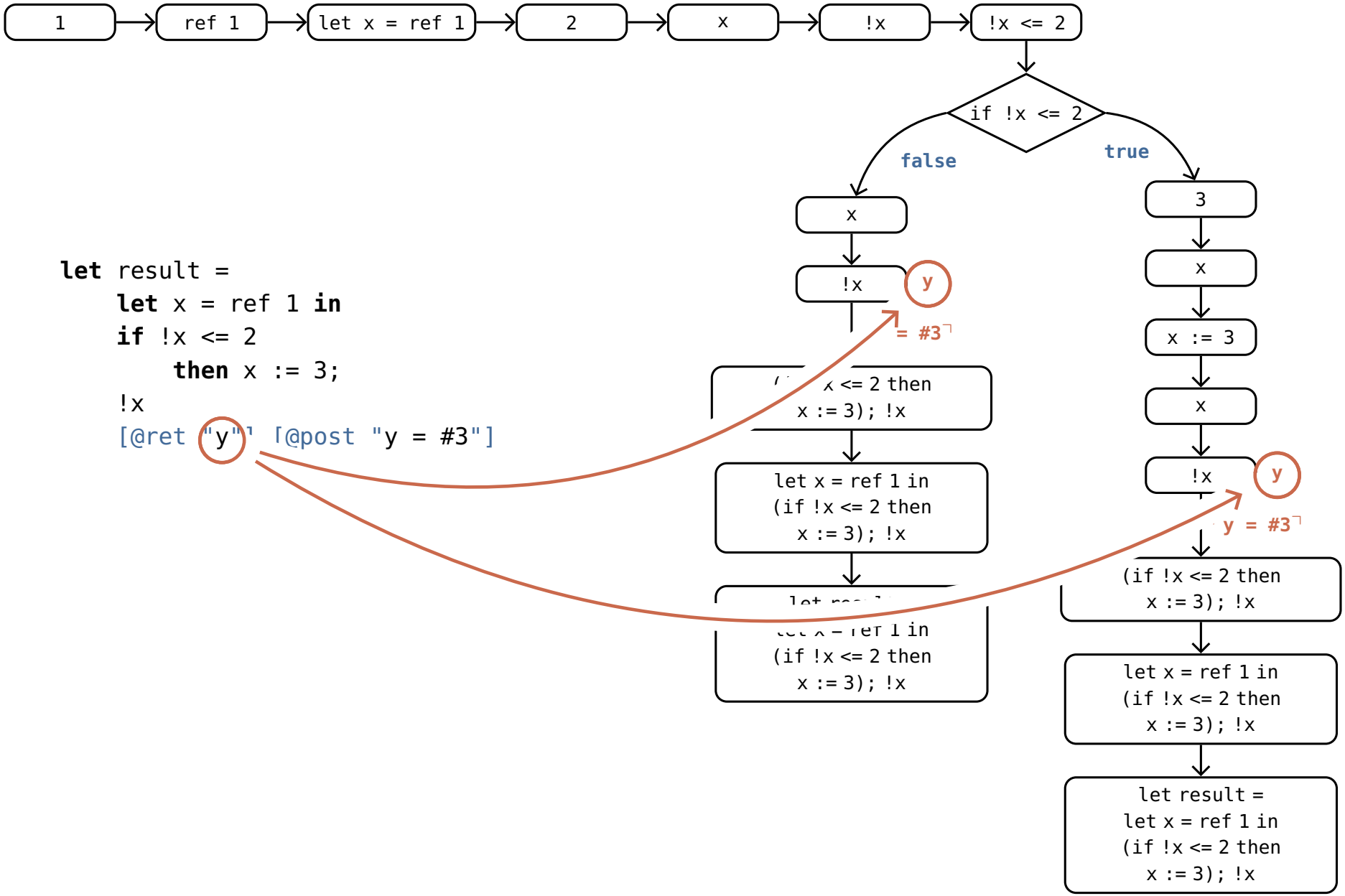
Control Flow Tree (CFT)

```
let result =
    let x = ref 1 in
    if !x <= 2
        then x := 3;
    !x
    [@ret "y"] [@post "y = #3"]
```



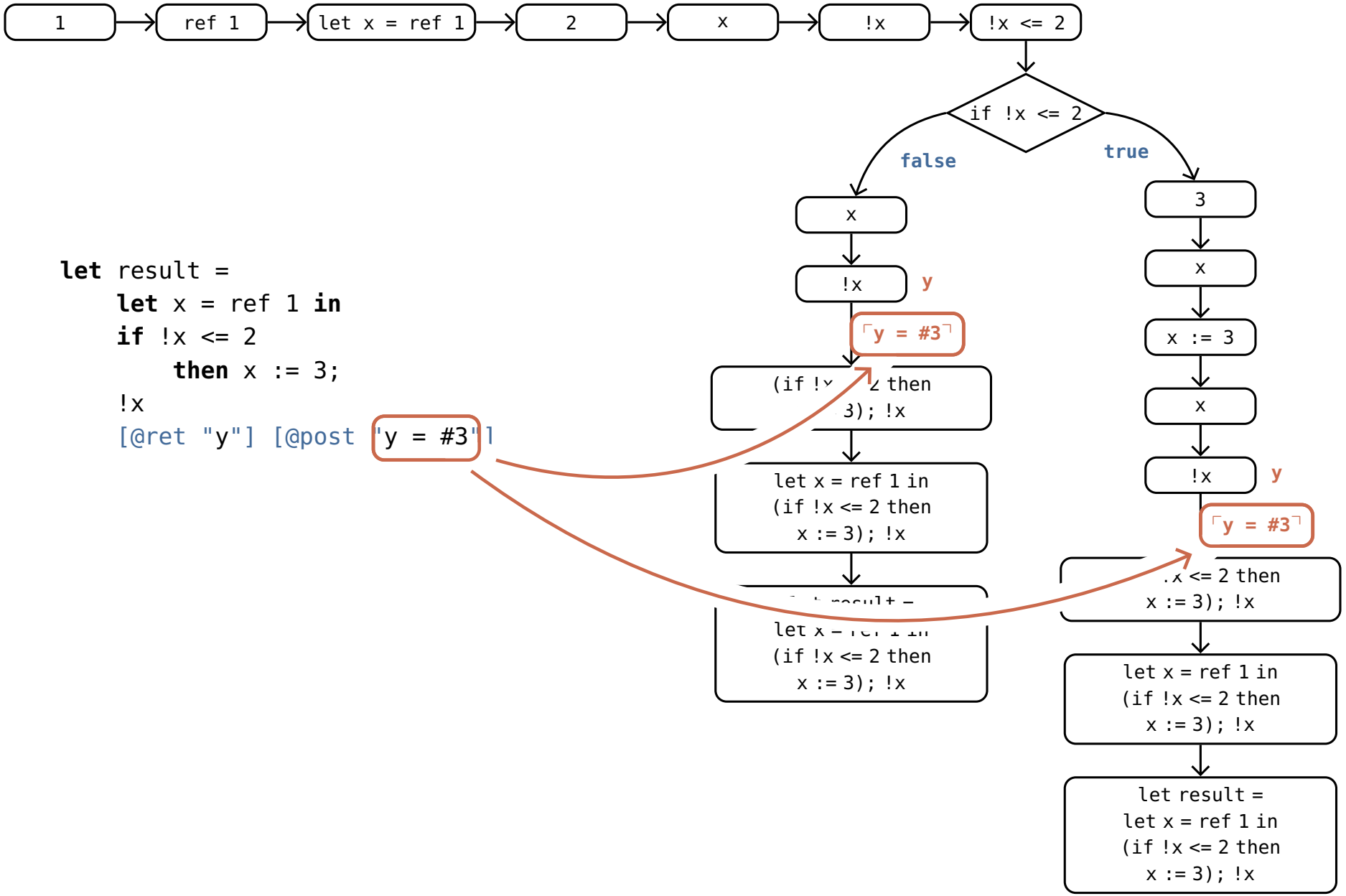
Control Flow Tree (CFT)

```
let result =
    let x = ref 1 in
    if !x <= 2
        then x := 3;
    !x
    [@ret "y"] [@post "y = #3"]
```

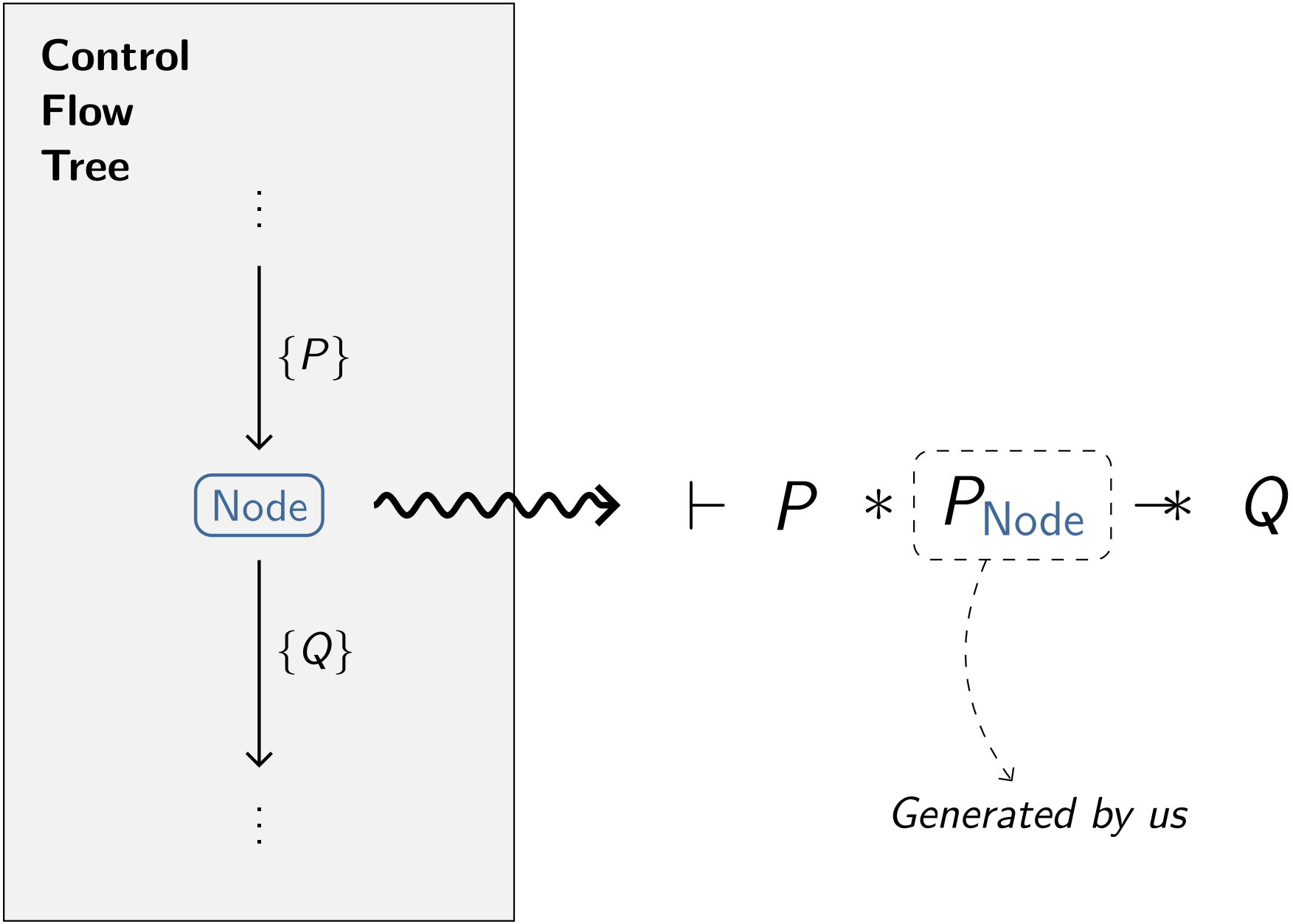


Control Flow Tree (CFT)

```
let result =
    let x = ref 1 in
    if !x <= 2
        then x := 3;
    !x
    [@ret "y"] [@post "y = #3"]
```



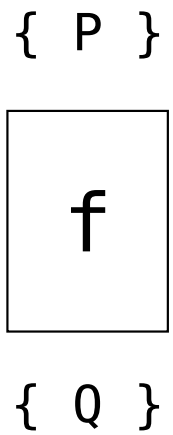
Obligation Generation



Outline

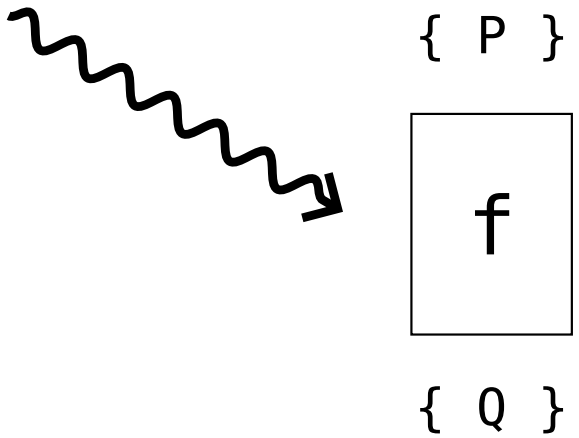
Handling functions

```
let f x =
  { P }
  ...
  { Q }
in
...
```



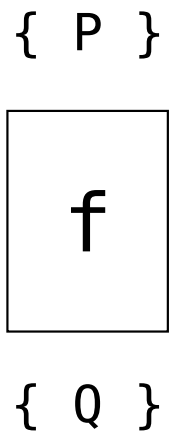
Handling functions

```
let f x =
  { P }
  ...
  { Q }
in
...
```



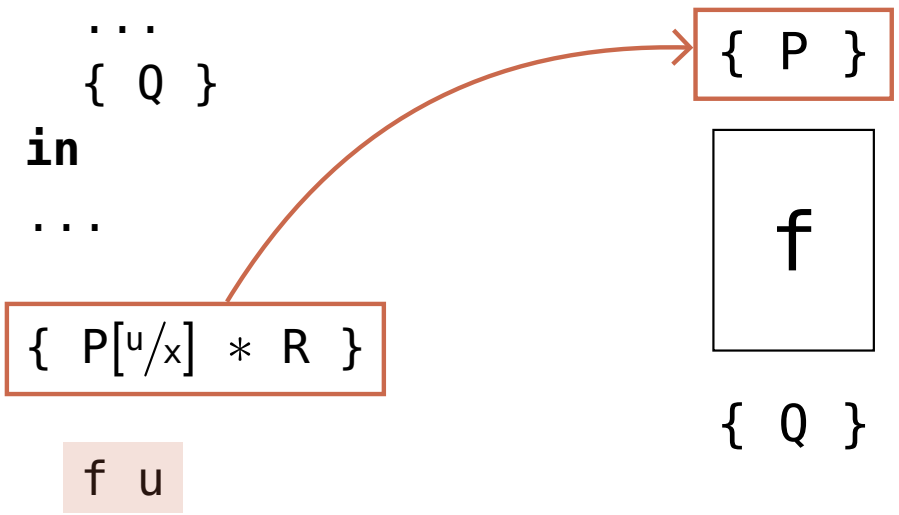
Handling functions

```
let f x =
  { P }
  ...
  { Q }
in
...
```



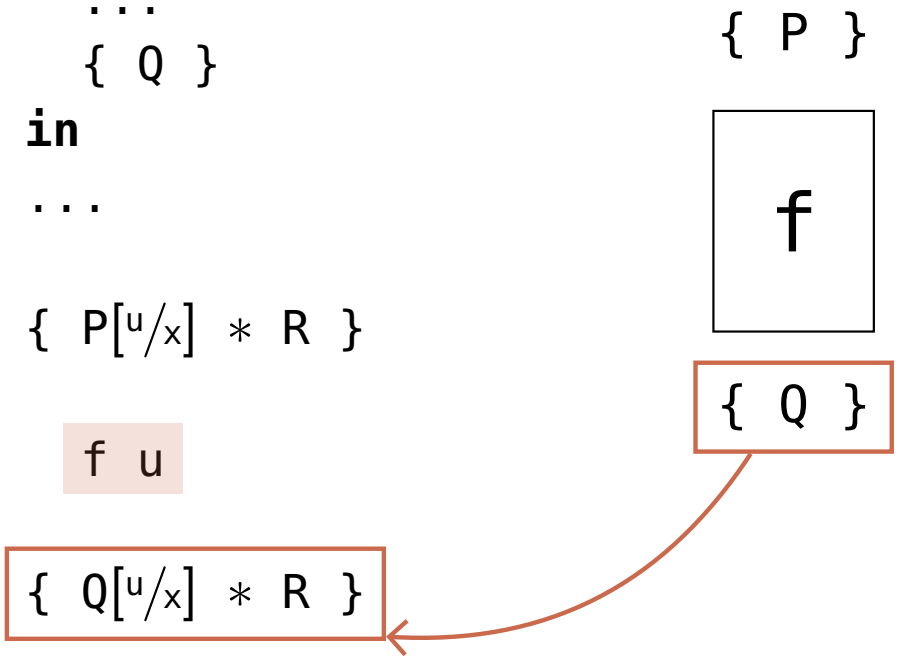
Handling functions

```
let f x =
  { P }
  ...
  { Q }
in
...
```



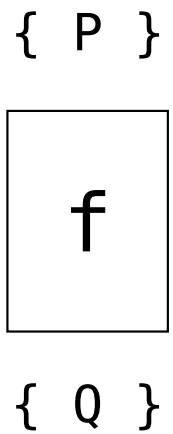
Handling functions

```
let f x =
  { P }
  ...
  { Q }
in
...

{ P[u/x] * R }

  f u
```

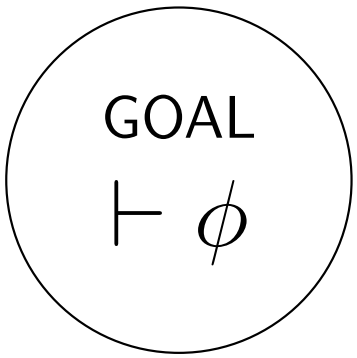


Handling functions

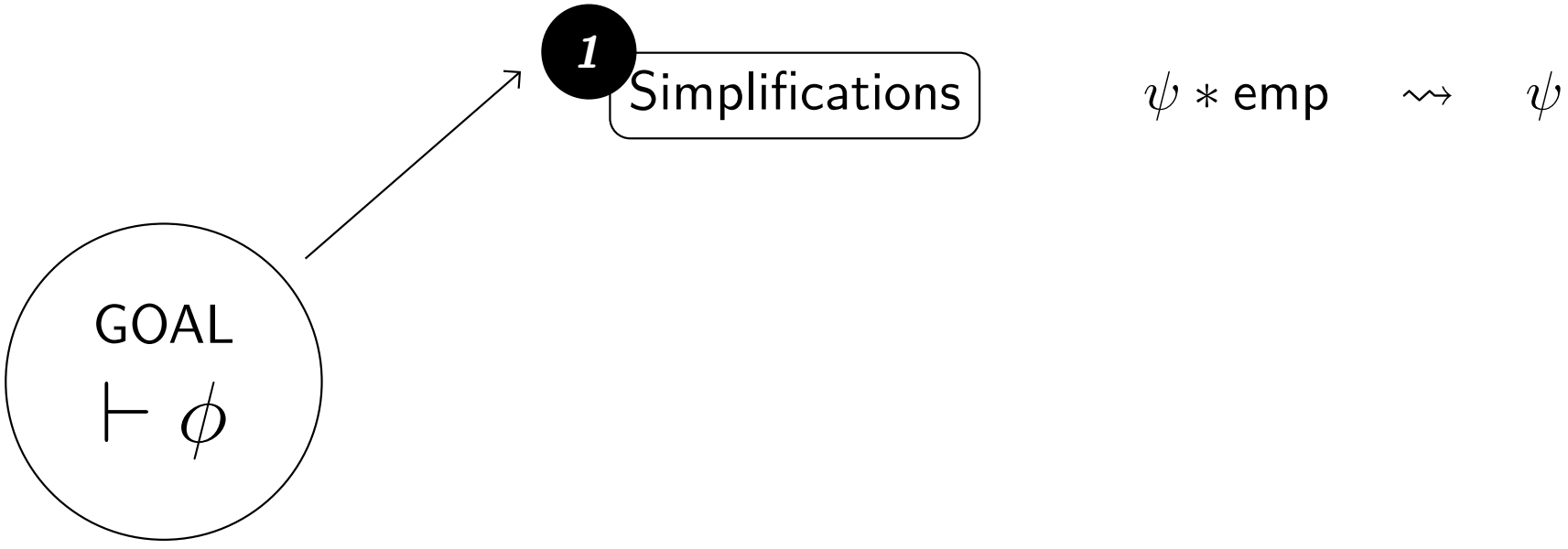
```
let f x =
  { P }
  ...
  { Q }
in
...

{ P[u/x] ∗ R }

  f u

{ Q[u/x] ∗ R }

...
```



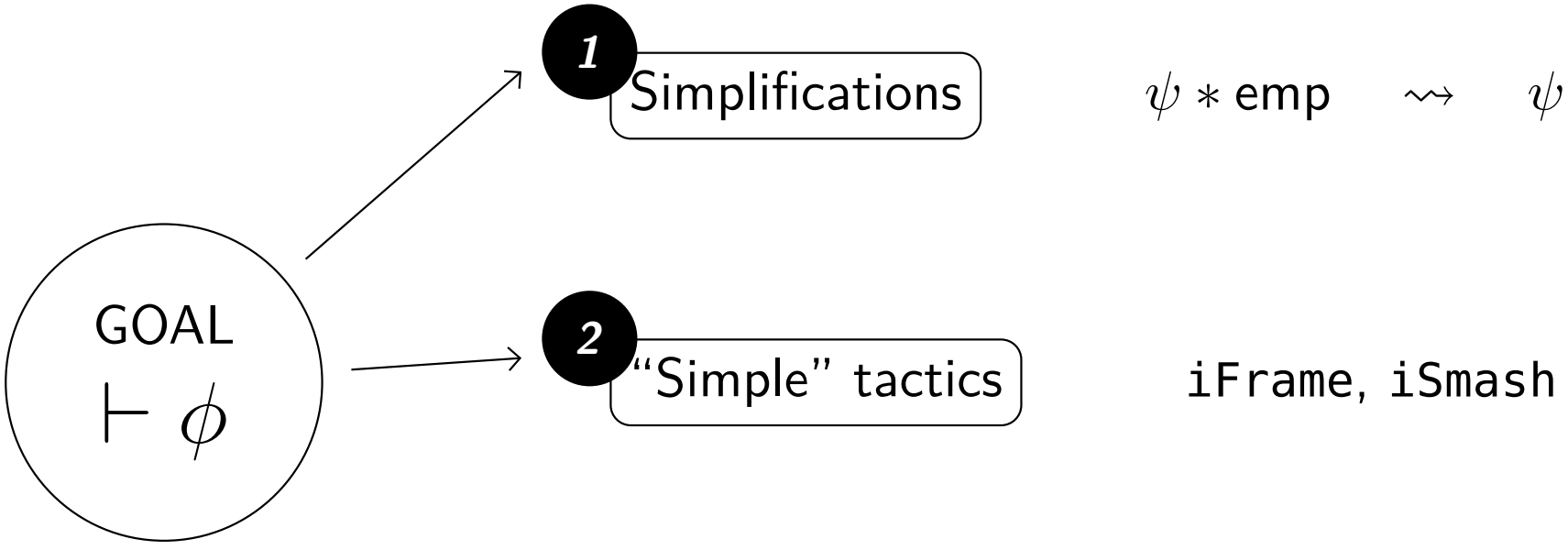
Automatically solving goals: Coq & SMT



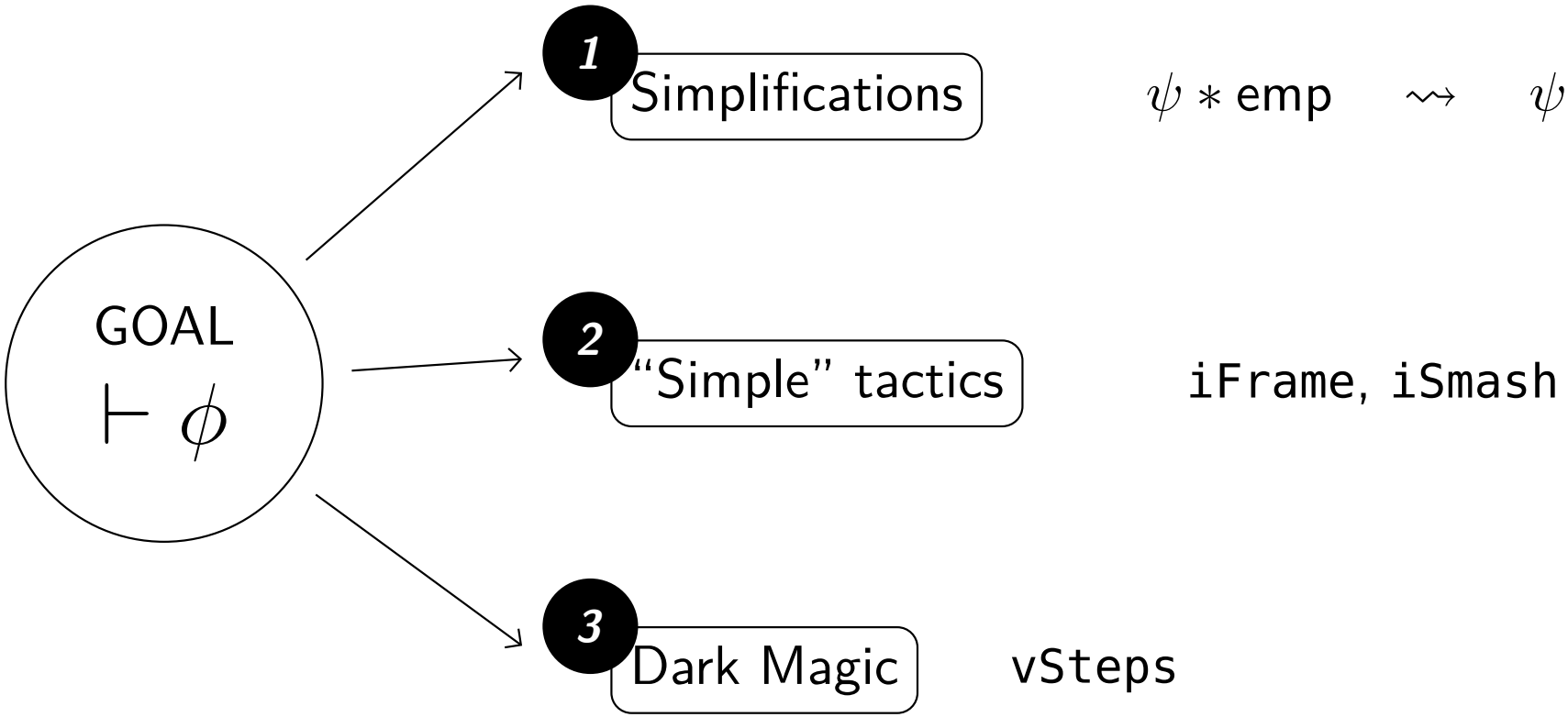
Automatically solving goals: Coq & SMT



Automatically solving goals: Coq & SMT



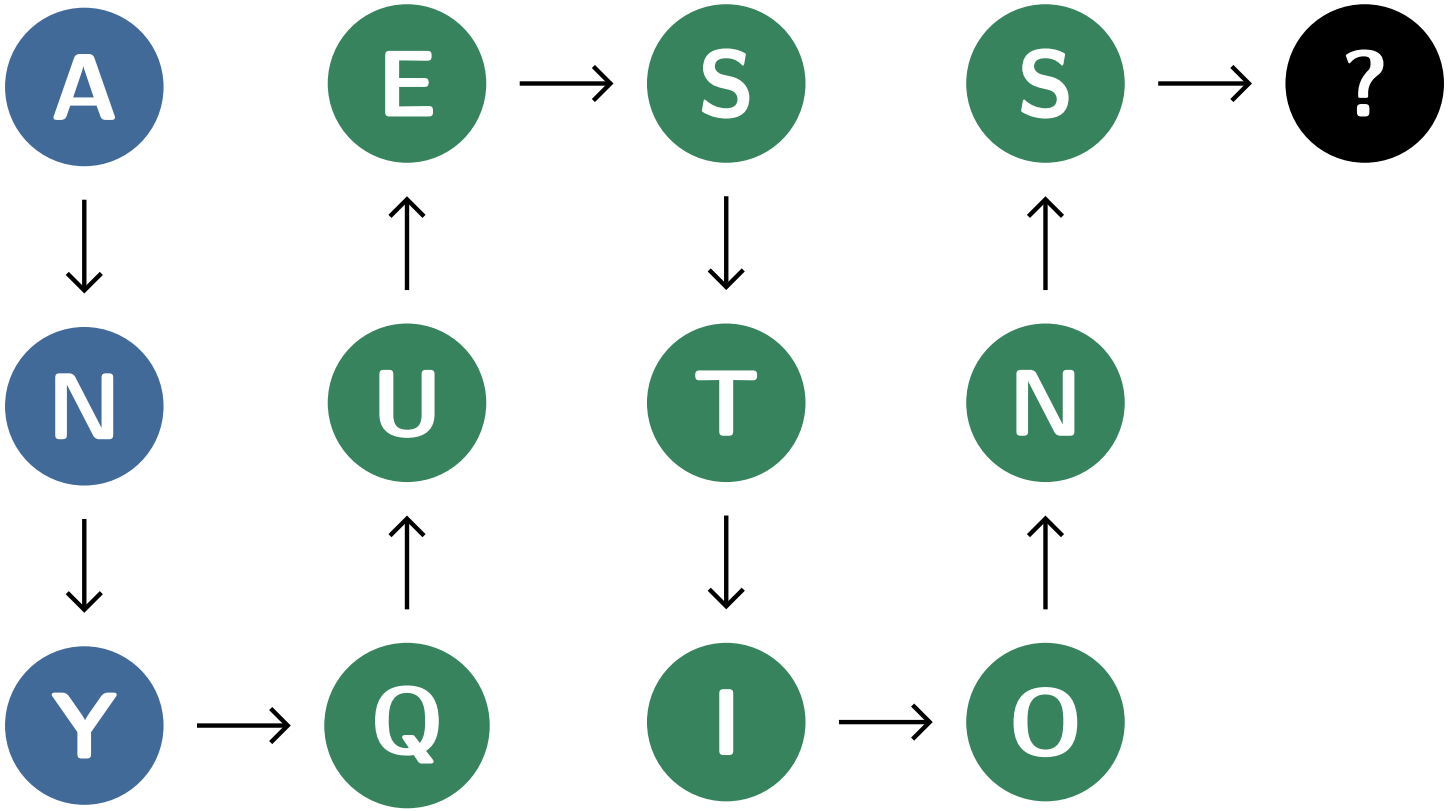
Automatically solving goals: Coq & SMT



*preparing the goal for the **SMT solver***

Outline

Demo time!


A
E
S
S
?
N
U
T
N
Y
Q
I
O