VeriSLO IP Midterm Presentation

Thibaut Blanc Amaury Mazoyer Juliette Ponsonnet Hugo Salou

École Normale Supérieure de Lyon, IP Project

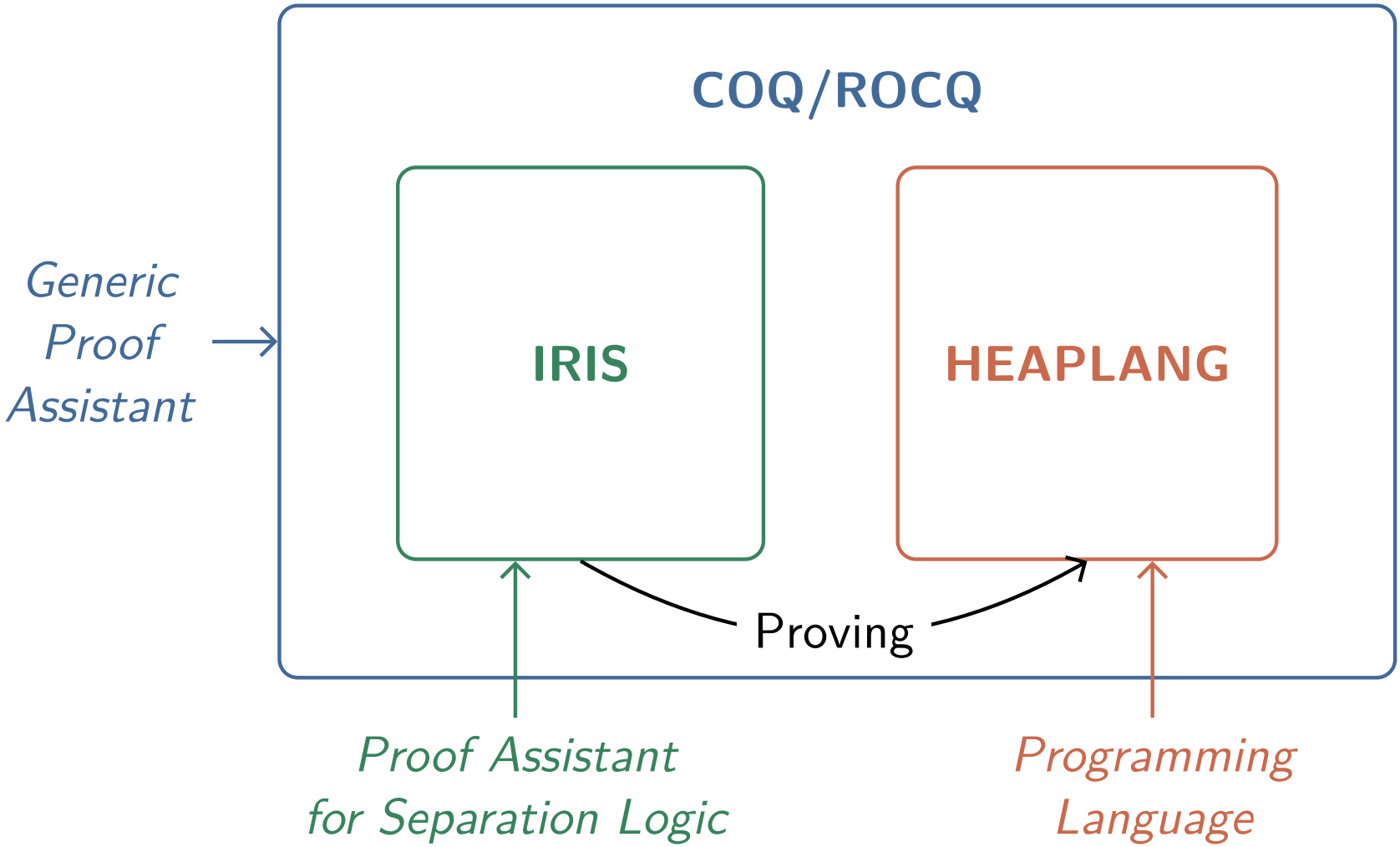
November 12 2025

You are here

Why this Tool?

- Software correctness is crucial
- Testing is insufficient:
  - It finds bugs but does not prove their absence
  - Limited coverage of execution cases
- Formal verification:
  - The machine checks for the absence of reasoning errors
  - Can reason directly about the implementation

Rocq and Iris



Issues We Address

"At present, our proofs rely on a manual transcription of our OCaml code into HeapLang. In future work, it would be desirable to use an automated translation, such as those offered by Zoo [All25] or Osiris [Sea+25].
We have encountered serious performance problems with the current implementation of Iris on top of Rocq. Iris's tactics can be very slow and can fail to terminate for unknown reasons; sometimes a change causes divergence in a seemingly unrelated part of the proof. Although we have eventually worked around or tolerated these problems, they have made our progress much slower and more painful than expected."

Issues We Address

"At present, our proofs rely on a manual transcription of our OCaml code into HeapLang. In future work, it would be desirable to use an automated translation, such as those offered by Zoo [All25] or Osiris [Sea+25].
We have encountered serious performance problems with the current implementation of Iris on top of Rocq. Iris's tactics can be very slow and can fail to terminate for unknown reasons; sometimes a change causes divergence in a seemingly unrelated part of the proof. Although we have eventually worked around or tolerated these problems, they have made our progress much slower and more painful than expected."

Issues We Address

"At present, our proofs rely on a manual transcription of our OCaml code into HeapLang. In future work, it would be desirable to use an automated translation, such as those offered by Zoo [All25] or Osiris [Sea+25].
We have encountered serious performance problems with the current implementation of Iris on top of Rocq. Iris's tactics can be very slow and can fail to terminate for unknown reasons; sometimes a change causes divergence in a seemingly unrelated part of the proof. Although we have eventually worked around or tolerated these problems, they have made our progress much slower and more painful than expected."

Issues We Address

"At present, our proofs rely on a manual transcription of our OCaml code into HeapLang. In future work, it would be desirable to use an automated translation, such as those offered by Zoo [All25] or Osiris [Sea+25].
We have encountered serious performance problems with the current implementation of Iris on top of Rocq. Iris's tactics can be very slow and can fail to terminate for unknown reasons; sometimes a change causes divergence in a seemingly unrelated part of the proof. Although we have eventually worked around or tolerated these problems, they have made our progress much slower and more painful than expected."

Issues We Address

"At present, our proofs rely on a manual transcription of our OCaml code into HeapLang. In future work, it would be desirable to use an automated translation, such as those offered by Zoo [All25] or Osiris [Sea+25].
We have encountered serious performance problems with the current implementation of Iris on top of Rocq. Iris's tactics can be very slow and can fail to terminate for unknown reasons; sometimes a change causes divergence in a seemingly unrelated part of the proof. Although we have eventually worked around or tolerated these problems, they have made our progress much slower and more painful than expected."

Issues We Address

"At present, our proofs rely on a manual transcription of our OCaml code into HeapLang. In future work, it would be desirable to use an automated translation, such as those offered by Zoo [All25] or Osiris [Sea+25].
We have encountered serious performance problems with the current implementation of Iris on top of Rocq. Iris's tactics can be very slow and can fail to terminate for unknown reasons; sometimes a change causes divergence in a seemingly unrelated part of the proof. Although we have eventually worked around or tolerated these problems, they have made our progress much slower and more painful than expected."

Issues We Address

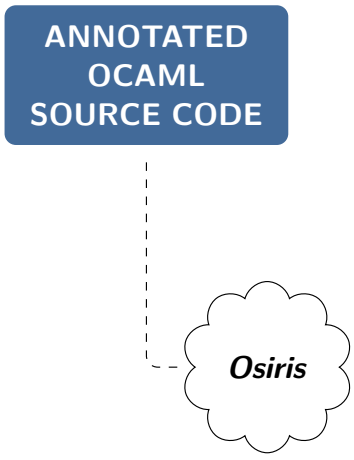
"At present, our proofs rely on a manual transcription of our OCaml code into HeapLang. In future work, it would be desirable to use an automated translation, such as those offered by Zoo [All25] or Osiris [Sea+25].
We have encountered serious performance problems with the current implementation of Iris on top of Rocq. Iris's tactics can be very slow and can fail to terminate for unknown reasons; sometimes a change causes divergence in a seemingly unrelated part of the proof. Although we have eventually worked around or tolerated these problems, they have made our progress much slower and more painful than expected."

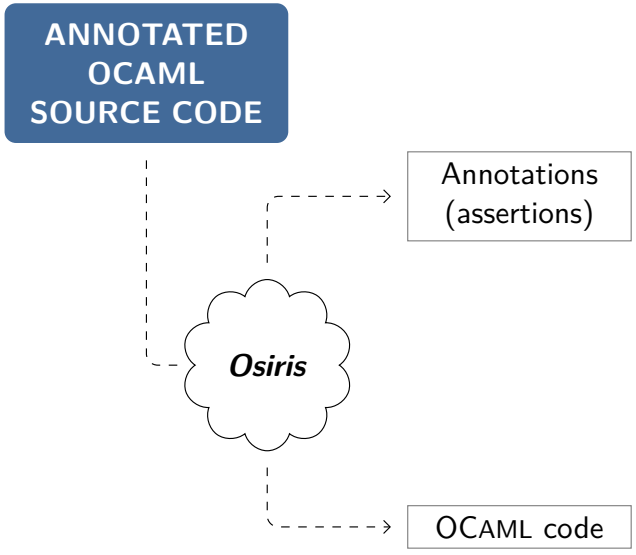
Issues We Address

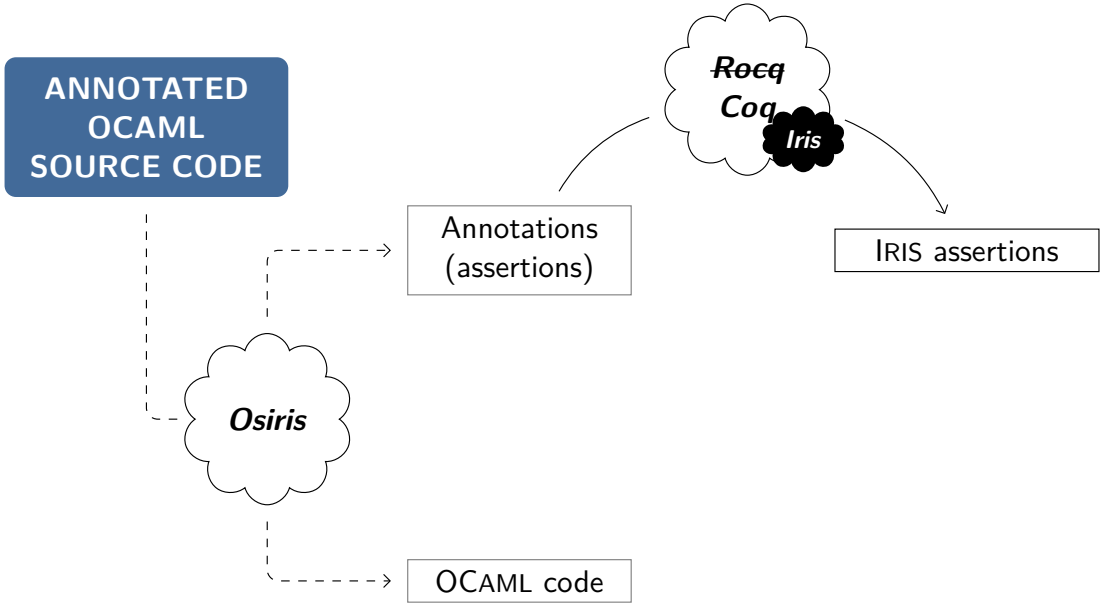
"At present, our proofs rely on a manual transcription of our OCaml code into HeapLang. In future work, it would be desirable to use an automated translation, such as those offered by Zoo [All25] or Osiris [Sea+25].
We have encountered serious performance problems with the current implementation of Iris on top of Rocq. Iris's tactics can be very slow and can fail to terminate for unknown reasons; sometimes a change causes divergence in a seemingly unrelated part of the proof. Although we have eventually worked around or tolerated these problems, they have made our progress much slower and more painful than expected."

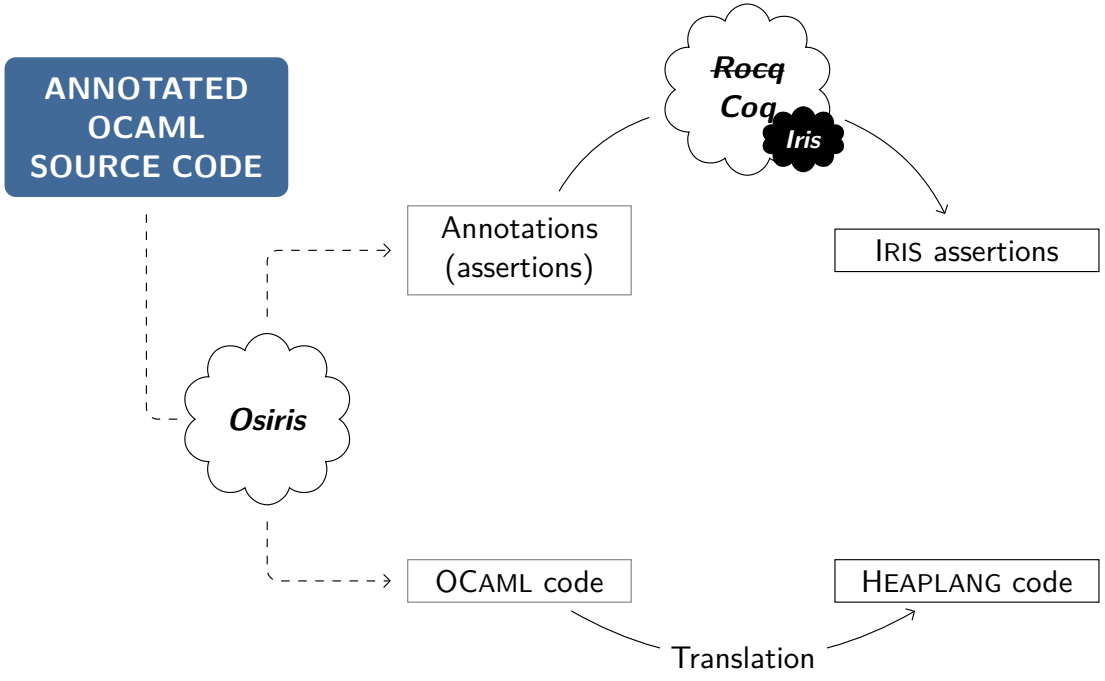
You are here

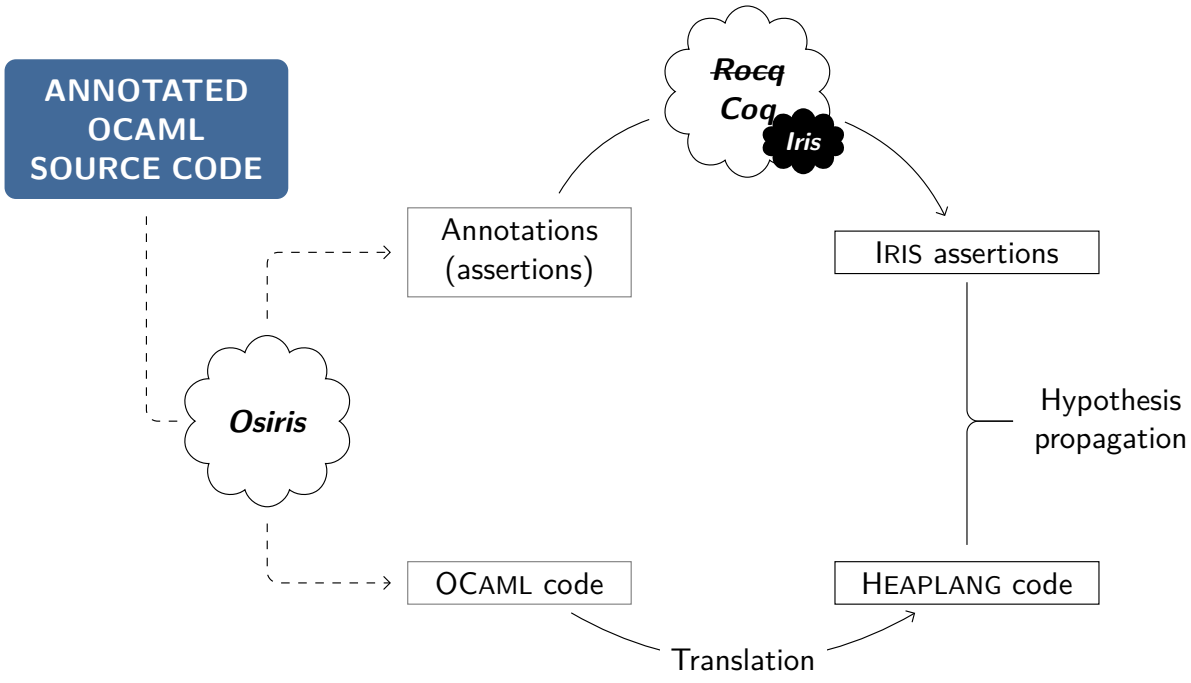

ANNOTATED
OCAML
SOURCE CODE

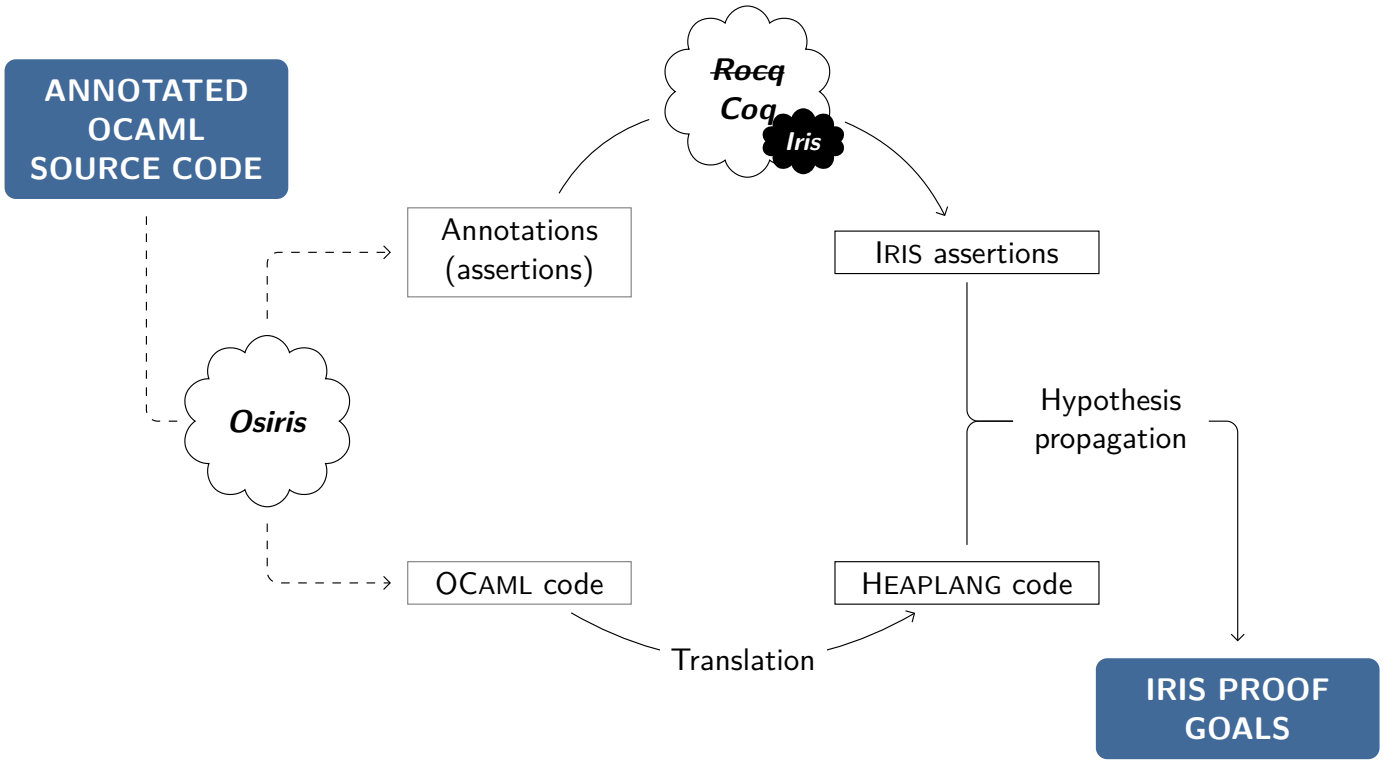

ANNOTATED
OCAML
SOURCE CODE
Osiris

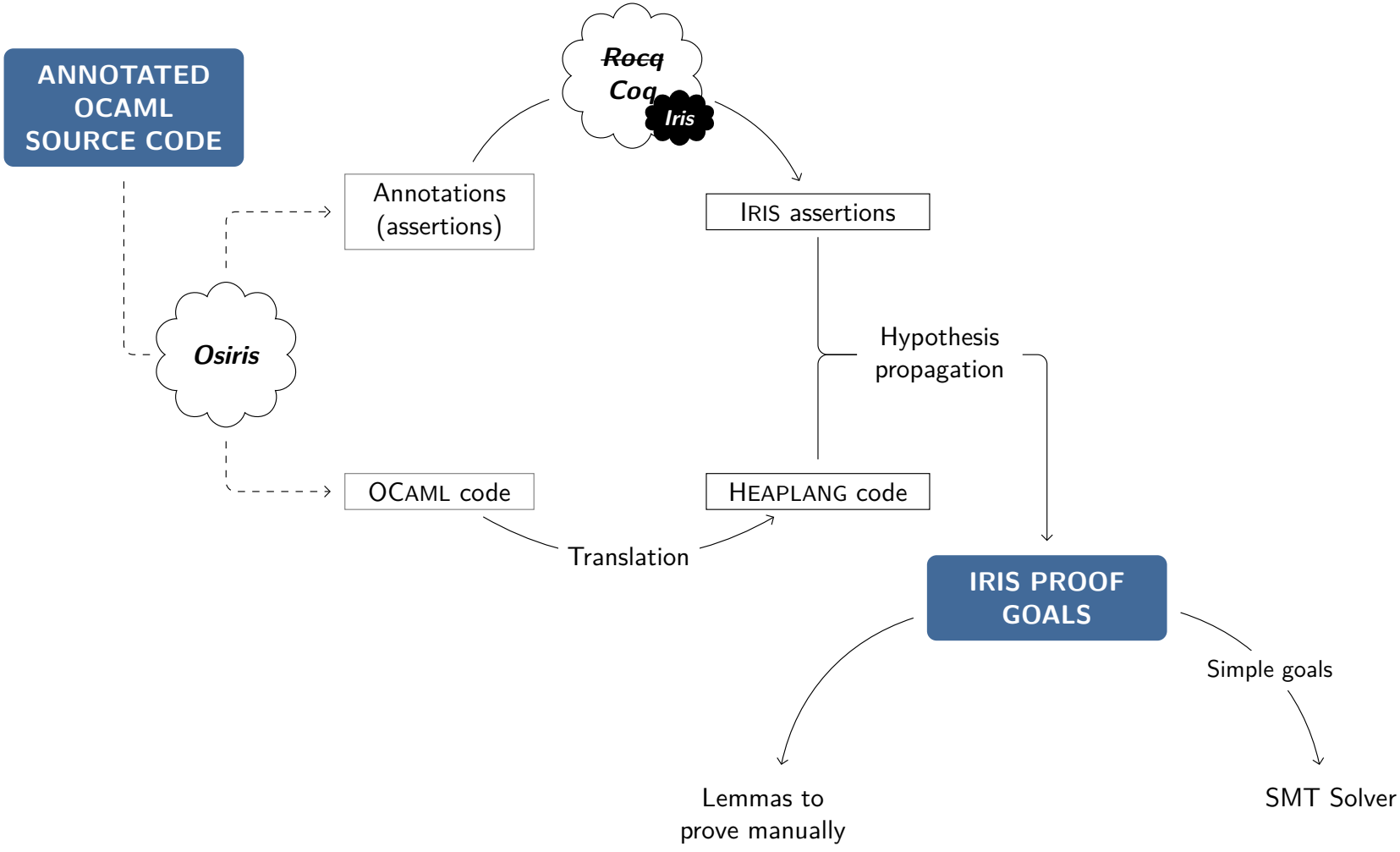

ANNOTATED
OCAML
SOURCE CODE
Annotations
(assertions)
Osiris
OCaml code

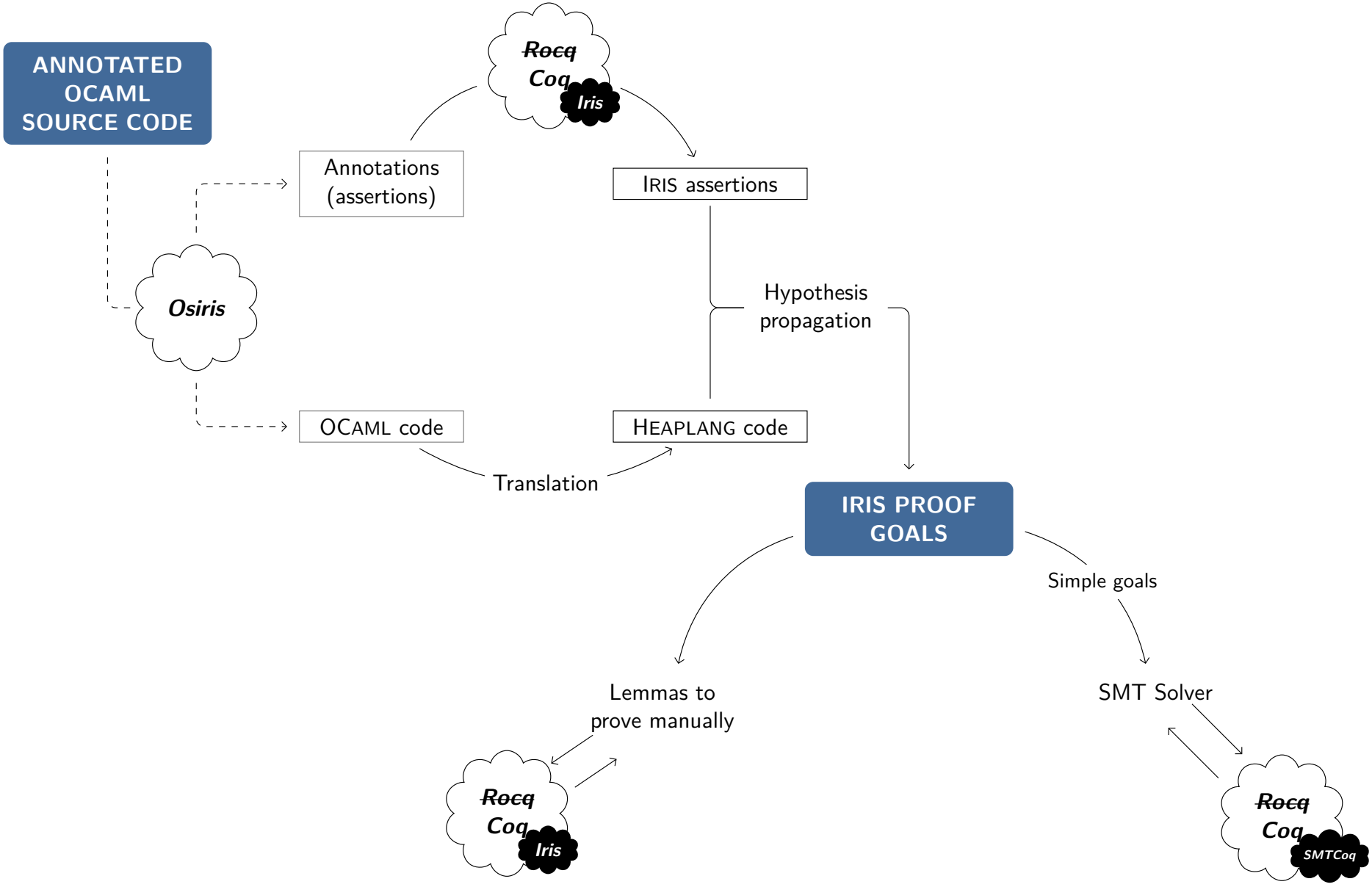

ANNOTATED
OCAML
SOURCE CODE
Rocq
Coq
Iris
Annotations
(assertions)
IRIS assertions
Osiris
OCAML code

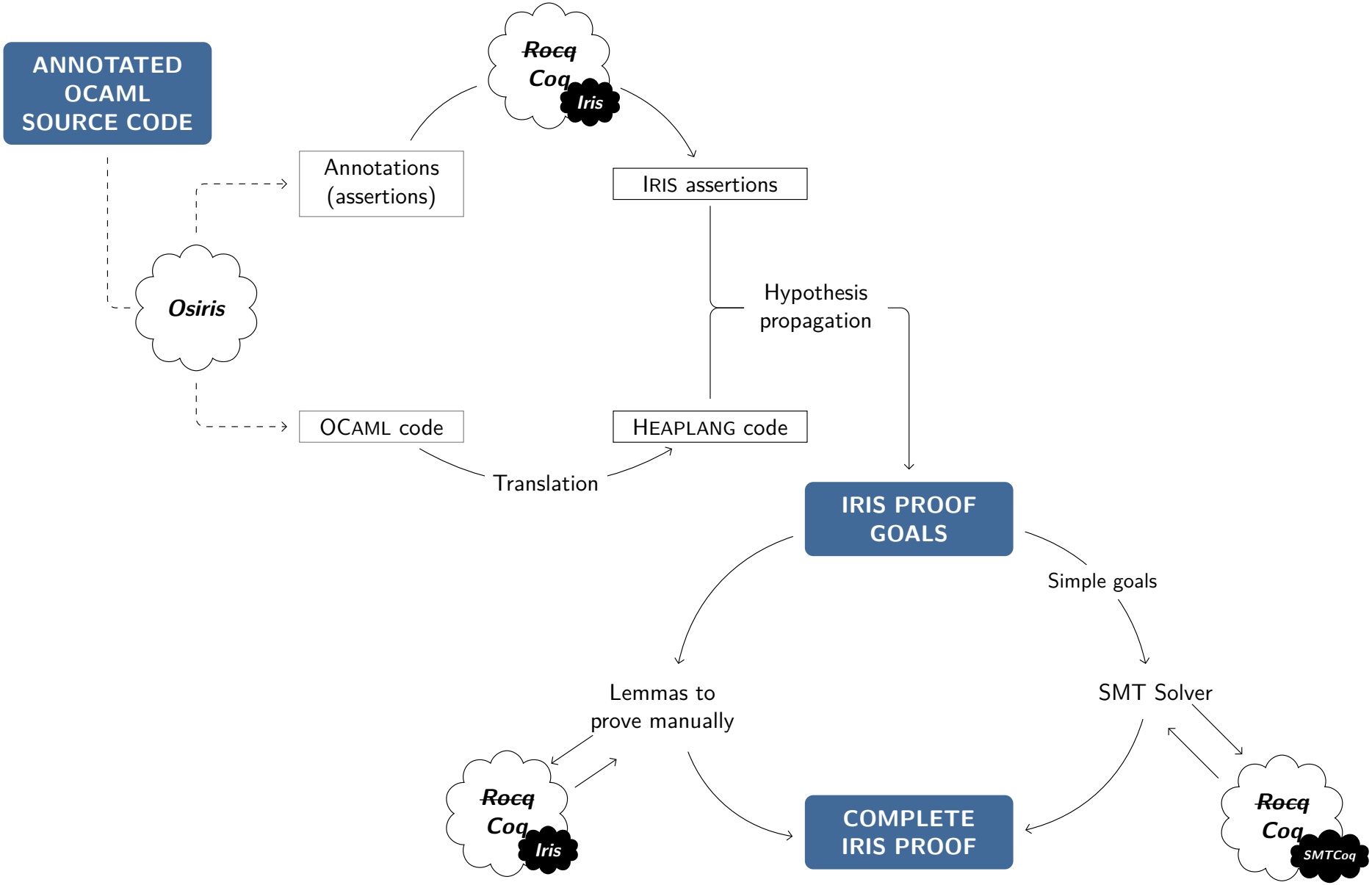

ANNOTATED OCAML SOURCE CODE
~~Rocq~~
Coq
Iris
Annotations (assertions)
IRIS assertions
Osiris
OCAML code
HEAPLANG code
Translation


ANNOTATED OCAML SOURCE CODE
~~Rocq~~
Coq
Iris
Annotations (assertions)
IRIS assertions
Osiris
Hypothesis propagation
OCAML code
HEAPLANG code
Translation


ANNOTATED OCAML SOURCE CODE
~~Rocq~~
Coq
Iris
Annotations (assertions)
IRIS assertions
Osiris
Hypothesis propagation
OCAML code
HEAPLANG code
Translation
IRIS PROOF GOALS


ANNOTATED OCAML SOURCE CODE
Rocq
Coq
Iris
Annotations (assertions)
IRIS assertions
Osiris
Hypothesis propagation
OCAML code
HEAPLANG code
Translation
IRIS PROOF GOALS
Simple goals
Lemmas to prove manually
SMT Solver


ANNOTATED OCAML SOURCE CODE
Rocq
Coq
Iris
Annotations (assertions)
IRIS assertions
Osiris
Hypothesis propagation
OCAML code
HEAPLANG code
Translation
IRIS PROOF GOALS
Simple goals
Lemmas to prove manually
SMT Solver
Rocq
Coq
Iris
Rocq
Coq
SMTCoq


ANNOTATED OCAML SOURCE CODE
Rocq
Coq
Iris
Annotations (assertions)
IRIS assertions
Osiris
Hypothesis propagation
OCAML code
HEAPLANG code
Translation
IRIS PROOF GOALS
Simple goals
Lemmas to prove manually
SMT Solver
Rocq
Coq
Iris
COMPLETE IRIS PROOF
Rocq
Coq
SMTCoq

A Small Program

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

Listing. *A simple annotated OCaml program*

A Small Program

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

Listing. *A simple annotated OCaml program*

```
⊢  ∀z : val,
   (⌜(#1 ≤ᵥ #2) = (z ≤ᵥ #2)⌝
   ∗ ⌜(z ≤ᵥ #2) = #false⌝)
   −∗ ⌜#1 = #3⌝
```

Listing. *The Iris proof obligation generated*

Loops and Complex Proofs

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

let [@post "x ↦ #12"] result =
  let x = ref 0 in
  while !x <= 10 do
    x := !x + 2;
  done
  [@invariant "∃ z : Z, x ↦ #z * ⌜even z⌝"]
```

You are here

From OCaml to HeapLang

```
let result =
    let x = ref 1 in
    if !x <= 2
        then x := 3;
    !x
```

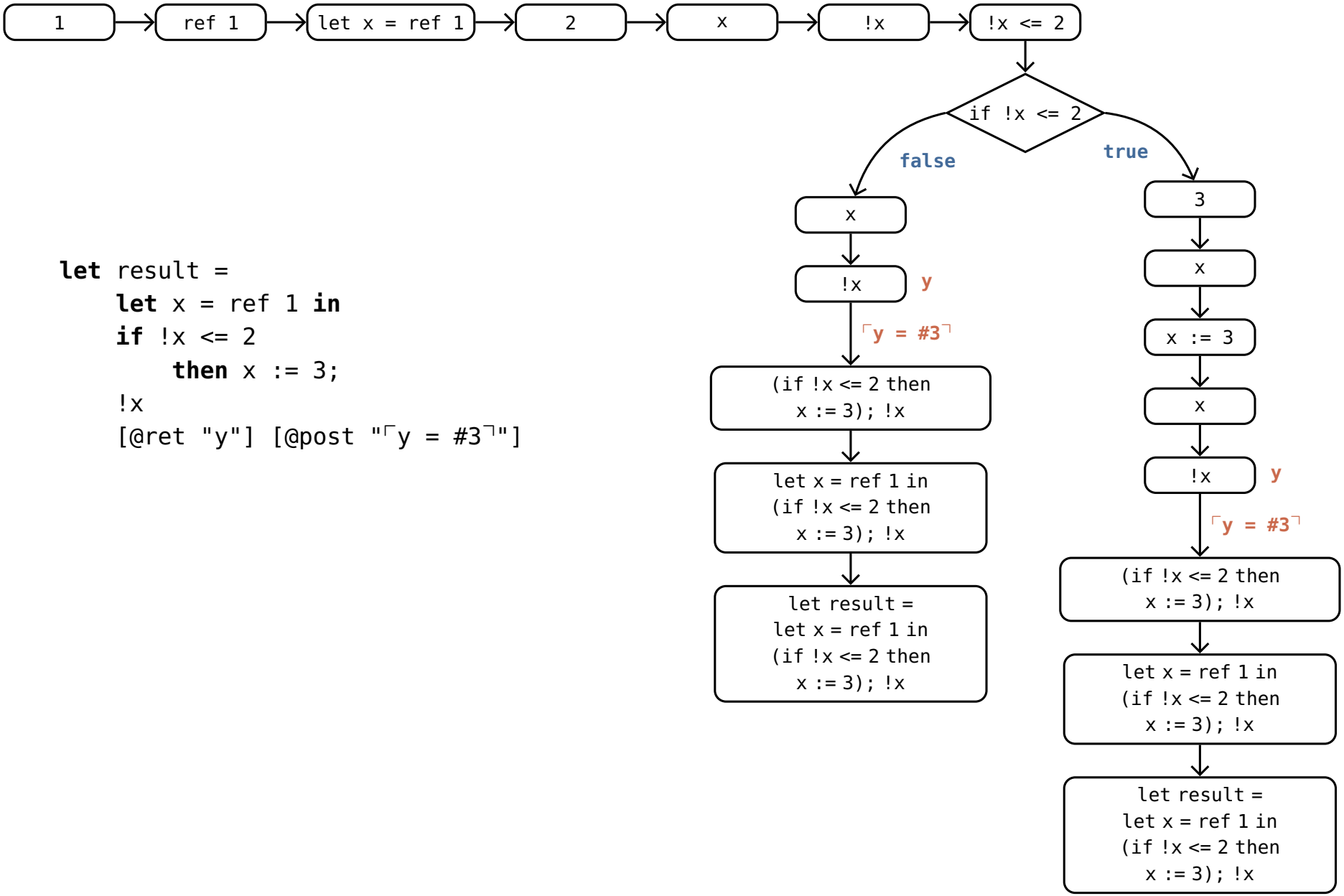
Listing. *A simple OCaml program*

```
Definition result : expr := (
    let: "x" := AllocN #1 #1 in
    if: (!"x" ≤ #2) then (
        "x" <- #3
    ) else (
        #()
    );;
    !"x"
).
```

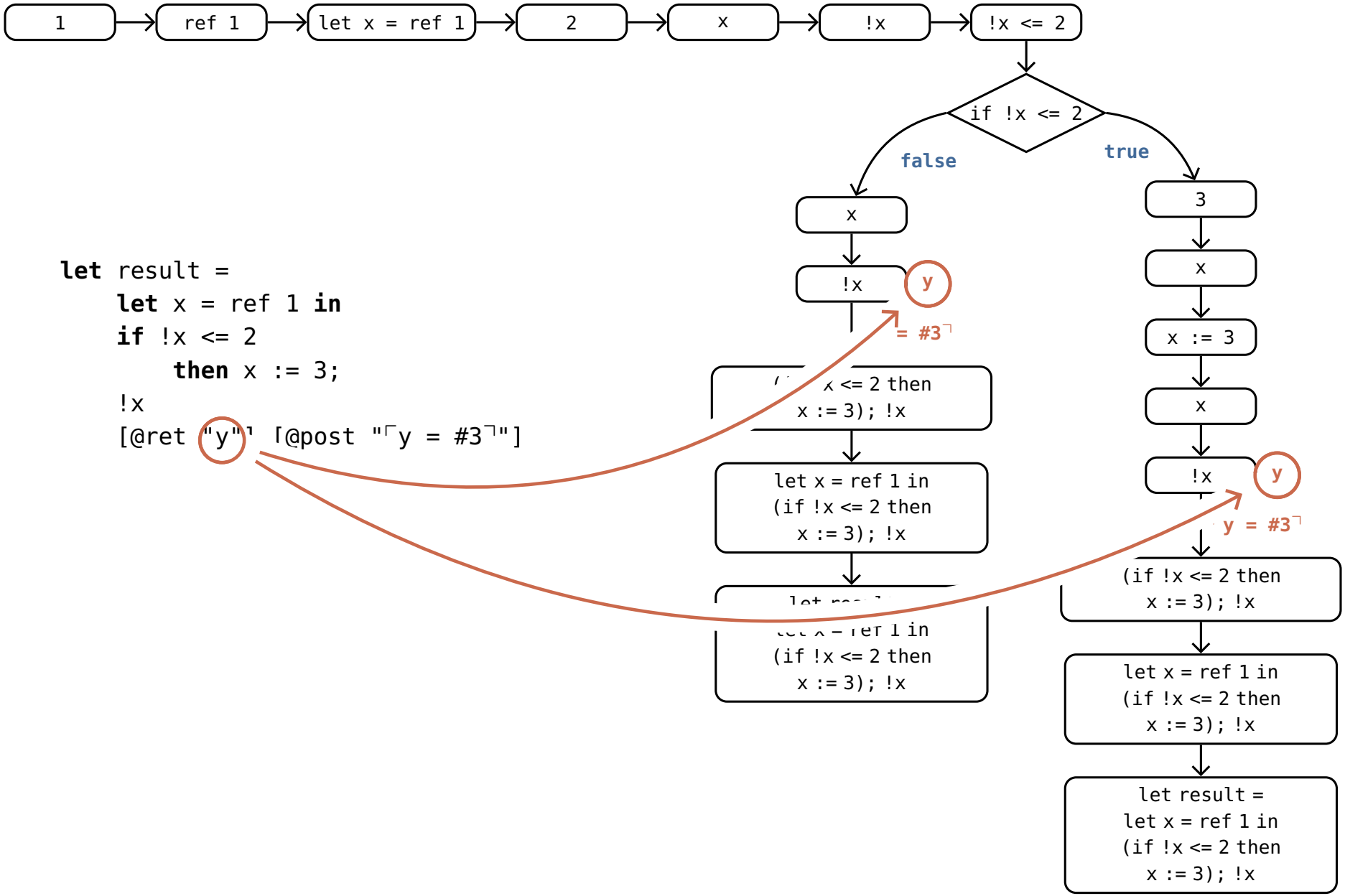
Listing. *Generated HeapLang code*

Control Flow Tree (CFT)

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



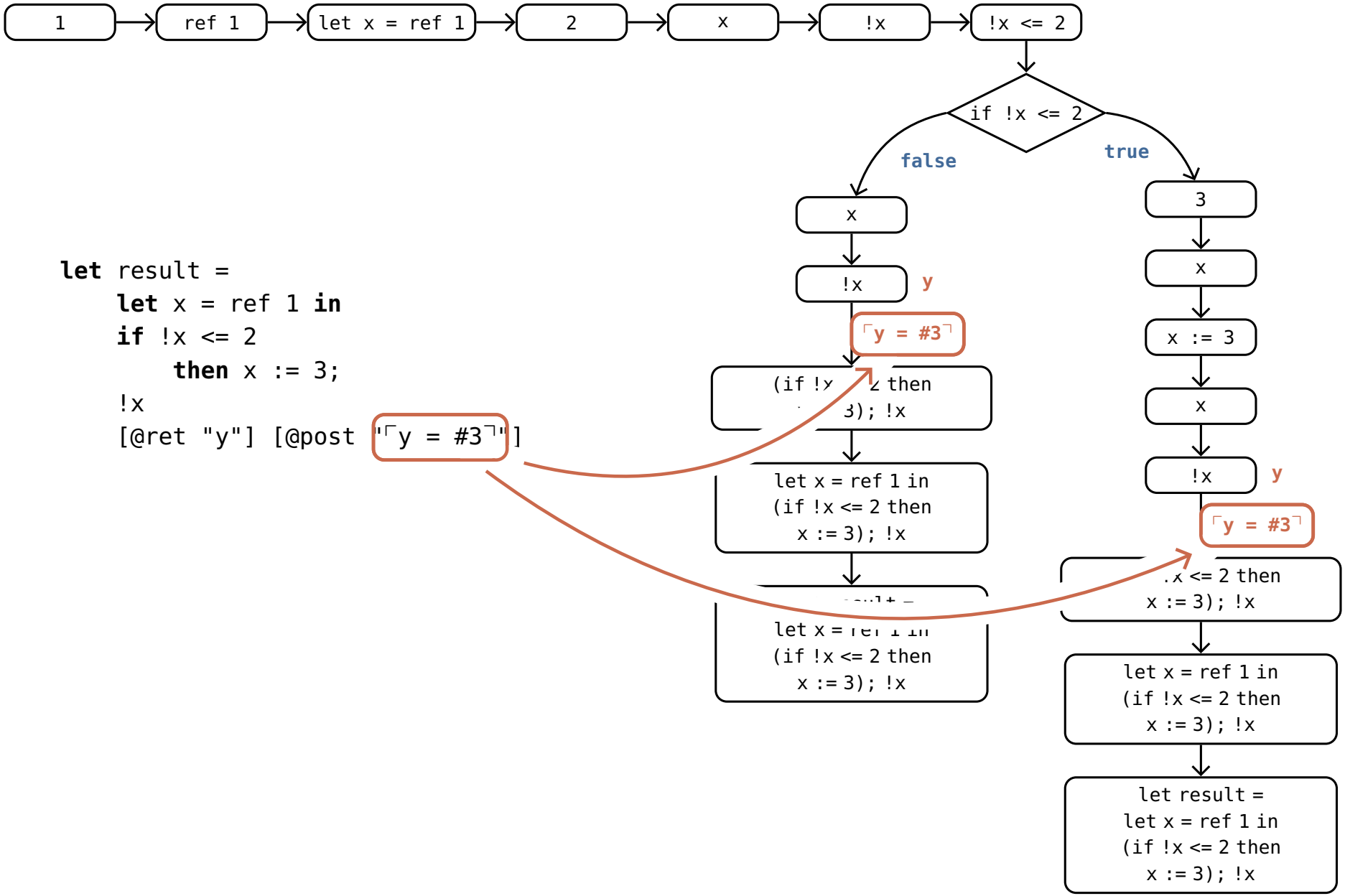
Control Flow Tree (CFT)



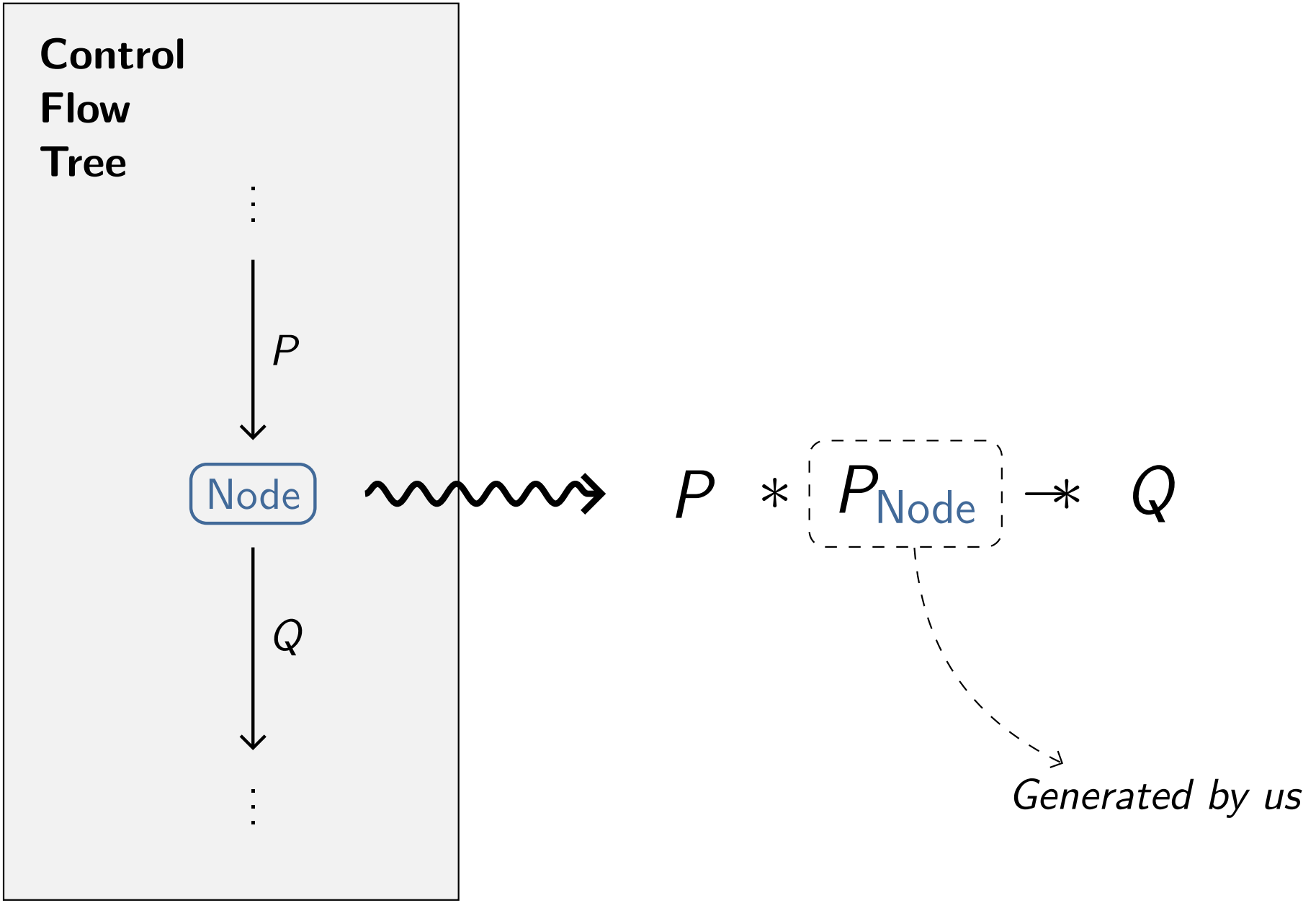

```
let result =
    let x = ref 1 in
    if !x <= 2
        then x := 3;
    !x
    [@ret "y"] [@post "⌜y = #3⌝"]
```

Control Flow Tree (CFT)

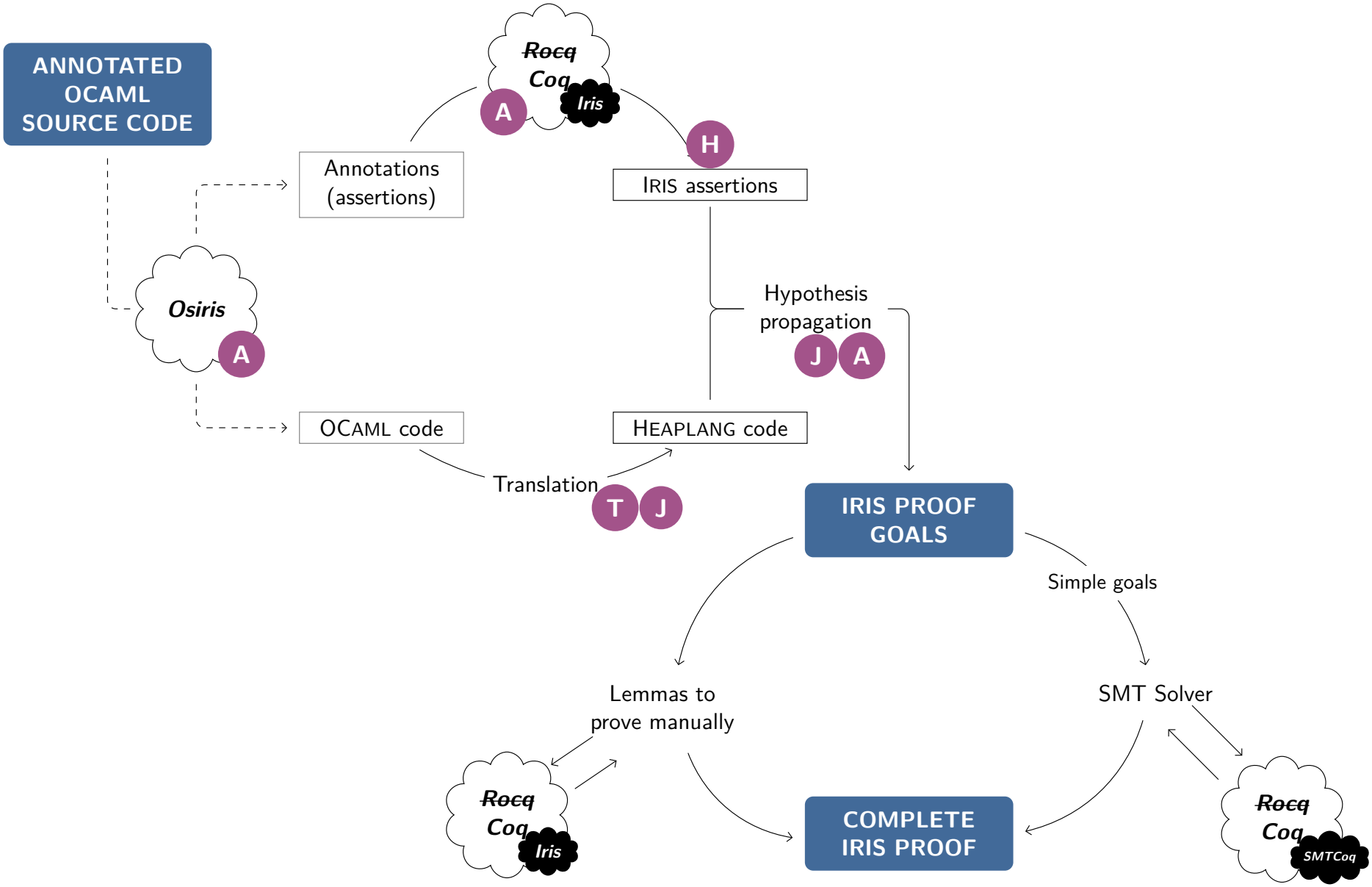
```
let result =
    let x = ref 1 in
    if !x <= 2
        then x := 3;
    !x
    [@ret "y"] [@post "⌜y = #3⌝"]
```



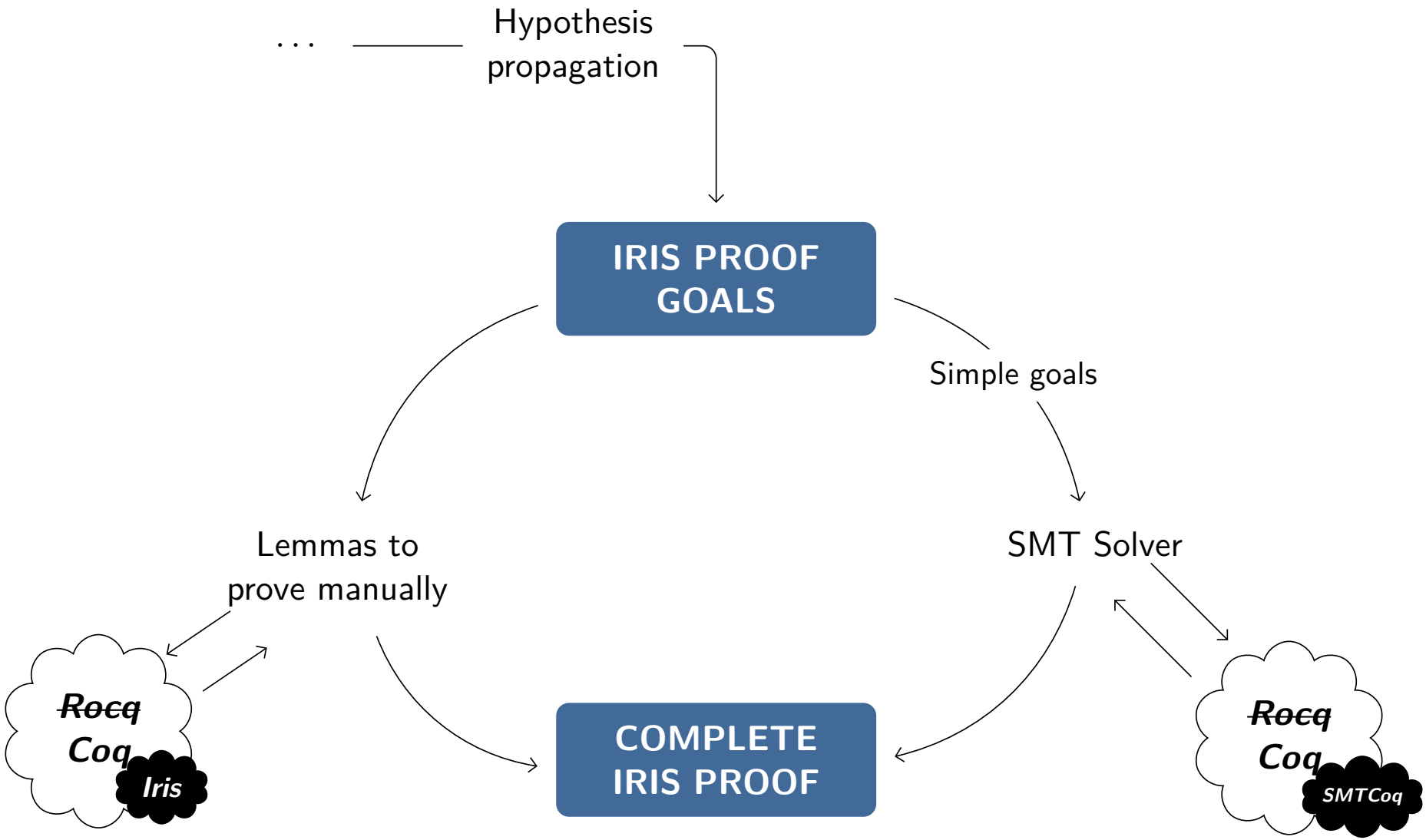
Obligation Generation



Team Organisation



You are here


...
Hypothesis propagation
IRIS PROOF GOALS
Simple goals
Lemmas to prove manually
SMT Solver
Rocq
Coq
Iris
COMPLETE IRIS PROOF
Rocq
Coq
SMTCoq

[All25] Clément Allain. “Zoo: A framework for the verification of concurrent OCaml 5 programs using separation logic”. In: *Journées Françaises des Langages Applicatifs (JFLA)*. Jan. 2025. URL: https://clef-men.github.io/publications/allain-25.pdf.

[Sea+25] Remy Seassau et al. “Formal Semantics and Program Logics for a Fragment of OCaml”. In: *Proceedings of the ACM on Programming Languages* 9.ICFP (Aug. 2025). URL: http://cambium.inria.fr/~fpottier/publis/seassau-yoon-madiot-pottier-osiris-2025.pdf.