From CompCert to Concurrent Cminor or: separation logic in Coq

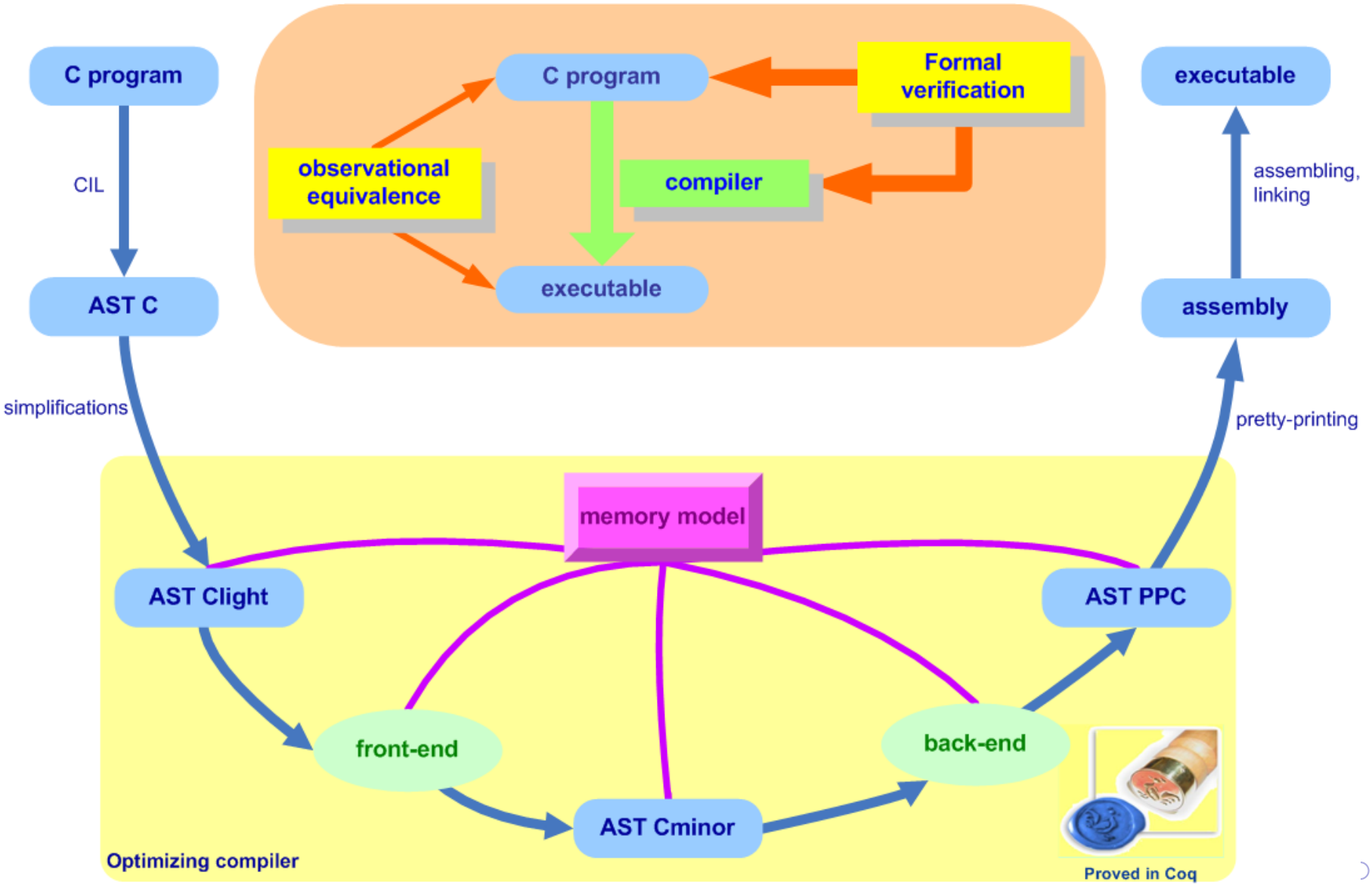
Sandrine Blazy

ensiie

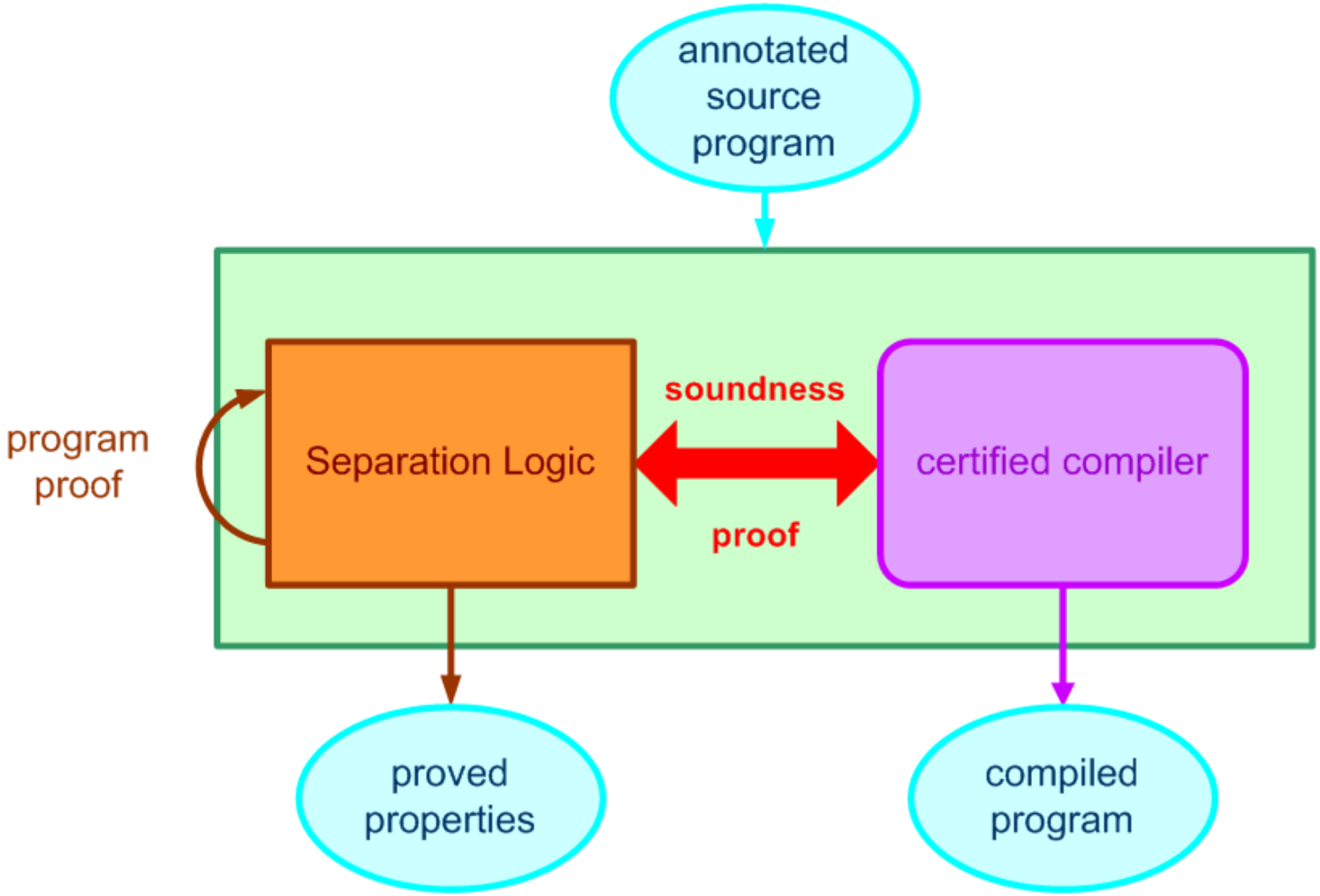
Separation logic meeting, Princeton University

The CompCert certified compiler in one slide

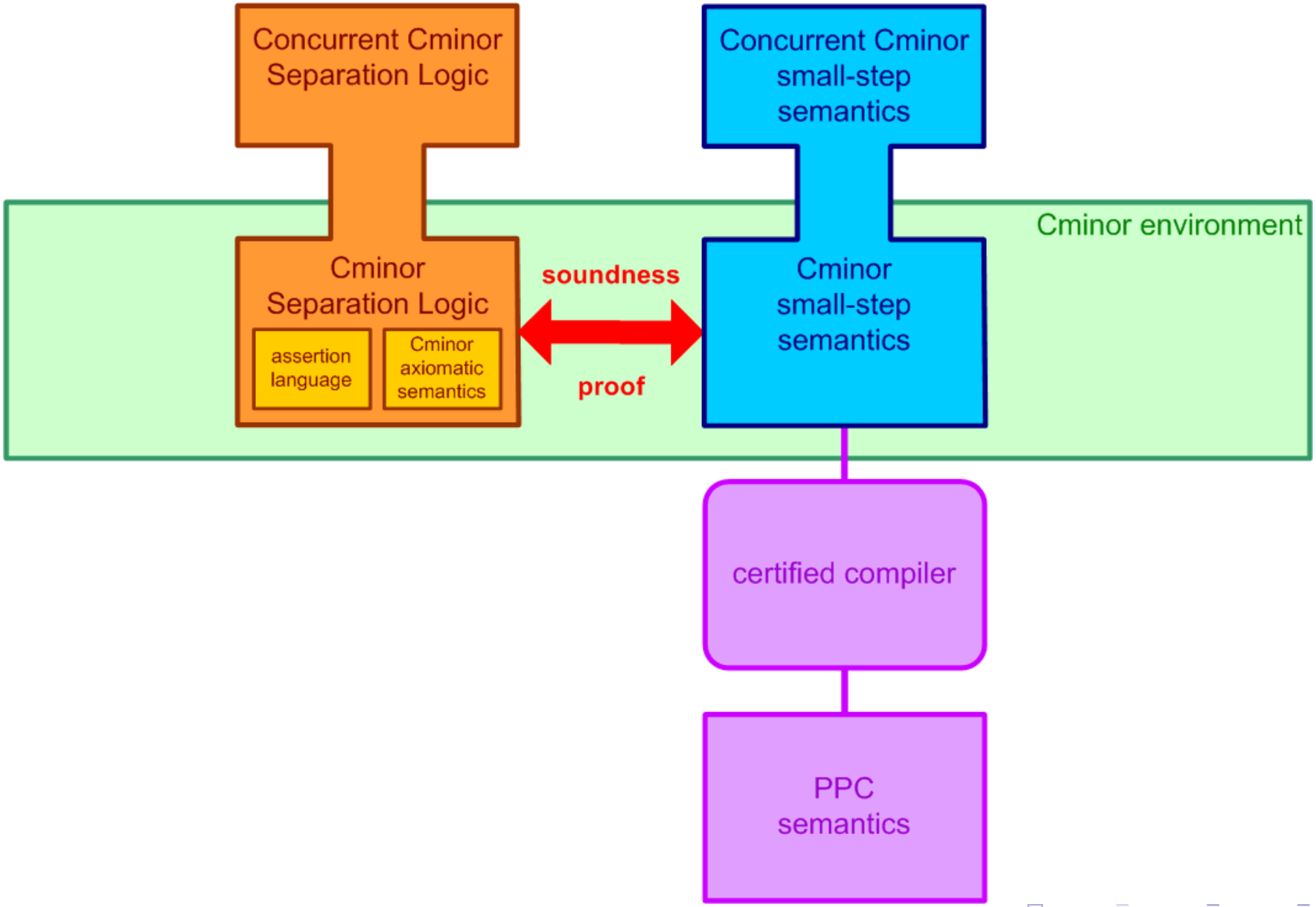
compcert.inria.fr



Proved-correct software



Separation logic and certified compilation (see TPHOL'2007 paper)



On-going work: a separation logic for Clight

Reuse of the separation logic for Cminor:

- same assertion language
- tactics for separation logic

The Clight language

- no goto statement
- realistic memory model that is byte- and word- addressable.
- pointer arithmetic within any malloc'ed block.
  - pointer = (block reference, byte offset)
  - expressions can evaluate to Vundef without getting stuck

Related work: the Caduceus tool for proving C programs
(see the first invited talk at SMT, by J.C. Filliatre, caduceus.lri.fr)

- no separation logic
- more abstract memory model
- most of the proofs are discharged to first-order automatic provers

Tactics

Coq is an interactive theorem prover

Tactics for separation logic:
`forward`, `assert_subst`, `sep_trivial` (see Andrew's paper)

Other tactics
e.g. symbolic evaluation of expressions

But, tactics are not enough.

Connection with automatic theorem provers

Recent redesign of the memory model at several levels of abstraction

Can the proofs be automated using a theorem prover for first-order logic ?

Use of the Why platform for program proof.

Experiments with 3 automated theorem provers (Ergo, Simplify, Z3).

The translations are done almost automatically by Why:

- from the higher-order logic of Coq into first-order logic,
- between the input syntaxes of the provers.
- Only the maps had to be axiomatized.

Connection with automatic theorem provers

Redesign of the memory model: results

Given the axiomatization of a relation defined in the memory model (*e.g.* `valid_block`), its derived properties are proved automatically.

Of 50 goals, 42 were proved by at least one of the 3 provers (5-min time limit of CPU time).

While interactive proof remains necessary for some of the most difficult theorems, integration of first-order theorem proving within a proof assistant has great potential to significantly shorten our proofs.

Connection with the Ergo automatic theorem prover

We chose Ergo to automate our proofs in separation logic.

Ergo is a typed theorem prover, dedicated to program verification.

Some properties were proved automatically by Ergo.

But, we encountered some difficulties:

- the translation from Coq to Ergo failed
  (no inductive definition in Ergo).
- Ergo does too many instantiations
  (*e.g.* the variable n is declared as an `int` but its values $\in [1; 10]$).
- A helpfull annotation may confuse a decision procedure.