MODULOGIC

Atelier de construction modulaire de logiciels certifiés
Application à la sécurité des systèmes informatiques

ACI Sécurité 2003

LIP6-CEDRIC-LOGICAL-MIRO-PROTHEO

T. HARDIN - LIP6

Ingredients

- Atelier FOC, ELAN, TOM, Cariboo, Coq

- Rho-calculus
- Deduction modulo
- Formal Specification of FOC with Coq ….

- Experience of building a huge library for symbolic computation

Objectives

- To develop a **programming environment** using these tools, in order to specify, to conceive, to code **trusted** software

- To develop a library dedicated to **formal** specification and implementation of **security policies**

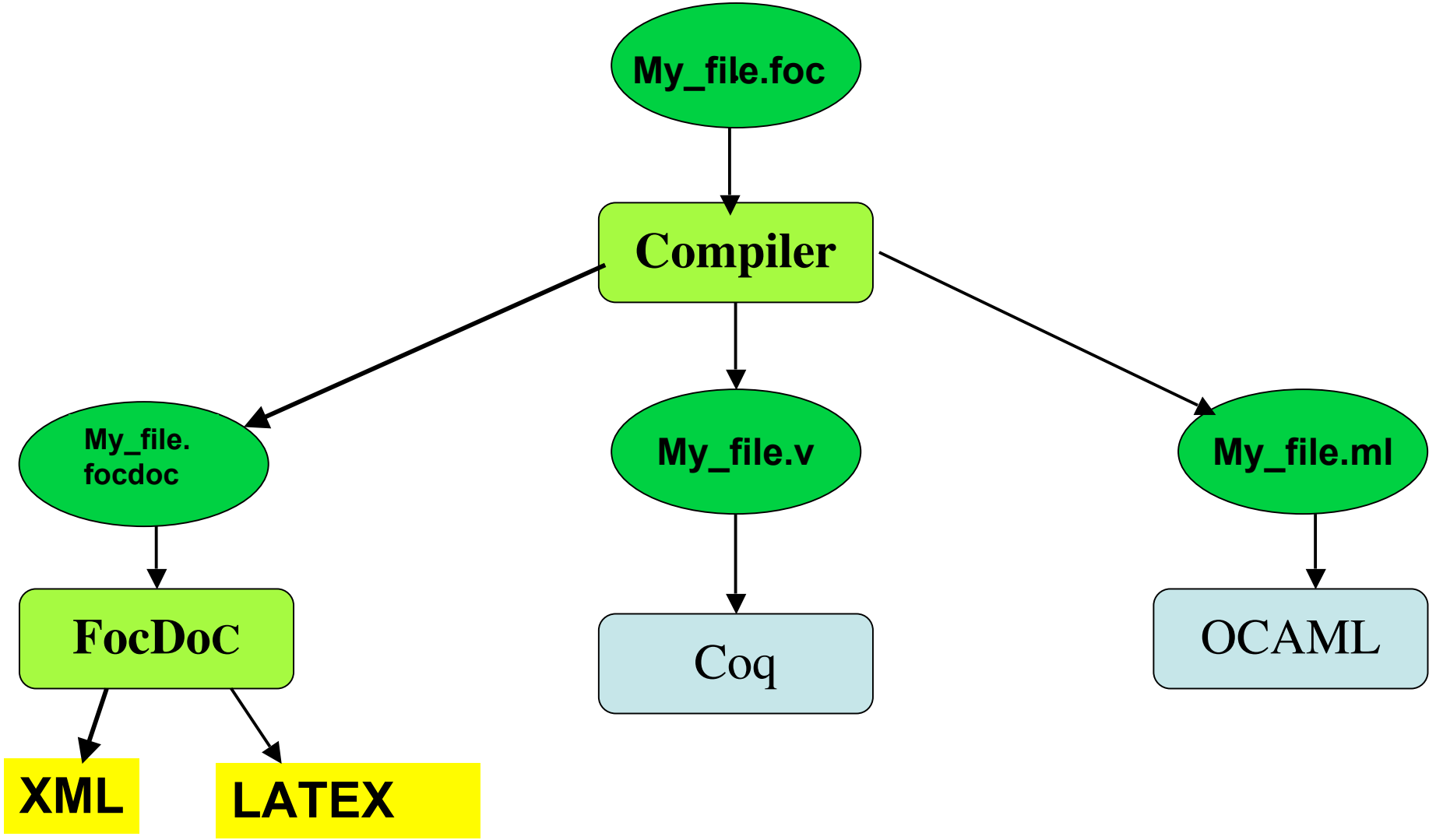
Results

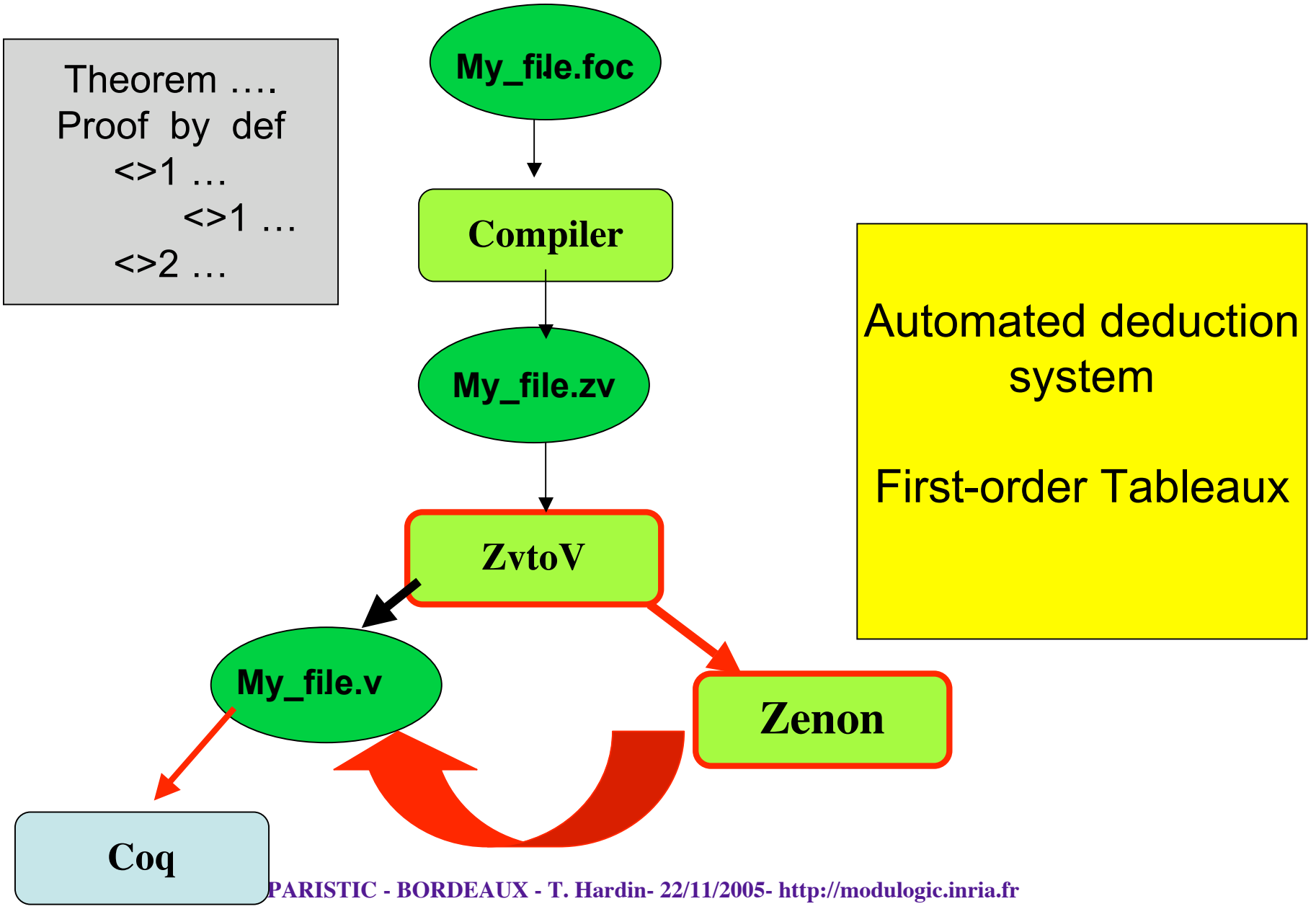
- **Distribution of CARIBOO (weak termination) and Development of TOUNDRA (completness)**

- **Extension of FOC ⇒ FOCAL**
- **Formal model of security policies : properties and implementation**
- **Coding with FOCAL of ACI *Edemoi example*, translation to UML**
- **Progress on deduction modulo, rho-calculus, operational semantics of FOCAL, …**

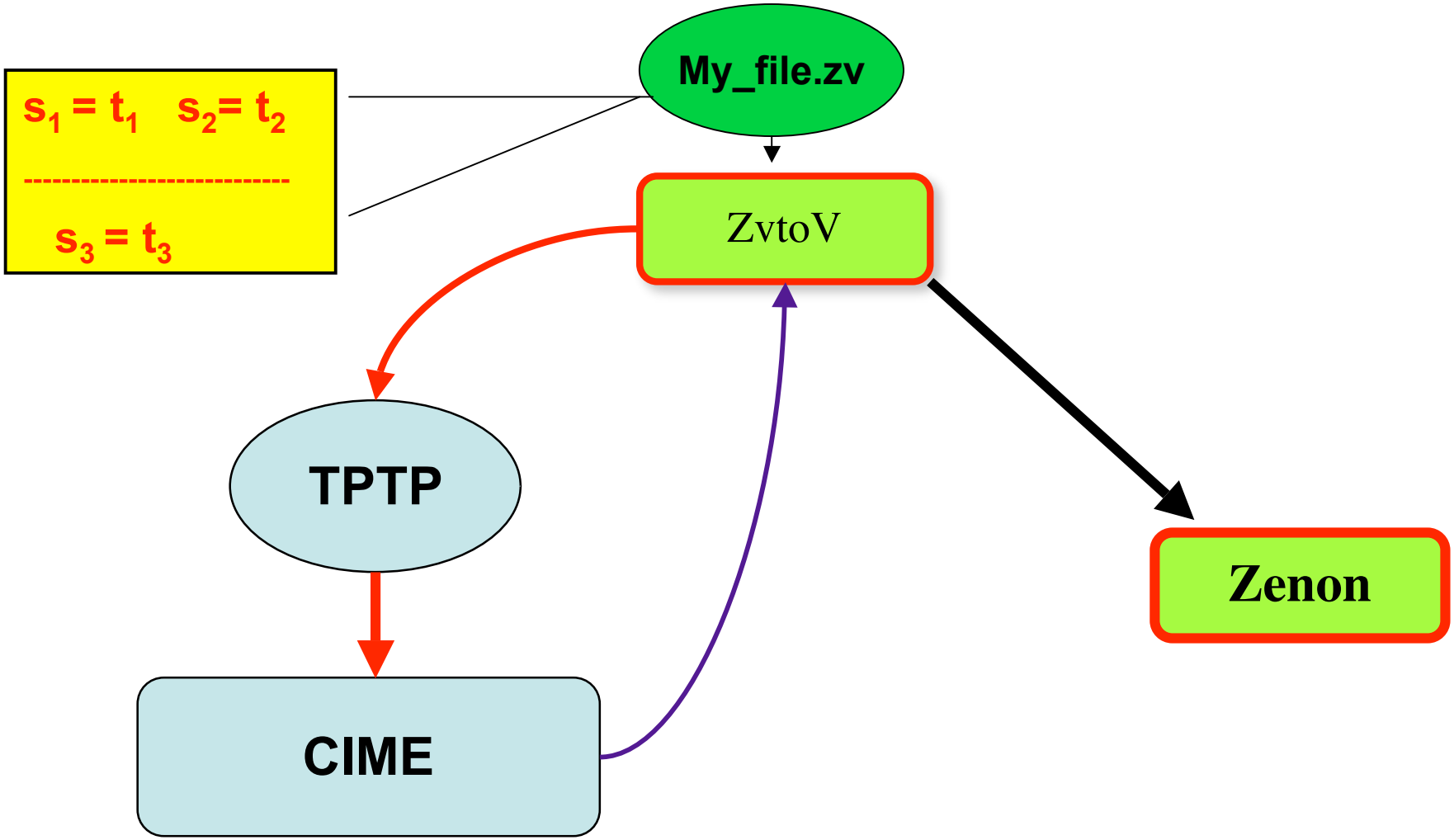
FOCAL

- Species : a carrier, declarations/definitions, properties/proofs
- Collection : Abstraction of the carrier of a *complete* species *(all defined and proved)*
- Multiple inheritance, late binding and redefinitions, parametrization by collections and values
- http://focal.inria.fr

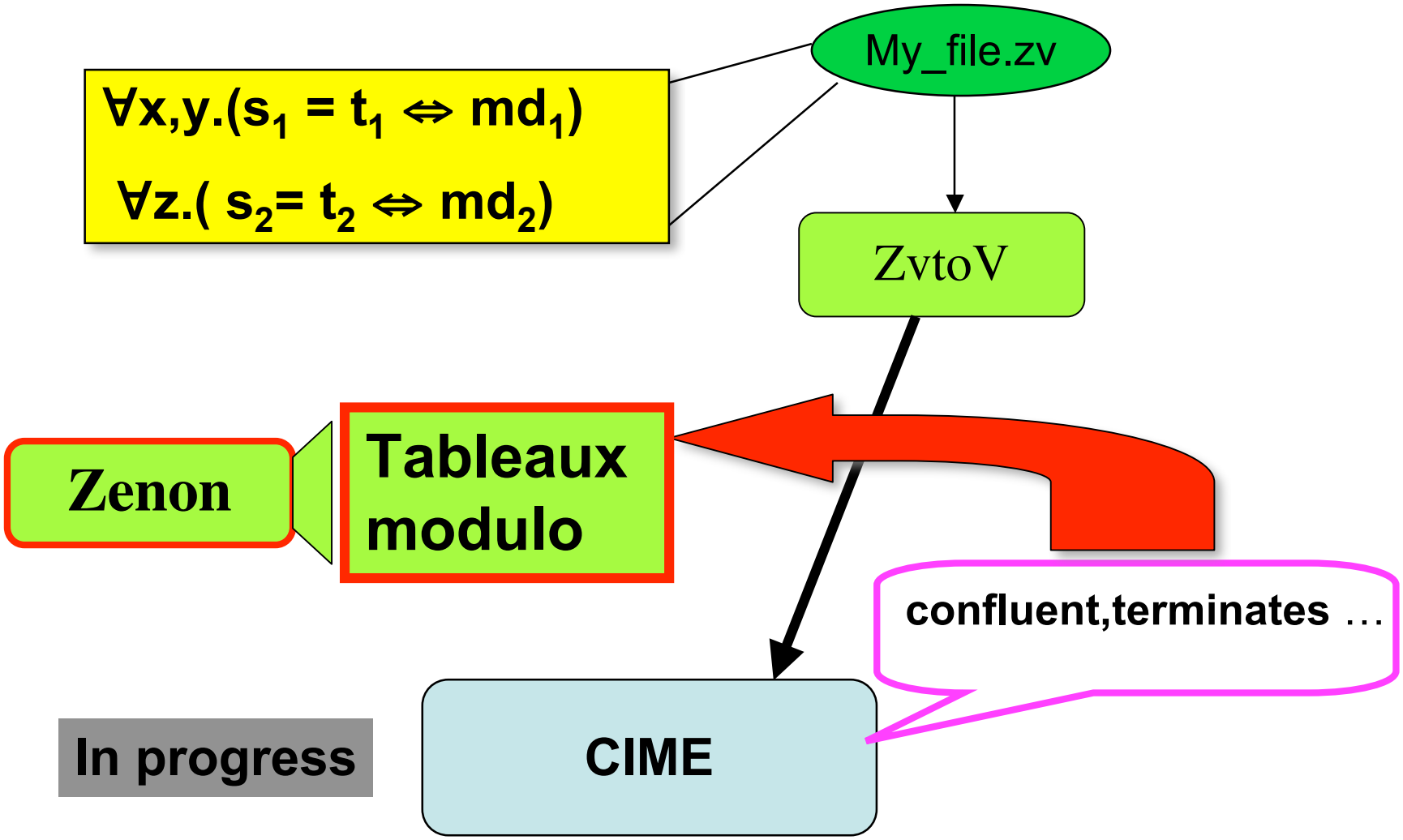

My_file.foc
Compiler
My_file.
focdoc
My_file.v
My_file.ml
FocDoC
Coq
OCAML
XML
LATEX


Theorem ….
Proof by def
<>1 …
<>1 …
<>2 …
My_file.foc
Compiler
My_file.zv
ZvtoV
My_file.v
Zenon
Coq
Automated deduction system
First-order Tableaux

Verification of equational lemmas with CIME



Deduction Modulo



PRIVATE TYPES (Ocaml)

```
type nat = N of int

(* type of positive integers*)
```

u+v is positive?

```
Let my-add = match (x,y) with
N u, N v → N (u + v)
```

PRIVATE TYPES

Module Nat

```
type nat =  PRIVATE  N of int

let build-nat p = if p>=0 then N p else failwith
"not positive"
(* a private construction of values of nat *)
```

Still Pattern-matching

```
use Nat;
let my-add =  match (x,y) with
N u, N v → try build-nat (u + v ) with….
```

use of N forbidden

Private Types with invariants:MOCA

Keywords to declare properties of data type constructors (Ass, AC, idempotent, neutral, …)

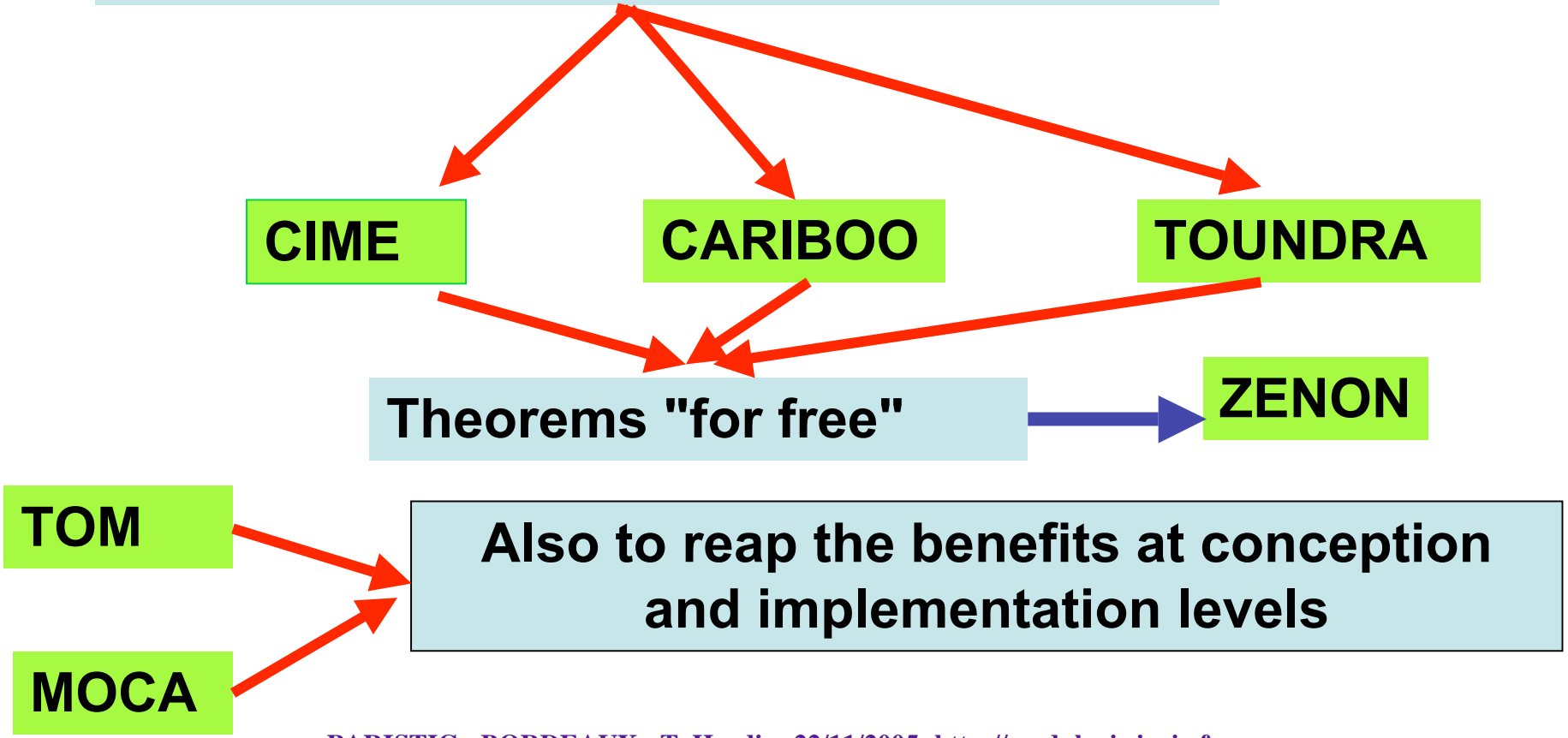
The compiler provides construction functions issuing only canonical forms.

What is the right notion of "canonical forms"?
How to guarantee invariants maintenance?

In progress

Further with equational reasoning

To describe parts of specifications with rewrite rules/equations



Conclusion

- A true common work between involved teams
- Theoretical results
- Issue of efficient tools, freely distributed
- Identification of users'needs through a methodological analysis of the development of a security framework done by academic and industrial teams.
- … In progress … Automatic generation of test cases, UML tools, tools to retrieve properties, …