Accelerated invariant generation

with Aspic and C2fsm

Paul Feautrier and Laure Gonnord

LIP(ÉNS Lyon)/LIFL(Univ. Lille)

http://laure.gonnord.org/pro/aspic
aspic@gonnord.org

Logo by Isabelle Borne

Goal

Computing invariants from C programs to

- prove correctness
- prove termination
- optimize compilers

Aspic

Invariant Computation

Aspic

Symbolic Invariant Computation

Aspic

Symbolic Polyhedral Invariant Computation

Aspic

Accelerated Symbolic Polyhedral Invariant Computation

Aspic and C2fsm : main characteristics

Aspic is **an invariant generator** :

- From counter automata with numerical variables.
- Invariants are **polyhedra**.

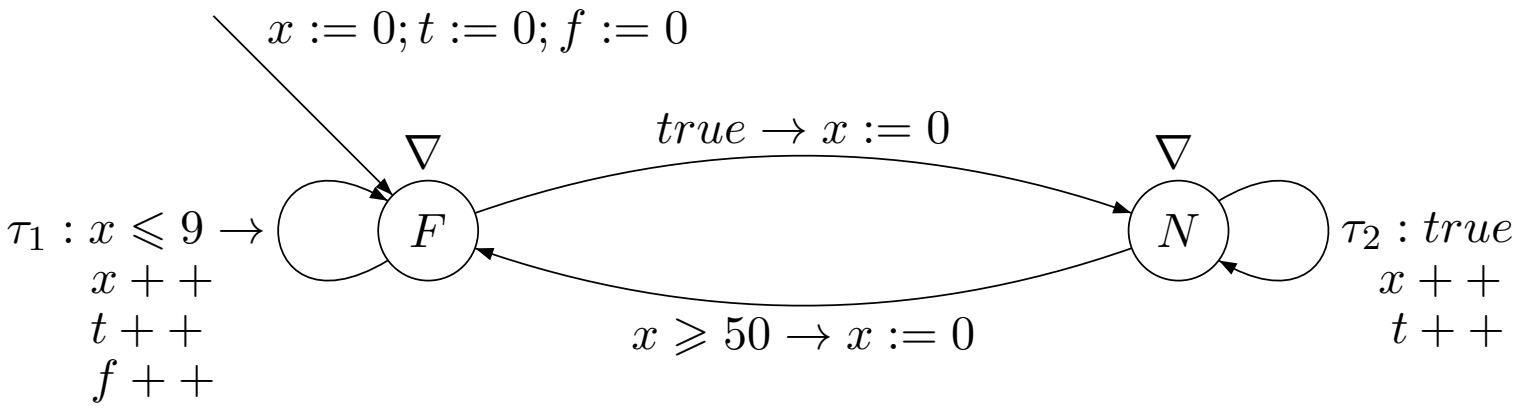
C2fsm is **a C parser** :

- From a source file in (a subset of) C into Aspic input language (fast).
- **Safe** abstractions of non numerical variables, structures, behaviors.

Aspic - Theoritical fundations (1)

Aspic implements a variant of Linear Relation Analysis :

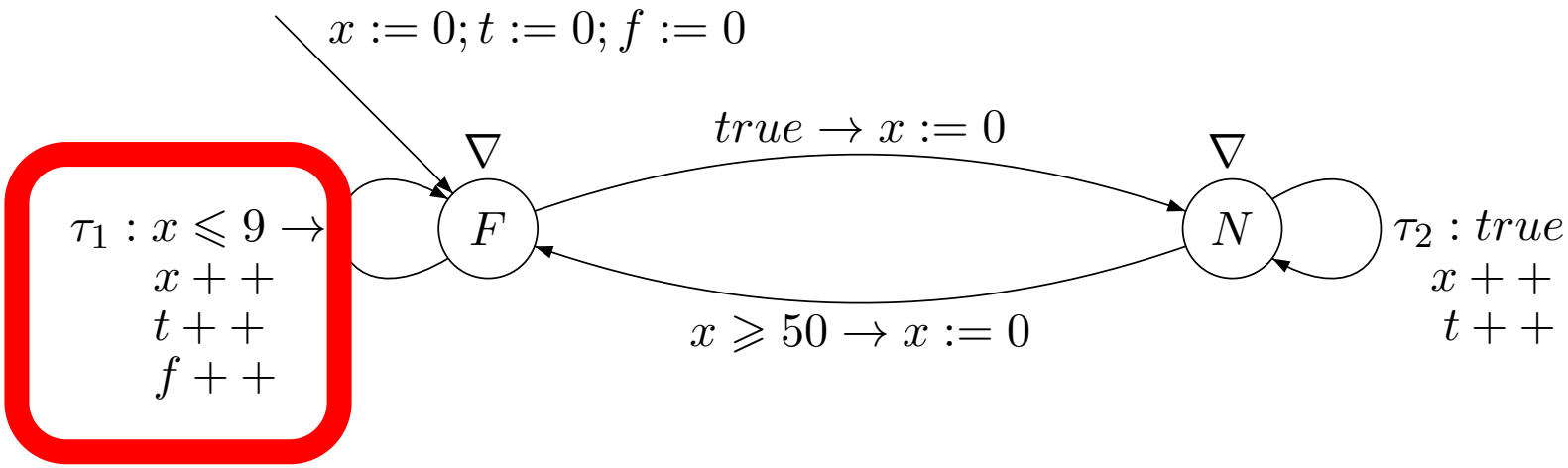
- Abstract interpretation with polyhedra on counter automata.
- Locally, the **exact** (abstract) reachability set is computed.



Aspic - Theoritical fundations (1)

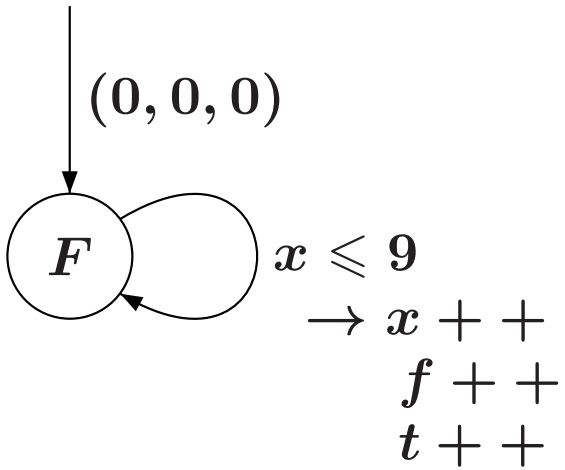
Aspic implements a variant of Linear Relation Analysis :

- Abstract interpretation with polyhedra on counter automata.
- Locally, the **exact** (abstract) reachability set is computed.



Aspic - Theoritical fundations (2)

Local **acceleration** :



- **exact effect** (Presburger Logic) :

$$\exists i \in \mathbb{N}, x = f = t = i, 0 \leqslant i \leqslant 10$$

- What is the **abstract exact effect** (rational polyhedron) ?

$$\{x = f = t, 0 \leqslant x \leqslant 10\}$$

Aspic - Theoritical fundations (3)

References :

- The notion of **abstract acceleration** was first defined in [Gonnord/Halbwachs,SAS2006].
- Large classes of accelerable loops are described in [Gonnord,Phd].

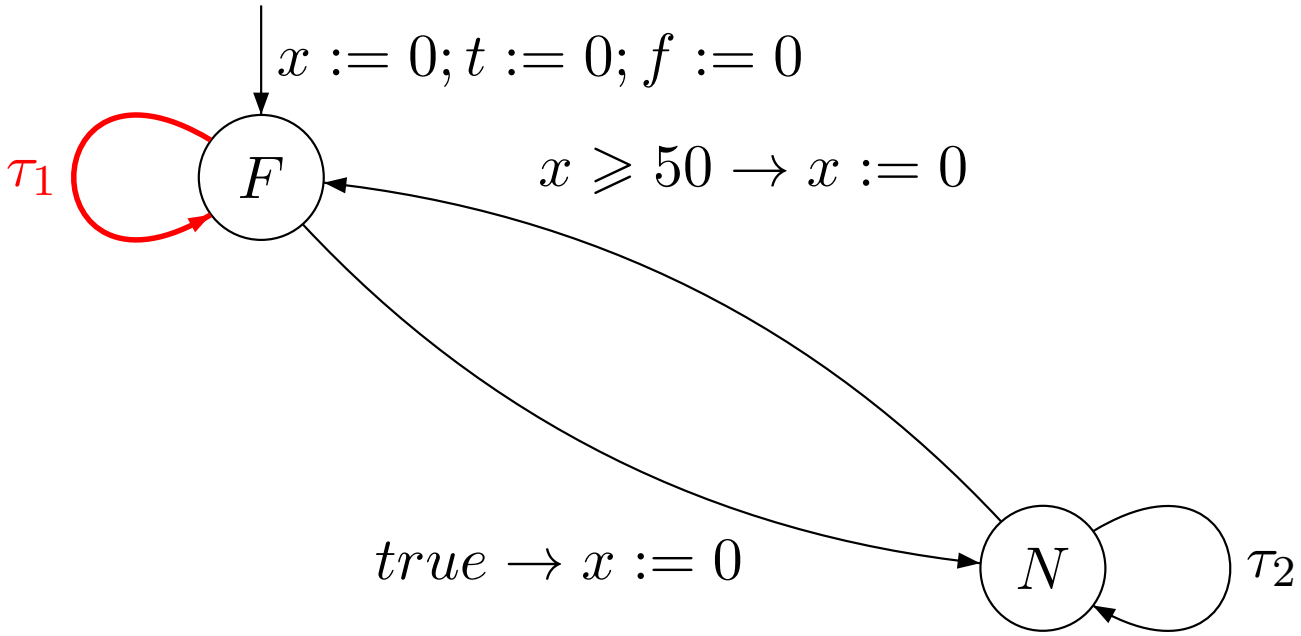
Algorithms and implementation :

- Acceleration algorithms for polyhedra, at low cost (only basic polyhedra operations).
- No computer arithmetic issue.
- Combination of acceration and classical LRA : graph issues, strategy issues, ...

▶ more details in the TAPAS paper and in [Gonnord, Phd].

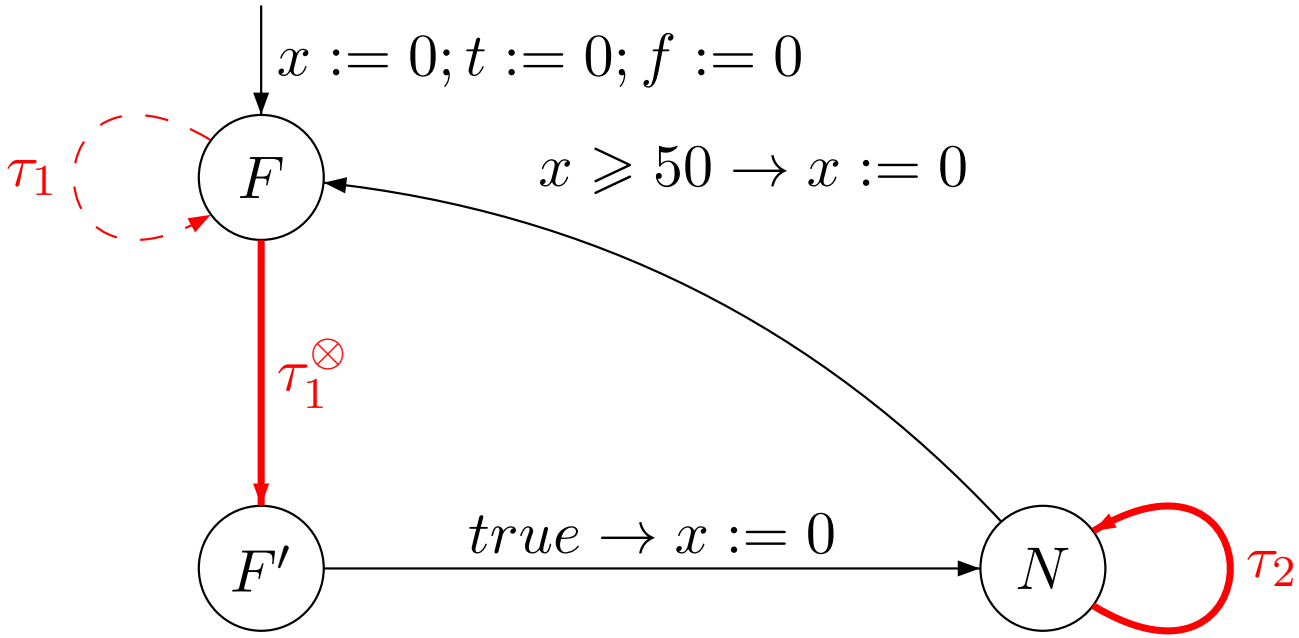
Aspic - acceleration

$\tau_1^{\otimes}$ is "add ray (1,1,1) to polyhedra"



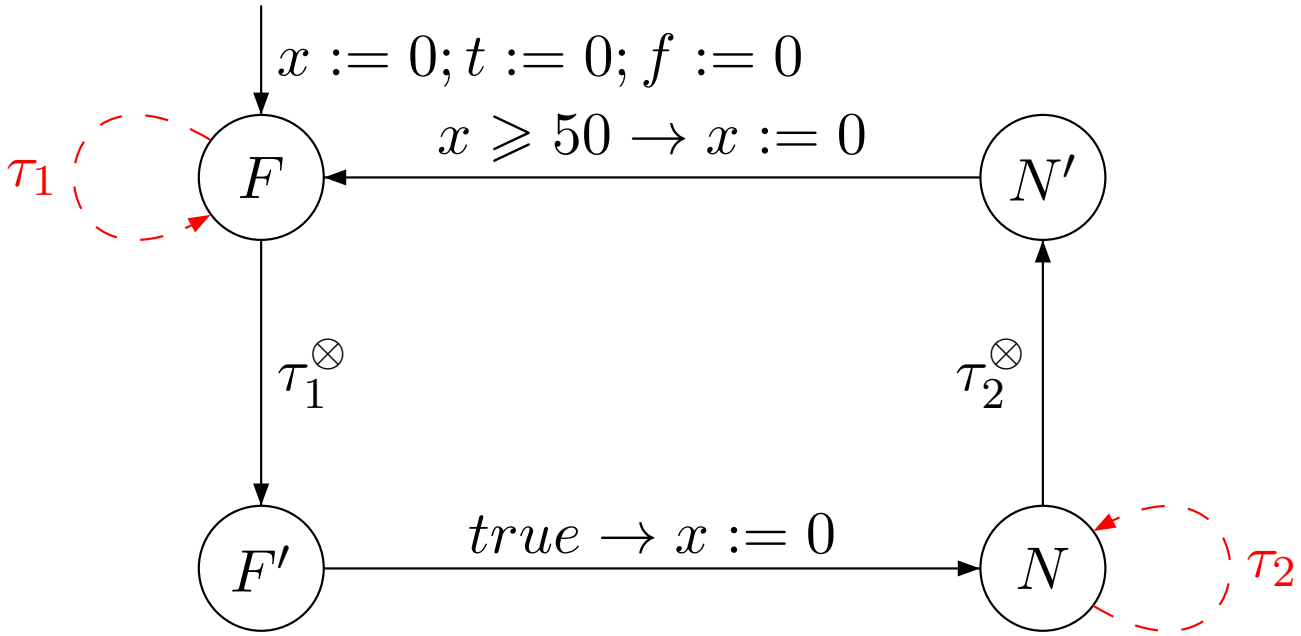
Aspic - acceleration

$\tau_1^{\otimes}$ is "add ray (1,1,1) to polyhedra"



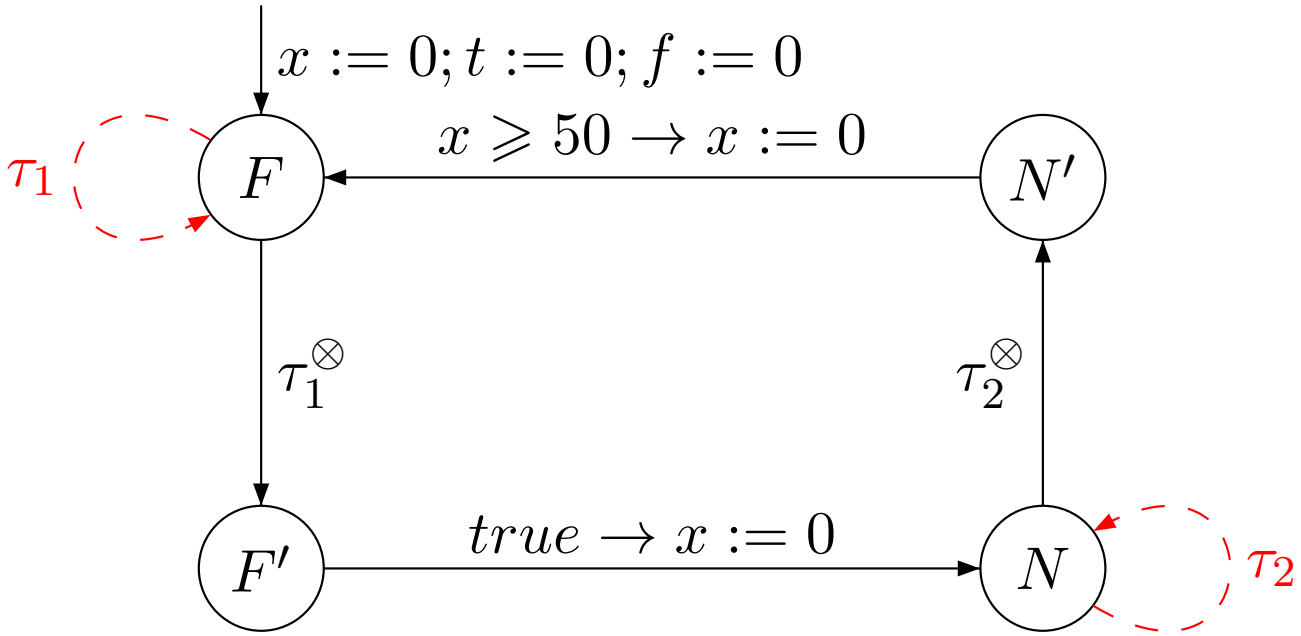
Aspic - acceleration

$\tau_1^{\otimes}$ is "add ray (1,1,1) to polyhedra"



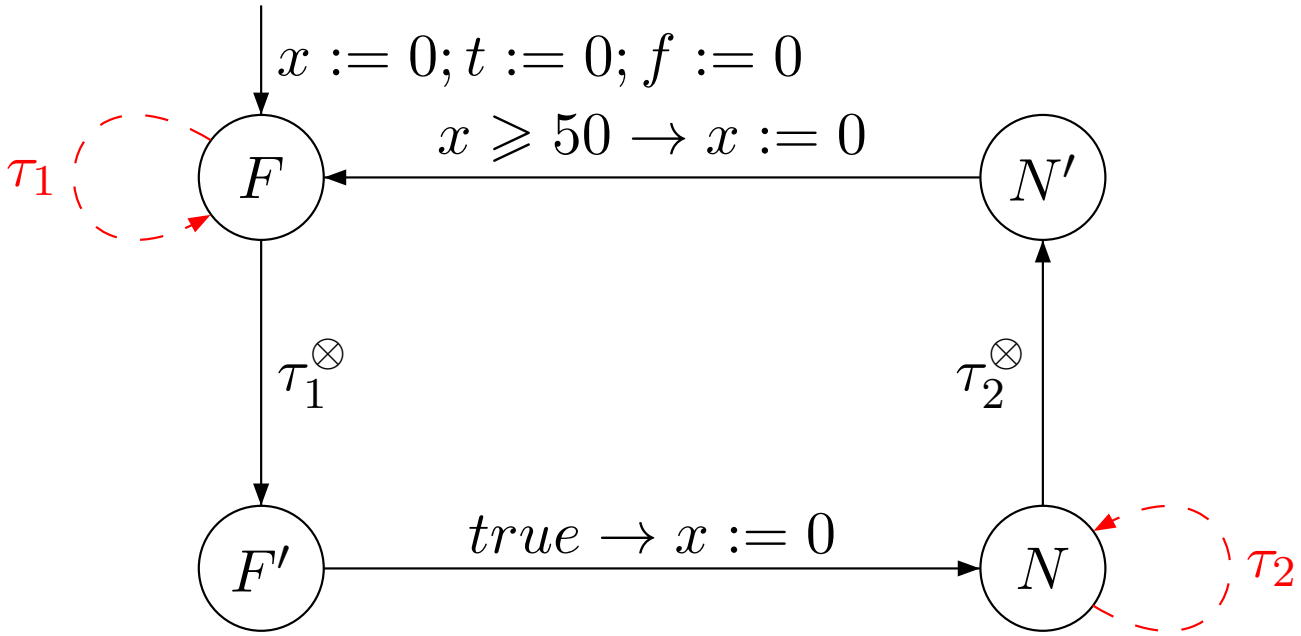
Aspic - acceleration

$\tau_1^{\otimes}$ is "add ray (1,1,1) to polyhedra"



Aspic - acceleration

$\tau_1^{\otimes}$ is "add ray (1,1,1) to polyhedra"



Aspic - Limitations

Theoritical limitations :

- Aspic only decides **safety**.
- Variables are rational ▶ no arithmetic issues.
- No boolean ▶ not designed to prove **protocols**.

Implementation limitations :

- Convex initial and bad regions/formulas.
- Aspic does not provide an API for the moment.

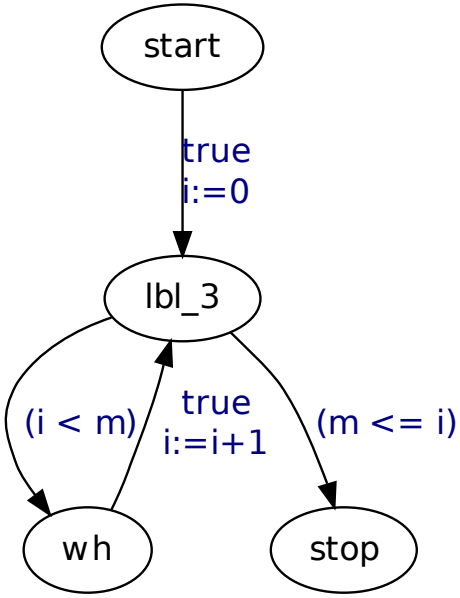
C2fsm

C2fsm is **more than** a C parser :

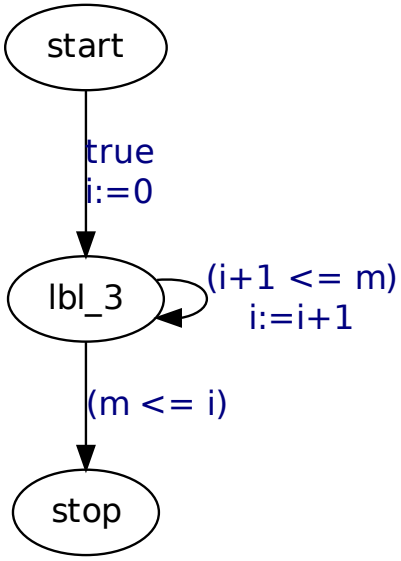
- It constructs an aspic input (automaton).
- When (safely) abstracting non affine conditions, it tries to keep as much useful information as possible.
- Before printing, it simplifies the output automaton either gently (removal of blank transitions) or drastically (cutpoints)

C2fsm - Options and Example

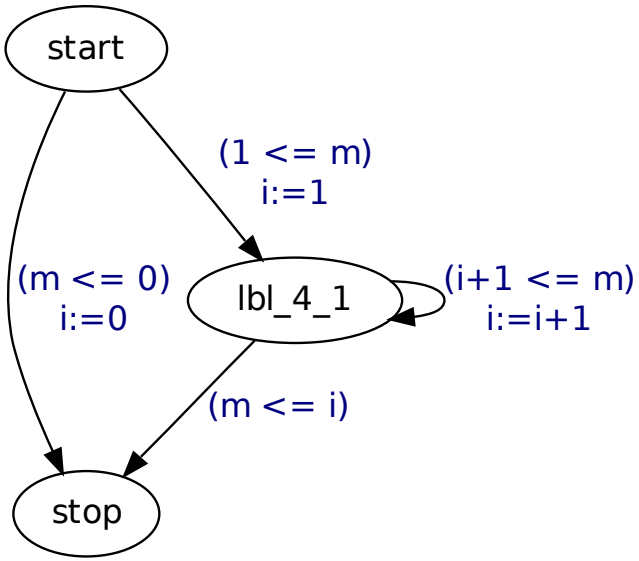
```
i=0;
while (i<m) {
  ++i;
}
```



-z (remove blank edges)



-s (packs basic blocks)



-cut (cutpoints)

Experiments

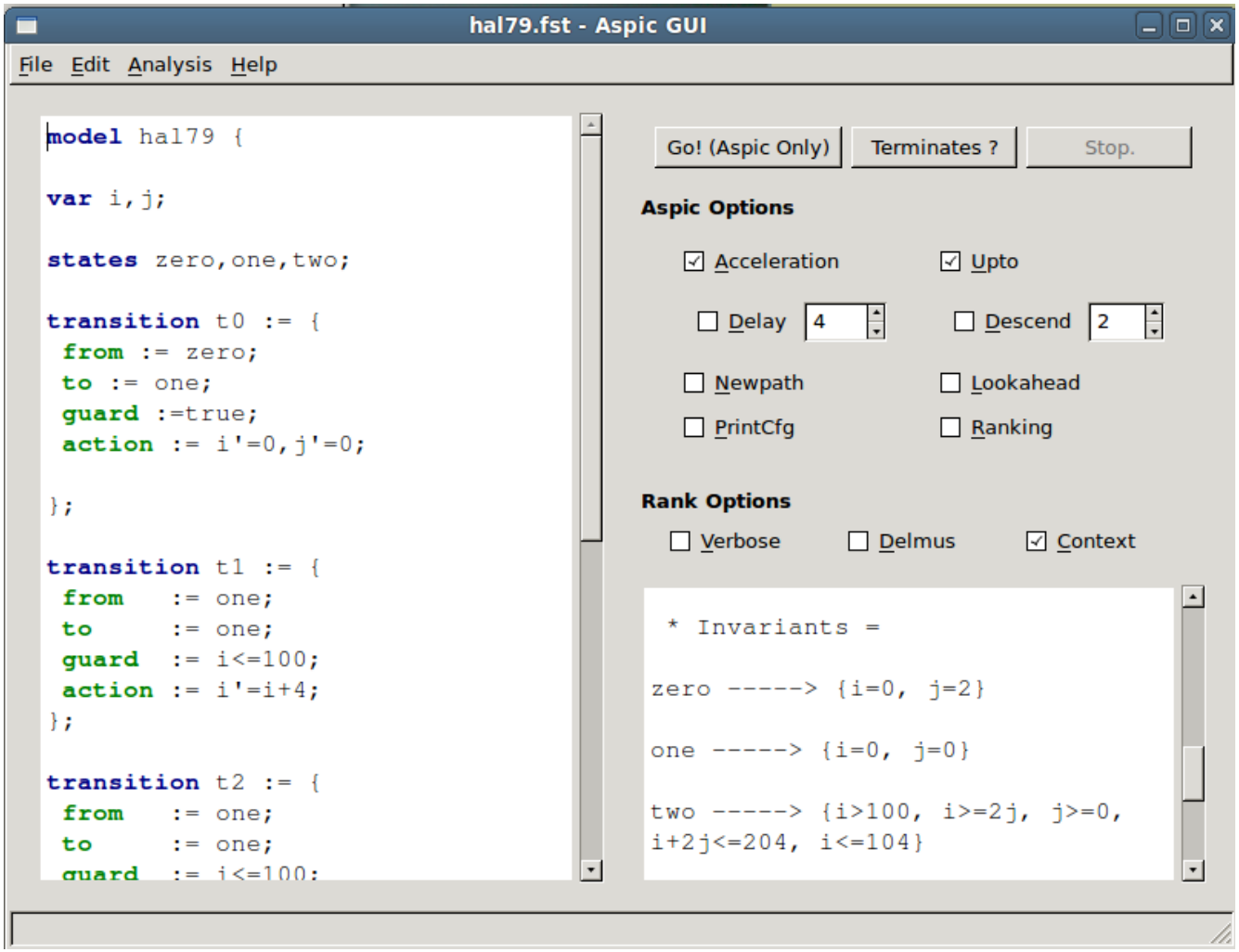
Cases of use :

- Proving **safety** :
  - of C code (c2fsm+aspic)
  - programs with lists via encoding (Bardin/Leroux+aspic) and (Iosif/Perarnau+ aspic)
  - of synchronous (Lustre) programs (oc2fst+aspic)
- Proving **termination** of C code with C2fsm and Rank : [Alias/Darte/Gonnord/Feautrier- SAS2010].

▶ Benchmarks/Comparison with other methods :

`http://laure.gonnord.org/pro/aspic`

Aspic and C2fsm Tour



Future Work

- Improvement of Aspic precision (eg, parameters)
- Aspic as API
- More precise affine abstractions in C2fsm
- Finalisation and test of oc2fst

Thanks

Questions ?