

# 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](mailto:aspic@gonnord.org)



# Goal

Computing invariants from C programs to

- prove correctness
- prove termination
- optimize compilers

# Aspic

## Invariant **C**omputation



# 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**.

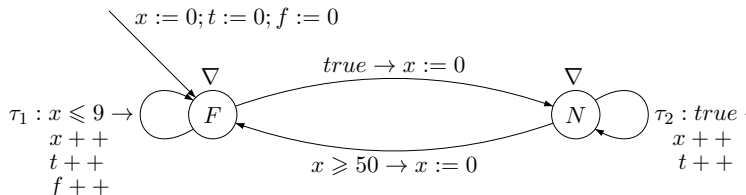
C2fsm is **a C parser** :

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

# Aspic - Theoretical foundations (1)

Aspic implements a variant of Linear Relation Analysis :

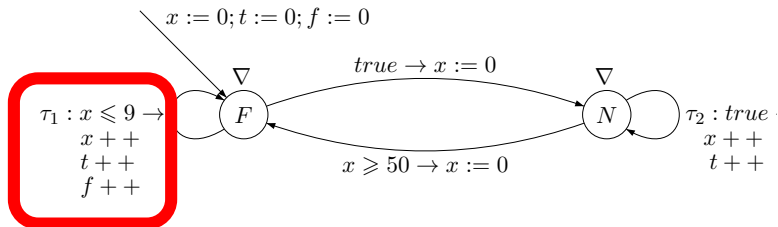
- Abstract interpretation with polyhedra on counter automata.
- Locally, the **exact** (abstract) reachability set is computed.



# Aspic - Theoretical foundations (1)

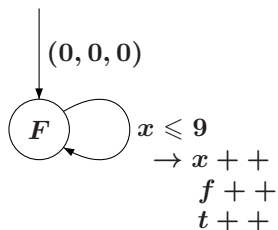
Aspic implements a variant of Linear Relation Analysis :

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



## Aspic - Theoretical foundations (2)

Local **acceleration** :



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

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

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

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

## Aspic - Theoretical foundations (3)

### References :

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

## Aspic - Theoretical foundations (3)

### References :

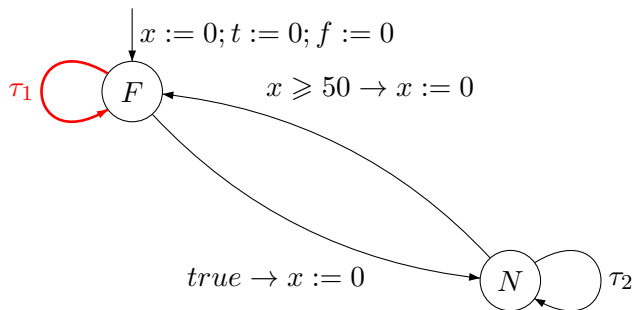
- The notion of **abstract acceleration** was first defined in [Gonnord/Halbwachs,SAS2006].
- Large classes of accelerable loops are described in [Gonnord,Phd].

### **Algorithms** and implementation :

- Uses `FixPoint` and `Newpolka`
  - 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].

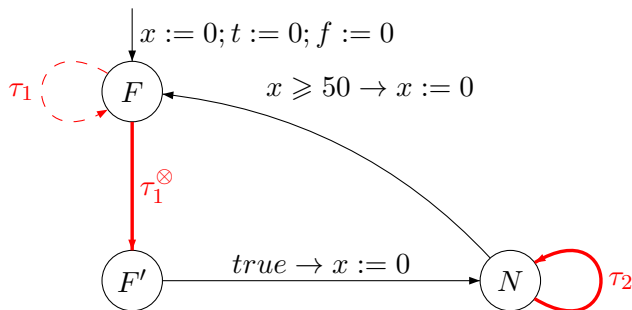
## Aspic - acceleration

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



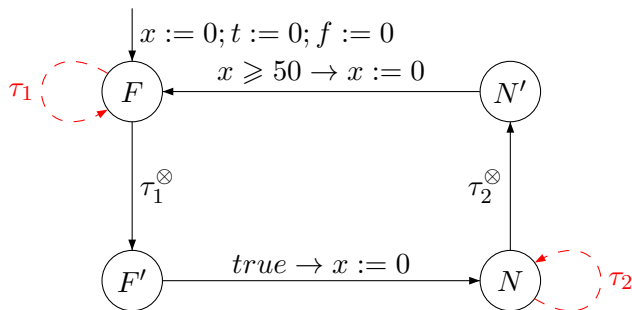
## Aspic - acceleration

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



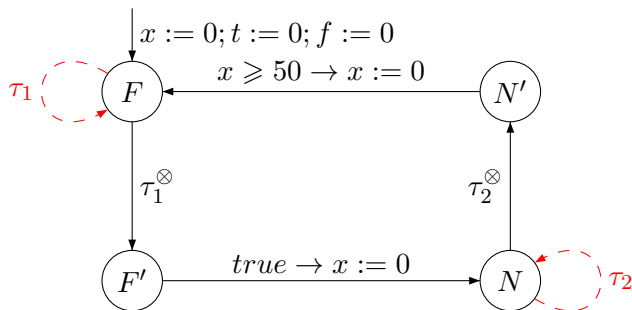
## Aspic - acceleration

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



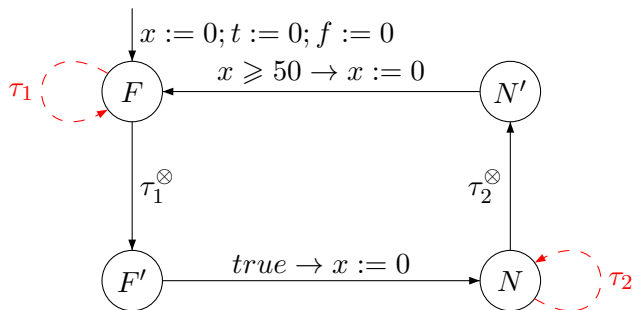
## Aspic - acceleration

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



## Aspic - acceleration

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



# Aspic - Limitations

Theoretical 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 :

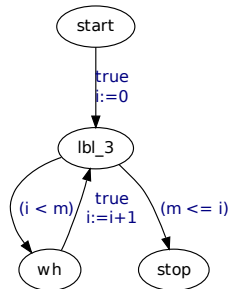
- It constructs an aspic input (automaton).
- When (safely) abstracting non affine conditions, it tries to keep as much useful information as possible.
- Before printing, it simplifies the output automaton either gently (removal of blank transitions) or drastically (cutpoints)

# C2fsm - Options and Example

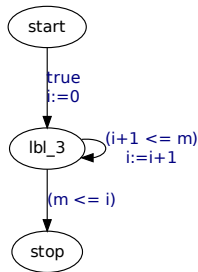
```

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

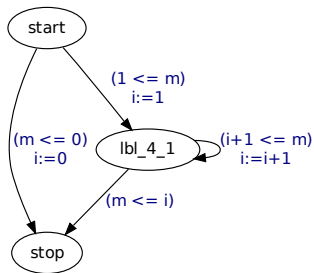
```



`-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

The screenshot shows the Aspic GUI window titled "hal79.fst - Aspic GUI". The window has a menu bar with "File", "Edit", "Analysis", and "Help".

The main text area contains the following model definition:

```

model hal79 {
  var i, j;

  states zero, one, two;

  transition t0 := {
    from := zero;
    to := one;
    guard := true;
    action := i'=0, j'=0;
  };

  transition t1 := {
    from := one;
    to := one;
    guard := i<=100;
    action := i'=i+4;
  };

  transition t2 := {
    from := one;
    to := one;
    guard := i<=100;
  };
}

```

At the top right, there are three buttons: "Go! (Aspic Only)", "Terminates ?", and "Stop".

Below the buttons are two sections of options:

**Aspic Options**

- Acceleration
- Upto
- Delay
- Descend
- Newpath
- Lookahead
- PrintCfg
- Ranking

**Rank Options**

- Verbose
- Delmus
- Context

At the bottom right, there is a text area showing the analysis results:

```

* Invariants =

zero ----> {i=0, j=2}

one ----> {i=0, j=0}

two ----> {i>100, i>=2j, j>=0,
i+2j<=204, i<=104}

```

# Future Work

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

# Thanks

Questions ?