Quantitative static analysis of Programs

applied to HLS ?

Laure Gonnord
http://laure.gonnord.org/pro/

Lille1 (USTL)/LIFL
Lille, France

Context

Verification / Static Analysis in circuits/HLS :

- Circuits are verified with Model Checking (**boolean**)
- As said **Florent**, we aims to develop higher-level tools, so let's focus on C **compilation**.
- As we have C, we can use our ≪ favorite ≫ techniques.

▶ Numerical Properties for the ≪ control ≫ part, for instance to bound the number of iterations (VHDL synthesis).

Context - 2

Infinite state programs (Static) **verification** :

- **safety** properties
- Infinite state : indecidability of most properties
  - interactive decision/proof
  - class restriction + abstractions.
  - if indecidable or too costly : approximations. Here : overapproximations ie ≪ conservative ≫ verification.

History of Abstract Interpretation

- French topic but now well used.
- People : Patrick and Radhia **Cousot** (1978), Nicolas Halbachs (1979), ... Papers in 1992, ... 2000-2010 David Monniaux, Antoine Miné, Laure Gonnord (Phd. 2007).
- Teams : ENS ULM, Verimag (Grenoble), Inria Popart team (Grenoble).
- Other People : Podelski (DE), Gulwani (US), Reps (US), ...
- Static Analysis in HLS : Compsys (ENS Lyon) and future inria team in Lille (embedded systems) ?.

Model - Notations

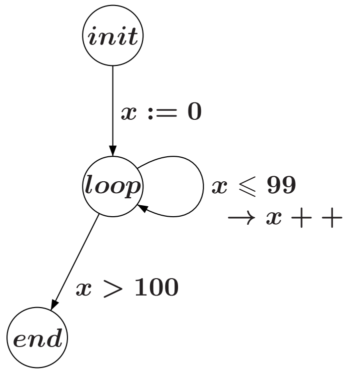
Numerical properties verification on control flow graphs with **affine** actions and tests :

$$k_1 \xrightarrow[\tau : g \to a]{AX \leqslant B \to X := CX + D} k_2$$

(counter automata, interpreted automata)

- A, C matrices, B, D vectors.
- « natural » semantic.
- Objective : generating invariants for **each control point**

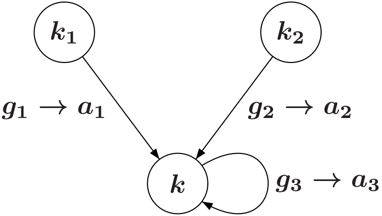
Model - Example



- $0 \leqslant x \leqslant 142$ is an **invariant** for "loop".

Problem Formalisation

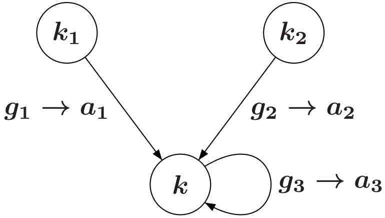
$\mathcal{R}_k$ = set of **valuations** at control point k :



$$\mathcal{R}_k = a_1(\mathcal{R}_{k_1} \cap g_1) \cup a_2(\mathcal{R}_{k_2} \cap g_2) \cup a_3(\mathcal{R}_k \cap g_3)$$

Problem Formalisation

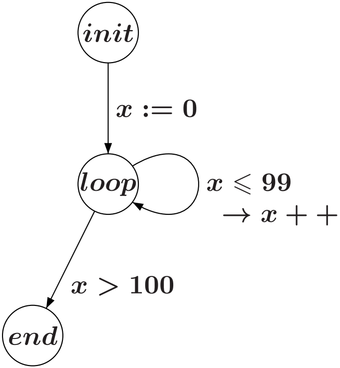
$\mathcal{R}_k$ = set of **valuations** at control point k :



$$\mathcal{R}_k = a_1(\mathcal{R}_{k_1} \cap g_1) \cup a_2(\mathcal{R}_{k_2} \cap g_2) \cup a_3(\mathcal{R}_k \cap g_3)$$

- (reccurent) equation system $\mathcal{R}_k = F(\mathcal{R}_k)$, **fixpoint**.

FixPoint - Exemple



$\mathcal{R}_{init} = \mathbb{R} = \top$, $R^1_{loop} = \{0\}$, puis $\{0, 1\}$, ... the **least** fixpoint is $\{0, 1 \dots 100\}$.

Fixpoint computation

- (Internal) représentation of valuations and computations

- Resolution convergence

▶ **Approximation**

Fixpoint computation

- (Internal) représentation of valuations and computations **convex polyhedra** :

$$P_k = \text{ if } k = k_{init} \text{ then } \top \text{ else } \bigsqcup_{(k,g,a,k')} a(P_{k'} \sqcap g)$$

- Resolution convergence

▶ **Approximation**

Fixpoint computation

- (Internal) représentation of valuations and computations **convex polyhedra** :

$$P_k = \text{ if } k = k_{init} \text{ then } \top \text{ else } \bigsqcup_{(k,g,a,k')} a(P_{k'} \sqcap g)$$

- Resolution convergence **widening operator**, with replacing

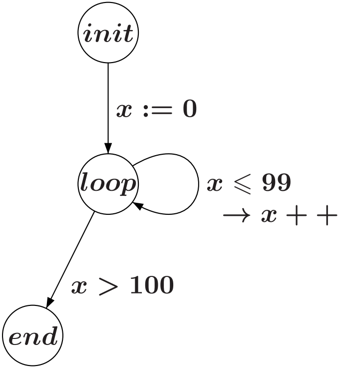
$$R_0, R_1 = F(R_0), R_2 = F(F(R_0)), \ldots \text{not convergent}$$

by

$$P_0, P_1 = P_0 \nabla F(P_0), P_1 \nabla F(P_1) \ldots \text{convergent}$$

▶ **Approximation**

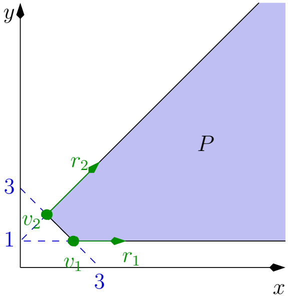
On the example



- $P_{loop}^{fin} = \{0 \leqslant x \leqslant 100\}$ if the analysis is precise enough

Resolution of the fixpoint system - 1

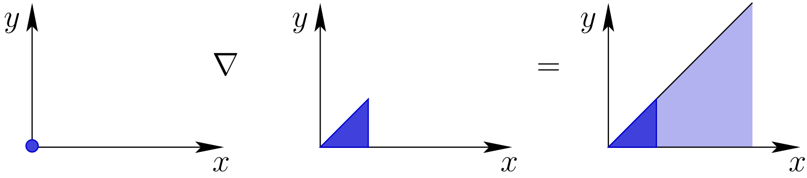
Convex polyhedra representation :



► Effective and efficient algorithmic (emptyness test, union, affine transformation . . .)

Fixpoint system resolution - 2

Widening : $P\nabla Q$: limit extrapolation.
$P\nabla Q$ constraints : take Q constraints and remove those which are not saturated by P.

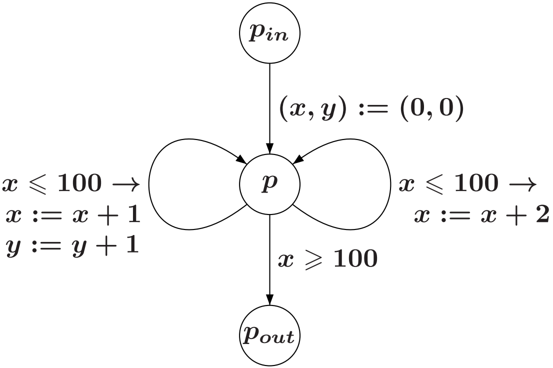


$x = y = 0$ $\qquad$ $0 \leqslant y \leqslant x \leqslant 1$ $\qquad$ $0 \leqslant y \leqslant x$

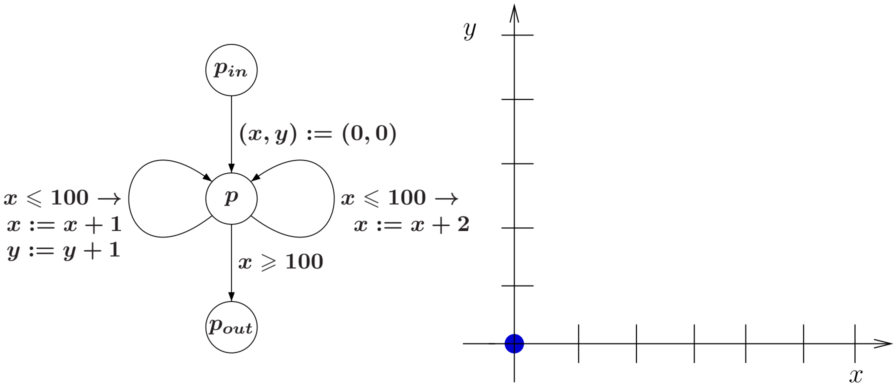
Trick (!) : $\{x = y = 0\} = \{0 \leqslant y \leqslant x \leqslant 0\}$

Analysis example - 1

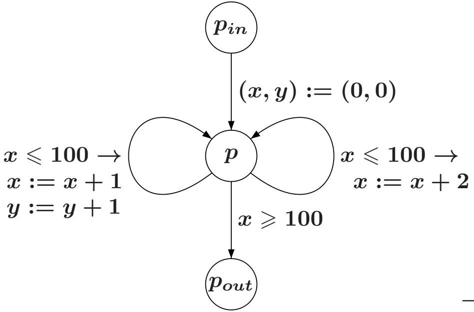
```
x:=0;y:=0
while (x<=100) do
  read(b);
  if b then
         x:=x+2
       else begin
         x:=x+1;
         y:=y+1;
       end;
  endif
endwhile
```



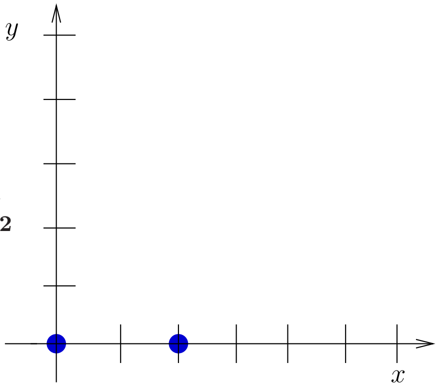
Example - 2


p_{in}
$(x, y) := (0, 0)$
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
p
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}
y
x

Example - 2

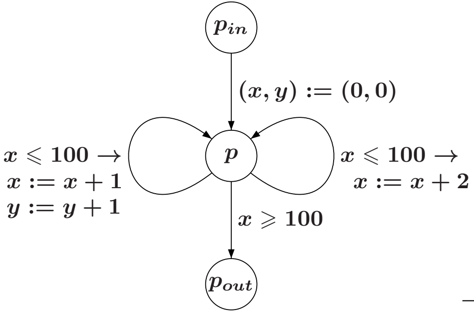


p_{in}
$(x, y) := (0, 0)$
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
p
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}

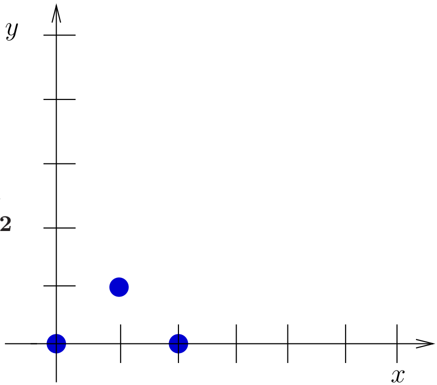


y
x

Example - 2

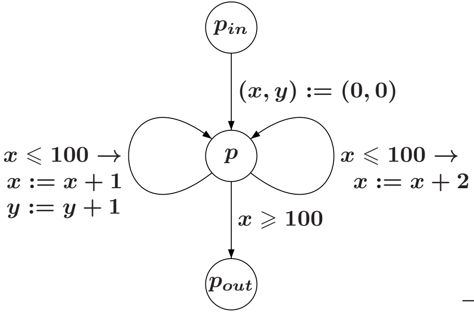


p_{in}
$(x, y) := (0, 0)$
p
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}

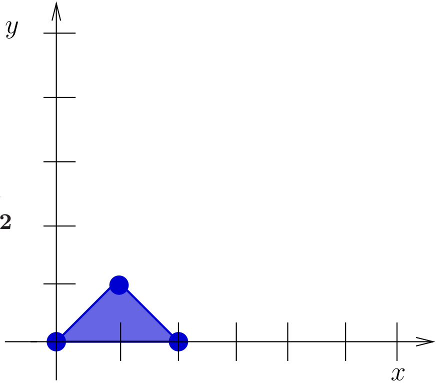


y
x

Example - 2

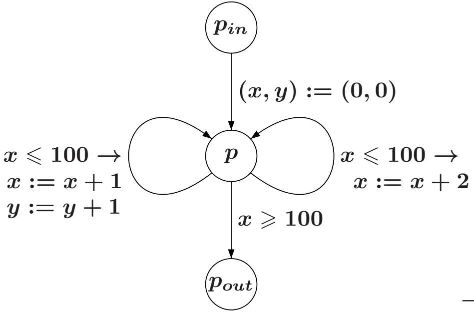


p_{in}
$(x, y) := (0, 0)$
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
p
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}

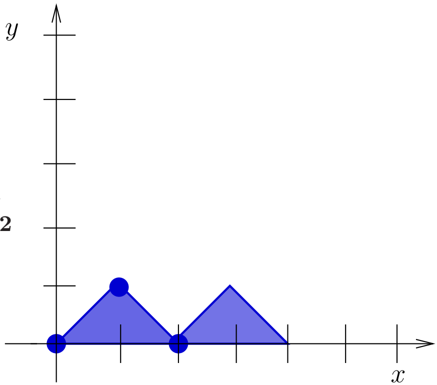


y
x

Example - 2

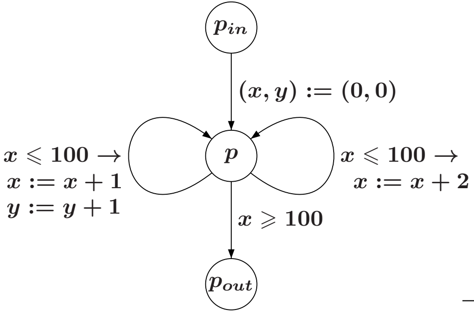


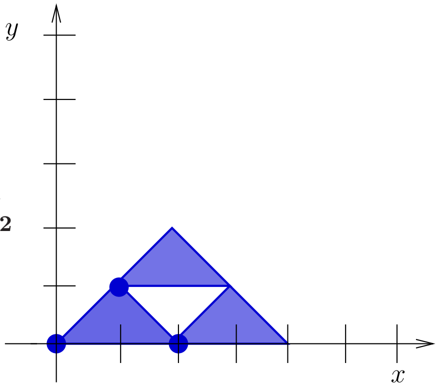
p_{in}
$(x, y) := (0, 0)$
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
p
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}



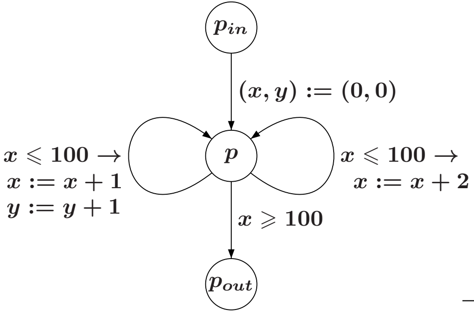
y
x

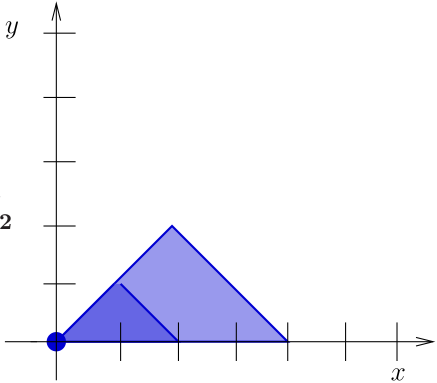
Example - 2


p_{in}
$(x, y) := (0, 0)$
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
p
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}

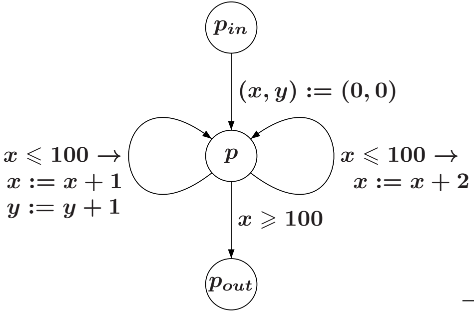

y
x

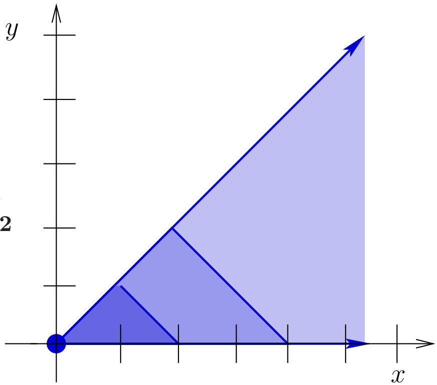
Example - 2


p_{in}
$(x, y) := (0, 0)$
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
p
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}

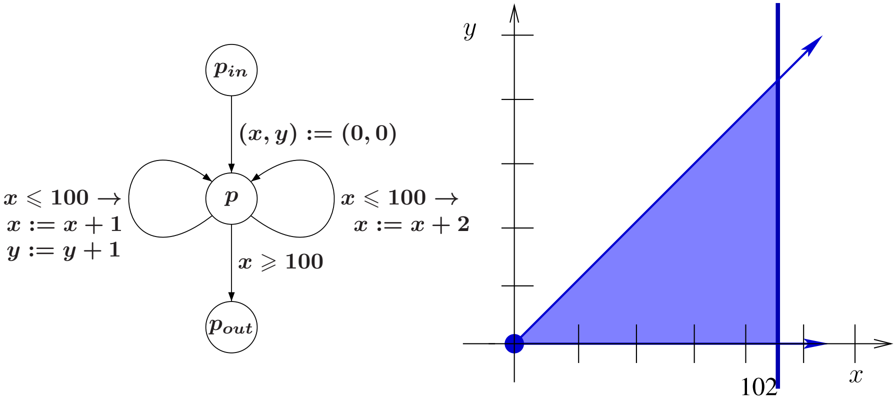

y
x

Example - 2


p_{in}
$(x, y) := (0, 0)$
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
p
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}


y
x

Example - 2


p_{in}
$(x, y) := (0, 0)$
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
p
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}
y
x
102

Linear Relation Analysis

Complexity increases with :

- number of control points
- number of numerical variables

Approximation is due to :

- Convex hulls
- **Widening** (my Phd.)

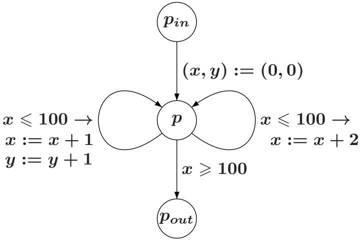
Aspic characteristics

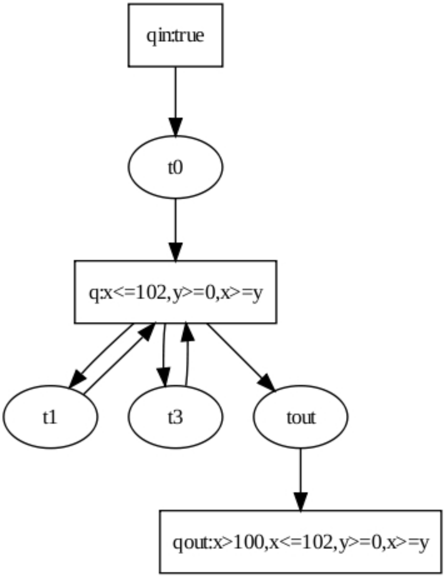
ASPIC : Accelerated Symbolic Polyhedral Invariant Computation

- Input : the automaton is described in textual language (Fast) with or without proof goal (formula). Non deterministic and random operations (`x:=?`)
- Classical computation + accelerations
- Ouput : invariants (+ diagnostic).

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

Invariants of the example

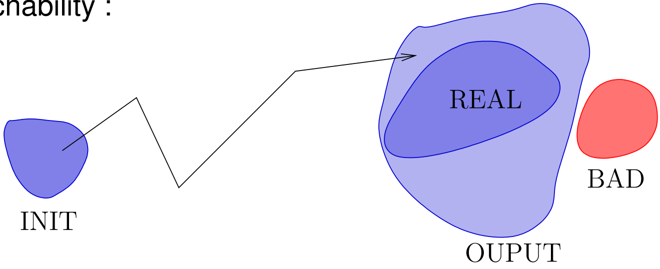




- Demo ?

Applications - 1

- Verification of **numerical** programs. « Proof » of non reachability :



- (non) reachability in counter automata coming from a SystemC Semantic (100 control points, J. Cornet (ST))

Applications - 2

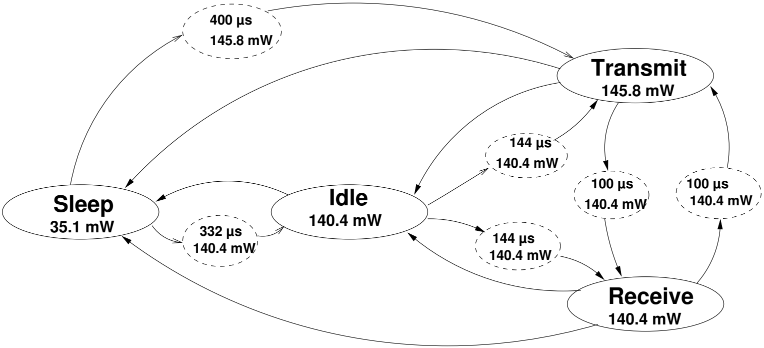
By encoding in counter automata :

- Programs with **lists** : R. Iosif and S. Perarnau (Verimag, Grenoble)
- Programs with **pointers** : A. Sangnier and A. Finkel (LSV, Cachan)

Applications - 3

By encoding

- Numerical invariants of energy automata (from **sensors**)
  L. Samper et F. Maraninchi (Verimag, Grenoble)



Applications - 4

Work with COMPSYS (ENS Lyon) : WTC (worst time complexity) estimation :

- compilation + static analysis
- scheduling

- With A. Darte, P. Feautrier, C. Alias.
- Demo ?

Summary

Linear relation analysis :

- computes numerical invariants
- on counter automata
- is not exact but sure (overapproximations)
- is performant
- is useful for other areas

Other analysis

Invariant generation

- Approximative : **intervals**, octogons, arrays, **floating point**. . .
- Exact : integer sets, intervals, booleans, constant propagation, . . .
- interprocedural, . . .

and combinations

Tools

An interesting **tool** :

`http://frama-c.cea.fr/`

and also **Apron Analyser** :

`pop-art.inrialpes.fr/interproc/interprocweb.cgi`

Who/Where

DART Team in Inria Lille :

- codesigning specialized embedded systems (software and hardware)
- Compiling/synthesis issues, network issues, hardware issues, high level programming, and also **parallel** processing.
- Upgoing work on FPGA's : expressing the architecture, architecture exploration (help for decision)
- Currently building a **group** to bring verification into the developpement process : rewriting, static analysis, model verification, ...

▶ postdocs positions. . .