Computing invariants of programs

Calcul Formel team seminar

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

Lille1 (USTL)/LIFL
Lille, France

Juin 2012

Context

Working on C programs (or **flowcharts programs**), we aim to :

- Optimize the compilation process
- Prove safety properties
- Prove termination

► Static Analyses, of different kinds. Here we focus on abstract interpretation

Goal of Abstract Interpretation

Propagating **information** about program variables (numerical,arrays, ...) in order to get **invariants**.

- We focus on **numerical variables** here.

Model - Notations

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

$$\begin{tikzcd} \boldsymbol{k_1} \arrow[rrr, "AX \leqslant B \to X := CX + D", "\tau : g \to a"'] & & & \boldsymbol{k_2} \end{tikzcd}$$

(counter automata, interpreted automata)

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

Vocabulary

- State : couple (pc, val)
- Initial states
- Reachable states.

▶ Reachability is indecidable.

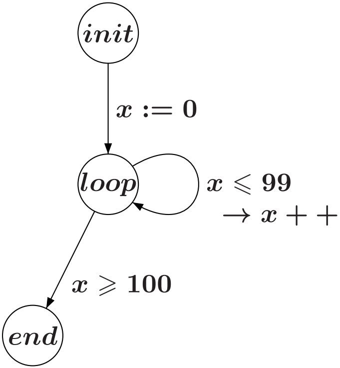
From C programs to counter automata

Our tool C2FSM (Feautrier) :

- Turns a sequential C file into a counter automaton.
- Performs **safe** abstractions of non numerical variables, structures, behaviors.

► other tools based on LLVM, Rose, ... still a research problem

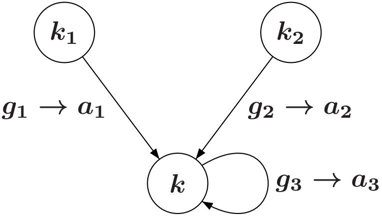
Model - Example



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

Problem Formalisation

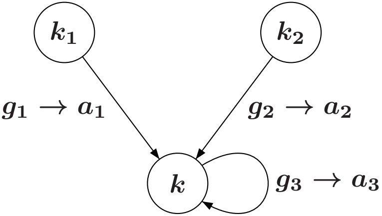
$\mathcal{R}_k$ = set of **valuations** at control point k :



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

Problem Formalisation

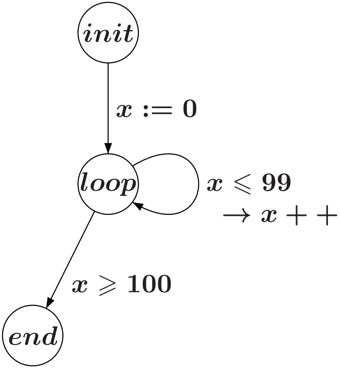
$\mathcal{R}_k$ = set of **valuations** at control point k :



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

- (reccurent) equation system $\mathcal{R}_k = F(\mathcal{R}_k)$, **fixpoint**, with $\mathcal{R}_k^0$ = initial val of variables.

FixPoint - Exemple



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

Fixpoint computation - Problems

- Representation (infinite sets, integers, . . .)
- Computation of the transition relation
- Convergence

► **Linear relation Analysis**, ie abstract interpretation with **polyhedra.**

Fixpoint computation with polyhedra

- (Internal) representation of valuations and computations

- Transition function :
- Resolution convergence

Fixpoint computation with polyhedra

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

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

- Transition function :
- Resolution convergence

Fixpoint computation with polyhedra

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

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

- Transition function : see later
- Resolution convergence **widening operator**, with replacing

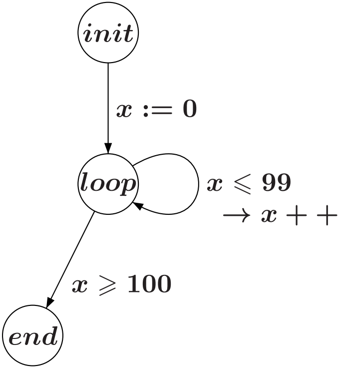
$$R_0, R_1 = F(R_0), R_2 = F(F(R_0)), \ldots \text{not convergent}$$

by

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

▶ **Approximation**

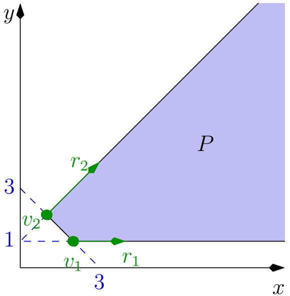
On the example



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

Resolution of the fixpoint system - 1

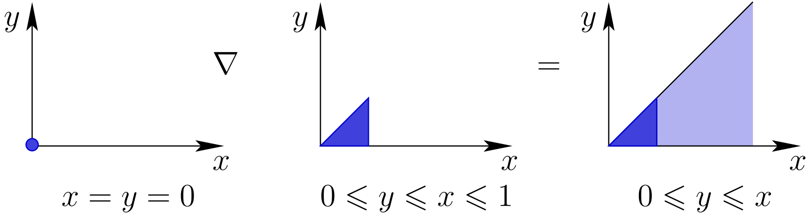
(pb 1 and 2) Convex polyhedra representation + transition function :



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

Fixpoint system resolution - 2

(pb 3) Coping with termination.
Widening : $P\nabla Q$: limit extrapolation.
$P\nabla Q$ constraints : take Q constraints and remove those which are not saturated by P.



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

Fixpoint system resolution - 3

The widening operator being designed, we compute ($x \subseteq F(x)$)

$$P_0, P_1 = P_0 \nabla F(P_0), P_2 = P_1 \nabla F(P_1) \ldots$$

finite computation instead of :

$$P_0, F(P_0), F^2(P_0), \ldots$$

which can be infinite.

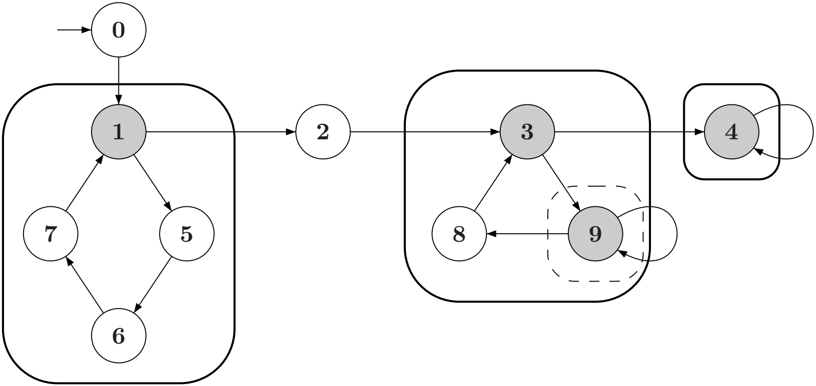
Theorem

(Cousot/Cousot 77) Iteratively computing the reachable states from the entry point with the domain operators and applying widening at entry nodes of loops converges in **a finite** *number of steps to a overapproximation of the least invariant (*__postfixpoint__*).*

- The widening operators must satisfy the non ascending chain condition (see Cousot/Cousot 1977).

Fixpoint system resolution - 4

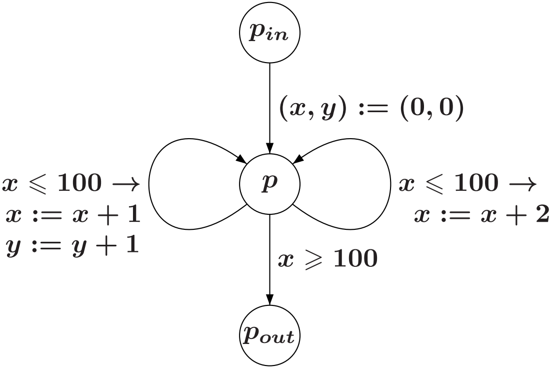
Iteration strategy :
(Bourdoncle,1992) Computing strongly connected subcomponents and iterate inside each :



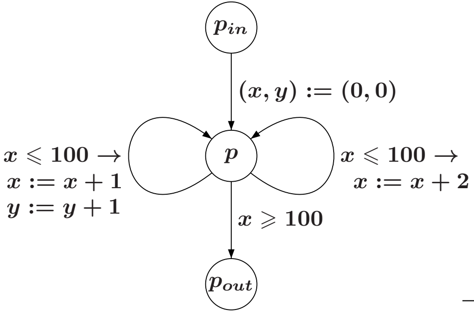
Gray nodes are **widening nodes**

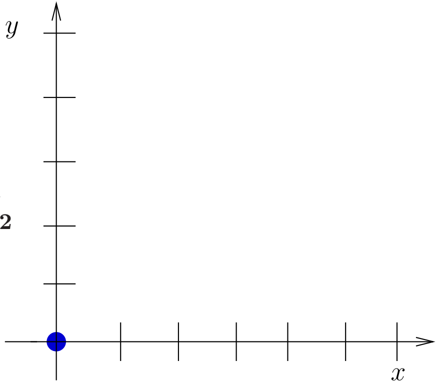
Analysis example - 1

```
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
```

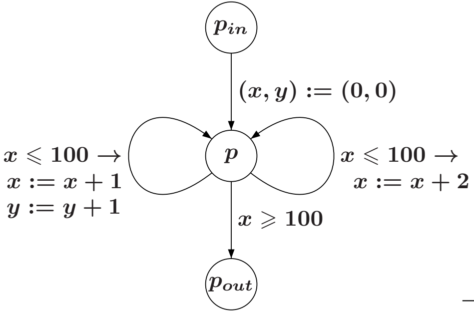


Example - 2

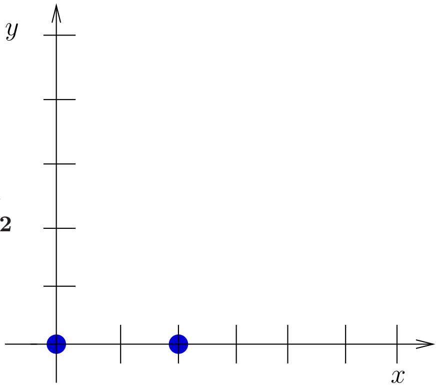

p_{in}
$(x, y) := (0, 0)$
p
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}


y
x

Example - 2

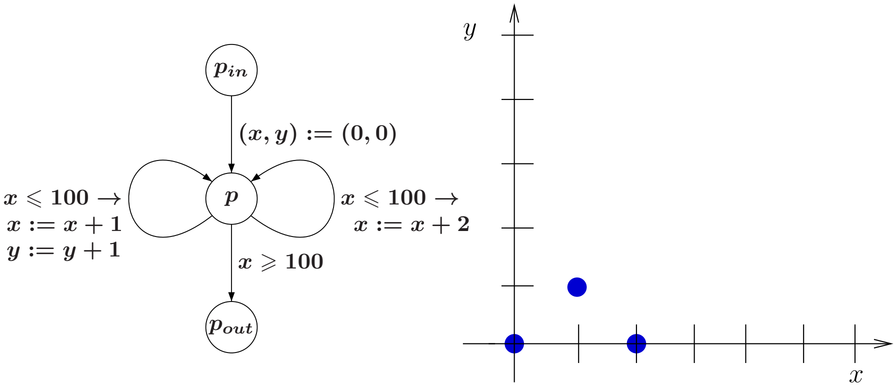


p_{in}
$(x, y) := (0, 0)$
p
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}



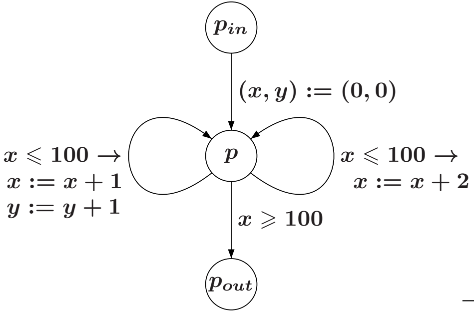
y
x

Example - 2

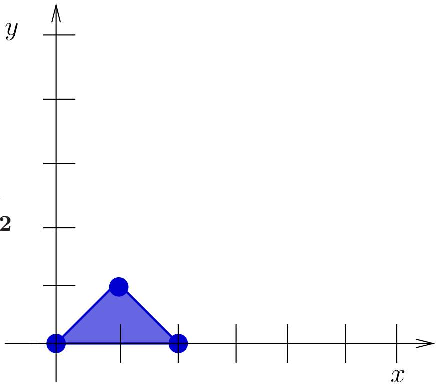


p_{in}
$(x, y) := (0, 0)$
p
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}
y
x

Example - 2

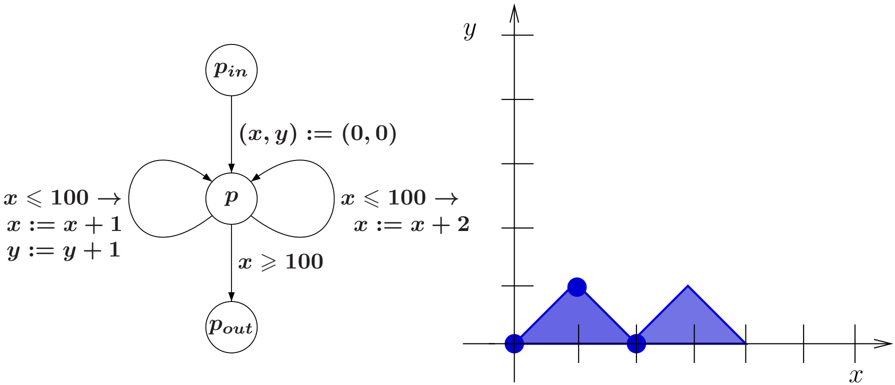


p_{in}
$(x, y) := (0, 0)$
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
p
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}

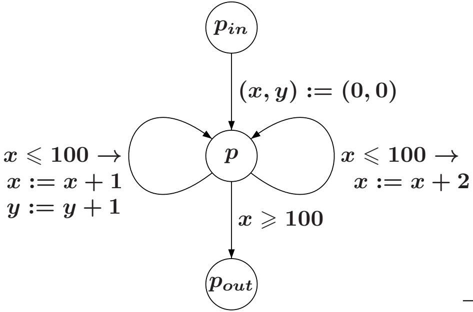


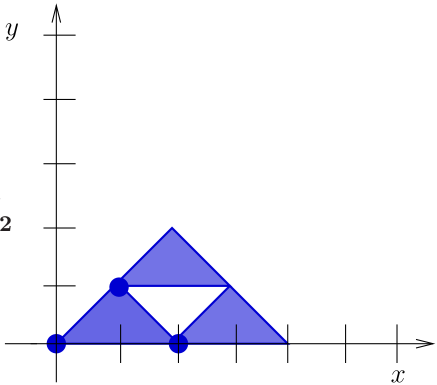
y
x

Example - 2

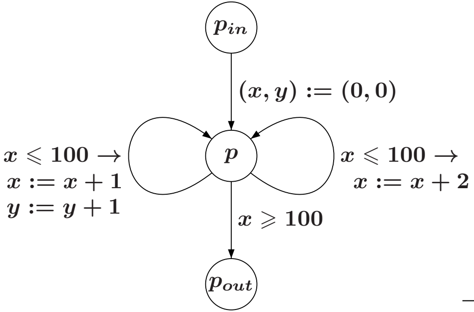

p_{in}
$(x, y) := (0, 0)$
$x \leqslant 100 \rightarrow$
$x := x + 1$
$y := y + 1$
p
$x \leqslant 100 \rightarrow$
$x := x + 2$
$x \geqslant 100$
p_{out}
y
x

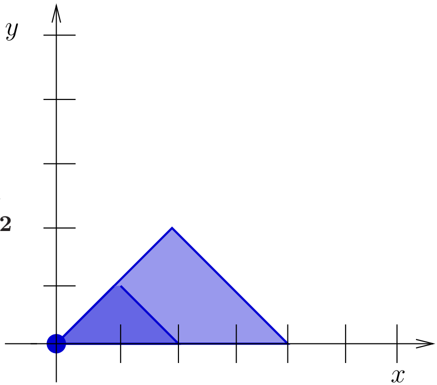
Example - 2


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

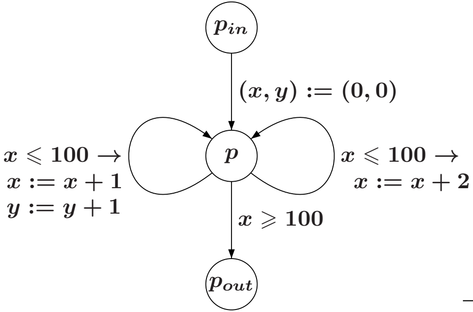

y
x

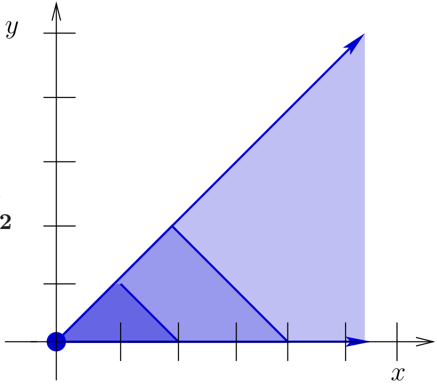
Example - 2


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

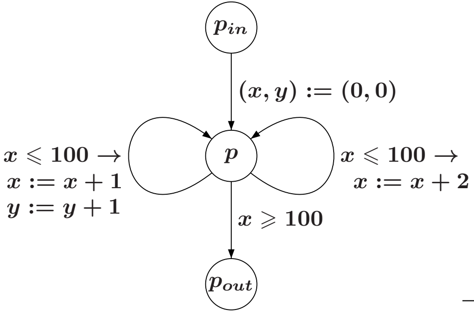

y
x

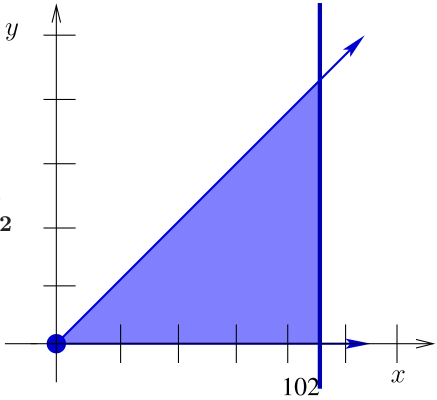
Example - 2


p_{in}
$(x, y) := (0, 0)$
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
102
x

Demo : 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).

Logo by Isabelle Borne

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

Linear Relation Analysis

Complexity increases with :

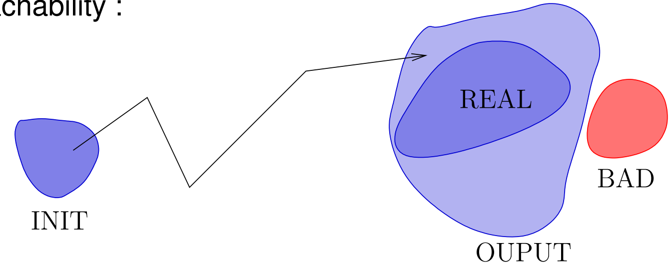
- number of control points
- number of numerical variables

Approximation is due to :

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

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

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

- compilation + static analysis
- scheduling

▶ With A. Darte, P. Feautrier, C. Alias.

Conclusion

Linear relation analysis :

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