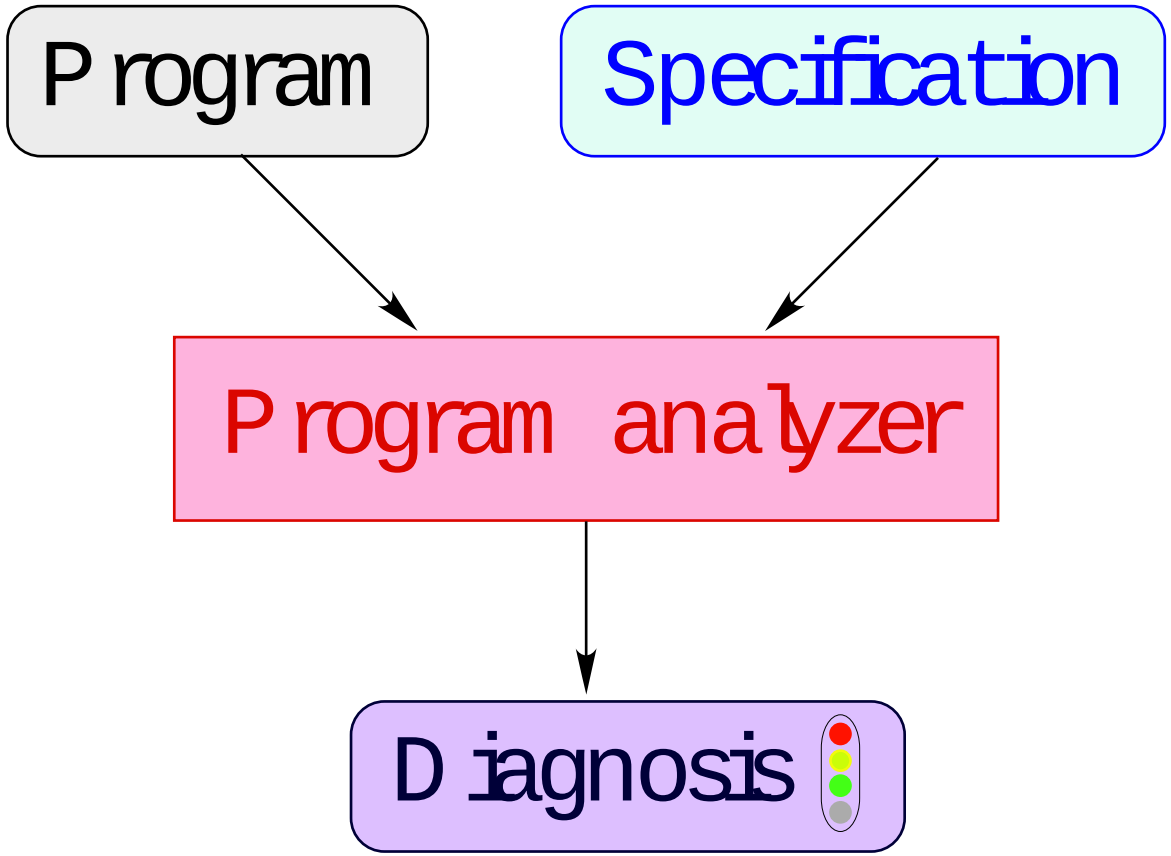
Overview of the scientific achievements of DAEDALUS

Patrick COUSOT
ENS, 45 rue d'Ulm
75230 Paris cedex 05, France
Patrick.Cousot@ens.fr
www.di.ens.fr/ cousot

Seminar for industrial users, Saarbrücken, 27 Sep. 2002

Formal methods

Static program verification



Indecidability of static program verification

Given an input program, a specification and finite computing resources, any static program verifier will either:

- answer positively, or
- answer negatively, or
- answer "I don't know" [1], or
- never terminate, or
- run out of memory, or
- ask for interactive user help.

[1] which may be a true or a false alarm.

Coping with finite computing resources

- Static program verifiers must avoid very long computations/non-termination and limit memory consumption:
  - Deductive methods: ask the user to help the theorem prover (e.g. by providing inductive invariants or assitance in proving theorems);
  - Model checking: ask the user to provide a (small) finitary model of the semantics of the program;
  - Abstract interpretation: the analyzer uses an approximate semantics of the program.

User interaction versus abstraction (1)

- User interaction (deductive methods, model checking) will always succeed positively or negatively for a given program by providing an appropriate proof/model, but:
  - The human cost for designing the proof/model is prohibitive and much larger than the effort for designing the program,
  - The proof/model is usually not reusable for different programs,
  - Additional maintenance cost (for maintaining both the program and proof/model);

User interaction versus abstraction (2)

- Abstraction (abstract interpretation), may partially fail for a program by not providing definite answers to all questions, but:
  - Entirely automatic without any user interaction [2],
  - Reusable for all programs of a given programming language,
  - No additional maintenance cost (since the static program analysis is automatic);

[2] But maybe to provide the required specification.

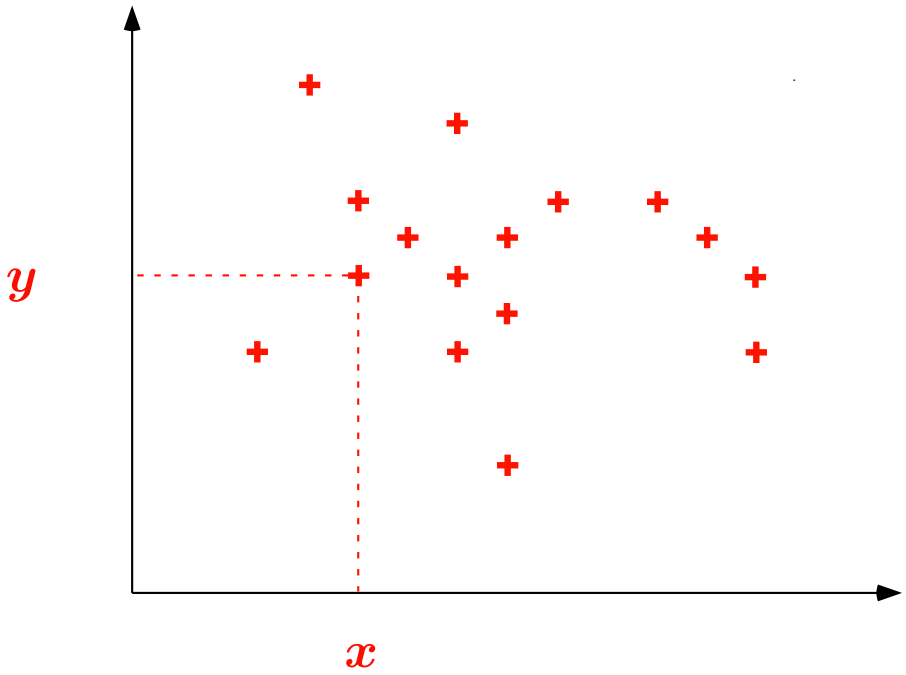
An Informal Introduction to Abstract Interpretation

Abstract Interpretation

- **Thinking tool**: the idea of abstraction is central to reasoning (in particular on computer systems);
- A framework for designing **mechanical tools**: the idea of effective approximation leads to automatic semantics-based formal systems/program manipulation tools.

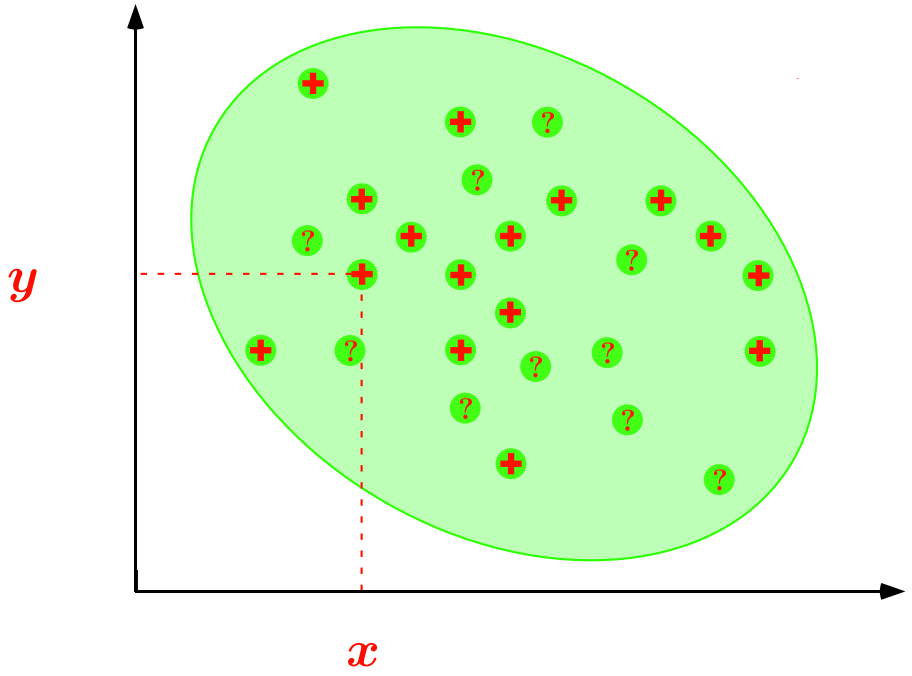
. . . An example . . .

An [in]finite set of points:



$$\{\ldots, \langle 19,\ 77\rangle, \ldots, \langle 20,\ 02\rangle, \ldots\}$$

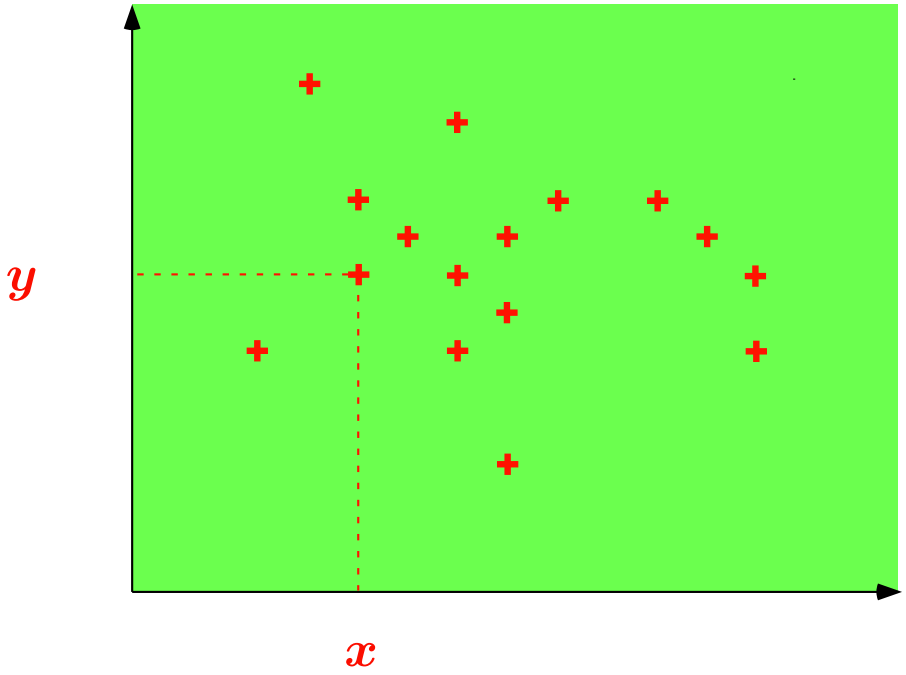
Abstraction from above:



$$\{\ldots, \langle 19,\ 77\rangle, \ldots,$$

$$\langle 20,\ 02\rangle, \langle ?,\ ?\rangle, \ldots\}$$

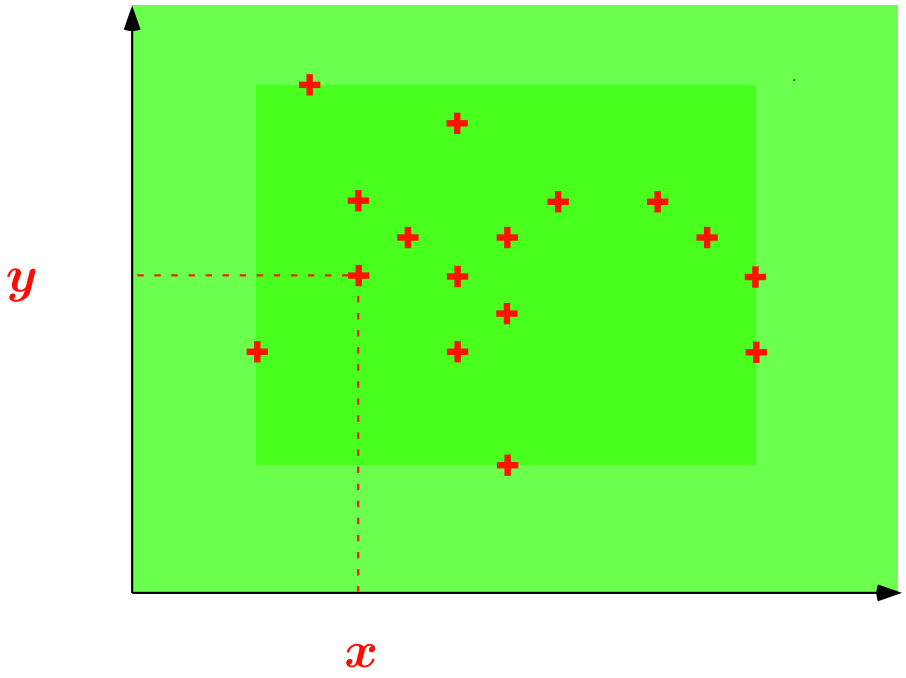
Effective abstraction from above: Signs [3]



$$\begin{cases} x \geq 0 \\ y \geq 0 \end{cases}$$

[3] P. Cousot & R. Cousot. *Systematic design of program analysis frameworks*. ACM POPL'79, pp. 269–282, 1979.

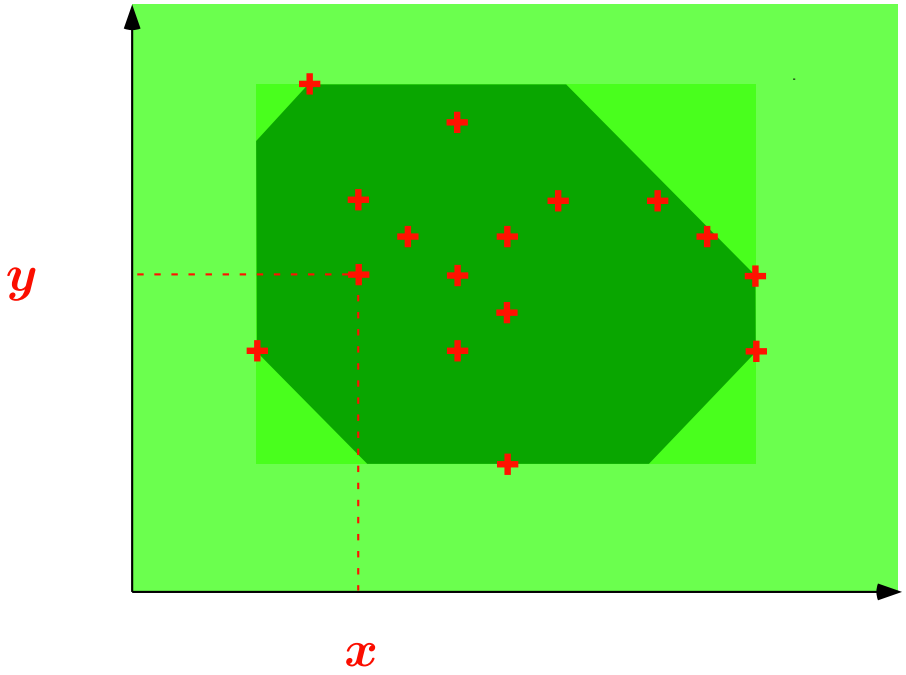
Effective abstraction from above: Intervals [4]



$$\begin{cases} x \in [19,\ 77] \\ y \in [20,\ 02] \end{cases}$$

[4] P. Cousot & R. Cousot. *Static determination of dynamic properties of programs*. Proc. 2nd Int. Symp. on Programming, Dunod, 1976.

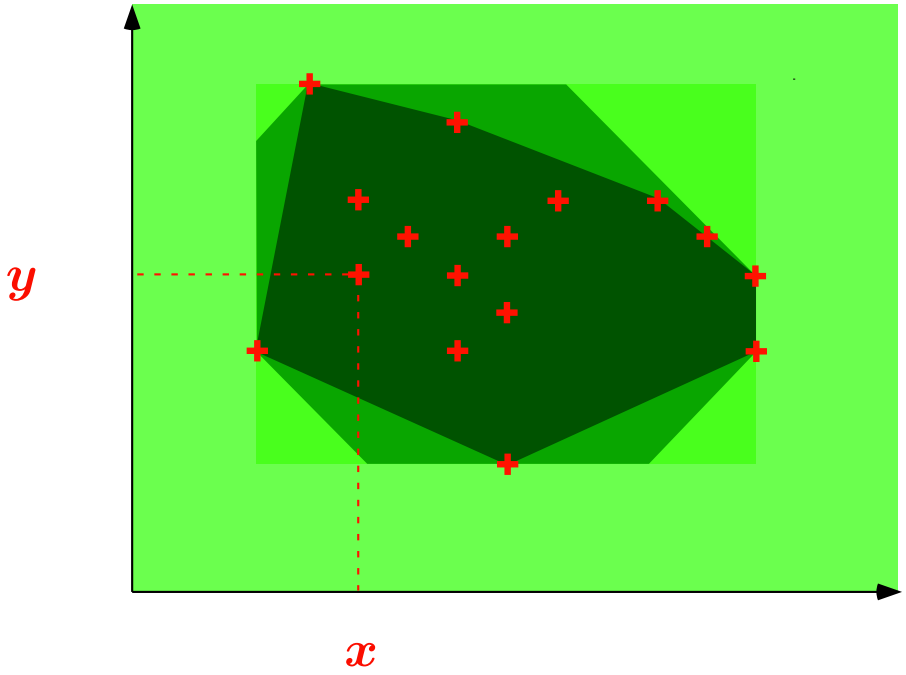
Effective abstraction from above: Octagons [5]



$$\begin{cases} 1 \leq x \leq 9 \\ x + y \leq 77 \\ 1 \leq y \leq 9 \\ x - y \leq 99 \end{cases}$$

[5] A. Miné. *A New Numerical Abstract Domain Based on Difference-Bound Matrices*. PADO'2001. LNCS 2053, pp. 155–172. Springer 2001.

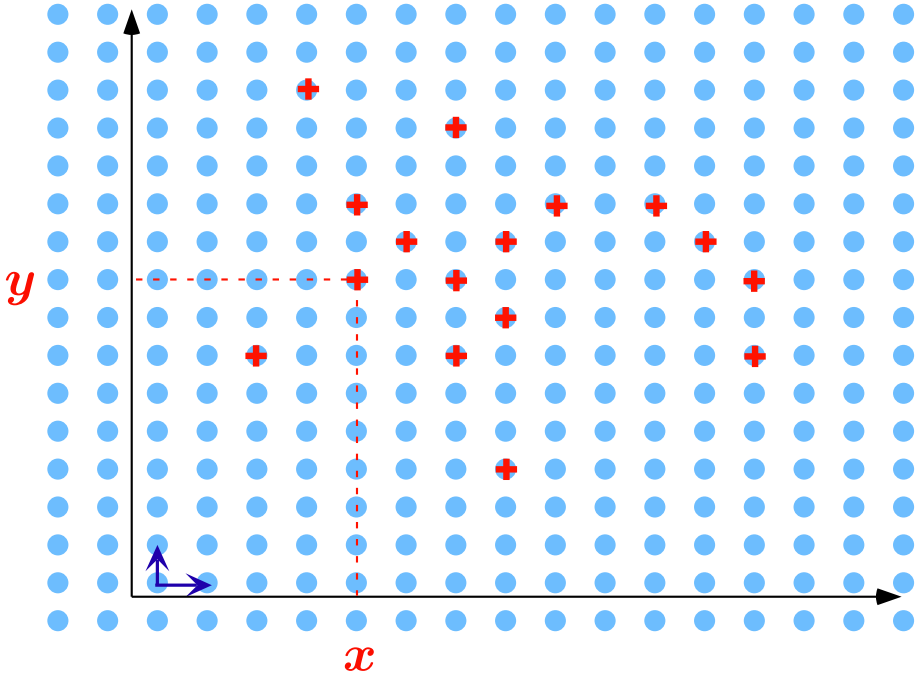
Effective abstraction from above: Polyhedra [6]



$$\begin{cases} 19x + 77y \leq 2002 \\ 20x + 02y \geq 0 \end{cases}$$

[6] P. Cousot & N. Halbwachs. *Automatic discovery of linear restraints among variables of a program*. ACM POPL, 1978, pp. 84–97.

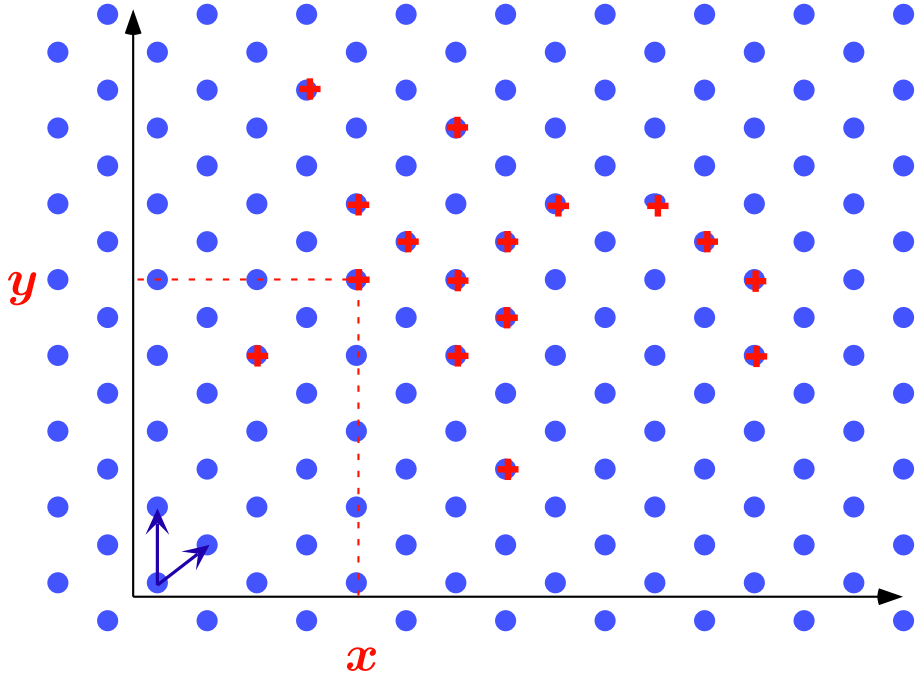
Effective abstraction from above: Simple congruences [7]



$$\begin{cases} x = 19 \bmod 77 \\ y = 20 \bmod 99 \end{cases}$$

[7] Ph. Granger. *Static Analysis of Arithmetical Congruences*. Int. J. Comput. Math. 30, 1989, pp. 165–190.

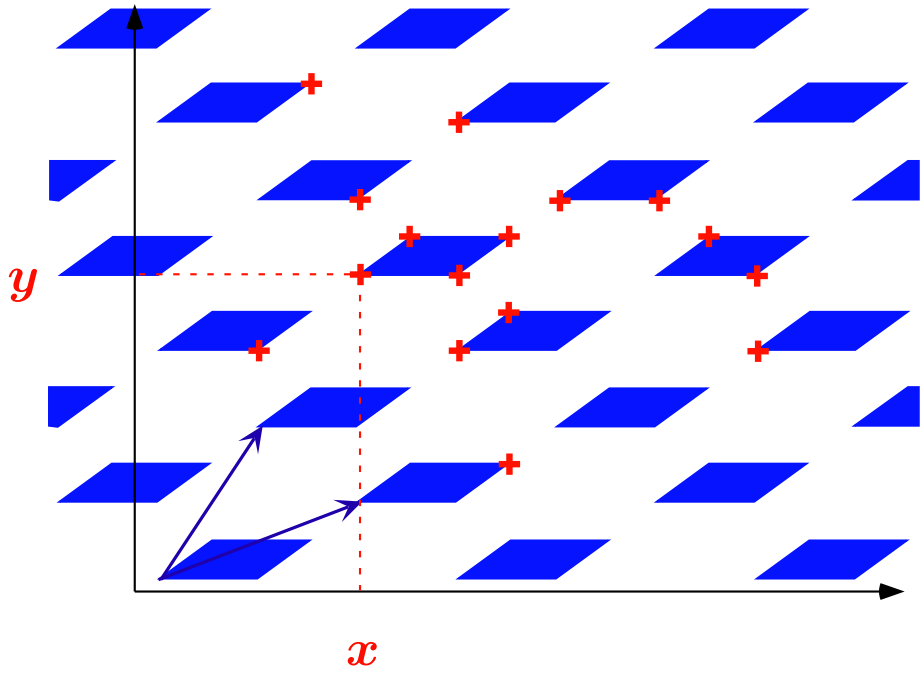
Effective abstraction from above: Linear congruences [8]



$$\begin{cases} 1x + 9y = 7 \bmod 8 \\ 2x - 1y = 9 \bmod 9 \end{cases}$$

[8] Ph. Granger. *Static Analysis of Linear Congruence Equalities among Variables of a Program.* TAPSOFT '91, pp. 169–192. LNCS 493, Springer, 1991.

Effective abstraction from above: Trapezoidal congruences [9]

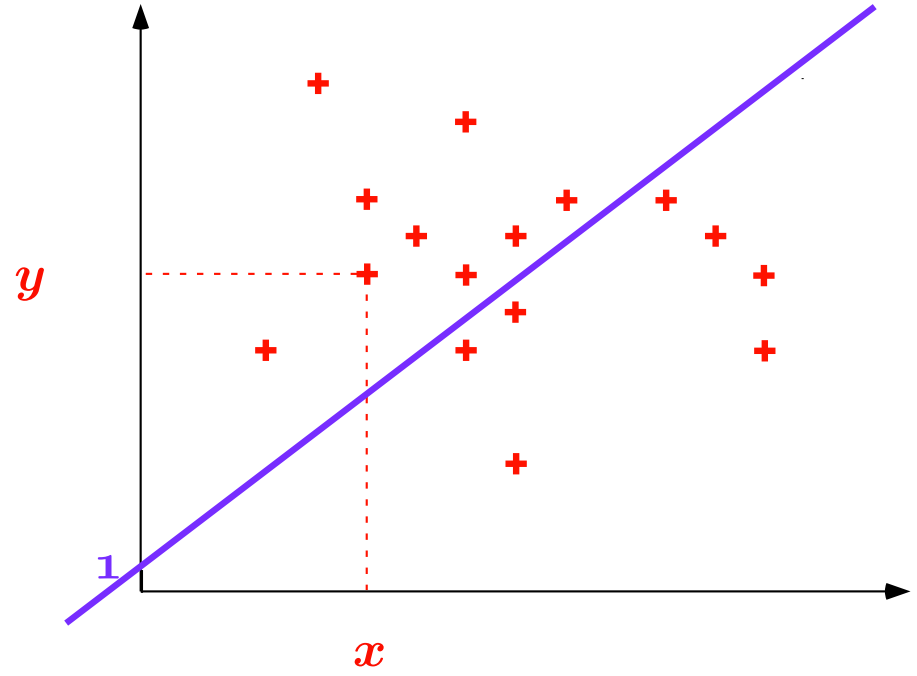


$$\begin{cases} 1x + 9y \in [0, 77] \bmod 10 \\ 2x - 1y \in [0, 99] \bmod 11 \end{cases}$$

[9] F. Masdupuy. *Array Operations Abstraction Using Semantic Analysis of Trapezoid Congruences*. ACM ICS '92.

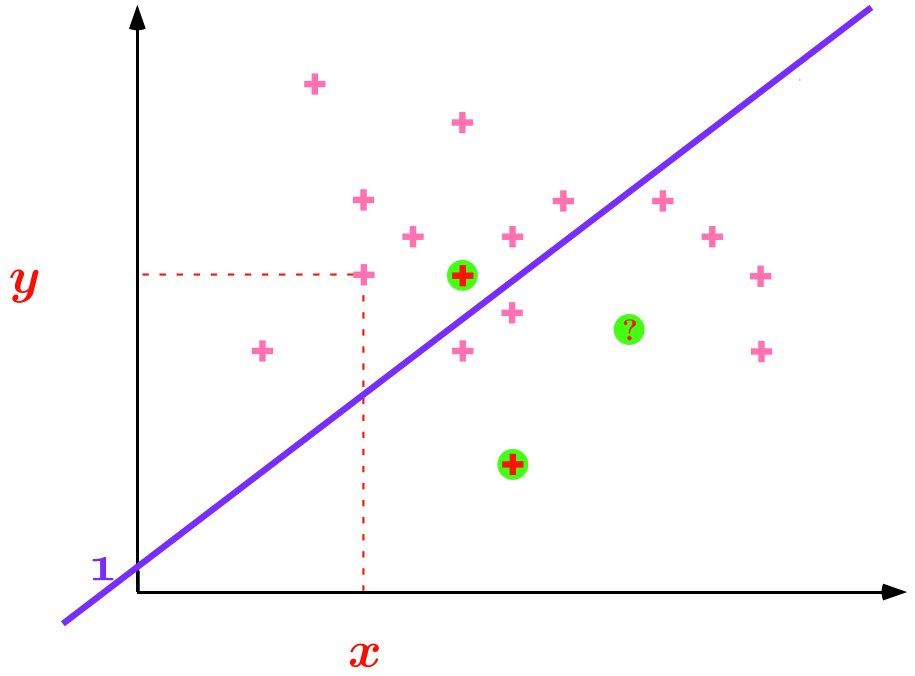
Conservative Approximation

- Is the operation `1/(x+1-y)` well defined at run-time?
- Concrete semantics: **yes**



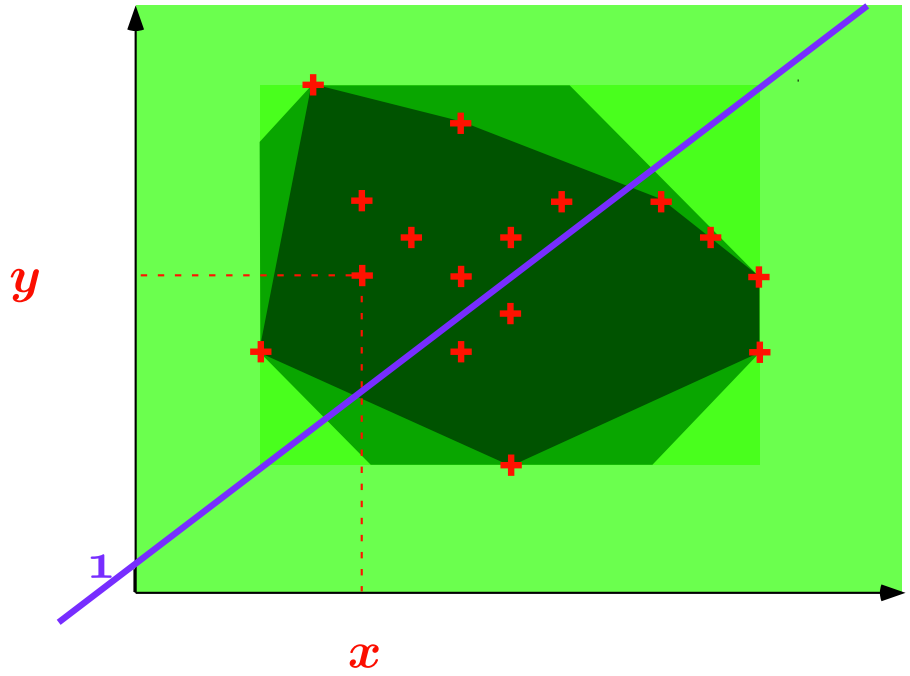
Conservative Approximation

- Is the operation `1/(x+1-y)` well defined at run-time?
- Testing : **You never know!**



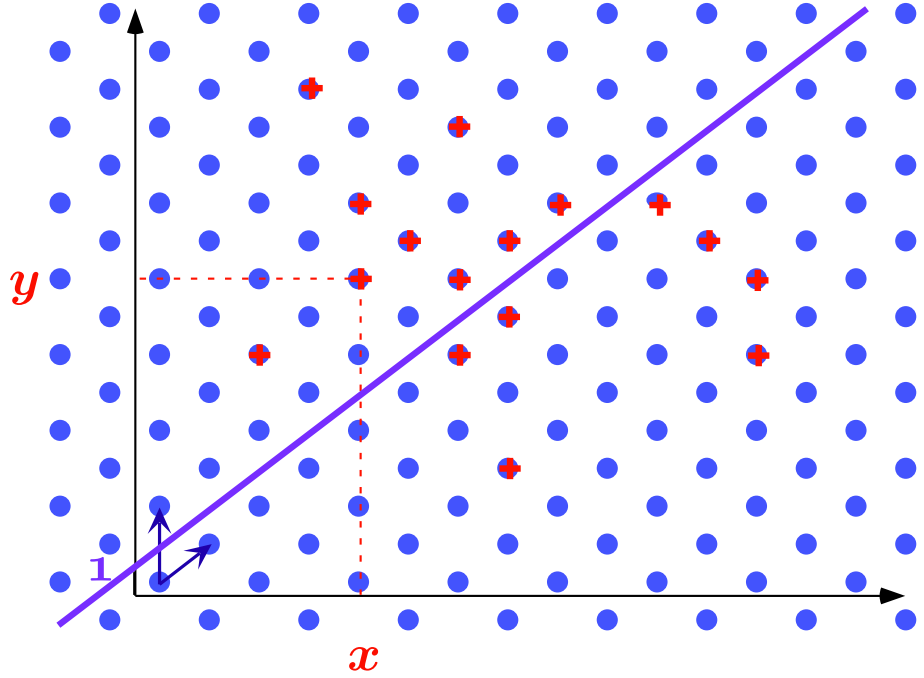
Conservative Approximation

- Is the operation `1/(x+1-y)` well defined at run-time?
- Abstract semantics 1: **I don't know**



Conservative Approximation

- Is the operation `1/(x+1-y)` well defined at run-time?
- Abstract semantics 2: **yes**



Achievements of DAEDALUS

Academic visibility

- The academic visibility of the DAEDALUS project was world-wide;
- E.g.: International symposium on static analysis SAS'2002, 86 papers submitted worldwide, 32 accepted, 12 papers (co-)authored by members of DAEDALUS teams.

Industrial productivity

- An uncommon number of high-quality software tools have been improved or newly designed;
- An unforeseen number of these tools will have direct industrial applications in the short or medium term;
- Thanks to the exceptional involvement of the end-user and its high-quality assessment.

A synthetic summary of DAEDALUS achievements

- Tools (to be presented in this seminar);
- Prototypes (prefigurating future tools);
- Basic research (theoretical results paving the way for future practice).

See the Synthetic summary of DAEDALUS achievements on the DAEDALUS web site.

European lead in Abstract Interpretation

- An expression of interest on a network of excellence on abstract interpretation (AINoE) has been submitted to FP6.

THE END, THANK YOU

Conclusions

Short-term applicability

- Some applications of abstract interpretation to software analysis are mature for short-term direct industrial application;