Improving Systems Quality
— Challenges and Trends —
An Abstract Interpretation Perspective

Patrick COUSOT

École Normale Supérieure
45 rue d'Ulm, 75230 Paris cedex 05, France

Patrick.Cousot@ens.fr

www.di.ens.fr/~cousot

Remise de la médaille d'argent du CNRS à Joseph SIFAKIS

Grenoble, France Jeudi 11 avril 2002

Motivations [1]

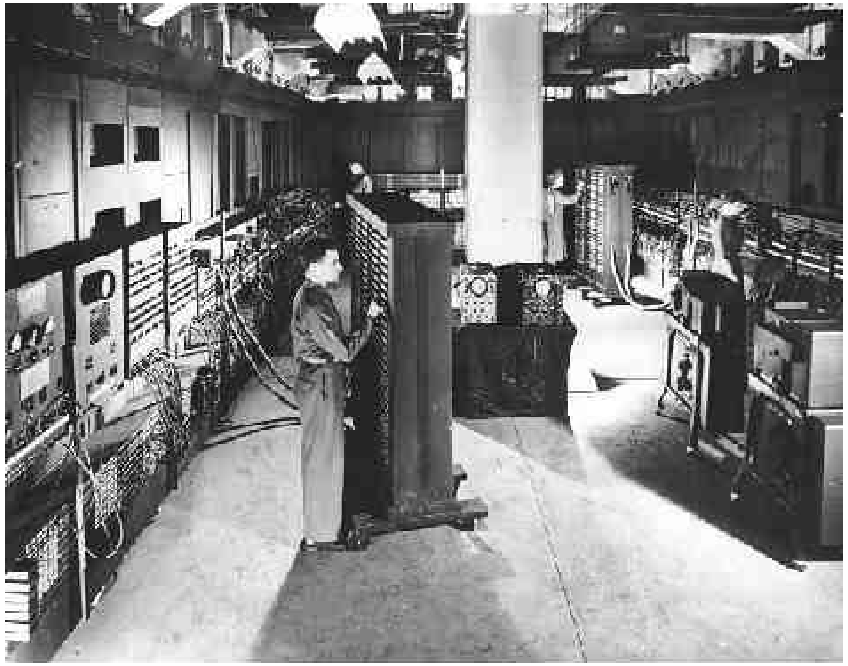
[1] It will be appreciated that the talks are not too technical. Email of J. Sifakis, Sun Mar 31 22:33:11 2002.

What is (or should be) the essential preoccupation of computer scientists?

The production of reliable software, its maintenance and safe evolution year after year (up to 20 even 30 years).

Computer hardware change of scale

The 25 last years, computer hardware has seen its performances multiplied by 10^4 to 10^6;



ENIAC (5000 flops)

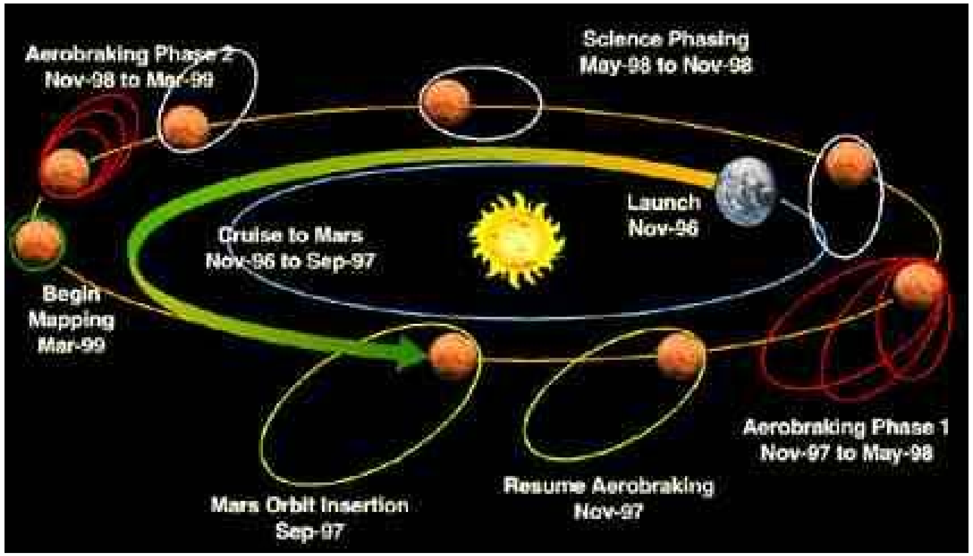
Intel/Sandia Teraflops System (10^{12} flops)

The information processing revolution

A scale of 10^6 is typical of a significant **revolution**:

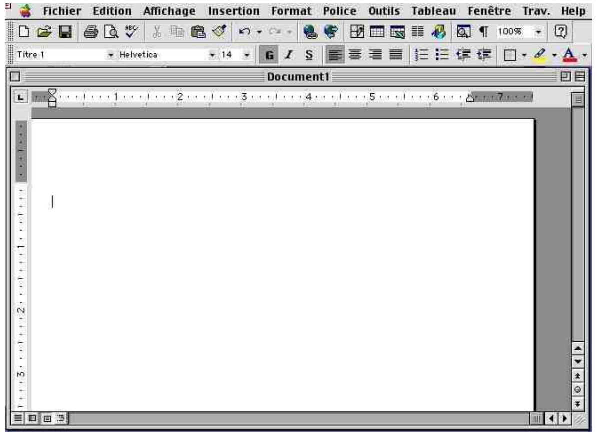
– Energy: nuclear power station / Roman slave;

– Transportation: distance Earth — Mars / Paris — Nice



Computer software change of scale

- The size of the programs executed by these computers has grown up in similar proportions;
- **Example 1** (modern text editor for the general public):
  - $>$ 1 700 000 lines of C [3];
  - 20 000 procedures;
  - 400 files;
  - $>$ 15 years of development.



[3] full-time reading of the code (35 hours/week) would take at least 3 months!

Computer software change of scale (cont'd)

- **Example 2** (professional computer system):
  - 30 000 000 lines of code;
  - 30 000 (known) bugs!

Bugs

- Software bugs
  - whether anticipated (Y2K bug)
  - or unforeseen (failure of the 5.01 flight of Ariane V launcher)

  are quite frequent;
- Bugs can be very difficult to discover in huge software;

Bugs

- Software bugs
  - whether anticipated (Y2K bug)
  - or unforeseen (failure of the 5.01 flight of Ariane V launcher)

  are frequent;
- Bugs can be very difficult to discover in huge software;
- **Bugs can have catastrophic consequences either very costly or inadmissible (embedded software in transportation systems);**

The estimated cost of an overflow

- **$ 500 000 000**
- Including indirect costs (delays, lost markets, etc):

  $ 2 000 000 000

Responsibility of computer scientists

- The paradox is that the computer scientists do not assume any responsibility for software bugs (compare to the automotive or avionic industry);
- Computer software bugs can become an important societal problem (collective fears and reactions? new legislation?);

$\Longrightarrow$ It is absolutely necessary to widen the full set of methods and tools used to eliminate software bugs.

Capability of computer scientists

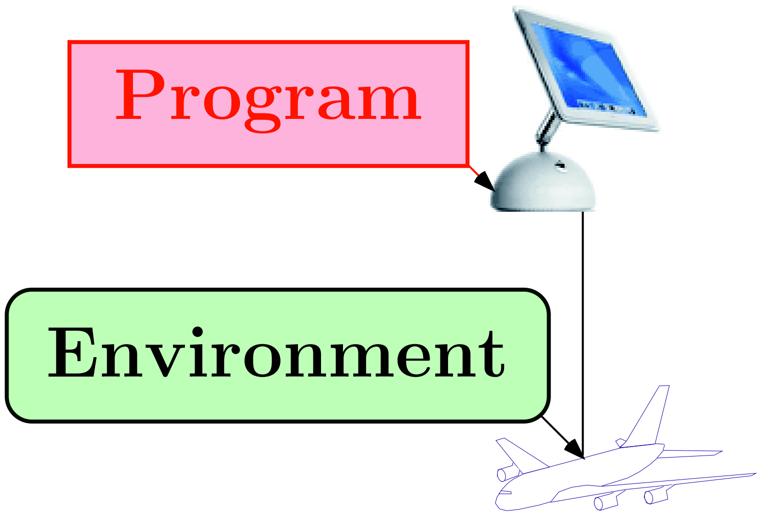
- The intellectual capability of computer scientists remains essentially unchanged year after year;
- The size of programmer teams in charge of software design and maintenance cannot evolve in such huge proportions;
- Classical manual software verification methods (code reviews, simulations, debugging) do not scale up;
- So we should use computers to reason about computers!

Capability of computers

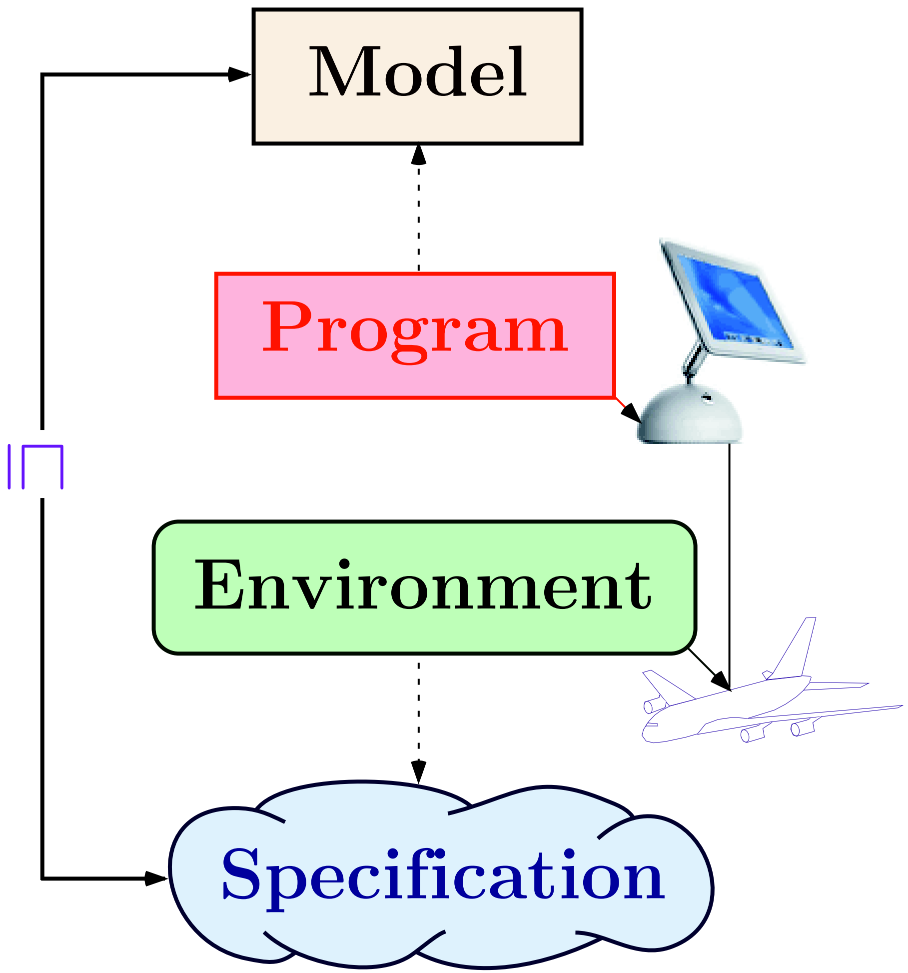
- The computing power and memory size of computers double every 18 months;
- So computer aided verification will scale up, scale up, scale up, scale up, scale up, scale up, scale up, scale up, scale up, scale up, scale up, scale up, ...;
- But the size of programs grows proportionally;
- And correctness proofs are exponential in the program size;
- So computers power growth is ultimately **not significant**.

Formal Methods

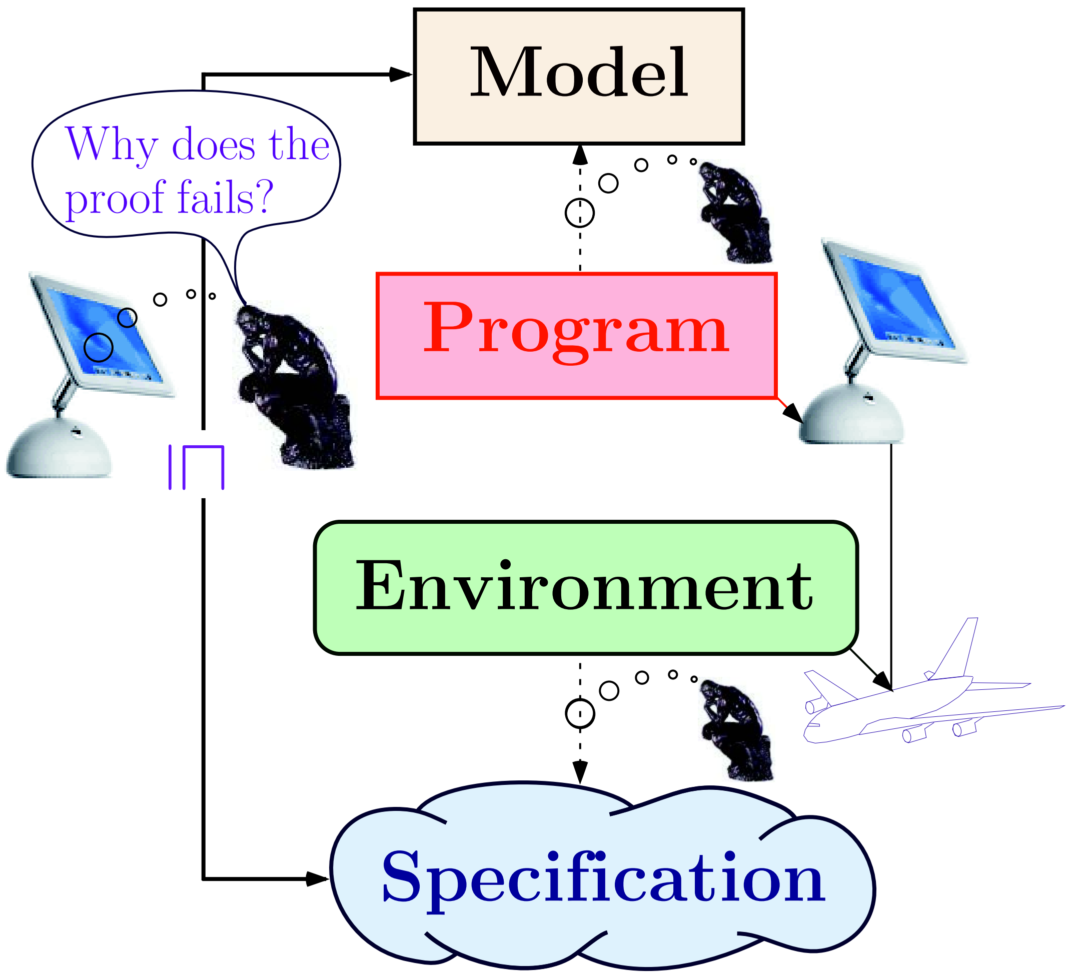
Computer Systems



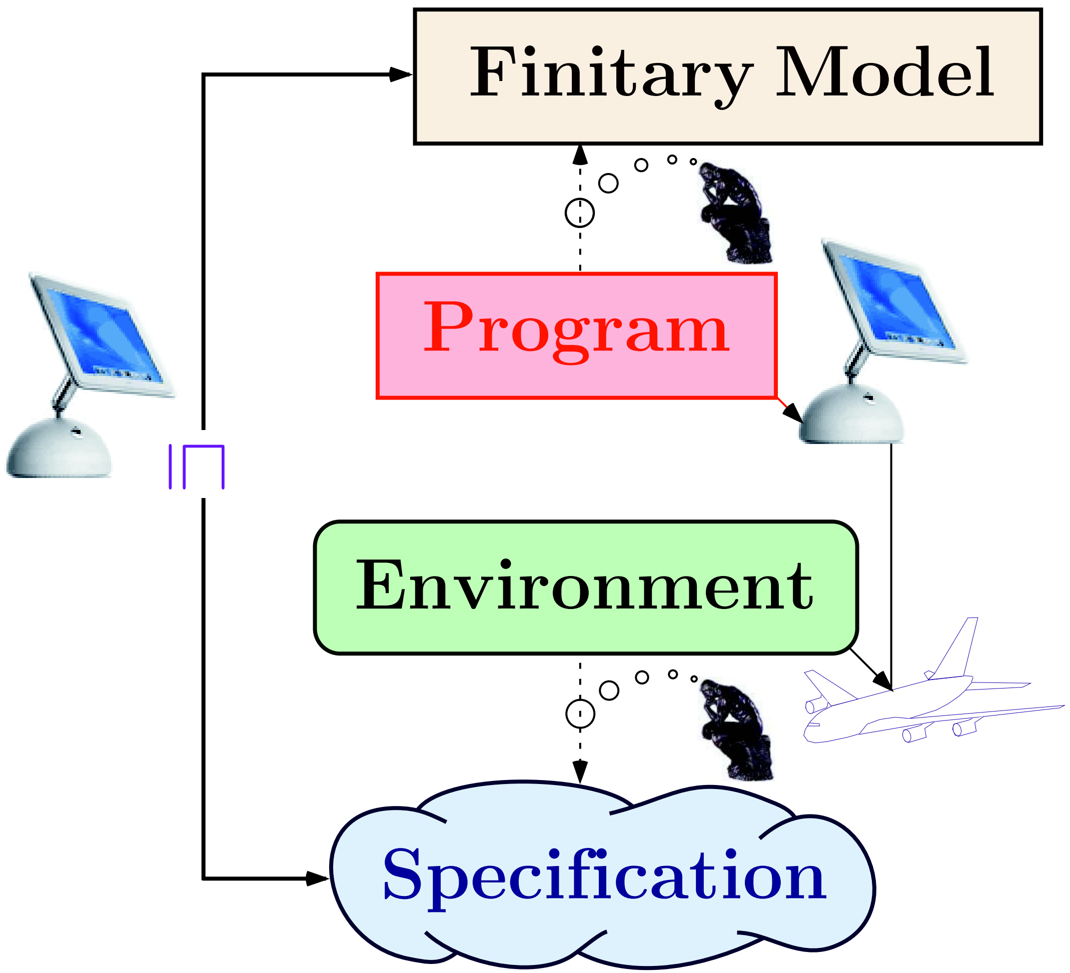
Formal Methods



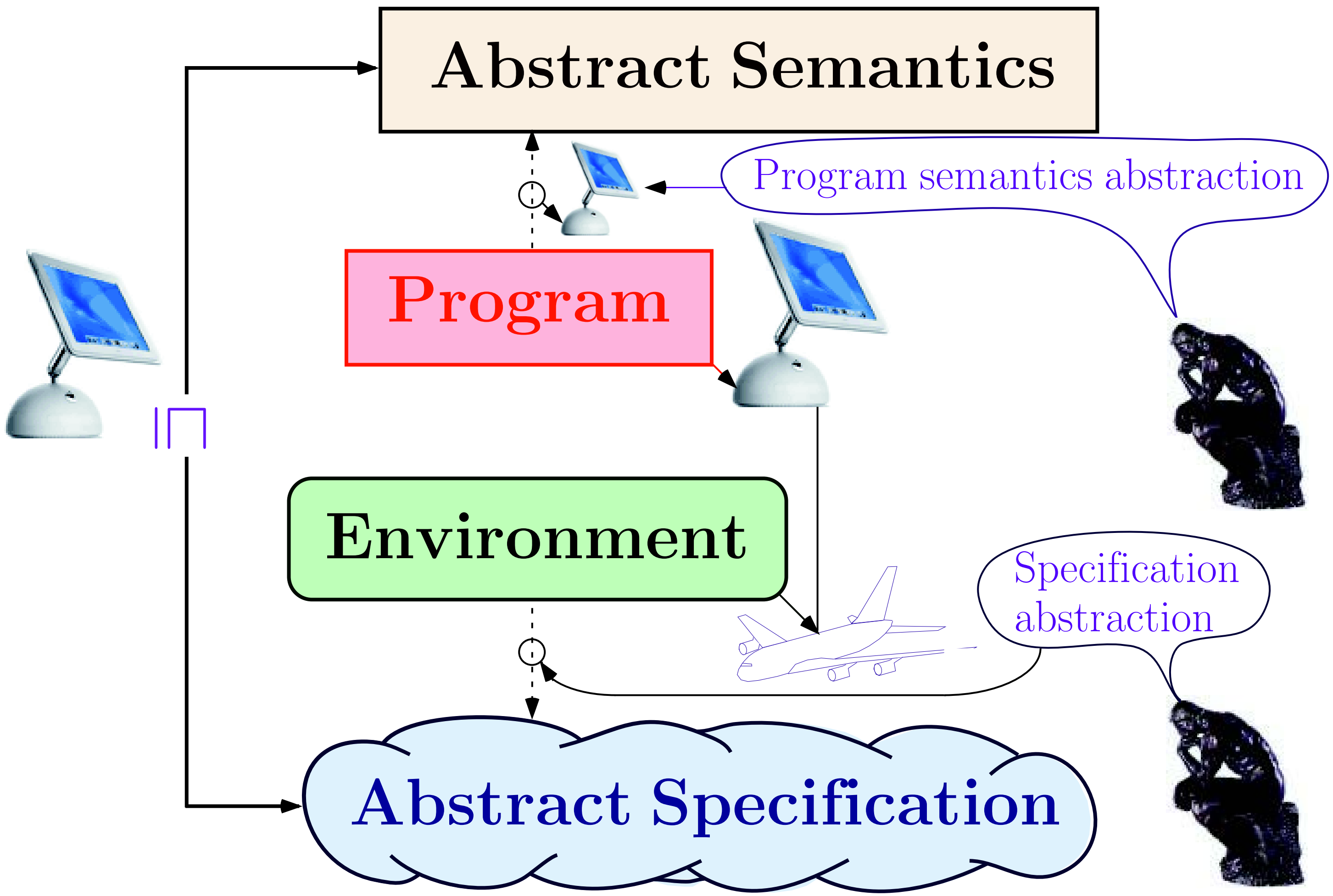
Deductive methods



Model Checking



Static Program Analysis



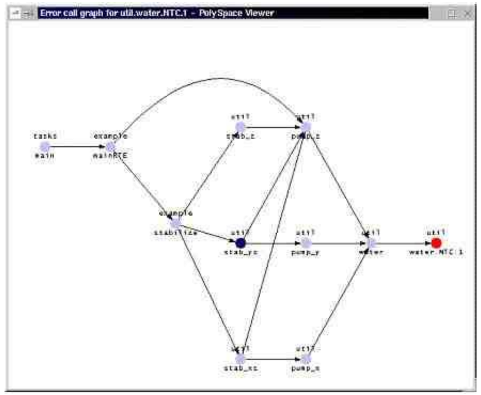
General-Purpose Static Program Analyzers

"The first product to automatically detect 100% of run-time errors at Compilation Time

Based on Abstract Interpretation, PolySpace Technologies provides the earliest run-time errors detection solution to dramatically reduce testing and debugging costs with :

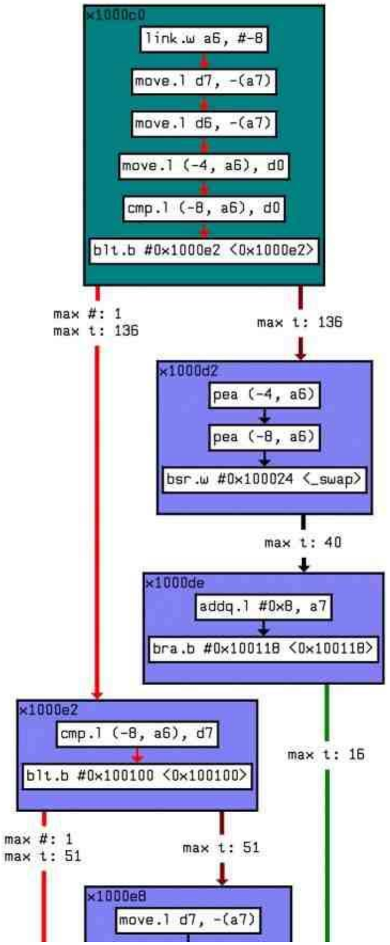
- No Test Case to Write
- No Code Instrumentation
- No Change to your Development Process
- No Execution of your Application" [4]

```
/* arithmetic exception */
void arith_1 (float alpha, double *y) {
    *y = (1.5 + cos ((double)(alpha)))/5.0;  /* 0.1 <= y <= 0.5 */
}

/* arithmetic exception */
void arith_2 () {
double v;
double p;
double y;
float u = random_float();
    arith_1 (u, &v);
    p = v - 0.75;
    y = sqrt (p);
}

/* unreachable or dead code by linear constraint */
void unr () {
   int x = random_int();
   int y = random_int();
       if (x > y) {
              x = x - y;
              if (x < 0) {
                     x = x + 1;
              }
       }
}
```



[4] http://www.polyspace.com/

Special-Purpose Static Program Analyzers



"The underlying theory of abstract interpretation provides the relation to the programming language semantics, thus enabling the systematic derivation of provably correct and terminating analyses." [5]

---

5 http://www.absint.com/pag/

Deductive methods

Model-checking

Static analysis

Abstract Interpretation

I will try to explain why tomorrow morning!

Challenges

Challenges for abstract interpretation

- Semantics of programming languages;
- Separate analysis (modules and libraries);
- Expressive non-numerical abstract domains;
- Liveness properties;
- Probabilistic properties;
- Automatic combination of abstractions;
- Automatic determination of the origin of the loss of precision;
- User interaction for refinement;
- Decomposition of complex properties;
- Proving the correctness of static analysers;
- . . .

All fascinating problems you are probably not interested in!

Societal challenge

- The correctness of computerized systems is essential to modern societies;
- This is hard to explain to the public and politicians;
- We should be able to popularize computer science (including formal methods)!

Research management challenge

- The development of new fundamental ideas requires 5 to 10 years;
- This timing is hardly compatible with the current short term management of research:
  - short thesis (2-3 years),
  - short projects (2 years) on technocratically selected themes,
  - high publication rate (> 3 per year);
- More flexible and liberal research management schemes are required!

Industrialization challenge

- Transfer to industry is required, tighter interaction through tools is a good way;
- The development cost of a high-quality academic prototype must be multiplied by 10 to 20 for a pre-industrialization;
- An effective support for industrialization of research is highly needed;

Educational challenge

- High-quality computer scientists are missing;
- We cannot attract students by teaching myriads of micro-techniques and partial results;
- A synthetic view/theoretisation of field is required!

Scientific challenge

- The computer industry has finally or will shortly understand that quality is a definite problem;
- We are faced with fundamental complexity limitations which cannot be solved by multiplying experiments in the small;
- The only way to think in the large is by divide and conquer!

THE END, THANK YOU