

Can **mechanical** formal methods solve the ultimate verification problem?

IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001

 $\triangleleft \triangleleft \triangleleft -2 - 1 \blacksquare - \triangleright \triangleright \triangleright$

(C) P. COUSOT

- Program typing;
- Static program analysis;
- Abstract model checking;
- Deductive methods;
- Combinations of mechanical formal methods.

Consider decidable analyses only, by restricting both on specifications (allowed types) and on programs;

Clean presentation of the type analysis (inference algorithm) through an equivalent logical formal system (type verification);

Extended to complex data structures, polymorphism, exceptions and separate modules in a way that scales up for large

¹ Obviously the ultimate verification problem is restricted to the unhappy many who are completely unable to derive their programs correctly from sound and complete formal specifications.

Can a single mechanical formal method ultimately solve the verification problem?

NO!

User designed abstraction: derive a program finite abstract model by abstract interpretation, prove the correctness of the abstraction by deductive methods, later verify the abstract model by model-checking;

 $\blacktriangleleft \triangleleft \lhd -5 - ! \blacksquare - \triangleright ! \blacksquare \triangleright$

³ even 1 400 000 lines for control-independent very weak properties.

IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001

© P. Cousot

Possible Alternative: Combine Empirical and Formal Methods

IFIP WG2.3 Santa Cruz Meeting, January 7-12, 2001

© P. Cousot

Fundamental limitation [1]: 1°) abstraction discovery and 2°) abstract semantics derivation is as difficult as doing the proof! (resp. 1°) invariant discovery & 2°) invariant verification).

P. Cousot. Partial completeness of abstract fixpoint checking, invited paper. In B.Y. Choueiry and T. Walsh, eds, Proc. 4th Int. Symp. on Abstraction, Reformulations and Approximation, SARA '2000, Horseshoe Bay, TX, USA, LNAI 1864, pp. 1–25. Springer-Verlag, 26–29 July 2000.

Can some combination of formal methods ultimately solve the verification problem?

NO!

IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001

© P. Cousot

Example: Abstract Program Testing

Debugging Run the program On test data Checking if all right

Providing more tests Until coverage

Example: Abstract Program Testing

Debugging	Abstract testing
Run the program	Compute the abstract semantics
On test data	Choosing a predefined abstraction
Checking if all right	Checking user-provided abstract assertions
Providing more tests	With more refined abstractions
Until coverage	Until enough assertions proved or no predefined abstraction can do.

IFIP WG2.3	Santa Cruz	Meeting	lanuary	7-12	2001

4 3 3 - 8 - 1 3 - 2 3 5

© P. Cousot

Example: Abstract Program Testing

⁴ thus replacing infinitely many test data.

◄ ≪ < - 8 - 1 **■** - ▷ ▷ ►

```
© P. Cousot
```

ruz Meeting, January 7–12, 2001

 $\blacktriangleleft \triangleleft \lhd -10 - | \blacksquare - \triangleright \triangleright \triangleright \models$

© P. Cousot

A Tiny Example

Abstract testing

No

assertions

The user provides the program specifications using predefined

Compute the abstract semantics

Choosing a predefined abstraction

Checking user-provided abstract

With more refined abstractions

Until enough assertions proved or no predefined abstraction can do.



⁶ similarly to different test data.

Debugging

On test data

Run the program

Checking if all right

Providing more tests

Until coverage

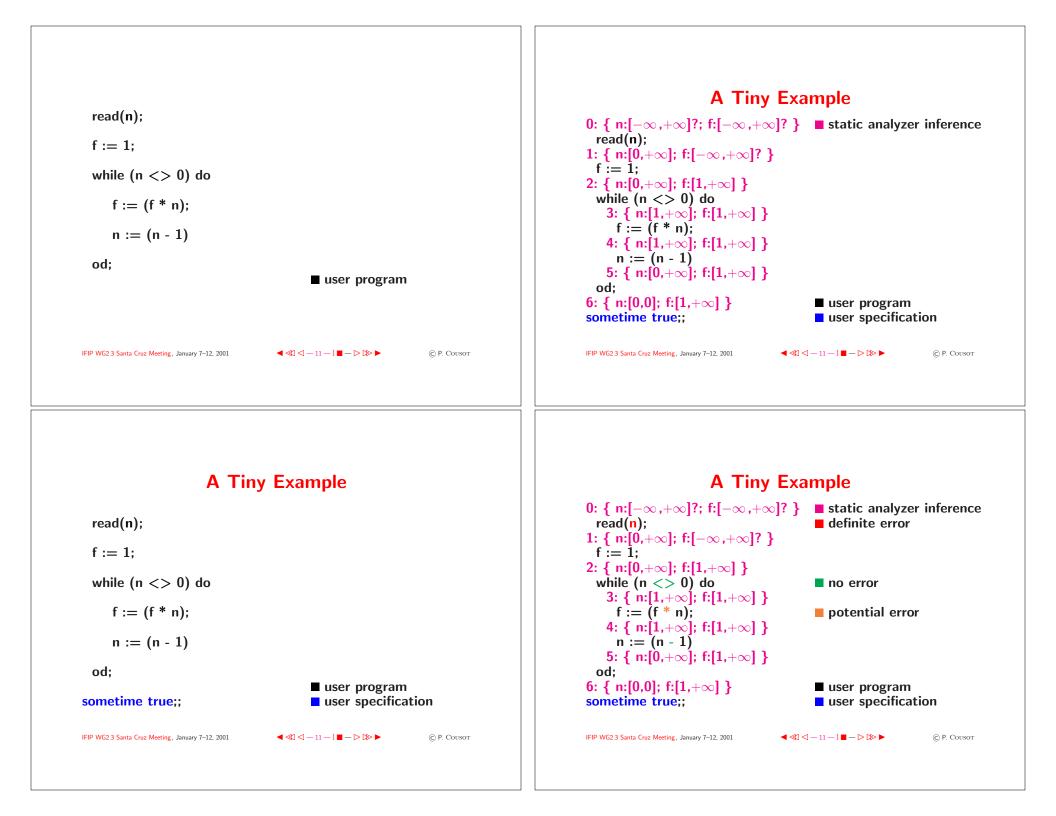
More details needed?

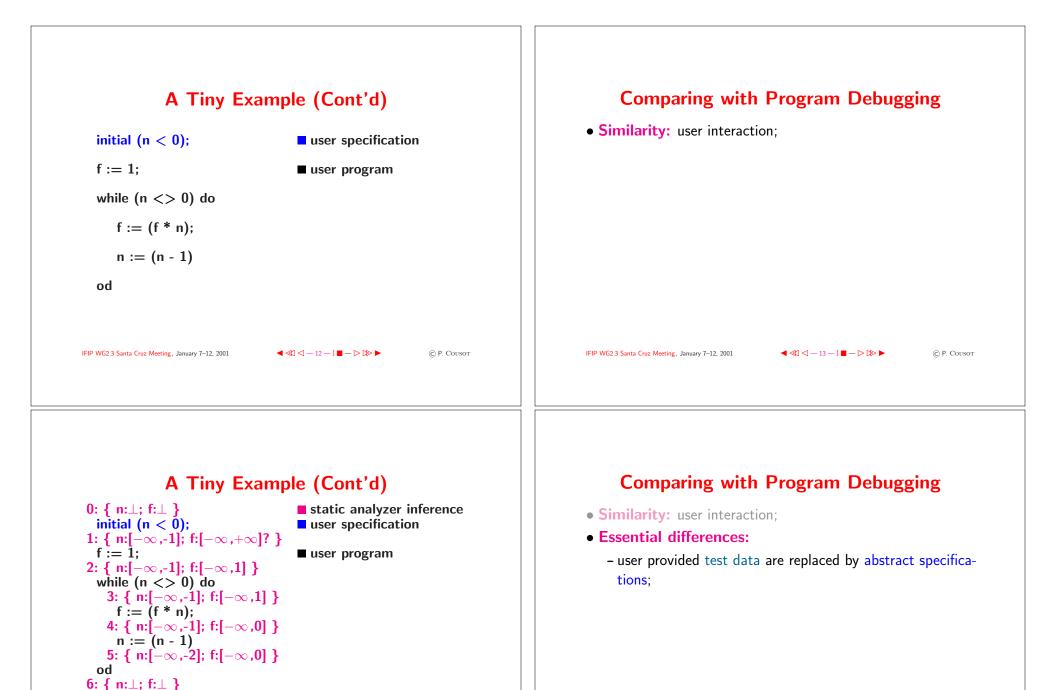
abstractions⁴;

		IFIP	WG2.3	Santa	Cru

"The program is evaluated by abstract interpretation of the formal semantics of the program ⁵:

Yes





IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001

 $\blacktriangleleft \triangleleft \neg -12 - | \blacksquare - \triangleright \square \triangleright \triangleright$

© P. Cousot

IFIP WG2.3 Santa Cruz Meeting, January 7-12, 2001

 $\blacktriangleleft \triangleleft \lhd -13 - |\blacksquare - \triangleright \square \triangleright \blacktriangleright$

© P. Cousot

Comparing with Program Debugging

- Similarity: user interaction;
- Essential differences:
 - user provided test data are replaced by abstract specifications;
 - evaluation of an abstract semantics instead of program execution/simulation;

IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001

C P. Cousot

Comparing with Program Debugging

• Similarity: user interaction;

• Essential differences:

- user provided test data are replaced by abstract specifications;
- evaluation of an abstract semantics instead of program execution/simulation;
- one can prove the absence of (some categories of) bugs, not only their presence;

Comparing with Program Debugging

- Similarity: user interaction;
- Essential differences:
 - user provided test data are replaced by abstract specifications;
 - evaluation of an abstract semantics instead of program execution/simulation;i
 - one can prove the absence of (some categories of) bugs, not only their presence;
 - abstract evaluation can be forward and/or backward (reverse execution).

```
IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001
```

 $\blacktriangleleft \triangleleft \neg - 13 - 1 \blacksquare - \triangleright \square \triangleright \blacktriangleright$

Comparing with Abstract Model Checking

- Similarities:
 - use of specifications instead of test data sets;

IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001

 $\blacktriangleleft \triangleleft \neg - 13 - 1 \blacksquare - \triangleright \square \triangleright \triangleright$

© P. Cousot

IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001

© P. Cousot

Comparing with Abstract Model Checking Comparing with Abstract Model Checking (Cont'd) • Essential differences: • Similarities: - use of specifications instead of test data sets; - reasoning on the concrete program (not on a program model); - ability to automatically produce counter-examples⁷; - no attempt to make a one-shot complete formal proof of the specification; ⁷ or specifications of infinitely many such counter-examples in the case of abstract program testing. IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001 $\blacksquare \blacksquare \blacksquare - 14 - 1 \blacksquare - \triangleright \blacksquare \blacktriangleright$ © P. Cousot IFIP WG2.3 Santa Cruz Meeting, January 7-12, 2001 © P. Cousot

Comparing with Abstract Model Checking

• Essential differences:

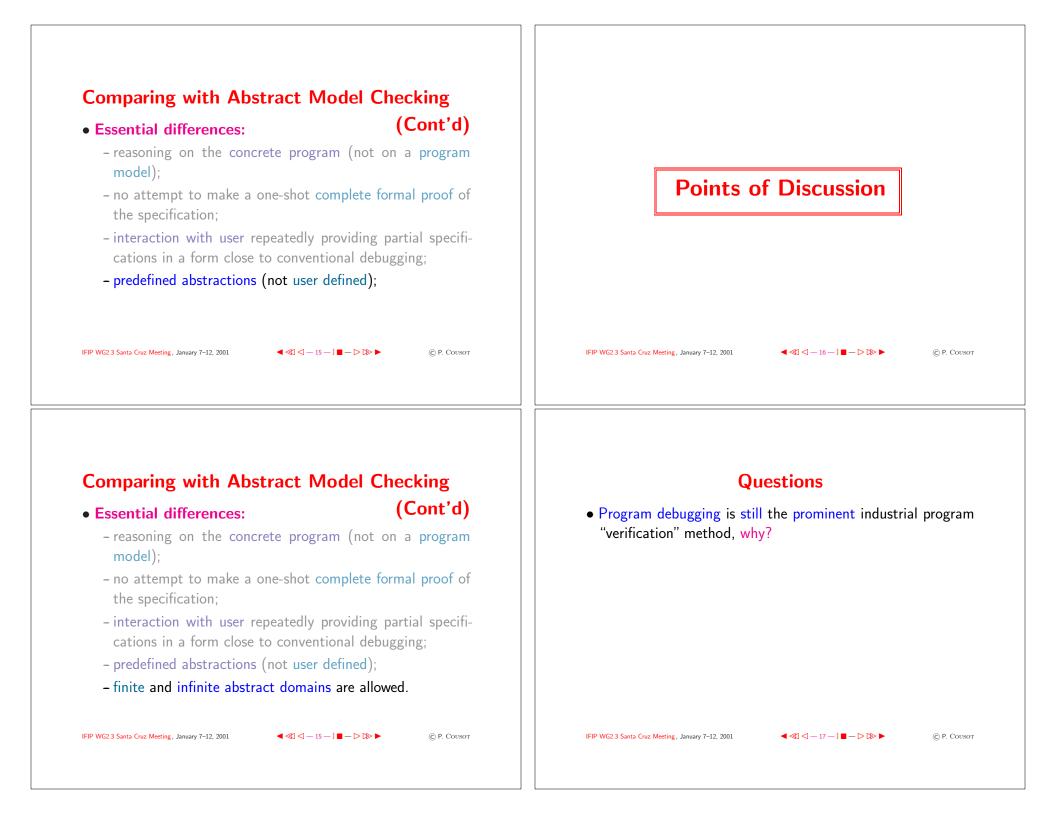
- (Cont'd)
- reasoning on the concrete program (not on a program model);

Comparing with Abstract Model Checking

- Essential differences:
 - reasoning on the concrete program (not on a program model);
 - no attempt to make a one-shot complete formal proof of the specification;
 - interaction with user repeatedly providing partial specifications in a form close to conventional debugging;

IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001

(Cont'd)



Questions

- Program debugging is still the prominent industrial program "verification" method, why?
- Full program verification by formal methods is either impossible (e.g. typing/program analysis) or costly since it ultimately requires user interaction (e.g. abstract model checking/deductive methods) so are mechanized formal methods widely applicable?

< <1 <1 -17 -1 ■ - ▷ ▷ ►

© P. Cousot

Questions (Cont'd)

If for large and complex programs, low cost complete verification by mechanized formal methods is not viable then:

• Universal and reusable hence commercializable abstractions lead to cost-effective⁸ and automatic program analyzers so can approximate program analysis be enhanced to partial program verification?

⁸ Less than 0.25\$ per program line costing 50 to 80\$.

IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001

 $\blacktriangleleft \triangleleft \neg - 18 - 18 - 2 \square$

© P. Cousot

Questions

- Program debugging is still the prominent industrial program "verification" method, why?
- Full program verification by formal methods is either impossible (e.g. typing/program analysis) or costly since it ultimately requires user interaction (e.g. abstract model checking/deductive methods) so are mechanized formal methods widely applicable?
- For program verification, semantic abstraction is mandatory but difficult whence hardly automatizable so can abstractions be designed by programmers?

C P. COUSOT

Questions (Cont'd)

If for large and complex programs, low cost complete verification by mechanized formal methods is not viable then:

- Universal and reusable hence commercializable abstractions lead to cost-effective⁸ and automatic program analyzers so can approximate program analysis be enhanced to partial program verification?
- Otherwise, if user interaction is definitely needed, can abstract program testing be viable alternative to both the exhaustive search of model-checking and the partial exploration methods of classical debugging?

IFIP WG2.3 Santa Cruz Meeting, January 7-12, 2001

IFIP WG2.3 Santa Cruz Meeting, January 7-12, 2001

⁸ Less than 0.25\$ per program line costing 50 to 80\$.

IFIP WG2.3 Santa Cruz Meeting, January 7–12, 2001

