A categorial mistake in the formal verification debate

Javier Blanco
Pio García

Facultad de Matemática Astronomía y Física
Facultad de Filosofía y Humanidades
Universidad Nacional de Córdoba

16/06/08

Contenidos

Introduction

Algorithms and programs

On models

Conclusion

E-CAP tracks

- Philosophy of Computer Science
- Ontology (Distributed Processing, Emergent Properties, Formal Ontology, Network Structures, etc)
- Computational Linguistics
- Global Information Infrastructures
- Philosophy of Information and Information Technology (Including: Information as structure; Semantic information)
- Interdisciplinary Approaches to the Problem of Consciousness and Cognition

E-CAP tracks (cont.)

- Computer-based Learning and Teaching Strategies and Resources & The Impact of Distance Learning on the Teaching of Philosophy and Computing
- IT and Gender Research, Feminist Technoscience Studies
- Information and Computing Ethics
- Biological Information, Artificial Life, Biocomputation
- New Models of Logic Software
- "Intersections" - e.g., work at the crossroads of logic, epistemology, philosophy of science and ICT/Computing, such as Philosophy of AI
- Ethical and Political Dimensions of ICTs in Globalization

Can programs be formally verified?

Two extreme positions

- computer programs are mathematical expressions (Hoare 1986, p. 115),
- program verification is an impossible and undesirable goal for computing science (De Millo, Lipton and Perlis 1979).

Which position is adopted has consequences concerning the nature of computer science (and its practice) and its relation with mathematics.

Two different criticism of program verification

- De Millo's et al. article started a long debate about formal verificaton. They claim that the differences between mathematics and computational science appear in the practices of each comunity.
- Fetzer attemps to reach the same conclusion: the impossiblity of program verification. However, Fetzer follows a completely different approach. He claims that trying to verify programs is a categorial mistake, that programs as such cannot be verified since they are causal rather than mathematical structures.

The program verification debate

- The program verification debate began several decades ago. McCarthy: *debugging* should be replaced by *formal proof*. these proofs could be checked by computational tools (McCarthy, 1962).
- But, the "agenda" was determined by De Millo, Lipton and Perlis 1979 article. They made a distinction between
  - proof, an incomplete draft of a logical demostration, and
  - demonstration, which should in principle complete all the logical steps to ensure the theorem,

  a proof only represents "one step towards reliability".
- The differences between mathematics and computational science appear in the practices of each comunity.

Contenidos

Introduction

Algorithms and programs

On models

Conclusion

Fetzer's position

Fetzer distinguishes

- algorithms which are indeed mathematical expressions and hence suitable to mathematical or logical verification
- programs which correctness can only be tackled by using empirical methods.

Different conceptions of programs

program is used with different meanings.

1. an algorithm,
2. encodings of algorithms,
3. encodings of algorithms that can be compiled,
4. encodings of algorithms that can be compiled and executed by a machine (Fetzer p. 1058).

A program in the sense 1 and 2 - in fact an algorithm- could be verified.

Qualitative distinctions

- Fetzer supposes that there is a sharp and *qualitative distinction* among program and algorithm, as causal process and logical structures respectively.
- His argument assumes that if an algorithm is executed in a physical machine, then it falls under causal relationships and therefore it cannot be understood only through logical properties.
- Hence, we cannot pursue a complete verification of a program.

Ontological assumptions

- It is unclear why mathematical proofs are qualitatively different from programs.
- Apparently, some kind of physical medium is needed also for mathematical proofs.
- These assumptions seem to lead to certain kind of ontological dualism.
- We do not need, however, to assume a dualistic position when we suggest a *relative* independence of software with its physical realization.

Hoare quoted by Fetzer

When the correctness of a program, its compiler, and the hardware of the computer have all been established with mathematical certainty, it will be possible to place great reliance on the results of the program, and predict their properties with a confidence limited only by the reliability of the electronics.

A quotation from Fetzer

As we discovered, the crucial problem confronting program verification is establishing the truth of claims of form (C)[1] *above, which might be done in two possible ways. The first is to interpret rules and axioms of form (C) as definitional truths concerning the abstract machine thereby defined. The other is to interpret rules and axioms of form (C) as empirical claims concerning the possible behavior of the target machine thereby described.*

[1] I c O - input I causes output O

Axioms as prescritpions

But, axioms of this form can also be understood as prescribing the possible behaviors the target machine can perform

Experiments or tests?

- Fetzer claims that programming is an applied mathematics activity, in Hempel's sense.
- In that case, we could make experiments to try to refute the whole theory of programming.
- This is very different to have grounds to consider testing the only choice to get knowledge about a particular program.

Programs as theories

Fetzer claims:

> *The limitations involved here are not merely practical: they are rooted in the very character of causal systems themselves. From a methodological point of view, it might be said that programs are conjectures, while executions are attempted -and all too frequently successful- refutations (in the spirit of Popper [32, 331).*

Contenidos

Introduction

Algorithms and programs

On models

Conclusion

Main thesis

- We criticize Fetzer's strategy that supposes a sharp and qualitative distinction between algorimthms and programs.
- We believe that algortihm and program are just two aspects of the same thing (let's call it *program*).
- The program is a formal object which can prescribe certain behaviours to certain machines.
- Those behaviours can be specified formally (mostly as an input/output relation) and one can mathematically verify this fact .

On testing

- According to de Millo et al. the testing operation against the specification is still needed since verifications are tiresome and not practically feasible.
- But the information that any test produces is only a constructive proof (in a somewhat unusual medium) that one execution belongs to the specified behaviors for this program.
- If the formal verification can be constructed, it obviously would imply all the knowledge that all the tests (perhaps an infinite number) can produce.
- In any case, the only debate here is about practical matters.

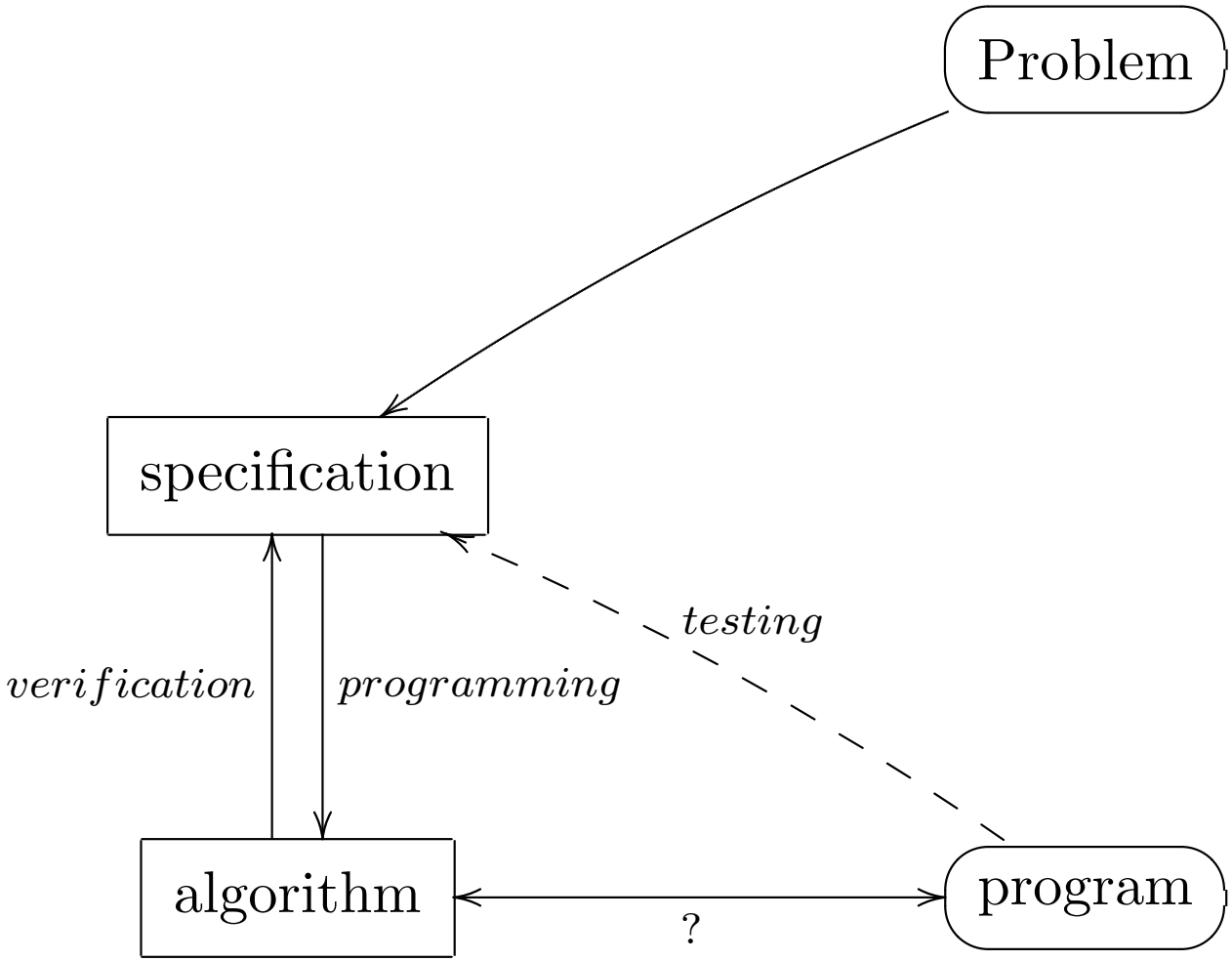

Figure: Programs as causal systems

Testing and validation

- We propose that testing may be seen as a step towards bridging the gap between the formal and the real world, comparing what has been modeled with the necessary informal original problem.
- Actually, this sort of testing can be replaced by a validation process, which can be performed earlier directly from the specification.
- The triangle with verification, validation and testing should commute.

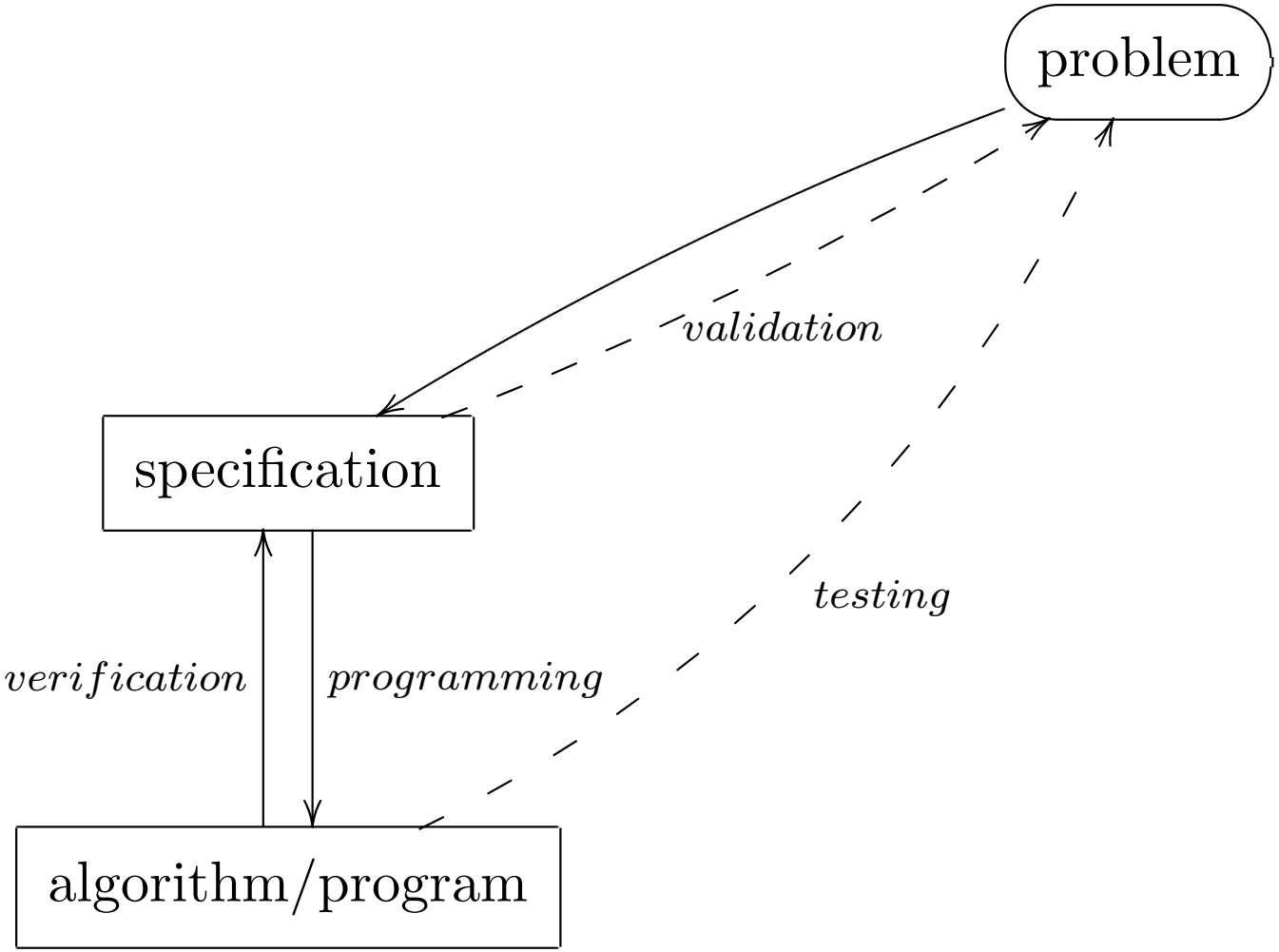

Figure: Programs as formal systems

Contenidos

Introduction

Algorithms and programs

On models

Conclusion

Final words

- In this work we tried to show a way to answer Fetzer's problem, which is founded in the supposition that there is a sharp distinction between programs -as causal process- and algorithms -as logical structures.
- As a consequence, we would not be able not make formal verifications of programs. But we suggest that if this is true, then we cannot make mathematical proofs either. We also suggest that Fetzer's proposal leaves many unsolved problems.
- We propose, instead of a qualitative distinction between algorithms and programs, that there are logical and causal aspects in programs, which explain both the possibility of formal verification and the possibility of material errors as well.