

Realist consequence, epistemic inference, computational correctness

Giuseppe Primiero

FWO – Research Foundation Flanders
Centre for Logic and Philosophy of Science, Ghent University



Giuseppe.Primiero@UGent.be

<http://www.philosophy.ugent.be/giuseppeprimiero/>

3rd April 2013

Workshop on the Scope of Logic through History
Unilog, Rio de Janeiro

- 1 A novel paradigm
- 2 Computational Correctness
- 3 Final Remarks

On logical systems

- The scope of logic is defined through the development of logical notions, relations, (meta-theoretical) results;
- Validity and correctness have a crucial role in defining the philosophical status of a logical system;
- This is a historically unstable notion: how does it appear it *today* in view of its historical background?

Validity by logical consequence

- 1 Tradition of the Bolzano-Frege-Quine-Tarski school;
- 2 Consequence as a relation between propositions as truth-bearers, expressing (the obtaining of) corresponding states of affairs;

Definition

A consequence from antecedents to conclusion is valid if the latter is true in the same state of affairs or models where the former are.

Correctness by epistemic inference

- 1 Tradition of assertoric perspective on formulas, since the BHK school;
- 2 Inference as a relation between judgements stating truth by verification of assertion conditions;

Definition

An inference from premises to conclusion is correct if the latter is knowable under the same conditions of the former ones.

Which current picture? A pluralist account

- (Luckily) Historical development has generated no unique framework;
- Both realist and anti-realist interpretations are still good and alive;
- There are a number of other paradigms around: substructural, dynamic, non-monotonic, ...;
- We shall focus on *one* significantly new approach, generated from the previous dialectic: the *computational* paradigm.

The correctness problem in CS

Correctness in the Logical Foundations of Computer Science (since [De Millo et al.(1979)]): *can computer systems satisfy correctly their designers aim?*

- 1 'Correct satisfaction' is a non-redundant expression;
- 2 It refers to the *intention* of the designer;
- 3 It requires properties that are non-standard in previous frameworks;

Correctness as Type reconstruction

Proos-as-programs paradigm and validity in typed (programming) languages:

Given a program p , there exists a specification S and network \mathcal{N} such that $\mathcal{N} \vdash p:S$ is a derivable expression?

New features of logical correctness

- 1 Are conditions for execution and termination of p admissible?
- 2 Are resources in \mathcal{N} reachable?
- 3 Where in \mathcal{N} are processes of p valid as to satisfy S ?
- 4 Finally, how to resolve non-correct executions of p ?

1: Typological Non-neutrality

Not all data are of the same type:

- some processes hold in view of explicit computational content;
- some depend on execution of implicit computational content;
- While normalization requires explicit data, correctness does not.

2: Data Accessibility

Processes need to be able to access data:

- provide meta-data on processes to express accessibility;
- formal extensions to induce labelling, modalities, mobility (not computationally irrelevant extensions) to index locally vs. globally terminating routines.

3: Local Validity

Validity becomes a localised concept:

- terminating programs induce global truth: define validity
- locally bounded programs induce local truth: define admissible computational steps

4: Failure

Local validity and truth admits failure;

Internal Sense	External Sense
[IL1] correctness of routine	[IL3] correctness of data dependency
[IL2] correctness of subcalls recursion	[IL4] correctness of data retrieval

4/2: Handling or Debugging

Failing processes require error handling:

- Errors of definition require re-typing
- Errors of location require resource re-assessment
- Errors of mobility require process re-execution

To Sum up

Computational Correctness admits:

- logical distinction among data (data are not all the same);
- metadata (the when/how/where of data is relevant);
- restricted well-formedness and termination (not every process satisfies these properties).
- failure is possible.

A new notion of valid logical process

Definition

A process P is valid iff at its execution, P is capable for any required P' of controlling

- 1 access to location(s) of P' ;
- 2 commands (reading/writing/exec/broadcasting) of P' ;
- 3 correctness of P' (global/local w.r.t. its locations).





Conclusion

- 1 The computational paradigm offers a more empirically-oriented, practical view on logical validity;
- 2 It embeds notions of locality, agency, partiality;
- 3 It represents a true extension of the 'old' paradigms.

Thanks!

Comments? Questions?

References

-  N. Alechina, M. Mendler, V. de Paiva, and E. Ritter.
Categorical and Kripke Semantics for Constructive S4 Modal Logic.
In *Proceedings of the 15th International Workshop on Computer Science Logic*, volume 2142 of *Lecture Notes In Computer Science*, pages 292 – 307, 2001.
-  G.M. Bierman and V. de Paiva.
On an intuitionistic modal logic.
Studia Logica, (65):383–416, 2000.
-  T. Borghuis and L.M.G. Feijs.
A constructive logic for services and information flow in computer networks.
The Computer Journal, pp.274–289, vol.43, n.4, 2000.
-  M. Burgin.
Super-recursive algorithms, Monographs in computer science, Springer, 2005.

References



B.J. Copeland.

The broad conception of computation.

American Behavioral Scientist, pp.690–716, vol.40, n.6, 1997.



B.J. Copeland.

Hypercomputation

Minds & Machines, pp.461–502, vol.12, 2002.



R. Davies and F. Pfenning.

A modal analysis of staged computation.

Journal of the ACM, 48(3):555–604, 2001.

References

 R. De Millo and J.R. Lipton and A.J. Perlis

Social processes and Proofs of Theorems and Programs.
Communications of the ACM, pp. 271–280, vol.22(15), 1979.

 L. Jia and D. Walker.

Modal Proofs as Distributed Programs.
In *Programming Languages and Systems, ESOP2004*, volume 2986 of
Lectures Notes in Computer Science. Springer Verlag, 2004.

 , R. Milner.

Communication and Concurrency, Prentice-Hall, 1989.

 J. Moody.

Modal logic as a basis for distributed computation.
Technical Report CMU-CS-03-194, School of Computer Science,
Carnegie-Mellon University, Pittsburgh, PA, USA, 2003.

References



T. Murphy.

Modal Types for Mobile Code.

PhD thesis, School of Computer Science, Carnegie Mellon University, 2008.

CMU-CS-08-126.



T. Murphy, K. Crary, and R. Harper.

Type-Safe Distributed Programming with ML5, volume 4912 of *Lectures Notes in Computer Science*, pages 108–123.

Springer Verlag, 2008.



A. Nanevski, F. Pfenning, and B. Pientka.

Contextual modal type theory.

ACM Transactions on Computational Logic, 9(3):1–48, 2008.

References



F. Pfenning and R. Davies.

A judgemental reconstruction of modal logic.

Mathematical Structures in Computer Science, 11:511–540, 2001.



S. Park.

A modal language for the safety of mobile values.

In *Fourth ASIAN Symposium on Programming Languages and Systems*, 2006, pp.217–233, Springer.



G. Primiero.

Constructive contextual modal judgments for reasoning from open assumptions.

In *Proceedings of the Computability in Europe Conference*, 2010.

References



G. Primiero..

A multi-modal dependent type theory for representing data accessibility in a network.

In A. Simpson (ed.), *Proceedings of the Proof Systems for Program Logics Workshop (LICS Affiliated at FLOC 2010, Edinburgh, UK)*, EasyChair Electronic Proceedings, 2010.



A.K. Simpson.

The Proof Theory and Semantics of Intuitionistic Modal Logic. PhD thesis, University of Edinburgh. College of Science and Engineering. School of Informatics, 1994.



P. Wegner.

Why Interaction is more powerful than algorithms.

Communications of the ACM, pp.81-91, vol.40(5), 1997.

References



P. Wegner.

Interactive foundation of computing.

Theoretical Computer Science, pp.315-351, vol.192, 1998.