# Logic and computer programs: a match made in heaven?

Liesbeth De Mol

(Note 1: this writing was driven by a worry that the logic cluster might become too much the work of logician's only and so be too internal)

(Note 2: in what follows, I have written down not a draft of something or an outline of what could become a chapter, but a rather an outline of some ideas I had in relation to the Logic cluster chapter)

What, exactly, are the historical *and* epistemological connections between logic and computer programs? That would be one basic question I would like to see raised within the logic cluster chapter.

One basic assumption that follows from the historico-epistemological viewpoint I stand for is that I do not believe logic can be /strictly/ separated, by humans, from the context in which it is used and produced. In other words, since *we* only know and produce knowledge through our practices (writing; programming; talking; formalizing; etc) it cannot be "pure" but is always affected (or, for some, infected) by the others and other things involved in that practice. Note that this assumption does not necessarily imply an ontological commitment which I prefer not to make.

## 0.1 No captains and no soldiers

In the introduction of the well-known and early collection of papers on the history of computing, mostly written by those involved, one reads:

> "The improbable symbolism of Peano, Russel, and Whitehead, the analysis of proofs by flowcharts spearheaded by Gentzen, the definition of computability by Church and Turing, all inventions motivated by the purest of mathematics, mark the beginning of the computer revolution. Once more, we find a confirmation of the sentence Leonardo jotted despondently on one of those rambling sheets where he confided his innermost thoughts: 'Theory is the captain, and application the soldier.' "

I am not buying the viewpoint of theory (as logic) being the captain and application the soldier since they cannot be strictly separated, historically nor epistemologically. This can be easily shown through some existing work in the history of computing. Most well-known in this group is of course the fact that a lot of so-called theoretical foundations have been retro-fitted to the discourse; eg the idea that the theoretical insights from the 1930s were the historical basis for the first computers.

Neither do I believe that application should be seen too much as the captain. This is the more popular view since the 1980s, not just in our context but that of science at large – see the historical analysis of Forman[1].

I am sure that I and several others would be able (and hopefully also willing) to write on this by considering different cases from the history of Computing poinitng at captain-like statements wrt to logic and what the conseuences of that have been/are. I think one of them was certainly an intensifying of certain communication gaps.

Besides, it would also be good to have some clear cases of how logic gets mixed up and infected by practice (eg Tomas' work on how types in logic are different from types in programming)

## 0.2 Logic and Control

Much motivation for developing logical and formal approaches are driven by the idea of a kind of perfect program, where, perfect means, being in control of everything; having certainty that there are no errors and that there will be no errors. But, as we all know, being in control also means that one is limiting the possibilities of something else (the machine, the user, the language, etc) – that is fine but it is basic to be open about that too and it would be nice to see some more reflection on that.

Also,the history of computing however is full of examples where lack of control is exactly what is needed to make progress. One basic question might be in which contexts such complete control is desired and if that is achievable.

Perhaps one well-known example is Turing's insistence on the machine's ability to make mistakes. This seems an important precondition in order to have genuine machine intellegince.

## 0.3 Communication gaps and opacity

I think it might be quite important to think a bit deeper about these two aspects wrt Logic. I alrady mentioned communication gaps at the end of Sec. 0.1. What about opacity? Isn't it a purpose of logic to create transparancy? It would certainly be interesting to see some aspects of this introcuced in the chapter. One question one could raise here is why formal approaches have not managed to "control" this issue?

One other aspect in connection to the commnuication gaps is the question of the relation between logic and the other three clusters. Perhaps it would be

[1]Forman, P., "The Primacy of Science in Modernity, of Technology in Postmodernity, and of Ideology in the History of Technology," History and Technology, vol. 23, nr. 1-2, 2007, pp. 1–152.

nice to have a section considering Logic in relation to these also from a more historical viewpoint.

## 0.4 The formal and informal

A chapter is being planned on this topic but perhaps some aspects of that might also be raised in this cluster chapter, viz. the general problem or question if and in which contexts we can draw a line (cfr e.g. the so-called separation of concerns but also the computable vs the so-called non-computable) between the twosides of the and. Also one can ask in how far either side of the *and*, when put on stage in opposition or in a hierarchical position to the other, can result in "good" programs.

Related to the latter: do we know a purely "logical" program (this relates back to Nicola's outline); do we know a purely "illogical" program?

## An example program

For the sake of coherence, it is asked that each chapter would start with a specific program. One option (but this is just a proposal to start thinking about this – I am sure other ptograms are better suited) is one of Curry's programs written in his notation. This has several advantages:

- it is quite hisorical (late 1940s)
- it allows to connect the machine level with logical ambitions
- to connect developments in logic that were independent of computer science (1920s and 30s) to the need for automated programs
- it allows to discuss the significance (or lack thereof) of a notation
- allows to discuss the idea of an algebra of programs (formal properties of programs like identity and associativity)
- use of logic to avoid errors