The Meaning of « Formal »

Pierre Wolper
University of Liege

When is a method formal? It is somewhat surprising that this simple question is rarely asked when discussing formal methods research. After all, we know when a method is formal, don't we? Nevertheless, let us try to define formality. First, notice that the "formal" in "formal methods" usually refers to the language in which one writes specifications. So a formal method is one based on a formal specification language. When we talk of formal languages, we usually mean languages whose sentences are defined in a mathematically precise way. I would go slightly further and require that the sentences be recognizable algorithmically.

So, is a formal method one that is based on a specification language with algorithmically recognizable sentences? Essentially all existing formal methods do satisfy this definition. However I believe it is much too weak. Indeed, while it is clearly useful to have a language with a checkable syntax, formal methods that do not go beyond this are only of very limited use. Of course, most formal methods also have languages with a "formal semantics". This usually means that the semantics is defined in a precise mathematical way, but nothing more. The question then is: what is the use of a formal language with a formal semantics? I would somewhat provocatively say that it is useless, except if the formal semantics can be meaningfully exploited by an algorithmic tool. For programming languages we have compilers, for specification languages we need checkers or synthesizers that are algorithmic! But, wouldn't it be enough to have a language in which it is possible to prove or derive programs in a semi-algorithmic way, i.e. the proof or derivation requires some user input but is algorithmically checkable. This is better than no algorithmic support, but will only be usable if the work required of the user is very limited. It is unrealistic to expect a formal method to be used if it requires more work than writing the program.

In summary, I would propose to classify formal methods as weak, semi-strong or strong. Weak formal methods are those than only provide syntactic algorithmic support. Semi-strong methods are those that offer algorithmic support for checking what has been done, but that still require user input for proving a property or deriving a program. Strong methods are those that offer full algorithmic support. Obviously one can argue that the inherent undecidability of the problems one is dealing with limits the applicability of strong formal methods. Correct, but I would extend the class of strong formal methods to include methods that are semi-algorithmic (i.e. that might sometimes not provide a result). Furthermore, what can be done with algorithmic or semi-algorithmic methods is already quite impressive and is being very significantly expanded by ongoing research. Finally, I would argue that it is much more useful to have a formal method that has limited capability but is easily usable than to have a very general one in which all you can do in practice is write formulas. To put it metaphorically, one can do much more with a screwdriver than with a universal tool that is too heavy and cumbersome to be moved or handled.