PRINCIPLED FORMAL METHODS IN HCI RESEARCH

S.V. Reeves

The aim of this essay is to suggest that formal methods, for which we give a working definition, have a place in HCI and, more, are a necessity if HCI is to progress and learn from mistakes made in the past in software engineering and artificial intelligence.

1. Trying to define formal methods

First we need to say something about two commonly used terms: formal systems and formal methods. A formal system consists of a language, which is a set of sentences, and a set of rules. A rule can be thought of as telling us how to move from one sentence to another. A concrete example is to think of a simple game where the sentences describe states of play and the rules describe how to move from one state to another. A game like chess is a good example of such a game, where the description of the board at any moment gives the state of play and the rules are the usual rules of chess.

We said that a language is a set of sentences, where a sentence is some string of symbols taken from a set called the alphabet of the language. The language is constructed according to some rules of formation, usually called a grammar. Hence, it turns out that a grammar is itself a formal system. This is a formal system in its simplest and most commonly occurring form, certainly in computer science.

There are many other ways of presenting a formal system, but they all can be reduced to the above definition. However, we have already seen one of the benefits that should come from any good characterization of a concrete idea via a formal system (which is what we mean by formalization). This benefit is the ability to characterize some abstraction so that properties of the abstraction can be worked on away from the muddling details that accompany particular concrete examples of the abstraction. This is a commonly recurring theme because it provides one of the motivations for formalization and formal systems.

One particular property of a formal system is to be able to tell, given your starting sentence, whether a given "destination" sentence will ever be reached. It turns out that there are some sorts of formal system where you can always tell, some where you can sometimes tell and some where you can never tell. The interesting thing is that you can discover, by looking purely at the form of the rules of the formal system, which sort of system you are dealing with. This is the sort of general result that relies on formalization and abstraction in order to even describe a property, let alone prove it.

A "short-cut" way of spotting whether we are dealing with a formal system is to ask the question "Could this be programmed, just by directly implementing the rules?". If it could be then we are dealing with a formal system, albeit presented as a program, which we might think of as a presentation in a "user friendly" format. This also provides us with a test of whether, when somebody says "Here is a formal system to do x", they really have provided a formal system, because if their system cannot in principle be directly turned into a program to do x then they have failed and they are misleading us.

The most usual way of failing is to require, at some point in the use of the formal system, some use of insight or experience or intelligence. To sum up: the whole point of having a formal system to do x is that x can be done with no input of knowledge, intelligence, expertise and so on. It is just a matter of following rules and to ask about the "meaning" behind, or motivating, the rules is to ask a literally meaningless question. This also is what gives rise to the use of the word "formal": the carrying out of the rules is a matter of dealing with form; meaning or content has no place. In short, if there is a formal system to do x then we can write a program to do it.

When we turn to formal methods things are less clear cut. We shall take a formal method to be a set of formal systems which are used together with some operations that are not described by a formal system. This means that the operations which are not in any of the formal systems involved in the formal method are probably not formalizable, at least with our current knowledge. This is why the notion of formal method is hard to pin down; it depends on rather varying ideas like "current knowledge".

The author is in the Computer Science Dept. of Queen Mary and Westfield College.

Authorized licensed use limited to: Universidade Estadual do Norte Fluminense. Downloaded on May 01,2022 at 11:45:34 UTC from IEEE Xplore. Restrictions apply.

2. Why we should use formal methods

In this section we look very briefly at two illustrations which support the case for using formal methods in HCI. The first illustration uses the experience of the AI community to relate a warning. The second illustration puts forward, at a very abstract level, some ways that formalization might be used to advantage and it uses an analogy with developments in software engineering.

2.1. AI and HCI

The state of some HCI research currently is rather like the state of AI during the 60s. During that time there were two ways of doing AI. The more "successful" way was to have some intuition about a problem which led you to decide to write a program which implemented your intuition, in some way. This way of working, however, led to people making claims that were inflated and building systems that, with a small perturbation in the problem, tended to fall apart. This style of research was attacked by the practitioners of the "unsuccessful" way of working. In this way, you take your intuitions and think about them and, probably by consulting an expert or doing some reading, you tried to find out about any work that had been done on formalizing the sort of intuitions that you had. This you did because you realized that it was likely that straightforwardly "implementing you intuitions" would lead to results that you could not explain and so could not be sure about.

An example of this might be of a system where you needed to have some sort of rule base which formalized your intuitions about a subject and which would be updated or otherwise modified in the light of experiment or experience. The "successful" practitioner would use, say, LISP (or some off-the-shelf rule base implementing system), and code up the rules so that the program that resulted could be consulted and updated and so on. A formally trained person might also have been worried about things like consistency, soundness and completeness, especially in the light of possible updating of the rule base. Here a rule base is consistent if and only if it cannot compute the result that two statements both follow from the rules if one of the statements is the opposite, or negation, of the other. This condition can be sharpened by saying that the rule base is consistent if and only if at least one sentence is not computed as following from it. The rule base is sound if and only if the things that follow from it according to the computation "really do" follow and the rule base is complete if and only if all the statements that "really should" follow from it are computed as doing so. The person who took time to think about these questions and find answers to them, and hence a definite, and in the ideal case formal, basis for the program, was not able to demonstrate a working program very quickly, as so was "unsuccessful", but they could be more sure that whatever they did construct had a solid foundation and, just as importantly, could be modified and have its computations explained in a manner which could be used to build-up confidence in its users.

Turning to more recent times, it seems that HCI, with its use of rule bases, use of natural language and the need for languages to both describe and implement these uses, might be in danger of falling into disrepute just as AI did. This would be a disaster, just as it was for AI, which is still, in some areas, tainted with the air of disgrace which comes from a body of research that has been judged to have been based on unclear thoughts, over enthusiasm and, at worst, knowledge that the techniques were questionable but were used nevertheless. This amounts to acting in a completely unscientific manner. The way that AI overcame many of its problems was to realize that it needed to use ideas that had proper, and currently, given the computational setting, that tends to mean formal, foundations. This pattern has also been followed, though largely without the bad science, in software engineering.

2.2. An abstract example

In the setting of HCI the operations that link together the formal systems are likely to be unformalizable because they need to draw on ideas like "ease of use", "naturalness", "cognitive fit" and other judgements that might be suggested or tested by experimentation. The operations would only become formalizable if these ideas were themselves formalizable. This would mean having a program which could make judgements about such things. In short, we would have to be able to sever the current reliance that we have on a person's knowledge, experience or intelligence. This clearly will not happen in the foreseeable future and it is quite reasonable to believe that it will never happen. So, work on formal methods in HCI should clearly be going on to push out the limits of what is describable by a formal system (or program) in the current methods used and also making the non-formal parts as well-defined as possible so that the methods can be repeatedly used in different settings to get results of a repeatable standard.

This will usually start with an attempt to make the expression of the well-known ideas more definite, with the longer-term aim of making the ideas themselves more definite. However, there is a

Authorized licensed use limited to: Universidade Estadual do Norte Fluminense. Downloaded on May 01,2022 at 11:45:34 UTC from IEEE Xplore. Restrictions apply.

danger here: many people seem to fall into the grips of GUS: Gratuitous Use of Symbols. This is the use of symbols not in order to advance towards the goal of more precise expression of an idea that can subsequently lead to formalization of the idea but in order to either make impressive an otherwise unimpressive or useless idea or simply to mislead the reader so that they may think that they are being told something interesting or useful. It is the main technique behind non-principled use of formal methods.

At a rather abstract level we want to look at how some formal systems might be used together with unformalized ideas to attack a certain problem. We imagine trying to design an interface to a program that performs some task given a description of how a person current interacts with the program.

A rather old-fashioned view of might be to write the interface and then train the person to use it.. It corresponds to the model of software engineering which is: write your specification (write the description of the task); write your program (design the interface); prove the program meets the specification (train the user to use the interface). In fact, it is not only old-fashioned it is clearly impractical in that we essentially perform the same (hard) task twice when we write the program and do the proof.

A more sensible SE model is: write the specification and using known correct rules, derive the program from the specification. This leads to another plan: given a description of the current interaction and using known correct rules, derive an interface from the description.

We should now look in more detail at the old-fashioned approach, to bring out why it is not the recommended way of proceeding, but space does not permit.

The second possibility that we gave for getting someone to understand an interface is less unfeasible than the first. Remember that we are assuming that we start with a description of a user's model for an interaction and we are to derive a specification for an interface from this model. If M is the description of the user's model and S is a description of the specification then we want to have, for any statement f about the model,

$$M \models f \text{ if and only if } S \models s(f) \qquad (1)$$

where s translates statements about the model into statements about the specification. (1) says, roughly, that the things that are true of the model are exactly the things that are true of the specification. This is clearly hard to achieve. However, current research in SE (where we can derive guaranteed correct programs from specifications) suggests ways of achieving this, though only in the sense of suggesting a new research topic. What we seek to do is to derive the specification from the model, using rules which are guaranteed to be correct. What we end up with is a specification which is guaranteed to have property (1).

More generally, work towards deciding which of the different tasks that arise in developing a system from a description of human tasks and then providing formal support for those tasks is, in perhaps rather a small amount at present, going on. The aim is to express as much as possible of the relations between the various components that go to make up a task so that, firstly, they may be checked to see whether they are consistent amongst themselves in their description of the tasks and, secondly, to provide a formal specification for use at the next level down, i.e. as we move towards a more and more algorithmic description of the task. Currently all this is being done using ideas from modal logic, specifically dynamic or action logics.

3. Conclusions

We have tried to suggest that formal methods should be seriously considered for use in HCI. The reasons we have given vary from pragmatic ones, because we believe they can make some tasks easier, to general, methodological ones, because people in the past have been in the same position in other areas and fallen into the traps of lack of rigour or unfamiliarity of other work which has bearing on their own endeavours.

Also, by using techniques that have prove useful in other areas in the past, HCI can profit from the fruits of research by using the programming support that these other areas have started to develop, without having to go through the long process of discovering them all over again and then developing them to the same pitch as they have now reached.

Finally, we would like to stress that since, ultimately, HCI has as a product programs, and programs are just collections of rules in some formal system, it is unlikely that HCI will flourish and deliver its promises if it does not, in a principled way, use formal methods.

Authorized licensed use limited to: Universidade Estadual do Norte Fluminense. Downloaded on May 01,2022 at 11:45:34 UTC from IEEE Xplore. Restrictions apply.