Piotr Witkowski: *Complexity of some logics extended with monadic Datalog programs* Examiner's report

Ian Pratt-Hartmann
University of Manchester

March 29, 2015

1 Summary

We are all familiar with the idea that computer programs occasionally contain errors. But might it be possible, under certain circumstances, automatically to establish that a given program contains no errors—or at least no errors of certain specific kinds? One approach to this problem employs techniques in mathematical logic: by describing the program in question, along with the data-structures it operates on, in some formal language, we aim to prove that certain undesirable program states or data configurations are never reached. If, in addition, the search for such proofs can be automated, we have solved our problem. Unfortunately, the best-understood and most widely-used formal language—namely, first-order logic—lacks the expressive power to describe many of the data-types most commonly encountered in programming. Mr. Witkowski's thesis develops a new method to combat this problem, based on extending the expressive power of some important fragments of first-order logic. Crucially, the presented extension preserves the applicability of certain algorithmic theorem-proving techniques. Thus, the aim of this thesis is to provide the theoretical foundations for future systems able to verify the correctness of certain computer programs automatically.

More technically, the thesis examines extensions of the two-variable fragment of first-order logic with counting, $\mathcal{C}^2$, by means of *Datalog programs*, i.e., collections of function-free Horn clauses whose predicates are subject to some minimality assumption. It is well-known that such logics allow us to specify familiar data-types—such as lists and trees—not definable in first-order logic alone. Here, variables are taken to range over objects in some (finite) program heap, unary predicates are taken to denote data-types, and binary predicates are taken to denote fields of those data-types. Various such logics are defined in Chapter 2, and examples are provided to illustrate their intended use. The main technical challenge which occupies the subsequent development in Chapter 3 is

IEP-M

to establish a framework for specifying conditions under which the search for proofs in $\mathcal{C}^2$ together with Datalog can be carried out algorithmically.

The point of departure in this analysis is an apparently simple observation concerning the special case where the heap satisfies the so-called *non-sharing restriction*—no heap objects can be the target of more than one pointer. Under the (unproblematic) assumption that the heap is finite, it turns out that the minimality assumptions enshrined in Datalog can, in this case, be enforced using $\mathcal{C}^2$ alone. This observation forms the essential content of Theorem 3.21. Of course, the non-sharing restriction is too severe for application to interesting cases: the entire the first part of this thesis can be seen as an attempt to relax this requirement—in particular to allow limited sharing of heap objects by pointers. There ensues a very complex series of definitions the upshot of which is a logic called $\mathcal{C}^2_{r2}$+Datalog+bsr+bir, which can be reduced to an extension of $\mathcal{C}^2$ (discussed below). This is the content of Lemma 3.25. Chapter 4 explains in detail how it is envisaged that such a logic might be used to specify program properties.

In Chapters 5 and 6, the thesis changes gear. Chapter 5 focuses on the finite satisfiability problem for the above-mentioned extension of $\mathcal{C}^2$, in which two distinguished binary predicates are required to be interpreted as trees of bounded rank. (A tree is a finite, connected, directed graph in which exactly one element has no predecessor and no element has more than one predecessor; the rank of that tree is the maximum number of sucessors of any element.) The author shows—very surprisingly—that the finite satisfiability problem for this logic is decidable. This is the content of Theorem 5.28. The proof proceeds by an ingenious modification of a known algorithm for the finite satisfiability problem for $\mathcal{C}^2$. In effect, the author shows that, while $\mathcal{C}^2$ cannot express the assumption that a given binary predicate is interpreted as a tree, it can express, in a single formula, the additional $\mathcal{C}^2$-expressible consequences of that assumption. Chapter 6 then goes on to consider the possibilities for relaxing some of the restrictions remaining in $\mathcal{C}^2_{r2}$+Datalog+bsr+bir. The most striking of these results is perhaps Theorem 6.11, which shows that a very natural such relaxation leads to a problem at least as hard as the well-known reachability problem for vector addition systems.

2 General evaluation

As indicated in the above summary, this thesis falls naturally into two parts: Chapters 1–4, which deal with logics for verifying programs, and Chapters 5–6, which tackle the underlying logics. Generally, speaking, the second half is of much greater quality and significance than the first. The main result of Chapter 5—namely, Theorem 5.28—came as a considerable surprise to me when I first learned of it, and in my opinion represents something of a breakthrough in the area. Theorem 6.11 is also a significant result, while the other results of Chapter 6, which all concern lower complexity-bounds, are well-executed and quite easy to follow.

I must say I enjoyed the first part of the thesis much less. It is heavy with perplexing definitions and unfriendly notation. Tiny mistakes (inevitable in a work of such complexity) form an almost insuperable barrier to understanding, given that all definitions have to unpicked symbol-by-symbol for the reader to see what is going on. I do not believe that the presentation needs to be as difficult as it is. Indeed, the ratio of ground covered to effort demanded (of the reader) falls below what would be considered normal in an article in this area. In the author's defence it might be added that these chapters do form an essential part of an organic whole—an exploration of the applicability of $\mathcal{C}^2$+Datalog to program verification. Considering the thesis in its entirety, therefore, and giving due consideration to the remarkable content of Chapters 5 and 6 (especially the former), I consider the thesis to meet the standards generally accepted for an award of the degree of PhD. at a leading university.

3 Areas for improvement

1. The material on proof-trees and partial proof-trees is difficult to penetrate. The reader is given very little help understanding the definitions, with the result that any mistakes—either on the part of the author or the reader—are almost impossible to locate and rectify. And mistakes are simply inevitable in a logical construction of this complexity. In particular, the definition of "normalized" tree set on p. 48 is incorrect, and has as a consequence the possibility that a set of trees might be normalized, but its decomposition might not be. My complaint here is not that this minor technical slip cannot be fixed. It can. Rather, it is that it took me weeks of writing out definitions and referring back and forth to find out what was going on at all. Matters are made worse in this regard by the occasional problems with the author's English (see last two points).

2. If Chapter 4 was intended as an advertisement for using the author's logic to verify the correctness of programs, it had the very opposite effect on me. I think it is easier to see that the programs are okay than to understand the formal specifications.

3. I thought Chapter 5 was presented in a rather user-unfriendly way (even though the mathematical content is very good). I see no point in separating the proofs from the statements of the various lemmas. The worst aspect of this from the reader's point of view is exemplified by p. 108, which has subsections entitled "Proof of Lemma 5.23" and "Proof of Lemma 5.24". So the reader has to leaf back through the pages to find what Lemma 5.23 (p. 101) and Lemma 5.24 (p. 102) are. A minor irritation, but I don't see what was gained by separating everything out. (I am perfectly able to skip over a proof if I want.) The same applies to the section entitled "Proof of Lemma 5.25" on p. 110.

4. Some of the definitions are a bit clumsy. Just one example: in Definition

3.15, p. 49, it is clear that every set of partial proof trees has a unique tree-arranging order, which renders the definition (and in particular the use of the English indefinite article) misleading. It would be much clearer to say: "Let Sd be a set of proof-trees. The *tree-arranging order*, $\sqsubset_{Sd}$, *for* Sd is defined as follows: ...". Incidentally, this tree-arranging order is, I believe, not in general transitive, which is for me an odd use of the word "order".

Likewise, some theorems are not quite properly formulated. One example: in Lemma 3.25 on p. 56, the noun-phrase "the $\mathcal{C}^2$ with trees formula" is not really grammatical English.

5. Actually, the English in this thesis is generally pretty good. There are, however, a few small problems.

   (a) Articles (definite/indefinite/none) are frequently incorrect, which sometimes leads to genuine confusion. (I realize this aspect of English grammar is hard for Polish speakers.)

   (b) The author doesn't understand object-control constructions. You cannot say "This allows to define a function ..."; you have to say "This allows *us* (or: *one*) to define a function ...". There must be an explicit direct object, to serve as an implicit subject of the infinitival clausal complement. A minor point (and a common error among non-native speakers), but it sounds very awkward.

   (c) I wonder if all of the technical vocabulary is well chosen. For example, a set of trees is "normalized" if it satisfies a certain condition. But *ized* words in English always denote the result of a process. Now, no doubt, the author *could* have defined normalized trees as the result of a process (of normaliztion). But he didn't. Why not just "normal"? Generally, I found all the technical definitions here murderously difficult to get through.

   (d) I think "de Morgan" should be "De Morgan". De Morgan always seemed to write it that way, even when he wasn't starting a sentence.

   (e) Sundry minor points. (I can provide a list.)

4 Impact on the field

Mr. Witkowski has produced a thesis containing significant contributions to the field of Computational Logic. Some of these contributions were originally communicated to the relevant academic community in two refereed conference presentations (and in papers published in the associated conference proceedings).

It is clear that one of these papers (the content of Chapter 5 of the thsis) can be regarded as breakthrough result: it has essentially advanced our

understanding of how logics can be used to describe data-structures of interest to Computer Scientists, and will certainly result in follow-up work by other groups.

5 Conclusion

I conclude that Mr. Witkowski should be awarded the degree of *Doktor*.

Ian Pratt-Hartmann.

18.5.15.