Software Assurance Arguments vs. Formal Mathematical Arguments – A Complementary Role

Ibrahim Habli, Zoë Stephenson, Tim Kelly, John McDermid
Department of Computer Science
University of York
York, United Kingdom
Ibrahim.Habli, Zoe.Stephenson, Tim.Kelly, John.McDermid@cs.york.ac.uk

Abstract—**This paper discusses the complementary role of software assurance arguments and formal mathematical arguments in justifying the achievement of safety and reliability properties within critical applications. This paper reviews the theoretical foundation of this area and proposes a way forward for combining the use of these two forms of arguments in systems and software engineering.**

Keywords – Software assurance arguments, formal logic, software certification

I. Introduction

In critical applications, where software systems have to achieve high reliability targets (e.g. less than 10^{-9} failure rate per flying hour for certain airborne software), software engineers have to produce rigorous evidence that demonstrates the satisfaction of these targets. In certain sectors, the assurance of the reliability and safety of software is justified by compliance with certain process-based certification standards, i.e. by applying a set of techniques and methods that these standards associate with a specific integrity level, typically in the form of a probability of dangerous failures. Recently, there has been a shift towards an alternative approach to certification that requires the substantiation of claims concerning reliability or safety by appealing to an explicit, well-structured and reasoned *argument*. This argument typically forms the core part of a *software assurance case*. Software assurance cases are inspired from the concept of safety cases, the submission of which is a core requirement in many safety certification standards, particularly in the United Kingdom [1] [2]. A safety case is a "*structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given operating environment*" [1]. Although structurally similar to safety cases, assurance cases serve a wider purpose beyond just safety to demonstrate other dependability qualities such as reliability, security and performance.

The argument in a software assurance case explicitly represents the claims, evidence, context and assumptions concerning certain software behaviours. These elements of an assurance argument are connected in such as way that shows how the claims regarding safety, reliability or security are supported by evidence within the assumed context that is defined for the argument. Software assurance arguments are predominantly *inductive*, i.e. offering support for the top-level claim which is *short of certainty*. Due to the subjective nature of a software assurance argument, it typically falls within the scope of informal logic, compared to formal and mathematical logic which is based on mathematical semantics and theory.

There are many applications of formal logic in systems and software engineering that have demonstrated the effectiveness of mathematical approaches in achieving high confidence in the satisfaction of some of the safety and reliability properties. These approaches produce mathematical evidence which substantiates a claim of safety or reliability by exhaustively examining the entire design space or all possible execution paths within a software artefact. The success of formal mathematical approaches in software engineering has been attributed to advances in model-checking and theorem-proving [3]. For example, in theorem-proving, a proof begins with a hypothesis (a theorem, which is typically some system claim) to be shown and an axiomatic basis (or interpretation) on which to base the proof. A typical proof method would combine substitution-based rewriting with strategies such as proof by cases and proof by contradiction. This process is often supported using tools such as automated theorem-provers. In many domains, mathematical evidence generated from model-checking and theorem-proving takes precedence over other forms of evidence generated from testing or simulation [1] [2].

That said, the central matter considered in this paper is the following: *what role does a software assurance argument play in the presence of formal analytical evidence which is based on a mathematical argument? Further, is there a contradiction in that the same standards that require the submission of an explicit assurance argument give precedence to mathematically-based approaches?*

II. Integrated Approach

In this paper, we contend that assurance arguments and mathematical arguments play a complementary role. This complementary role often takes these two forms (*Fig. 1*):

- Mathematical arguments supporting assurance arguments: this is when the mathematical argument forms a core part of the assurance argument and therefore provides one of the strongest forms of evidence that supports certain claims within the assurance argument;

- Assurance arguments supporting mathematical arguments: this is when the assurance argument is used to provide justification concerning the process by which the mathematical argument is constructed.

In some cases, both arguments completely merge. One of these cases is the following:

- Mathematical arguments communicated and explained in the format of assurance arguments.

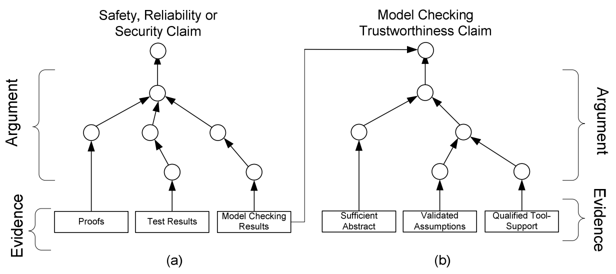

Figure 1. (a) Assurance argument supported by formal evidence (b) Assurance argument justifying trustworthinesss of formal evidence

The discussion regarding combining formal and informal logic is an area of active research within the philosophy of science community. This is summarised rather eloquently by Ian Dove as follows [4]:

"*Mathematicians already, though perhaps tacitly, use the techniques of informal logic. They use them when they appraise proofs, and they use them when they assess mathematical reasoning that isn't proof. This is not to say that mathematicians ought to pay more attention to informal logic or argumentation theory. Rather, this suggests that an accurate philosophy of mathematics ought to recognize this use. Hence, inasmuch as informal logic is already a part of mathematical practice, it makes sense to make the use explicit as part of a larger project to construct a philosophy of mathematics that takes practice seriously.*"

Our objective in this paper is to propose a pragmatic consideration of this problem within the high-integrity software engineering domain. We believe that plenty of research has been done on mathematical arguments, mainly targeting model checking and theorem proving. Also, plenty of work has been done on assurance arguments, particularly in the safety area. However, little has been done on how to combine the two approaches in this domain.

The relationship between the domain expert developing the assurance argument and the mathematical expert generating the mathematical proof can be seen as follows. The domain expert defines the claims that need to be substantiated and the context within which these claims are made. The mathematical expert then formulates the hypothesis regarding these claims and the assumptions that the proof relies on. Then, together, the domain expert and the mathematical expert validate that the proof supports the claims, given the assumptions made during the proof and the context assumed for the claims. Finally, the mathematical expert assures the process by which the proof is generated by communicating and justifying the methods used, the competency of the mathematicians deriving the proof and the acceptability of the assumptions made using the proof.

Over the last four years, we have been supporting our industrial partners by integrating formal mathematical methods within their engineering processes, particularly helping them with embedding mathematical evidence within their assurance arguments. More interestingly, we have been supporting on-going effort addressing the justification of the trustworthiness of mathematical evidence by means of assurance arguments. For example, we have devised a prototype mathematical approach to justifying the absence of run-time exceptions caused by floating-point arithmetic approximation errors. We have shown how mathematical evidence generated from this approach can form part of the assurance argument relating to robustness properties. We have also shown how an assurance argument can be used to justify the soundness of the mathematical approach, potential limitations of the approach, assumptions regarding the completeness of the mathematical rules and also the integrity of the tool implementing this mathematical approach. We have presented this work to representatives of certification authorities in the civil aerospace domain. It was well-received as a potential way to bridge the gap between existing certification practices and current state-of-the-art with regard to use of mathematically-based software engineering techniques.

III. Way Forward

Many standards have started to recognise and accept the role of both assurance arguments and mathematical arguments [1] [2]. This should be complemented by enriching current software engineering literature by the publication of successful patterns on the combined use of these two arguments. This should support industrial practices by providing guidance and worked examples on how the use of mathematical approaches can improve confidence in the safety and reliability of software systems. In particular, we plan to develop an assurance argument catalogue which captures, in a reusable format, how formal mathematical arguments, based on theorem proving and model checking, have been successfully used in practice within an assurance argument approach. This catalogue will be a live artefact, updated regularly, providing practical guidance on how engineers can embed mathematical arguments within an assurance case, particularly for satisfying various certification requirements.

References

[1] UK Ministry of Defence, 00-56 Safety Management Requirements for Defence Systems, Part 1: Requirements, Issue 4, UK MoD, 2007.

[2] UK Civil Aviation Authority, SW01 - Regulatory objective for software safety assurance in air traffic service equipment, CAP 670: air traffic services safety requirements, published by the UK Civil Aviation Authority, 2009.

[3] E. M. Clarke, O. Grumberg, D. A. Peled, Model Checking, The MIT Press, 1999.

[4] I. J. Dove, "Towards a Theory of Mathematical Argument", Foundations of Science, Springer, vol 14, pp 137-152, March, 2009.