Adaptive Safety Arguments and Explanation-Based Learning

Matt Timperley, Maizura Mokhtar, Joe Howe

Adaptive Safety Arguments and Explanation-Based Learning

Matt Timperley, Maizura Mokhtar, and Joe Howe

Centre for Energy and Power Management
University of Central Lancashire
Preston, Lancashire, United Kingdom, PR1 2HE
{mtimperley,mmokhtar,jmhowe}@uclan.ac.uk

Abstract. Software for use in aviation requires certification. This certification is based on a safety argument. These arguments are formed of claims that are linked to evidence about the system. Adaptive systems are a grey area within the current certification guidelines (DO-178 document). Safety cases (sometimes called safety arguments) link claims and evidence in support of an overall safety argument. This paper argues that it is rational to have an adaptive safety argument for an adaptive system. This is illustrated by considering an adaptive controller that uses Explanation-Based Learning (EBL) to generate both control laws and a safety argument, represented using Goal-Structuring Notation. An adaptive safety argument, when coupled with analytical tools, could be used to form the adaptive portion of an otherwise standards-based certification argument. If the rest of the argument holds then the argument should hold for any state where the adaptive safety cases remain valid.

1 Introduction

Certification for aviation software, as part of a platform, can be gained by any sufficient safety argument. A safety argument makes claims about the system that are substantiated by evidence. DO-178 (currently at revision C) is a guideline on how to develop a system that is likely to be certifiable. Addressing the guidelines in DO-178, satisfying criteria for levels of integrity, is a prescriptive approach to obtaining certification. This relates to the software system and is part of a larger safety argument for the whole platform, whose different aspects have different guideline documents.

A technique that has gained some influence in making safety arguments is the safety case. A safety case links claims and evidence into overall arguments. An argument for certification could be presented using safety cases. Adaptive control is an area highlighted as having additional challenges in an entirely specification-based framework. Adaptive control may benefit from having an adaptive safety argument [9].

This argument will be illustrated assuming a controller on board an Unmanned Autonomous System (UAS), which elicits control laws using Explanation-Based Learning (EBL). The structures produced by EBL could be used to form

safety cases that reflect a learning system by adapting alongside the controller. Consider an adaptive controller capable of generating safety cases, in addition to control laws, sufficient to form a safety argument for its own action-response mappings. This could demonstrate both coverage and determinism at the point where the generated safety cases hold i.e. until further changes to the system are made. An explanation of each control law would show, if unconstrained by training information, all rules that are used by the system until further learning occurs. Past arguments could be stored as baselines. Or a separate baseline, which constrains the form of future arguments, could be developed. Before introducing any control law the argument could be updated and validated, possibly using the approach in [2], to give some measure of confidence that the new rule would have no compromising impact on system behaviour. This could automate part of the certification process for an adaptive controller.

This paper proposes the adaptive certification of EBL-based controllers. The paper is divided into 3 sections. Section 2 describes certification and introduces the argument notation being considered alongside the adaptive control method proposed, introduced in Section 3. Section 4 concludes the paper.

2 Aviation Certification

The reason for certification is to have a level of confidence that the software being produced will safely serve its purpose. Certification is awarded by a regulatory body, such as the FAA in the US [8] or CAA in the UK. Once certified, the software can be included in airborne avionics. DO-178 (US), or ED12 in Europe, is a guideline on how to develop a software-based system that is likely to be certifiable. Since these documents are guidelines they are not absolute, arguments may be presented in other ways, though frequently these guidelines are followed [8]. Flight critical software must demonstrate compliance to airworthiness standards. DO-178 is identified as the method of choice for this. Alternatives are permitted but not defined. There are limitations to specification-oriented certification [10], which particularly apply to adaptive systems. One problem that can occur, when certifying adaptive systems, is in interpreting the model that the system has learned [1].

It is worth noting that not all cases are covered by the document, for example UAS specific software and adaptive control. Adaptive flight control software changes its gains, which could be conceived as its situation-action bindings, after deployment. This creates difficulties in adhering to the certification guidelines. The main areas that adaptive systems may require additional work to conform, from [5], based on DO-178B, are: (i) defining software performance requirements, (ii) providing a software verification plan, (iii) defining software requirements and derived requirements, (iv) providing software verification test cases and procedures, (v) providing a Plan for Software Aspects of Certification (PSAC), and providing software life cycle data.

Non-adaptive controllers cannot alter their gains to match evolving situations, such as decay in system responsiveness. Adaptive controllers can alter

gains after deployment automatically, but this comes with the issue that certification guidelines are lacking. Adaptive systems would require adaptation of the component safety cases, to keep the safety argument valid for each new adaptation.

3 Certifying Adaptive Systems using EBL

Goal-Structuring Notation (GSN) has been adopted in the aviation industry for presenting safety cases [6]. GSN is a graphical notation used to present safety arguments in a manner that aims to minimise ambiguity and avoid poorly written safety cases. GSN presents arguments as a hierarchy of nodes representing supporting claims. There are nodes for representing the goals, strategies for achieving goals, solutions (evidence to satisfy arguments), context information and unfinished goals.

Adaptive generation of safety cases would be advantageous, especially in line with adaptations to the controller. A technique that can potentially derive control laws and generate a safety case using GSN is Explanation-Based Learning (EBL). Explanation Based Learning (EBL) was developed as an alternative analytic learning technique that, rather than using many examples to draw a conclusion, draws a conclusion and generalises it from one example [3]. This was posited on the basis that the inclusion of domain knowledge would allow the algorithm to explain an example. This explanation can then be generalised so as to be useful in similar situations. EBL typically links rules using deduction (e.g. Unification) to form an explanation structure. By employing deduction, EBL is potentially of a high enough fidelity for application to safety-critical software. However, the conclusions drawn by EBL are a direct result of the specifics of the domain theory, which are usually hand-coded expert knowledge. The confidence in the control laws elicited by EBL is therefore dependant on the confidence engendered by the knowledge engineering phase, which produces the domain theory.

Adaptations have been developed that concern the generalisations between examples, though each can potentially gain useful knowledge from a single example [4]. EBL can be considered as a search through the space of all possible explanations. In order to make this search tractable a bias is used. The domain knowledge encoded provides a bias by restricting the explanations that fit within the model defined by the domain knowledge. This search is further restricted by a training example to give a specific explanation. This explanation can be generalised and chunked into a new rule. Chunking is often done by computing the most general preconditions i.e. flattening the explanation structure and taking the lowest nodes which are domain knowledge then using these in concert to infer the goal. This, if all the nodes used are operational, gives the new rule. Operationality is expressed with an operationality criterion. Operational, in terms of EBL, has has several posited definitions and in many cases relates in some way to computational efficiency [3], [11]. This can be considered as a filter for rules whose inclusion improves the performance of the controller. The final com-

ponent in EBL is the goal. The algorithm attempts to deductively explain why the training instance is a positive example of the goal concept.

EBL has been applied within the aerospace domain [7]. Using EBL to generate control laws is an adaptive technique as explanations form new rules that can change the action of the controller for a given situation. This is analogue to changing the gains of the controller. EBL explanations help to justify any situation-action bindings and thereby support the determinism property. In order to show coverage a series of safety cases could be produced using the GSN notation as there is some parallel between GSN and a typical way of representing explanation structures (Table 1).

Table 1. Parallels in representation between EBL and GSN.

EBL	GSN
Hierarchical structure based on goal decomposition.	Hierarchical structure based on goal decomposition.
Goal Node.	Goal Node.
Unification with observation.	Solution Node.
Unification with domain knowledge.	(Sub)Goal Node / Strategy Node.
Preconditions.	Context Node.
Conceptual gaps in knowledge	Undeveloped Goal.

3.1 Parallel Representations

EBL and GSN are both hierarchical decompositions headed by a goal. Both structure (EBL) and argument (GNS), via a series of interconnecting nodes, and may eventually terminate in branches where a claim is substantiated. The goals and claims need to be considered analogous in order for the comparison to hold. Context nodes, which show when a node applies, could be considered similar to preconditions. These preconditions could be rule meta-data, preconditions on operational actions or even be embedded in the domain knowledge. Solution nodes would be the analogue of unifications to training data terms in EBL. Undeveloped goals needs no analogue though could represent dearths in domain knowledge. If expanded upon, the undeveloped goals can be used to drive further knowledge engineering efforts and improve the controller.

Since there is a parallel in representation it is possible to use GSN to represent an EBL explanation. Presenting explanations in this form could form a (partial) safety case, depending upon the system. A system incorporating EBL could be used as an aid to safety argumentation, possibly by generating skeleton arguments. This would be particularly useful when EBL is the basis for control laws that are automatically generated. This clear and human-readable

representation aid interpretation of the internal model to find and correct faulty assumptions or rules. In addition, being able to generate safety cases, or update them, for an adaptive technique is a good way to illustrate both the determinism and coverage of the system at any time. Choosing appropriate goals and domain representation directly impacts the value of the generated safety case.

4 Conclusions

An argument for adaptive safety cases forming part or all of a safety argument has been presented alongside consideration of EBL within the aviation industry. The ability for a system to explain its actions in relation to safety concerns could be useful in the context of aviation certification. EBL could be used as an aid towards generating safety arguments, especially where the conclusions that require justification were reached by EBL initially. This representation could also be useful in identifying faulty logic, thus aiding interpretation of the internal model. Additionally, these safety arguments can be used to demonstrate coverage and determinism, both key properties of safety arguments. Further work will involve applying this approach to a case study.

Acknowledgments This work is funded by EPSRC CASE award EP/I501312/1, and is supported by the BAE Systems, UK.

References

1. Alexander, R., Hall-May, M., Kelly, T.: Certification of autonomous systems. In: Proceedings of the 2nd Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre (DTC) Annual Technical Conference (2007)
2. Denney, E., Pai, G., Habli, I.: Perspectives on software safety case development for unmanned aircraft (2012)
3. Ellman, T.: Explanation-based learning: a survey of programs and perspectives. ACM Comput. Surv. 21(2), 163–221 (Jun 1989)
4. Flann, N.S., Dietterich, T.G.: A Study of Explanation-Based Methods for Inductive Learning. Machine Learning 4(2), 187–226 (Nov 1989)
5. Jacklin, S.A.: Closing the Certification Gaps in Adaptive Flight Control Software (2008)
6. Kelly, T., Weaver, R.: The Goal Structuring Notation A Safety Argument Notation. Elements (2004)
7. Levi, K.R., Perschbacher, D.L., Hoffman, M.A., Miller, C.A., Druhan, B.B., Shalin, V.L.: An explanation-based-learning approach to knowledge compilation: a Pilot's Associate application. IEEE Expert 7(3), 44–51 (Jun 1992)
8. Romanski, G., Inc, V.: The Challenges of Software Certification (2001)
9. Rushby, J.: Runtime Certification. Lecture Notes in Computer Science, vol. 5289, pp. 21–35. Springer Berlin / Heidelberg (2008)
10. Rushby, J.: New challenges in certification for aircraft software. In: Proceedings of the ninth ACM international conference on Embedded software. pp. 211–218. EMSOFT '11, ACM, New York, NY, USA (2011)
11. Steven, M.: Quantitative results concerning the utility of explanation-based learning. Artificial Intelligence 42(23), 363–391 (Mar 1990)