Formal Assurance Arguments: A Solution In Search of a Problem?

Patrick John Graydon
Mälardalen University, Västerås, Sweden
(The author is now employed by NASA Langley Research Center)
Email: patrick.j.graydon@nasa.gov

***Abstract*—An assurance case comprises evidence and argument showing how that evidence supports assurance claims (e.g., about safety or security). It is unsurprising that some computer scientists have proposed formalising assurance arguments: most associate formality with rigour. But while engineers can sometimes prove that source code refines a formal specification, it is not clear that formalisation will improve assurance arguments or that this benefit is worth its cost. For example, formalisation might reduce the benefits of argumentation by limiting the audience to people who can read formal logic. In this paper, we present (1) a systematic survey of the literature surrounding formal assurance arguments, (2) an analysis of errors that formalism can help to eliminate, (3) a discussion of existing evidence, and (4) suggestions for experimental work to definitively answer the question.**

***Keywords*-safety case, security case, assurance argument, formal argumentation**

I. Introduction

A *safety case* is a 'compelling, comprehensible, and valid' case that a system is adequately safe for use in a given application [1]. Safety cases have received academic attention and are mandatory in some domains [1]–[3]. Similar cases have been used for properties such as security or dependability more broadly [4], [5]. All *assurance cases* comprise both evidence and a structured *assurance argument* explaining how that evidence supports an assurance claim such as 'the system is adequately safe to use in its intended operating context'. Some authors have proposed formalising all or part of safety, security, or similar arguments [6]–[25]. Because most computer science researchers associate formality with greater rigour, it is not surprising that they would attempt to formalise assurance arguments. But it is not clear whether such formalisation brings benefits that outweigh its costs.

Formal methods have been used to show that source code refines a formal specification (e.g. [26]). But source code is a formal specification for software; proving that one formal specification refines another is different from proving a claim about a system in the real world. Moreover, demonstrating the validity of a safety rationale is not the (only) purpose of an assurance argument: assurance arguments are meant as a tool for managing safety through the life of a system [2], [27]. This requires communicating the safety rationale to managers considering operational changes, engineers making changes to the system, and safety professionals tracking system performance trends, amongst others.

Formalisation is not without costs such as limiting the reading audience to those with the skill needed to read formal logic and making the argument structures less comprehensible. Whether formalisation of assurance arguments brings benefits that outweigh the imposed costs is an open but largely unacknowledged research question.

In this paper, we make four contributions: (1) a systematic literature survey of proposals for formal assurance arguments and related research, (2) an analysis of the argument errors that formal analysis can and can't find, (3) a discussion of the existing evidence regarding the problem to be addressed by formalisation, and (4) suggestions for experimental work to definitively answer the question.

In section II, we provide background on assurance cases, existing argument notations, and the meaning of 'formal'. In section III, we present a literature survey on formal assurance arguments. In section IV, we discuss fallacies and the kinds of checks that formalisation might facilitate. In section V, we summarise what is known about the costs and benefits of formalisation. In section VI, we sketch research activities for determining whether formalisation is worthwhile. Finally, we conclude in section VII.

II. Background

In this section, we recount the purposes of assurance cases, survey existing argument notations, and clarify what it might mean for an assurance argument to be 'formal'.

A. The Purposes of Assurance Cases

Any discussion of proposed argument techniques must be informed by the purposes of those arguments, which are discussed in standards and in the academic literature [1], [3], [27], [28]. Because safety cases are used in many regulatory contexts, engineering disciplines, and domains, any statement of their purpose will be a generalisation. The following might be incomplete, but it will serve as a basis for identifying some functions that safety arguments serve.

UK military contractors must 'develop, maintain, and refine the Safety Case through the life of the contract' [1]. The safety case must be 'sufficient to demonstrate that the system is safe, so far as is reasonably practicable', 'address the full life of the system', and contain evidence that is 'commensurate with the potential risk posed by the system, the complexity of the system, and the unfamiliarity of the

circumstances involved'. This includes 'relevant data from the use of the system' to validate or refute the safety rationale. The safety argument must also record 'key decisions made by the safety committee' for future reference.

Clearly, a safety argument must organise the information that is needed to judge safety claims. But standards and common sense *also* require maintenance of safety throughout the system lifecycle, including planned and unplanned changes. Playing its full role across the safety lifecycle will require a safety argument to communicate many things, including:

- The system-specific operational definition of 'adequately safe' (or 'unacceptable risk', etc.) to be used
- How the system will manage risk (e.g., safety concept)
- How the designers assume the system will be used
- Which claims developers think the evidence supports
- Which safety considerations the developers (and operators) think are most important

These details must be conveyed to many people, including:

- Engineers creating or refining the initial design
- Stakeholders judging how safe a system is or will be
- Certifiers and safety assessors
- Operators changing operating procedures
- Safety engineers monitoring safety in the field
- Developers making changes to existing systems

B. Assurance Argument Notations and 'Formality'

Assurance arguments have been presented in a number of forms. Many have been written in prose (e.g. [29]). Others have been presented in tabular notations [2]. Many papers discussing safety arguments use one of two graphical notations: the Goal Structuring Notation (GSN) [2], [30] and the Claims Argument Evidence (CAE) notation [31]. Opinions about which notation is the 'best' vary: while many authors favour GSN or CAE, others favour prose text [32].

Prose, tables, GSN, and CAE are means of recording 'informal' arguments. When discussing 'formal' assurance arguments, it is important to be clear that 'formality' covers three distinct issues: (1) specification of syntax, (2) symbols versus natural-language, and (3) deduction versus induction.

1) Specification of syntax: An argument language might specify the argument's syntax without providing a means to compute the truth of the conclusion. For example, a graphical argument notation might specify which kinds of elements can be connected to which other kinds of elements, yet not constrain the text within those elements (beyond simple part-of-speech restrictions). These syntax rules might be expressed in natural language or in a formal notation such as Backus-Naur Form (BNF). Both the GSN and CAE notation have syntax rules expressed in prose text [30], [31]. For example, in GSN, goals can directly support other goals but solutions cannot be in the context of an away goal [30].

2) Symbols versus natural language: An argument's claims and premises might be expressed as either (a) natural-language text or (b) symbols connected by operators. For example, an arguer using natural-language might claim that 'the thrust reversers are inhibited when the aircraft is not on the ground'. In a symbolic language, an arguer might claim that '$\neg\mathrm{on\_grnd} \Rightarrow \neg\mathrm{threv\_en}$' (after first defining those symbols using natural language). Both GSN and CAE use natural-language text in graphical argument elements.

3) Deduction versus induction: Regardless of whether it is expressed in text or symbols, an argument might be either deductive or inductive. (Some writers have pointed out that these terms can themselves obscure other relevant distinctions between forms of argument [33]. But for the purposes of this paper, those common terms will serve.)

In deductive arguments, the truth of the premises *entails* the truth of the conclusion. For example, consider the famous syllogism about Socrates' mortality: (P1) all men are mortal and (P2) Socrates is a man, therefore (C) Socrates is mortal. This argument is deductive even though expressed in natural language: if (P1) and (P2) are true, (C) *must* be true.

In inductive arguments, the truth of the premises means that the conclusion is *likely* but might be shown to be false by new evidence. For example, most safety arguments contain logic along the lines of the following: (P1) hazard identification was done adequately and (P2) all identified hazards are acceptably managed, therefore (C) all hazards are acceptably managed. An accident involving an unidentified hazard would refute (C). We could make this argument deductive by changing (P1) to 'all hazards have been identified', but this would merely relocate the unavoidable induction. One difficulty with inductive logic is in specifying and determining how likely claims are or must be. Researchers have proposed many mechanisms for doing this, none of which is known to be adequate in all cases [34].

Discussions of assurance argument semantics sometimes reference Toulmin's model of inductive argumentation [2], [33]. Definitions of GSN reference Toulmin but stop short of defining GSN in terms of Toulmin's model [2], [30]. (There is some ambiguity in how GSN is defined and used [35].)

III. A Survey of Proposed Argument Formalisms

In this section, we present the method and results of our survey of proposals to formalise assurance arguments.

A. Research Questions

The objective of the survey was to identify proposed formalisms and characterise what is known and claimed about each. More specifically, for each paper:

1) What uses of the argument are discussed? More specifically, what artefact (or part or aspect of one) is formalised? How is the formalised artefact to be used? What sort of formalism is to be used? (For example, which logical notation is used?)
2) Does the formalism replace or augment an informal argument written in natural-language text?

3) Does the formalisation constrain or affect the argument structure? If so, how?
4) What benefits of formalisation are noted? More specifically, what potential benefits are mentioned? Which benefits are specifically claimed? What evidence of benefit is given or cited?
5) What potential drawbacks are mentioned?

B. Search Strategy

We knew that formalisation had been proposed in both the safety and security domains. In order to capture both kinds of proposals, we used two search terms: 'formal safety argument' and 'formal security argument'. Relevant proposals might have appeared at safety-, security-, or dependability-themed conference and workshops. To capture a broad selection of these, we searched four digital libraries: (1) IEEE Xplore, (2) ACM Digital Library (publications from ACM and affiliated organisations only), (3) Springer Link, and (4) Google Scholar (case law and patents excluded). The first three of these are the libraries of the most relevant publishers. We added the fourth to include papers published by smaller, special-purpose publishers. We considered only English-language results but did not constrain our search to specific publications, authors, or publication dates. Where electronic searches returned many results – Springer Link returned 40,283 results for the query 'formal security argument' – we restricted our attention to the first sixty.

C. Selection Strategy

We selected papers from the search results in two phases. In the first phase, we examined titles and abstracts and eliminated papers matching any of the following criteria:

- The title and abstract convey no hint that the paper might be about a safety, security, or similar assurance argument or technology related to such an argument
- The paper is about an item of evidence (e.g., proof that an algorithm has specified properties) rather than formalisation of a safety or security argument
- The word 'formal' is used in a sense other than that of formalised syntax or symbolic or deductive logic

In the second phase, we examined the papers' full texts. We eliminated papers matching any of the following criteria:

- The paper is not concerned with a system of documenting support for a safety claim, security claim, or other dependability claim (regardless of whether this system is explicitly named a safety case, assurance case, security case, dependability case, or trust case)
- The paper does not discuss (even in passing) a means of recording the linkage from evidence to dependability claim that uses symbolic or deductive logic

We then read each selected paper and recorded our answers to the questions listed in subsection III-A. Paper selection and characterisation was done by a single researcher.

Table I
NUMBER OF PAPERS SELECTED IN THE FIRST SELECTION PHASE

Digital library	**Safety**	**Security**
IEEE Xplore	12	13
ACM Digital Library	17	7
Springer Link	24	2
Google Scholar	8	1
Unique results (72 total):	**54**	**23**

We might obtain more complete and accurate results by querying more databases, considering more results from each, or including multiple researchers. However, we are not relying upon inclusion or exclusion decisions to make a quantified conclusion about a proposition. Our approach is sufficient to provide a reasonably-complete list of the kinds of argument-formalisation proposals that have been made.

D. Survey Findings

Table I gives the number of papers we selected from each digital library in phase one. Phase two yielded twenty selected papers [6]–[25]. Since many of the papers report on ideas that the same authors had reported in other selected papers, we summarise related papers together below.

The papers and proposals vary dramatically. Some propose formalising syntax, while others propose symbolic notation. Some propose formalising safety arguments, while others propose formalising similar arguments such as security requirements satisfaction arguments. But few make explicit claims about the benefit of formalisation, and none backs up such a claim with substantial evidence.

E. Basir, Denney, Fischer, Pai, and Pohl: Automatically-Generated Arguments

Three of the selected papers detail efforts by Basir, Denney, Fisher, Pai, and Pohl to automatically generate safety arguments from symbolic, deductive proofs [6], [7], [10].

1) Uses of the formalised argument: In all three papers, the authors propose using a theorem prover to create a symbolic, deductive proof and then automatically generate a safety argument from that proof [6], [7], [10]. In the 2009 paper, premises used in the proof, such as

```
hi(attitudeBodyToNav_lv, 0)=A, 3::A)
&hi(attitudeBodyToNav_0, 0)=B, 3::B
&hi(dcmtoQuat_Single2, 0)=C, 3::C)
& ... => ... &has_unit(J, ang_vel))
```

become goals in the generated GSN argument [6]. In the 2010 paper, the generated argument contains more natural language text surrounding the formal predicates [7]. For example, one goal reads, 'Formal proof that $Quat_4$::quat(NED, Body) holds for Fc.cpp'. (This goal, like others in the paper, is not a proposition as GSN requires [2].) In the 2012 paper, goals are also formal expressions within text frames [10].

The proposed scope of the generated arguments decreases as the authors refine their work. In 2009, the authors

'believe[d] that the result of [their] research [would] be a comprehensive safety case (i.e., for the program being certified, as well as the safety logic and the certification system) that [would] clearly communicate the safety claims, key safety requirements, and evidence required to trust the software' [6]. By 2012, the formalism is limited to proof that source code refines a formal specification: 'We auto-generate safety argument fragments for the software components of the example system from formal proofs of correctness. Then, we automatically assemble these with the manually created safety arguments, obtained from traditional safety analysis, which are applicable to the wider system context'.

2) Relationship to informal argument: Basir et al. suggest automatically generating an argument from a proof [6], [7], [10]. The generated argument presents the logic of the proof to the reader, ostensibly in an easier-to-understand form.

3) Effect on argument structure: In the proposed scheme, the structure of the generated argument will follow that of the proof from which it is generated [6], [7], [10]. Since automatically-generated proofs can be obscure, the authors 'concentrate on natural deduction style proofs, which are closer to human reasoning than resolution proofs' [6].

4) Noted benefits and drawbacks: In the 2009 and 2010 papers, Basir et al. claim that the generated argument makes proofs more readable and gives the information needed to trust the proof evidence [6], [7]. However, they claim no benefit that an argument derived from a proof might have over a hand-generated, informal argument. Moreover, they note that 'the straightforward conversion of ... proofs into safety cases is far from satisfactory as they typically contain too many details' and suggest future work on abstraction [6].

In the 2012 paper, Denney et al. claim to have shown that automatic generation of argument from proof is feasible [10]. In passing, they also imply that automatic generation is needed to make safety-case-based certification practical, asserting that writing arguments 'quickly becomes unmanageable during iterative systems and software development'. Although development has not halted everywhere safety cases have been adopted, the authors neither elaborate nor offer evidence for this claim. Moreover, since the usual alternative is to cite formal proof evidence as a solution [2], the magnitude of any reduction in effort is unclear.

F. Bishop and Bloomfield: Deterministic Arguments

Bishop and Bloomfield raise the possibility of machine-checked, deductive safety arguments in passing [8].

1) Uses of the formalised argument: The authors mention that *part* of a safety argument 'could be ... *deterministic*, where evidence can be axioms, the inference mechanism is the rules of predicate logic, and the safety argument is a proof using those rules' [8]. They provide no example but reference Gentzen: 'There is much work on structuring and representation of arguments in mathematical logic [36]'. In later sections, the authors write that a deterministic argument should support 'a claim or sub-claim by showing that, given some assumptions and a model of the real world, certain hazardous behaviours are "incredible"'. Such arguments 'would normally require a formal model of the system and a proof that the system is safe with respect to its safety requirements .... The supporting evidence could include ... explicit validation of the model assumptions [and] an independent check of the formal argument'.

2) Relationship to informal argument: Bishop and Bloomfield seem to suggest that symbolic, deductive logic could replace part of a textual safety argument [8].

3) Effect on argument structure: It is not clear what effect the proposal from Bishop and Bloomfield (again, made briefly and in passing) would have on argument structure [8].

4) Noted benefits and drawbacks: None.

G. Brunel and Cazin: Arguments in LTL

Brunel and Cazin propose formalising a requirements engineering goal structure into symbolic, deductive logic so that it can be mechanically verified [9].

1) Uses of the formalised argument: Brunel and Cazin propose a formal argument semantics 'that allows automatic validation of the argumentation' and 'a pragmatic approach based on this framework to easily edit and validate a safety argumentation [sic]' [9]. Claims are expressed in a symbolic, deductive language, namely linear temporal logic (LTL). For example, the claim that the 'Detect and Avoid function is correct' is formalised as '$G\left(d_{\text{obstacle}} < d_{\min}\right) \rightarrow \left(\left(d_{\text{obstacle}} \neq 0\right) \cup \left(\left(d_{\text{obstacle}} > d_{\min}\right)\right)\right)$'.

2) Relationship to informal argument: Brunel and Cazin seem to propose first developing a KAOS goal structure [37] and then deriving the formalised argument from this [9].

3) Effect on argument structure: In the proposed scheme, the formal argument's structure reflects that of the KAOS goal structure from which it is derived [9].

4) Noted benefits and drawbacks: Brunel and Cazin claim to have addressed the 'objective of developing a clear, convincing, formal, and verifiable argumentation [sic]' by developing 'a formal semantics, based on LTL, ... [that facilitates] tackl[ing] the problems of validity and completion' [9]. They offer no evidence that the formal argument is clear, that the entire argument can be formalised, or that informal arguments are frequently invalid or incomplete. But they do note that ' "pretty" presentation of an argumentation should be also a major concern as we must keep in mind that the ultimate objective is to convince, not a specialist of temporal logic, but a certification authority instead'.

H. Denney, Naylor, and Pai: Annotated Informal Arguments

Denney, Naylor, and Pai propose that developers 'semantically enrich ... GSN nodes' to enable developers to express structured queries about an argument's contents [13].

1) Uses of the formalised argument: Denney et al. propose that 'in addition to the descriptive text' that arguments comprise, developers should 'associate nodes with metadata' [13]. This metadata has the form 'attribute ::= attributeName param*' where 'param ::= String | Int | Nat | ... userDefinedEnum'. The paper gives 'element ::= aileron | elevator | flaps ...' as an example enumeration.

2) Relationship to informal argument: The authors propose adding formal annotations to informal arguments [13].

3) Effect on argument structure: Annotating an informal argument shouldn't affect its structure.

4) Noted benefits and drawbacks: Denney et al. claim that formalisation enables rich querying [13]. For example, they claim to be able to 'generate a view ... of traceability to only those hazards whose likelihood of occurrence is remote, and whose severity is catastrophic'. The authors support this claim with an example query. The only potential drawback they mention is the cost of creating the necessary ontologies, which they plan to address with tool support. The paper neither makes nor supports the claim that the benefits of rich querying over simple text search outweigh the costs of developing the ontology and annotating the argument.

I. Denney, Pai, and Whiteside: Formally-Specified Syntax

In two papers, Denney, Pai, and Whiteside propose formalising the syntax of GSN so as to ease pattern instantiation and enable the construction of 'hicases' that allow readers to collapse or expand parts of arguments on screen [11], [12].

1) Uses of the formalised argument: In both papers, the authors propose the same formalism of GSN syntax [11]:

> Let $\{s, g, e, a, j, c\}$ be the node types strategy, goal, evidence, assumption, justification, and context respectively. A partial safety case argument structure S is a tuple $\langle N, l, t, \rightarrow \rangle$, comprising the set of nodes, N, the labeling function l ... that gives the node type, t ... giving the node contents ... and the connector relation, $\rightarrow$ ....

This formalism allows them to define syntax rules. For example, both papers give a formalisation of the rule that 'goals cannot connect to other goals' [11]: $(n \rightarrow m) \wedge [l(n) = g] \Rightarrow l(m) \in \{s, e, a, j, c\}$. (Note: GSN explicitly *allows* goals to support other goals [30].)

In their hicases paper, Denney et al. show formalised content in the example arguments [12]. For example, one goal reads, 'output->m_aileron_m1p1 has property desired(aileron) (i.e., has_unit(output->m_aileron_m1p1, desired(aileron)) holds.)'. This formalisation of the example argument's content appears to the result of the choice of example rather than a consequence of the paper's proposal.

2) Relationship to informal argument: In both papers, the authors propose formalising the argument syntax; the GSN elements might still contain natural-language text [11], [12].

3) Effect on argument structure: Except for errors in their formalisation of GSN, the syntax of any GSN argument could be formalised as the authors propose [11], [12].

4) Noted benefits and drawbacks: In their patterns paper, Denney and Pai claim that formal syntax enables 'automated instantiation, composition, and transformation-based manipulation' [11]. (This claim is perhaps misleading: humans must still supply the *content* for the instantiated pattern, whether in a table or other form.) The authors claim that

> the main benefit of our work ..., we anticipate, is a reduction in the effort involved in safety case creation/management ... together with improved assurance. Specifically, given the assurance afforded by automated instantiation that a pattern instance is well-formed and meets its specification, ... safety engineers ... and certification/qualification authorities ... can divert efforts to domain-specific issues, e.g., selecting the appropriate patterns for assurance, evaluating a smaller, abstract argument structure for fallacies/deficits instead of its larger concrete instantiation, determining the evidence required to support the claims made, etc.

Nevertheless, the authors offer no direct evidence of reduced effort and no evidence that developers make frequent or problematic syntax errors when instantiating GSN patterns.

In the paper on hicases, Denney et al. make no specific claims about the benefits of formalising syntax save that it enabled the creation of their display and editing tools [12].

J. Forder: A Safety Argument Manager

Forder describes an early argument editing tool [14]. In passing, he suggests the possibility of 'formal statements' in arguments. We include this paper despite the brevity of its suggestion because it indicates that interest in formalisation dates from the early days of safety argumentation research.

1) Uses of the formalised argument: Describing the internal structure of safety argument software, Forder writes that 'the use of formal statements in models and arguments will allow automatic detection of inconsistencies in models, in arguments, and between arguments and models' [14].

2) Relationship to informal argument: No clear relationship between formal and informal argument is specified [14].

3) Effect on argument structure: The paper does not clarify the effect of formality on the argument structure [14].

4) Noted benefits and drawbacks: This paper does not discuss any benefits or drawbacks of formalisation beyond the allusion to automatic checking above [14].

K. Franqueira, Haley, Laney, Moffet, Nuseibeh, Tadeschi, Tun, and Yu: Security Requirements Satisfaction Arguments

In selected papers from 2006 and 2008, Haley et al. suggest 'security requirements satisfaction arguments' split into formal and informal parts [15], [16]. Tun et al. extend these ideas in a 2010 paper [24] and Yu et al. present a

related tool in 2011 [25]. While these security requirements satisfaction arguments are not security assurance arguments, we include them because they are broadly similar.

1) Uses of the formalised argument: In their 2006 paper, Haley et al. suggested splitting security requirements satisfaction arguments into *outer* and *inner* arguments [15]:

> The [outer] part of the argument consists of a formal argument to prove that a system can satisfy its security requirements, drawing upon claims about the behavior and properties of domains in a system. The claims about behavior of the domains are trust assumptions .... The [inner] part of the argument consists of structured informal arguments to support the trust assumptions ... made in the formal argument.

Examples in the 2008 paper make the nature of both parts clear. The outer argument is given in symbolic, deductive logic. The authors provide the following example [16]:

1	$I \rightarrow V$	(Premise)
2	$C \rightarrow H$	(Premise)
3	$Y \rightarrow V \& C$	(Premise)
4	$D \rightarrow Y$	(Premise)
	5 D	(Premise)
	6 Y	(Detach ($\rightarrow$ elimination), 4, 5)
	7 $V \& C$	(Detach, 3, 6)
	8 V	(Split ('&' elimination), 7)
	9 C	(Split, 7)
	10 H	(Detach, 2, 9)
11	$D \rightarrow H$	(Conclusion, 5)

The inner argument justifies the premises assumed by the outer argument. Inner arguments are given in extended Toulmin notation. The authors give the following example [16]:

```
given grounds G2: "Valid credentials are
  given only to HR members"
warranted by (
 given grounds G3: "Credentials are given
   in person"
 warranted by G4: "Credential administrators
   are honest and reliable"
 thus claim C1: "Credential administration
   is correct")
thus claim P2: "HR credentials provided -->
  HR member"
rebutted by R1: "HR member is dishonest", ...
```

Security requirements satisfaction arguments are not security assurance requirements: they are meant to help requirements engineers to develop and explore security requirements, not as a means to collect and explain all of the evidence that shows that a given system is acceptably secure.

In their 2010 paper, Tun et al. present more examples, but the nature and purpose of the proposed formalism remains essentially the same [24]. In their short 2011 paper, Yu et al. describe their tool for creating these arguments [25].

2) Relationship to informal argument: Following the proposal of Haley et al., developers create formal outer arguments in lieu of informal argumentation [15], [16]. These ideas are used in later work [24], [25].

3) Effect on argument structure: The formal outer arguments have a different structure than informal arguments (such as the inner arguments) [15], [16].

4) Noted benefits and drawbacks: In the 2006 paper, the authors introduce their requirements engineering 'framework' but make no claims about its properties (e.g., feasibility, relative ability to reveal subtle security issues, etc.) [15]. The authors make no specific reference to the benefits or drawbacks of the formalism they propose.

In the 2008 paper, the same authors present the same framework in greater detail [16]. They conclude, in part, that the 'satisfaction argument facilitate[s] showing that a system can meet its security requirements. ... By first requiring the construction of the formal argument based on domain properties, one discovers which domain properties are critical for security'. The authors note that 'the more rigorous the process used to establish [requirements] satisfaction, the more confidence one can have that the system will be secure. The strongest process is a proof. A weaker alternative to a proof is an [informal] argument'. The authors note no drawbacks of formality, except to point out the relative benefits and advantages of specific formalisms. They theorise that 'use of a more fine-grained logic in the outer argument may lead to fewer trust assumptions in the inner argument. On the other hand, more expressive logics come at the expense of tractability of reasoning and of potential decidability problems'. The authors note that some industrial partners 'did not see the utility of [formal] outer arguments and wanted to proceed directly to the inner arguments'. They respond, 'it is the outer argument that provides the assumptions to be tested in the inner arguments' but do not explain why these must be formal.

In their 2010 paper, Tun et al. make no specific claims about the benefits or drawbacks of argument formality [24].

In their 2011 paper, Yu et al. claim that 'the use of informal and formal arguments is helpful to domain experts' [25]. They do not elaborate, and cite for support a 'case study' described without the detail required to assess it, e.g., what data they gathered and how these support the claim [38].

L. Matsuno and Taguchi: Formalised GSN (Patterns)

Matsuno and Taguchi propose formalising GSN patterns [17], [18]. They define both a formal syntax and a means of replacing placeholder text during instantiation.

1) Uses of the formalised argument: In his 2011 paper, Matsuno presents 'a proposal towards [formalised] parameterisation of patterns in GSN' [17]. He gives a formal syntax for the pattern's structure and a formal mechanism for replacing placeholder text. Annotations record instantiation values. For example, '$[2/x, \epsilon/y, \text{“hello”}/z]$ represents that x and z are instantiated with 2 and "hello", respectively, whereas y is not instantiated'. Parameters might be integers, strings, or user-defined sets. The authors discuss placing

further limits on parameter values, giving the example of restricting a claimed CPU utilisation to the range 0–100%.

In their 2014 paper, Matsuno and Taguchi claim to 'give a formal definition and the semantics for GSN and its extensions' [18]. Despite this claim, the paper does not define what an argument *means* (i.e., its 'semantics'). The authors present essentially the same formal syntax and pattern mechanism given in the earlier paper.

2) Relationship to informal argument: The proposal formalises the syntax of GSN and gives a mechanism for replacing placeholder text [17], [18]. Non-placeholder content in patterns is given informally as natural language text.

3) Effect on argument structure: The proposed formalism does not seem to encode any more restrictions on argument structure than are given in the GSN standard [17], [18], [30].

4) Noted benefits and drawbacks: In his 2011 paper, Matsuno claims that his parameterised and typed expressions and scoping rules 'provide the safeguard to misuses of patterns ... and the means to automate checking [instantiations'] type consistency' [17]. In their 2014 paper, Mastuno and Taguchi claim that machine checking of formalised patterns 'will help to avoid misuses of parameterized expressions and to detect errors in early stages. ... If a user instantiates [a placeholder reading 'System X'] with e.g., "Railway hazards", then the argument does not make sense. It is fairly obvious that type checking prevents such a misplacement'. Neither paper presents evidence of widespread or problematic misuse of patterns that could be caught by the proposed checks. Neither paper mentions any drawbacks.

M. Rushby: Partial Formalisation Into Proofs

Rushby proposes formalising as much as possible of the safety argument into deductive, symbolic logic that can be checked by a proof checker [19], [20].

1) Uses of the formalised argument: In his 2010 paper, Rushby recognises that some parts of a safety argument cannot be formalised into symbolic, deductive logic [19]:

> Although a safety case is an argument, it will generally contain elements that are not simple logical deductions: some elements of the argument will be probabilistic, some will enumerate over a set that is imperfectly known (e.g., "all hazards are adequately handled"), and others will appeal to expert judgement or historical experience.

He proposes formalising the parts of the argument that 'lend themselves to this process' in symbolic logic such as '`good_doc(approp_claim_doc) IMPLIES appropriate(claim, system, context)`'. In his 2013 paper, Rushby endorses the same formalism [20].

2) Relationship to informal argument: In both papers, Rushby proposes encoding the formalisable portions of the argument in symbolic logic that will be manipulated using theorem proving and proof checking tools [19], [20]. In the 2010 paper, he mentions that existing informal techniques might serve as the means of *developing* the argument: 'my aim is not to supplant GSN or other methods for developing and documenting safety cases in a systematic and reader-friendly manner: rather, it is to provide a means for mechanically checking the logical soundness of cases developed through these or any other methods'. While it is not clear whether the formalised argument would replace or be maintained in parallel with its informal source, the remaining informal part establishes the truth of formal premises. For example, after reviewing `approp_claim_doc` and finding it appropriate for some unspecified purpose, 'reviewers can indicate their assent by adding `good_doc(approp_claim_doc)` as an axiom'. It is not clear whether every axiom would *require* support from a structured informal argument.

3) Effect on argument structure: It not clear whether the proposed formal argument should have the structure of an informal argument. In his 2013 paper, Rushby suggests use of tool support that might, in his view, obviate the need for human readability [20]. Specifically, he suggests real-time interaction with proof tools as a means of understanding how premises affect an argument [20]: 'Evaluators ... could actively probe the argument using "what-if" exploration (e.g., temporarily remove or change an assumption and observe how the proof fails, or inspect a counterexample)'.

4) Noted benefits and drawbacks: In his 2010 paper, Rushby suggests that mechanical verification might *replace* human scrutiny of the formalised parts of the argument [19]: 'By formalizing the elements that do lend themselves to this process, we may be able to reduce some of the analysis to mechanized calculation, thereby preserving the precious resource of expert human review for those elements that truly do require it'. However, he notes what whether formal argument verification is worthwhile 'depends on whether unsoundness is a significant hazard to real safety cases'. He suggests that it might be, noting that theorem provers have found subtle flaws in his formal verification proofs. He concludes, 'the most important tasks for the future ... are experiments to determine whether formalization does deliver benefit in the development and assessment of safety cases'.

In his 2013 paper, Rushby proposes that [19]

> evaluation of a safety case argument can – and should – largely be reduced to calculation. ... [My] basic claim is that evaluation of large safety cases will benefit from – indeed, requires – automated assistance. ... Formal verification systems provide tools that can be adapted to represent, analyze, and explore the logic of our case, thereby largely eliminating logic doubt.

Rushby presents no empirical evidence that evaluation of safety arguments either benefits from or requires the assistance of formal tools. Instead, he concludes that 'the diagnosis and proposals in this paper are deliberately speculative and, perhaps, provocative. ... Suggested research directions are simple: try this out and see if it works'.

N. Sokolsky, Lee, and Heimdahl: First-Order Logic

Sokolsky et al. mention in passing that they are 'exploring the use of multi-sorted first-order logic for ... formalization [of safety arguments]' for medical devices [39].

1) Uses of the formalised argument: They note that a formalism that 'does not capture the meaning of the argument, but only its logical structure, might be possible' [39].

2) Relationship to informal argument: The authors do not elaborate on the proposal in this paper [39].

3) Effect on argument structure: While the authors do not elaborate, the use of a symbolic, deductive logic would likely have an effect on the argument's structure.

4) Noted benefits and drawbacks: The authors claim that 'logical fallacies are common in assurance cases,' citing Greenwell et al. [39], [40]. They claim that the proposed formalisation 'will be able to capture logical fallacies, that is, inconsistent arguments that cannot be true regardless of what is being argued'. (However, as we show in subsection V-B, the fallacies that can be detected by formal verification alone are not the sort that Greenwell et al. found.)

O. Tolchinsky, Modgil, Atkinson, McBurney, and Cortés: Decision Support

Tolchinsky et al. propose using non-monotonic logic as an on-line decision-making tool for humans performing safety-critical tasks [23]. Safety arguments are not typically used this way, but have broadly similar scope and content.

1) Uses of the formalised argument: Tolchinsky et al. describe using a non-monotonic logic and associated tools to implement an on-line aid for making safety-critical decisions [23]. Claims are expressed using symbolic predicates (e.g., `treat(r,penicillin)`) and stored in the tool's database. Using dialogue games, the argument is updated with the details relevant to the safety of a proposed action (e.g., transplanting a given organ into a given patient) and used to explore factors that might make that action unsafe.

2) Relationship to informal argument: Tolchinsky et al. do not relate their work to traditional safety arguments [23].

3) Effect on argument structure: Tolchinsky et al. do not relate their work to traditional safety arguments [23].

4) Noted benefits and drawbacks: Tolchinsky et al. discuss some of the limits and challenges of the non-monotonic logic tools in question but not the pros and cons of formalisation as opposed to informal logic presentation [23].

P. Tun, Bandara, Price, Yu, Haley, Omoronyia, and Nuseibeh: Policy checking

Tun et al. 'propos[e] an extended argumentation language for ... selective disclosure requirements' [22].

1) Uses of the formalised argument: Tun et al. propose formalising arguments about privacy into the Event Calculus [41]. The authors give the following example:

```
[time,user,subject,loc]
(HoldsAt(SamePF(user,subject),time) |
HoldsAt(Friends(user,subject),time)) &
(Happens(Tap(user,subject),time) ->
(Happens(Query(subject,loc),time+1) &
Happens(At(subject,loc),time+2))).
```

The formalised arguments are input into a reasoning tool 'so that requirement satisfaction can be reasoned about'.

2) Relationship to informal argument: The arguments might be initially formulated in extended Toulmin form, but are ultimately formalised [22].

3) Effect on argument structure: It is not clear whether the formalisation process imposes any limits on the structure of the original informal arguments [22].

4) Noted benefits and drawbacks: Tun et al. claim that 'formalization of privacy norms and arguments are [sic] useful because they can be used to check some important privacy properties in the system. These properties include: (1) information availability ..., (2) denial ..., and (3) explanation'. The authors mention no potential drawbacks.

IV. Assurance Argument Fallacies

Six of the twenty papers make or imply claims that mechanical validation will justify greater confidence in the argument's conclusions [9], [11], [16]–[18], [39]. None provide substantial evidence for this claim. We cannot know why authors did not hypothesise specific benefits or provide appropriate evidence. However, some readers might assume that formality reduces the risk of human error. In this section, we explore the errors that formality might help to catch.

The philosophy community has long been concerned with the ways in which arguments might be defective. Whole books have been written on the subject [42]. Philosophers and safety researchers have even collaborated to define a taxonomy of fallacies specific to safety argumentation [40]. Understanding the potential of formalism to detect and prevent faulty reasoning requires understanding the difference between *formal fallacies* and *informal fallacies*.

A. The Formal Fallacies

A formal fallacy is a flaw in the *form* of an argument [42]. If we replaced an argument's identifiers with meaningless symbols, the flaws we could identify in the relationships between those symbols are formal fallacies. Names and definitions of fallacies vary, but one textbook lists eight: (1) *begging the question*, (2) *incompatible premises*, (3) *contradiction between premise and conclusion*, (4) *denying the antecedent*, (5) *affirming the consequent*, (6) *false conversion*, (7) *undistributed middle term*, and (8) *illicit distribution of an end term* [42]. Formal fallacies are often defined in terms of symbols. For example, an argument in which C is both the conclusion and a premise begs the question.

B. The Informal Fallacies

An informal fallacy cannot be detected through examination of argument form alone. Philosophers have been cataloguing informal fallacies for millennia: Aristotle identified what we now call the *equivocation* fallacy – in which one

identifier represents different meanings in different parts of the argument – in 350 BCE [43]. There are many modern catalogues of fallacies (e.g., [42]). As with formal fallacies, lists differ: some include fallacies that others omit, and different lists might define the same fallacy differently.

Care is needed to identify informal fallacies. For example, *arguing from ignorance* is defined as 'arguing for the truth (or falsity) of a claim, because there is no evidence or proof to the contrary' [42]. Such arguments look very like legitimate arguments for the absence of something. For example, suppose that a suburban householder argues the that there is no car in her garage because she has opened it, looked inside, and seen no car. Her argument is sound even though it claims the absence of a car based on the lack of a counterexample. We should accept her conclusion to a degree limited by the adequacy of her search procedure.

C. What Formalism Can't Help With

Computers process the form of arguments but not their real-world meaning. Thus, mechanical verification might identify formal fallacies but cannot show the absence of informal fallacies. For example, consider the formalised argument about the fictional Desert Bank given in Figure 1. Banks are adjacent to rivers and syllogisms are valid deductive arguments. Given these (true) premises, Prolog falsely concludes that the Desert Bank is adjacent to a river. The equivocation is obvious to a human: 'bank' refers to two different real-world entities. But because equivocation is an informal fallacy, automatic verification cannot catch it.

Merely rendering an argument into a symbolic, deductive form and subjecting it to proof checking cannot show that the conclusion is trustworthy. Submitting *only* the informal parts of a partially-formalised argument to human review is also insufficient. In a full-sized software safety argument, there will be seemingly-problematic formal premises such as claims about execution times measured under the false assumption that tasks are never interrupted. Only by examining how such premises are used in the argument can humans determine whether the argument is sound.

V. What We Know So Far

Given the role argumentation now plays in system development and certification (e.g., [1], [3]), the potential to increase confidence in safety conclusions or solve practical problems is of interest. But it is not clear that the problems that formalisation could solve are unacceptably prevalent in practice. In this section, we explore what is presently known and how this relates to claimed benefits of formalisation.

A. Do Writers Instantiating Patterns Make Syntax Errors?

Four of the selected papers suggest formalising the syntax of graphical arguments whose elements contain natural language text [11], [12], [17], [18]. These papers offer no evidence that writers of assurance arguments routinely err in

From these premises:

```
is_a(desert_bank, bank).
adjacent(bank, river).
adjacent(X, Y) :- is_a(X, Z), adjacent(Z, Y).
```

We can 'prove' that:

```
adjacent(desert_bank, river).
```

Figure 1. A flawed argument (in Prolog) that would pass formal validation

ways that formal verification could detect, e.g., by omitting claims from pattern instantiations or replacing placeholders standing for the same concept with incompatible text. We know neither of such evidence nor of evidence that such errors would not be caught by a simple manual review of the argument. If formality solves a rare problem, the side effects of the cure might be worse than the disease.

B. Are Formal Fallacies Prevalent in Practice?

Eleven of the selected papers suggest formalising all or part of the content of arguments into symbolic, deductive logic [8], [9], [14]–[16], [19], [20], [22], [24], [25], [39]. Four of these explicitly mention mechanical verification of the formalised argument [9], [19], [20], [22]. Mechanical verification is an obvious reason to use a symbolic, deductive logic. However, only one paper makes any effort to show that the problem mechanical checking could solve actually exists [39]. That paper claims that a review of three safety arguments [40] showed that 'logical fallacies' are common in safety arguments and that formalisation will allow tools to identify these. But this line of reasoning requires showing that formal fallacies, not informal fallacies, are common. In the three arguments, Greenwell et al. actually found seven kinds of fallacies [40]: (a) 3 instances of *drawing the wrong conclusion*, (b) 10 instances of *fallacious use of language*, (c) 2 instances of *fallacy of composition*, (d) 4 instances of *hasty inductive generalisation*, (e) 5 instances of *omission of key evidence*, (f) 5 instances of *red herring*, and (g) 16 instances of *using the wrong reasons*. Greenwell's fallacy taxonomy for safety arguments does not divide fallacies into formal and informal categories [44]. But none of seven kinds of fallacies found is strictly formal, and it is not clear that formalisation would help to identify or avoid these.

A proof checker can prevent drawing the wrong conclusion from symbolic premises using deductive logic. However, one can still assert a formal premise on the basis of evidence that doesn't support it (e.g., `wcet(task_1, 250)` because of 'unit test results'). One might also assert that a conclusion follows from formal premises that don't support it (e.g., `code_reviewed` $\wedge$ `unit_tests_passed` $\Rightarrow$ `meets_deadlines`). Human review of the asserted premises is needed to detect this fallacy.

The identified fallacious uses of language were ambiguity. Symbols might be unambiguous, but the natural language that binds them to a real-world meaning can be ambiguous.

In a hasty inductive generalisation, the arguer claims that because a proposition is true for some set members, it is true for all. Formalisation into deductive logic might drive such reasoning into the informal part of the argument. Alternatively, the arguer might simply assert the inductive generalisation as a deductive rule. A proof checker cannot know whether a set used in a formal, deductive argument is complete with respect to the real world entity it models.

A fallacious composition is an argument that erroneously concludes that something has a property because each of its components does. But the fallacy exists only where the parts *can* interact to affect the property. A theorem prover cannot know how elements in the real world can interact.

Detecting omission of key evidence requires understanding what evidence is key to a given argument. Formalisation might force arguers to assert that a conclusion can be drawn from certain premises, but cannot validate such assertions.

Red herrings are a little different. Proof checkers are not distracted by (formally) irrelevant premises. But red herrings are just one form of non sequitur; an arguer might assert a rule that allows them to draw a conclusion from an irrelevant premise (amongst others). If argument confidence is assessed mechanically (e.g., through BBN modelling [34]), asserting such a rule would artificially raise the assessed confidence. Symbol names raise a separate issue: while names are meaningless to machines, arguers might choose names that mislead human readers.

An arguer uses the wrong reasons when the premises are not 'appropriate to the claim' [42]. This is similar to drawing the wrong conclusions and could be missed in the same way.

Human review is needed to catch such informal fallacies. Formalisation might move such errors to the informal part of the argument or encode them as false premises, but machine checking alone cannot eliminate them.

C. Do Human Reviewers Miss Formal Fallacies?

Human reviewers can fail to spot fallacies: Greenwell et al. report results from two different reviewers that show that each overlooked some fallacies that the other flagged [44]. (Perfect agreement between reviewers is not expected: a single bad reasoning step might plausibly satisfy the definitions of more than one informal fallacy.) But it is the efficacy of humans at spotting formal fallacies that is at issue in the argument for formalisation, and this remains unknown.

VI. How We Could Answer the Question of Cost-Effectiveness

A sound assessment of formalisation of assurance arguments must consider whether the benefits outweigh the costs, including the costs of side effects. If making it harder for some readers to understand the argument leads to more safety-related errors than formal verification finds, formalisation might make systems less safe. Direct, timely, holisitic assessment of the overall benefit of formalisation would be difficult to provide. But we could assess specific benefits, costs, and side effects that might affect the overall benefit of arguments playing the roles discussed in subsection II-A. In this section, we propose assessments of five such factors: (1) the ability to automatically identify formal fallacies, (2) the effort of formalisation, (3) restriction of the reading audience, (4) more reliably correct pattern instantiation, and (5) complication of evidence sufficiency judgments.

A. The Ability to Automatically Identify Formal Fallacies

Even if informal fallacies necessitate human review of assurance arguments, automatic detection of formal fallacies might yield faster reviews, more reliable identification of formal fallacies, or both. We could measure the effect on effort experimentally: one group of volunteers reviews an argument for informal fallacies only, the other for both informal and formal fallacies, and the experimenters measure time taken. The number of formal fallacies missed in manual review can be counted. The results might vary from system to system, but even data from one or two systems would be better than having no assurance-argument-specific data.

B. The Effort of Formalisation

Three of our selected papers proposed constructing arguments first in informal form and then formalising them [9], [19], [22]. In these cases, formalisation poses a cost that must be compared to any benefits. This cost could be measured by observing volunteers performing the formalisation task and measuring the time needed. (The study design would have to account for learning effects and for the impact of formal methods expertise.) Comparing this cost to an operational risk benefit might be difficult, but theories of risk acceptability [45] address this standard challenge.

C. Restriction of the Reading Audience

Assurance arguments serve several purposes, including communicating concepts such as how the system will manage risk to a wide variety of stakeholders (see subsection II-A). But while software engineers learn symbolic, deductive logics at university, this is not necessarily true of managers, mechanical engineers, or safety assessors. We should know the degree to which a proposed formalism will hinder the argument's communication purpose. To assess this, we could experimentally measure reading speed and comprehension, using an informal version of the specimen argument as a control. Subjects should be selected from the backgrounds that might be expected of an argument reader. A questionnaire should be used to collect information about each subject's background and training to explore how these factors affect reading ability.

D. More Reliably Correct Pattern Instantiation

Three of our selected papers proposed formalising argument pattern structure [11], [17], [18]. Two also propose

formalising pattern parameters [17], [18]. But, as discussed in subsection V-A, we do not know whether the errors that such formalisation might prevent are prevalent in practice. We could measure this. Even better, we could measure and compare defect rates between volunteers who instantiate informal patterns and review them and volunteers that use a formalised pattern instantiation tool with parameter checking. We could also measure whether the proposed mechanism speeds up or slows down argument creation. Again, the effect of formalisation might vary from practitioner to practitioner and argument to argument. But even a few measurements would be better than no data at all.

E. Complication of Evidence Sufficiency Judgments

As Rushby and others have noted, judging evidence sufficiency requires determining whether evidence is good enough for a given purpose [19], [34]. Because the meaning of 'good enough' depends on the consequence of drawing a false conclusion from the evidence, an assessor making this judgement must consider the claims that the evidence directly and indirectly supports. Graphical argument notations such as GSN and CAE are thought to ease this task by reducing it to tracing a path in a graph. Rushby proposes instead that developers should assess impact by eliminating the corresponding formal premise and rerunning the proof checker to assess the impact on the safety argument [20]. (He does not explain how evidence sufficiency should be judged in cases where an error is likely to be a matter of degree, e.g., execution time evidence.) We can measure the impact of formalisation on both the time needed to make evidence sufficiency judgments and the reliability of those judgments by experimentally assessing the performance of volunteers. We are unlikely to know the ground truth but could measure inter-assessor agreement: if many assessors report similar values, they might be right or wrong, but if they report very different values, at least some *must* be wrong.

VII. Conclusion

Several researchers have proposed formalising assurance arguments. In this paper, we have surveyed twenty such papers. Readers of papers proposing a technique or method need to know how mature that proposal is and what is known about its cost, side effects, and efficacy. (Such information both underpins cost-benefit decisions and informs other researchers.) A proposal is mature when its benefits are clearly defined and evidence shows that benefits outweigh costs and side effects. But while several of the selected papers claim or speculate on some benefit of formalism, none supplies substantial empirical evidence to support such claims. None supplies even substantial evidence that there is a problem that formalisation could solve.

Preliminary work on immature ideas is necessary. However, such work should be clearly identified as preliminary lest naïve readers (e.g., some practitioners new to safety arguments) be misled. The work's assumptions should be stated and the proposed future development sketched for the reader's benefit. Yet in the surveyed work, only Rushby correctly and candidly acknowledges that any benefit of formal assurance argumentation is a *hypothesis* that requires substantiation through empirical assessment [19], [20].

Safety and security assurance is important; we urgently need to know whether the proposals will yield overall benefit. (We hope they do.) To that end, we have surveyed what is presently known and how this relates to argument formalisation. Where evidence is lacking, we have sketched empirical studies that could provide it.

Acknowledgment

This work was funded by the Swedish Foundation for Strategic Research as part of the SYNOPSIS project and by Artemis as part of the nSafeCer project (grant 295373).

References

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

[2] T. P. Kelly, "Arguing safety — A systematic approach to managing safety cases," Ph.D. dissertation, University of York, 1998.

[3] ISO 26262-2:2011, *Road vehicles — Functional safety — Part 2: Management of functional safety*. International Organization for Standardization, 2011.

[4] C. B. Haley, J. D. Moffett, R. Laney, and B. Nuseibeh, "Arguing security: Validating security requirements using structured argumentation," in *Proc. Symp. on Requirements Engineering for Information Security (SREIS)*, 2005.

[5] G. Despotou, "Managing the evolution of dependability cases for systems of systems," Ph.D. dissertation, University of York, 2007.

[6] N. Basir, E. Denney, and B. Fischer, "Deriving safety cases from automatically constructed proofs," in *Proc. IET Int'l Conf. on Systems Safety*, 2009.

[7] ——, "Deriving safety cases for hierarchical structure in model-based development," in *Proc. Int'l Conf. on Computer Safety, Reliability and Security (SAFECOMP)*, 2010.

[8] P. Bishop and R. Bloomfield, "The SHIP safety case approach," in *Proc. Int'l Conf. on Computer Safety, Reliability and Security (SAFECOMP)*, 1995.

[9] J. Brunel and J. Cazin, "Formal verification of a safety argumentation and application to a complex UAV system," in *Proc. Wksp. on Dependable and Secure Computing for Large-Scale Complex Critical Infrastructures (DESEC4LCCI)*, 2012.

[10] E. Denney, G. Pai, and J. Pohl, "Heterogeneous aviation safety cases: Integrating the formal and the non-formal," in *Proc. Int'l Conf. on Engineering of Complex Computer Systems (ICECCS)*, 2012.

[11] E. Denney and G. Pai, "A formal basis for safety case patterns," in *Proc. Int'l Conf. on Computer Safety, Reliability, and Security (SAFECOMP)*, 2013.

[12] E. Denney, G. Pai, and I. Whiteside, "Hierarchical safety cases," in *Proc. NASA Formal Methods Symp. (NFM)*, 2013.

[13] E. Denney, D. Naylor, and G. Pai, "Querying safety cases," in *Proc. Int'l Conf. on Computer Safety, Reliability, and Security (SAFECOMP)*, 2014.

[14] J. Forder, "A safety argument manager," in *IEE Colloq. on Software in Air Traffic Control Systems — The Future*, 1992.

[15] C. B. Haley, J. D. Moffett, R. Laney, and B. Nuseibeh, "A framework for security requirements engineering," in *Proc. Int'l Wksp. on Software Engineering for Secure Systems (SESS)*, 2006.

[16] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh, "Security requirements engineering: A framework for representation and analysis," *IEEE Transactions on Software Engineering*, vol. 34, no. 1, pp. 133–153, 2008.

[17] Y. Matsuno and K. Taguchi, "Parameterised argument structure in GSN patterns," in *Proc. Int'l Conf. on Quality Software*, 2011.

[18] Y. Matsuno, "A design and implementation of an assurance case language," in *Proc. IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN)*, 2014.

[19] J. Rushby, "Formalism in safety cases," in *Proc. Safety-Critical Systems Symposium (SSS)*, 2010.

[20] ——, "Logic and epistemology in safety cases," in *Proc. Int'l Conf. on Computer Safety, Reliability, and Security (SAFECOMP)*, 2013.

[21] ——, "Mechanized support for assurance case argumentation," in *Proc. Int'l Wksp. on Argument for Agreement and Assurance (AAA)*, 2013.

[22] T. T. Tun, A. K. Bandara, B. A. Price, Y. Yu, C. Haley, I. Omoronyia, and B. Nuseibeh, "Privacy arguments: Analysing selective disclosure requirements for mobile applications," in *Proc. IEEE Int'l Requirements Engineering Conf. (RE)*, 2012.

[23] P. Tolchinsky, S. Modgil, K. Atkinson, P. McBurney, and U. Cortés, "Deliberation dialogues for reasoning about safety critical actions," *Autonomous Agents and Multi-Agent Systems*, vol. 25, no. 2, pp. 209–259, 2012.

[24] T. T. Tun, Y. Yu, C. Haley, and B. Nuseibeh, "Model-based argument analysis for evolving security requirements," in *Proc. Int'l Conf. on Secure Software Integration and Reliability Improvement (SSIRI)*, 2010.

[25] Y. Yu, T. T. Tun, A. Tedeschi, V. N. L. Franqueira, and B. Nuseibeh, "OpenArgue: Supporting argumentation to evolve secure software systems," in *Proc. IEEE Int'l Requirements Engineering Conf. (RE)*, 2011.

[26] X. Yin, J. C. Knight, and W. Weimer, "Exploiting refactoring in formal verification," in *Proc. IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN)*, 2009.

[27] T. Kelly, "A systematic approach to safety case management," in *Proc. SAE 2004 World Congress*, 2004.

[28] P. J. Graydon, "A perspective on safety argumentation: Aims, achievements, challenges, and opportunities," in *Proc. Int'l Wksp. on Argument for Agreement and Assurance (AAA)*, 2013.

[29] Nationale Genossenschaft für die Lagerung Radioaktiver Abfälle, "Project opalinus clay: Safety report," NAGRA, Tech. Rep. NTB 02-05, 2002.

[30] K. Attwood *et al.*, *GSN Community Standard, Version 1*. Origin Consulting Limited, 2011.

[31] P. Bishop and R. Bloomfield, "A methodology for safety case development," in *Proc. Safety-Critical Systems Symposium (SSS)*, 1998.

[32] C. M. Holloway, "Safety case notations: Alternatives for the non-graphically inclined?" in *Proc. IET Int'l Conf. on System Safety*, 2008.

[33] S. E. Toulmin, *The Uses of Argument*, 2nd ed. Cambridge University Press, 2003.

[34] P. J. Graydon, "Uncertainty and confidence in safety logic," in *Proc. Int'l System Safety Conf. (ISSC)*, 2013.

[35] ——, "Towards a clearer understanding of context and its role in assurance argument confidence," in *Proc. Int'l Conf. on Computer Safety, Reliability and Security (SAFECOMP)*, 2014.

[36] G. Gentzen, *The Collected Papers of Gerhardt Gentzen*. North Holland, 1969.

[37] Respect-IT, *A KAOS Tutorial, V1.0*. Respect-IT, 2007.

[38] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, pp. 131–164, 2008.

[39] O. Sokolsky, I. Lee, and M. Heimdahl, "Challenges in the regulatory approval of medical cyber-physical systems," in *Proc. Int'l Conf. on Embedded Software (EMSOFT)*, 2011.

[40] W. S. Greenwell, J. C. Knight, C. M. Holloway, and J. J. Pease, "A taxonomy of fallacies in system safety arguments," in *Proc. Int'l System Safety Conf. (ISSC)*, 2006.

[41] E. T. Mueller, *Commonsense Reasoning*. Morgan Kaufmann, 2006.

[42] T. Damer, *Attacking Faulty Reasoning: A Practical Guide to Fallacy-Free Arguments*, 5th ed. Thomson Wadsworth, 2005.

[43] Aristotle, *On Sophistical Refutations*. The Internet Classics Archive, 350 BCE, translated by W. A. Pickard-Cambridge.

[44] W. S. Greenwell, J. J. Pease, and C. M. Holloway, "Safety-argument fallacy taxonomy," Electronic document, 2006.

[45] A. J. Rae, "Acceptable residual risk: Principles, philosophy and practicalities," in *Proc. IET System Safety Conf.*, 2007.