SRI International

CSL Technical Report SRI-CSL-2024-01 • May 16, 2024

Defeaters and Eliminative Argumentation In Assurance 2.0

Robin Bloomfield (Adelard, part of NCC Group, and City, Univ. of London), Kate Netkachova (Adelard), and John Rushby (SRI)

As Members of the CLARISSA Team
Honeywell, Adelard, UT Dallas, and SRI

Also issued as a CLARISSA Technical Report under the title Defeaters and Eliminative Argumentation in CLARISSA

SRI Project 100651 under subcontract to Honeywell in support of AFRL and DARPA ARCOS Program, Contract FA8750-20-C-0512.
Distribution Statement “A” (Approved for Public Release, Distribution Unlimited).

Computer Science Laboratory • 333 Ravenswood Ave. • Menlo Park, CA 94025 • (650) 326-6200 • Facsimile: (650) 859-2844

Abstract

A traditional assurance case employs a positive argument in which reasoning steps, grounded on evidence and assumptions, sustain a top claim that has external significance. Human judgement is required to check the evidence, the assumptions, and the narrative justifications for the reasoning steps; if all are assessed good, then the top claim can be accepted.

A valid concern about this process is that human judgement is fallible and prone to confirmation bias. The best defense against this concern is vigorous and skeptical debate and discussion in the manner of a dialectic or Socratic dialog. There is merit in recording aspects of this discussion for the benefit of subsequent developers and assessors. *Defeaters* are a means doing this: they express doubts about aspects of the argument and can be developed into subcases that confirm or refute the doubts, and can record them as documentation to assist future consideration.

This report describes how defeaters, and multiple levels of defeaters, should be represented and assessed in Assurance 2.0 and its CLARISSA/ASCE tool support. These mechanisms also support *eliminative argumentation*, which is a contrary approach to assurance, favored by some, that uses a negative argument to refute all reasons why the top claim could be false.

Contents

List of Figures

1 Introduction to Defeaters

This report assumes general familiarity with Assurance Cases [19], and with Assurance 2.0 [4] and its CLARISSA toolset [22] in particular.

The primary criterion for a satisfactory assurance case in the Assurance 2.0 methodology is that it should justify *indefeasible* confidence in its top claim, meaning that in addition to confidence that the claim is `true`, we must also be confident that there are no overlooked or unresolved doubts that could change that judgement [5, 20]. We refer to any concern about a case as a *doubt* and we annotate the case by adding a doubt node to the graphical representation of the assurance argument, pointing to a node that is under suspicion. The doubt node contains a claim indicating the nature of the doubt (e.g., "I think there is something wrong here"). At some point, we must return to investigate the nature and origin of the doubt and will either dismiss it as unwarranted, or refine and sharpen it into a *defeater* with a possibly more specific (counter-)claim (e.g., "the justification for this step is inadequate") whose investigation is recorded in a subcase attached to the defeater. Thus, a doubt is simply a defeater that has not yet been investigated (i.e., has no subcase) and so we will generally refer to both as defeaters. The back and forth investigation of an assurance case argument against doubts and defeaters is an application of the *Socratic* or *dialectical* methods for exposing error and refining beliefs.[1] These date back to ancient Greece but retain their potency. In particular, defeaters play a dialectical rôle in argument that is similar to falsification in science [11]. Thus, identification of potential defeaters should not be seen as criticism but as a contribution to the development and clear formulation of an assurance case and part of a process to establish its indefeasibility. In addition, developers should consciously generate doubts, and vigorously investigate their associated defeaters as a guard against confirmation bias, and evaluators may raise potential defeaters as a way to elicit additional explanation or to clarify their understanding of some part of an assurance case.

This report is concerned with the representation, evaluation, and recording of defeaters within an assurance case. It does not address systematic search for defeaters, which is an important topic akin to hazard analysis in systems (a defeater for an argument is like a hazard to a system) and, indeed, some defeaters may reveal previously unconsidered hazards. We will examine methods of searching for potential defeaters in a separate document.

Because they make (contrary) claims, the CLARISSA/ASCE graphical tool for Assurance 2.0 uses the same oval node for defeaters as for claims, but they are colored differently (red for defeaters). An example (described in more detail in Section 5) is shown in Figure 1, where a defeater (at upper right) attached to

[1] In philosophy, the Socratic method is considered an instance of dialectic [26]; the precise distinctions do not concern us here.

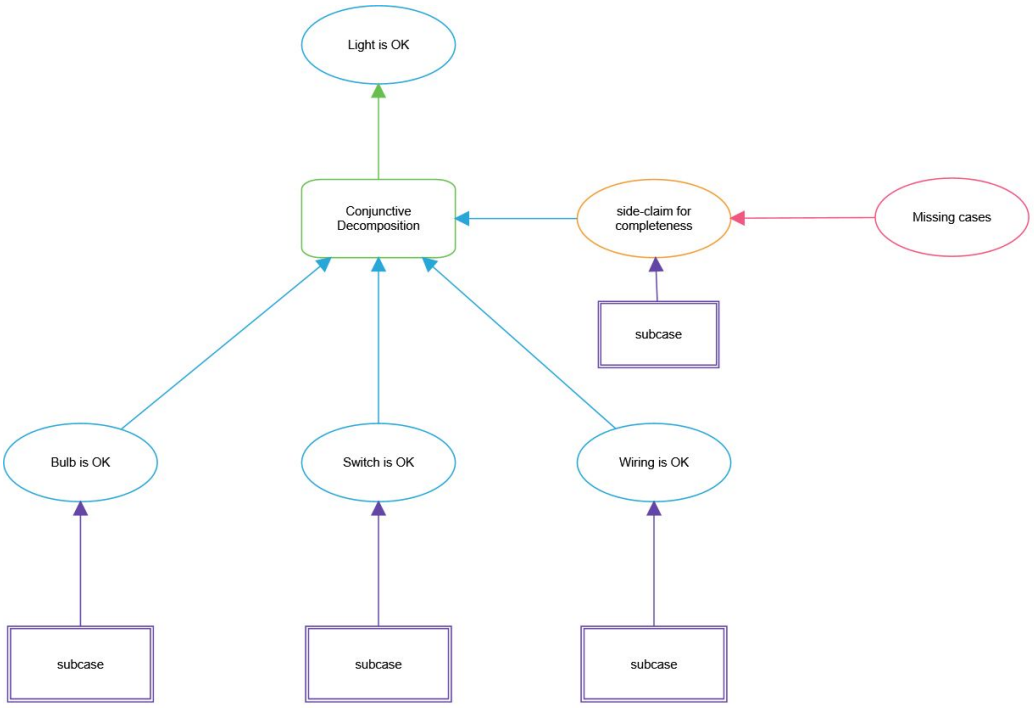

Figure 1: Doubt as a Defeater with No Subcase (upper right)

the sideclaim for the decomposition block claims that there are overlooked cases in the decomposition. This defeater has no subcase below it, so it has not yet been investigated and represents a doubt. The presence of a doubt or incompletely investigated defeater causes the node that it indicates to be considered `unsupported` and this will propagate (as described in Section 2) to the top claim and thereby prevents the assurance case being considered complete or "closed." Investigation of a doubt/defeater will be recorded as an assurance subcase that confirms or refutes its claim, as shown (for a more developed treatment of the same example) in Figure 4.

If a defeater is supported by an assurance subcase that is adjudged to be sound, so that its claim is `true`, then the defeater is said to be confirmed or *sustained* and the original case, and possibly the system it is about, must be modified to overcome the flaw that has been identified. (Alternatively, the flaw may be explicitly accepted as a *residual risk*, provided it is judged suitably insignificant [5, Section 6].)

After these modifications, the defeater and its subcase will (or should) no longer apply, but we might like to retain them in the case as documentation to assist future developers and evaluators. Thus, Clarissa/asce allows a defeater to be marked as *addressed* and it and its subcase are then treated as a comment. Because the

defeater does not apply to the modified primary case, a narrative description of the original problem and its resolution should be added to the defeater node. This may be difficult to understand (because the context is changed in the modified case), so an alternative is to modify the previously sustaining subcase for the defeater into a refuted subcase (see below) for the now modified primary case.

If we suspect that a defeater is a "false alarm," or it is one that has been overcome by modifications to the original case (as above), then our task is to *refute* it: that is, to provide it with a subcase that shows it to be `false`. One way to do this is with a second-level defeater that targets the first defeater or some part of its subcase (for example, the assurance case shown in Figure 4 has a second defeater that attacks one of the claims in the subcase for the first). If the assurance subcase for that second-level defeater is adjudged to be good, then the first defeater is said to be *refuted* and it and its subcase play no part in the interpretation of the primary case, but can be retained as a kind of commentary to assist future developers and evaluators who may entertain doubts similar to that which motivated the original defeater.

Another way to initiate refutation is by means of *counter-evidence*: that is, evidence that contradicts the claim it is meant to support (for example, test evidence may reveal failures). We discuss refutational reasoning in general in Section 2 below.

1.1 Representation of Defeaters

We first recap our terminology for the elements of graphical assurance cases in Clarissa/asce. An assurance case is composed of various kinds of *nodes*, each having a distinctive shape (e.g., oval for claims, and rounded rectangle for argument nodes). An argument (building) *block* consists of an argument node with a parent claim (usually drawn above it), optional sideclaims (usually drawn to the side), and subclaims or evidence (usually drawn below it). The parent claim of one block will be a subclaim or sideclaim to another block (except for the top claim). Thus, the overall argument forms a tree (sometimes with cross links, so it is technically a graph). There are just five kinds of argument block: concretion, substitution, decomposition, calculation, and evidence incorporation.

A consequence of our determination that defeaters function more like a critique or commentary than part of the logical evaluation of an assurance case is that in Assurance 2.0 we allow defeaters to be attached (i.e., point) to any node in an assurance argument. Specifically, in the graphical representation of arguments used in Clarissa/asce, a defeater node has the same oval shape as a claim node (but is colored differently—red instead of blue) and contains a logical claim, but we allow it to point to another claim with no intervening (rounded rectangular) argument node. Similarly, it may point to an argument, evidence, or subcase node without being a subclaim or sideclaim of that node. And, of course, it can point directly to

another defeater node. To aid visual recognition of defeaters and their subcases, we recommend that defeater nodes are placed to the side of (rather than below) the node they point to and that their subcase is developed below them. This allows defeater subcases to be isolated and highlighted, as portrayed in Figure 2.

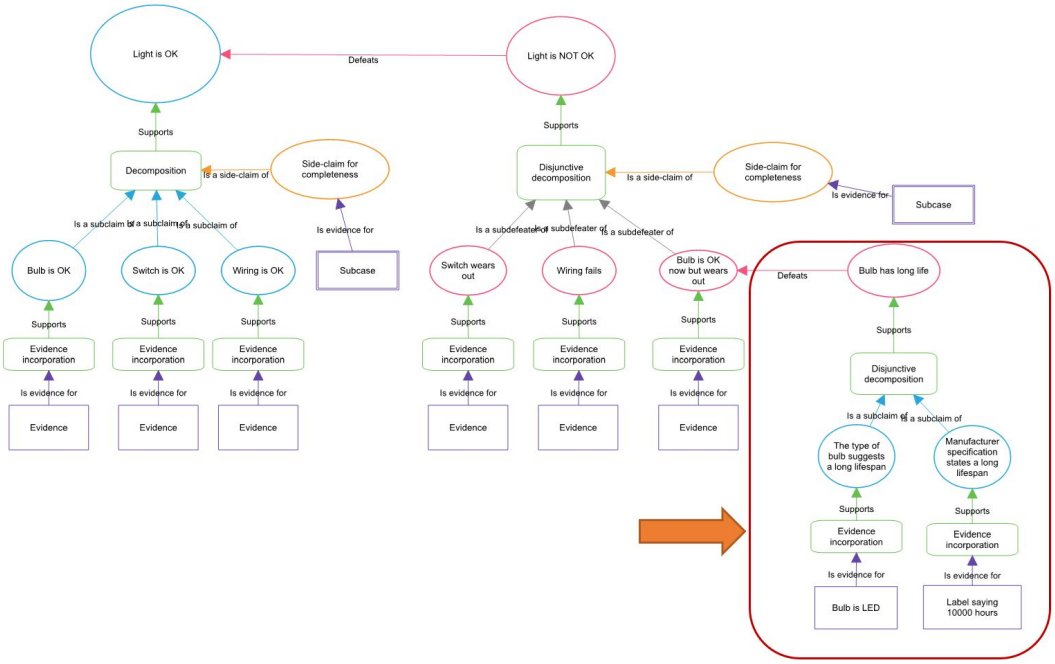

Figure 2: Highlighted Defeater Subcase

The claim in a defeater may be vague (e.g., "I think there's something missing here"), which is appropriate when the defeater is indicating an as yet unexplored doubt, or it may be specific, such as "this claim is false," where "this claim" is the one pointed to, or "the justification in this argument node is inadequate." It is a human judgment whether the claim in the defeater truly affects the credibility or relevance of the node pointed to, but in the propagation of suspicion performed by CLARISSA/ASCE (described in Section 2.2) it is assumed that defeaters, if `true`, really do "defeat" the node pointed to.

To deal systematically with refutation and with defeaters at several levels, we need to extend our tool support to include *refutational reasoning*. That is to say, CLARISSA/ASCE needs to assess whether claims may be `false`, in addition to its prior assessment of whether they are `true` or `unsupported`. We develop rules for determination and propagation of these assessments in the following sections.

2 Assessing Arguments With Defeaters

We saw that investigation of a refuted defeater could introduce a second-level defeater and, in general, investigation and resolution of defeaters may lead to defeaters at multiple levels. Thus, the focus of this report is how to assess assurance cases in the presence of defeaters, including those at multiple levels. Since the argument for an assurance case in Assurance 2.0 employs Natural Language Deductivism (NLD) [5, Section 1.3], it represents something close to a logical proof,[2] and it might seem that we could look towards logical notions such as nonmonotonic logics and defeasible reasoning[3] for ideas on how arguments with defeaters should be interpreted.

However, the goal of nonmonotonic logic and of defeasible reasoning is to work out what can be concluded when there are contradictory premises or when exceptions are added to premises (e.g., a premise "birds fly" gains the exception "unless they are penguins"), but these concerns do not apply in the same way to assurance cases. In Assurance 2.0, defeaters are a transient phenomenon and are "active" (as opposed to serving as commentary) only during development or exploratory assessment of the case. So, when we say that our topic is how to assess assurance cases in the presence of multiple levels of defeater, we do not mean how to evaluate the truth of contested claims but how to determine which parts of a case *are* contested—that is, called into question by active defeaters, and therefore considered "unclosed."

Because we do not fully evaluate assurance cases with active defeaters, but merely assess which parts of the case are closed and which parts are still open to question, the actual claim made by a defeater plays little part in the analysis. In particular, if a defeater with claim X pointing to a claim A is sustained, we do not suppose that some logical combination of A and X is thereby justified; we accept that the claim A is challenged and revise it and/or its supporting subcase to overcome the source of doubt. Of course, as noted earlier, we must make the human judgement that X has some impact on the credibility or relevance of A but we do not reduce this to some logical requirement such as $X \equiv \neg A$.[4] Having said that, in Section 3 we will introduce a circumstance where we do recognize the special case where the claim in a defeater is the negation of that in the node that it points to; we call these *exact* defeaters (and the general kind are then known as *exploratory* defeaters).

When a first-level defeater is sustained, the assurance case is unsound and must be adjusted; furthermore, the system concerned may be flawed or even unsafe and

[2] It is close rather than equivalent to a logical proof because an assurance case may choose to accept residual doubts, such as nondeductive reasoning steps, or imperfectly justified premises.

[3] These topics are outlined in our Confidence Report [5, Section 5.1].

[4] We use and $\neg$ for negation, $\wedge$ for conjunction, $\vee$ for disjunction, $\supset$ for material implication, and $\equiv$ for equivalence.

can require adjustment also.[5] Thus, sustained first-level defeaters are a serious matter and should be transient phenomena that arise only briefly during development or while exploring a case during assessment, and are then fixed. Hence, the assurance goal most generally associated with first-level defeaters is to refute them; this annuls the doubt that motivated the defeater and leaves the original case unscathed.

One way to refute a first-level defeater is to add a second-level defeater, but this must not degenerate into a cascade where a claim A is challenged by a first-level defeater with the counter-claim $\neg A$ and a second-level defeater challenges that with the counter-counter-claim $\neg\neg A$; this is equivalent to A in classical logic and we have achieved nothing.[6] However, the claim in a defeater need not be the negation of the claim that it points to (and, indeed, it can point to a node that is not a claim); it merely needs to assert something that calls the targeted claim or node into question. Thus, a cascade of defeaters may be acceptable when they are based on different concerns. Usually, however, lower-level defeaters (i.e., those that challenge other defeaters) should dispute claims in the *subcase* of the defeater above them: thus, the subcase for a defeater claiming A may decompose into subclaims for B, C, and D and it is reasonable for a second-level defeater to challenge B (Figure 4 illustrates this).

As noted earlier, dealing systematically with refutation and with defeaters at several levels, requires our tool support to be extended with *refutational reasoning*. That is to say, Clarissa/asce needs to assess whether claims may be `false`, in addition to its prior assessment of whether they are `true` or `unsupported`. We develop rules for determination and propagation of these assessments in the following subsections, starting with positive assessments, and then proceeding to refutations.

2.1 Propagation of Positive Assessments

Prior to the introduction of defeaters, Clarissa/asce needed to consider only positive arguments, and did so using Natural Language Deductivism, NLD. That is, assessment focused on the credibility and ultimately the indefeasibility of each argument block. In particular, developers and assessors would assure themselves that the parent claim to each reasoning block was indefeasibly entailed by its subclaim(s), given any side claim, and that the narrative justification provided adequate documentation for this. When evidence incorporation blocks also passed equivalent scrutiny, and assumptions likewise, then it could be concluded by compositionality that the overall assurance case provided indefeasible support for the top claim.

[5] When a defeater reveals a flaw in the system, one possible response is to add a runtime monitor [12,18] that will detect and mask the hazardous condition; a systematic approach can be developed along these lines [8].

[6] The reference to *classical logic* is in opposition to *intuitionistic logic*, which eschews the law of the excluded middle or, equivalently, the law for elimination of double negation. We briefly discuss this topic in our Confidence Report [5, page 57].

More formally, the argument blocks of an assurance case argument function as logical premises with the top claim as conclusion. The case is then logically *valid* if the argument blocks "fit together": that is, the claims of the argument should form a tree where the subclaims and sideclaim of each argument block match the parent claim of another block, with the exception of claims representing assumptions and residual risks (the rule is simple because an assurance argument uses only propositional logic). This is enforced automatically by the graphical construction of arguments used in CLARISSA/ASCE: the *same* claim node is used for those that should match, connected by explicit arrows to the argument nodes concerned.

A logically valid argument is (logically) *sound* if we believe its premises to be `true`: for CLARISSA/ASCE, this means the narrative provided for each argument block must be judged to provide (indefeasibly) credible justification for the argument step represented by that block. Notice that whereas validity concerns only the "form" of the argument (i.e., do the claims match), soundness concerns the *meaning* of the claims and argument blocks involved.

CLARISSA/ASCE could record assessment of soundness by attaching a checkmark to each argument or evidence incorporation node, so that developers or assessors can check the box if they consider the node's narrative does provide indefeasible justification for its parent claim, and not otherwise. However, CLARISSA/ASCE does not do this: instead, it makes the default assumption that argument and evidence incorporation nodes are sound, and defeaters are used to indicate otherwise. We prefer to use defeaters rather than checkmarks for this purpose because the defeater and its subcase can explicitly record the reason for unsoundness.

This use of defeaters means that assessment of soundness is now represented in the argument and thereby enters into determination of validity: for example, if we point a defeater to an argument node and claim "I do not believe this narrative justification," then (un)soundness of that argument node is represented in the logical evaluation of the defeater, and hence in validity of the overall case. This is useful because it eliminates the need for a separate check for soundness and for ways to perform and indicate its propagation: it is all handled by the mechanisms for validity and defeaters.

However, determination of validity becomes a little more complicated than before because we have to account for defeaters, and we also allow incompletely developed arguments. In these cases, we cannot expect the top claim to be adjudged `true`; instead, we can ask which claims *are* `true` and which are `unsupported` (i.e., have contested or incomplete subcases), and we can do this by propagating logical assessments upward from the leaf nodes.[7]

[7]It can be argued that doubt should propagate downward as well as upward: for example, if a concretion block below the top node is challenged, then surely it calls the whole development into question as it suggests that the case is establishing an incorrect claim. Unfortunately, determination of which parts of a case should be adjusted to correct a contested argument cannot be reduced to

There are five kinds of leaf nodes: claims lacking a subcase (typically indicating undeveloped parts of the case), which are considered `unsupported`; assumptions, which are claims lacking a subcase that are specially designated and justified, and are assessed `true`; residual risks, which are are defeaters lacking a subcase that are specially designated and justified, and are assessed `false`; references to external subcases and theories, which inherit any existing assessment for the top claim of the subcase or theory instantiation concerned, and are otherwise assessed as `unsupported`; and evidence nodes, which are rather more complicated and are discussed later.

The parent claims of interior argument blocks (i.e., concretion, substitution, decomposition, and calculation) are assessed `true` when all their subclaim(s) and their sideclaim (if any) are assessed `true`; otherwise they are `unsupported`. As explained above, the narrative justification provided with each argument node plays no part in the propagation of these assessments, which are concerned only with logical *validity*. The narrative justifications are critical in assessing *soundness* and developers are expected to ensure (and evaluators to confirm) that the narrative supplied for the argument node of each block justifies (indefeasible) belief in the parent claim, given its subclaims and any sideclaims. During development or, later, evaluation, any concern about the soundness of a narrative justification can be registered by attaching a defeater to the argument node concerned, which will then be factored into determination of validity.

Notice that these interior argument blocks represent premises that are *a priori*, meaning that we believe them by virtue of thinking about, and understanding, the system and the claims and the argument block concerned. This is in contrast to evidence incorporation blocks, which are *a posteriori* premises, meaning that our belief about them rests on observations or measurements of the system and its environment.

We now turn to assessment of evidence incorporation blocks. As with other argument blocks, for the purposes of logical validity we could just check that the evidence is available and assume that it is good (and will be challenged by a defeater if not) and assess the parent claim as `true`. However, evidence incorporation blocks are different to other blocks in that they are not solely focused on logical reasoning but provide the bridge between logical reasoning and the external world—which is manifested as evidence. Hence, we must choose how much of the evaluation of evidence we wish to factor into determination of logical validity, and how much into soundness. The choice made in Clarissa/asce is that validity considers only the

calculation. Hence, there can be no uncontested rule for propagation of doubt. The purpose of propagation is, firstly, to determine if an argument is valid and, secondly, to direct attention to those parts most likely in need of attention if it is not, and we consider that our approach accomplishes this in a manner that is effective and readily understood.

presence of the evidence, not its merit; interpretation and evaluation of evidence is the responsibility of argument blocks above the evidence incorporation block.

Specifically, the parent node of an evidence incorporation block is a claim that usually states "something measured" about the evidence (e.g., "the tests achieve MC/DC coverage"): that is, it indicates what the evidence *is*. Above that there is usually a substitution block whose parent claim states "something useful" derived from the evidence (e.g., "there is no unreachable code"): that is, it indicates what the evidence *means* (see [5, Sections 1.3 and 2.2]). In the interests of grammar, we will call these *evidentially measured* and *evidentially useful* claims, respectively.[8]

The narrative justification of the evidence incorporation node is expected to support (indefeasibly) the determination that the supplied evidence really does deliver the measured results (given any sideclaims). If the narrative justification for this is considered inadequate during development or evaluation, then a defeater can be attached to the evidence incorporation node. As with other argument blocks, the narrative justification plays no part in assessing the logical validity of an evidence incorporation block; thus, its evidentially measured claim is assessed `true` provided the evidence is present; otherwise it is `unsupported`.

Measured evidence is transformed into useful evidence by a substitution block placed above the measured claim of the evidence incorporation block. Usually, we expect the measured evidence to affirm the evidentially useful claim and we use *confirmation measures* to help make this determination in a principled way (confirmation measures are discussed in detail in our Confidence Report [5, Section 2.2]). An example is Good's confirmation measure $\log \frac{P(E \mid C)}{P(E \mid \neg C)}$, where E represents the evidence and C the evidentially useful claim. It is not necessary to apply these measures numerically: what matters are the concepts underlying them, so that in Good's measure we are asked to consider the likelihood of the evidence given the claim *vs.* its likelihood given the counterclaim. This is intended to ensure that the evidence not only supports the claim but that it discriminates between this claim and others (and the counterclaim in particular). Developers and evaluators can use these measures informally or can apply them to numerical or qualitative (e.g., low, medium, high) estimates of the subjective probabilities involved, if they find it useful to do so. Conceptually or numerically large positive confirmation measures indicate highly affirming evidence and this will be recorded and explained in the narrative justification of the substitution block, which can be challenged by a defeater in case of doubt. The usual rule for logical validity of a substitution block applies, so the

[8] Here, we are assuming the evidentially useful claim is supported by an argument block directly above the evidentially measured claim. It is possible that additional reasoning is required (e.g., adequacy of tools involved in generating the evidence) so that the useful claim is further removed from the measured claim; alternatively the additional reasoning could appear in a sideclaim to the substitution block relating the two evidential claims. Further practical experience is needed to develop recommendations for this topic.

parent (i.e., evidentially useful) claim will be assessed `true` provided the (evidentially measured) subclaim and any sideclaims are also `true`. Also as usual, the narrative justification, supported by a confirmation measure, is assumed to affirm soundness of this determination, and will be challenged by a defeater otherwise.

However, the usual calculations can be overridden—because evidence sometimes refutes a claim (e.g., when tests reveal a failure), which transforms it into *counter-evidence*. This is examined below, where we consider propagation of assessments in refutational arguments.

2.2 Propagation With Refutations

Note: the CLARISSA/ASCE implementation does not currently support refutational reasoning.

In refutational arguments, we need to consider the possibility that claims may be `false`, in addition to `true` or `unsupported` as considered above.[9] There are two ways that `false` assessments may be introduced into an argument: one is through counter-evidence, and the other is via defeaters; we begin with counter-evidence.

The discussion here is a continuation of that in the previous section and concerns the case where evidence does not merely fail to affirm its evidentially useful claim, but contradicts it. We considered affirming evidence in the previous section: this is evidence that delivers a strong positive confirmation measure and justifies the assessment that its evidentially useful claim is `true`; we now consider the other cases. Conceptual or numerical confirmation measures close to zero indicate weak evidence and in this case the evidentially useful claim will be assessed `unsupported`. But sometimes evidence contradicts the claim it is meant to support: for example, as noted previously, testing may reveal failures. We refer to this as *counter-evidence* and it should lead to a strongly negative confirmation measure. In this case, we assess the evidentially useful claim as `false`, provided its evidentially measured subclaim and any sideclaims are `true`; otherwise it is `unsupported`.

We next turn to analysis of defeaters. Since a defeater can point to any kind of node, we define the claim *affected by* the defeater to be the node pointed to if this is a claim or defeater, and otherwise the parent claim (which may be a defeater) of the node pointed to.

When the claim in a (non-exact) defeater is assessed `false` it means the defeater is refuted; hence, the main case (or subcase for lower-level defeaters) is exonerated and its claims are assessed as if the defeater were absent. Exact defeaters (those

[9] There is a subtle point concerning the interpretation of negative assessments, where `unsupported` can sometimes function like `false`. If a top claim assessed as `unsupported` states something like "the system is safe" then more work is needed to refine the case so that it delivers a definitive assessment (i.e., `true` or `false`). But if the claim is "*the argument establishes that* the system is safe" then `unsupported` carries stronger significance and its external interpretation will be the same as `false`.

that point directly to a claim or other defeater and whose claim is the negation of that pointed to) are a special case that is considered later, in Subsection 2.3.

When the claim in a defeater is assessed `unsupported` (which also applies when the defeater has no subcase—i.e., it is merely a doubt), then so is the claim affected by the defeater. And when the claim in the defeater is assessed `true`, then the affected claim is also assessed `unsupported`;[10] again, exact defeaters are a special case and will be considered later.

These assessments override the assessments due to any other nodes pointing to the affected claim (which may affirm it as `true`: when a claim is challenged by a `true` or `unsupported` defeater, we have to accept that it is unresolved. And this is true even if the defeater has no subcase (i.e., is a doubt); the mere existence of the doubt calls the affected claim into question. These assessments are also independent of the logical relationship between the affected claim and the claim in the defeater that challenges it (again, excluding exact defeaters): we have made a human judgement that the defeater does call the node pointed to (and hence the affected claim) into question, even if thee two claims are logically unrelated. As a result, there is some ambiguity here: when the claim affected by a defeater is assessed as `unsupported`, it could either be because the defeater's subcase has sustained the defeater or because that subcase is incomplete; we need to examine the assessment of the defeater (`true` or `unsupported`, respectively) to discriminate the two cases. In the latter case, the defeater's subcase needs more work, while in the former the main argument needs to be revised (and possibly also the system concerned).

Finally (apart from exact defeaters), we consider propagation of `false` assessments through the remaining kinds of argument blocks (i.e., concretion, substitution, decomposition, and calculation). In NLD, individual argument blocks of these kinds are intended to be deductively valid: that is, they are interpreted as material implications of the form

$$\texttt{sideclaim} \wedge \texttt{subclaims} \supset \texttt{parent claim} \tag{1}$$

where the subclaims, if there is more than one, are usually conjoined (we introduce disjunctive subclaims in Section 4).

The sideclaim and subclaims constitute the *antecedent* to this implication. As described in the previous section, when all claims in the antecedent are assessed `true` then, by the rules of classical logic, so is the parent claim. And if any antecedent claims are `unsupported`, then the parent claim is also. But suppose some claims in the antecedent are assessed `false`. Since they are conjoined, this means the whole antecedent is `false`; does this mean we should assess the parent claim as `false` too?

[10] It cannot be assessed `false` because the defeater may not precisely refute the affected claim (unless it is an exact defeater, which is considered later), but merely call it into question.

It does not: it would be attempting to derive $\neg A \supset \neg B$ from $A \supset B$, and this is the logical fallacy of "denying the antecedent" [25].[11] Moreover, there is a further problem: if the antecedent is `false`, then it can imply anything: this is the *false implies everything* problem.[12] Thus, in general, we cannot propagate `false` upward through these four kinds of assurance blocks;[13] we must do something weaker and the appropriate response is to assess the parent claim as `unsupported`.

2.3 Exact Defeaters

As noted above, exact defeaters are a special case; their purpose is to introduce negation into an assurance case and this primarily finds application in an alternative form of argumentation to be described in Section 3.

An exact defeater is one that: a) points to a node that is either a claim or another defeater that b) lacks a subcase, and c) whose own claim is the negation of the one pointed to. An example is shown in Figure 3; the example in Figure 1 cannot be an exact defeater, independently of its claim, because the claim it points to has a subcase, contradicting b) above.

Because claims in Clarissa/asce are written in natural language, it is not trivial to determine if one claim is the negation of another. Accordingly, Clarissa/asce provides an explicit selection in its interface to indicate that a defeater should be treated as the exact negation of the claim or defeater that it points to. Furthermore, the node pointed to may have a subcase, but it will be ignored (and indicated so in the graphical presentation) when the node becomes the target of an exact defeater.This is to support exploratory development of a case without having to undo or redo previous work.

The propagation rules for exact defeaters are simple: if the exact defeater is assessed `unsupported`, then so is the node that it points to; otherwise the assessment of the node pointed to is the logical negation of the assessment of the claim in the defeater.

[11] An informal illustration of denying the antecedent uses the subclaim/premise "if college admission is fair, then affirmative action is unnecessary" to fallaciously infer the claim/conclusion "college admission is not fair, so affirmative action is needed."

[12] $A \supset B$ is equivalent to (or is defined as) $\neg A \vee B$ so, if A is `false`, $\neg A$ is `true`, and $\neg A \vee B$ is `true` independently of B.

[13] There is a special case where the (conjunction of) subclaims is *equivalent* to the parent claim (given the sideclaim) rather than merely entailing it: it is legitimate to propagate `false` in this case but Clarissa/asce does not do so (because it does not attempt to interpret the language of claims). Instead, the case should be modified to use an exact defeater.

2.4 Summary of Propagation Rules

Here we present a summary of the propagation rules for truth assessments described in the previous subsections. Remember, we are using a three-valued logic: `true`, `false`, and `unsupported`.

Assumptions: assigned `true`

Unsupported claims (i.e., claims with no subcase): assigned `unsupported`

External subcases: parent claim inherits whatever the subcase delivers for its top claim, with default assignment `unsupported`

Evidence Incorporation: if evidence *present*
then (*evidentially measured*) parent claim is assigned `true`,
otherwise `unsupported` (see below for *evidentially useful* claims)

General assurance blocks (concretion, substitution, decomposition, and calculation): if all subclaims and sideclaim `true`,
then parent claim is assigned `true`, otherwise `unsupported`

Special case for substitution blocks delivering *evidentially useful* parent claims:
Justification should reference confirmation measures then, provided sideclaim and evidential measured subclaim are `true`,
if confirmation measure is `strongly positive`, then parent claim is `true`
confirmation measure is `neutral`, then parent claim is `unsupported`
confirmation measure is `strongly negative`, then parent claim is `false`
otherwise `unsupported`

Ordinary (non-exact) defeaters (includes doubts): if claim in defeater is `false`,
then rest of the case is unaffected (defeater is defeated),
otherwise *affected claim* is `unsupported`

Exact defeaters:
if claim in defeater is `false`, then parent claim is `true`
claim in defeater is `true`, then parent claim is `false`
claim in defeater is `unsupported`, then parent claim is `unsupported`

2.5 Logic Programming Interpretation

Our interest in this topic is motivated by the possibility that a suitable Logic Programming language could mechanize the propagation rules described in the previous subsections. Although these rules are easy to implement directly in CLARISSA/ASCE, translation to Logic Programming could allow additional automated exploration and analysis that provide added value.

As noted in formula (1), CLARISSA/ASCE argument blocks are interpreted as a material implication, which can be rewritten[14] as

$$\texttt{parent claim} \vee \neg\texttt{sideclaim} \vee \neg\texttt{subclaim}_1 \vee \cdots \vee \neg\texttt{subclaim}_n. \tag{2}$$

The claims are *ground terms* (i.e., they contain no variables) and a ground term or its negation is called a *literal*; a disjunction of literals such as (2) is a *clause*. A clause with exactly one positive (i.e., unnegated) term, again such as (2), is called a *definite clause*; one with with no positive terms is called a *goal*; and a single positive term is a *fact*; together, these constitute *Horn clauses*. The arguments of assurance cases can be represented as collections of Horn clauses: evidence and assumptions will be facts, and the other reasoning blocks will be definite clauses. Goals arise when we pose questions about the argument (e.g., "is the top claim `true`?").

Horn clauses are also the basis of *logic programming*: a collection of Horn clauses has an interpretation in logic (namely, the set of literals that must be `true` to satisfy all clauses in the collection) and another, operational, one that computes this set (the set will be empty if the clauses are *unsatisfiable*).

In Logic Programming, clause (2) above is usually written

$$\texttt{parent claim :- sideclaim, subclaim}_1\texttt{,..., subclaim}_n. \tag{3}$$

where the term on the left side of the `:-` symbol is called the *head*, and the list of terms on the right hand side is called the *body*. An example `fact` is written as

```
claim
```

and indicates that this literal is `true`.

There are several Logic Programming languages; examples include Prolog and Datalog and there are further variants within these. A significant source of variation is the treatment of negation and how the logic and operational interpretations are kept aligned under these different treatments.

In assurance cases, exact defeaters require strict, or logical (i.e., classical) negation because a `true` defeater claim (body) entails a `false` target claim (head) and vice-versa. Other defeaters are more complex; first, the affected claim will usually have an existing assessment derived from the primary case (i.e., ignoring the defeater) and a defeater can *override* this. In particular, the affected claim becomes `unsupported` if the defeater is `true` or `unsupported`, and is left alone if the defeater is `false`. This means we need a way to represent `unsupported` and its special rules in our logic program. Fortunately, these resemble "unproved" and "weak" negation, respectively, in certain forms of Logic Programming. In particular, the form of Logic Programming known as Answer Set Programming (ASP) supports both classical or

[14] Here we are replacing $A_1 \wedge \cdots \wedge A_n \supset B$ by $\neg(A_1 \wedge \cdots \wedge A_n) \vee B$, applying commutativity of $\vee$, and then using De Morgan's law to replace $\neg(A_1 \wedge \cdots \wedge A_n)$ by $\neg A_1 \vee \cdots \vee \neg A_n$.

strict negation (written as prefix `-`) and the autoepistemic (stable model semantics) interpretation of weak negation (negation as failure), written as prefix `NOT`. That is, `NOT p` is `true` if p is unproved, and `NOT -p` is `true` if `-p` (i.e., the strict negation of `p`) is unproved. Notice that `NOT p` and `NOT -p` may both be `true` (meaning that neither `p` nor its strict negation has been proved), whereas if `p` is `true` then `-p` must be `false`, and vice-versa.

CLARISSA/ASCE has a *Prolog Export* plugin that uses the s(CASP) system [23] for answer set programming to perform semantic analysis of assurance cases [14]. Our treatment of assurance cases with defeaters is interpreted in s(CASP) using the following translation rules, which are derived directly from those stated informally in the previous subsection. However, in the informal rules we implicitly assumed a "two pass" interpretation where, in the first pass we ignored ordinary (non-exact) defeaters while propagating truth values and then applied the effects of defeaters in the second pass. Here, we will translate the assurance argument together with its defeaters and submit it all to the s(CASP) engine, which will interpret it as a complete program. So we augment each rule with a `defeater` term that will be instantiated by the translation of the claim(s) in the actual defeater(s) pointing to this claim or its argument node (and be absent if there are none).

Assumptions: we simply state the `claim` as a fact, provided it is not defeated.

```
claim :- -defeater
```

It is worth examining the cases here. If the `defeater` is `false` or absent, the `claim` is `true`. If the `defeater` is `true`, then `-defeater` is `false`, but this does not make `claim` `false` (the body affects the head only when it is `true`). Finally, if the `defeater` is unproved, then so is the `claim`.

Unsupported claims (i.e., claims with no subcase): we say nothing about the claim; s(CASP) will treat it as `unproved`.

External subcases: the parent claim inherits whatever the subcase delivers for its top claim.

Evidence Incorporation:

```
measured_parent_claim :- evidence_present, -defeater
```

Here, the `evidence_present` flag is set `true` (i.e., stated as a fact) when the evidence is present and is not mentioned otherwise.

General assurance blocks (concretion, substitution, decomposition, and calculation):

```
parent_claim :- subclaims, sideclaims, -defeater
```

Here, `subclaims` is a list of one or more claims and `sideclaims` is also a list of zero or more.

Special case for substitution blocks delivering *evidentially useful* parent claims: Justification should reference confirmation measures, then `confirmation` can be set `true`, `unproved`, `false` (i.e., respectively stated as a fact, not mentioned, stated as a negated fact) according to whether the confirmation measure is strongly positive, neutral, or strongly negative.

If confirmation measure is strongly positive:

```
useful_parent_claim :- sideclaims, measured_claim, -defeater
```

If confirmation measure is strongly negative:

```
-useful_parent_claim :- sideclaims, measured_claim, -defeater
```

Note that this causes the evidentially useful parent claim to be set `false` when the evidence is strongly negative.

Ordinary (non-exact) defeaters (includes doubts):

if the defeater is unsupported (i.e., is a doubt), then it is not mentioned and will default to `unproved`. Otherwise it must be the parent node of some argument node and will be set according to its type using the rules above.

Exact defeaters:

```
 parent_claim :- -defeater_claim
-parent_claim :- defeater_claim
```

Note that we need two rules here: one to propagate `true` and the other to propagate `false`.

2.6 Comparison with Dialectics in GSN

The Goal Structuring Notation (GSN) has a "Dialectic Extension" [1, Section 1:6] that serves purposes and provides capabilities very similar to our defeaters. Whereas we have nodes explicitly marked as defeaters that can point to other nodes, in GSN a "dialectic challenge" is indicated by a special kind of link (i.e., arrow) that can point to other links as well as to nodes. (Links in GSN can indicate "supported by"

or "in context of" relationships, whereas arrows in Assurance 2.0 merely identify the nodes that constitute a block and it suffices to point defeaters at nodes.) GSN does not explicitly distinguish exploratory and exact defeaters, but informally it can accomplish equivalent effects.

GSN does not make the strong distinction between logical validity and indefeasible soundness that is a focus of Assurance 2.0, nor does it perform refutational reasoning, so argument nodes are considered `true` or `false` (i.e., refuted), without the additional `unsupported` valuation of Clarissa/asce. A refuted goal or link is indicated by a large X. Furthermore, propagation of defeat is performed informally and regarded as "a matter for expert judgement and the more likely outcome is that the original goal structure is refactored at this point" [1, Section 2:11.3.4]. Similarly, retention of defeaters and "how to present dialectic argument in the final goal structure" are regarded as "choices to be made by the practitioner" [1, Section 2:11.4/6].

The dialectic extension for GSN assurance cases is supported by the commercial version of asce while a more complex form of defeasible reasoning [5, Section 5.1.4] is supported by the Astah GSN tool [2,21]. Naturally, we consider that the treatment of defeaters in Assurance 2.0 and their implementation in Clarissa/asce strike the best balance of utility and rigor.

3 Eliminative Argumentation and Exact Defeaters

"Eliminative Induction" is a method of reasoning that dates back to Francis Bacon who, in 1620 [3], proposed it as a way to establish a scientific theory by refuting all the reasons why it might be false (i.e., its defeaters).[15] Weinstock, Goodenough, and Klein [7] build on the idea of Eliminative Induction to develop a means of assurance that they call *Eliminative Argumentation*. Here, instead of attempting to confirm a positive claim such as "the system is safe" we instead attempt to refute the negative claim "the system is *un*safe." A successful refutation will establish the negation of that claim, namely "the system is *not* unsafe." In classical logic this is equivalent to establishing the positive claim by virtue of the rule for elimination of double negation (recall Footnote 6), and thereby provides the desired assurance. Diemert and Joyce [6] and others [13] report successful application of eliminative argumentation in assurance of real systems

The methodology of Assurance 2.0 favors *positive cases* where a constructive argument is developed in support of some beneficial claim about a system. Of course, in a formal or regulated safety process, there will generally be several layers of review and challenge to the case that should eliminate flaws and undue optimism;

[15] This was a precursor to modern methods and theories of science, which see falsifiability as the key characteristic [15]; however, they can be related via Bayesian Epistemology [9,24].

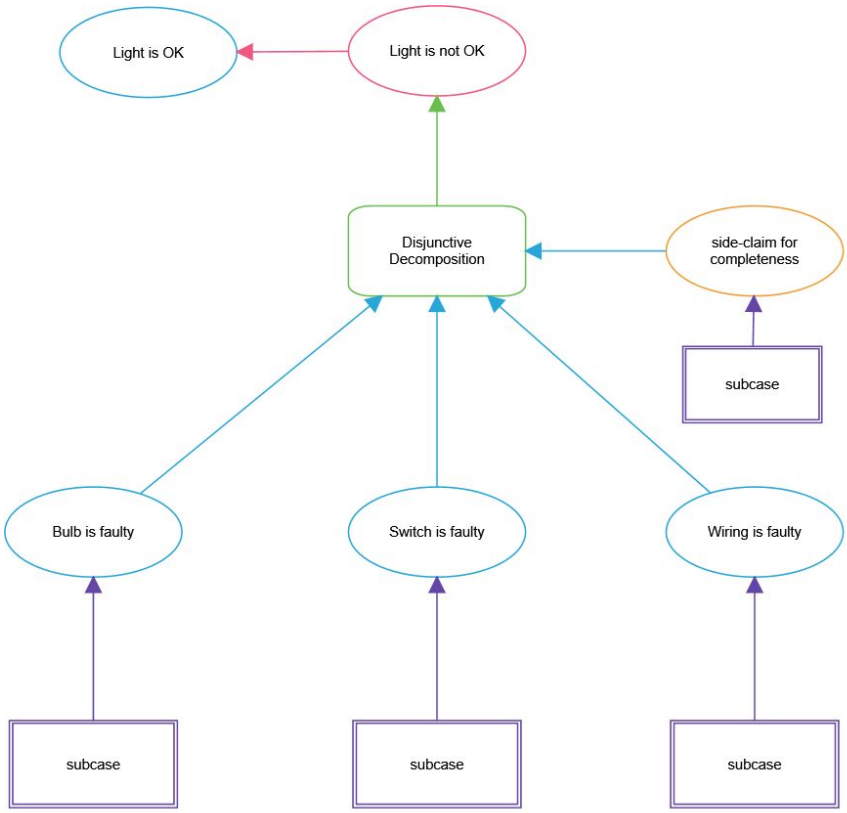

Figure 3: Eliminative Argument Using an Exact Defeater

nonetheless, to combat confirmation bias and other complacency, we accept that it can be useful to explicitly consider different points of view using contrary claims and negative arguments. Furthermore, when attempting to refute a defeater to a positive case, we are in a context similar to eliminative argumentation and therefore need to support this kind of reasoning for our own purposes.

In the framework of Assurance 2.0, eliminative argumentation can be represented by attaching an exact defeater to a positive claim, and then attempting to refute it. We sometimes refer to this use of defeaters as *eliminative* and to the conventional use as *exploratory*. Notice the whereas exploratory defeaters *augment* the main argument, providing an exploratory investigation or commentary, eliminative or exact defeaters are used as a reasoning step *within* the main argument.

An eliminative argument is shown in Figure 3 where we provide assurance that a newly installed electric light is OK by introducing an exact defeater that asserts it is not OK (i.e., faulty) and then refuting all the reasons that could make it so. Notice that this uses a disjunctive decomposition, which will be explained in the following

section. See Figure 4 for a more elaborated development of a traditional argument for the same example, which has a first-level exploratory defeater in the upper right and an exact defeater (with a disjunctive decomposition) at the second-level in the lower center.

4 Conjunctive and Disjunctive Decomposition Blocks

We stated earlier that the subclaims of decomposition blocks are treated as a conjunction and we might wonder if this is always appropriate. For example, suppose we are reasoning about a fault tolerant system S that has two redundant subsystems A and B and it is sufficient for safety that either one of these is working. It might seem that we should have a substitution block with parent claim "S is safe" and subclaim "A is working correctly or B is working correctly," which would seem to invite the support of a decomposition block where the subclaims are disjoined instead of conjoined. But this is not correct: it would allow us to provide a subargument that considers only the case where A is working correctly—but we do not know in advance whether it is A or B that will be working correctly. We also need to consider the case where both are working correctly (because they might get in each other's way). A valid argument is a standard (conjoined) decomposition block with three subclaims "A and B are both working correctly," "A alone is working correctly," and "B alone is working correctly." Another candidate for disjunctive decomposition is when we have two (or more) alternative arguments in support of a given claim, so that if the assessors do not like Argument 1, they can consider Argument 2. But, again, if these are disjoined, we can establish the parent claim by developing only Argument 1, thereby subverting the motivation for having alternative cases. So, once again, we should interpret decomposition as a conjunction.

There is, however, one situation where disjunctive decomposition can be appropriate: this is when the base assurance case is a generic template (i.e., part of what we call a *theory*) that may be instantiated in several ways. For example, we may have a claim P that can be ensured either by subcase Q or subcase R, depending on the realization chosen in the actual system. Here, we can use a disjunctive decomposition (which needs no sideclaim) in the generic template and elaborate only the appropriate subcase. We give an example in Section 5, where the claim that a light bulb will last a certain time can be satisfied either by using an LED, or by using an incandescent bulb that comes with a label "guaranteeing" adequate life.

In our experience, and apart from their use in theories and templates, the appropriate interpretation for the subclaims to a decomposition is always conjunction—at least in positive cases. But what about negative cases: those where the local goal is to refute some higher claim? Here, applying eliminative argumentation to system S above, we might have the counter-claim "S is unsafe" and we support this with the three subclaims "A is not working safely," "B is not working safely," and "neither

is working safely." But any one of these is sufficient for S to be unsafe, so this step of the argument needs to be represented by a decomposition block in which the subclaims are disjoined. An explanation for this can be seen by supposing the original, positive case can be represented as

$$A_{ok} \wedge B_{ok} \wedge AB_{ok} \supset S_{ok}.$$

Then from this we postulate a negative case[16]

$$\neg(A_{ok} \wedge B_{ok} \wedge AB_{ok}) \supset \neg S_{ok},$$

which is equivalent (by De Morgan's law) to

$$\neg A_{ok} \vee \neg B_{ok} \vee \neg AB_{ok} \supset \neg S_{ok}.$$

We conclude that in negative cases it can be useful to have a disjunctive form of decomposition block. One way to do this is to maintain just a single kind of decomposition block, but to interpret it conjunctively in positive subcases and disjunctively in negative ones. Alternatively, we can introduce an explicitly disjunctive form of decomposition, and this is the approach employed in Clarissa/asce. We prefer this choice because it requires an active decision by the developers of the assurance case and explicitly records it for the assessors. Furthermore, the disjunctive form is available for use in positive cases (and the conjunctive form in negative cases) should this ever be considered appropriate.

The propagation of truth values over a disjunctive decomposition requires that if the sideclaim and *any* subclaim are `true`, then the parent claim is also `true`; otherwise it is `unsupported`.

5 An Illustrative Example

The general propagation rules and the use of conjunctive and disjunctive decomposition blocks, as described in the previous sections, are illustrated in Figure 4. Here, we suppose than an electrician has installed a new light and we want assurance that it works correctly. On the left we have a positive case that decomposes conjunctively into three subcases concerning whether the light bulb, the switch, and the wiring are OK. For brevity, we omit concretion and other steps needed to properly connect the top claim to this decomposition. We suppose that these subcases all support `true` subclaims. A sideclaim must establish that these are the only cases that need to be considered and we suppose that its subcase is also assessed `true`.

[16] Remember, this is invalid in general (it is denying the antecedent), and therefore needs careful justification; it is valid if the implication can be strengthened to equivalence (i.e., to *if and only if*).

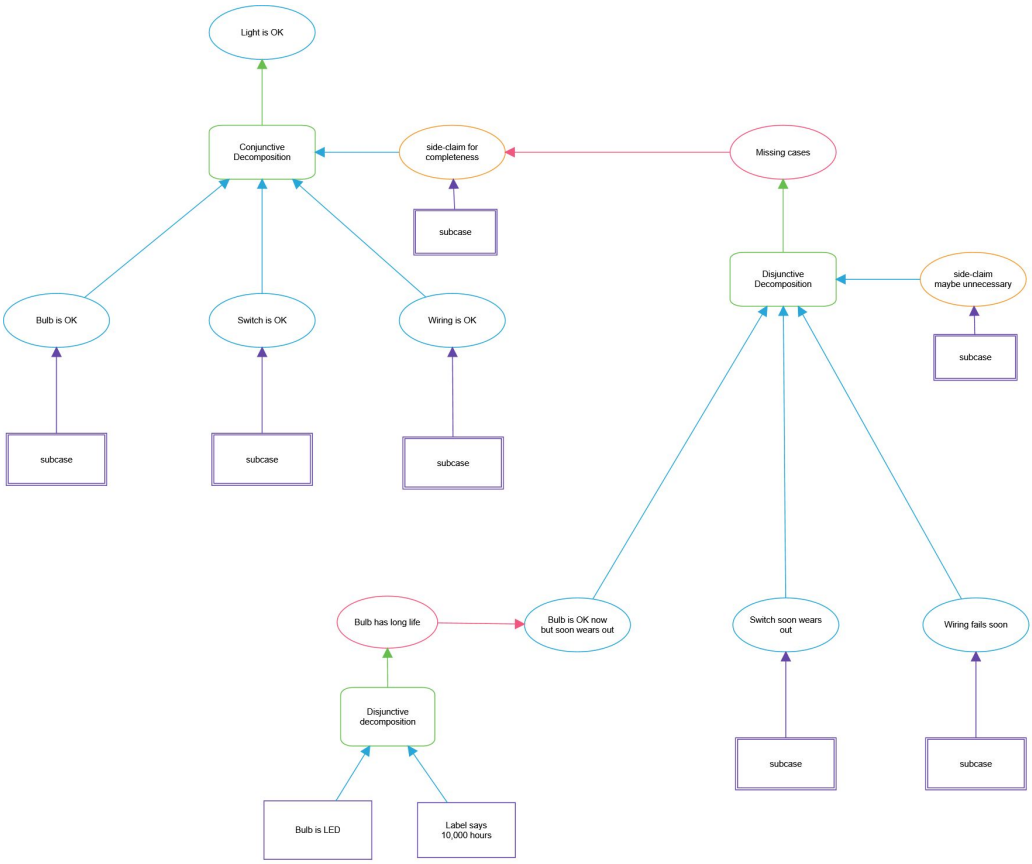

Figure 4: Two Levels of Defeaters: Exploratory Top Right, Exact Bottom Center

Then, provided the narrative justification for the decomposition is judged credible, the top claim can be assessed `true`.

But on the right, we add an exploratory defeater that introduces a negative subcase challenging the sideclaim by asserting that the three cases are insufficient.[17] It would be sufficient to sustain this by an assumption that states "working now is not enough: we need to be sure it will continue to do so, for some specified minimum time." Instead, for the purposes of illustration we suppose that this defeater is supported by consideration of the possibilities that any of the bulb, switch, and wiring are OK now, but may soon fail. This is represented by a disjunctive decomposition:

[17] In a more fully developed case, the defeater would probably challenge a missing concretion step where, "OK" is implicitly concreted as "OK now" but should surely also include the expectation that it will continue to be OK for the near future.

if any of these can be sustained, then the defeater is `true` and the affected sideclaim (and hence the top claim also) becomes `unsupported`.

The failure subcases for the switch and wiring are abstracted, but that for the bulb is developed. It seems that only "long life" bulbs will be used in this installation, so we can refute the claim that the bulb soon wears out; we do this with an eliminative argument using an exact defeater that claims the bulb has a long life. (The narrative justification for this block should explain why "Bulb has long life" is the negation of "Bulb is OK now but wears out," possibly by refining these claims.) As a second-level defeater, this introduces a positive subcase and so we expect any decompositions below it to be conjunctive. However, what we wish to assert is that the bulb used will be new (but "burned in") and either an LED, or one with a label claiming it is good for 10,000 hours. Both of these ensure long life and so we use them as subclaims to a disjunctive decomposition. As explained in Section 4, this is the only situation we know of where a disjunctive decomposition is appropriate in a positive case. The idea is that we wish to provide assurance for a parameterized system and we disjunctively decompose the argument into subcases according to different possible instantiations: the argument serves as a template and in any particular instantiation only one subcase will be used.

Here we may suppose that the evidence indicates we have an LED bulb and we will not use the subcase that requires evidence from the label. Note that these applications of evidence are abbreviated in the interests of concision and simplicity; usually, an evidence incorporation block supports an evidentially measured claim and then a substitution block is used to lift this to an evidentially useful claim.

Because the evidence indicates the bulb is an LED, the corresponding evidential node is `true` and this propagates through the lower disjunctive decomposition to make the exact defeater's subclaim `true`—and this makes its parent claim `false`. This claim (i.e., "bulb is OK now but wears out") is a subclaim to the upper disjunctive decomposition. As the other subclaims to this decomposition have undeveloped subcases, they are assessed `unsupported` and the parent claim to the decomposition is assessed `unsupported` also (since none of its subclaims are `true`). This parent claim is that in the first-level defeater and so it propagates as `unsupported` also and causes the defeater's target (the sideclaim to the conjunctive decomposition) to become `unsupported` and hence the top claim also. Thus, the act of investigating this defeater has alerted us that future failure is a valid concern, and so we will expand the positive case to reflect this. We can then mark the defeater as `addressed`

Note that we developed the subcase to the first, exploratory defeater in a particular way in order to illustrate exact defeaters, disjunctive decompositions, and refutational reasoning. In normal practice, once we had recognized the need to consider future failures, we would use an assumption "light could fail in the near future" to sustain the defeater, leading to immediate revision of the case.

The revised case might add a concretion block below the top claim to define what it means for the light to be OK and this might say that it works now and can be expected to continue to do so for some specified time in the future, given a specified pattern of use, and "expected" might be further concreted into some probabilistic claim. The subcase below that might then decompose into "now" and "future" branches and the subcase to the exploratory defeater, suitably adjusted (e.g., it becomes a positive case so its disjunctive decomposition is changed to conjunctive), will become the subcase to the "future" branch.

6 Defeaters and Confidence

Assurance 2.0 provides two complementary ways for assessing an assurance case argument and Clarissa/asce provides automated support for these. The first, definitive, assessment is logical indefeasibility; this requires that the argument is logically valid, the narrative justifications provide good reasons why each argument block is sound, and there are no credible reasons that would cause us to revise these judgments. We do not use checkmarks or other annotations to record the judgement that a given block is indefeasibly sound; instead, we assume this is so and use defeaters to indicate dissent. Automated validity checking using the methods developed in the previous sections takes defeaters, and thereby soundness assessments, into account and indicates the (in)completeness of the overall argument and the status of each claim.

It is worth recapitulating our terminology for the various stages of (in)completeness that an assurance argument may occupy.

Valid: this is the standard judgment from logic. It means that each claim is supported by an argument block whose subclaims and sideclaims are likewise supported by further argument blocks, ultimately terminating in assumptions, evidence, or defeaters that have been accepted as residual risks.

The graphical interface of Clarissa/asce is intended to eliminate many forms of invalidity "by construction" and thereby to encourage the development of arguments that will be valid when completed.

Validity does not depend on the meaning of the claims in an argument: it is solely concerned with the "form" of the argument. Soundness takes the meaning into account.

Sound: again, this judgment comes from logic. It means that each argument block has a narrative justification that is considered (by a human developer or evaluator) to establish that its evidence or subclaims truly entail the parent claim, given the sideclaim (if any). For the parent claims of reasoning (i.e., interior)

steps the entailment should be deductive (i.e., have no "gaps"), and for evidentially useful claims it is an epistemic judgment that should be supported by informal or explicit confirmation measures applied to the evidence. Likewise, the credibility of assumptions must be supported by adequate justifications, and defeaters accepted as residual risks must be supported by narrative justifications that the risk they pose is tolerable (considering both likelihood and impact).

Again, Assurance 2.0 and its support by Clarissa/asce are intended to encourage the development of sound arguments "by construction": that is why it has sideclaims, confirmation measures, and a limited selection of blocks.

Deductively valid: soundness sets a high bar; a useful intermediate step between validity and soundness focuses on whether the interior argument blocks are *deductive*: that is whether the subclaims truly entail the parent claim, given the sideclaim (if any). If an argument block is not deductive, it means (at best) that the subclaims "strongly suggest" the parent claim, and might entail it given some additional, but presumably unknown, information. Hence, there are "gaps" between the subclaims that should be filled by identifying and supplying this absent information (or else the argument block is completely unsound). Thus, a deductively valid argument is one that has no "gaps" and a sound argument is a deductively valid argument that has withstood even stronger scrutiny, focusing on the details and credibility of its narrative justifications and confirmation measures. An alternative to supplying missing information is to identify it as a residual risk and justify why it is acceptable.

Indefeasibly sound: this judgment comes from epistemology. It means that our judgement of soundness must be so strong that no credible new information would change it.

Open: doubts about an assurance argument are indicated by attaching (exploratory) defeaters. Defeaters that are themselves defeated (i.e., are refuted by a subargument) are said to be "retired," meaning they no longer play a part in the assurance argument, but may be retained as commentary. Unrefuted or true defeaters may be explicitly accepted and justified as residual risks; otherwise they are said to be "active." An assurance argument that has active defeaters or is incomplete (e.g., has disconnected subarguments or has unsupported claims) is said to be "open," otherwise it is "closed."

The validity plugin of Clarissa/asce checks the validity of closed arguments and provides helpful feedback on open ones.

The use of defeaters and refutational reasoning within validity assessment supports a cooperative balance between human judgment and automation in assessment

of soundness and indefeasibly: formulating defeaters, and exploring their subcases, expresses human judgment about local soundness, while validity assessment incorporating refutational reasoning integrates these judgments over the full case.

The second form of assessment for assurance case arguments concerns confidence and, in its strict form, this applies only to arguments that have already been assessed as indefeasible, or at least sound. Now, it is reasonable to ask how we can have anything less than full confidence in an argument that is indefeasibly sound. The explanation is that our judgements of soundness—that is, whether each evidential block supports its evidentially useful claim and whether the subclaims of each reasoning block indefeasibly entail their parent claim (given any sideclaims)—are human judgments, and confidence, quantified as a subjective probability, indicates our *degree of belief* in that judgement, which may be less than total even for indefeasible claims (e.g., the judgment of indefeasibility may rest on hazard analysis, which can never be declared perfect).

We want to assess confidence compositionally: that is, confidence in a (sub)case should be calculated from confidence in its parts and this requires that confidence is represented numerically, so that we can do arithmetic with it.[18] The natural way to do this is to represent confidence as a subjective probability.

For evidence, confidence is derived from the quantitative or qualitative assessments used in calculating confirmation measures for the evidentially useful claim: usually, $P(C \mid E)$ is used as the confidence measure, that is, the posterior likelihood of the evidentially useful claim, given the evidence. For other blocks, the default assessment is derived by a "sum of doubts" calculation, where doubt is the dual of confidence (i.e., $\mathit{doubt} = 1 - \mathit{confidence}$), so the doubt in a parent claim is given by the sum of the doubts in its subclaims and sideclaim. This default assessment can be modified by the user to reflect human judgment. The Clarissa/asce confidence plugin propagates these human and automated local assessments throughout the case.

Confidence assessments of completed cases are not used to deliver judgment on a case (that is the rôle of indefeasible soundness) but to help ensure balanced effort across a case and to support graduated assessments. Graduated assessments assist in trading confidence for cost in applications considered to pose less risk, as exemplified by the Design Assurance Levels (DALs) of DO-178C [16] and System Integrity Levels (SILs) of IEC 61508 [10]. Cost can be reduced by simplifying the system and thereby its assurance case (e.g., providing less fault checking or redundancy), and/or by weakening the assurance case (e.g., providing less evidence, or less costly evidence). It is also possible that system-level assurance cases may employ different levels of confidence in different parts of the case. For example, an architectural framework that enforces partitioning [17] may require the highest level of confidence

[18] We could use some ordinal scale, such as low medium, and high, or even acceptable and unacceptable, and then devise rules for their combination, but there seems little merit in doing so.

in its assurance, while lesser levels of confidence may be adequate for some of its partitioned applications (because partitioning limits fault propagation).

Logical validity and confidence are evaluated in CLARISSA/ASCE by separate plugins and both of these can decorate the graphical representation of the case to indicate their assessments. Logical validity uses the propagation rules of Section 2 (as augmented by Section 4 if disjunctive decompositions are present). Confidence is assessed by the methods sketched above and described in detail in our Confidence Report [5, Section 3].

Strictly, both logical and confidence assessments apply only to closed cases: that is, those in which there are no unsupported claims nor unresolved doubts nor active exploratory defeaters. Nonetheless, it can be helpful to calculate approximations to these assessments for open cases during construction or assessment (e.g., when assessors have used defeaters to flag or explore doubtful elements of a case), either to estimate progress or to focus attention, and the CLARISSA/ASCE plugins can do this.

We suggest using CLARISSA/ASCE and its plugins to support these activities as follows. Logical validity should be checked frequently during development; in the early stages, it is expected that most parts of the argument will be assessed `unsupported`, with only some areas assessed `true`. Temporary assumptions can be used for the purpose of exploration: for example, in the absence of some evidence we could assert the intended evidentially useful claim as an assumption in order to check that a valid case can be built above it. Negative assumptions (i.e., assessments of `false`) can be achieved by adding an exact defeater above a positive assumption. Narrative justifications can be supplied either as the argument is developed, or once a valid skeleton for it is in place. Deductiveness, soundness, and indefeasibility can then be considered, either serially or all together, and any doubts about these can be recorded and explored by adding defeaters to the argument and exploring their impact using the validity checker.

Confidence assessments are typically propagated upward from evidence and assumptions using the sum of doubts calculation, and are performed only when the argument is moderately complete (i.e., has no disconnected subarguments nor unsupported claims). Confidence values can be adjusted by the human user: the purpose of in-progress confidence assessments is not to deliver judgement but to assist allocation of effort and to direct attention to those parts of an assurance argument most in need of attention.

7 Summary and Conclusion

We have introduced two ways of using defeaters in Assurance 2.0 [4] and its CLARISSA/ASCE toolset [22]. Ordinary, or *exploratory* defeaters provide a way of recording and exploring doubts about an assurance case argument or its narrative

justifications and can be retained as commentary to assist future developers and evaluators who may have similar doubts. Exact or *eliminative* defeaters introduce negation into an argument: that is, instead of providing a subcase sustaining claim A, we instead attempt to provide one that refutes $\neg A$. Both kinds of defeater help challenge confirmation bias by inviting consideration of a contrary point of view in the manner of a dialectic or Socratic dialog. Exact defeaters allow an assurance case or subcase to proceed by *eliminative argumentation*, which some find more persuasive than a conventional, positive assurance case.

By a *positive* assurance case or subcase we mean one that employs an argument where the assurance goal is to *sustain* the local top (sub)claim (i.e., to assess it as `true`); the alternative is a *negative* case, where the goal is to *refute* the local top subclaim (i.e., to assess it as `false`). The subcase to any first-level defeaters of a positive case will be negative and vice-versa. Defeaters may appear in the subcases of other defeaters so these assessments can alternate. Thus, a subargument is positive if it is under an even (or zero) number of defeaters and negative if it is under an odd number.

The claims of an assurance argument may be assessed as `true`, `false`, or `unsupported` and these assessments propagate in suitable ways upward from the leaf nodes of the argument (i.e., from evidence, assumptions, and residual risks). Defeaters must be treated suitably and, in particular, care is needed when propagating `false` as this can introduce the fallacy of *denying the antecedent*. These concerns, and their correct treatment in Clarissa/asce, are described in detail in Section 2.2.

Truth assignments to claims propagate the same in positive and negative cases and there is no difference in interpretation or meaning between positive and negative cases, it is just that we usually try to sustain the former and refute the latter. In a finished assurance case argument, all positive cases should be sustained and all negative ones refuted (or, exceptionally, accepted as *residual risks* [5, Section 6]) as otherwise some defeaters must still be unresolved (i.e., the case is still *open*).

There is one difference between positive and negative arguments: subclaims to decomposition blocks are typically conjoined in positive arguments, but disjoined in negative ones. We introduce disjunctive decomposition blocks to Clarissa/asce so that this choice can be represented explicitly. This also allows disjunctive decompositions to be used in positive arguments (and conjunctive in negative ones) and Section 4 examined the (rare) circumstances where this can be useful.

When the claim of an exploratory defeater is refuted, it means that the doubt that motivated it is annulled and the original case stands; however, the defeater and its subargument can be retained as commentary. If refutation is unsuccessful, then we can attempt instead to sustain the defeater's claim (this is an instance where we attempt to sustain a negative subcase and will generally require adjustment to the subcase). If this succeeds, it means that the doubt is justified and the primary

argument, and possibly the system itself, must be revised. Once that has been accomplished, it should be possible to refute the defeater and to retain it and its (likely once again adjusted) subcase as commentary.

The claim in an *exact* defeater must be the negation of the claim or defeater that it points to, so that refutation of the defeater claim sustains the target claim and vice-versa. Because claims expressed in natural language can be hard to analyze (to see if one is the negation of another), users can direct CLARISSA/ASCE to mark defeaters as exact, in which case the subclaim and parent claim are assumed to be negations of each other, and any existing subcase for the affected claim is temporarily ignored. Those who favor eliminative argumentation can employ it by simply introducing an exact defeater near the top of the argument, while those who do not approve this style of argumentation can eschew this construction.

Refutational arguments, as introduced by defeaters, invite a different perspective than conventional positive arguments and can be valuable in challenging confirmation bias and other forms of complacency in the construction of assurance cases. However, the finished assurance case delivered to evaluators must be indefeasibly sound (modulo residual risks that are documented and explicitly accepted), so all exploratory defeaters must have been refuted and no longer play an active part in the argument. Nonetheless, consideration of refuted defeaters can greatly assist evaluators (and future (re)developers) to comprehend a case, and the CLARISSA/ASCE tool for Assurance 2.0 can therefore retain them and their subcases in completed cases, where they function as a kind of commentary. To avoid cluttering the main argument, and also to accommodate those who consider that evaluators should review a finished argument independently of its developers, CLARISSA/ASCE can selectively hide or reveal defeaters and their subcases.

Evaluation of an assurance case and, on that basis, authorizing deployment of the system concerned, are topics of delicate judgement. It is not the task of the evaluators to repeat the work of the assurance case developers, but they must form some judgement about it and there is contention whether defeaters should play a part in this. Our colleague Shankar says: "an assurance argument is a brief, not a debate" but, on the other hand, evaluators may want to know that developers have considered contrary points of view, and they may have doubts of their own and will want to see if these were considered and how they were resolved. We refer to this process of review and assessment as the "case about the case" or *metacase* and are preparing a report on the topic. We are also preparing a report on systematic methods of searching for potential defeaters.

In summary, defeaters and the refutational (sub)arguments that they introduce are vital tools in the construction of sound and persuasive assurance case arguments, and potentially also in their assessment. However, their use and assessment (particularly at multiple levels) is not straightforward, and we hope that this re-

port has adequately explained and justified their treatment in Assurance 2.0 and its CLARISSA/ASCE tool.

Acknowledgments. The work described here was developed in partnership with other members of the CLARISSA project, notably Srivatsan Varadarajan, Anitha Murugesan, and Isaac Hong Wong of Honeywell, Gopal Gupta of UT Dallas, and Robert Stroud of Adelard.

This material is based upon work performed under subcontract to Honeywell supported by the Air Force Research Laboratory (AFRL) and DARPA under Contract No. FA8750-20-C-0512. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Air Force Research Laboratory (AFRL) and DARPA.

References

[1] *Goal Structuring Notation Community Standard Version 3*. The Assurance Case Working Group, York, UK, May 2021.

[2] Astah. *Astah GSN home page*. http://astah.net/editions/gsn.

[3] Francis Bacon. *The Novum Organon: Or, A True Guide to the Interpretation of Nature*. Oxford University Press, 1855. English translation by G. W. Kitchin; the original Latin is from 1620.

[4] Robin Bloomfield and John Rushby. Assurance 2.0: A Manifesto. In Mike Parsons and Mark Nicholson, editors, *Systems and Covid-19: Proceedings of the 29th Safety-Critical Systems Symposium (SSS'21)*, pages 85–108, Safety-Critical Systems Club, York, UK, February 2021. Preprint available as arXiv:2004.10474.

[5] Robin Bloomfield and John Rushby. Confidence in Assurance 2.0. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, November 2021, updated May 2024. Also available as arXiv:2205.04522

[6] Simon Diemert and Jeff Joyce. Eliminative argumentation for arguing system safety—a practitioner's experience. In *IEEE Systems Conference*, 2020.

[7] John B. Goodenough, Charles B. Weinstock, and Ari Z. Klein. Eliminative induction: A basis for arguing system confidence. In *Proceedings International Conference on Software Engineering, New Ideas and Emerging Results*, pages 1161–1164, IEEE Computer Society, San Francisco, CA, May 2013.

[8] Richard Hawkins and Philippa Ryan Conmy. Identifying run-time monitoring requirements for autonomous systems through the analysis of safety arguments. In *Computer Safety, Reliability, and Security (*SafeComp *2023)*, Volume 14181 of Springer *Lecture Notes in Computer Science*, pages 11–24, Springer, Toulouse, France, September 2023.

[9] James Hawthorne. Bayesian induction IS eliminative induction. *Philosophical Topics*, 21(1):99–138, 1993.

[10] *IEC 61508—Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems.* International Electrotechnical Commission, Geneva, Switzerland, March 2004. Seven volumes; see http://www.iec.ch/zone/fsafety/fsafety_entry.htm.

[11] Imre Lakatos. *Proofs and Refutations.* Cambridge University Press, Cambridge, England, 1976.

[12] Bev Littlewood and John Rushby. Reasoning about the reliability of diverse two-channel systems in which one channel is "possibly perfect". *IEEE Transactions on Software Engineering*, 38(5):1178–1194, September/October 2012.

[13] Laure Millet et al. Assurance case arguments in the large: The CERN LHC machine protection system. In *Computer Safety, Reliability, and Security (*SafeComp *2023)*, Volume 14181 of Springer-Verlag *Lecture Notes in Computer Science*, pages 3–10, Springer-Verlag, Toulouse, France, September 2023.

[14] Anitha Murugesan, Isaac Hong Wong, Robert Stroud, Joaquín Arias, Elmer Salazar, Gopal Gupta, Robin Bloomfield, Srivatsan Varadarajan, and John Rushby. Semantic analysis of assurance cases using s(CASP). In *International Conference on Logic Programming 2023 Workshops: Goal-Directed Execution of Answer Set Programs (GDE)*, Volume 3437 of *CEUR Workshop Proceedings*, London, UK, July 2023.

[15] Karl Popper. *The Logic of Scientific Discovery.* Routledge, 2014. First published in German 1934, English 1959.

[16] RTCA. *DO-178C: Software Considerations in Airborne Systems and Equipment Certification.* Requirements and Technical Concepts for Aviation (RTCA), Washington, DC, December 2011.

[17] John Rushby. Partitioning for avionics architectures: Requirements, mechanisms, and assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999. Available at https://www.csl.sri.com/~rushby/abstracts/partitioning.

[18] John Rushby. Runtime certification. In Martin Leucker, editor, *Eighth Workshop on Runtime Verification: RV08*, Volume 5289 of Springer-Verlag *Lecture Notes in Computer Science*, pages 21–35, Springer-Verlag, Budapest, Hungary, April 2008.

[19] John Rushby. The interpretation and evaluation of assurance cases. Technical Report SRI-CSL-15-01, Computer Science Laboratory, SRI International, Menlo Park, CA, July 2015. Available at http://www.csl.sri.com/users/rushby/papers/sri-csl-15-1-assurance-cases.pdf.

[20] John Rushby. The indefeasibility criterion for assurance cases. In Yamine Ait-Ameur, Shin Nakajima, and Dominique Méry, editors, *Implicit and Explicit Semantics Integration in Proof Based Developments of Discrete Systems*, Communications of NII Shonan Meetings, pages 259–279, Springer, Kanagawa, Japan, July 2020. Postproceedings of a workshop held in November 2016.

[21] Toshinori Takai and Hiroyuki Kido. A supplemental notation of GSN to deal with changes of assurance cases. In *4th International Workshop on Open Systems Dependability (WOSD)*, pages 461–466, IEEE International Symposium on Software Reliability Engineering Workshops, Naples, Italy, November 2014.

[22] Srivatsan Varadarajan, Robin Bloomfield, John Rushby, Gopal Gupta, Anitha Murugesan, Robert Stroud, Kateryna Netkachova, and Isaac Hong Wong. Clarissa: Foundations, tools and automation for assurance cases. In *42nd AIAA/IEEE Digital Avionics Systems Conference*, The Institute of Electrical and Electronics Engineers, Barcelona, Spain, October 2023.

[23] Sarat Chandra Varanasi, Joaquín Arias, Elmer Salazar, Fang Li, Kinjal Basu, and Gopal Gupta. Modeling and verification of real-time systems with the event calculus and s(CASP). In *Practical Aspects of Declarative Languages: 24th International Symposium (PADL 2022)*, pages 181–190, Springer, Philadelphia, PA, January 2022.

[24] Susan Vineberg. Eliminative induction and Bayesian confirmation theory. *Canadian Journal of Philosophy*, 26(2):257–266, 1996.

[25] *Denying the Antecedent*. Wikipedia. https://en.wikipedia.org/wiki/Denying_the_antecedent.

[26] *Dialectic*. Wikipedia. https://en.wikipedia.org/wiki/Dialectic.