Certus: A Domain Specific Language for Confidence Assessment in Assurance Cases

Simon Diemert – Critical Systems Labs Inc.
Jens H. Weber – University of Victoria

WWW.CRITICALSYSTEMSLABS.COM

Certus: A domain specific language for confidence assessment in assurance cases

Simon Diemert[1,2][0000−0001−9493−7969] and Jens H. Weber[1][0000−0003−4591−6728]

[1] University of Victoria, Victoria, Canada
[2] Critical Systems Labs Inc., Vancouver, Canada

Abstract. Assurance cases (ACs) are prepared to argue that a system has satisfied critical quality attributes. Many methods exist to assess confidence in ACs, including quantitative methods that represent confidence numerically. While quantitative methods are attractive in principle, existing methods suffer from issues related to interpretation, subjectivity, scalability, dialectic reasoning, and trustworthiness, which have limited their adoption. This paper introduces *Certus*, a domain specific language for quantitative confidence assessment. In *Certus*, users describe their confidence with fuzzy sets, which allow them to represent their judgment using vague, but linguistically meaningful terminology. *Certus* includes syntax to specify confidence propagation using expressions that can be easily inspected by users. To demonstrate the concept of the language, *Certus* is applied to a worked example from the automotive domain.

Keywords: Assurance Cases · Safety Cases · Confidence Assessment · Domain Specific Language · Fuzzy Sets

1 Introduction

During the development of a critical system, engineers often prepare an Assurance Case (AC), as a "reasoned and compelling argument, supported by a body of evidence, that a system, service or organisation will operate as intended for a defined application in a defined environment" [1]. ACs are usually focused on a single quality attribute, such as safety (i.e., a "safety case") or security (i.e., "security-case"). Preparing an AC is required for compliance with a range of industrial standards; a notable and recent example being ISO/PAS 8800 which identifies preparing an AC as a central pillar in the AI assurance process for automotive technology [2]. Several notations exist for organizing the AC's argument, including the Goal Structuring Notation (GSN), Claims-Argument-Evidence (CAE), Eliminative Argumentation (EA), and the Friendly Argument Notation (FAN) [1,18,10,3]. While notation provides a foundation for describing AC arguments in terms of syntax and basic semantics, this is not enough to determine if an AC is acceptable. A question arises: *is there sufficient confidence that the claims in the AC are true?*

Many Confidence Assessment Methods (CAMs) for ACs exist. Some CAMs are qualitative [12,9,10,16] whereas others are quantitative and produce a numerical valuation of confidence in the claims in the argument [15,17,14], and some

Preprint. Submitted to SASSUR'25

mix qualitative and quantitative aspects [4]. Quantitative CAMs are attractive because they "sum up" the confidence in a claim into a number (or handful of numbers), which has benefits in terms of communication with interestholders and higher-level (potentially automated) decision-making about critical systems. However, there are also concerns about quantitative CAMS arising from the methods themselves and their validation [8,11].

This paper introduces *Certus*, a domain specific language (DSL) for AC confidence assessment that uses fuzzy sets to represent confidence in the argument and its supporting evidence. Using fuzzy sets, authors of ACs can express their confidence using vague, but linguistically meaningful, terms like "high" or "very low" confidence. Additionally, *Certus* authors can express propagation operations that compute confidence in a parent node based on its children. In prior work we suggested expressing confidence linguistically but did not formalize the concept [6]. To our knowledge, *Certus* is the first proposal to both model confidence in an AC with fuzzy sets and propagate confidence with a DSL.

The remainder of this paper is structured as follows. First, Section 2 discusses limitations of existing quantitative CAMs. Next, Section 3 describes our approach for modeling confidence using fuzzy set and presents the *Certus* language. Section 4 applies *Certus* to small fragment from a larger automotive AC. Finally, Section 5 closes with an outline for future work and concluding remarks.

Before proceeding, we note that this paper does not aim to compare qualitative or quantitative CAMs, nor does it take a position on whether qualitative or quantitative CAMs are preferred. In fact, as with many areas of engineering, different tools are applicable in different contexts and the choice to use a specific CAM depends on many factors. Our aim in this paper is to propose a tool to address known weaknesses of quantitative CAMs.

2 Motivation

Despite many quantitative CAMs existing in the literature, they appear to have limited use in practice. Possible reasons for a lack of adoption are introduced below, based on previous work and our own practical experiences [11,8]. Addressing these challenges is the motivation for the *Certus* project.

Interpretation. Quantitative CAMs produce a numerical valuation of confidence that must be interpreted by decision makers. There are at least two challenges with interpretation. First, it can be difficult to determine whether the calculated level of confidence is acceptable (e.g., is 0.93 confidence acceptable? Why not 0.92 or 0.94?). Second, numbers can easily be taken out of context or misinterpreted by non-experts (e.g., "But 0.93 confidence means there is 0.07 probability of system failure!"). Ideally, quantitative CAMs should represent results in a manner that resists misinterpretation or have interpretation guides readily available.

Subjectivity. Many existing quantitative CAMs require users to express judgement precisely, as one or more numbers. This is a subjective activity due to the

natural variability in human judgement. Even experts in the same field might ascribe different weights to evidence or arguments based on their education and experience. We hypothesize that reducing confidence to numbers and "one size fits all" calculations result in users loosing (qualitative) nuance or conceptual depth that is often important in matters of engineering judgement. Ideally, quantitative CAMs should allow judgement to be expressed vaguely and in a manner that is flexible enough to capture nuanced reasoning.

Scaling. Applying quantitative CAMs requires additional effort. For instance, to apply the Dempster-Shafer Theory method, a user must input four values per argument leaf node and at least three values per argument step [17]; when scaled to a large AC, this requires significant effort. Ideally, quantitative CAMs should either limit the number of inputs required, or provide mechanisms to reduce the number of inputs required in the most frequent use cases.

Defeaters. Dialectic reasoning (aka "defeaters") are increasingly used by practitioners to reason about doubt in an AC. However, despite the role of defeaters in expressing reasons to reduce confidence, few methods accommodate defeaters [7,13]. Ideally, quantitative CAMs should be capable of handling the "negative confidence" expressed by defeaters.

Trustworthiness. There are at least two issues relating to trustworthiness for quantitative CAMs. First, the mathematic frameworks used are complex, making them less accessible to busy practitioners. Tools that help automate calculations become "black boxes". As a result, practitioners are less likely to place their trust in a method where the means of calculation are not easily understood. Second, there is a lack of empirical evidence (e.g., case studies and controlled experiments) showing that methods produce trustworthy results [11]. Ideally, quantitative CAMs should be based on easily understood mathematical theories (or otherwise provide accessible guidance), and be validated to demonstrate they produce repeatable results that match intuition.

Summary. In summary, there are several limitations or challenges that prevent use of quantitative CAMs. Addressing them, in whole or in part, is our motivation for creating *Certus*. Fuzzy sets are a strong candidate for addressing challenges related to interpretation and subjectivity due to their ability to represent vague, but linguistically meaningful, information [20]. Additionally, using a DSL for specifying confidence propagation will support users to describe more complex propagation rules, accommodate defeaters, scale to large ACs, and promote trustworthiness by increasing transparency for how confidence is propagated.

3 The *Certus* Language

This section introduces the core concepts of the *Certus* language, beginning with how to model AC confidence using fuzzy sets and then describing how to specify

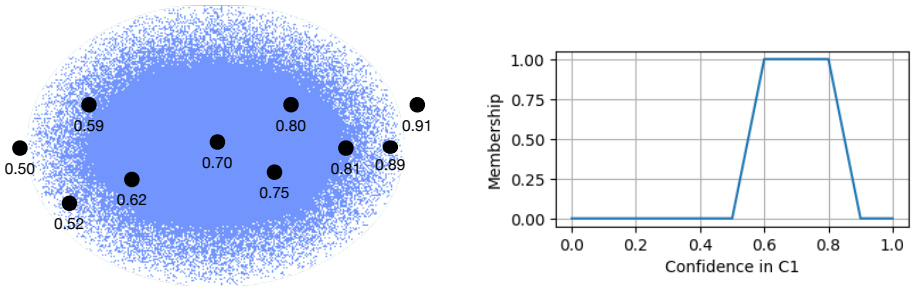

Fig. 1: Visualization of a fuzzy set for "'high" confidence and plot of the corresponding fuzzy membership function.

the propagation of confidence through an AC argument. For both brevity and clarity, we sketch the semantics of the language, and defer a formalization to future work. *Certus* assumes that AC arguments can be modelled as Directed Acyclic Graphs (DAGs). We use a subset of the EA notation in our examples: claims are shown as rectangles, evidence appears in blue ovals, and defeaters are red irregular octagons [10]. We selected EA as a matter of preference; *Certus* is applicable to other notations that represent arguments as a DAG.

3.1 Modelling Confidence with Fuzzy Sets

Let $\alpha_i \in \beta$ represent the degree of confidence that claim C_i is true, for any ordered set β describing degrees of confidence. Usually we take $\beta = [0, 1]$ with 1.0 corresponding to maximum confidence (i.e., $\alpha_i = 1.0$ reflects absolute confidence that C_i is true) and 0.0 is the minimum confidence (i.e., $\alpha_i = 0.0$ means that there is no confidence in the truth of C_i).

Consider the statement "*my confidence in claim* C_1 *is high*." This is a vague linguistic expression of confidence: how should it be interpreted? Specifically, what degree(s) of confidence in $\beta = [0, 1]$ are considered "high"? Suppose we decide degrees of confidence in $[0.6, 0.8]$ to definitively correspond to high belief, and then degrees of belief in $[0.5, 0.6)$ and $(0.8, 0.9]$ somewhat less corresponding to high belief. The left-hand diagram in Fig. 1 visualizes this scenario. Degrees of confidence are annotated as points: the points (e.g., 0.70, 0.62, 0.75) in the "core" are definitely in the set; some points are only partially in the set (e.g., 0.59, 0.52); and some points are entirely outside the set (e.g., 0.91 and 0.50). The same information can be shown as a membership function on the degrees of confidence in claim C_1, which appears on the right-hand side of Figure 1. Denote this fuzzy set membership function for "high" belief as $\mu_{high} : \beta \to [0, 1]$

This membership function μ_{high} maps degrees of confidence in a claim into membership in a fuzzy set denoting a "high" level of confidence. In doing so, it gives meaning to the statement: *confidence in* C_1 *is high*. In other words, the fuzzy set μ_{high} characterizes one's confidence in the truth of the claim C_1. This formulation can be generalized to arbitrary fuzzy sets for describing confidence

in a claim, allowing for statements like: *confidence in* C_1 *is* $\underline{A}$, for some vague qualifier that can be encoded as a fuzzy set membership function. Throughout this paper, we denote the set of all confidence describing fuzzy sets on the domain β (usually $[0, 1]$) as $\mathcal{C}$.

In *Certus* users can define any convex and normalized fuzzy set to describe their confidence. However, it is convenient to have a set of pre-defined canonical fuzzy sets that represent common confidence expressions. For the purpose of this paper, we identify five such sets: *zero*, *very low*, *low*, *med*, *high*, *very high*, and *certain*. For reference, some of these are depicted in Fig. 2. Note that *zero* and *certain* are "crisp" singleton sets with only one member: $\{0\}$ and $\{1\}$, respectively.

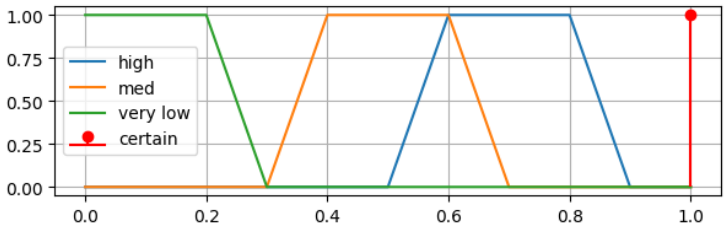

Fig. 2: Some canonical sets defined by the *Certus* language.

3.2 Propagating Confidence with *Certus*

Building on the ability to describe confidence in a claim using fuzzy sets, the two fundamental aspects of the *Certus* language can be defined: confidence assignment and confidence propagation. Confidence is assessed upward through the DAG in a recursive manner, from confidence assignment (usually the leaves) to the top-level node. At each logical step in the argument, a propagation operator is used to determine the confidence in the parent node based on the confidence assessed for the children. An example for a single argument step is shown in Fig. 3, where *Certus* annotations appear within partial rectangles connected to nodes with a dashed line. In this example, leaf nodes E_1 and E_2 have their confidence assigned and the parent node, C_0, has a confidence propagation operator. When assessed by *Certus*, the confidence in C_0 will be *high*.

Confidence Assignment. Confidence assignments allow a user to specify their confidence in an argument's nodes using a simple expression (e.g., `E1 is high`). The result is that the canonical fuzzy set *high* is associated with the evidence node E_1. For *Certus* to assess confidence in an AC, every path in the argument's DAG must include a confidence assignment from which confidence assessment can begin. It is preferable to apply assignments at the leaves of the DAG, which usually correspond to evidence typed nodes (i.e., "*how confident are we in this*

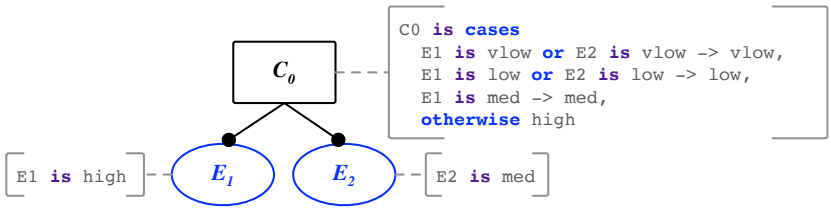

Fig. 3: Simple propagation step in *Certus*.

piece of evidence?"). However, it is also possible to specify confidence on a non-leaf node, which we call "shorting"[3]. In this case, *Certus*'s confidence assessment will not proceed further down the path and the shorting assignment will be used for propagation instead. Shorting is to be discouraged, because it ignores the arguments below the shorted node and might result in confidence assessments that are disconnected from real-world observations about a system. Even so, it can be useful in practice, especially assessing large or complex arguments.

Confidence Propagation. Confidence propagation is an operation that, when applied to a single argument step, determines the confidence in a parent node based on the confidence(s) in its children. Propagation is performed either by: 1) direct child to parent assignment, or 2) using the `cases` expression. For direct propagation, the confidence of a single child node is assigned directly to the parent. This is denoted as `C0 is C1`.

The `cases` expression matches (in order of appearance) conditions on the child confidences. Logical connectors such as `and` and `or` are used to match multiple nodes' confidences in one condition in the usual manner. The right-hand side of a single case can either be a fuzzy set (e.g., `E1 is med -> med`) or the identifier of a node (e.g., `E2 <= low -> E1`). In the later case, the fuzzy set for the identified node is propagated upward.

The `cases` expression is the basis of all propagation operators in *Certus*. This ensures that the core semantics are simple and easy for users to understand. Methods for specifying more complex and re-usable propagation operators are described below. The scope of a `cases` expression is limited to the direct descendants of the node it is applied to in the argument's DAG. We require that `cases` expressions in *Certus* be total functions that map from the set of all in-scope confidence assignments to the set $\mathcal{C}$.

Other quantitative CAMs encode the relationship between a child and parents using numerical weight parameters that are an input to the confidence propagation formula(s) [15,17]. This allows users to separate confidence in a premise (e.g., "*I am confident this is high-quality evidence*") from the strength of association between its parent claim (e.g., "*I am confident this evidence supports the parent*"). In *Certus*, there are no explicit weighting parameters. Instead, the user captures the relationship between a child and parent using the `cases` expression.

[3] This is a reference to short-circuiting an electrical circuit and also short-circuiting logical operators in some programming languages.

3.3 Comparison Semantics in *Certus*

There are several comparison operators that can be used in *Certus*'s conditional expressions to compare two fuzzy set membership functions. These are described below as binary relations on $\mathcal{C}$.

The `is` operator has two meanings in *Certus*: assignment (described above) and comparison. Given two fuzzy sets A and B, the comparison `A is B` evaluates to true if A is a subset of (or equal to) B. More formally, $is : \mathcal{C} \times C \to \mathbb{B}$ such that $is(A, B)$ if $\forall x \in [0, 1] : \mu_A(x) \leq \mu_B(x)$.

The `contains` operator is the reciprocal of `is`. Given two fuzzy sets A and B, the comparison `A contains B` evaluates to true if B is a subset of (or equal to) B. Formally, $contains : \mathcal{C} \times C \to \mathbb{B}$ such that $contains(A, B)$ if $\forall x \in [0, 1] : \mu_A(x) \geq \mu_B(x)$.

The `overlaps` operator is much weaker than `is` and `contains`. Given two fuzzy sets A, and B, the comparison `A overlaps B` evaluates to true if they have some non-zero overlap in their membership functions. Formally, $overlaps : \mathcal{C} \times \mathcal{C} \to \mathbb{B}$ such that $overlaps(A, B)$ if $\exists x \in [0, 1] : \mu_A(x) > 0 \wedge \mu_B(x) > 0$.

The *greater than* (denoted `>` or `gt`) and *less than* (denoted `<` or `lt`) operators require an ordering function to rank fuzzy sets. We use Yager's unit-interval fuzzy set ordering function to compare sets [19]. The ordering function, $F(A)$, computes the integral of the mean of level sets to produce a number that represents the position of the fuzzy set in $[0, 1]$. Comparing these numbers for different fuzzy sets allows one to order the sets, i.e., declare one set to "greater than" another. The computation is: $F(A) = \int_0^1 M(A_\alpha)\ d\alpha$, where $A_\alpha = \{x : x \in [0, 1], \mu_A(x) \geq \alpha\}$ is the α-cut of the set A (i.e., a "level set") and $M(A_\alpha)$ is the mean of the values in the level set. Then for fuzzy sets A and B, the *greater than* operation is formally defined $gt : \mathcal{C} \times \mathcal{C} \to \mathbb{B}$ such that $gt(A, B)$ if $F(A) > F(B)$. And vice versa for the *less than* operation. These operators can be extended to check equality (e.g., `>=`, `<=`) by comparing the membership functions of the fuzzy sets directly. They also enable operations such as `min(...)` and `max(...)`, which have the usual definitions. It is worth noting that, while Yager's ordering function works well most of the time, it fails to correctly order sets in some scenarios where membership functions are non-convex or non-normal [5].

3.4 Defined Propagation Operators

Even though the `cases` operator is flexible, it would be onerous to define lengthy propagation rules for every reasoning step in an AC. Therefore, *Certus* allows users to define named propagation operators and invoke them by name. This is similar to the notion of a pre-defined function in many programming languages. There are two ways to define propagation operators in *Certus*: parameterized operators and macro operators. Importantly, both use the `cases` expression described above.

Parameterized Propagation Operators. Using a parameterized operator, a user can define a propagation operator in advance and then invoke it by name. The

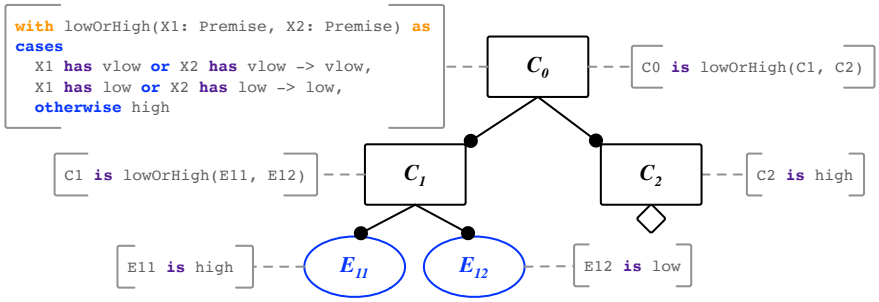

Fig. 4: Using parameterized propagation operators.

operator must be defined at an ancestor node in the argument's DAG or as a global definition provided separately from the argument. The inputs to the operator are typed by argument node type. The user assigns the nodes to the parameters when the operator is used in an argument step. The syntax: `with <name>(<parameters>) as <definition>`, is used to define a parameterized propagation operator. This is demonstrated in Fig. 4. An operator called `lowOrHigh` is defined to accept two nodes of type Premise (e.g., a Claim or Evidence nodes). If either of these nodes have fuzzy set membership functions that overlap the membership function of *low* or *very low*, then the output of the operation is respectively *low* or *very low*, otherwise the output is *high*. The operator is used two times, once at node C_1 and again at node C_0. Using the assignments in the leaves of Fig. 4, the overall confidence at C_0 is *low*.

Macro Propagation Operators. The propagation operators described above depend on a fixed number of inputs (parameters or nodes in an argument step). As a result, they could be cumbersome to use in scenarios where the number or type of nodes are likely to change. To address this challenge, *Certus* allows the user to define macros that are expanded into `cases` expressions as a pre-processing step before confidence is assessed. Macro expansion takes into account the current context of a node, including the number and type of children. Formally, macros are higher-order functions that map from the set of nodes in an argument step into the set of possible `cases` expressions. The expanded `cases` expressions can be inspected by users so that they can concretely understand how confidence is propagated through the argument.

Certus does not currently have a macro definition language. Instead, it provides an interface to scripting languages, such as Python. To define a macro, the user must define a function that accepts a list of argument nodes and then returns a `cases` expression. The `#MACRO_NAME` syntax is used to invoke macros, which must match the name of a function defined in the scripting language. By convention, macro names are defined with uppercase letters.

At present, *Certus* provides one built-in macro called `FUSE` that expands to a `cases` expression that merges (i.e., "fuses") multiple fuzzy sets together in a balanced manner. It uses an integer scoring function, $S : \{zero, ..., certain\} \rightarrow \mathbb{Z}$,

to determine the output for each case in the expression, e.g., $S(zero) = 0$ and $S(certain) = 6$. Premise type nodes are assigned positive scores and defeater nodes are given negative scores. Then the average of all clauses in a given case is computed and used to determine the output set for that case using the same integer scale. The expansion of this `FUSE` is shown in Fig. 5 for the case of two nodes. For example, in the third case of the expanded expression we have: $\big(S(med) + S(very\ low)\big)/2 = \lfloor (3+1)/2 \rfloor = S^{-1}(2) = low$.

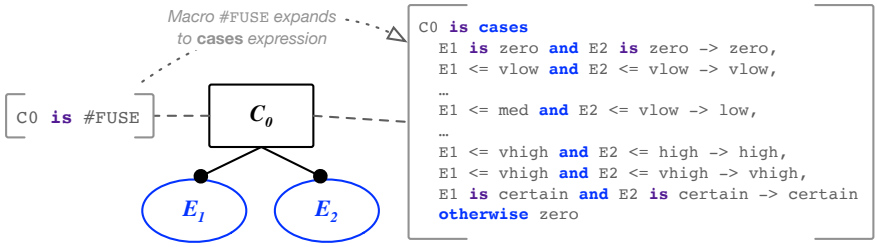

Fig. 5: Example showing the expansion of the `FUSE` macro.

3.5 Defeaters in *Certus*

As described above, dialectic elements (or "defeaters") are an important part of AC arguments, and quantitative CAMs should support defeaters in some manner. To include defeaters in *Certus*, we consider both how to model them as fuzzy sets and how defeaters are to be handled as part of confidence propagation.

Modelling Defeaters with Fuzzy Sets. In the most basic sense, defeaters are negative premises that capture a reason to doubt their parent claim. Whereas a normal premise (e.g., claim or evidence) contributes positively to the confidence in a parent, a defeater decreases confidence in its parent. Like for positive premises, the confidence in the credibility of a defeater is also a matter of degree [7]. It follows that confidence in a defeater can also be described as a fuzzy set. For instance, one could have *low* confidence in a defeater, which corresponds to a scenario where a doubt is not very credible, but cannot be entirely ruled out. Conversely, a defeater with *very high* confidence is likely (but not certainly) true.

Rules for Incorporating Defeaters. In prior work, we introduced 12 rules for incorporating defeaters into quantitative CAMs [7]. CAMs like the BBN or DST methods use pre-determined formulas to propagate confidence through the argument. These formulas can be extended and then analyzed to demonstrate they handle defeaters appropriately. In *Certus* the propagation operators are defined by the user and are not known in advance. Therefore, it is not possible to show *a priori* that confidence propagation in *Certus* respects the 12 rules for defeaters. However, it is still desirable for defeaters to be managed as an integral part

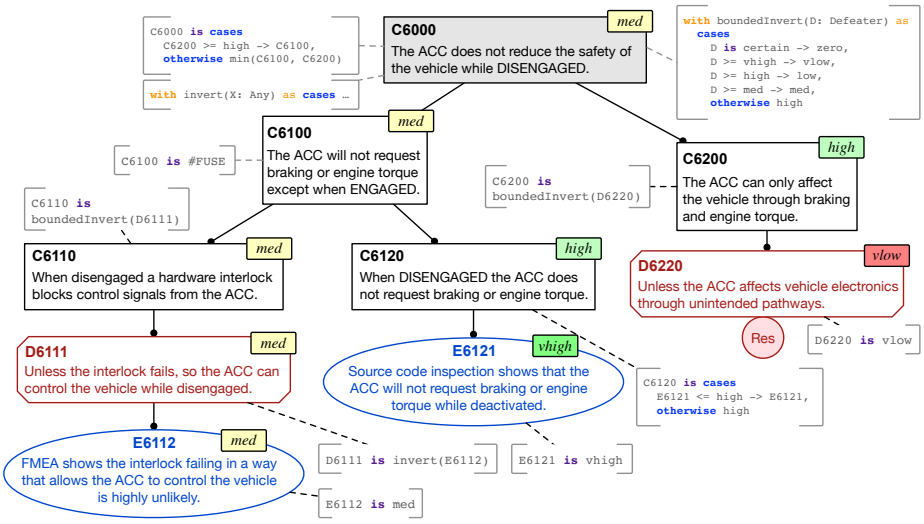

Fig. 6: Fragment from ACC AC with *Certus* applied.

of the method. So, immediately prior to computing confidence, after all macros have been expanded, *Certus* performs a static analysis of all `cases` expressions to confirm they comply with the 12 rules for defeaters. This applies to user-created expressions and expressions generated by expanding macros. These "pre-flight checks" are analogous to the checks performed by a compiler for many programming languages.

4 Worked Example

As a means of preliminary validation we have implemented a proof-of-concept version of *Certus*[4] and applied it to an AC fragment from an exemplar automotive adaptive cruise control (ACC) system, which is shown below in Fig. 6. *Certus* annotations appear in partial rectangles as above, and the name of the computed fuzzy for each node is provided for reference in the top-right of each node. The remainder of this section highlights points of interest in the example.

First, the example shows the use of two parameterized propagation operators: `invert` and `boundedInvert`. The `invert` operation returns the inverted confidence of its input (e.g., *low* becomes *high*), it is not fully defined in Fig. 6 for brevity. The `boundedInvert` is intended for reasoning steps where a premise is challenged by a single defeater. It is "bounded" in the sense that it does not output confidence greater than *high*, even if the input confidence is *very low*. Bounding the confidence that a parent derives from a single child defeater reflects the intuition that, in the absence of additional evidence to support the parent, there is a limit to the confidence that can be gained from resolving defeaters.

[4] `https://gitlab.com/sdiemert/certus`

Second, the `cases` expression for node `C6120` was designed to limit credit that can be taken from `E6121` to at most *high* confidence, regardless of the confidence in the evidence. In this case, the limitation is justified on the basis that source code inspection on its own would be inadequate to achieve *very high* confidence in the correct behaviour of a software function.

Third, the `cases` expression for node `C6000` weights `C6100` more heavily than `C6200`. Provided that `C6200` has at least *high* confidence, then the confidence in `C6000` depends on `C6100`, otherwise the minimum confidence from the two branches is used. Examining the text of nodes `C6100` and `C6200` justifies this logic. `C6200` can be defeated if there is a case where an unknown interaction of the ACC with the other vehicle systems exists. Provided such an interaction is ruled out with reasonable confidence, then the remainder of the argument rests on whether ACC is well-behaved within its known scope of operation, as contemplated by `C6100`.

Finally, to compare with the existing BBN method, we used the same AC fragment as in [7], and we mapped the leaf-level confidences from the BBN example to fuzzy sets for this example (e.g., belief in `E6112` was 0.6, which became *med*). The BBN example had an overall confidence of 0.39, which is consistent with, but perhaps on the low side, of the result produced with *Certus*.

5 Discussion

This paper has introduced *Certus*, a DSL for specifying confidence propagation in ACs using fuzzy sets. *Certus* addresses some of the challenges and limitations that prevent AC practitioners for using quantitative CAMs on real-world ACs.

Modelling confidence using fuzzy sets addresses challenges related to interpretation and subjectivity by allowing users to describe confidence using vague, but linguistically meaningful, expressions. For instance, in *Certus*, users do not need to select specific numbers to represent confidence; instead, they select a fuzzy set (e.g., *very high*) to represent their confidence. The output of *Certus* is similarly a fuzzy set which can be associated with a linguistic expression for interpretation by decision makers.

By using a DSL to specify confidence propagation in an AC, *Certus* allows users to represent sophisticated propagation rules that capture their reasoning and that accommodate dialectic reasoning. Further, *Certus*'s capability to define and re-use propagation operators reduces the number of inputs required and helps with scaling to larger ACs. Finally, in *Certus* confidence propagation operations are either represented by, or can be reduced to, simple expressions that can be inspected and understood by users, prompting trustworthiness.

This paper is our first description of *Certus*. There is significant work required to develop the concept and DSL prototype into a practice-ready CAM. From the perspective of developing the language, we plan to create a formal specification for the language; explore graphical representations of confidence propagation operators to improve readability; and create additional built-in macros for common operations. From an evaluation perspective, studies are required to show that

Certus is usable for practitioners and produces results that are trustworthy such that it can be used to support decision-making for critical systems.

References

1. Goal Structuring Notation Community Standard (Version 3) (2021)
2. ISO/PAS 8800:2024 Road vehicles — Safety and artificial intelligence (2024)
3. Bloomfield, R., Bishop, P., Jones, C., Froome, P.: ASCAD – adelard safety case development manual. Tech. rep., Adelard (1998)
4. Bloomfield, R., Rushby, J.: Assessing confidence with assurance 2.0 (2023)
5. Bortolan, G., Degani, R.: A review of some methods for ranking fuzzy subsets. Fuzzy Sets and Systems **15**(1), 1–19 (Feb 1985)
6. Diemert, S., Goodenough, J., Joyce, J., Weinstock, C.: Incremental Assurance Through Eliminative Argumentation. Journal of System Safety **58**(1), 7–15 (2023)
7. Diemert, S., Millet, L., Joyce, J., Weber, J.H.: Including Defeaters in Quantitative Confidence Assessments for Assurance Cases. In: Computer Safety, Reliability, and Security. SAFECOMP 2024 Workshops. pp. 239–250. Springer (2024)
8. Diemert, S., Shortt, C., Weber, J.H.: How do practitioners gain confidence in assurance cases? Information and Software Technology (2025), forth coming
9. Fenn, J., Hawkins, R., Nicholson, M.: A New Approach to Creating Clear Operational Safety Arguments. In: Computer Safety, Reliability, and Security. SAFECOMP 2024 Workshops. pp. 227–238. Springer Nature Switzerland (2024)
10. Goodenough, J.B., Weinstock, C.B., Klein, A.Z.: Eliminative argumentation: A basis for arguing confidence in system properties. Tech. rep., Carnegie Mellon University -Software Engineering Institute Pittsburgh United States (2015)
11. Graydon, P.J., Holloway, C.M.: An investigation of proposed techniques for quantifying confidence in assurance arguments. Safety Science **92**, 53–65 (2017)
12. Hawkins, R., Kelly, T., Knight, J., Graydon, P.: A new approach to creating clear safety arguments. In: Advances in Systems Safety, pp. 3–23. Springer (2011)
13. Herd, B., Kelly, J., Zacchi, J.V., Heinzemann, C., Diemert, S.: Integrating Defeaters into Subjective Logic-based Quantitative Assurance Arguments. In: European Dependable Computing Conference (EDCC). Lisbon, Portugal (2025)
14. Herd, B., Zacchi, J.V., Burton, S.: A Deductive Approach to Safety Assurance: Formalising Safety Contracts with Subjective Logic. In: Computer Safety, Reliability, and Security. SAFECOMP 2024 Workshops. pp. 213–226. Springer Nature Switzerland (2024)
15. Hobbs, C., Lloyd, M.: The application of bayesian belief networks to assurance case preparation. In: Achieving Systems Safety. pp. 159–176. Springer (2012)
16. Holloway, C.M., Wasson, K.S.: A Primer on Argument Assessment. Tech. rep., National Aeronautics and Space Administration, Langley Research Center Hampton, Virginia, United States (2021)
17. Idmessaoud, Y., Dubois, D., Guiochet, J.: Confidence assessment in safety argument structure - quantitative vs. qualitative approaches. International Journal of Approximate Reasoning **165**, 109100 (2024)
18. Kelly, T.P.: Arguing Safety - A Systematic Approach to Safety Case Management. Ph.D. thesis, University of York, York, UK (1998)
19. Yager, R.R.: A procedure for ordering fuzzy subsets of the unit interval. Information Sciences **24**(2), 143–161 (1981)
20. Zadeh, L.: Fuzzy Sets. Information and Control **8**(3), 338–353 (1965)