EC6

Methodological guideline for Assurance Cases Development

L6.1.5.1 - Recommandations de stratégies d'IVVQ de systèmes à base d'IA

L6.3.3.1 - Méthodologie de qualification basée Assurance Cases

contact@confiance-ai.fr | www.confiance.ai

Document reference: 613B

Contributors

	Name	Organisation	Role
Responsible for the deliverable	JENN Eric	IRT Saint Exupery	EC6 Contributor
Co-authors	JENN Eric	IRT Saint Exupery	EC6 Contributor
	CONEJO-LAGUNA Ramon	IRT Saint Exupery	EC6 Contributor
	MUSSOT Vincent	IRT Saint Exupery	EC6 Contributor
	CHENEVIER Florent	Thales	EC6 Contributor
	FERNANDES-PIRES Anthony	IRT System-X, ONERA	EC6 contributor
	ID MESSAOUD Yassir	IRT System-X	EC6 contributor
	FARGES Jean Loup	ONERA	EC6 Contributor
Reviewers	BRAUNSCHWEIG Bertrand	INRIA	Confiance.ai scientific coordinator

Document control

Revision	Date	Commentary	Author
0.9	2022/11/01	All document	All contributors
1.0	2022/12/05	Release version including reviewers comments	Ramon CONEJO-LAGUNA
2.0	2023/09/01	Updated version	Ramon CONEJO-LAGUNA
2.1	2023/12/22	Release version including reviewers comments	Vincent MUSSOT Eric JENN

Table of contents

Abbreviations and Acronyms

AI	Artificial Intelligence
ML	Machine Learning
IVVQ	Integration Verification Validation and Qualification
AC	Assurance Case
V&V	Verification and Validation
GSN	Goal Structuring Notation
ODD	Operational Design Domain
ACP	Assurance Claim Point

A. Introduction and abstract

A.1 General introduction to trustworthy AI challenges

Trustworthiness in AI within critical systems (systems that can directly or indirectly affect human life and moral entities) is essential for its widespread adoption (by the industry, the decision makers, the general public, etc.) and poses the following significant challenges.

- First, how to design AI models, so that, by construction, they satisfy trustworthy properties (accuracy, robustness…).
- Secondly, how to characterize these AI models, for example to understand and explain their behavior and their adequacy to the operational domain.
- Then, how to implement and embed those AI models on hardware, by making them fit for the target without losing their trustworthy properties.
- Another question is, what methods of data engineering to apply in order to, among other topics, manage important volumes of data and adapt to the evolution of the operational domain.
- At system level, what verification and certification processes to consider specifically for AI-based systems.
- Finally, a federation of all these matters is necessary to build an end-to-end methodological approach, supported by a consistent engineering environment compatible with industrial practices.

These are the challenges, among others, that the Confiance.ai program addresses.

A.2 Context

As Machine Learning (ML)[1] is becoming more present in the industry, its integration in safety critical systems remains slow, partly due to the absence of regulation and more fundamentally the lack of confidence in these data-driven systems. Therefore, establishing justified confidence in ML-based systems forms a necessary condition of the safety case for these systems. This requirement can be met using Assurance Cases and it will provide an argumented IVVQ strategy.

An Assurance Case (AC) provides a structured argument to justify certain claims about the system, based on evidences concerning both the system and the environment in which it operates. However, there are certain challenges that should be addressed:

Complexity of systems: The targeted systems are typically complex ones such as ML-based systems. The ML components they contain are usually difficult to understand and analyze, which makes it challenging to develop a comprehensive Assurance Case.
Heterogeneity of evidence: Assurance Cases must typically rely on a variety of evidence, including formal proofs, informal arguments, and empirical results. This heterogeneity of evidences is difficult to integrate and to reason about in a consistent way.

[1] In this document, the terms 'Artificial Intelligence (AI)' and 'Machine Learning (ML)' are used interchangeably and without distinction.

Scalability and maintainability: Assurance Cases can become very large and complex, especially in the current context where new ML methods and techniques are emerging at an ever-increasing pace. This can make it difficult to manage and maintain the Assurance Case, and to ensure that it is kept up-to-date.

Human factors: Assurance Cases are ultimately about convincing people that a system meet certain requirement. These arguments must therefore be understandable and convincing for a wide range of stakeholders, including technical experts, non-technical users, and regulators.

This guide is formulated to respond to these last challenges and advance towards a globally accepted argumented IVVQ strategy for ML components.

This document proposes a list of recommendations, techniques, patterns and rules to guide the construction of Assurance Cases, to simplify their writing and improve their consistency. The rules are presented with a specific format and numbering, and are recommended both for the building of assurance cases and for the reviewing process, where they should be verified.

This guideline is intended for the following type of users:

- Engineers and developers who are responsible for the development and integration of ML-based systems in safety-critical applications.
- Safety engineers and assurance professionals who are responsible for the safety assessment of ML-based systems.
- Regulators and standards organizations who are responsible for developing and enforcing safety requirements for ML-based systems.

In the context of ConfianceAI, this guideline provides steps for creating an argumentation around any trustworthy attribute. Please see document ‘613A: IVVQ Strategy - Assurance cases - Release 2.0’ for detailed argumentations over each attribute.

A.3 Overview

We provide guidance for the construction of assurance cases in Section **Erreur ! Source du renvoi i ntrouvable.**, where we detail how top-down approach should be applied to decompose high level properties, while bottom-up approach can be used to exploit the available evidences. In the end of the section, we also gather rules that should be followed to ensure consistency in the building process, especially when assurances cases are written in a collaborative manner. These rules are either taken from the literature reference in the state of the art of Batch 1, or are the results of our own experience and consensual decision of team members of EC6.7.

The following Section **Erreur ! Source du renvoi introuvable.** presents the main elements which c omposes the assurances cases developed in the context of Confiance.AI, and the associated formalism used in the textual format. While it closely follows the GSN formalism, it also differs on minor aspects to support specific needs in our context. We also provide guidance for using these elements, still taken from both the literature and our experience. Moreover, an important part of this section is dedicated to the presentation of strategies, which makes the decomposition of properties explicit when building an argumentation.

Finally, Section B.4 presents important argument patterns used in our assurance cases, for which we provide a comprehensive description and an illustration of their instantiation.

Note: In the rest of this document, we use the GSN formalism, but the concepts presented are applicable whatever the notation. For instance, the term “Goal” is used everywhere, while we reserve the term “claim” for the statement supported by a goal or a strategy.

B. Description of the method

B.1 Assurance Case construction approaches

B.1.1 Top-down and bottom-up approaches

This section provides guidance on building Assurance Cases in the context of the Confiance.AI program. It relies on the two main approaches which either starts from high level properties expressed on an engineering item to develop an argument in a top-down fashion, or starts from the available methods and tools to provide evidences in a bottom-up approach. In practice, both approaches are used simultaneously as described in the section B.1.4.

B.1.1.1 Top-down approach

The following steps describes how Assurance Cases can be built in a top-down fashion, starting from a property on a specific item of interest, and decomposed until the goals are sufficiently simple to be answered with a specific method or tool, which can be linked to a specific V&V activity.

Steps

1. Select an **engineering item** produced by some activity in the system development process.
2. Select a **property** applicable to the item (or requirement). This property becomes the root goal of the Assurance Case. The role of the Assurance Case is to build a rigorous argumentation to show that the property holds for the item for a given context.
3. Once the main goal (property) is selected, construct an explicit record of the information necessary for the reader to understand the **context** in which the goal/s identified are put forward. The author must make use of context elements: Refer to 'Auxiliary node – [C] Context' chapter.
4. Determine if the satisfaction of the goal can be obtained using some available solution (see next paragraph).
   - If a solution is available (**evidential step**), the argumentation stops
   - If no solution is available (**reasoning step**), identify how the initial goal can be decomposed into smaller / simpler sub-goals. Then beginning the argument development process again at Step 3, although this time obviously the goals are one level further down the goal structure
     - This decomposition must ideally be deductive, i.e., the sub-goals must logically entail the parent goal.
     - When this decomposition is not deductive, a level of confidence on the evidence can be given.
     - As far as possible, the decomposition shall apply one of the strategies or one of the patterns proposed in Sections B 3.2.7 'List of Strategies' and B 4 'Patterns'.
     - The contextual basis for the argument strategy may include rationale information as to why the strategy has been adopted. In GSN, this is achieved with the use of assumptions and

justifications. If necessary, additional Context and Definition elements can be added to the strategy as well.

- Once the new goals are identified and the strategy is defined, the author should go back to step 3 for each goal until all branches end with a solution.

The approach described above is essentially **"top-down"**, since properties are refined progressively down to the point where the final goals are simple enough to be verified or proven. It implies the existence of some verification artefact (a demonstration, a test result, etc.).

B.1.1.2 Bottom-up approach

The following steps present the inverse approach, which consists in starting from available methods and tools that can be provided as solutions or evidences, and going up in the argument to try to link them to higher-level properties.

Steps

1. Identify the Solutions available. That is, engineering artifacts that can serve as evidences.
2. Deduce the claims that can be deduced by these solutions, and link them as a GSN goal
3. Select and infer higher-level goals that are supported by the previous goals
4. Explain how each parent goal is supported by its lower-level goals as a Strategy element
5. Check any necessary additional information to be included as a context
6. Link the whole structure to an existent top goal

In the context of Confiance.AI, this approach was used to identify the relevance of some solutions available in the program as follows:

a) Analyze the deliverables of Confiance.ai in order to identify the methods and solutions already selected by the program

b) Estimate the level of trust one can place on those solutions and the effort / cost to implement them

c) Identify the properties for which these methods can bring evidences

These steps are not strictly bottom-up, as it is necessary to know what kind of property we are aiming for and to have a broad idea of how we expect to unroll the argument in the first place, which is why in practice, both approaches are usually combined, as explained in next section.

B.1.1.3 Combining both approaches

In practice, building an Assurance Case actually combines top-down and bottom-up reasoning. In particular, having a ready-made list of solutions can be used as building blocks to build part of the argumentation "bottom-up" (in the same way as having a list of software building blocks can be used to make appropriate design choices …). On the other hand, when no predefined solution exists, it will be required to produce

specific evidences and to define the associated V&V activities, which also implies to verify that they are feasible and applicable in current industrial context.

B.1.1.4 Verifying and consolidating the argument

In order to consolidate and ensure validity of the argument, it is recommended to perform a critic of the assurance case product, which is also the way that can be followed by external actors to review an assurance case. This verification can be done by trying to identify all scenarios that could invalidate the reasoning (the potential defeaters) and justify why those scenarios are not possible (or prevented). This information may be kept outside the Assurance Case but could be important for the future reviewer. See Evaluation of an Assurance Case reference.

B.1.2 Reasoning steps and evidential steps

Assurance cases are not aimed at providing absolute guarantees. They provide argumentation elements deemed sufficient to justify confidence to be placed on the capability of the system under design to deliver some expected services. Ideally, the argument should be purely deductive. It should rely on formally defined rules and ensure that a goal can be deductively related to its subgoals.

In practice, an argument involves two different steps [Rushby-15]:

- Reasoning steps establishing a **logical** relation between steps
- Evidential steps establishing **epistemic** relation between steps.

Logical steps can themselves be either deductive or inductive.

In a deductive step, the satisfaction of sub-goals logically entails the satisfaction of the parent goal, with no place for interpretation, uncertainty. On the contrary, inductive steps only provide “strong and convincing evidence for the truth of the conclusion” [Rushby-15], but no absolute certainty. Even though the degree of certainty may be increased by adding new sub-goals, the reasoning step still remain questionable. To consider that the step is “acceptable” is a matter of conviction, hence the need to provide as many elements of explanation (strategies, context, justification,…) as possible to be convincing.

In summary

- In an inductive step, the conjunction of subgoals **suggests** that the goal is true.
- In a deductive step, the conjunction **implies**, **entails**, or **proves** the goal.

Evidential steps cannot be deductive since they refer to knowledge and observations, knowledge may be about test results, application of or other observations, etc. For those steps, the argument does not refer to pure logic (i.e., the manipulation of symbols according to well defined rules). An evidence does not imply a goal in the logical sense, but simply gives confidence on the fact the proposition stated in the goal actually hold. The stronger the evidence, the stronger the confidence on the truthiness of the proposition.

Rushby proposes not to mix deductive (or “reasoning”) and evidential steps to ease the interpretation of the assurance case and preserve the decompositions as formal as possible. In other words, to support a specific strategy or goal, one and only type of reasoning step (deductive or evidential) must be used. This is illustrated on Figure 1 where the initial argumentation for claim C, which was mixing two types of arguments (left side), is “normalized” in order to clearly distinguish the reasoning and evidential steps (right side).

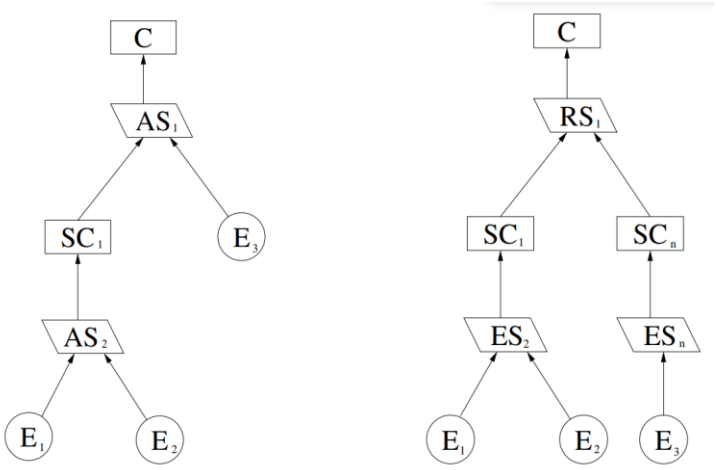


Figure 1: From a Free-Form Assurance Case to a normal-form AC (extracted from [Rushby-15])

This led us to define the following first set of rules for correctly designing an Assurance Case:

[Rule 001] Discrimination between reasoning steps and evidential steps

The Assurance cases must clearly discriminate *reasoning steps* and *evidential steps.* Reasoning steps must apply "logical strategies" (cf. section "strategies").

[Rule 002] Deductive then inductive

Inductive steps must be introduced as late as possible in the argumentation (ideally at last step where the solution is introduced).

[Rule 003] Confidence in the evidential steps must be evaluated

As far as possible, confidence on evidential steps must be evaluated (“weighted”) in order to support analysis of the strength of the argument.

Note: This rule is introduced in batch#3. See ‘613D Assurance Cases - Uncertainty Assessment’.

B.1.3 Propagation of properties

Assurance cases are an argumentation tool to prove a property on an item (representativeness of a dataset, robustness of an ML model, completeness of an ODD, etc.). However, it must be highlighted that those items are modified by other activities and transformed into other engineering items. For instance, the property of representativeness of a dataset may be destroyed during the splitting of this dataset. This section introduces the notion of properties propagation along the engineering workflow.

Pre- and post- conditions and propagation of properties

The properties considered in our Assurance Cases are pre- and post-conditions associated with workflow activities (noted Pr and Po on Figure 2): if the input engineering items of activity A satisfy pre-conditions Pr, then the output engineering items produced by A satisfy post-conditions Po.

In software engineering, this is referred to as the Hoare Triple in program verification: {A} p {B} such that if the initial state satisfies assertion A and the execution of program p terminates then the final state satisfies assertion B.

In our context, we consider that all activities terminate and that they have no internal state (or “memory”). In other words, an activity has no other effect than modifying its outputs so that its specification can be written as properties about its inputs and its outputs. For instance, the activity “Implement ML model” only involves the input model (which must be well formed, non-ambiguous, complete, etc.) and the implemented model (which must strictly preserve the semantics of the input model).

Note: *Only the post-conditions approach is explored in Batch #3 of Confiance.AI. The pre-conditions approach will be investigated in the future.*

Figure 2. Pre / post conditions and activities

The pre-condition for an activity is usually implied by the post-condition of a previous activity in the workflow. In the previous figure, for instance, if post-condition $Po_{A_{n-1}}$ is satisfied then pre-condition Po_{A_n} is satisfied too.

Those pre- and post-conditions represent part of the *specification* of an activity. Other requirements may not refer to the output themselves, but to the way the activity is performed, such as the use of such or such method or tool, etc.

In the general case, verifying that some property P is satisfied for some engineering item X produced by some activity A can be achieved in different ways:

- It can be verified "**directly**" on the Engineering Item to which the property refers.
- It can be verified **indirectly**, either **earlier** on some Engineering Item P' from which P derives, or **later** on some Engineering Item P' deriving from P. In both cases, demonstration shall be given that P is satisfied if P' is satisfied. This may require, for instance, demonstrating that activities involved in the derivation process preserve the property at stake.

The alternative between "process-based" and "product-based" approaches is a typical example of this kind of choice. Both approaches aim at ensuring the correctness of the product, but in different ways:

- The "process-based" approach relies on the verification of the intermediate development artefacts. For instance, demonstrating the absence of (some) runtime errors may be achieved by ensuring

compliance with coding standards and source code reviews. In the ML domain, ensuring the robustness of a model can rely on the use of *by-design* methods such as Lipschitz Networks.

- The "product-based" approach relies on the verification of the end-product. Here, demonstrating the absence of runtime errors (so demonstrating that the source code is free form such error, i.e., a property applicable to the source code) can be done by performing static analysis of the binary code. In the ML domain, one can verify the robustness of an ML model in a *post-hoc* fashion with formal verification typically.

Accordingly (in principle) for each property P, the user has to identify the relation between P and the properties applicable to other Engineering Items in order to see where verification is the most efficient in terms of confidence and cost. The idea is to either move upward or downward in the workflow in order to see where a property P' applicable to some other engineering item would imply P and would be demonstrated at a smaller cost (possibly because it is already verified for another reason). However, two additional elements must be considered:

- "Propagating" the property downward in the process means that, in case of error, all activities involved in the production of the faulty Engineering Item will have to be redone to remove the error, so the later the verification, the higher the number of activities to redo.
- The difficulty to demonstrate that P' implies P becomes more and more difficult as the number of activities "between" P and P' increases.

Let us consider the *TRAIN* activity (A3) on Figure 3. A precondition Pr_{3} for this activity is that the *train dataset is clean* ($Pr_{3.a}$) *and labelled correctly* ($Pr_{3.b}$, see figure below). Should one of these properties being violated, the result of the training process would certainly not be satisfying (*). In this example, no activity other than A3 actually requires the dataset to be clean and correctly labelled (**). (In fact, the "clean" activity would normally not exist in a fault-free world. This activity is here to ensure that pre-condition P is satisfied.)

Verifying that $Pr_{3.b}$ is satisfied (i.e., building the corresponding assurance case and producing the associated evidences) can be done either directly on the input engineering items of A3 or indirectly by verifying that the output of A1.2 are *correctly labelled,* and by demonstrating that the *correctness of the labelling is preserved by activity A2*.

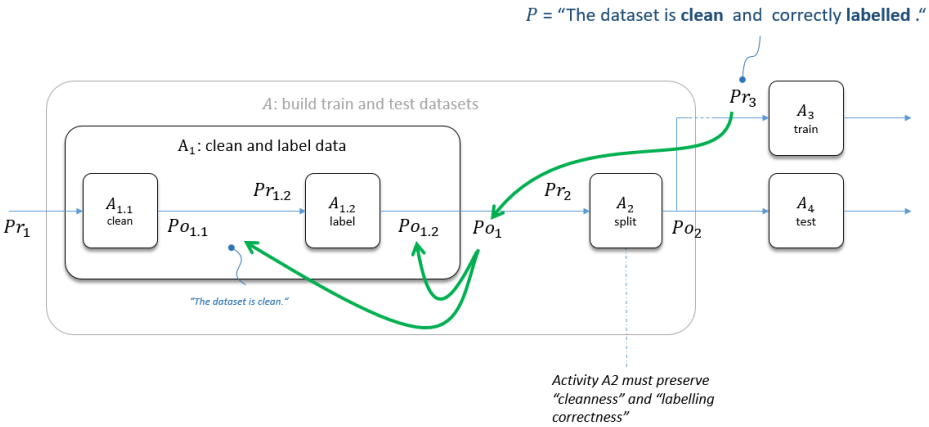


Figure 3. Property preservation

(*) Note that we do not give the post-condition of A3.
(**) The labelling process would probably also require data to be clean…

Challenges in propagation of properties

Currently the focus is actually placed at the level of a single activity: *assuming* that the pre-condition holds, how can we demonstrate that the post-condition also holds? Demonstrating the relation between a pre-condition and the corresponding post-conditions of engineering items produced upward in the workflow is not considered.

Among the topics that could be analyzed:

- The definition of the chain of implications relating two properties across several engineering items, upward or downward in the workflow.
- The clear identification of properties not related to the correct realization of the activity producing the engineering item but related to the propagation chain.
- The definition of the list of activities to be repeated in case of a violation of a property.

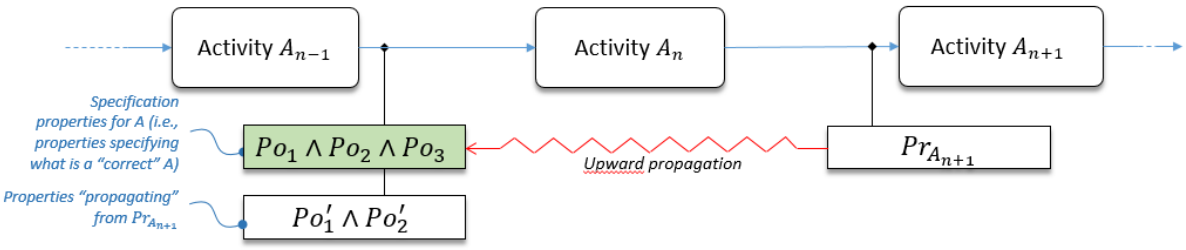


Figure 4. Property "propagation"

B.2 Rules for building Assurance Cases

The following list of recommendations is strongly inspired from [Holloway-16] and [Kelly-07].

[Rule 004] Assurance Cases must be understandable.

An assurance case must provide enough information so that everyone who will use it knows what it means, and to what it applies. The links between the elements (goals, strategies, solutions, etc.) of the argumentation must be clear. The meaning of a goal must be clear. In particular, this applies to the terms of the goal: each term must be well-defined. This can be achieved using appropriate definition nodes or contextual nodes (when the term refers to an element that depends on the context). The relation between a parent goal and a child goal must be explained (using "strategy" nodes).

Comments: part of this recommendations are ensured by the application of standard notation and appropriate tooling.

[Rule 005] Assurance cases must be non-ambiguous

'Ambiguity' is defined as "the capability [of a word or phrase] of being understood in two or more ways" (Oxford English Dictionary). Two types of ambiguity are commonly distinguished: lexical and syntactic.

- Lexical: Avoid multiple meanings inherent in a single word or phrase.
- Syntactic: Avoid grammatical structures that allow for multiple correct interpretations

  Example from GSN: "System functional software requirements development is acceptably safe" --> Multiple interpretations --> Possible syntactic correction: "The development of system functional software requirements is safe".

[Rule 006] Assurance cases must use well defined terms

An assurance case must provide the definitions of important terms which are not considered "common knowledge".

Comment: There is no absolute criteria to discriminate those terms, as it might even depend on the target audience of the assurance case. The appreciation is left out to the writer, but it should also be a focus point to identify such terms during the review process.

[Rule 007] Assurance cases must use terms consistently

An assurance case must use the same meaning for a specific term. Reciprocally, the assurance case must always use the same term to designate a given concept.

Comment: For instance, writers should avoid 'blanket terminology' [GSN-21], where a single word is used to refer to different things. In GSN, there is an assumption that the scope of terms is inherited from statements at a higher level. However, a given term can be refined downstream in the argument

[Rule 008] Context must be clearly defined

An assurance case must define clearly the context in which the goal takes place. For instance, if an item is claimed to comply with some "specification document", the "specification document" must be clearly identified as part of the context.

Rationale: the context is aimed at clarifying the meaning of the goal and providing a description of the elements that should be later instantiated.

[Rule 009] Complexity of reasoning steps must be controlled

The writers should limit the number of sub-goals at each refinement step, typically up to 4 or 5. Otherwise it indicates that a partitioning of the subgoals or a re-design of the argument may be required.

[Rule 010] Assurance cases must be up to date w.r.t. the system

An assurance case must accurately represent the current status of the system, its service and its development workflow in all relevant aspects.

[Rule 011] Assurance cases must be complete

An assurance case must cover all aspects related to the top-level property of the system or service in an appropriate way, for a given context.

Comment: Having multiple people considering the same problem from different angles and collaborate into building the same assurance case is valuable to ensure a form of completeness in the reasoning.

[Rule 012] Assurance cases must be grounded

An assurance case must terminate in premises whose ‘truth’ can be agreed by all relevant parties. The "relevant parties" must be well-defined.

Comment: This aspect should be one of the main focus of the assurance case review process. See 613C Evaluation of Assurance cases - Release 2.0.

[Rule 013] Assurance cases must be realistic

An assurance case must only rely on correct assumptions on what will happen (or is happening) in the actual world.

[Rule 014] Assurance cases must not be suppositious

An assurance case must not be based on implicit assumptions.

[Rule 015] Assurance cases must be balanced

An assurance case must identify not only the strengths of the system or service but also its known weaknesses.

[Rule 016] Assurance cases must be well-formed

An assurance case must be well-formed. In particular, it shall show no circularity in the argument. Goals must be supported. The role of evidences must be clear. The argument must be "fully-connected".

[Rule 017] Explicit reasoning strategy

Assurance cases must indicate the strategy applied to decompose a goal into a sub-goal.

[Rule 018] Evidence / Solutions

The description of a piece of evidence or a solution must describe the content of the evidence / solution, using an additional context if needed. For instance, if the evidence is a "test report", the actual summary of the expected content this report could be described in a context.

Comment: In our context, this rule allows for building the required V&V activity while knowing what to expect to support a specific part of the argument.

B.3 Assurance Case elements

This section presents the elements (or "nodes") used to build assurance case arguments, following the GSN formalism.

Nodes are separated into:

- **Core** nodes, which contains the main element of the argument: the goals, the solutions, and the strategies used to decompose goals into subgoals
- **Auxiliary** nodes, which support the argument with additional information, such as context elements, definitions considered for specific terms, informal justifications, or assumptions considered in the process of building the argument

B.3.1 Core node – [G] Goal

A **goal** is a **claim**, an assertion that must be proven within the argument [GSN-21].

B.3.1.1 Writing Rules

[Rule 019] Goals are expressed as properties

A goal must be expressed as a property to be satisfied by some item (an engineering item, a procedure, etc.). In particular, a goal shall not express the means / ways to achieve a certain property (e.g., "the element must be developed using method xxx."). These "achievement means/ways" are part of the solutions presented to achieve the goal, but not part of the goal itself.

[Rule 020] Goals are propositions

A goal shall claim the satisfaction of a property about some item. Therefore, a goal shall be either *true* (the property is satisfied) or *false* (the property is not satisfied).
For instance:

- The <test dataset> satisfies the <test completeness criterion>

- The <model> passes the <test success criterion>
- The <accuracy> is greater or equal to 90%

[Rule 021] Goals are understandable and non-ambiguous

A goal shall be understandable and non-ambiguous. Those criteria being very subjective, they shall be ensured via an appropriate review process.

Comment: This rule is also strongly related to [Rule 005] which indicates that assurance cases must be non-ambiguous

[Rule 022] Goal independence

All sub-goals of the same argumentation step shall be independent. In other terms, when a goal is decomposed into several subgoals, the truth of each subgoal must be independent from the truth of the others.

[Rule 023] Syntactic and semantic rules

- A goal starts with the prefix [G]
- If a term of the goal refers to some artifact, the exact name of the artifact shall be used in the goal (e.g., <training dataset>, <ODD specification>, etc.). Additionally, a Context [C] element should be added referring to the same artifact.
- A goal shall preferably follow the structure: <item> <verb> <property> with <verb> being a state verb ("is", or "satisfies", or "conforms to"…).
- A goal shall not describe an action. For instance: "The <data selection procedure> generates a <test dataset> that satisfies the <test criterion>"
- A goal can be compound in a disjunctive or conjunctive manner, with potential negations. A compound goal is a group of two or more statements connected using the following words: 'or' or 'and' and potentially 'not'.

Examples

We provide in the following a running example to clarify some of the abovementioned rules:

A. Not recommended

```
[G] The <test dataset> conforms to the <test criterion> by means of a <data selection procedure>
```

[Ambiguous]

This sentence is ambiguous because it is not clear what part of the sentence the property refers to: if the conformity to the <test criterion> is a fact, then the claim of the goal supports the fact that it is

achieved using a specific <data selection procedure>. On the other hand, the goal claim could be related to the conformity, considering that a <data selection procedure> is effectively applied. Alternatively, this goal could be about the satisfaction of both properties, I.e., the conformity and the way it is achieved. If there is a conjunction of the two properties, there must be an "and" between them.

[Includes how-to]

This goal includes the manner or way to achieve the property.

B. Not recommended

```
[G] The <test dataset> is generated by a <data selection procedure> that satisfies the <test criterion>
```

[Includes how-to]

The author cannot add the property (satisfies the test criterion) and the way to achieve it (is generated by a data selection procedure)
[Wrong compound goal]

Although a compound goal is allowed, we discourage to use other linking words out of 'or' or 'and'.

C. Recommended

```
[G] The <test dataset> satisfies the <test criterion>
```

To avoid the multiplication of low-value goals (such as "<item1> is defined and correct"), we state the existence of items in the AC using the context [C]. Therefore, a context is just a list of its elements (not a sentence with a verb).

Example:

```
[G] A <test dataset> is correct if it satisfies a <test criterion>
    [C] Applicable <test criterion>
```

B.3.1.2 Decomposition Goals into sub-Goals

See the previous chapter B.1 ‘Approaches’ which describes how to decompose goals into sub-goals.

B.3.1.3 Assumptions and goals

In some cases, the meaning of a goal only makes sense if another goal is true. As illustrated by [Rushby-15], a goal involving a ratio x/y makes sense only if y≠0. A goal may be added to state that y≠0, and this goal must be verified before the goals depending on x/y. To solve this issue, the author proposes to read the assurance case from left to right (when represented as a tree) and consider that the "previous" goals are satisfied.

If such goal (y≠0), does not need to be supported by further arguments, is a proposition that is asserted as true and therefore we consider it as a GSN assumption element.

B.3.1.4 Supporting a Goal with evidence

Solutions should be attached to a goal when the decomposition makes these goals simple enough to be supported by a direct factual evidence. In our case, we typically consider the result of V&V activities as direct evidence. However, one should pay attention to the risk of closing goals too early, without providing the necessary steps for the full comprehension of the argument.

As an example, Figure 5: Premature Solution example. , shows how the goal could be decomposed in additional sub-goals to explain how the robustness to adversarial examples has been met in the Training and Test report.

On the other hand, long chains of single Goals with no branch should be avoided, as it only provides rephrasing and redefinition of terms, is not a practical approach for evidential steps.

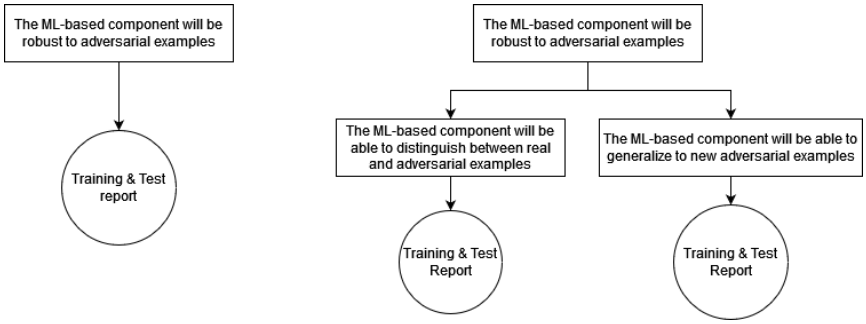


Figure 5: Premature Solution example. On the left side, the goal is closed too early. On the right a further argumentation is given with intermediate goal statements that explain the reasoning behind the solution.

As a general rule, we suggest to descend in the argumentation tree until reaching a leaf-goal which:

- Induces directly the necessary evidence
- Does not require further expansion

Once the Solution is placed to support a leaf-goal, additional explanatory information can be added to the relationship between the Solution and the Goal such as Justification and Context Elements. If the justification is not evident and need further clarification, then a further decomposition of goals is suggested as indicated previously.

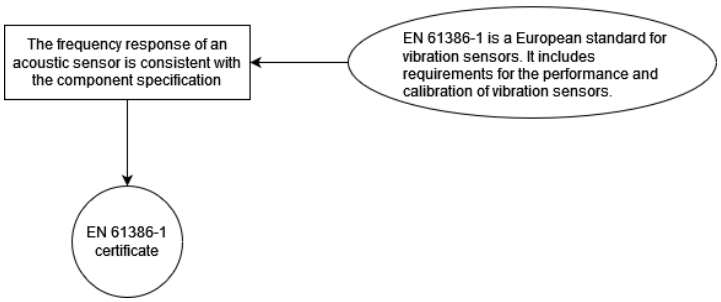


Figure 6: Example of adding a Justification to a Goal.

Note that it is possible to support a Goal with multiple solutions.

B.3.2 Core node – [S] Strategy

A strategy justifies the decomposition of goals into sub-goals.

B.3.2.1 Usage of strategies

A strategy element must be introduced whenever the parent-goal to sub-goal decomposition is not "obvious". Obvious cases are those where the decomposition is the application of a simple logical procedure. For instance, when terms of a goal are replaced by elements of their definition, or when a goal expresses a conjunction of properties and the sub-goals express each of the properties, strategy is optional.

The idea behind using a strategy is to verify if the decomposition is logical, or if it requires some indication about why it makes sense to consider that the set of subgoals is equivalent to the parent goal.

B.3.2.2 Writing Rules

[Rule 024] Strategy are explicit decompositions

A strategy must justify why the sub-goals suggest (inductive step) or imply (deductive step) their parent goal. It is similar to a justification element, but it specifically and explicitly refers to the relation between a parent goal and its sub-goals.

Comment: It should be possible to remove all of the strategy elements from an argument without affecting the logical flow of the claims being made. Reciprocally, it should be possible to explicitly write a strategy for every decomposition steps in the assurance case.

[Rule 025] Strategy are not part of the argument

Strategies should not contain claims.

[Rule 026] Syntactic and semantic rules

- A strategy starts with prefix [S]
- Strategy statements contain a brief description of the argument approach.
- It is useful to introduce a summary of the argument approach with a phrase such as “Argument by appeal to…”, “Argument by …”, “Argument across …” [GSN-21]

[Rule 027] Strategy contextual information

If the relationship between a goal and a set of sub-goals has any complexity, or if the strategy being used requires additional context, it should be explicit.

Note that, in our context, the concept of a GSN strategy is different from the concept of an (assurance) strategy applied to a complete assurance case. The latter refers to the choices made on one particular assurance case.

Examples of Good strategies	Examples of Bad strategies
"Argument by appeal to the Byzantine Agreement protocol."	"Use Byzantine Agreement protocol."

Table 1: Extracted from [Weinstock-07]

B.3.2.3 List of Strategies

This section presents a list of *strategies* that can be applied to build an assurance case. The use of these strategies is recommended to maintain a certain level of consistency in the set of assurance cases produced in Confiance.AI. However, by using inference rules stemming from *formal logic*, we also detail in this section strategies that are usually considered trivial and are optional when writing an assurance case.

Three main categories of strategies can be identified:

- Strategies related to the **logical inference process**
- Strategies related to the **syntactic manipulation** of goals (rewording)
- Other strategies

Inference rules

The following strategies correspond to application of the classical inference rules. Those rules are purely syntactic: they do not rely on the interpretation (or semantics) of the clauses.

The elaboration of the Assurance Case is usually performed from the **conclusion** to the **premises** (top-down), so the inference rules will usually be considered the other way around: from the conclusion (the goals) to the premises (the sub-goals).

For instance, in our case, the "conjunction" inference rule goes from the conclusion ($P \wedge Q$) to the premises (P, Q).

The following table gives a list of inference rules and corresponding patterns. Symbol $\neg$ represents the négation ; in the expression $S \vdash q$, symbol $\vdash$ means that assuming that all propositions in S hold then proposition q holds.

Negation	Negation: $\neg\ (\neg p) \vdash p$
	
	`[G] Property P is satisfied`
	`  [S] Negation`
	`  [G] Property not P is not satisfied`

Conjunction	Conjunction
	
	`[G] Property P1 and P2 and ..., and Pn is satisfied`
	`  [S] Conjunction`
	`  [G] Property P1 is satisfied`
	`  [G] Property P2 is satisfied`
	`  ...`
	`  [G] Property Pn is satisfied`

Disjunction	Disjunction
	
	`[G] Property P1 or P2 or ... Pn is satisfied`
	`  [S]1 Disjunction`
	`  [G] Property P1 is satisfied`
	`  [G] Property P2 is satisfied`
	`  …`
	`  [G] Property Pn is satisfied`

Modus Ponens	Modus ponens: $p,\ p \rightarrow q \vdash q$
	
	`[G] Property P is satisfied`
	`  [S] Modus ponens`
	`  [G] Q implies P`
	`  [G] Property Q is satisfied`

Modus Ponens(multiple)	`[G] Property P is satisfied`
	`  [S] Modus ponens (multiple)`
	`  [G] Satisfaction of Q and R and S and ... implies P`
	`  [G] Property Q is satisfied`
	`  [G] Property R is satisfied`
	`  [G] Property S is satisfied`
	`  [G] ...`

Modus Tollens	Modus tollens: $p \rightarrow q,\ \neg q \vdash \neg p$
	
	`[G] Property P is satisfied`
	`  [S] Modus tollens`
	`  [G] Property not P implies Q`
	`  [G] Property not Q is satisfied`

Disjunctive Syllogism	Disjunctive syllogism: $p \vee q,\ \neg p \vdash q$
	
	`[G] Property P is satisfied`
	`  [S] Disjunctive syllogism`
	`  [G] Property (P or Q) is satisfied`
	`  [G] Property Q is not satisfied`

Hypothetical Syllogism
Nb: this pattern is equivalent to a disjunction of two modus ponens
	Hypothetical syllogism: $p \rightarrow q,\ q \rightarrow r \vdash p \rightarrow r$
	
	`[G] Property P implies R is satisfied`
	`  [S] Hypothetical syllogism`

	[G] Property (P implies Q) is satisfied [G] Property (Q implies R) is satisfied
Constructive Dilemma *Nb: this pattern is equivalent to a conjunction of two modus ponens*	Constructive dilemma: $(p \rightarrow q) \wedge (r \rightarrow s),\ p \vee r \ \vdash q \vee s$ [G] Property Q or S is satisfied [S] Constructive dilemma [G] Property (P implies Q) and (R implies S) is satisfied [G] Property P or R is satisfied
Destructive Dilemma	Destructive dilemma: $(p \rightarrow q) \wedge (r \rightarrow s), \neg\, p \vee \neg s \ \vdash \neg p \vee \neg r$ [G] Property (not Q or not R) is satisfied [S] Destructive dilemma [G] Property (P implies Q) and (R implies S) is satisfied [G] Property not P or not S is satisfied

Strategy patterns for informal rules

Rewording	[G] Property P is satisfied [S] Reformulation [G] Property P(reworded) is satisfied [J] Property P(reworded) is semantically equivalent to P
Substitution	[G] Property P1 is satisfied [D] <P1>: is defined as P2 [S] Argument over the satisfaction of P2 [G] The property P2 is satisfied [J] We rely on the equivalence of P1 and P2 by definition
Compositionality	[G] Set S satisfies property P [S] Compositionality [J] Satisfaction of Q for each element s of S and Satisfaction of R for S implies satisfaction of P for S [G] s_1 in S satisfies P' ... [G] s_n in S satisfies P' [G] Set S satisfies R
Enumeration	[G] Set S satisfies property P [S] Enumeration [J] Satisfaction of Q for each element s of S implies satisfaction of P for S [G] s_1 in S satisfies P' ... [G] s_n in S satisfies P'
Partitioning	[G] Set S satisfies property P [S] Partitioning [G] The set is partitioned into subsets [C] The subsets, resulting from the partitioning [G] All subset satisfy the property P' [G] The satisfaction of P' for all subsets of S implies P
Contextualization	[G] Property P1 is satisfied

	[C] <C1>: context of application [S] Argument over the sufficiency of a property P2 in context [J] We expect P2 to be simpler to prove than P1 [G] The property P2 implies P1 in context C1 [G] The property P2 is satisfied
Propagation	[G] Property (concerning artefact A) P is satisfied [S] Propagation [G] Q (concerning another artefact B), implies P (in artefact A) [G] Property Q (concerning another artefact B) is satisfied

Notes:

- The rewording strategy is used to express the same property in different teams, for instance by replacing a word by its definition. A substitution is a special case of rewording where the semantical equivalence is guaranteed by the existence of a definition.
- Enumeration is a particular case of compositionality where the is no additional property R
- Partitioning is another case of compositionality where the elements of S are subsets.
- “Propagation" is a specific case of the modus ponens relation p→q, p ∴q where the implication p→q refers to properties concerning other artefacts of the workflow, possibly produced before or after the artefact concerned by the top-most goal. Please refer to Section "propagation of properties". In the following rule, property Q may be "identical" to property P in the sense that the only difference being P and Q being the artefact involved in the properties. For instance, property P is "(training dataset) is clean" and property Q is "(dataset before splitting) is clean". In both cases, the property is of the form "X is clean" where X is a specific artefact of the workflow. In that case, justification must be given the splitting activity preserves the "cleanness" property of the data.

B.3.2.4 Other strategies

Product-based or Process-based (Design vs. Evaluation)

```
[G] The property P is satisfied
    [S] Argument over the satisfaction of P on the <product> or on the <process>
       [D] <product>: Engineering item of interest for the given property
       [D] <process>: Sequence of activities generating the <product>
       [G] The process ensures the property P
       [G] The evaluation of the product demonstrates the property P
```

Note that intermediary verification steps along the chain of activities of the process are considered product-oriented steps.

Multi-legged argument

A multi-legged argument is aimed at increasing confidence in the satisfaction of goals by combining diverse argumentations paths. To some extent, it is similar to the "n-version programming" in software or 1-out-of-

2 architectures in system design: the two (or more) legs "are not supposed to fail at the same time". For more explanation on this concept, please refer to [Littlewood-07] and [Bloomfeld-03].

The multiple "legs" (especially "assumptions" and "evidences") of the reasoning are combined to demonstrate the satisfaction of a given goal. This is different from the usual goal to sub-goal decomposition because even if the confidence level for one leg is considered sufficiently high, this single leg will be sufficient to entail the parent goal. In other terms, the multiple-legs are not combined in a logical way to show the satisfaction of the parent goal.

As for any method based on diversity, confidence brought by multiple legs strongly depends on the independence of the legs.

```
[G] Property P is satisfied
   [S] Multi-legged argumentation
       [G] P1 and P2 and ... and Pn support P with an acceptable level of confidence
       [G] P1, P2,...,Pn are independent
       [G] Property P1 is satisfied
       [G] Property P2 is satisfied
       ...
       [G] Property Pn is satisfied
```

B.3.3 Core node – [SO] Solution

A **solution** refers to **evidence** that is deemed sufficient to establish the truth of the parent goal.

It "reflects the point when a goal needs no further refinement and can be directly supported by **evidence**" [Weinstock-07].

Exemple of solution: “The <attribution expert selection report> contains a section which describes the qualification of the domain experts”

Additionally, extensions of GSN allow to attach context elements in order to communicate the basis on which a piece of evidence (solution) supports its parent Goal [Graydon-14].

The evidence always refers to an engineering item which is produced by a dedicated activity in the workflow.

B.3.3.1 Writing Rules

[Rule 028] Solutions as artefacts

GSN solution elements should refer unambiguously and precisely to the section in an evidence item which is required to support the goal element. [GSN-21]

[Rule 029] Syntactic and semantic rules

- A solution starts with prefix [SO]
- Solutions shall be stated as noun-phrases.
- A solution shall refer to one or several piece of evidence.

It is possible to cite multiple solutions as providing evidential support for a particular parent goal.

B.3.3.2 Assurance Claim Points in Solutions

If there are doubts about the strength the evidence/Solution brings to the leaf-goal, additional argumentation, that increases our confidence, can be added in the form of Assurance Claim Points (ACP). An ACP attached to a link incoming to a Solution/evidence in particular yield an *asserted solution*. If true, it supports the belief in the other subgoals or in the argument. However, unlike conventional subgoals, if it is false, it does not invalidate the claim.

B.3.4 Auxiliary node – [C] Context

Along with definitions, **contexts** are used to remove ambiguities about a claim. Contexts "define or constrain the scope over which the claim is made" [GSN-21].

When defining the scope, there are three key roles of context elements:

- Providing information about the referred system;
- Providing information about the context of the system;
- Providing information that can help the comprehension of the argument (for example, definitions of terminology used, circumstances under which an argument was made, …).

B.3.5 Writing rules

GSN provides a set of grammar tips to write a Context element. However, it seems not to be sufficient when pursuing Rushby's conception of a MEANS clause. Nevertheless, in [Graydon-14], the author proposes a detailed definition of Context which augments the GSN guideline, while focusing on the use of contexts as explications. The following rules are extracted from [Graydon-14]:

[Rule 030] Context as explication

Context elements should be introduced as explications. That is, building a sentence/s that describes the added value of the term to the argumentation tree.

[Rule 031] Coherence

The applied context should not contradict or undermine the downstream argument. If so, the design of that argument should be reviewed.

[Rule 032] Context is not evidence

Context elements should not be used as evidence. All information that intend to support the validity of a goal shall be provided in the form of a Solution.

Comment: In our assurance cases, we generally use contexts to detail the expected content of an evidence. However in those cases, they only add information to the Evidence as explanations, they do not provide direct evidence.

[Rule 033] Non-circularity.

An explication in a context shall not contain the terms that the context is targeting.

As an addition for contexts, we provide the following rules that were produced during EC6.7 iterative reviews:

[Rule 034] Syntactic and semantic rules

- A context starts with prefix [C]
  - Context element must be an explication phrase, beginning with the <reference> to the item/document. For instance:

```
[G] The <test dataset>, <training dataset> and <validation dataset> are obtained by
splitting an <annotated dataset> constituted of <independent examples>
    [C] <test dataset>: The test dataset is used to evaluate the performances of the
    model. These tests can cover operational performances or robustness tests.
```

[Rule 035] Context of a Solution

The context of a Solution must be explicitly defined as:

```
[SO] <Name-of-artifact>
    [C] The <name-of-artifact> contains the …
```

Example:

```
[G] The method can produce the expected outputs

    [C] the <method application context> contains a description of the targeted <detection
    rate> and the associated confidence in relation with the <robustness objective> defined
    in the <robustness requirements>
    [C] the <method description> contains a section which describes the possible detection
    rate of the method as well as means to transpose and estimate the results for <method
    application context>
    [SO] <Method selection report>
        [C] the <method selection report> shows the <detection rate> of the method can meet
the <robustness objectives> defined in <method application context>
```

B.3.6 Auxiliary node – [D] Definitions

Definitions are special types of contexts, which are used to define terms used in a goal or any other element, to make the argument understandable and unambiguous.

GSN considers definitions as a Context element: "*Two kinds of GSN context statement exist. Where a context statement is a reference to an artefact …. Where a context statement draws attention to explanatory contextual information (such as the definition of some term), this information shall be stated briefly using complete sentences of a noun-phrase + verb-phrase structure.* "

However, in this guide we treat the use of Definitions as a different element to simplify the argument tree.

B.3.6.1 Writing rules

[Rule 036] Definition scope

The scope of a definition is global. It will be inherited upstream and downstream on the whole assurance case.

[Rule 037] Syntactic and semantic rules

- At each GSN element (Goal, Solution, Strategy …), the definition reference will be cited when placing *Chevron* brackets.
  - For instance:

    [G] The test dataset, training dataset and validation dataset are obtained by splitting an annotated dataset constituted of **<independent examples>**

    [D] <Independant examples>: ...
- A definition starts with prefix [D]
- A definition must have a unique name which can be referenced throughout the rest of the argument.

  For instance:

```
[G] The label conforms to the <reality>
    [D] <reality>: State of the world at the time of data capture
```

B.3.6.2 Writing suggestions

When writing a textual Assurance Case, a definition can be written in two different locations.

a) At the beginning of the Assurance. It can be useful for having a collection of all definitions as premises before starting the assurance case.

b) After each GSN element who cites a definition reference.

When to add a definition element

The use of a definition at each reasoning step is not compulsory. There will be some phrases which are self-explanatory and all involved parts will easily understand. On the other hand, sometimes additional information would be required to completely understand a predicate. In that case we would need to choose between a Definition or a Context element.

If the info to be augmented concerns an artifact from the Workflow, then we will use a Context element.

In case we just need to explain a concept or a part from a phrase without referencing any artifact, then a Definition will be used.

For better readability, definition should be added at the beginning of the AC whenever possible.

B.3.7 Auxiliary node – [A] Assumptions

An **assumption** is a statement, linked to a property, *considered* true.

These assumptions must be valid for the related goal/strategy to be valid. Assumptions can be documented explicitly in GSN using the assumption element. [GSN-21]

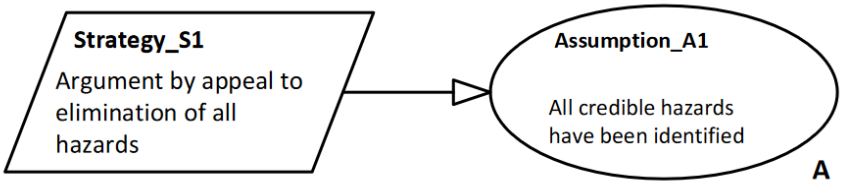


Figure 7: Assumption example. Extract taken from [GSN-21].

Assumptions adds conditions for the argument to be sound. [Rushby-15]

An assumption is added the node and "propagates" to the rest of the argument starting at that point.

B.3.7.1 Writing rules

[Rule 038] Assumptions as a true assertion [GSN-21]

An assumption is a premise asserted as true. It is not supported by further argument.

[Rule 039] Syntactic and semantic rules

An assumption starts with prefix [A]

For instance:

```
[G] The <ML system> maintains its <level of performance> under a set of <faults>
    [A] Only circumstances corresponding to the occurrences of <faults> are considered.
```

B.3.8 Auxiliary node – [J] Justifications

When the author wonder if a strategy or goal should be considered acceptable, then it can be explained through the use of Justifications in GSN.

B.3.8.1 Writing rules

[Rule 040] Justifications are informative [GSN-21]

A justification shall not alter the meaning of the goal or Strategy

[Rule 041] Scope [GSN-21]

Justifications are local to the element to which they are linked. They are not inheritable and they should be noted (or pointed) again if the same justification is needed elsewhere in the argument.

[Rule 042] Syntactic and semantic rules

A justification starts with prefix [J]

For instance:

```
[G] The method with weakest constraints on execution requirements is selected
    [J] Selecting the method with weakest constraints implies more flexibility and less
    requirements in resources
```

B.3.9 Confidence claims

Confidence claim points (ACPs) may be introduced to strengthen some aspects of the argument. Some authors (Hawking at al [27]) propose to separate the argument in two parts: the primary argument directly supports the top-level-goal, while the secondary argument based on ACPs is focused on the confidence of the argument.

Assurance claim points can be attached to several links on the assurance case, including links incoming to a strategy, goal, context or evidence. Even though there is not a consensual agreement about the uses of ACP, there is one function of the ACP that is accepted and seen as necessary: adding confidence to an evidential step.

In other terms, since an evidential step will be considered as inductive, it will be useful to use as many confidence arguments as possible to prove the cited evidence (Solution element in GSN) supports the associated Goal.

At this version of the guideline, the use of ACPs is not explored and supported by the Confiance.AI Assurance Cases of Batch 2. However, this section leaves open the possibility to integrate them in the future.

B.4 Patterns

This section presents a set of argumentation patterns that appear in a recurrent manner.

Each pattern is described according to the following schema (see next section for examples):

Name
Name of the pattern

Condition of application
When to use the pattern

Description
Detailed description of the pattern

Template
Textual description of the structure of the Assurance Case corresponding to this pattern

Context items and evidence items
Context and evidence items used by the pattern (as per GSN)

Example of instantiation
Example of instantiation of the pattern

Compact instantiation (optional)
Simplified version of the template instantiation

B.4.1 Procedure selection and application

Name

Pattern for selecting and using procedure or method

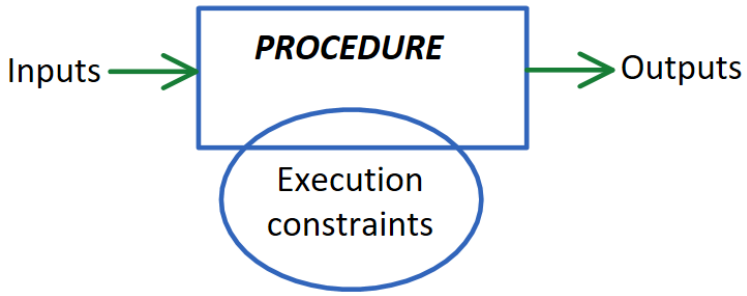
Condition of application

When a procedure or a method needs to be chosen among several and applied.

We consider to simplify that a method can also be seen as a procedure, in other terms as a set of rules to be followed to obtain a specific result, therefore the following applies for methods too.

Description

We consider that a procedure is a *set of rules* that consumes *inputs*, and produces *outputs* under *execution constraints*.



- In order to choose and apply a procedure, we need to obtain the list of procedures that may support the parent goal. For each candidate procedure, we need to verify:
  - Is the procedure suited for **inputs** in our settings?
  - Does it produce the expected outputs?
  - Are the **execution constraints** compatible with our settings?
- Then, the selected procedures must be compared, (typically according to the **execution constraints** setting or expected results) and the best one must be selected
- Finally, the best procedure must be applied on our target object (the subject of the property of the goal)

Template

```
[G] The best procedure is selected and applied
    [G] The list of applicable procedures is defined
        [C] the <procedure description> contains a section which provides a list of candidate procedures
        [S] Argument over each procedure of the list
            [G] The procedure is suited for inputs of the <procedure application context>
                [C] the <procedure application context> contains a description of the expected range of inputs
                [C] the <procedures description> contains a section which describes valid inputs for the procedure
                [SO] the <procedure selection report> shows that the range of inputs of <procedure application context> corresponds to valid
                   inputs
            [G] The procedure can produce the expected outputs
                [C] the <procedure application context> contains a description of expected outputs
                [C] the <procedures description> contains a section which describes outputs of the procedure
                [SO] the <procedure selection report> shows that the outputs of the procedure are compliant with the expected outputs
            [G] The execution requirements of the procedure are compatible with current execution constraints
                [C] the <procedure application context> contains a description of execution constraints
                [C] the <procedures description> contains a section which describes the execution requirements
                [SO] the <procedure selection report> shows that the execution requirements are compatible with the execution constraints
    [G] The best procedure of the list is selected
        [G]1 Procedures of the list are compared according to specific criteria
            [G]1 The procedures of the list are compared according to their execution requirements or expected results
                [G] The procedure selected is the one that best fits the criteria
                    [SO] the <procedure selection report> shows a comparison of results for each procedure
                [G] The procedure with weakest constraints on execution requirements is selected
                    [J] Selecting the procedure with weakest constraints implies more flexibility and less requirements in resources
                    [SO] the <procedure selection report> shows a comparison of execution requirements for each procedure
            [G] The procedures of the list are compared according to new criteria
                [C] the <procedure application context> describes the applicable criteria for selection of the best procedure (With order
                  relation)
                [G] The procedure selected is the one that best fits the criteria
                    [SO] the <procedure selection report> shows a comparison of all procedure for the applicable criteria
    [G] The best procedure is applied
        [SO] the <procedure application report> shows the results of application of the procedure
```

Context items and evidence items

There are **2 context items** required by this pattern:

- <procedure application context>, which contains the description of all the known aspects of the problem before application of the procedure, such as inputs, expected outputs, execution constraints, selection criteria for best procedure…
- <procedures description>, which is a set of description of each procedure of interest for current goal

There are **2 evidence items** required by this pattern:

- <procedure selection report>, which contains the elements supporting the choices made during the selection process, such as the validity and applicability of each candidate procedure to the main goal, the compatibility of execution constraints, and the comparison of selected procedure that led to the selection of the "best" method
- <procedure application report>, which shows that the best procedure was effectively applied

Example of instantiation

The following example covers the selection and application of the best method for detection of adversarial samples at runtime:

[G] The best method for **adversarial samples detection** at runtime is selected and applied
    [G] The list of applicable **adversarial sample detection methods** is defined
        [C] the <adversarial sample detection methods description> provides a list of candidate **adversarial sample detection methods**
        [C] <Input reconstruction>, <Feature squeezing>, <OOD-based detection>, as defined in <EC4_trust>
        [S] Argument over each **adversarial sample detection method** of the list
            [G] The **method** is suited for inputs of the <method application context>
                [C] the <method application context> specify the type and format of the data consumed by the model
                [C] the <method description> contains a section which describes valid inputs for the method
                [SO] the <method selection report> shows that the type and format of data corresponds to valid inputs for the method
            [G] The **method** can produce the expected outputs
                [C] the <method application context> contains a description of the targeted <detection rate> and the associated confidence in relation with the <robustness objective> defined in the <robustness requirements>
                [C] the <method description> contains a section which describes the possible detection rate of the **method** as well as means to transpose and estimate the results for <method application context>
                [SO] the <method selection report> shows the <detection rate> of the **method** can meet the <robustness objectives> defined in <method application context>
            [G] The execution requirements of the **method** are compatible with current execution constraints
                [C] the <method application context> specifies that the method must be applied *at runtime*, under specific *resource and timing constraints*, with a specific *version of the model*, on a specific *hardware*, while preserving the initial *performance of the model*
                [C] the <method description> contains a section which describes the execution requirements, including *resource and timing* aspects when applied on specific ***hardwares*** and its applicability on different ***types of models*** and phases of the <MLDL>, and the preservation of the ***performances of the model***
                [SO] the <method selection report> shows that the method can respect the execution requirements when applied with the specific model on the target hardware at runtime, and has no impact on the performances of the model
    [G] The best **adversarial sample detection method** of the list is selected
        [G]1 The **methods** of the list are compared according to specific criteria
            [G]1 The **methods** of the list are compared according to their execution requirements or expected results
                [G] The **method** with the best <detection rate> is selected
                    [SO] the <method selection report> shows a comparison of <detection rate> for each **method** for the <robustness objective>
                [G] The procedure with weakest constraints on execution requirements is selected
                    [J] Selecting the ****method**** with weakest constraints implies more flexibibility and less requirements in resources
                    [SO] the <method selection report> shows a comparison of execution requirements for each **method**
            [G] The **method** of the list are compared according to new criteria
                [C] the <method application context> describes the applicable criteria for selection of the best **method**
                [G] The **method** which best fits the criteria is selected
                    [SO] the <method selection report> shows a comparison of all **method** for the applicable criteria
    [G] The **adversarial sample detection method** is applied
        [SO] the <adversarial sample detection application report> shows the results of application of the **method

Compact instantiation

Instantiated pattern

"Procedure selection and application"

Subject

Adversarial samples detection methods at runtime

Context items

- The <method application context> describes:
  - The type and format of data consumed by the model
  - The targeted <detection rate> and the associated confidence, in relation with <robustness objective> defined in the <robustness requirements>
  - The context of application of the method: at runtime, under specific resource and timing constraints, on a specific hardware, with a specific type of model
  - The applicable criteria for the selection of the best method for the task
- The **<adversarial samples detection methods description>** contains:
  - A list of candidate methods for adversarial sample detection at runtime
  - From <EC4_trust>: <Input reconstruction>, <Feature squeezing>, <OOD-based detection>
  - A description of valid types of input data for each method
  - An estimation of the detection rate of each method
  - The requirements for each method in terms of resources, timing constraints when applied on specific hardwares, the type of models supported, and guarantees about the preservation of the model performances

Evidence items

- The **<method selection report>** contains:

  - A proof of compliance between inputs of the <method application context> and valid inputs for each method selected
  - A proof that the selected methods allows to reach the targeted <detection rate> with the appropriate confidence as defined in the <robustness objectives>
  - A proof that the selected methods can respect the execution requirements when applied with the specific model on the target hardware at runtime, and have no impact on the performances of the model
  - A comparison of all methods on the applicable criteria as defined in the <method application context>
- The **<method application report>** contains:
  - A proof that the method was effectively applied

Note

This pattern is especially useful when the IVVQ engineer is not interested in a detailed argumentation of each method. This situation can occur due to different reasons, for instance, the available methods are numerous and not yet mature.
However, once each method is mature and well-studied, we can develop a complete assurance case that argues the truth behind each selected method without needing to prove the ‘best method’. See the comparison between the Assurance Case Robustness from Batch#2 and Batch#3.

B.4.2 Verification approach selection

Name

Pattern for the use of test-based or formal verification methods

Condition of application

When the satisfaction of a property must be verified, be it by tests or using formal verification

Description

We consider that the satisfaction of a property can either be verified using a set of tests, or proven using formal methods, with a different level of confidence. For ML-based systems, the verification using tests may be defined using the test dataset, or by the definition of a new formal dataset dedicated to the verification of properties.

Template

```
[D] <SOE>: System of interest and its operating environment
[G] Property P is satisfied
    [S] Multi-legged argumentation
        [G] <Test-based verification> of P and <Formal verification> of P support the satisfaction of P with an
        acceptable level of confidence
        [G] <Test-based verification>: the satisfaction of P is verified by tests on an abstraction of the <SOE>
            [C] <Test inputs> /*test scenarios for classical test, or test data for ML-based systems*/
            [G] The <Test inputs> used are sufficient to verify the satisfaction of the property on the abstraction
            of <SOE>
                [SO] <Test specification report>
                    [C] The <Test specification report> shows that the test inputs are sufficient to cover the
                    satisfaction of property P
            [G] Verification test results show that property is satisfied
                [SO] <Verification test results>
            [G] The abstraction is sufficiently representative of the <SOE>
                [SO] <Abstraction specification report>
        [G] <Formal verification[2]>: the satisfaction of P is formally proven on a formal model representative of the
        <SOE>
            [G] P is covered by a set of <formally specified properties>
                [C] <Formally specified properties>
                [SO] <Formal properties specification report>
                    [C] The <Formal properties specification report> justifies that the properties are sufficient to
                    cover the satisfaction of property P
            [G] Formal verification demonstrates that the <formally specified properties> are proven
                [SO] <Formal verification results>
            [G] The formal model is sufficiently representative of the <SOE>
                [SO] <Formal model specification report>
```

[2] This argumentation is partial. A formal method relies on a model M that is an abstraction of the system of interest S. For the results of formal verification to be useful, one shall demonstrate that the abstraction preserves property P so that if P is true for M then it is also true for S.

Context items and evidence items

There are **2 context items** required by this pattern:

- `<Test inputs>`, which are used to verify the property, which can be test scenarios of test data for ML-based systems
- `<Formally specified properties>`, which are used as a formal representation of the property to verify and can be proven using formal methods

There are **6 evidence items** required by this pattern:

- `<Test specification report>`, which shows that the test inputs are sufficient to cover the satisfaction of the property
- `<Verification test results>`, which presents the results of the test-based verification activity
- `<Abstraction specification report>`, which describes the model of the system and how it is sufficiently representative in regards to the property
- `<Formal properties specification report>`, which justifies that the formal properties are sufficient to cover the satisfaction of the property
- `<Formal verification results>`, which presents the results of the formal verification activity
- `<Formal model specification report>`, which describes the formal model of the system and how it is sufficiently representative in regards to the property

B.4.3 Expert judgment

Name

Pattern for using expert judgment as the main source of evidence for a goal

Condition of application

When there is no other way of verifying a subgoal and we need to ensure the validity of the expert judgment

Description

In [Knight-15], the author describes the expert as someone who uses inputs provided by a *specialist*, and determines if the inputs are sufficient to provide a judgment. If not, he might require additional inputs, possibly from additional specialists.

In this description, the *specialist* is in charge of acquiring, analysing and interpreting evidences to provide inputs to the expert for the judgment process.

Template

```
[G] Property P is satisfied
    [S] Argument by expert judgment
```

```
[G] The team validating the property is adequate
    [G] The expert is legitimate
        [SO] <Property verification team selection report>
            [C] The <property verification team selection report> contains a section which
            describes the qualification of the expert for the property of interest
    [G] The expert relies on a team of appropriate specialists for the property of interest
        [SO] <Property verification team selection report>
            [C] The <property verification team selection report> contains a section which
            describes the qualification of the specialists for the property of interest
[G] The inputs provided are sufficient to provide a judgment
    [G] The team of specialists provided the required inputs
        [SO] <Property expert report>
            [C] The <property expert report> details the acquisition, analysis and
            interpretation by the specialists of the inputs provided to the expert
[G] The expert judges that the property is satisfied
    [SO] <Property expert report>
        [C] The <property expert report> contains the judgment of the expert which
        validates the property
```

Evidence items

There are 2 **evidence items** required by this pattern:

- The `<property verification team selection report>`, which contains:
  - A section describing the qualification of the expert for the property of interest
  - A section describing the qualification of the specialists for the property of interest
- The `<property expert report>`, which details:
  - The acquisition, analysis and interpretation performed by the specialist of the inputs provided to the expert
  - The judgment of the expert which validates the property

Example of instantiation

The following example covers the validation of the precision of the <ODD description>:

```
[G] <Precision>: The <ODD description> is <precise>
    [D] <Precise>: The <ODD description> excludes every situation in which the system is unable to
operate
    [S] Argument by expert judgment
        [G] The team validating the precision of the ODD description is adequate
            [G] The expert is legitimate
                [SO] <ODD precision verification team report>
                    [C] The <ODD precision verification team report> contains a section which describes
                      the qualification of the expert for the estimation of the precision of the ODD
                      description
            [G] The expert relies on a team of appropriate specialists for the property of interest
                [SO] <ODD precision verification team report>
```

```
                [C] The <ODD precision verification team report> contains a section which describes
                 the qualification of the specialists for the estimation of the precision of the ODD
                 description
    [G] The inputs provided are sufficient to provide a judgment
        [G] The team of specialists provided the required inputs
            [SO] <ODD precision expert report>
                [C] The <ODD precision expert report> details the acquisition, analysis and
                  interpretation by the specialists of the inputs composing the <ODD description>
                  provided to the expert
    [G] The expert judges that the precision of the ODD description is sufficient
        [SO] <ODD precision expert report>
            [C] The <ODD precision expert report> contains the judgment of the expert which
              validates the precision of ODD description
```

Compact instanciation

Instanciated pattern

- "Expert judgment"

Subject

- Validation of the precision of the ODD description

Evidence items

- The <ODD precision verification team report> contains:
  - A section which describes the qualification of the expert and of the specialists for the estimation of the precision of the ODD description
- The <ODD precision expert report> contains:
  - Details about the acquisition, analysis and interpretation by the specialists of the inputs composing the <ODD description> provided to the expert
  - The judgment of the expert which validates the precision of ODD description

C. Conclusion

In conclusion, this document has proposed a list of recommendations, techniques, patterns, and rules to guide the construction of Assurance Cases. These recommendations are intended to simplify the writing of Assurance Cases and improve their consistency. The rules are presented with a specific format and numbering, and are recommended both for the building of Assurance Cases and for the reviewing process, where they should be verified.

However, it is important to note that even though these recommendations provide a common framework, there is still flexibility in how arguments are built. Thus, different practitioners may have different styles and preferences for how to construct arguments. That is why this guideline cannot eliminate the need for review meetings. While the guideline can help to speed up the consensus process, it is still important to have a group of experts review the Assurance Case to ensure that it is sound and complete.

For more information on the evaluation process of an assurance case, please see document “613C – Methodological Guideline for Assurance Cases Evaluation”.

D. Bibliography

[AMLAS] Richard Hawkins, Colin Paterson, Chiara Picardi, Yan Jia, Radu Calinescu, and Ibrahim Habli. “Guidance on the Assurance of Machine Learning in Autonomous Systems (AMLAS).” Assuring Autonomy Program, March 2021.

[Bloomfeld-03] R. Bloomfield et B. Littlewood, « Multi-legged arguments:the impact of diversity upon confidence in dependability arguments », in 2003 International Conference on Dependable Systems and Networks, 2003. Proceedings., juin 2003, p. 25-34. doi: 10.1109/DSN.2003.1209913.

[Bloomfield-14] Robin Bloomfield et al. Building Blocks for Assurance Cases. 2014 IEEE International Symposium on Software Reliability Engineering Workshops.

[Graydon-14] Patrick John Graydon. Towards a clearer understanding of context and its role in assurance argument confidence. In SafeComp 2014: Proceedings of the 33rd International Conference on Computer Safety, Reliability, and Security, Volume 8666 of Springer-Verlag Lecture Notes in Computer Science, pages 139–154, Florence, Italy, September 2014

[GSN-21] The Assurance Case Working Group (ACWG). “Goal Structuring Notation Community Standard Version 3.” . . May, 2021, 130.

[Holloway-16] C. M. Holloway, ‘Understanding Assurance Cases: An Educational Presentation in Five Parts Module (V1)’,

[Kelly-07] Kelly, Tim. “Reviewing Assurance Arguments-a Step-by-Step Approach,” January 1, 2007.

[Knight-15] J. C. Knight and P. J. McGee, ‘Expert judgment in assurance cases’, in 10th IET System Safety and Cyber-Security Conference 2015, Bristol, UK, 2015, p. 6 .-6 . doi: 10.1049/cp.2015.0273

[Littlewood-07] B. Littlewood et D. Wright, « The Use of Multilegged Arguments to Increase Confidence in Safety Claims for Software-Based Systems: A Study Based on a BBN Analysis of an Idealized Example », IIEEE Trans. Software Eng., vol. 33, no 5, Art. no 5, mai 2007, doi: 10.1109/TSE.2007.1002.

[Rushby-15] Rushby, John, Xidong Xu, Murali Rangarajan, and Thomas L Weaver. “Understanding and Evaluating Assurance Cases,” n.d., 136.

[Weinstock-07] Weinstock, Charles B, Howard F Lipson, and John Goodenough. “Arguing Security – Creating Security Assurance Cases,” n.d., 22.

Title : Methodological guidelines for Assurance Cases

Keywords : Assurance cases, guidance

This document presents the guidelines for the design of Assurance Cases that have been applied in the context of Confiance.IA.

Our partners