On Safety Assurance of Symbolic Artificial Intelligence

M. Guzman[1*], R. Bouchekir[1*], A. Cook[2], and J. Haindl[2]
[1]fortiss GmbH, An-Institut Technische Universität München, Guerickestraße 25, 80805 München, Germany
[2]Third party, Munich, Germany
{guzman, bouchekir}@fortiss.org, alasdaircook2014@gmail.com, johannes.haindl@gmx.de
*corresponding authors

Abstract—Artificial Intelligence (AI) has gained popularity in recent years. Symbolic Artificial Intelligence (symbolic AI), a subset of AI, holds significant promise, particularly in avionics software, offering potential enhancements to various aspects of its functionality, such as automatic decision inference or generating optimal decisions. Despite its benefits, the integration of symbolic AI into avionics software introduces distinctive challenges, particularly in ensuring the safety and reliability of software powered by this technology. In this paper, we propose a set of supplementary objectives intended to support the DO-178C standard, specifically addressing features associated with symbolic AI. Additionally, we outline a set of metrics designed to support the generation of evidence for the proposed objectives. To illustrate the proposed assurance objectives, we made use of a TPN solution finder, which employs symbolic AI techniques to generate optimal routes for remotely piloted aircraft.

Keywords–Symbolic AI; DO-178C; Formal Verification; Safety Assurance; Dependability Metrics; Temporal Plan Network (TPN).

1. Introduction

The development of avionics software products has to conform to the DO-178C [17] standard. DO-178C provides a set of *certification objectives* aimed at guiding the development of avionics software products. The results of all development processes must be then verified for compliance with the certification objectives. Verifying software products can be a complex task. The verification, according to DO-178C, is performed by reviews, analysis, or tests. However, when using formal methods, DO-178C recommends to make use of the guidelines provided by RTCA DO-333 [16], which is a supplement to DO-178C. DO-333 identifies the modifications and additions to DO-178C objectives, activities, and software life cycle data that should be addressed when formal methods are used as part of the software development process. It includes artifacts that would be expressed using some formal notation and the verification evidence that could be derived from them.

Nowadays AI technology is ubiquitous in the development of industrial products and avionics software products are not an exception [9], [11], [3]. In particular, symbolic artificial intelligence (symbolic AI) techniques, have been used actively to develop efficient planning components for aircraft systems [2]. Symbolic AI involves representing knowledge in a structured symbolic form to mimic the human cognitive processes [10]. In this paper, we consider two types of symbolic AI, *search-based AI methods* and *expert systems*. Search-based AI methods employ algorithms to explore solution spaces systematically and generate an *optimal* solution. Expert systems use knowledge representation and inference rules to emulate human expertise in specific domains, facilitating problem-solving and decision-making tasks.

The current standards in avionics, including DO-178C and DO-333, do not dedicate any objectives specifically for symbolic AI. A recent concept paper from the European Union Aviation Safety Agency (EASA) proposes an AI road-map that demonstrates a possible way towards certification for AI applications in aviation [9]. However, this concept paper does not cover symbolic AI components.

In this paper, we analyse the adequacy of the DO-178C standard guidelines for the verification and validation of symbolic AI components. We show that specific objectives regarding the main features of symbolic AI can be proposed to help the design, implementation and validation of AI components. We shall apply our findings in a use case considering a path planning component making use of symbolic AI techniques.

The contributions of the paper are summarized as follows:

- We propose a series of objectives we believe have to be considered in the verification and validation of search-based AI components.
- We provide as series of objectives that we think should be considered in the DO-178C guidelines to account for verification and validation of expert systems.
- We propose a series of dependability attributes and specific metrics/tools that can be used to account for the proposed objectives.
- As an example, we applied some of the proposed metrics to cover the objectives regarding the heuristics used in a $A*$-like algorithm. The $A*$-like algorithm and the heuristic are used in the implementation of a route planner.

The structure of the paper is as follows. We shall present a background with the current avionics certification standards, how to use formal methods as means of compliance and the main features of symbolic AI. Then, we shall present some of the proposed symbolic AI objectives that we believe shall be taken into account in the design and implementation processes. Then, for each of the proposed symbolic AI main features, we present dependability attributes and metrics that could work as means of compliance. We then apply some of the proposed metrics in a use case considering a planning

component. Finally, we provide a section with the related work and concluding remarks.

2. Background

2.1 Current Avionics Certification Standards

The guideline DO-178C identifies key steps inside the development process and defines certification objectives for each of these steps. This standard considers four main development processes, (i) the software requirements process develops high-level requirements (HLR) from the outputs of the system process, (ii) the software design process develops low-level requirements (LLR), and software architecture from the HLR, (iii) the software coding process develops source code from the software architecture and LLR, and (iv) the software integration process loads executable object code into the target hardware for hardware/software integration. The results of each development process must be verified. DO-178C defines verification objectives for each step of the development, typically some objectives are defined on the output of a development process itself and also on the compliance of this output to the input of the process that produced it. For example, LLR shall be accurate and consistent, compatible with the target computer, verifiable, ensure algorithm accuracy, and LLR shall be compliant and traceable to HLR.

For formal verification methods, the DO-333 standard, supplement to DO-178C and DO-278A, provides guidance for software developers making use of formal methods in the certification of airborne systems and air traffic management systems. The supplement identifies the modifications and additions to DO-178C objectives, activities, and software life cycle data that should be addressed when formal methods are used as part of the software development process.

Although DO-178C and DO-333 have been successfully used in the certification of avionics software in the past decades, they do not provide any specific objective for AI-based software. The nature of AI-based software introduces challenges in meeting the DO-178C objectives, particularly in specifying requirements and ensuring the verification and traceability of such systems. While DO-178C may not be directly applicable to AI software, there are ongoing efforts within the avionics industry to develop guidelines and standards specifically tailored for the certification of AI in safety-critical systems. Organizations like the European Union Aviation Safety Agency (EASA) and the Federal Aviation Administration (FAA) are actively exploring ways to address the challenges associated with certifying AI-based systems in aviation. In particular, the recent EASA AI concept paper proposes an AI road-map that demonstrates a possible way towards certification for AI-based software in aviation [9].

2.2 Formal Methods as Means of Compliance

Formal methods can be used in each phase of the development cycle to provide the required analysis and proofs needed for certification [18], [5]. In Fig. 1, a description of the use of formal methods in the different development phases is presented. Once the Concept of Operations (ConOps) has been created, we obtain the system requirements which shall work as a reference for the generation of the software high-level requirements. Testing and formal methods tools can be used to check and guarantee the consistency, safety, and compliance of the software's high-level requirements w.r.t the system requirements. From the Software's high-level requirements, we shall obtain the low-level requirements that would specify the functionalities of the components. Formal methods can be used in this step to state the need for accuracy and consistency. From the low-level requirements, we can generate the source code implementing the functionalities of the components. Formal methods would check the accuracy, safety, security, and compliance of the generated code w.r.t the requirements.

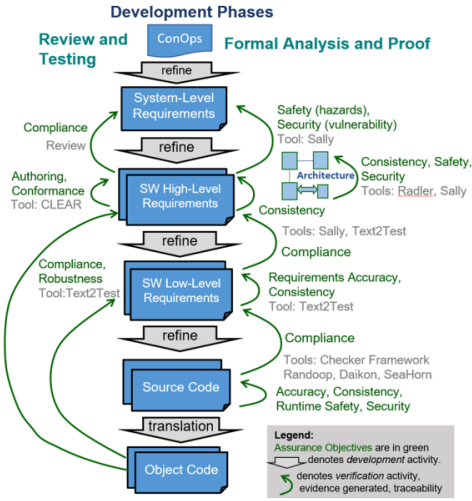

Figure 1. Potential use of formal methods and tools for the generation of assurance artifacts throughout software development (Image extracted from [18].

Some of the formal methods and tools that can be used in the different steps of the design workflow are listed in Table I. Even though we can use formal methods to provide evidence for the different development phases of the component, in this paper, we focus on the use of formal methods e.g., Theorem proving and SMT-based verification, to provide evidence for the *"software low-level requirements"* compliance checking.

Phase	**Tools**	**Artifacts**
Tool Qualification	Self-Analysis	Test+Analysis
System Requirements	Operational Scenarios	Consistency Completeness
Hazard Analysis	Sally	Model Checking
Software High Level Requirements	**PVS**	Consistency Completeness Validation
Software Low Level Requirements	**PVS** **SMT (Z3)**	Static/Dynamic Analysis
Executable Object Code	BeepBeep3 Text2Test	Safety Monitoring

TABLE I
AUTOMATION SUPPORT TOOLS FOR ASSURANCE ARTIFACTS IN SOFTWARE DEVELOPMENT PHASES.

2.2.1 SMT-based Verification

Satisfiability Modulo Theories (SMT) is a formal verification method that combines propositional logic (SAT solving) with specialized theories to handle specific data types and operations. SMT-based verification is widely used in various domains, e.g. hardware and software verification. SMT solvers, including Z3 [15], support arithmetic theories, enabling reasoning about integer and real number constraints and operations. For instance, the low-level requirements (the properties to verify) and the implementation could be formally specified as Linear Real Arithmetic (LRA) predicates and constraints. The final system is represented as a conjunction of the system (implementation) and the negation of the property (LLR) and is verified using Z3 to determine satisfiability i.e., the presence of a model satisfying the conjunction of constraints (a counter-example).

2.2.2 Theorem Proving

Another formal verification approach considered in this paper is theorem proving, which is a formal and systematic method of establishing the truth or validity of mathematical statements or propositions. For software verification, theorem proving helps us to provide mathematical proofs for the properties we want to verify over the system design and implementation. Program Verification System (PVS) [7] is a well-known theorem-proving tool used to assist with the verification process. PVS provides a collection of libraries with many proven properties, e.g. the trigonometric library, which facilitates the generation of the proofs, allowing us to focus on the details of the design and implementing verification.

2.3 Symbolic AI

We are interested in working with components making use of symbolic AI [10]. Symbolic AI is the part of the AI field focused on processing and manipulating symbols and concepts rather than data. In this work, we are considering two symbolic AI techniques that have been successfully applied in safety-critical systems (mainly for decision-making tasks), namely, *expert systems* and *search-based AI methods*. Algorithms making use of symbolic AI are designed to systematically explore and navigate through a search space to find optimal solutions for a given problem. Expert systems and search-based AI methods play a crucial role in various safety-critical applications, including for example Unmanned Aircraft Vehicle (UAV) route planning.

2.3.1 Expert Systems

Expert systems can be defined as mechanisms to simulate how an expert human works. Modelling the mental process of human experts involves extracting the knowledge and skills from the experts to create rules that could be used in inference tasks.

As illustrated in Fig. 2, the main components of an expert system are: The *User interface* which allows communication between the user and the expert system. The *Inference engine*

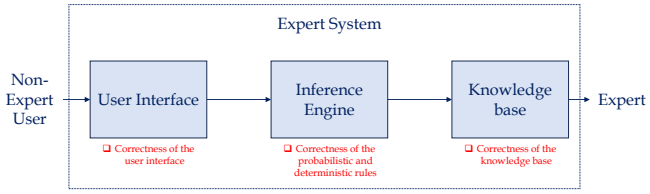

Figure 2. Main components to consider for the verification and validation of expert system.

: Uses information from the knowledge base to reach conclusions about different situations. It uses various inference mechanisms, such as rule-based reasoning, fuzzy logic, or probabilistic reasoning, to reach conclusions [10]. *Knowledge Base:* which is a repository of information, data, facts, and rules that represent the expertise in a specific domain. The knowledge base is created with the help of human experts and is used by the expert system to draw conclusions and make decisions.

2.3.2 Search-base AI Methods

In this paper, we shall define an search-based AI method as a method that makes use of *search algorithms*, e.g., A^*, together with *an heuristic*, to explore and navigate the search spaces to find optimal or satisfactory solutions to problems [10]. Search-based AI methods are usually considered as a subset of symbolic AI. The key components of search-based methods, as described in Fig. 3, correspond to the use of *domain knowledge*, containing information regarding the environment, a *heuristic* that shall guide the system in exploring the search space and finding a solution, and *special constraints*, which are specific to the system being analyzed.

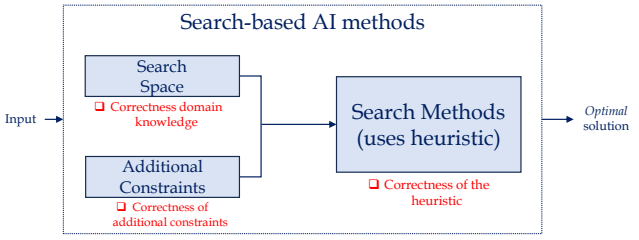

Figure 3. Main components to consider for the verification and validation of search-based AI methods.

3. Symbolic AI Objectives

In this section, we illustrate the additional objectives associated with symbolic AI. An intriguing question arises when applying DO-178C to provide safety evidence for symbolic AI: Are the existing objectives within the available guidelines sufficient to substantiate the safety of this technology? If the answer to this question is no, the subsequent question arises: *How do we affirm the safety of software-driven by symbolic AI? Is it necessary to introduce additional objectives?*

It could be postulated that as long as requirements are accurately specified, DO-178C objectives remain adequate, given their technology-agnostic nature. However, to ensure traceability and facilitate the design and verification, it may be necessary to define additional objectives specific for such technology.

3.1 Additional Objectives - Symbolic AI

The intrinsic nondeterministic and probabilistic characteristics of components employing symbolic AI present a complex challenge in ensuring safety guarantees. DO-178C currently lacks specific objectives tailored for such technology. A primary goal of this standard is to ensure traceability from high-level requirements to low-level requirements and, subsequently, to source code. However, given that it does not have specific objectives regarding symbolic AI, it can be challenging to cope with traceability requirements. Thus, it is imperative to introduce additional sub-objectives specifically tailored for symbolic AI. These sub-objectives should be incorporated into DO-178C Table A2 software development process, and Table A5 software coding and integration processes.

DO-178C Table A2 includes seven objectives such as the development and derivation of high-level requirements, the development of software architecture, and the development of low-level requirements. In this table, the following functionalities related to symbolic AI components need to be included:

- ***Knowledge Base (KB) & Domain Knowledge (DK):*** Ensure the inclusion of KB and DK, representing information crucial for the inference or search step within the system.
- ***Inference Engine (IE) & Search Method (SM):*** Account for the probabilistic behavior of the Inference Engine (IE) and Search Method (SM) during the decision-making process.
- ***Domain-Specific Constraints:*** Incorporate considerations for domain-specific constraints, encompassing any additional constraints pertinent to the specific domain under consideration.

Moreover, within DO-178C Table A2, we need to consider the safety assessment for the functionalities we have specified for symbolic AI components:

- Account for the safety of the KB/DK. Estimation of the hazards that imply having a failure or a wrong KB/DK.
- Account for the safety of the IE/SM step. The hazard estimation should also encompass potential risks arising from failures or inaccuracies in the IE/SM.

The additional sub-objectives related to symbolic AI must be interconnected with the objectives outlined in Table A2, specifically focusing on the following key objectives listed in Table II.

Table A2 Objcetives	**Additional sub-objectives related to Symbolic AI**
- A2-2: Derived high-level requirements are defined and provided to the system process, including the system assessment process. **- A2-3**: Software architecture is developed. **- A2-4**: Low-level requirements are developed. **- A2-5**: Derived low-level requirements are defined to the system processes, including the system safety assessment process.	- Account for the knowledge base and domain knowledge, containing information crucial for the inference or search step. - Account for the probabilistic behavior of the inference step. - Account for the domain-specific constraints. - Account for the safety of the knowledge base or domain knowledge. - Account for the safety of the inference step.

TABLE II
THE ADDITIONAL SYMBOLIC AI SUB-OBJECTIVES AND CORRESPONDING OBJECTIVES IN DO-178C TABLE A2.

DO-178C Tables A3 and A4 have to do with compliance of the requirements and design aspects generated in Table A2. In DO-178C Table A5, we have to add the following sub-objectives to account for the verification of the output coming from the source code generation:

- *Heuristics are admissible*
- *Heuristics are complete and terminate*
- *Heuristics are accurate (if not admissible)*
- *Domain knowledge is correct*
- *Domain knowledge is complete*
- *Domain knowledge is accurate*
- *Additional specific constraints*

The sub-objective regarding symbolic AI shall be added to the following objectives in DO-178C Table A5:

Table A5 Objectives	**Additional sub-objectives related to Symbolic AI**
- A5-1: Source code complies with low-level requirements - **A5-2**: Source code complies with software architecture **- A5-5**: Source code is traceable to low-level requirements	- Heuristics are admissible - Heuristics terminate - Heuristics are accurate (if not admissible) - Domain knowledge/knowledge base is correct - Domain knowledge/knowledge base is complete - Domain knowledge/knowledge base is accurate

TABLE III
THE ADDITIONAL SYMBOLIC AI SUB-OBJECTIVES AND CORRESPONDING OBJECTIVES IN DO-178C TABLE A5.

4. METRICS

In this section, we outline the metrics that can aid in substantiating the evidence for the sub-objectives introduced in Section 3. We establish a set of dependability attributes designed to assess the safety and reliability of components leveraging symbolic AI techniques. The dependability attributes we shall focus on are the following:

- ***Completeness:*** In this context, completeness is interpreted as having sufficient information to generate the desired result.
- ***Correctness:*** Correctness refers to the degree of compliance in the implementation of the component with respect to the required or expected behavior.
- ***Admissibility:*** Admissibility pertains to the ability of heuristics to provide an optimal solution.
- ***Termination:*** Termination relates to the ability of heuristics to provide a solution within a reasonable time frame.

- ***Accuracy:*** In case correctness proofs cannot be provided, we shall make use of metrics stating the accuracy of the implemented algorithms.

We illustrate the Symbolic AI features together with the proposed dependability attributes and metrics in Table IV.

Symbolic AI features	**Dependability attributes**	**Metrics**
Knowledge Base Domain Knowledge	Completeness	Scenario coverage
	Correctness	Domain accuracy
	Specific requirements	Depends on specific constraints
Rules (heuristics)	Admissibility	Formal methods, e.g., theorem proving
	Termination	Formal methods, e.g., theorem proving
	Accuracy	Accuracy rate
Domain specific constraints	Correctness	Depends on specific problem

TABLE IV
SYMBOLIC AI FEATURES WITH DEPENDABILITY ATTRIBUTES AND METRICS.

These dependability attributes serve as a means of articulating the safety and reliability of the symbolic AI features, namely domain knowledge (knowledge base), inference rules (heuristics), and specific constraints. Our objective is to propose specific metrics and procedures for each dependability attribute, and thus facilitating the generation of assurance evidence for these attributes.

4.1 Knowledge Base & Domain Knowledge

4.1.1 Completeness of KB & DK

To ensure the safety of a component making use of symbolic AI techniques, we need to show the pertinent evidence of the completeness of the Knowledge Base or Domain Knowledge. Recall that completeness in the case of the KB & DK corresponds to considering all the necessary information for the component to provide adequate results. For example, for data completeness, a measure that could be used is the one of scenario coverage. In that regard, the authors of [4] proposed a scenario coverage metric that is based on the concept of *2-projection* and is tightly connected to the existing work of combinatorial testing, covering arrays and their quantitative extension. Computing the scenario coverage metric requires to have a labeled dataset such that for each data point it can be determined whether it is used in a certain scenario. In [4] they compute $Mscene$ by preparing a table recording all possible pairs of operating conditions, followed by iterating each data point to update the table with occupancy. Then, they compute the ratio between occupied cells and the total number of cells. The following equation summarizes the formula:

$$Mscene = \frac{Nr.CellsOccupiedByTheDataset}{Nr.CellsFrom2ProjTable} \quad (1)$$

4.1.2 Correctness of KB & DK

The verification task for the correctness of the KB involves ensuring that the KB accurately captures the intended specifications defined by the expert. In the validation process, domain validation is employed to verify the accuracy of the expert's knowledge about the target domain. Concurrently, procedural validation assesses the correctness of the procedures delineated in the KB to derive valid conclusions. Moreover, procedure optimization endeavors to ascertain that the procedures outlined in the KB are not only acceptable but also meet the desired optimization criteria. A key method for accomplishing this task is through a comprehensive review process.

To ensure the correctness of DK, a useful metric involves computing the number of inaccuracies concerning the data points or entries within the knowledge base. This can be quantified through metrics such as the percentage rate of inaccuracies, aiming for a lower error rate and higher accuracy. These measures provide a quantitative assessment of the fidelity of the domain knowledge, allowing for a more precise evaluation of its correctness.

$$Correctness = Accuracy = \frac{NrInAccuracies}{NrDataPoints} \quad (2)$$

4.1.3 Specific Requirements for KB & DK

The specific requirements metrics for the KB & DK are related to constraints that the data in the KB & DK has to satisfy. The choice of metrics depends on the nature of these constraints, and as a result, distinct metrics and procedures should be employed to assess compliance with specific requirements. The goal is to tailor the assessment to the unique characteristics and constraints associated with the KB and DK.

4.2 Inference Rules & Heuristics

When using a heuristic, it becomes imperative to generate evidence regarding the *admissibility, termination*, and *correctness* of the heuristics used. Admissibility in this context refers to the utilization of heuristic methods that yield an optimal or near-optimal result. Affirming the admissibility of a specific heuristic often requires a formal proof, achieved either through manual verification or with the assistance of a theorem-proving tool. This rigorous process involves demonstrating, in a formal and systematic manner, that the heuristic consistently provides solutions that are optimal or near-optimal under the specified conditions. Such formal proofs add a layer of confidence and assurance to the claim of admissibility for the heuristic being employed. Likewise, asserting the *termination* of heuristics—ensuring that a solution is consistently provided through their application—requires the provision of formal proof. This proof can be established either through manual verification or by utilizing a theorem-proving tool. The formal demonstration aims to validate that the heuristic process reliably concludes within a reasonable time frame, contributing to the overall confidence in the termination properties of the employed heuristics.

When working with non-admissible heuristics, it becomes necessary to present accuracy and precision metrics allowing us to state an interval of confidence. In [8] the authors propose to use a measure of accuracy as the distance between the heuristic value and the actual value by a multiplicative factor.

Definition 4.1. *Let h be a heuristic function on a search space (G, c, x_0, S). For any non-solution node x, we define*

the accuracy rate *of* h *at node* x *to be*

$$ARN(h,x) \stackrel{def}{=} \frac{h(x)}{h*(x)} \tag{3}$$

The accuracy rate of h*, denoted* $ARN(h)$ *, is the average of* $ARN(h,x)$ *for all non-solution nodes* $x \in V(G) \setminus S$*.*

This notion of accuracy rate is particularly meaningful for admissible heuristics. Intuitively, if h is admissible, then the larger $ARN(h,x)$, the more accurate the heuristic is at node x. The accuracy rate is, in fact, related to the how informed admissible heuristics are: for any two admissible heuristics h_1 and h_2 on the same search space, h_1 is more informed than h_2 iff $ARN(h_1,x) > ARN(h_2,x)$ for all non-solution node x.

4.3 Domain Specific Constraints

To address the specific constraints employed in a component driven by symbolic AI techniques, the metrics would be intrinsically tied to the nature of the constraints themselves. Therefore, these metrics cannot be predefined but should be tailored to the specific requirements and characteristics of the specific constraints. This approach ensures a customized and context-sensitive evaluation of how well the symbolic AI component adheres to the imposed constraints, allowing for a more accurate and relevant assessment.

5. Use Case: TPN solution finder

5.1 Concept of Operations

Unmanned Aerial Vehicles (UAVs) have been employed in a diverse range of applications from cargo transportation, to search-and-rescue operations or environmental monitoring [1]. Remotely Piloted Aircraft (RPA) is a UAV that is piloted from a remote pilot station. A main component of RPA is flight planning which generates a *safe* and **optimal** route to be followed by the RPA. The flight planning component considered in this paper is provided by an industrial third party and it is based on the Temporal Plan Network (TPN). TPN solution finder makes use of search-based AI algorithms to find an optimal route, following a sequence of predefined waypoints, and preventing the RPA from traveling into unwanted areas called no-fly-zones (NFZ). NFZ is a term related to the zones in the airspace that are not available to fly through due to some temporary (e.g. because of weather conditions) or permanent restrictions (e.g. because of a military base in the area). Therefore, as part of the mission configuration, the TPN solution finder has NFZ pre-specified. These NFZ can be of two types:

- ***Static NFZ:*** corresponds to restricted parts of the airspace that cannot be violated due to, typically, restrictions on the ground. For example, due to governmental buildings on the ground, military bases, airports, etc.
- ***Dynamic NFZ:*** corresponds to restricted parts of the airspace that cannot be violated due to adversarial weather conditions, e.g. strong winds or an area affected by bad weather. As for Static NFZ, dynamic NFZ are going to be represented as geo-fences containing the coordinates of the restricted moving area, the azimuthal angle, and the velocity.

More concretely, the TPN solution finder makes use of an $A^* - like$ algorithm to find the best route and suggests it as the flight plan the RPA should take from a given starting point to the goal. The TPN solution finder makes use of a formalism called *Temporal Plan Networks* (TPN) which is based on the work of reactive systems [14]. TPN are extensions of *Simple Temporal Networks* (STN), augmented with symbolic constraints and decision nodes. The nodes of a TPN represent temporal events, whereas the edges represent temporal relations. The TPN solution finder checks also for geo-fence constraints to be satisfied. Such constraints come from static NFZ and dynamic NFZ.

5.2 Safety analysis

Safety analysis within the context of DO-178C encompasses various critical elements, such as safety assessment and the establishment of safety objectives. It is important to clarify that the intention of this section is not to completely outline every aspect of the safety analysis for the given use case. Instead, the focus is on elucidating specific safety objectives in accordance with the prevailing standards. One safety objective related to our use case is *route verification*, which implies that the planned routes generated by the TPN solution finder adhere to predefined safety constraints and regulations. For instance, article 15 [6], GM3 Article 3(4) U-space airspace (Internal Geographical Zones) specifies sub-geographical zones where RPA operations are prohibited (e.g. static NFZ). In addition, for safe operation, the TPN solution finder should avoid dynamic NFZ, e.g., zones with bad weather conditions. We summarized in Table V the dependability attributes related to the TPN solution finder, focusing on the search-based functionality.

TPN solution finder features	Dependability attributes	Description
Knowledge Base	Complete	The waypoints, temporal and geo-fences constraints are complete
	Correct	The waypoints, temporal and geo-fences constraints are correctly defined
	Specific requirements	Geo-fence constraints should be specified as polygons. Temporal constraints should be specified in seconds for dynamic NFZ.
Heuristic used in search method	Admissible	TPN solution finder should generate an optimal or near-optimal route
	Terminate	TPN solution finder should generate a safe route within a reasonable amount of time
	Accurate	N/A
Domain specific constraints	Correctness	Avoidance of NFZ (static & dynamic)

TABLE V
DESCRIPTION OF DEPENDABLE ATTRIBUTES RELATED TO THE TPN SOLUTION FINDER.

5.3 Assurance Evidence Generation

5.3.1 Assurance of the Knowledge Base

SMT-based verification can be used to ensure the accuracy of the knowledge base. However, in the context of the TPN solution finder, it is used to verify that the geo-fence constraints are correctly defined.

TPN solution finder takes these geo-fence constraints as a set of polygons in the NED (North East Down) coordinate system. We used the same concept described in *PolyCARP*[1], where the verification process relies on two constraints, the first one ensures that the polygon must have more than two distinct vertices and that the vertices of the polygon are distinct from one another. The second constraint checks proximity conditions for certain edges w.r.t predefined threshold a. We express these constraints in SMT-LIB format as follows:

```
;; near_poly_edge function
(define-fun near_poly_edge ((N posnat) (p (Array Int Vect2))
       (s Vect2) (BUFF posreal) (i below))
  Bool
  (let ((next (ite (< i (- N 1)) (+ i 1) 0)))
    (near_edge (select p i) (select p next) s BUFF)
  )
)

;; acceptable_polygon_2D function
(define-fun acceptable_polygon_2D ((N posnat)
      (p (Array Int Vect2)) (BUFF posreal))
  Bool
  (and (> N 2)
       (forall ((i Int))
         (forall ((j Int))
           (let ((mj (mod (+ j 1) N))
                 (mi (mod (+ i 1) N))
                 (pj (select p j))
                 (pi (select p i))
                 (pmi (select p mi))
                 (pmj (select p mj)))
             (ite (= i j) true
             (= pi pj) false
               (and (= j mi) (or
                   (near_poly_edge N p pmj BUFF i)
                   (near_poly_edge N p pi BUFF j)))
                (and (= i mj) (or
                   (near_poly_edge N p pmi BUFF j)
                   (near_poly_edge N p pj BUFF i)))
             )
           )
         )
       )
  )
)
```

5.3.2 Assurance of Heuristic

We make use of theorem-proving techniques in order to provide the evidence for *completeness related to the avoidance of NFZ (static and dynamic)*. Specifically, we check for design aspects that would allow us to state the safety of the component implemented. Part of the evidence related to the safety of TPN solution finder has been shown [3]. In this paper, we focus on the generation of evidence for the sub-objectives "Heuristics are admissible" and "Heuristics terminates (provide a solution)" presented in Table III, where we made use of PVS as theorem prover. The property shall help us to state the admissibility of the heuristic used in the algorithm implementing the TPN planner. The algorithm makes use of the angular distance to compute the heuristics. The idea is to show that the A^*-like algorithm used in the TPN-planner is admissible if it uses a *monotonic heuristic*. Therefore, we should create a lemma stating the monotonicity of the heuristic function. We first need to specify the function related to the heuristic, which in this case is the function in charge of computing the Haversine distance (angular distance) between two nodes. The function is implemented in Python as follows:

```
def h_function(self, node: GraphNode):
        # admissible heuristic / cost estimation
        dist = geoid.haversine_dist(
            node.wp.lat,
            node.wp.lon,
            self.goal.wp.lat,
            self.goal.wp.lon)
        return dist
```

The specification of the function in PVS is given as follows:

```
h_function(N:Vect2, G:Vect2): real=
LET
n_lat = N`x,
n_lon = N`y,
g_lat = G`x,
g_lon = G`y
IN
angular_distance(n_lat, n_lon, g_lat, g_lon)
```

The function `angular_distance` is already implemented in a PVS, so we do not have to specify it again. The following lemma states the monotonicity of the heuristic. A monotonic heuristic will never decrease the estimated value. The lemma states that it cannot be the case that if we have a node N to expand in the path of the optimal cost to the goal node G, and a given goal suboptimal goal G2 is selected for expansion, then the estimation of the cost for G2 is bigger than the estimation of N.

```
Lemma_h: LEMMA FORALL(S:Vect2, N:Vect2, G:Vect2):
   NOT EXISTS (G2:Vect2):
      h_function((S,G) >= h_function (S, N)
      AND h_function (S,N) >= h_function (S,G2)
      AND h_function (S,G2)> h_function (S,G)
```

Since the heuristic is admissible, then it means it terminates (since an admissible heuristic always provides an optimal solution, then it provides a solution).

5.3.3 Assurance of the Domain Specific Constraints

SMT-based verification is used to verify domain-specific constraints. In the context of the TPN solution finder, these constraints are related to guaranteeing the avoidance of both static and dynamic NFZ. Specifically, we consider the following requirement related to the domain-specific constraints: *"Req: none of the edges of a mission plan (route) should intersect with any boundary of NFZ"*. The complete formal specification and verification of the static and dynamic NFZ for TPN solution finder can be found in [3]. The SMT-based verification effectively generated counter-examples where both static and dynamic NFZ constraints were violated.

6. Related Works

Research on safety assurance of AI-based systems in avionics has gained considerable attention in recent years. For instance, the second issue of the EASA AI concept paper [9] highlights the pressing challenges surrounding the trustworthiness of AI-based systems, particularly focusing on Level 2 AI, which involves human operators collaborating closely with AI systems. While this issue considers AI in general, it specifically focuses

[1]https://github.com/nasa/PolyCARP

on data-driven approaches, addressing their implications for safety assurance within avionics contexts. The assuring autonomy international program introduces AMLAS [13] as a methodology for the safety assurance of Machine Learning (ML) in autonomous systems. This methodology comprises a set of safety case patterns and processes for systematically integrating safety assurance into the development of ML components. Additionally, it is used to generate the evidence base by explicitly justifying the acceptable safety of these components when integrated into autonomous system applications. However, the proposed methodology does not consider symbolic AI, its scope is limited to ML-based systems.

In [19], [12], [20], authors focus on the verification and validation of expert systems, particularly emphasizing the challenges associated with rule-based expert systems. They highlight that the verification and validation processes for such systems tend to be informal, subjective, time-consuming, tedious, and arbitrary, in the creation and execution of test cases. While the authors propose an approach and set objectives for the verification and validation of rule-based expert systems, they do not provide guidance on how to align these objectives with existing standards, such as DO-178C. This gap underscores the need for further research to bridge the methodology proposed by the authors with established standards and practices in the avionics industry, ensuring that safety-critical systems meet rigorous regulatory requirements throughout their development and deployment phases.

7. Conclusions

In this paper, we introduced supplementary objectives intended to augment the DO-178C standards, specifically addressing features associated with symbolic AI. Additionally, we outlined a set of metrics designed to support the generation of evidence for the proposed objectives. The addition of supplementary objectives helps the traceability from design to code implementation and evidence generation. We illustrate this via a use case in which we generate evidence for one of the proposed objectives over an optimal route planner for aerial vehicles implemented using symbolic AI techniques.

References

[1] Uav applications. https://droneii.com/237-ways-drone-applications-revolutionize-business. Visited on 14/12/2022.

[2] VS Ajith and KG Jolly. Unmanned aerial systems in search and rescue applications with their path planning: a review. In *Journal of Physics: Conference Series*, volume 2115, page 012020. IOP Publishing, 2021.

[3] R Bouchekir, M Guzman, A Cook, J Haindl, and R Woolnough. Formal verification for safe ai-based flight planning for uavs. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 275–282. IEEE, 2023.

[4] Chih-Hong Cheng, Chung-Hao Huang, Harald Ruess, Hirotoshi Yasuoka, et al. Towards dependability metrics for neural networks. In *2018 16th ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, pages 1–4. IEEE, 2018.

[5] Darren Cofer and Steven P Miller. Formal methods case studies for do-333. Technical report, 2014.

[6] THEE Commission et al. Commission implementing regulation (eu) 2019/947 of 24 may 2019 on the rules and procedures for the operation of unmanned aircraft, 2019.

[7] Judy Crow, Sam Owre, John Rushby, Natarajan Shankar, and Mandayam Srivas. A tutorial introduction to pvs. Wift, 1995.

[8] Hang Dinh and Hieu Dinh. Inconsistency and accuracy of heuristics with a* search. *arXiv preprint arXiv:1307.2200*, 2013.

[9] European Union Aviation Safety Agency. Artificial intelligence concept paper (proposed issue 2). Concept Paper Proposed Issue 2, EASA, 2023.

[10] Mariusz Flasiński and Mariusz Flasiński. Symbolic artificial intelligence. *Introduction to Artificial Intelligence*, pages 15–22, 2016.

[11] EASA AI Task Force and AG Daedalean. Concepts of design assurance for neural networks (codann) ii. 2021.

[12] Lewey Gilstrap. Validation and verification of expert systems. *Telematics and Informatics*, 8(4):439–448, 1991.

[13] Richard Hawkins, Colin Paterson, Chiara Picardi, Yan Jia, Radu Calinescu, and Ibrahim Habli. Guidance on the assurance of machine learning in autonomous systems (amlas). *arXiv preprint arXiv:2102.01564*, 2021.

[14] Phil Kim, Brian C Williams, and Mark Abramson. Executing reactive, model-based programs through graph-based temporal planning. In *IJCAI*, pages 487–493. Citeseer, 2001.

[15] Microsoft. Microsoft z3. https://www.microsoft.com/en-us/research/project/z3-3. Visited on 01.09.2023.

[16] Radio Technical Commission for Aeronautics. Do-333 / ed-216 formal methods supplement to do-178c and do-278a. DO-333 DO-333, RTCA, Inc., 2011.

[17] Radio Technical Commission for Aeronautics. Do-178c / ed-12c software considerations in airborne systems and equipment certification. DO-178C DO-178C, RTCA, Inc., 2012.

[18] Natarajan Shankar, Devesh Bhatt, Michael Ernst, Minyoung Kim, Srivatsan Varadarajan, Suzanne Millstein, Jorge Navas, Jason Biatek, Huascar Sanchez, Anitha Murugesan, et al. Descert: design for certification. *arXiv preprint arXiv:2203.15178*, 2022.

[19] Suzanne Smith and Abraham Kandel. *Verification and validation of rule-based expert systems*. CRC Press, 2018.

[20] Jaak Tepandi. Verification, testing and validation of rule-based expert systems. *IFAC Proceedings Volumes*, 23(8):175–180, 1990.