A formal approach for identifying assurance deficits in unmanned aerial vehicle software

Adrian Groza[†], Ioan Alfred Letia[†], Anca Goron[†] and Sergiu Zaporojan[‡]

[†]Department of Computer Science
Technical University of Cluj-Napoca, Cluj-Napoca, Romania
EMAIL: {Adrian.Groza,Letia,Anca.Goron}@cs.utcluj.ro
[‡] Department of Computer Science
Technical University of Moldova, Chisinau, Moldova
EMAIL: zaporojan@mail.utm.md

Abstract. While formal methods have proved to be unfeasible for large scale systems, argument-based safety cases offer a plausible alternative basis for certification of critical software. Our proposed method for increasing safety combines formal methods with argumentation-based reasoning. In a first step, we provide a formal representation of the the argumentative-based Goal Structuring Notation (GSN) standard used in industry. In a second step, our solution exploits *reasoning in description logic* to identify assurance deficits in the GSN model. The identified flaws are given to a *hybrid logic-based model checker* to be validated against a Kripke model. The method is illustrated for an unmanned aerial vehicle software, with reasoning performed in RacerPro engine and the HLMC model checker based on hybrid logic.

Keywords: safety cases, argumentation, description logic, hybrid logic

1 Introduction

Assuring safety in complex technical systems is a crucial issue [6] in several critical applications like air traffic control or medical devices. Safety assurance and compliance to safety standards such as DO-178B [10] may prove to be a real challenge when we deal with adaptive systems, which we consider with continuous changes and without a strict behavioral model. Traditional methods, which are mainly based on previous experiences and lessons learned from other systems are not effective in this case. Argument-based safety cases offer a plausible alternative basis for certification in these fast-moving fields [10].

Goal Structuring Notation (GSN) is a graphical notation for structured arguments used in safety applications [7]. GSN diagrams depict how individual goals are supported by specific claims and how these claims or sub-goals are supported by evidence. A GSN diagram consists of the following nodes: achieved goals, not achieved goals, context, strategy, justification, assumption, validated evidence and not validated evidence. The nodes are connected by different supporting links like: has-inference or has-evidence. To support automatic reasoning on safety cases, we formalise the GSN standard in DL.

Our solution exploits *reasoning in description logic* to identify assurance deficits in the GSN model. The identified flaws are given to a *hybrid logic-based model checker* to be validated in a given Kripke structure. All formulas were verified using the Hybrid Logic Model Checker (HLMC) [5] extended to include Next, Future and Until operators, while the reasoning in Description Logic (DL) was performed on RacerPro [8].

2 System Architecture

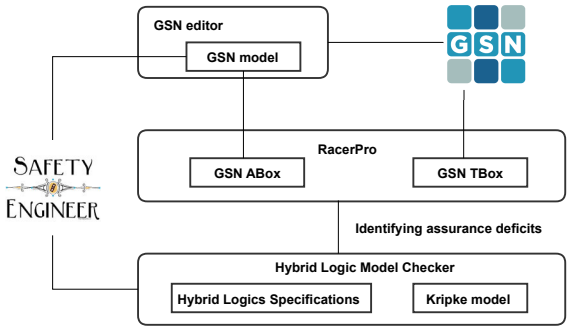

Fig. 1. System architecture

The solution is based on three technical instrumentations: (i) the $\mathcal{SHI}$ version of DL, (ii) the GSN standard, and (iii) hybrid logics (HLs). For the syntax, the semantics and explanation about families of description logics, the reader is referred to [2]. For the GSN graphical notation the minimum elements are introduced in section 3, while for a complete description the reader is referred to [7]. We assume also that the reader is familiar with model checking in temporal logic. However, in the following we provide specific details about HLs.

Hybrid logics extend temporal logics with special symbols that name individual states and access states by name [1]. With nominal symbols $\mathcal{N} = \{i_1, i_2, \ldots\}$ called *nominals* and $\mathcal{S}_{\text{var}} = \{x_1, x_2, \ldots\}$ called *state variables* the syntax of hybrid logics is $\varphi := TL \mid i \mid x \mid @x_t\varphi \mid\downarrow x.\varphi \mid \exists x.\varphi$. With $i \in \mathcal{N}$, $x \in \mathcal{W}_{\text{var}}$, $t \in \mathcal{N} \cup \mathcal{W}_{\text{sym}}$, the set of *state symbols* $\mathcal{W}_{\text{sym}} = \mathcal{N} \cup \mathcal{W}_{\text{var}}$, the set of *atomic letters* $\mathcal{A}_{\text{let}} = \mathcal{P} \cup \mathcal{N}$, and the set of *atoms* $\mathcal{A} = \mathcal{P} \cup \mathcal{N} \cup \mathcal{W}_{\text{var}}$, the operators $@, \downarrow, \exists$ are called *hybrid operators*. The semantics of hybrid logic is formalized by the following statements:

$$
\begin{array}{lll}
\mathcal{M}, g, m \models a & \text{iff} & m \in [V, g](a),\ a \in \mathcal{A} \\
\mathcal{M}, g, m \models @_t\varphi & \text{iff} & \mathcal{M}, g, m' \models \varphi, \text{ where } [V, g](t) = \{m'\}, t \in W_{sym} \\
\mathcal{M}, g, w \models\downarrow x.\varphi & \text{iff} & \mathcal{M}, g^x_m, w \models \varphi \\
\mathcal{M}, g, m \models \exists x.\varphi & \text{iff} & \text{there is } m' \in \mathcal{M} \text{ such that } \mathcal{M}, g^x_{m'}, w \models \varphi
\end{array}
$$

The semantics, where $\mathcal{M} = \langle M, R, V \rangle$ is a Kripke structure, $m \in M$, and g is an assignment, specifies the roles of the @ operator (shifts evaluation to the state named by nominal t), the downarrow binder $\downarrow$, respectively the existential binder $\exists$, binding the state variable x to the current state, respectively to some state in the model [5] . A hybrid Kripke structure $\mathcal{M}$ consists of an infinite sequence of states $m_1, m_2, ...$, R a family of binary accessibility relations on $\mathcal{M}$ and a valuation function L that maps ordinary propositions and nominals to the set of states in which they hold, i.e. $\mathcal{M} = \langle \langle m_1, m_2 ... \rangle, R, L \rangle$ [4]. In the graph oriented representation of $\mathcal{M}$, the nodes correspond to the sequence of states brought about by different modalities represented as links between states. Each state is labeled by a different nominal, while links are labeled by the relation connecting two states.

Running scenario. The illustrative scenario regards the safe insertion of a UAV into the civil air traffic as shown in [3]. The presented Unmanned Aircraft System consists of the UAV itself equipped with an autonomous control system, a ground station and the Air Traffic Management, which provides the required coordinates for the UAV. The goal is to prove that an UAV can complete safely its mission inside the civil air traffic and that all the major implied risks (e.g. collision with other objects or UAVs, loss of critical functions) are mitigated. For space considerations, we will restrict ourselves to those safety cases related to collision risks. In this specific context, an autonomous decision making system must consider at all times the set of safety regulations elaborated to deal with collision detection and avoidance imposed during a mission [11].

The corresponding hybrid Kripke structure is illustrated in Fig. 2. Its states correspond to the basic functions of the system (table 1): path following along the established corridor (*PathFollowing*), detection of possible obstacles (*DetectObstacles*), avoidance maneuver (*AvoidObstacles*), fault control (*FaultControl*) and landing (*Land*). The transition from one state to another is triggered by an event that leads to a change in the system's parameters: *obs* (signals presence of obstacles), *d* (returns distance between UAV and obstacle), *errObs* (signals an error in the *Detect* function) and *errAvoid* (signals an error in the *Avoid* function). For example, the signaling of an approaching obstacle when in the *DetectObstacles* state leads to the transition of the system in the *AvoidObstacles* state. The signaling of an error in any of the *DetectObstacles* or *AvoidObstacles* functions, leads the system in the *FaultControl* state. If the system recovers from the error state, it returns to the *PathFollowing* state. Otherwise, the mission is aborted by landing to a designated location, hence entering in the *Land* state.

3 A Formal Model for the GSN Standard

3.1 The GSN Safety Case for the UAV Scenario

In this section we build the safety case for our scenario as a GSN diagram. The top level goal states that all risks of collisions are managed (Fig. 3). This claim is refined into two more specific sub-goals, each capturing a different possible

Table 1. Set of states of the UAS

State	Parameters	Specification
PathFollowing	$\neg obs$	UAV follows the path on the given corridor
DetectObstacles	obs	Obstacles are signaled by sensors
AvoidObstacles	$obs \wedge d$	UAV performs an avoidance maneuver
FaultControl	$errObs \vee errAvoid$	Error signaled by *Detect* or *Avoid*
Land	$\neg obs \vee errObs \vee errAvoid$	UAV performs the landing procedure

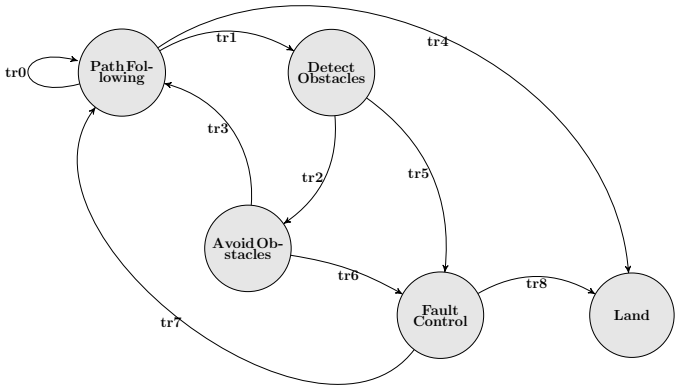

Fig. 2. Kripke model for the UAV.

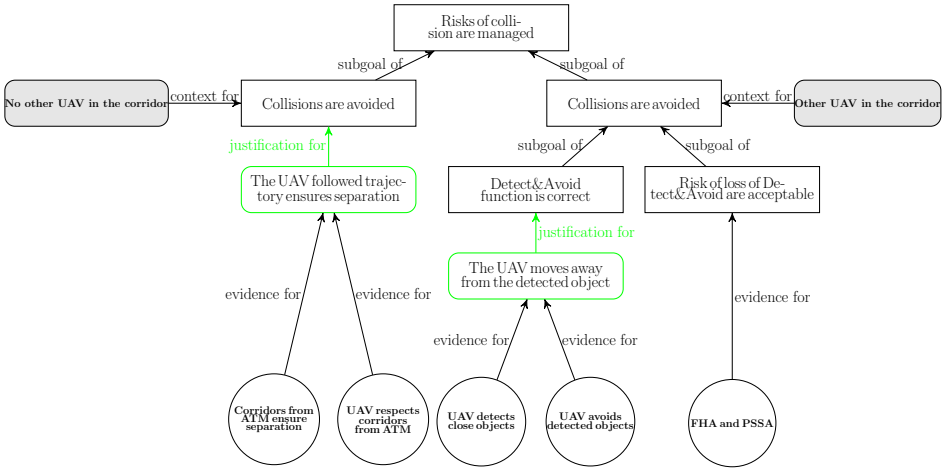

Fig. 3. Goal Structuring Notation

context: with another UAV present in the same corridor space or with another object. We further decompose the sub-claim referring to the exceptional case

of collision with another UAV, arguing that the avoidance must be ensured by a specific emergency procedure (based on the so-called Detect&Avoid function) and by mitigating all the risks in case of function loss. We continue the refinement process until we come to elementary claims that may be ensured by evidences. Considering the sub-claim referring to the situation in which no other UAV is present in the corridor space, we argue that the ATM transmits the correct coordinates for the right path following and that the UAV acknowledges correctly the commands received and sets its trajectory based on them. Moreover, we argue that the Detect&Avoid function is correct and therefore it must ensure that an obstacle is identified within a certain distance which allows the safe application of avoidance maneuver and that the UAV performs the indicated safety avoidance commands. Additionally, a safety claim is associated to the Detect&Avoid function stating that the risks of loss for this function are acceptable. The risks are calculated as specified in [3] using the Functional Hazard Assessment (FHA) for identifying a severity for each failure and flight mode(automatic or manual) of the UAV, and the Preliminary System Safety Assessment (PSSA) for deriving the safety requirements. These analyses are taken as evidences in validating as acceptable the risks of loss for the Detect&Avoid function.

3.2 Modeling the Goal Structuring Notation in DL

The relationship *supportedBy*, allows inferential or evidential relationships to be documented. The allowed connections for the *supportedBy* relationship are: goal-to-goal, goal-to-strategy, goal-to-solution, strategy to goal. Axiom A_1 specifies the range for the role *supportedBy*, axiom A_2 the range, axiom A_3 introduces the inverse role *supports*, and A_4 constraints the role *supportedBy* to be transitive.

(A_1) $\top \sqsubseteq \forall supportedBy.(Goal \sqcup Strategy \sqcup Solution)$
(A_2) $\exists supportedBy \top \sqsubseteq Goal \sqcup Strategy$
(A_3) $supportedBy^{-} \equiv supports$
(A_4) $supportedBy \sqsubseteq supportedBy$

Inferential relationships declare that there is an inference between goals in the argument. Evidential relationships specify the link between a goal and the evidence used to support it. Axioms A_5 and A_8 specify the range of the roles *hasInference*, respectively *hasEvidence*, while A_6 and A_9 the domain of the same roles. Definitions A_7 and A_{10} say that the *supportedBy* is the parent role of both *hasInference* and *hasEvidence*, thus inheriting its constraints.

(A_5) $\top \sqsubseteq \forall hasInference.Goal$
(A_6) $\exists hasInference \top \sqsubseteq Goal$
(A_7) $hasInference \sqsubseteq supportedBy$
(A_8) $\top \sqsubseteq \forall hasEvidence.Evidence$
(A_9) $\exists hasEvidence \top \sqsubseteq Goal$
(A_{10}) $hasEvidence \sqsubseteq supportedBy$

Goals and sub-goals are propositions that we wish to be true that can be quantified as quantified or qualitative, provable or uncertainty.

(A_{11}) $QuantitativeGoal \sqsubseteq Goal$
(A_{12}) $QualitativeGoal \sqsubseteq Goal$
(A_{13}) $ProvableGoal \sqsubseteq Goal$
(A_{14}) $UncertaintyGoal \sqsubseteq Goal$

A sub-goal supports other high level goals. Each safety case has a top level *Goal*, which does not support other goals.

(A_{15}) $SupportGoal \equiv Goal \sqcap \exists supports.\top$
(A_{16}) $TopLevelGoal \equiv Goal \sqcap (\neg SupportGoal)$

For each safety argument, the elements is instantiated and a textual description is attached to that individual by enacting the attribute *hasText*:

(A_{17}) $\top \sqsubseteq \forall hasText.String)$, (A_{18}) $\exists hasText.Statement \sqsubseteq \top$
(f_1) $gt : TopLevelGoal$, (f_2) $(gt, "The\ system\ meets\ requirements") : hasText$
(f_3) $gp : ProvableGoal$, (f_4) $(gp, "Quick\ release\ are\ used") : hasText$
(f_5) $gu : UncertaintyGoal$, (f_6) $(gu, "Item\ has\ a\ reliability\ of\ 95\%") : hasText$

Intermediate explanatory steps between goals and the evidence include statements, references, justifications and assumptions.

(A_{20}) $Explanation \sqsubseteq Statement \sqcup Reference \sqcup Justification \sqcup Assumption$
(A_{21}) $Statement \equiv \neg Reference$, (A_{22}) $Statement \equiv \neg Justification$
(A_{23}) $Statement \equiv \neg Assumption$, (A_{24}) $Reference \equiv \neg Justification$,
(A_{25}) $Reference \equiv \neg Assumption$, (A_{26}) $Justification \equiv \neg Assumption$,

The evidences or solutions form the foundation of the argument and will typically include specific analysis or test results that provide evidence of an attribute of the system. In our approach, the evidence consists in model checking the verification for a specification of the system.

(A_{27}) $Evidence \sqsubseteq \exists hasFormula.Formula \sqcap \exists hasSpecification.Statement$
$\exists hasModel.KripkeModel \sqcap \exists hasTestResult.\top$

Given the above formalization for GSN, our scenario depicted in Fig. 3 is formally represented in Fig. 4.

Table 2. Retrieving information about the GSN model.

Query	RacerPro query	RacerPro answer
Top level goal	$(concept - instances\ TopLevelGoal)$	g_1
Support goals	$(concept - instances\ SupportGoal)$	g_2, g_3, g_4, g_5
Evidence supporting goal g_1	$(individual - fillers\ g_1\ hasEvidence)$	e_1, e_2
Evidence verified against the model m_1	$(individual - fillers\ m_1)$ $(inverse\ hasModel))$	e_1, e_2, e_3, e_4, e_5
Evidence not verified	$(concept - instances(and\ Evidence$ $(some\ hasTestResult\ False))$	e_1, e_2, e_3, e_4, e_5
Goals supported by not verified evidence	$(concept - instances\ NotVerifiedGoals)$	g_1, g_2, g_3, g_4, g_5

(A_{28}) $NotVerifiedGoal \equiv Goal \sqcap \exists hasEvidence.NotVerifiedEvidence$
(A_{29}) $NotVerifiedEvidence \equiv Evidence \sqcap \exists hasTestResult.False$

$g_1 : Goal$, $g_2 : Goal$, $g_3 : Goal$, $g_4 : Goal$, $g_5 : Goal$
$e_1 : Evidence$, $e_2 : Evidence$, $e_3 : Evidence$, $e_4 : Evidence$, $e_5 : Evidence$
$(g_2, g_1) : supports$, $(g_3, g_1) : supports$, $(g_4, g_3) : supports$, $(g_5, g_3) : supports$
$(g_2, e_1) : hasEvidence$, $(g_2, e_2) : hasEvidence$, $(g_4, e_3) : hasEvidence$
$(g_4, e_4) : hasEvidence$, $(g_5, e_5) : hasEvidence$
$(g_1, "Risks\ of\ collision\ are\ managed.") : hasText$
$(g_2, "Collisions\ are\ avoided\ -\ No\ UAV.") : hasText$
$(g_3, "Collisions\ are\ avoided\ -\ UAV.") : hasText$
$(g_4, "Detect\&Avoid\ function\ is\ correct.") : hasText$
$(g_5, "Risk\ of\ loss\ of\ Detect\&Avoid\ is\ acceptable.") : hasText$
$(e_1, "Corridors\ from\ ATM\ ensure\ separation.") : hasSpecification$
$(e_2, "UAV\ respects\ corridors\ from\ ATM.") : hasSpecification$
$(e_3, "UAV\ detects\ close\ objects.") : hasSpecification$
$(e_4, "UAV\ avoids\ detected\ objects.") : hasSpecification$
$(e_5, "FHA\ and\ PSSA.") : hasSpecification$
$c_1 : Context$, $(c_1, "No\ other\ UAV\ in\ the\ corridor.") : hasText$
$c_2 : Context$, $(c_2, "Other\ UAV\ in\ the\ corridor.") : hasText$
$m_1 : KripkeModel$, $(e_1, m_1) : hasModel$, $(e_2, m_1) : hasModel$
$(e_3, m_1) : hasModel$, $(e_4, m_1) : hasModel$, $(e_5, m_1) : hasModel$

Fig. 4. The Abox of the UAV scenario.

4 Interleaving Reasoning with HL and DL for Identifying Assurance Deficits

Our method interleaves two steps: First, we check with hybrid logic if the evidence nodes from the GSN representation have their corresponding formulas validated against the Kripke model. Second, by reasoning in DL, we identify which goals in the GSN model are not supported by verified evidence.

4.1 Validating Evidence with Model Checking

For the given scenario, we start by verifying the first two pieces of evidence e_1 and e_2 in the model M_1. The verification uses three parameters: (i) the minimum distance d_{min} allowed between the UAV and another object without risk of collision; (ii) the reported coordinates c_{uav} by the UAV; and (iii) the given coordinates c_{ATM} by the ATM. Formula f_1 attached to evidence e_1 through the assertion $(related\ e_1\ f_1\ hasFormula)$ in DL is expressed in HL as:

$$f_1 =\downarrow i(c_{ATM}) \rightarrow @_i[F](c_{ATM} > d_{min}) \tag{1}$$

f_1 states that if the ATM starts transmitting coordinates at a state i, then for all future states the coordinates will be transmitted such that to ensure that the minimum safe distance is preserved between the UAV and other objects.

The formula corresponding to the evidence e_2 is:

$$f_2 =\downarrow i(c_{ATM}) \rightarrow @_i[Next](c_{uav} = c_{ATM}) \tag{2}$$

According to f_2, if the ATM starts transmitting coordinates at a state i, then in the next state the UAV should acknowledge the newly received coordinates by reporting the exact coordinates as the ones transmitted in the previous state. The justification j_2 of the sub-goal g_2 supported by e_1 and e_2 is expressed as:

$$j_2 =\downarrow i(c_{uav} = c_{ATM}) \rightarrow @_i[F](c_{uav} > d_{min}) \tag{3}$$

The implication $f_1 \wedge f_2 \rightarrow j_2$ is true (the acknowledgment and following of the coordinates from the ATM ensures the required minimum safe distance) .

Evidences e_3 and e_4 are used to validate the sub-goal g_4 about the correctness of the Detect&Avoid function. To check the supporting evidences, two parameters are required: (i) the reported distance d_{obs} between the UAV and another approaching UAV; and (ii) the minimum distance d_{min} allowed without any risk of collision. The justification j_4 for the sub-goal g_4 is formalized as:

$$j_4 =\downarrow i(d_{obs} < d_{min}) \rightarrow @_i[F]((d_{obs} \neq 0)U(d_{obs} > d_{min})) \tag{4}$$

Justification j_4 states that if we bind to i the state in which the reported distance between the UAV and another approaching UAV is less than the minimum one then for all future states the reported distance must be kept higher then 0, increasing it, at the same time, until no danger of collision ($d_{obj} > d_{min}$).

Evidence e_3 (*UAV detects close objects*) is formally expressed as:

$$f_3 =\downarrow i(d_{obs}) \wedge @_i(d_{obs} < d_{min}) \rightarrow\downarrow i(obs) \tag{5}$$

According to f_3, if in the current state named by nominal i, the distance to a possible obstacle is transmitted to the UAV and the distance is less than the minimum allowed one, the presence of an obstacle is reported by the sensors to the UAV signaling a risk for collision.

Evidence e_4 (*UAV avoids detected objects*) is formally expressed as:

$$f_4 =\downarrow i(obs) \rightarrow @_i((d_{obs} \neq 0)U(d_{obs} > d_{min})) \tag{6}$$

Equation 6 states that if we bind to nominal i the state in which an obstacle is signaled by the sensors to the UAV, then the reported distance to the obstacle must be maintained different than 0 until the increase of distance between the UAV and the obstacle becomes higher then the minimum established threshold, indicating that the avoidance maneuver was performed.

To complete the validation of g_4, we have to prove the formula $f_3 \wedge f_4 \rightarrow j_4$, which is true (the presence of an obstacle indicated by an observed distance, which is less than the minimum accepted one will entail an avoidance maneuver).

4.2 Identifying Assurance Deficits

At this time check, the formal GSN model is updated with the assertions in Fig. 5. Given the new information, the GSN model can be interogated to retrieve goals and evidence which are not validated yet. Querying the RacerPro engine for the

*NotVerifiedGoal*s, we obtain g_1, g_3, g_5, while the concept *NotVerifiedEvidence* includes only one instance, the evidence e_5. The RacerPro system is able to provide explanations why a specific goal belongs to a specific concept. In this way, the safety engineer can figure that the goal g_3 is not validated because of g_5, which relies on the piece of evidence e_5 whose formula was not checked in the given kripke model M_1. This reasoning mechanism is particularly useful in real application where a GSN model has hundreds of nodes.

$(e_1, f_1) : hasFormula$, $(e_1, "true") : hasTestResult$
$(e_2, f_2) : hasFormula$, $(e_2, "true") : hasTestResult$
$(g_2, j_2) : hasJustification$, $(f - g_2, "true") : hasTestResult$
$(e_3, f_3) : hasFormula$, $(e_3, "true") : hasTestResult$
$(e_4, f_4) : hasFormula$, $(e_4, "true") : hasTestResult$
$(g_4, j_4) : hasJustification$, $(f - g_4, "true") : hasTestResult$

Fig. 5. Updating the Abox for the GSN model with the newly validated evidences.

Given the above knowledge, the safety engineer is aware that the *SupportGoal* g_5 should be validated. In the given scenario, the validation is based on the analysis results of the FHA and PSSA considering the reported error parameters *errObs* and *errA*. The maximum acceptable degree of risk will be referred as r_a. If in the FaultControl state the parameters *errObs* and *errA* will lead to a risk result r_{err} which is higher than the maximum degree of allowed risks, then the emergency landing is performed. Formally:

$$f_7 =\downarrow i(FaultControl) \wedge @_i(r_{err} < r_a) \rightarrow ([Next]i \rightarrow Land) \tag{7}$$

One can observe from the Kripke structure in Fig. 2 that there is a valid transition from state $FaultControl$ to state $Land$ in case that the risk is higher than the acceptable limit, but also to $PathFollowing$ in case that the returned result is a positive one and it allows the UAV to continue its mission safely. Therefore, formula f_7 proves as true. With this new information sent to the RacerPro engine, the concept *NotVerifiedGoals* will contain no instances, which formally validates the safety case from the GSN model.

5 Discussion and Related Work

While both argumentation [7, 6, 10] and model checking [11] have been applied for certification of safety systems, we aimed to demonstrate that combining the two methods might bring about additional advantages such as preliminary validation of argumentation schemes constructed to support safety cases, ensuring in advance that the stability of the system will not be affected by the available choices and, at the same time, foreseeing possible impediments in selecting one option over another. Considering the benefits of abstractization by combining DL with model checking [9], we complemented the graphical GSN standard with a formalized model. We argue that this joint approach increase the transparency and trust when certifying critical safety systems.

6 Conclusion

The contributions of the paper are: 1) integrating hybrid logic with argumentation theory, and 2) providing a formal model of the GSN standard in description logic. While the GSN graphical argumentation language structures safety cases and facilitates understanding for the human agent, the hybrid logic is able to validate the evidence nodes of the diagram. Description logic was used as a middleware language to lightly integrate GSN and model checking. DL's reasoning capabilities are used to analyze the status of the arguments and their supporting evidence. In our view, the proposed method is a step towards a formal model for the GSN standard. Currently, we are investigating the feasibility of our solution against large-scale safety cases.

Acknowledgments

This work was supported by the Romania-Moldova Bilateral Agreement entitled "ASDEC: Structural Argumentation for Decision Support with Normative Constraints", from the National Research Council of the Romanian Ministry of Education and Research and Moldova Ministry of Education.

References

1. Areces, C., ten Cate, B.: Hybrid logics. In: Blackburn, P., Van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, pp. 821–868. Elsevier Amsterdam (2007)
2. Baader, F.: The description logic handbook: theory, implementation, and applications. Cambridge university press (2003)
3. Brunel, J., Cazin, J.: Formal methods for the certification of autonomous unmanned aircraft systems. In: Formal Verification of a Safety Argumentation and Application to a Complex UAV System. pp. 307–318. SAFECOMP'11, Springer-Verlag, Berlin, Heidelberg (2012)
4. Cranefield, S., Winikoff, M.: Verifying social expectations by model checking truncated paths. Journal of Logic and Computation 21(6), 1217–1256 (2011)
5. Franceschet, M., de Rijke, M.: Model checking hybrid logics (with an application to semistructured data). Journal of Applied Logic 4, 279–304 (2006)
6. Graydon, P., Habli, I., Hawkins, R., Kelly, T., Knight, J.: Arguing conformance. Software, IEEE 29(3), 50–57 (2012)
7. Graydon, P., Kelly, T.P.: Using argumentation to evaluate software assurance standards. Information and Software Technology 55(9), 1551–1562 (2013)
8. Haarslev, V., Hidde, K., Möller, R., Wessel, M.: The racerpro knowledge representation and reasoning system. Semantic Web 3(3), 267–277 (2012)
9. Letia, I.A., Groza, A.: Compliance checking of integrated business processes. Data Knowl. Eng. 87, 1–18 (2013)
10. Rushby, J.: A safety-case approach for certifying adaptive systems. In: AIAA Infotech@Aerospace Conference, American Inst. of Aeronautics and Astronautics (2009)
11. Webster, M., Fisher, M., Cameron, N., Jump, M.: Formal methods for the certification of autonomous unmanned aircraft systems. In: Proceedings of the 30th International Conference on Computer Safety, Reliability, and Security. pp. 228–242. SAFECOMP'11, Springer-Verlag, Berlin, Heidelberg (2011)