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

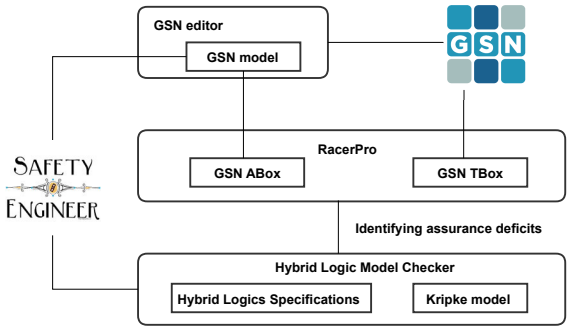

Fig. 1. System architecture

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:

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