January 2014 -4 Decembrie 2014
Bilateral Agreement Romania-Argentina
Contract No. 731

Project tile:	ARGSAFE: Using Argumentation for Justifying Safeness of Complex Technical Systems
Romanian partner:	Technical University of Cluj-Napoca
Argentinian partner:	Universidad Nacional del Sur
Duration:	24 months: 11 Septembrie 2013 -11 Septembrie 2015)

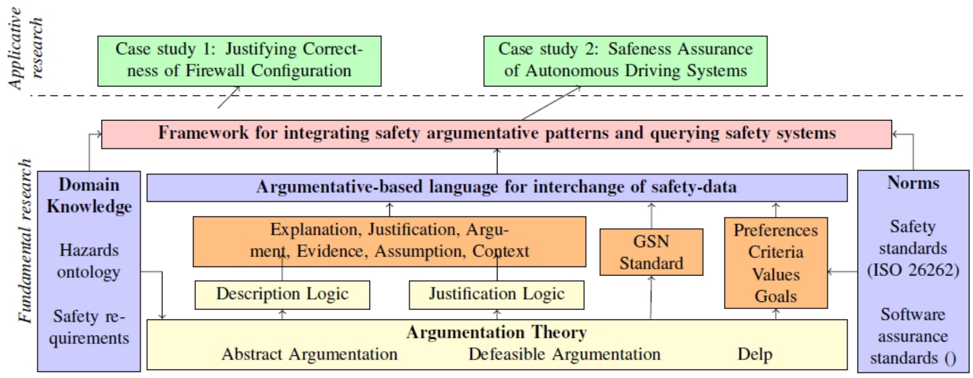
Contents

0.1 Objectives 2
0.2 Modeling the GSN Standard in Description Logic 4
0.3 SafeEd Tool 6
0.4 Formal Verification of Safety Cases 8
  0.4.1 Vehicle Overtaking Scenario 8
  0.4.2 Validating the Safety Case 9
  0.4.3 Generation of Safety Case Metrics 12
  0.4.4 Generating Natural Language Reports on the Safety Case 13
0.5 Interleaving Argumentation and Model Checking 13
0.6 Model Repair for an Unmanned Aircraft Vehicle 15
  0.6.1 Illustrative Example 15
  0.6.2 Kripke Model for the Unmanned Aerial Vehicle 16
  0.6.3 Verifying Compliance to Safety Regulations 17
  0.6.4 Adapting the Model to New Specifications 18

0.1 Objectives

Objectives. The top level scientific objective regards safety assurance of software systems by means of argumentation theory.

Date	Objectives	Novelty	Associated Tasks
Jun 2013	O1. Analysis the problem of justifying safeness of complex technical systems.	Identifying the possibilities of integrating argumentation theory, quality standards and ontologies.	Formal analysis of quality standards. Identifying factors affecting confidence in safeness of software systems.
Sep 2013	O2. Developing the assurance model based on argumentation theory.	Justificative reasoning in the context of heterogeneous and contradictory evidence	Developing the defeasible justification logic. Contextualising evidence.
May 2014	O3. Developing the system of justifying safeness of complex technical system.	Automatic identification of inconsistent justifications.	Developing a generic ontology for hazards. Organising the first ARGSAFE workshop.
Sep 2014	O4. Applying the system for safeness assurance of autonomous driving systems.	Organising evidence, building arguments and counter-arguments.	Formalising safeness requirements. Formalising assumptions regarding operating mode and specific hazards.
Dec 2014	O5 Applying the system for verifying correctness of firewall configuration	Presenting arguments for decision support under temporal constraints	Identifying inconsistency in security rule-based systems. Organising the second ARGSAFE workshop.
Mar 2015	O6. Developing a methodology of exploiting structured arguments in safeness assurance	Re-using safety cases. Re-engineering complex software systems based on arguments.	Defining patterns of safety cases. Stating the principle of building arguments when developing complex software systems.



Team:

- Technical University of Cluj-Napoca: Assoc. Prof. dr. .eng.Adrian Groza, Prof. dr. eng. Ioan Alfred Letia, Phd stdeunt Anca Goron.
- Universidad National del Sur: Assoc. Prof. Sergio Alejandro Gomez, Prof. Carlos Ivan Chesnevar

Publications: List of publications [8, 6, 9, 11, 7]

1. A. Groza, N. Marc - Consistency Checking of Safety Arguments in the Goal Structuring Notation Standard, IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP2014), Cluj-Napoca, Romania, 4-6 September 2014, pp, 59-66
2. S. A. Gomez, A. Goron, A. Groza - Assuring Safety in an Air Traffic Control System with Defeasible Logic Programming, Argentine Symposium on Artificial Intelligence (ASAI14), 1-5 September 2014, Buenos Aires, Argentina
3. S.A. Gomez, A. Groza, C.I. Chesnevar - An Argumentative Approach to Assessing Safety in Medical Device Software using Defeasible Logic Programming, International Conference on Advancements of Medicine and Health Care through Technology (MEDITECH2014), Ed. S. Vlad, R. Ciupa, ISBN 978-3-319-07652-2, IFMBE, Vol 44, Springer, pp. 167-172
4. S. Gomez, A. Groza, C Chesnevar, I. A. Letia, A. Goron, M Lucero - ARGSAFE: Usando Argumentacion para Garantizar Seguridad en Sistemas Tecnicos Complejos, WICC, Ushuaia, Tierra del Fuego, Argentina, 7-8 May 2014
5. A. Goron, A. Groza, S. A. Gomez, I. A. Letia - Towards an argumentative approach for repair of hybrid logics models, ARGMAS@AAMAS, Paris, France, 5-9 May 2014

Deliverables:

(D1.1) Web page: http://cs-gw.utcluj.ro/∼adrian/projects/argsafe

(D1.2) Presentation poster (available on the project web page);

(D1.3) Workshop: "Agreement Technologies in Software Engineering":
http://cs-gw.utcluj.ro/∼adrian/workshops/ATSE2013.html

(D1.4) Ontology for the Goal Structuring Notation standard (available at project web page);

(D1.6) First year technical report (available at project web page);

(D2.1) EdSafe tool (available on the project web page);

(D2.2) Second year technical report (available on the project web page).

Novelty. We propose an *argumentation approach for hybrid logics model update.* Argumentation theory is used to assist the process of updating the model. We view a Hybrid Kripke model as a description of the world that we are interested in. The update on this Kripke model occurs when the system has to accommodate some newly desired properties or norm constraints. When the model fails to verify a property, a defeasible logic program is used to analyze the current state. Depending on the status of the arguments, the system can warrant four primitive operations on the model: updating state variables, adding a new transition, removing a transition, or adding a new state. A running scenario is presented showing the verification of an unmanned aerial vehicle, by interleaving reasoning in Defeasible Logic Programming and the Hybrid Logic Model Checker.

Assuring safety in complex technical systems is a crucial issue in several critical applications like air traffic control or medical devices. We developed a *framework based on argumentation for assisting flight controllers to reach a decision related to safety constraints* in an ever changing environment in which sensor data is gathered at real time.

Modern health-care technology depends to a large extent on software deployed in medical devices, which brings several well-known benefits but also poses new hazards to patient safety. As a consequence, assessing safety and reliability in software in medical devices turns out to be a critical issue. We developed a *method for safety assessment of medical devices based on Defeasible Logic Programming (DeLP)*, which provides an argumentative framework for reasoning with uncertain and incomplete knowledge. We contend that argumentation theory as defined in DeLP can be used to integrate and contrast different evidences for assessing the approval and commercialization of medical devices, aiming at increasing transparency to all the stakeholders involved in their certification. The outlined framework is validated by modeling the infamous Therac-25 accident.

Economic impact. Increasingly, safety regulatory bodies require the developers of critical software systems to provide explicit safety cases - defined in terms of structured arguments based on objective evidence - in order to prove that the system is acceptable safe. Argumentative-based safety cases are progressively adopted in the defense (UK), automotive, railways, off-shore oil & gas, or medical device domains. Consequently, this research aims i) to identify links between argumentation theory and engineering of safety systems, ii) to develop argumentation methods to transfer confidence in safety-critical software systems. iii) to apply the developed technical instrumentation at two case studies: 1) safeness of autonomous driving software, respectively 2) justifying correctness of firewall configuration. System capabilities include 1) automatic norm checking for compliance, 2) safety reports generation, 3) facilitating understanding and confidence transfer.

0.2 Modeling the GSN Standard in Description Logic

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*:

$$(A_1) \quad \top \quad \sqsubseteq \quad \forall\, supportedBy.(Goal \sqcup Strategy \sqcup Solution)$$

Axiom A_2 specifies the domain of the role *supportedBy*, axiom A_3 introduces the inverse role *supports*, and A_4 constraints the role *supportedBy* to be transitive.

$$\begin{array}{lllll} (A_2) & \exists\, supportedBy.\top & \sqsubseteq & Goal \sqcup Strategy \\ (A_3) & supportedBy^{-} & \equiv & supports \\ (A_4) & supportedBy & \sqsubseteq & supportedBy \end{array}$$

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.

$$
\begin{array}{llcl}
(A_5) & \top & \sqsubseteq & \forall\, hasInference.Goal \\
(A_8) & \top & \sqsubseteq & \forall\, hasEvidence.Evidence \\
(A_6) & \exists\, hasInference.\top & \sqsubseteq & Goal \\
(A_9) & \exists\, hasEvidence.\top & \sqsubseteq & Goal \\
(A_7) & hasInference & \sqsubseteq & supportedBy \\
(A_{10}) & hasEvidence & \sqsubseteq & supportedBy
\end{array}
$$

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

$$
\begin{array}{llcl}
(A_{11}) & QuantitativeGoal & \sqsubseteq & Goal \\
(A_{13}) & ProvableGoal & \sqsubseteq & Goal \\
(A_{12}) & QualitativeGoal & \sqsubseteq & Goal \\
(A_{14}) & UncertaintyGoal & \sqsubseteq & Goal
\end{array}
$$

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

$$
\begin{array}{llcl}
(A_{15}) & SupportGoal & \equiv & Goal \sqcap \exists\, supports.\top \\
(A_{16}) & TopLevelGoal & \equiv & Goal \sqcap \neg SupportGoal
\end{array}
$$

For each safety argument, the elements are instantiated and a textual description is attached to that individual by enacting the attribute *hasText* with domain *Statement* and range *String*:

$$
\begin{array}{llcl}
(A_{17}) & \top & \sqsubseteq & \forall\, hasText.String \\
(A_{18}) & \exists\, hasText.Statement & \sqsubseteq & \top
\end{array}
$$

Three individuals *gt*, *gp*, and *gu* of type goal and their textual descriptions are instantiated by assertions f_1 to f_6:

$$
\begin{array}{ll}
(f_1) & gt : TopLevelGoal \\
(f_2) & (gt, \text{“}\textit{The system meets its requirements}\text{”}) : hasText \\
(f_3) & gp : ProvableGoal \\
(f_4) & (gp, \text{“}\textit{Quick release are used}\text{”}) : hasText \\
(f_5) & gu : UncertaintyGoal \\
(f_6) & (gu, \text{“}\textit{The item has a reliability of 95\%}\text{”}) : hasText
\end{array}
$$

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

$$
\begin{array}{llcl}
(A_{20}) & Explanation & \sqsubseteq & Statement \sqcup Reference \sqcup \\
 & & & Justification \sqcup Assumption
\end{array}
$$

where these top level concepts are disjoint:

$$
\begin{array}{llcl}
(A_{21}) & \mathit{Statement} & \equiv & \neg \mathit{Reference} \\
(A_{22}) & \mathit{Statement} & \equiv & \neg \mathit{Justification} \\
(A_{23}) & \mathit{Statement} & \equiv & \neg \mathit{Assumption} \\
(A_{24}) & \mathit{Reference} & \equiv & \neg \mathit{Justification} \\
(A_{25}) & \mathit{Reference} & \equiv & \neg \mathit{Assumption} \\
(A_{26}) & \mathit{Justification} & \equiv & \neg \mathit{Assumption}
\end{array}
$$

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.

$$
\begin{array}{llcl}
(A_{27}) & \mathit{Evidence} & \sqsubseteq & \exists\, \mathit{hasFormula}.\mathit{Formula} \sqcap \\
 & & & \exists\, \mathit{hasSpecification}.\mathit{Statement} \sqcap \\
 & & & \exists\, \mathit{hasModel}.\mathit{KripkeModel} \sqcap \\
 & & & \exists\, \mathit{hasTestResult}.\top
\end{array}
$$

A non-verified goal is a goal which has at least one piece of evidence that is not formally proved.

$$
\begin{array}{llcl}
(A_{28}) & \mathit{NotVerifiedGoal} & \equiv & \mathit{Goal} \sqcap \exists\, \mathit{hasEvidence}. \\
 & & & \mathit{NotVerifiedEvidence}
\end{array}
$$

$$
\begin{array}{llcl}
(A_{29}) & \mathit{NotVerifiedEvidence} & \equiv & \mathit{Evidence} \sqcap \\
 & & & \exists\, \mathit{hasTestResult}. \\
 & & & (\mathit{False} \sqcap \mathit{Unknown})
\end{array}
$$

0.3 SafeEd Tool

Our tool consists of a set of Eclipse plugins. The tool is structured on layers (Fig. 1). At the bottom, there is the layer consisting of the core framework of the tool.

The second layer consists of several eclipse plug-ins used to implement the tool. The Eclipse Modelling Framework (EMF) was used for developing the model part. The Graphical Modelling Framework (GMF) and the Graphical Editing Framework (GEF) were used to implement the graphical user interface of the tool. Epsilon was used to construct the plug-ins for model management tasks.

The third layer contains the GSN and ARM metamodels, plus tool plug-ins through which all tool functionality is provided. This layer consists of: GSN plug-ins, which implements the GSN editor functionality, ARM plug-ins, which implements the ARM editor functionality, ONTOLOGY plug-ins, provides semantic reasoning facility.

The user interface layer consists of the GSN editor, the ARM editor and the model management tools: i) GSN to ARM transformation, ii) GSN validation using ontology-based reasoning, iii) safety case transformation in ABOX, iv) querying the safety case, v) various GSN editing wizards.

The main aim of the tool is to enhance end-user capabilities to build reliable assurance cases. The system supports management and assessment of the safety case, the Ontology plugin being responsible for this.

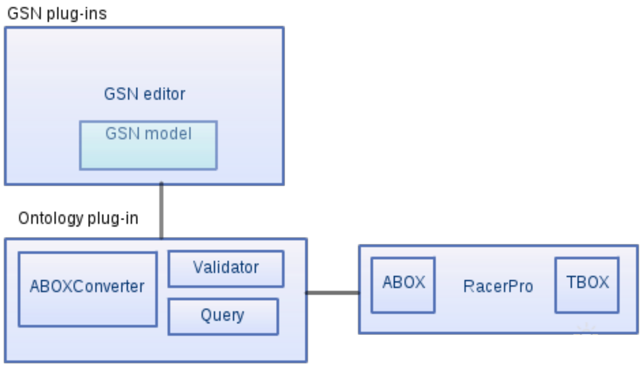


Figure 1: System Architecture

Based on the description logic, we developed an ontology that formalises the Goal Structuring Notation. The resulted GSN ontology is loaded from the Ontology plugin onto RacerPro engine using the jRacer library. The Ontology plugin provides also the engine used for translating the safety case diagram into an Abox. Furthermore, a connection to the RacerPro [10] reasoning engine is established in order to load the Abox in the GSN tbox so that the users could validate the abox against GSN tbox and query the safety case from the console.

An advantage is the possibility for the user to load many diagrams into the ontology and set the current safety case to be analyzed and query it from the console. Having the abox and tbox loaded in RacerPro, the user can select from the editor to create the OWL ontology of the safety case used to generate a documents containing description of the safety case in natural language or other reports.

The workspace of the system is presented in Fig. 2. A safety project (top-left) consists of several assurance cases, developed either as a graphical diagram (files with gsn extension) or as an abox in description logic (files with racer extension). In case of need the system automatically translated between these two input formats. For a selected diagram file the user can transform into abox, validate the diagram and generate reports.

The main window (top-cencer) depicts the active gsn diagram. The elements of the GSN standard are represented as follows: goals with rectangular, strategies with paralelograms, evidence and solutions are represented by circle, assumptions and justifications with ellipse, context by a rectangular with rounded corners, the supportedBy relation is an arrow with the head filled, while the inContextOf is represented by an arrow with empty head.

The title and description of a node can be entered by clicking on the node in the head part for the title, and in the field with the placeholder ‘description‘. The diagram is constructed by using a drag-and-drop pallet (top-right).

The command console (bottom-center) shows the reasoning performed on the active diagram above. In the command line, specific queries for interrogating ontologies can be added and the reasoning engine will return the results for each query. The syntax of the queries corresponds to the RacerPro tool. In Fig. 2, the four queries exemplified are: i) retrieving all the goals in the diagram, ii) identifying the top level goal, iii) listing all pieces of evidence supporting the goal g_2, and iv) checking the consistency of a diagram with respect to the GSN standard encoded as axioms in description logic.

In the bottom-left corner the red rectangle represent the view part of the diagram visible in the main window.

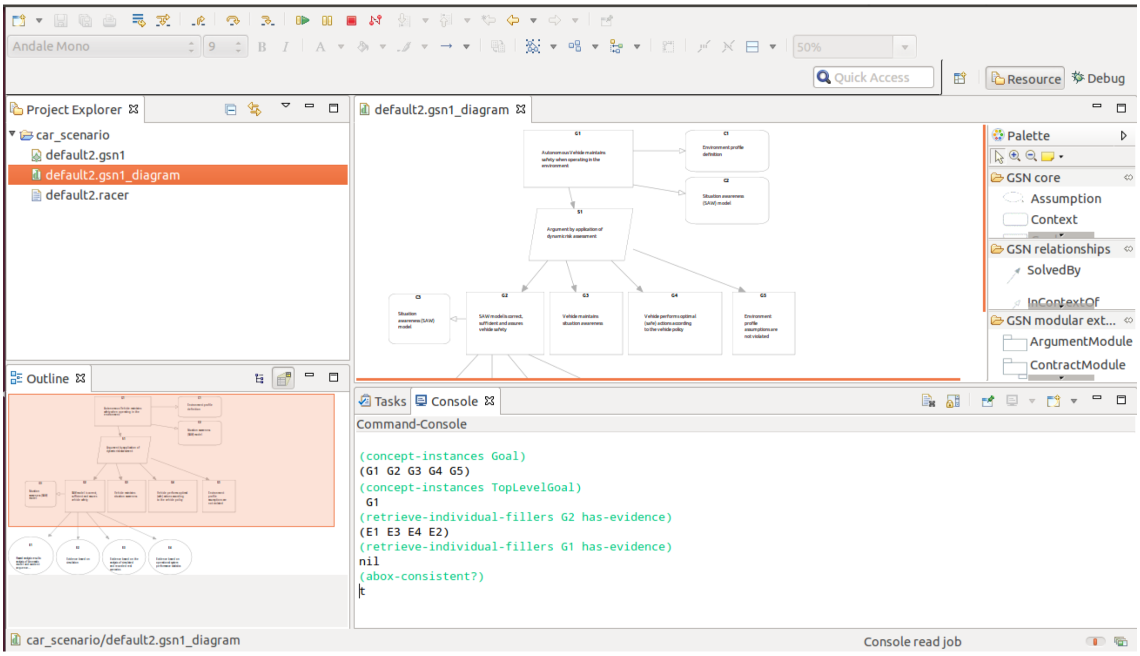


Figure 2: Application Interface.

0.4 Formal Verification of Safety Cases

0.4.1 Vehicle Overtaking Scenario

A GSN diagram built in our *SafeEd* tool is represented in Fig. 3. The considered safety scenario is taken from the autonomous driving domain. The top level goal g_1 states that any autonomous vehicle should ensure safety when operating in the environment. The goal holds in two contexts: the existence of an environment formalisation (context c_1), respectively the existence of a mechanism providing situation awareness. One solution for ensuring safety is dynamic risk assessment approach [12]. The corresponding strategy s_1 used to support the goal g_1 is to dynamically assess the risk. The sub-goals g_2, g_3, g_4, and g_5 are used to fulfill the strategy s_1. For instance, the sub-goal g_2 claims the correctness of the model, statement that is supported by various pieces of evidence, including formal verification e_2

The diagram in Fig. 3 is translated into the Abox represented in Fig. 4. Here, the facts f_{51} to f_{54} assert the individuals to their corresponding GSN core elements. The structure of the GSN diagram based on the two relationships *supportedBy* and *inContextOf*

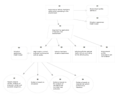


Figure 3: Autonomous vehicle scenario.

is formalised by the facts f_{55} to f_{62}. The natural language text describing claims, solutions, contexts or evidences are encapsulated as concrete attributes [10] in Racer syntax (assertions f_{63} to f_{70}).

The ISO 26262 standard states that any electrical/electronic product must ensure an acceptable level of safety and requires building a safety case, but it does not tell you the steps of building it [2]. Fig. 5 shows how such an analysis is performed in order to comply to the ISO26262 requirements, according to [4]. The figure presents only the "hazard analysis and risk assessment" component. The top level goal *Goal1* is to show if the product ensures a sufficient and acceptable level of safety.

The user should structure the safety case into product assurance cases and process related assurance cases.

In figures 5, 6, 7 only the hazard analysis and risk assessment claim of the product is developed and shown the corresponding process-based (Goal 2) and product-based(Goal 6) arguments.

The process-based goal *Goal2* in refined in Fig. 6. The goal claims that the process adopted to develop the product is correct and successfully completed. *Goal2* is divided, taking in account the roles (*Strategy1*) and activity steps(*Strategy2*), in 3 sub-goals: *Goal3*,*Goal4* and *Goal5*. *Goal4* claims that the hazards regarding the adapted process of building the product have been identified and classified, using the Hazard identification and analysis using HAZOP technique(HAZard and Operability analysis) to provide the evidence, representing *Evidence2* node, while *Goal5* claims that all the hazard have been carefully analyzed backward and forward, providing as solution hazard identification and analysis using HAZOP technique(HAZard and Operability analysis) represented as *Evidence3* and Failure Modes and Effects Analysis (FMEA) procedure and Fault Tree Analysis technique (FTA) as *Evidence4*.

The product-based goal *Goal6* is justified in Fig 7. This claims that the system has the required safe behavior, if something fails then the system should be able to fail in a safe way.The goal is divided in two goals: *Goal7* and *Goal8*. *Goal7* claims that all the hazards regarding the product have been found, while *Goal8* states that the the effects and causes of hazardous events have been analyzed. The goals have as solution techniques the same nodes *Evidence2*, *Evidence3*, *Evidence4*.

0.4.2 Validating the Safety Case

The RacerPro [10] reasoning engine is used by a tool to query and validate the Abox against the GSN tbox. When analysing the diagram by querying the RacerPro engine the safety engineer can simply identify the goals from the diagram that are still undeveloped or not supported by evidence, goal descriptions or retrieve explanation why a goal belongs to a

(f_{51}) g_1 : *Goal*, g_2 : *Goal*, g_3 : *Goal*, g_4 : *Goal*, g_5 : *Goal*
(f_{52}) c_1 : *Context*, c_2 : *Context*, c_3 : *Context*
(f_{53}) e_1 : *Evidence*, e_2 : *Evidence*,
e_3 : *Evidence*, e_4 : *Evidence*
(f_{54}) s_1 : *Context*, c_2 : *Context*, c_3 : *Context*
(f_{55}) (g_1, s_1) : *supportedBy*
(f_{56}) (g_1, c_1) : *inContextOf*
(f_{57}) (g_1, c_2) : *inContextOf*
(f_{58}) (g_2, c_3) : *inContextOf*
(f_{59}) (g_2, e_1) : *hasEvidence*
(f_{60}) (g_2, e_2) : *hasEvidence*
(f_{61}) (g_2, e_3) : *hasEvidence*
(f_{62}) (g_2, e_4) : *hasEvidence*
(f_{63}) (g_1, "*Autonomous Vehicle maintains safety when operating in the environment*") : *hasText*
(f_{64}) (g_2, "*SAW model is correct, sufficient and assures vehicle safety*") : *hasText*
(f_{65}) (g_3, "*Vehicle maintains situation awareness*") : *hasText*
(f_{66}) (g_4, "*Vehicle performs optimal (safe) actions* "*according to the vehicle policy*") : *hasText*
(f_{67}) (g_5, "*Environment profile assumptions are not violated*") : *hasText*
(f_{68}) (s_1, "*Argument by application of dynamic risk assessment*") : *hasText*
(f_{69}) (e_1, "*Hazard analysis results : analysis of kinematic model and accident sequence*") : *hasText*
(f_{70}) (e_2, "*Evidence based on simulation*") : *hasText*
(f_{70}) (e_3, "*Evidence based on the analysis of simulated and recorded real scenarios*") : *hasText*
(f_{70}) (e_4, "*Evidence based on operational system performance statistics*") : *hasText*
(f_{70}) (c_1, "*Environment profile definition*") : *hasText*
(f_{70}) (c_2, "*Situation awareness (SAW) model*") : *hasText*
(f_{70}) (c_3, "*Situation awareness (SAW) model*") : *hasText*

Figure 4: Translating the GSN diagram in a description logic Abox.

Figure 5: Partial goals structure

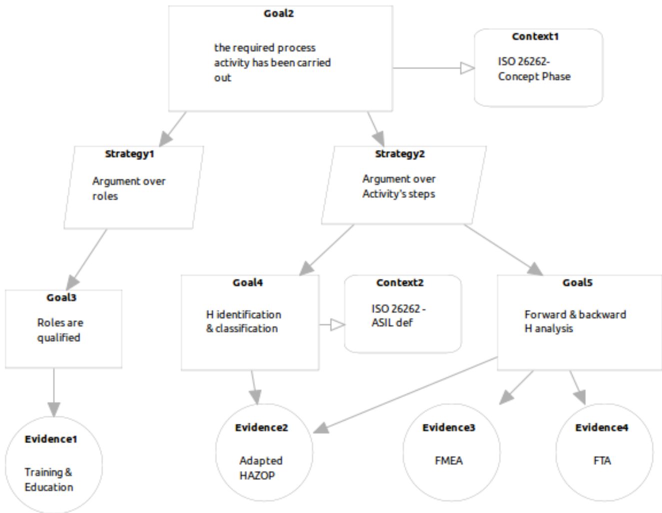


Figure 6: Goal structure for the process based argument.

specific concept, check the consistency of the Abox. In this way, the safety engineer can repair the problems and validate.

In our running scenario, after analysing the diagram, the engineer observers that g_3, g_4, g_5 are undeveloped goals that needs evidences or have to be divided into sub-goals. If the engineer provides evidence for g_3 then the goal will no longer be part of undeveloped goals.

The following formal verifications are provided by the *SafeEd* system:

1. Every node can be traced back to the top-level claim. That is, there are no "dangling" nodes or sets of nodes.

2. Each "leaf" node should either evidence or a reference to some previously reviewed assurance case

3. Circular reasoning: identified by the RacerPro engine in the form of cycle concepts

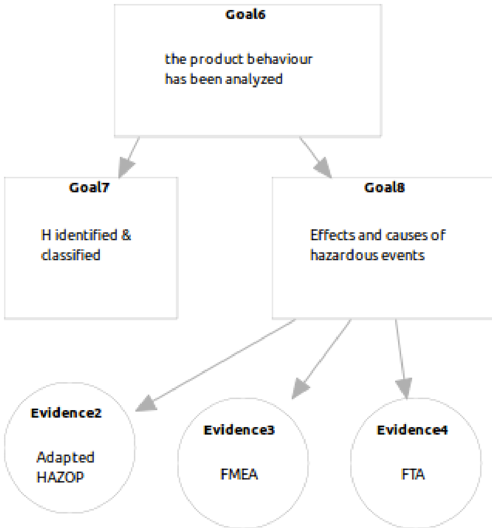


Figure 7: Goal structure for the product based argument.

Table 1: Retrieving information about the safety case.

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_2	$(retrieve-individual-fillers\ g_2\ hasEvidence)$	e_1, e_2, e_3, e_4
Undeveloped Goals	$(concept-instances\ UndevelopedGoals)$	g_3, g_4, g_5
Generate OWL	$(save-kb``PATH/kb.owl''\ :syntax:owl)$	
Check if Abox is consistent	$(abox-consistent?)$	
Get all contexts of a specific goal	$(individual-fillers\ g_1\ inContextOf)$	c_1, c_2

0.4.3 Generation of Safety Case Metrics

Complementarily to supporting semantic reasoning, our system provides also quantitative assessment of a safety case through several metrics developed.

The metrics are developed with the LISP API of RacerPro system. For instance, the number of non-verified goals for safety case given as the ABox $sc1$ is computed with:

$$(length\ (concept-instances\ NotVerifiedGoal))$$

The main use case of metrics is to assess the progress during different stages of validating the safety case. Given large safety cases, one can monitor the rate to which the number individuals of type $NotVerifiedGoal$ decreases.

0.4.4 Generating Natural Language Reports on the Safety Case

The tool supports the generation of documentation and reports for the safety case. As technical instrumentation, we use use the RacerPro engine to further translate the ontology from description logic syntax into Web Ontology Language (OWL). The OWL file is fed to NaturalOWL [3] engine to transform the owl ontology into natural language and save the files as pdf. The generated files will contain texts describing individuals or classes of individuals from owl ontology.

Also reports with what still needs to be done or a report containing the assessment and validation of the safety case can be generated. An example can be found in figure 8 representing a validation report generated by our tool for the car overtaking safety case represented in Fig. 3. The report includes:

- nodes that do not have a description;
- elements that are not linked directly or indirectly through other elements of the diagram to the top level goal;
- goals that do not have evidence or solution;
- incomplete goals that have undeveloped sub-goals.

The report provides also quantitative information of the diagram, in terms of number of nodes and their types. With this report the safety engineer knows at any moment what still needs to be added to the safety case to have a complete and well-build safety case. Having the diagram and diagram documentation facilitate the work of the safety engineer or certification auditors.

```
===================06/04/14 08:48:20=====================================
All the nodes have claims!

Evidence or solution must be provided for the following goals:
G3, G4, G5
G3 goal parents: G1
G4 goal parents: G1
G5 goal parents: G1

G1 goal has undeveloped childs

All goals are linked to the top-level goal!
==========================================================================
```

Figure 8: Example of a validation report.

0.5 Interleaving Argumentation and Model Checking

Given a Kripke structure $\mathcal{M}$ and a formula ϕ, with $\mathcal{M}\neg \models \phi$, the task of *model repair* is to obtain a new model $\mathcal{M}'$ such that $\mathcal{M}' \models \phi$. We consider the following primitive update operations [15].

[Primitive update operations] Given $\mathcal{M} = (S, R, L)$, the updated model $\mathcal{M} = (S', R', L')$ is obtained from $\mathcal{M}$ by:

1. (PU_1) Adding one relation element: $S' = S$, $L' = L$, and $R' = R \cup \{(s_i, s_j)\}$ where $(s_i, s_j) \notin R$ for two states $s_i, s_j \in S$.

2. (PU_2) Removing one relation element: $S' = S$, $L' = L$, and $R' = R \setminus \{(s_i, s_j)\}$ where $(s_i, s_j) \notin R$ for two states $s_i, s_j \in S$.

3. (PU_3) Changing labeling function in one state: $S' = S$, $R' = R$, $s^* \in SL'(s^*) \neq L(s^*)$, and $L'(s) = L(s)$ for all states $s \in S \setminus \{s^*\}$.

4. (PU_4) Adding one state: $S' = S \cup \{s^*\}, s \notin S$, $R' = R$, $\forall\, s \in S, L'(s) = L(s)$.

Our task is to build an argumentative based decision procedure that takes as input a model $\mathcal{M}$ and a formula ϕ, it outputs a model $\mathcal{M}'$ where ϕ is satisfied. The task addressed here focuses on a situation on which the specification of the model is not consistent. Consider the following two "rules of the air" [13]:

R_3: *Collision Avoidance – "When two UAVs are approaching each other and there is a danger of collision, each shall change its course by turning to the right."*

R_4: *Navigation in Aerodrome Airspace – "An unmanned aerial vehicle passing through an aerodrome airspace must make all turns to the left unless [told otherwise]."*

Let

$$\mathcal{A}_2 \;=\; \left\{ \begin{array}{l} \mathit{alter\_course}(\mathit{uav_1}, \mathit{right}) \prec \mathit{aircraft}(\mathit{uav_1}), \mathit{aircraft}(\mathit{uav_2}) \\ \qquad \mathit{collision\_hazard}(\mathit{uav_1}, \mathit{uav_2}) \\ \mathit{collision\_hazard}(\mathit{uav_1}, \mathit{uav_2}) \prec \mathit{approaching\_head\_on}(\mathit{uav_1}, \mathit{uav_2}), \\ \qquad \mathit{distance}(\mathit{uav_1}, \mathit{uav_2}, X), X < 1000 \end{array} \right\}$$

in the argument $\langle \mathcal{A}_2, \mathit{alter\_course}(\mathit{uav_1}, \mathit{right})\rangle$, a collision hazard occurs when two aerial vehicles uav_1 and uav_2 approach head on, and the distance between them is smaller than a threshold. The collision hazard further triggers the necessity to alter the course to the right, according to the R_3 specification. Let

$$\mathcal{A}_3 \;=\; \left\{ \begin{array}{l} \mathit{alter\_course}(\mathit{uav_1}, \mathit{left}) \prec \mathit{aircraft}(\mathit{uav_1}), \mathit{nearby}(\mathit{uav_1}, \mathit{aerodrom}), \\ \qquad \mathit{change\_direction\_required}(\mathit{uav_1}) \\ \mathit{change\_direction\_required}(\mathit{uav_1}) \prec \mathit{collision\_hazard}(\mathit{uav_1}, \mathit{uav_2}) \end{array} \right\}$$

in the argument $\langle \mathcal{A}_3, \mathit{alter\_course}(\mathit{uav_1}, \mathit{left})\rangle$, if a change of direction is required in the aerodrome airspace, the direction should be altered to the left. A possible conflict occurs between arguments $\langle \mathcal{A}_2, \mathit{alter\_course}(\mathit{uav_1}, \mathit{right})\rangle$ and $\langle \mathcal{A}_4, \sim \mathit{alter\_course}(\mathit{uav_1}, \mathit{right})\rangle$ where:

$$\mathcal{A}_4 \;=\; \left\{ \sim \mathit{alter\_course}(\mathit{uav_1}, \mathit{right}) \prec \mathit{alter\_course}(\mathit{uav_1}, \mathit{left}) \right\}.$$

The command $\langle \mathcal{A}_5, \sim \mathit{alter\_course}(\mathit{uav_1}, \mathit{left})\rangle$ conveyed from the ground control system to change direction to the right acts as a defeater for the argument $\mathcal{A}_3$, where (notice that strict rules should not form part of argument structures as they are not points of attack, we abuse the notation here just for emphasis):

$$\mathcal{A}_5 \;=\; \{ \; \sim alter\_course(uav_1, left) \;\leftarrow\; conveyed\_command\_course(uav_1, right) \; \}$$

Assume that the current model $\mathcal{M}$ satisfies the specification R_3. The problem is how to repair $\mathcal{M}$ with the model $\mathcal{M}'$ which also satisfies R_4. Our solution starts by treating rules R_3 and R_4 as structured arguments. The conflict between them are solved by a defeasible theory encapsulated as DeLP program, which outputs a dialectical tree of the argumentation process. The information from this tree is further exploited to decide which primitive update operations PU_i are required to repair the model.

Firstly, consider the uav_1 is in the obstacle detect $od \in S$ state, where S is the set of states in $\mathcal{M}$ with the labeling function $L(od) = \{uav_2, \neg a\}$. It means that uav_1 has detected another aerial vehicle uav_2. Assume that in this state the DeLP program will warrant the opposite conclusion a. This triggers the application of the primitive operation PU_3 which updates the labeling function $L(od) = \{uav_2, \neg a\}$ with $L'(od) = \{uav_2, a\}$.

Secondly, assume that the DeLP program based on the state variables uav_2, and $\neg a$ and the nominal od infers a relation r_i between od and another nominal $i \in \mathcal{N}$ of the model. The repair consists of applying the operation PU_1 on $\mathcal{M}$, where the relation set R' is extended with a relation between the two states ob and i: $R' = R \cup \{(od, i)\}$. The reasoning mechanism is possible because hybrid logic provides the possibility to directly refer to the states in the model, by means of nominals.

Thirdly, the program can block the derivation of a relation r between the current state and a next state. For instance, if $L(od) = \{uav_2, a\}$ and the argument A_3 succeeds, the transition between state od and state $turn\_right$ can be removed. Formally, $R' = R \setminus \{(od, turn\_right)\}$.

Fourthly, if the DeLP program warrants, based on the current state variable and available arguments, a nominal i which does not appear in S, the set of states is extended with this state: $S' = S \cup \{i\}$.

These four heuristics are illustrated in the following section, by verifying the specifications in hybrid logics on the updated models.

0.6 Model Repair for an Unmanned Aircraft Vehicle

0.6.1 Illustrative Example

We consider the scenario presented in [14], referring to the safe insertion of an Unmanned Aircraft Vehicle (UAV) into the civil air traffic. The scope is to demonstrate that safety requirements are being met by such an UAV so that they do not interfere or put in danger human controlled aircrafts. A mission is considered safe if all the major risks for the UAV are identified and managed (e.g. collision with other objects or human-piloted aircrafts and loss of critical functions). An UAV comes equipped with an autonomous control system, responsible for decision making during the mission and keeps a communication link open with a ground-base system (GBS), which provides all the required coordinates for the UAV.

The autonomous decision making performed by the UAV control system must consider the general set of safety regulations imposed to a UAS during a mission at all times.

We propose a solution for modeling such Unmanned Aircraft Systems (UASs) in compliance to the set of safety regulations. We will zoom over the following subset of the "Rules of the Air" dealing with collision avoidance:

R_1: *Obstacle Detection – "All obstacles must be detected within an acceptable distance to allow performing safely the obstacle avoidance maneuver"*

R_2: *Obstacle Avoidance – "All obstacles must be avoided by performing safely a slight deviation from the preestablished path and an immediate return to the initial trajectory once all collision risks are eliminated."*

R_3: *Collision Avoidance – "When two UAVs are approaching each other and there is a danger of collision, each shall change its course by turning to the right."*

The first rule states that all obstacles (e.g. human-controlled aircrafts, other UAVs, etc.) that are interfering with the initial trajectory of the UAV must be signaled within a certain limit of time such that to allow avoidance maneuvers to be performed by the UAV in safe conditions. The avoidance maneuver as shown by rules R_2 and R_3 consists of a slight change of the initial path to the right such that to allow the safe avoidance of the approaching UAV followed by a repositioning on the initial trajectory.

0.6.2 Kripke Model for the Unmanned Aerial Vehicle

We will further represent the behavior of the UAV noted by uav_1 captured in an obstacle avoidance scenario. The following states will be considered in constructing the Kripke model: path-following (pf), obstacle detection(od), turn left(tl) and turn right(tr). To each state we will attach the boolean state variable uav_2, which will indicate the presence or absence of another approaching UAV. In the path-following state pf, the UAV uav_1 performs a waypoint following maneuver, which includes periodical turns to the left or to the right. The appearance of an obstacle ($uav \rightarrow \top$) leads to the transition of the UAV into obstacle detection state od and from there in turn right tr state as part of the obstacle avoidance maneuver, followed by a return to the initial path-following state.

The initial model $\mathcal{M}_0$ is presented below:

$$\mathcal{M}_0 = \langle \{od, tr, tl, pf\},$$

$$\{r_0, r_1, r_2, r_3, r_4, r_5, r_6\},$$

$$\{(pf, \{\neg uav_2\}), (od, \{uav_2\}), (tr, \{\neg uav_2\}), (tl, \{\neg uav_2\})\}\rangle$$

The corresponding hybrid Kripke structure is illustrated in Figure 9.

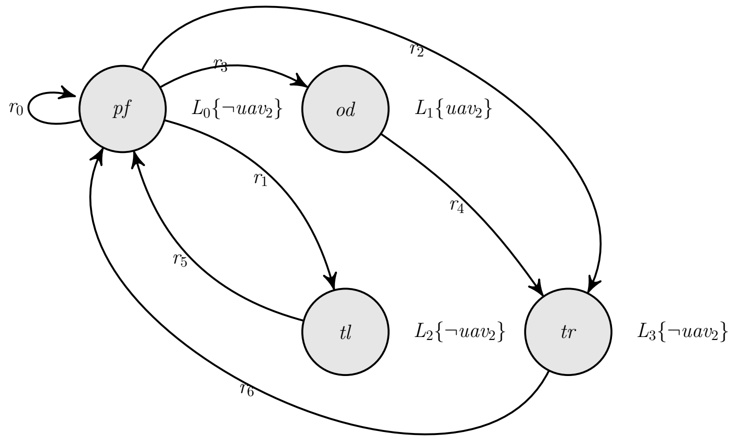


Figure 9: Kripke Model for the UAV.

0.6.3 Verifying Compliance to Safety Regulations

Once the modeling of the UAS is done, we have to verify whether the mentioned safety regulations hold for this model. To be able to perform model checking, we will further express the two safety regulations using hybrid logics:

$$R_1 : [Next](od) \rightarrow tr \tag{1}$$

The above formula corresponds to the first safety regulation R_1 and states that once the *od* (*ObstacleDetect*) state is reached then the immediate transition step should be done towards an avoidance maneuver state, for our case here, state *tr*, meaning that the obstacle was detected in time and it allowed the avoidance maneuver to be safely performed.

$$R_2 : [Next](tr \vee tl) \rightarrow pf \tag{2}$$

The formula corresponding to safety regulation R_2 states that all the next transitions from the *TurnRight* or *TurnLeft* state should always lead to the *PathFollow* state.

The formula below corresponding to safety regulation R_3 states that if another UAV is detected in the *od* (*ObstacleDetect*) state then all next transitions should be done towards the state *tr* (*TurnRight*):

$$R_3 : \bullet_{od}\, uav_2 \rightarrow ([Next]od \rightarrow tr) \tag{3}$$

Model checking is performed to verify whether the formulas hold or not for that model. To perform the model checking automatically, the Kripke structure corresponding to the UAS model is translated into an XML file and given as input for the Hybrid Logic Model Checker (HLMC) [5]. Each formula in HL is also given as input to the HLMC. Once the tests are performed for each formula against the Kripke model, we can complete the verification of the model. The result confirms that the modeled Kripke structure of the UAS complies with the defined safety regulations.

0.6.4 Adapting the Model to New Specifications

We consider again the UAV scenario and we will present a solution for modeling the existing UAS to include the introduction of new rules. For this, we will consider the initial set of rules extended by a newly adopted norm for UAVs navigating in an Aerodrome Airspace:

R_4: *Navigation in Aerodrome Airspace – "An unmanned aerial vehicle passing through an aerodrome airspace must make all turns to the left [unless told otherwise]."*

As a first step we will check whether the existing UAS model complies to the new regulation R_4. For this we will express the new rule as a HL formula and we will add to each possible state the boolean variable a, which will become true when the UAV enters an aerodrome airspace:

$$R_4 : @_i a \to ([Next]i \to (\neg tr)) \tag{4}$$

The formula states that all transitions from the states in which the state variable aerodrome a holds should not lead to the tr (*TurnRight*) state, the only state which is forbidden when navigating inside the aerodrome space. Since the only states from which turns are possible are pf and od, we will consider only this subset for model checking. One can observe that the formula does not hold for the existing model. Considering that the aerodrome a state variable is true in the od (*ObstacleDetect*) state, one can observe that the only allowed transition in the current model is to the tr (*TurnRight*) state. Therefore, the existing model does not comply to the new regulation. Moreover, from the pf state transitions are possible to the tl (*TurnLeft*) state, but also to the tr (*TurnRight*) states. We argue that the existing model could be extended to include also the new rules without having to construct a new model from the beginning. Although different solutions were proposed for Kripke Model repairing [1], we propose a solution based on argumentation for extending the model such that it complies to the updated set of regulations.

As a first step in our approach, we represent several possible extensions to the Kripke Model as defeasible arguments and include them in DeLP for choosing the best possible solution between different conflicting arguments. The proposed solution does not only eliminate the complexity of proposed repair/updating algorithms [1], but it allows the system to adapt to new information in a faster and more efficient manner.

Going back to our example, one can observe that there is no possibility for the UAV to go into the tl state once it has reached the od state, but only to the tr state. Since inside the aerodrome space, only turns to the left are permitted, then the link connecting *od* and *tr* (r_4) should be taken out from the model.

We will consider a new argument $\langle \mathcal{A}_6, \mathit{alter\_course}(\mathit{uav_1}, \mathit{left})\rangle$, which suggests updating rule R_3 by allowing the obstacles to be avoided to the left, instead of to the right when inside the aerodrome space, where:

$$\mathcal{A}_6 = \left\{ \begin{array}{l} \mathit{alter\_course}(\mathit{uav_1}, \mathit{left}) \prec \mathit{aircraft}(\mathit{uav_1}), \mathit{aircraft}(\mathit{uav_2}) \\ \qquad \mathit{collision\_hazard}(\mathit{uav_1}, \mathit{uav_2}) \mathit{nearby}(\mathit{uav_1}, \mathit{aerodrom}) \\ \mathit{collision\_hazard}(\mathit{uav_1}, \mathit{uav_2}) \prec \mathit{approaching\_head\_on}(\mathit{uav_1}, \mathit{uav_2}), \\ \qquad \mathit{distance}(\mathit{uav_1}, \mathit{uav_2}, X), X < 1000 \end{array} \right\}.$$

We argue that for compliance to the new regulations, we only need to change all the links in the model to point from the *od* and *pf* states only to the *tl* state instead of *tr* state to avoid the collision.

Therefore, we need to perform the following PU operations for updating the model:

1. (PU_2) Remove the relation elements (od, tr) and (pf, tr) such that we have: $S' = S$, $L' = L$, and $R' = R \setminus \{(od, tr), (pf, tr)\}$
2. (PU_1) Add the relation element (of, tl) such that we have: $S'' = S'$, $L'' = L'$, and $R'' = R' \cup \{(od, tl)\}$

However, the remove operation should be necessary only when that specific relation element causes a conflict between two arguments. In our case, if we consider arguments $\mathcal{A}_2$, sustaining the application of the initial rule R_2 and $\mathcal{A}_6$, sustaining a slight modification of the rule R_2 for navigation in aerodrome space, one can see that they do not attack each other as they offer solutions for different contexts: the $\mathcal{A}_2$ argument refers to collision avoidance outside the aerodrome space, while the $\mathcal{A}_6$ argument considers the case of collision avoidance when the UAV is nearby an aerodrome. A similar reasoning applies for the transition (pf, tr), which will be possible only when the state variable a does not hold at pf. Therefore, the PU_2 step can be left out and the updating of the model can be done only through a PU_1 operation. The decision to turn left or turn right will be taken in accordance to the value of the state variable a, which indicates the presence or absence of an aerodrome in the vicinity of the UAV.

We illustrate the update operation by adding a link r_7 from the *od* state to the *tl* state. Additionally, we attach to each state the boolean state variable a, such that it allows the UAV to perform only those transitions that comply to the set of regulations in different contexts, respectively inside or outside the aerodrome space. One can observe that if the UAV reaches the *od* state, then it will decide to perform the transition to the next state that has the same value for the state variable a as the *od* state. Therefore, if the UAV uav_1 detects another approaching UAV uav_2 and it is outside the aerodrome space ($\neg a$), it will look for the next possible state that has the same value for the a state variable. As one can see from Figure 10, the state that complies to this condition is tr. Also, if uav_1 is in the *pf* state and the state variable a holds at that state, then the possible transitions will be *tl* or *od*.

If uav_1 reaches the *od* state, while in the vicinity of an aerodrome, it will perform a transition to the *tl* state, where the state variable a also holds. If uav_1 reaches *pf* then it will perform a transition to either *tl* or *od* states. The other transitions from the model are not dependent on the state variable a, therefore they will remain the same as in the initial model. By adding the condition $\neg a$ for reaching state *tr*, we can avoid transitions to that state when a holds for the model.

The updated model $\mathcal{M}_1$ is presented below:

$$\mathcal{M}\infty_1 = \langle \{od, tr, tl, pf\},$$

$$\{r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7\},$$

$$\{(pf, \{\neg uav_2\}), (od, \{uav_2\}), (tr, \{\neg uav_2, \neg a\}), (tl, \{\neg uav_2\})\}\rangle$$

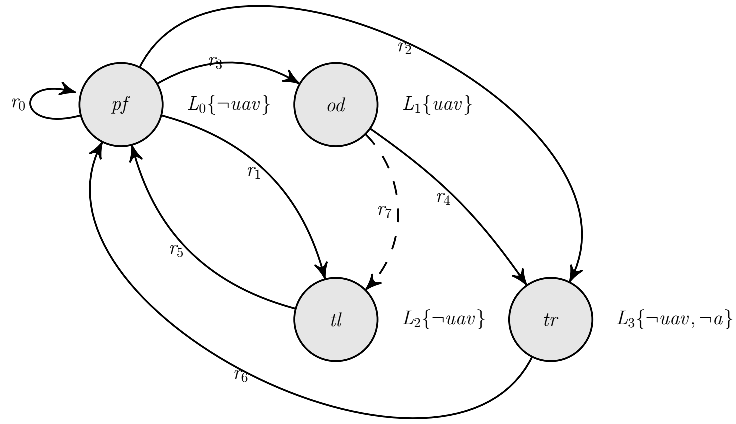


Figure 10: Extended Kripke model for the UAV compliant with the new regulation.

By checking next the R_1, R_2, R_3 and R_4 formulas against $\mathcal{M}\infty$, the results returned by HLMC showed that they hold for the updated model.

The illustrated example captures a simple scenario for UAV missions, but we argue that more complex conflicting situations can be handled by the presented argumentation framework.

Bibliography

[1] George Chatzieleftheriou, Borzoo Bonakdarpour, ScottA. Smolka, and Panagiotis Katsaros. Abstract model repair. In AlwynE. Goodloe and Suzette Person, editors, *NASA Formal Methods*, volume 7226 of *Lecture Notes in Computer Science*, pages 341–355. Springer Berlin Heidelberg, 2012.

[2] Mirko Conrad, Patrick Munier, and Frank Rauch. Qualifying software tools according to ISO 26262. In *MBEES*, pages 117–128, 2010.

[3] I. Androutsopoulos D. Galanis. Generating multilingual descriptions from Linguistically Annotated OWL Ontologies: the NaturalOWL system. 2007.

[4] Raghad Dardar, Barbara Gallina, Andreas Johnsen, Kristina Lundqvist, and Mattias Nyberg. Industrial experiences of building a safety case in compliance with ISO 26262. In *Software Reliability Engineering Workshops (ISSREW), 2012 IEEE 23rd International Symposium on*, pages 349–354. IEEE, 2012.

[5] Massimo Franceschet and Maarten de Rijke. Model checking hybrid logics (with an application to semistructured data). *Journal of Applied Logic*, 4:279–304, 2006.

[6] Sergio Alejandro Gómez, Anca Goron, and Adrian Groza. Assuring safety in an air traffic control system with defeasible logic programming. In *XLIII Jornadas Argentinas de Informática e Investigación Operativa (43JAIIO)-XV Argentine Symposium on Artificial Intelligence (ASAI)(Buenos Aires, 2014)*, 2014.

[7] Anca Goron, Adrian Groza, Sergio Alejandro Gómez, and Ioan Alfred Letia. Towards an argumentative approach for repair of hybrid logics models.

[8] Adrian Groza and Nicoleta Marc. Consistency checking of safety arguments in the goal structuring notation standard. In *IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP2014), Cluj-Napoca, Romania, 4-6 September 2014*, pages 59–66. IEEE, 2014.

[9] S.A. Gómez, A. Groza, and C.I. Chesñevar. An argumentative approach to assessing safety in medical device software using defeasible logic programming. In Simona Vlad and Radu V. Ciupa, editors, *International Conference on Advancements of Medicine and Health Care through Technology; 5th – 7th June 2014, Cluj-Napoca, Romania,*

volume 44 of *IFMBE Proceedings*, pages 167–172. Springer International Publishing, 2014.

[10] Volker Haarslev, Kay Hidde, Ralf Möller, and Michael Wessel. The RACER Pro knowledge representation and reasoning system. *Semantic Web*, 3(3):267–277, 2012.

[11] Carlos Chesñevar Ioan Letia Anca Goron Mauro Gómez Lucero Sergio Alejandro Gómez, Adrian Groza. Rgsafe: Usando argumentación para garantizar seguridad en sistemas técnicos complejos. In *XVI Workshop de Investigadores en Ciencias de la Computación (WICC 2014)Ushuaia, Tierra del Fuego, Argentina, 5 y 6 de mayo de 2014*, pages 86–90. 2014.

[12] Andrzej Wardziński. Safety assurance strategies for autonomous vehicles. In *Computer Safety, Reliability, and Security*, pages 277–290. Springer, 2008.

[13] Matt Webster, Michael Fisher, Neil Cameron, and Mike Jump. Formal methods for the certification of autonomous unmanned aircraft systems. In *Computer Safety, Reliability, and Security*, pages 228–242. Springer, 2011.

[14] Matt Webster, Michael Fisher, Neil Cameron, and Mike Jump. Model checking and the certification of autonomous unmanned aircraft systems. Technical Report ULCS-11-001, Department of Computer Science, University of Liverpool, Liverpool, United Kingdom, 2011.

[15] Yan Zhang and Yulin Ding. CTL model update for system modifications. *J. Artif. Intell. Res.(JAIR)*, 31:113–155, 2008.