Raport Tehnic II

Ianuarie 2014 - Decembrie 2014
Proiect bilateral Romania-Argentina
Contract No. 731

*
□◇Ṽ

Titlu proiect:	Utilizarea argumentarii pentru justificarea sigurantei sistemelor tehnice complexe
Partener din Romania:	Universitatea Tehnica din Cluj-Napoca
Partener din Argentina:	Universitatea Nationala de Sud
Durata:	24 luni: 11 Septembrie 2013 -11 Septembrie 2015)

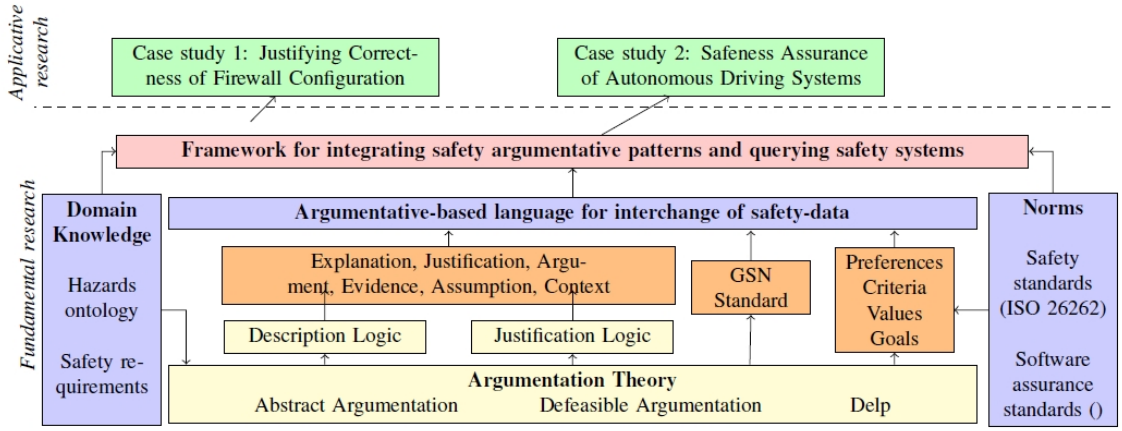
Cuprins

Plan	Obiective	Noutate	Activitati Asociate
Iun 2013	O1. Analiza si fundamentarea tehnico-'stiin'tific'a a justific'arii siguran'tei 'in sisteme tehnice complexe.	Identificarea posibilit'a'tilor de integrare a teoriei argument'arii, standardelor de calitate 'si ontologiilor.	Analiza formal'a a standardelor de siguran't'a. Identificarea factorilor care afecteaz'a confiden'ta 'in siguran'ta produselor software.
Sep 2013	O2. Construirea modelului de validare a siguran'tei pe baz'a de argumente.	Ra'tionare justificativ'a 'in contextul eviden'telor eterogene, valide, dar contradictorii.	Dezvoltarea logicii defezabile justificative. Contextualizarea eviden'telor.
Mai 2014	O3. Dezvoltarea sistemului de justificare a siguran'tei 'in sisteme tehnice complexe.	Identificarea automat'a a justific'arilor inconsistente.	Dezvoltarea unei ontologii generice a hazardurilor. Organizare seminar.
Sep 2014	O4. Aplicarea sistemului 'in justificarea siguran'tei 'in sistemele de condus vehicole autonome.	Organizarea eviden'telor, construirea argumentelor 'si contra-argumentelor pentru justificarea siguran'tei 'in sistemele de condus autonome.	Formalizarea cerin'telor de siguran't'a. Formalizarea asump'tiilor legate de mediul de operare 'si a hazardurilor specifice.
Dec 2014	O5. Aplicarea sistemului 'in verificarea configur'arii sistemelor de tip firewall.	Prezentarea argumentelor pentru suport decizional sub constrngeri temporale.	Identificarea inconsisten'telor 'in sisteme cu reguli de securitate. Organizare workshop.
Mar 2015	O6. Dezvoltarea unei metodologii de utilizare a argument'arii structurate 'in validarea siguran'tei.	Reutilizarea cazurilor de siguran't'a. Reingineria sistemelor software complexe pe baz'a de argumente.	Definirea 'sabloanelor de cazuri de siguran't'a. Enun'tarea principiilor de construc'tie a argumentelor 'in dezvoltarea unei aplica'tii critice. Identificarea altor aplica'tii pentru modelul dezvoltat.

Tabela 1: Planificarea obiectivelor specifice 'si caracterul inovativ al acestora.

0.1 Obiective

Obiective. Obiectivul general se refera la justificarea sigurantei sistemelor software prin utilizarea teoriei argumentarii.



Echipa:

- Universitatea Tehnica din Cluj-Napoca: Assoc. Prof. dr. eng. Adrian Groza, Prof. dr. eng. Ioan Alfred Letia, Phd student Anca Goron.
- Universitatea Nationala de Sud: Assoc. Prof. Sergio Alejandro Gomez, Prof. Carlos Ivan Chesnevar

Publicatii: List publicatiilor pentru anul 2014: [9, 6, 4, 7, 11, 5, 10]

1. Letia, Ioan Alfred, and Anca Goron. "Model checking as support for inspecting compliance to rules in flexible processes." Journal of Visual Languages Computing (2014).

2. 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

3. 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

4. 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

5. 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

6. 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

7. Letia, Ioan Alfred, and Anca Goron. "A temporal view on Model Checking Hybrid Logics." Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on. IEEE, 2014.

Lucrarea: *Assuring Safety in Air Traffic Control Systems with Argumentation and Model Checking*, Sergio Alejandro Gomez, Anca Goron,Adrian Groza, Ioan Alfred Letia se afla in evaluare la revista Expert Systems and Applications.

Deliverabile:

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

(D1.2) Poster de prezentare (disponibil pe pagina proiectului);

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

(D1.4) Ontologia standardului Goal Structuring Notation standard (disponibila pe pagina proiectului);

(D1.6) Raport tehnic pe primul an (disponibil pe pagina proiectului);

(D2.1) Unealta *EdSafe* (disponibil pe pagina proiectului);

(D2.2) Raport tehnic pe anul doi (disponibil pe pagina proiectului).

Noutate. Am integrat *argumentarea structurat'a cu logicile hibride* pentru repararea modelelor de stare ale sistemelor tehnice. Teoria argumentarii este utilizata in asistarea procesului de actualizare a modelului. Descrierea comportamentului este realizata prin modele Kripke hibride. Actualizarea modelului Kriple are loc 'in momentul 'in care sistemul trebuie sa se confomeze unor noi specificatii sau constrangeri normative. Cand o proprietate nu poate fi verificata pe modelul curent, un program in logica defezabila este utilizat pentru a analiza starea curenta. Pe baza statusului argumentelor, sistemul recomanda patru operatii primitive asupra modelului:

1. adaugarea unei variabile de stare
2. adaugarea unei noi tranzitii
3. eliminarea unei tranzitii din modelul Kripkpe curent
4. adaugarea unei noi stari

Scenariul utilizat pneutr validarea solutiei se refeta la comportamentul adaptiv al unei drone autonome. Instrumentatia tehnica utilizeaza Programarea in Logica Defezabila, respectiv Hybrid Logic Model Checker.

Garantarea sigurantei in sistemel tehnice complexe este un subiect esential in aplicatii precum controlul traficului sau dispozitive medicale.

In domeniul controlului aerian, am dezvoltat un *sisteme bazat pe teoria argumentarii pentru a asista controlorii de trafic in suportul deciziilor sub constrangeri normative*. Sistemul presupune ca mediul este dinamic, iar datele de la sensori sunt colectate continuu in timp real.

Tehnologia moderna de asistare medicare se bazeaza intr-o masura ridicata pe pachete software instalate in diferite dispozitive medicale. Acesta tehnologie software pe langa beneficiile aduse, poate introduce hazarduri suplimentare cu privire la siguranta pacientului. Astfel, garantarea sigurantei software-ului instalat pe dizpozitive medicale este considerat un subiect critic. Am dezvoltat o *metoda pentru evaluarea sigurantei dispozitivelor medicale pe baza semanticii argumentative a logicii defezabile*. Unealta supoort a fost DeLP (Defeasible Logic Programming), care a furnizat capabilitatile de rationare cu informatii incomplete. Argumentele strucurate analizate de Delp pot constitui o sursa de evidente cerute in timpul proceselui de audit tehnic inainte de aprobarea punerii in circulatie sau comercializarii unui dispozitiv medical. Beneficiul principal se refera la cresterea transparentei intre actorii implicati in certificare. Sistemul dezvotat a fost analizat si validat prin modelarea accidentului Therac-25.

Impact economic Organismele resposanbile cu certficarea diferitelor produse critice impun companiilor de dezvoltare software furnizarea de cazuri de siguranta explicite. Aceste cazuri sunt definite prin argumente structurate pe baza unor evidente obiective, scopul fiind de de demonstra ca un sistem este acceptabil din punct de vedere al sigurantei.

Cazurile de siguranta pe baza argumentelor structurate sunt adaptate deja in domeniile apararii (UK), rutier, feroviar, petrol, gaz sau medical. Ca urmare, cercetarea noastra a urmarit:

- Identificarea legaturilor dintre teoria argumentarii si ingineria sistemelor critice
- Dezvoltarea metodelor argumentative de transfer a confidentei in aplicatii critice
- Aplicarea instrumentatiei tehnice dezvoltate in doua scenarii: drone autonome, respectiv dispozitive medicale.

Capabilitatile sistemului dezvoltat includ:

1. Verificarea automata a respectarii normelor
2. Generarea de rapoarte cu privire la siguranta sistemului analizat
3. facilitarea intelegerii si a transferului de confidenta

0.2 Modelarea standardului GSN in Logica Descriptiva

Relatia *supportedBy*, permite documentare evidentelor utilizate. Legaturile permise conform standardului GSN pentru *supportedBy* sunt: tel-2-tel, tel-2-strategie, tel-2-solutie, strategie-2-tel. Axioma A_1 specifica domeniul rolului *supportedBy*:

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

Axioma A_2 specifica domeniul rolului *supportedBy*, iar axioma A_3 introduce rolul inverse *supports*. A_4 constrainge rolul*supportedBy* sa fie tranzitiv.

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

Relatiile de tip inferential declara ca exista o inferenta intru telurile argumentului. Ele specifica legatura dintre teluri si evidentele disponibile. Axiomele A_5 si A_8 specifica codomeniul rolurilor *hasInference*, respectiv *hasEvidence*, in timp ce A_6 si A_9 domeniul acestor roluri. Definitiile A_7 si A_{10} afirma ca *supportedBy* este rolul parinte pentru *hasInference* si *hasEvidence*, mostenind astfel constrangerile acestora.

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

Telurile si subtelurile sunt propozitii care se doresc a fi adevarate, fiind clasificate ca si calitative/cantitative sau demonstrabile/nesigure.

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

Un sub-tel sustinte teluri de nivel superior Un caz de siguranta are un singur tel la nivel superior *Goal*, care nu sustine ale teluri.

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

Elementele unui argument de siguranta sunt instantiate, iar o descriere textuala este astasata instantelor prin atributul *hasText* avand domeniul *Statement* si codomeniul *String*:

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

Trei indivizi *gt*, *gp* si *gu* de tipul tel si descrierile textuale sunt asertate de f_1 si f_6:

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

Pasi intermediari cu rol explicativ intre teluri precum si evidentele includ propozitii, refrinte, justificari si asumptii.

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

unde aceste concepte sunt disjuncte:

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

Evidentele sau solutiile reprezinta baza argumentelui, acestea incluzand de obicei analize specifice si teste. In abordarea noastra, o parde din evidente vin din verificarea formala a modelului.

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

Un tel neverificat este un tel care are cel putin o evidenta care nu a fost demonstrata formal.

$$(A_{28}) \quad NotVerifiedGoal \equiv Goal \sqcap \exists\, hasEvidence.NotVerifiedEvidence$$

$$(A_{29}) \quad NotVerifiedEvidence \equiv Evidence \sqcap \exists\, hasTestResult.(False \sqcap Unknown)$$

0.3 SafeEd Tool

Unealta dezvoltata consta dintr-un plug-in pentru Eclipse, fiind proiectata pe mai multe nivele (Fig. 1). Primul nivel consta din nucleul sistemului. Nivelul doi integreaza o serie de plug-inuri existente in Eclipse. Eclipse Modelling Framework (EMF) a fost utilizat pentru dezvoltarea acestui nivel. Graphical Modelling Framework (GMF) si the Graphical Editing Framework (GEF) au fost utilizate pentru dezvoltarea interfetei grafice. Epsilon pentru construirea componenetei responsabile pentru gestiunea taskurilor. Al treilea nivel contine metamodelele GSN si ARM. The GSN plugin implementeaza functionalitatea de editare pentru GSN. Pluginul Ontology faciliteaza rationarea peste ontologii. Nivelul de interfata grafica consta din editorul GSN, editorul ARM si uneltele de gestiune a modelului: i) transformare GSN to ARM ii) validarea GSN pe baza serviciilor de rationare in logica descriptiva iii) salvarea unui caz de siguranta ca Abox in ontologia GSN iv) interogarea cazului de siguranta v) diferite facilitati de editare GSN.

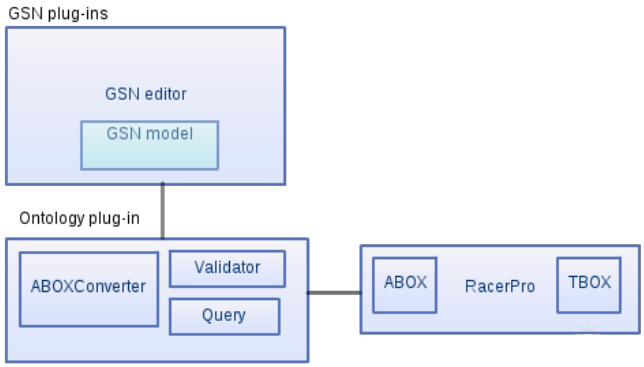


Figura 1: Arhitectura systemului

Principalul scop al uneltei este de a ajuta utilizatorul in constructia de cazuri de siguranta valide. Sistemul permite verificarea automata si rationarea pe diagramele dezvoltate.

Am dezvoltat o ontologie care fomalizeaza Goal Structuring Notation. Ontologia este incarcata prin pluginul Ontology in rationatorul RacerPro utlizand libraria jRacer. Prin conectarea la RacerPro [8], fiecare caz de siguranta este validat in conformitate cu axioele GSN. O serie de interogari spefifice pot fi adresate pe modelul dezvoltat.

Spatiul de lucru este prezentat in Fig. 2. Un proiect (top-left) consta dintr-un set de cazuri de siguranta, dezvoltate atat ca model grafic GSN (fisiere cu extensia gsn) sau ca un abox in logica descriptiva. (fisiere cu extensia racer). Sistemul este capabil sa translateze intre cele doua

formate. Dupa translatarea diagramei gsn in racer are loc validarea acesteia si generarea de rapoarte.

Fereastra principala (centru-sus) ilustreaza diagramele gsn active. Elementele standard GSN sunt reprezentate astfel: telurile cu dreptunghiuri, strategiile cu paralelograme, evidentele si solutiile cu cercuri, asumptiile si justificarile cu elipse, contextele prin dreptunghiuri cu colturile rotunjite. Relatia *supportedBy* reprezinta o sageata cu varful plin, iar relatia *inContextOf* o sageata cu varful gol. Titlul si descrierea unui nod pot fi introduse prin interactiune grafica. Diagrama este construita prin tehnica drag-and-drop (centru-dreapta).

Consola de comanda (centru-jos) indica pasii de rationare efectuati pe diagrama activa. Sintaxa interogarilor corespund syntaxei RacerPro. In Fig. 2, cele patru interogari exemplificate sunt:

- listarea telurilor existente in diagrama
- identificarea automata a telului principal
- enumerarea dovezilor care sustine telul g_2
- verificarea automata a consistentei diagraei din punctul de vedere al standardului GSN

  In partea stanga-jos, dreptunghoul rosu indica partea vizibila a diagramei cazului de siguranta activ.

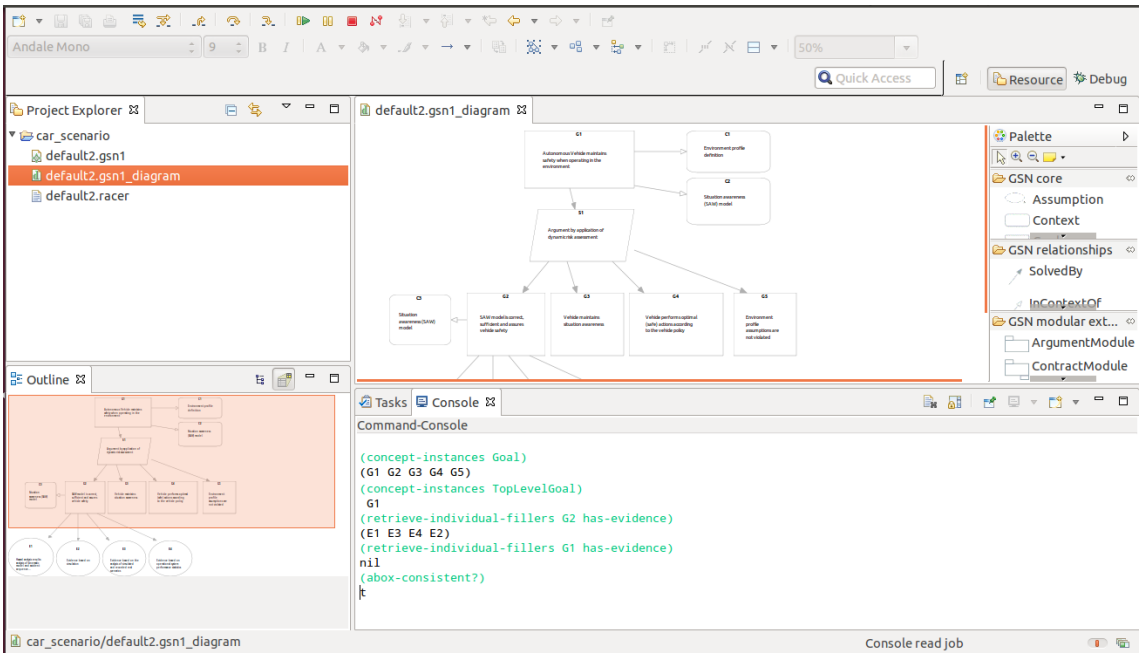


Figura 2: Interfata aplicatiei

0.4 Verificarea formala a cazurilor de siguranta

0.4.1 Vehicle Overtaking Scenario

Un exemplu de diagrama GSN dezvoltat in *SafeEd* este ilustrata in Fig. 3. Scenarioul considerat apartine domeniului de conducere autonoma a vehiculelor. Telul g_1 urmareste garantarea ca vehiculele autonome opereaza sigur in mediul prevazut. Telul are validate in doua contexte: existenta unui model formal al mediului de operare (contextul c_1), respectiv existenta unui mecanism de furnizarea a situatiei curente. O solutie pentru garantarea

sigurantei urmareste evaluarea dinamica a riscului [12]. Sub-telurile g_2, g_3, g_4 si g_5 sunt definete pentru a indeplini strategia s_1. De exemplu, sub-telul g_2 afirma corectitudinea modelului, afirmatie sustinuta de evidente, printre care verficarea formala e_2

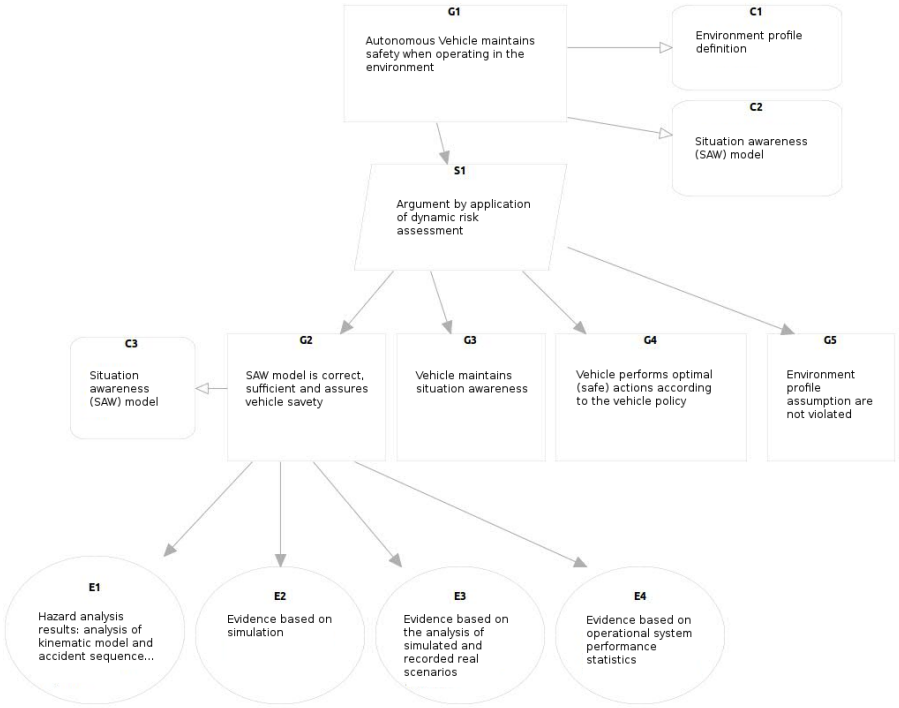


Figura 3: Scenariu in domeniu vehiculelor autonome

Fig. 3 este translatata automat intr-un Abox reprezentata in Fig. 4. Aici, faptele f_{51} - f_{54} aserteaza instantele la elementele corespunzatoare din GSN. Structura diagramei utilizeaza cele doua relatii *supportedBy* si *inContextOf*, fiind formalizata de faptele f_{55} to f_{62}. Textul in limbaj natural este modelat cu atribute concrete in logica descriptiva si libajul Racer [8] (asertiile f_{63} - f_{70}).

In figurile 5, 6, 7 doar analiza hazardului si evaluarea riscului sunt teluri dezvoltate. Sunt furnizate de asemenea argumentele pe baza de process (Goal 2) si pe baza de produs (Goal 6)

Argumentul pe baza procesulu de dezvoltare este rafinat in Fig. 6, iar argumentul pe baza de produs in Fig 7.

0.4.2 Validarea cazurilor de siguranta

RacerPro [8] este utilizat pentru a interoga si valida cazurile in sintaxa KRSS. Sistemul identifica automat telurile care nu au inca toate dovezile validate, descrierea telurilor sau generarea de explicatii de ce un anuit tel apartine unui concept, verificare consistentei Abox-ului. Sistemul ajuta astfel inginerul in procesul de validare a sistemului software critic.

In scenariul considerat, dupa analiza diagramei, inginerului i se specifica o lista cu teluri nedezvoltate g_3, g_4, g_5. In momentul in care agentul uman furnizeaza evidenta pentru g_3 telul nu mai apare ca instanta a conceptului *UndevelopedGoals*.

Urmatoarele verificari formale sunt furnizate de *SafeEd*:

1. Fiecare nod poate fi urmarit carul tel de nivel superior apartine. Sunt identificate automat nodurile care nu au aceasta proprietate.

(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*") : $hasText$

Figura 4: Translatarea automata a diagramei GSN in logica descriptiva.

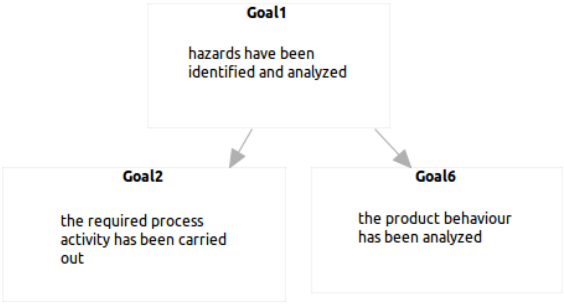


Figura 5: Structura partiala a telurilor

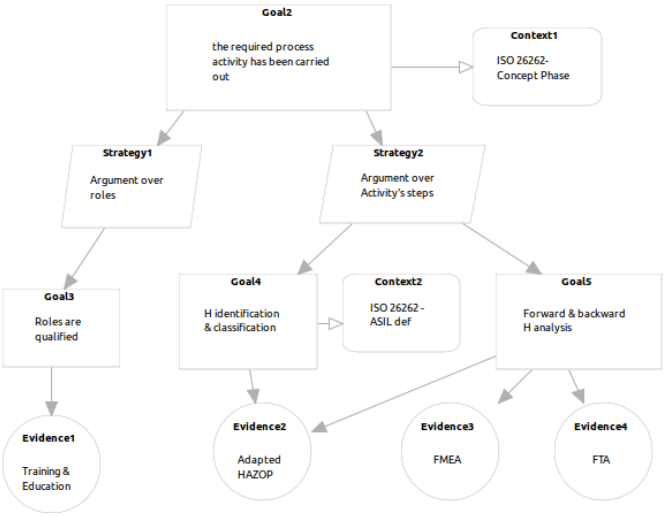


Figura 6: Argument pe baza procesului de dezvoltare.

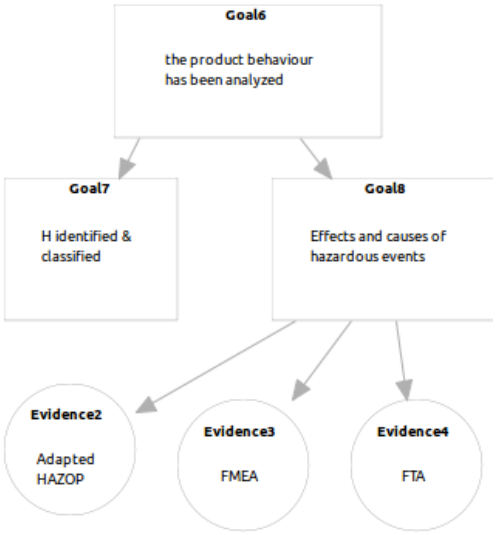


Figura 7: Argument pe baza produsului dezvoltat.

2. Fiecar nod "frunza" trebuie sa fie de tipul *Evidenta* sau o referinta la un caz de siguranta valid anterior
3. Identificarea conceptelor ciclice

Tabela 2: Interogarea si analiza unui caz de siguranta.

Interogare	RacerPro	Raspuns automat
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 Generarea metricilor de siguranta

Complementar rationarii semantice, sistemul furnizeaza metrici cantitative legate de cazul de siguranta considerat. Metricile au fost dezvoltate cu API-ul pentru LISP a rationatoruluu RacerPro De exemplu, numarul de teluri neverificate inca la un moment dat fiind dat un Abox $sc1$ este calculat cu:

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

Principalul beneficiu al acestor metrici cantitative este de a putea estima progresul in timpul procesulu de analiza si evaluarea a unui caz de siguranta. De exemplu, fiind dat un caz de siguranta real (sute te teluri si dovezi asociate) se poate monitoriza modul de descrestere a numarului de indivizi in clasa $NotVerifiedGoal$.

0.4.4 Generare de rapoarte in limbaj natural

Unealta poate genera documentatii si rapoarte in limbaj natural referitor la cazurile de siguranta analizate. Ca instrumentatie tehnica utilizam RacerPro pentru translatarea ontologiei in format OWL. Ontologia OWL este furnizata sistemului NaturalOWL [2]care genereaza un documt pdf in limabj natural cu analiza cazului continut in Abox Un exemplu este ilustrat in figura 8 Raportul include

- nodurile care nu au o descriere;
- elementele care nu sunt legate direct sau indirect telului general $g1$
- telurile care nu au evidente sau solutii atasate;
- telurile incomplete care se bazeaza pe sub-teluri inca incomplete.

Raportul include de asemenea informatii cantitative ale diagramei analizate. Cu o astfel de instrumentatie inginerul stie in orice moment elementele care mai trebuie adaugate diagramei pentru a obtine un caz de siguranta complet.

0.5 Intercalarea argumentarii cu metode de verificare formala

Avand o structura Kripke $\mathcal{M}$ si o formula ϕ, with $\mathcal{M}\neg \models \phi$, sarcina de *reparare model* este de a obtine un nou model $\mathcal{M}'$ astefl incat $\mathcal{M}' \models \phi$. Consideram urmatoarele patru operatii primitive de actualizare [15].

[] Avand $\mathcal{M} = (S, R, L)$, modelul actualizat $\mathcal{M} = (S', R', L')$ este obtinut din $\mathcal{M}$ prin:

```
==================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!
=========================================================================
```

Figura 8: Raport in limbaj natural pentru validarea diagramei GSN.

1. (PU_1) Adaugarea unui element de relatie: $S' = S$, $L' = L$, 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) Eliminarea unui element de relatie: $S' = S$, $L' = L$, 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) Schimbarea functiei de etichetare pentru o stare: $S' = S$, $R' = R$, $s^* \in SL'(s^*) \neq L(s^*)$, si $L'(s) = L(s)$ pentru toate starile $s \in S \setminus \{s^*\}$.
4. (PU_4) Adaugarea unei stari noi: $S' = S \cup \{s^*\}, s \notin S$, $R' = R$, $\forall\, s \in S, L'(s) = L(s)$.

Sarcina noastra este sa implementam o procedura de decizie pe baza de argumente care foloseste ca si date de intrare un model $\mathcal{M}$ si o formula ϕ, returnand un modell $\mathcal{M}'$ in care ϕ este satisfacut. Sarcina adresata aici este focalizata pe o situatie in care specificatiile modelului nu sunt consistente. Consideram urmatoarele doua "reguli ale aerului" [13]:

R_3: *Evitarea Coliziunilor – "In momentul in care doua drone se apropie una de alta existand pericol de coliziune, fiecare dintre ele isi va schimba cursul printr-o intoarcere la dreapta."*

R_4: *Navigarea in Spatiul unui Aerodrom – "Un vehicula erian autonom aflat in trecere prin spatiul aerian al unui aerodrom trebuie sa efectueze toate intoarcerile spre stanga in afara cazului in care [este altfel specificat]."*

Fie

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

in argumentul $\langle \mathcal{A}_2, alter\_course(uav_1, right) \rangle$, se mentioneaza faptul ca un pericol de coliziune are loc in momentul in care doua vehicule aeriene uav_1 si uav_2 se apropie una de alta, si distanta dintre ele este mai mica decat un minim prestabilit. Pericolul de coliziune subliniaza necesitatea de a schimba cursul la dreapta, in conformitate cu regula R_3 specification. Fie

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

in argumentul $\langle \mathcal{A}_3, alter\_course(uav_1, left)\rangle$, daca o schimbare de directie este necesara in spatiul aerian al aerodromului, este sustinut faptul ca directia trebuie schimbata spre stanga. Un posibil conflict apare intre argumentele $\langle \mathcal{A}_2, alter\_course(uav_1, right)\rangle$ si $\langle \mathcal{A}_4, \sim alter\_course(uav_1, right)\rangle$ unde:

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

Comanda evidentiata de $\langle \mathcal{A}_5, \sim alter\_course(uav_1, left)\rangle$ de schimbare a directiei la dreapta convenita de sistemul de control de la sol actioneaza ca si un *defeater* (invingator) pentru argumentul $\mathcal{A}_3$, unde (a se nota faptul ca regulile stricte nu trebuie sa fie incluse in structura argumentului intrucat nu sunt puncte de atac, abuzand in acest caz de notatie strict pentru o mai buna evidentiere):

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

Presupunem faptul ca modelul curent $\mathcal{M}$ satisface specificatia R_3. Problema este cum sa reparam $\mathcal{M}$ cu modelul $\mathcal{M}'$ care satisface si R_4. Solutia noastra incepe prin tartarea regulilor R_3 and R_4 ca argumente structurate. Conflictul dintre ele este solutionat de o teorie anulabila inclusa in programul DeLP, ce returneaza un arbore dialectic al procesului argumentativ. Informatia din arbore este exploatata in continuare pentru a decide asupra operatiilor primitive de actualizare PU_i necesare pentru repararea modelului.

In primul rand, consideram faptul ca vehiculul uav_1 se afla in starea de detectie obstacole $od \in S$, unde S este setul de stari in $\mathcal{M}$ cu functia de etichetare $L(od) = \{uav_2, \neg a\}$. Inseamna ca uav_1 a detectat un alt vehicul aerian uav_2. Afirmam ca in starea respectiva programul DeLP va garanta concluzia opusa a. Astfel se aplica operatia de actualizare PU_3 care actualizeaza functia de etichetare $L(od) = \{uav_2, \neg a\}$ cu $L'(od) = \{uav_2, a\}$.

In al doilea rand, daca programul DeLP este bazat pe variabilele de stare uav_2, si $\neg a$ si nominalul od infera o relatie r_i intre od si un alt nominal $i \in \mathcal{N}$ al modelului. Repararea consta in aplicarea operatiei PU_1 pe $\mathcal{M}$, unde multimea relatiilor R' este extinsa printr-o relatie intre doua stari ob si i: $R' = R \cup \{(od, i)\}$. Mecanismul de rationare este posibil intrucat logica hibrida ofera posibilitatea de a ne referi in mod direct la starile modelului prin intermediul nominalelor.

In al treilea rand, programul poate bloca derivarea unei relatii r intre starea curenta si starea urmatoare. De exemplu, daca $L(od) = \{uav_2, a\}$ si argumentul A_3 invinge, tranzitia intre starile od si $turn\_right$ poate fi eliminata. In mod formal, $R' = R \setminus \{(od, turn\_right)\}$.

In al patrulea rand, daca programul DeLP garanteaza, pe baza variabilei de stare curente si a argumentelor disponibile, avand un nominal i care nu apare in S, multimea starilor este existinsa cu starea: $S' = S \cup \{i\}$.

Aceste patru aspecte sunt ilustrate in sectiunea urmatoare, prin verificarea specificatiilor in logica hibrida pe modelele actualizate.

0.6 Repararea modelului pentru un vehicul aerian autonom

0.6.1 Examplu ilustrativ

Consideram scenariul prezentat in [14], referitor la insertia sigura a unui Vehicul Aerian Autonom (UAV) in spatiul aerian dedicat traficului civil. Scopul este de a demonstra faptul ca regulile de siguranta sunt respectate pentru un astfel de UAV astfel incat acestea sa nu interfereze sau sa puna in pericol vehiculele aeriene controlate de oameni. O misiune este

considerata sigura daca toate riscurile majore pentru UAV sunt identificate si rezolvate (ex. coliziunea cu alte obiecte sau cu vehiculele aeriene pilotate de oameni si pierderea functiilor critice). Un UAV vine echipat cu un sistem de control automat, responsabil pentru luarea deciziilor pe durata misiunii si tine o legatura de comunicare deschisa cu un sistem de la sol (GBS), ce ofera toate coordonatele necesare pentru UAV. Decizia automata luata de sistemul de control al UAV-ului trebuie sa ia in considerare setul general de reguli de siguranta impuse unui sistem aerian autonom aflat in misiune.

Propunem o solutie pentru modelarea UAS-urilor (Sistemele Aeriene Autonome) in conformitate cu setul de reguli de siguranta.

Ne vom focusa pe urmatoarele "Reguli ale Aerului" care iau in considerare detectarea coliziunilor:

R_1: *Decetare Obstacol – "Toate obstacolele trebuie detectate de la o distanta acceptabila care sa permita realizarea in siguranta a manevrei de evitare a obstacolului."*

R_2: *Evitare Obstacol – "Toate obstacolele trebuie evitate prin realizarea in siguranta a unei devieri de la calea predefinita si o revenire imediata la traiectoria initiala in momentul in care toate riscurile de coliziunie sunt eliminate."*

R_3: *Evitare Coliziune – "In momentul in care doua UAV-uri se apropie una de cealalta si exista pericolul unei coliziuni, fiecare isi va schimba cursul printr-o intoarcere la dreapta."*

Prima regula se refera la faptul ca toate obstacolele (e.g. sisteme de zbor controlate de om, drone, etc.) care interfereaza cu traiectoria initiala a UAV-ului trebuie reperate intr-o anumita limita de timp care sa permita realizarea manevrelor de evitare in siguranta de catre UAV-uri. Manevra de evitare, dupa cum este specificat in regulile R_2 si R_3, consista din o deviere a caii initiale la drepata pentru a permite evitarea in siguranta a UAV-ului care se apropie, manevra urmata de o repunere pe traiectoria prestabilita.

0.6.2 Model Kripke pentru vehiculul aerian autonom

Vom prezenta in continuare comportamentul UAV-ului reprezentat prin notatia uav_1 surprins intr-un scenariu de evitare a obstacolelor. Urmatoarele stari vor fi considerate pentru constructia modelului Kripke: urmare cale (pf), detectare obstacol (od), intoarcere stanga (tl) si intoarcere dreapta (tr). Fiecarei stari ii vom atasa variabila booleana uav_2, cu rolul de a indica prezenat sau absenta unui alt UAV care se apropie. In starea de urmarire cale prestabilita pf, UAV-ul uav_1 realizeaza o manevra de urmarire a coordonatelor prestabilite, ce include intoarceri periodice la stanga sau la dreapta. Aparitia unui obstacol ($uav \rightarrow \top$) duce la trecerea UAV-ului UAV in starea de detectie obstacol od si de acolo in starea de itnoarcere spre dreapta tr ca parte a manevrei de evitare obstacol, urmata de o revenire la traiectoria initiala.

Modelul Kripke initial $\mathcal{M}_0$ este prezentat mai jos:

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

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

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

Structura hibrida Kripke corespunzatoare este ilustrata in figura 9.

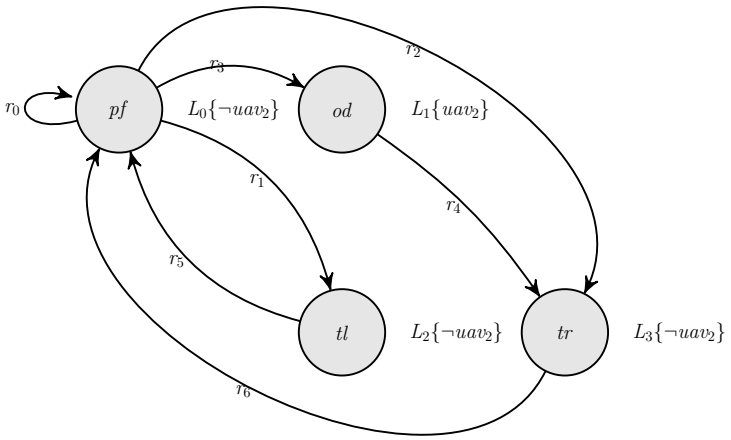


Figura 9: Model Kripke pentru UAV.

0.6.3 Verificare conformare la regulile de siguranta

O data ce modelarea UAS este realizata, trebuie sa verificam daca regulile de siguranta mentionate sunt respectate pentru model. Pentru a putea realiza verificarea formala, vom exprima cele doua reguli de siguranta cu ajutorul logicii hibride:

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

Formula d emai sus corespunde primei reguli de siguranta R_1 si spune ca o data cu trecerea in starea de detectie obstacol *od* (*ObstacleDetect*) atunci o manevra de evitare obstacol trebuie sa urmeze, in cazul nostru, sa se realizeze trecerea in starea *tr*, ceea ce inseamna ca obstacolul a fost detectat la timp si ca a permis realizarea manevrei de evitare in siguranta.

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

Formual corespunzatoare regulei de siguranta R_2 spune ca toate trecerile din starile *TurnRight* si *TurnLeft* trebuie sa fie in starea *PathFollow*.

Formula corespunzatoare regulei de siguranta R_3 spune ca daca un alt UAV este detectat in starea *od* (*ObstacleDetect*) atunci toate tranzitiile ce urmeaza trebuie sa fie spre starea *tr* (*TurnRight*):

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

Verificarea formala este aplicata pentru validarea formulelelor pentru modelul initial. Pentru automatizarea procesului de verificare, structura Kripke corespunzatoare modelului UAS-ului este redata intr-un fisier XML si utilizata ca date de intrare pentru Hybrid Logic Model Checker (HLMC) [3]. Fiecare formula in LH este de aseemneea utilizata ca data de intrare in HLMC. Odata ce fiecare formula este testata pentru modelul Kripke, verificarea formala este completa. In cazul de fata, rezultatele, confirma conformarea modelului UAS-ului redat ca structura Kripke la regulile de siguranta predefinite.

0.6.4 Adaptarea modelului la specificatii noi

Ne referim din nou la scearіul cu UAV-uri si prezentam in cele ce urmeaza o solutie pentru mdoelarea UAS-ului astfel incat s aincluda si regulile noi introduse. Pentru aceasta vom considera setul initial de reguli extins cu cu norma nou adoptata cu privire la navigarea UAV-ului in Spatiul Aerian al Aerodromului:

R_4: *Navigarea in Spatiul Aerian al Aerodromului – "Un vehicul aerian autonom care trece prin spatiul aerian al unui aerodrom trebuie sa realizeze toate intoarcerile spre stanga [cu exceptia cazurilor in care exista alte specificatii]."*

Ca prim pas vom verifica daca modelul UAS initial respecta regula noua R_4. Pentru aceasta vom exprima noua regula ca o formula LH si vom adauga fiecarei stari posibile variabila booleana a, care va lua valoarea de adevarata in momentul in care UAV-ul intra in spatiul aerian al aerodromului:

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

Formula spune ca toate tranzitiile din starile in care variabila a este adevarata sa nu duca la starea de intoarcere spre dreapta tr (*TurnRight*), singura stare interzisa in timpul survolarii spatiului aerian al aerodromului. Din momentul in care intoarceri sunt posibile din starile pf si od, vom considera doar subsetul mentionat pentru verificare. Se observa faptul ca formula nu tine pentru modelul initial. Considerand faptul ca variabila de stare a este adevarata in starea od (*ObstacleDetect*), se observa faptul ca singura tranzitie permisa in modelul curent este catre starea tr (*TurnRight*). Astfel, modelul existent nu este in confirmitate cu regula nou introdusa. Mai mult, din starea pf tranzitii sunt posibile catre starea tl (*TurnLeft*), dar si catre starea tr (*TurnRight*). Afirmam faptul ca modelul existent poate fi extins pentru a include noi reguli fara a construi modelul de la inceput. Desi au fost propuse diferite solutii pentru repararea modelului Kripke [1], propunem o solutie bazata pe argumentare pentru extinderea modelului astfel incat sa respecte setul actualizat de reguli.

Ca prim pas in abordarea noastra, reprezentam cateva extensii posibile pentru modelul Kripke ca argumente anulabile, pe care le introducem in DeLP pentru alegerea celei mai bune solutii. Solutia aleasa nu va elimina doar complexitatea altor algoritmi de reparare propusi [1], dar permite sistemului si sa se adapteze la informatii noi mai rapid si mai eficient.

Revenind la exemplul initial, observam faptul ca nu exista nici o posibilitate ca UAV-ul sa ajunga in starea tl din momentul in care a ajuns in starea od, dar doar in starea tr. Din moment ce in interiorul aerodromului, numai intoarceri spre stanga sunt permise, legatura dintre od si tr (r_4) trebuie exclusa din model.

Vom considera un nou argument $\langle \mathcal{A}_6, alter\_course(uav_1, left)\rangle$, ce sugereaza actualizarea regulei R_3 prin permiterea evitarii obstacolelor la stanga, in loc de manevra de evitare la dreapta in timpul navigarii in spatiul aerian al aerodromului:

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

Afirmam faptul ca pentru conformarea la reguli noi, este suficient sa modificam toate legaturile dinspre od si pf spre tl in loc de tr pentru evitarea coliziunilor.

Astfel trebuie realizate urmatoarele operatii de actualizare PU:

1. (PU_2) Eliminarea elementelor de relatie (od, tr) si (pf, tr) astfel incat sa avem: $S' = S$, $L' = L$, si $R' = R \setminus \{(od, tr), (pf, tr)\}$
2. (PU_1) Adaugarea elementului (of, tl) astfel incat sa avem: $S'' = S'$, $L'' = L'$, si $R'' = R' \cup \{(od, tl)\}$

Cu toate acestea, operatia de eliminare ar trebui sa fie necesara doar cand elementul ales pentru a fi exclus cauzeaza un conflict intre doua argumente. In cazul nostru, daca consideram argumentele $\mathcal{A}_2$, xare sustine aplicarea regulei R_2 si $\mathcal{A}_6$, care sustine modificarea regulei R_2 pentru navigarea in spatiul aerian, se poate observa ca acestea nu se ataca, ci ofera solutii pentru contexte diferite: argumentul $\mathcal{A}_2$ se refera la coliziuni posibile in afara spatiului aerian, in timp ce $\mathcal{A}_6$ consider cazurile de evitare a coliziunilor in momentul in care UAV-ul survoleaza spatiul aerodromului. Un rationament similar este aplicat pentru tranzitia (pf, tr), ce va fi posibila doar in momentul in care variabila de stare a nu tine la pf. Prin urmare, pasul PU_2 poate fi exclus, aplicandu-se doar operatia PU_1. Decizia de intoarcere la stanga lsau la dreapta va fi luata in concordanta cu valoarea de adevar a variabilei a, care poate indica prezenta sau absenta unui aerodrom in vecinatatea UAV-ului.

Ilustram operatia de actualizare prin adaugarea unei legaturi r_7 intre starile od si tl. Mai mult, atasam fiecarei stari variabila booleana a, pentru a i se permite UAV-ului sa realizeze doar acele actiuni permise in contexte diferite, mai precis in momentul in care se afla in interiorul sau exteriorul spatiului aerian al aerodromului. Putem observa ca in momentul in care UAV-ul atinge starea od, atunci va decide sa treaca in starea pentru care valoarea de adevar a variabilei a sa coincida cu starea od. Prin urmare, daca UAV uav_1 detecteaza un alt UAV uav_2 care se apropie si este in afara spatiului aerodromului ($\neg a$), va verifica starile urmatoare posibile cu aceeasi valoare de adevar pentru variabila de stare a. Dupa cum se observa din figura 10, starea care este conforma cu conditia este tr. Astfel, daca uav_1 este in starea pf, stare in care variabila a tine,atunci tranzitiile posibile vor fi catre tl sau od.

Daca uav_1 ajunge in starea od in momnetul in care survoleaza spatiul aerian al unui aerodrom, va realiza o tranzitie catre starea tl, in care variabila de stare a tine. Daca uav_1 ajunge in pf va efectua o tranzitie fie catre starile tl sau od. Celellate tranzitii din model nu sunt dependente de variabila de stare a, astfel ca vor ramane neschimbate. Prin adaugarea conditiei $\neg a$ pentru ajungerea in starea tr, putem evita tranzitiile catre starea respectiva in momentul in care variabila booleana a tine pentru model.

Modelul actualizat $\mathcal{M}_1$ este prezentat mai jos:

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

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

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

Prin verificarea regulilor R_1, R_2, R_3 si R_4 pentru modelul $\mathcal{M}\infty$, rezultatele returnate de catre HLMC atesta faptul ca acestea sunt respectate de catre modelul actualizat.

Exemplul ilustrat surprinde un scenariu simplu pentru misiuni cu UAV-uri, dar sustinem faptul ca framework-ul de argumentare prezentat poate fi aplicat si pentru situatii conflictuale mai complexe.

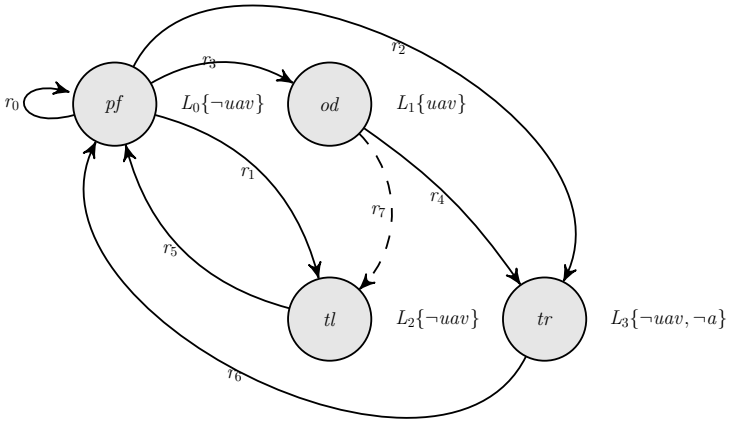


Figura 10: Model Kripke Extins in conformitate cu Regulile Noi.

Bibliografie

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

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

[4] 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.

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

[6] 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.

[7] 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.

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

[9] Ioan Alfred Letia and Anca Goron. Model checking as support for inspecting compliance to rules in flexible processes. *Journal of Visual Languages & Computing*, 2014.

[10] Ioan Alfred Letia and Anca Goron. A temporal view on model checking hybrid logics. In *Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on*, pages 55–58. IEEE, 2014.

[11] Carlos Chesñevar Ioan Letia Anca Goron Mauro Gómez Lucero Sergio Alejandro Gómez, Adrian Groza. Argsafe: 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.