Unmanned Aerial Vehicle compliance checking using Goal-Directed Answer Set Programming

Sarat Chandra Varanasi[1,2], Baoluo Meng[2], Christopher Alexander[2], Szabolcs Borgyos[2] and Brendan Hall[2]

[1]General Electric Research, Niskayuna, NY, USA
[2]The University of Texas at Dallas, Richardson, TX, USA

Abstract

We present a novel application of Goal-Directed Answer Set Programming that digitizes the model aircraft operator's compliance verification against the Academy of Model Aircrafts (AMA) safety code. The AMA safety code regulates how AMA flyers operate Unmanned Aerial Vehicles (UAVs) for limited recreational purposes. Flying drones and their operators are subject to various rules before and after the operation of the aircraft to ensure safe flights. In this paper, we leverage Goal-Directed Answer Set Programming to encode the AMA safety code and automate compliance checks. To check compliance, we use the s(CASP), a goal-directed ASP engine. By using s(CASP) the operators can easily check for violations and obtain a justification tree explaining the cause of the violations in human-readable natural language. We develop a front end questionnaire interface that accepts various conditions and uses s(CASP) as backend engine to evaluate whether the conditions adhere to the regulations. We also leverage s(CASP) implemented in SWI-Prolog, where SWI-Prolog exposes the reasoning capabilities of s(CASP) as a REST service. To the best of our knowledge, this is the first application of ASP in the AMA and Avionics Compliance and Certification space.

Keywords

Goal-Directed Answer Set Programming, Automated Flight Readiness Approval

1. Introduction

We present a novel application of Answer Set Programming (ASP) that hastens the Academy of Model Aircraft (AMA) Flight Safety Compliance checking process. The AMA Safety Code Compliance rules are written as a set of English language rules describing the situations under which violations can occur. We capture the different sets of conditions that could lead to violations and translate the violations into ASP rules. Exceptional conditions in the rules can be directly mapped to ASP's support for negation-by-failure. We have developed an application in the form of a questionnaire detailing various conditions which require a compliance check. We have used the AMA Safety Code rules, which are simple enough to be coded as propositional answer set programs. That is, the entire questionnaire is a series of yes-or-no questions that can be completed in less than five minutes by an experienced aircraft operator applying for

2nd Workshop on Goal-directed Execution of Answer Set Programs (GDE'22), August 1, 2022
sxv153030@utdallas.edu (S. C. Varanasi); baoluo.meng@ge.com (B. Meng); christopher.alexander@ge.com (C. Alexander); szabolcs.borgyos@ge.com (S. Borgyos); hall.brendan@gmail.com (B. Hall)
0000-0002-4620-4266 (S. C. Varanasi); 0000-0002-3284-1969 (B. Meng)

certification. The s(CASP) system is capable of printing a justification tree for any identified violations which is presented in plain English. This helps the user understand the cause of the violation without having to understand ASP rules. To the best of our knowledge, this is the first application of ASP in aviation rules compliance checking.

2. Background

2.1. Goal-Directed Answer Set Programming

Goal-directed approaches to Answer Set Solving do not perform grounding to find stable models. One such approach is Goal-Directed ASP[1]. Goal-directed ASP assumes that a query $?-p(X_1, X_2, ..., X_n)$ is provided to the ASP solver. The goal-directed search finds the predicates that support the submitted query and finds bindings for $X_1, X_2, .., X_n$. At an interactive level, this looks similar to Logic Programming, however, the underlying search uses the co-SLDNF resolution algorithm to find stable models[2, 3]. A salient feature of goal-directed ASP is that, the co-SLDNF algorithm only searches for rules that are relevant to finding the support for the given query. The state-of-the-art implementation of goal-directed ASP is s(CASP). When given a query, s(CASP) returns a *partial stable model* of the query for the given ASP program. Along with the partial stable model, s(CASP) also prints a justification tree to the user conveying how the proof for the given query was performed. The justifications of ASP queries have been used widely in several application areas such as Explainable AI, Legal Reasoning and Natural Language Understanding [4, 5, 6, 7]. Further, the justifications can also be rendered in human understandable natural language [8]. For example, consider the below ASP program with the predicates mapped to their English equivalents. An example justification tree (s(CASP) justification tree) and partial stable model (s(CASP) model) for the program below is shown in Figure 1.

```
flies(X) :- bird(X), not penguin(X).
bird(tweety).
#pred flies(X)::'@(X) flies'.
#pred bird(X)::'@(X) is a bird'.
#pred penguin(X): '@(X) is a penguin'.
```

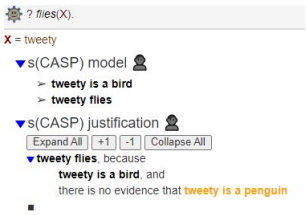

Figure 1: An English justification tree that tweety flies

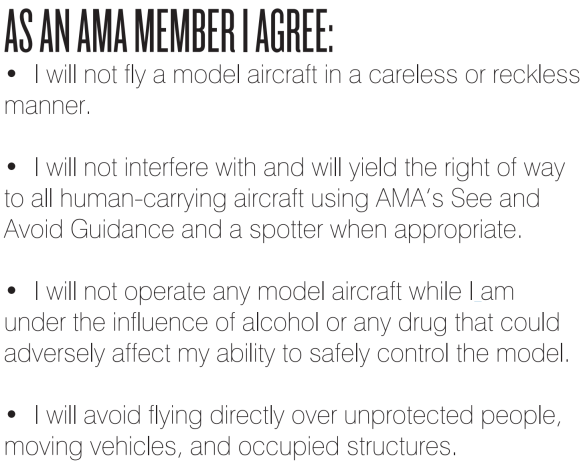

Figure 2: Few rules from AMA General Safety Code[11]

2.2. Modern SWISH Interface and REST API for s(CASP)

The s(CASP) system is available on the modern SWISH Interface for SWI-Prolog [9, 10]. SWISH allows a user to create interactive Logic Program Notebooks for s(CASP) answer set programs. The s(CASP) module can be enabled by using the `:- use_module(library(scasp)).` Along with the online interface, SWISH also provides REST APIs for external clients to run ASP programs. The AMA Safety Code application in this paper uses the SWISH API for s(CASP).

We next explain the Flight Compliance and Certification process used to ensure compliance by recreational UAV operators with AMA rules, followed by their translation in ASP. We also show snippets of the user interface involved in the certification process.

3. The AMA Safety Code for Aircraft Operators

The Academy of Model Aeronautics (AMA) defines a set of rules and regulations to be followed by the operator(s) of model aircraft during their flight. We have adopted the rules defined in the AMA Safety code[11]. The AMA Safety Code has conditions pertaining to general operation of aircrafts along with regulations pertaining to radio control and free flight. The rules are written in plain English. Each rule either prescribes a certain set of conditions desirable for flight safety or proscribes a set of conditions that violate safety requirements. We encode each rule as a violation rule. A few AMA safety code rules are shown in Figure 2.

For example, the AMA rule 3 states that *I will not operate my model aircraft while under the influence of alcohol or while using any drug which could adversely affect my ability to safely control the model.* This potential violation is capture by a rule in ASP as: `violation_3 :- under_alcohol_or_drug_influence.`

Many of the rules are written in terms of *defaults* and *exceptions*. In such case, the rules will be of the form, `some_violation :- default, not exception.`

For instance, first clause of rule 7 (not shown in Figure 2, but found here[11]) states that:

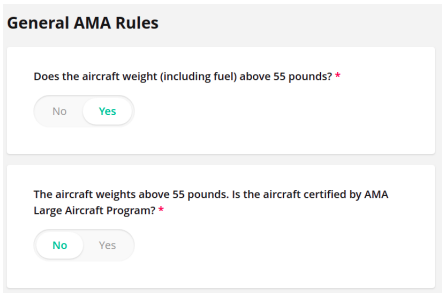

Figure 3: Questionnaire corresponding to aircraft weight

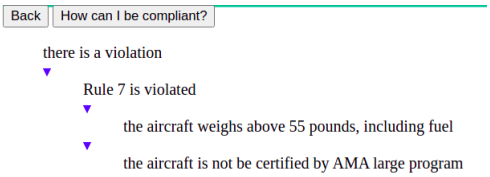

Figure 4: Violation:aircraft weighs above 55 pounds, not certified by AMA Large aircraft program

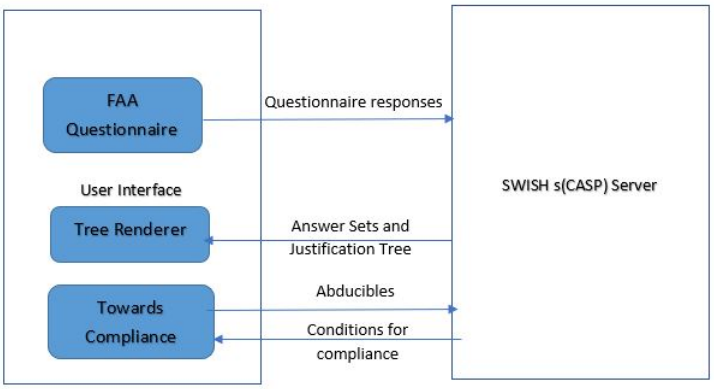

Figure 5: Architecture of the Flight Readiness Approval Application

I will only fly models weighing more than 55 pounds, including fuel, if certified through AMA's Large Model Airplane Program. The sub-clause before *if* represents a default and the sub-clause after *if* represents the exception to the default. This violation is encoded as:
`violation_7 :- aircraft_above_55, not ama_large_program_certified.`

The AMA flight compliance rules are therefore translated into ASP.

The architecture of the web application and its backend interface to SWISH is shown in Figure 5. The app's source itself is accessible from github [12].

On the front-end of the compliance application, the questionnaire elicits answers from the user by asking the following sequence of questions. Each of the questions maps to the literals in the body of a potential violation rule. Shown in figure 3. The proof tree returned by s(CASP) is rendered graphically on the front-end 4. From the viewpoint of the end-user, the ASP translation

and proof tree are completely transparent. Showing the proof graphically helps the operator easily understand the violations and also to figure out how to achieve compliance. The app's source itself is accessible from github [12].

References

[1] J. Arias, et al. (Eds.), Proceedings of the International Conference on Logic Programming 2021 Workshops co-located with the 37th International Conference on Logic Programming (ICLP 2021), Porto, Portugal (virtual), September 20th-21st, 2021, volume 2970 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2970.

[2] R. K. Min, Predicate answer set programming with coinduction, The University of Texas at Dallas, 2009.

[3] J. Arias, M. Carro, E. Salazar, K. Marple, G. Gupta, Constraint answer set programming without grounding, Theory and Practice of Logic Programming 18 (2018) 337–354.

[4] J. Arias, M. Moreno-Rebato, J. A. Rodriguez-García, S. Ossowski, Modeling administrative discretion using goal-directed answer set programming, in: Conference of the Spanish Association for Artificial Intelligence, Springer, 2021, pp. 258–267.

[5] K. Basu, F. Shakerin, G. Gupta, Aqua: Asp-based visual question answering, in: International Symposium on Practical Aspects of Declarative Languages, Springer, 2020, pp. 57–72.

[6] K. Basu, S. C. Varanasi, F. Shakerin, G. Gupta, Square: Semantics-based question answering and reasoning engine, arXiv preprint arXiv:2009.10239 (2020).

[7] K. Basu, S. Varanasi, F. Shakerin, J. Arias, G. Gupta, Knowledge-driven natural language understanding of english text and its applications, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 2021, pp. 12554–12563.

[8] J. Arias, et al., Justifications for goal-directed constraint answer set programming, in: Proceedings 36th International Conference on Logic Programming (Technical Communications), ICLP Technical Communications 2020, (Technical Communications) UNICAL, Rende (CS), Italy, 18-24th September 2020, volume 325 of *EPTCS*, 2020, pp. 59–72. URL: https://doi.org/10.4204/EPTCS.325.12. doi:10.4204/EPTCS.325.12.

[9] J. Wielemaker, F. Riguzzi, R. A. Kowalski, T. Lager, F. Sadri, M. Calejo, Using swish to realize interactive web-based tutorials for logic-based languages, Theory and Practice of Logic Programming 19 (2019) 229–261.

[10] J. Wielemaker, J. Arias, G. Gupta, s(casp) for swi-prolog, in: J. Arias, et al. (Eds.), Proceedings of the International Conference on Logic Programming 2021 Workshops co-located with the 37th International Conference on Logic Programming (ICLP 2021), Porto, Portugal (virtual), September 20th-21st, 2021, volume 2970 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021.

[11] AMA, Ama safety code, https://www.modelaircraft.org/sites/default/files/documents/100.pdf, 2021.

[12] S. C. Varanasi, Flight readiness asp, https://github.com/ge-high-assurance/flight-readiness-asp, 2022.