Reasoning about Airport Security Regulations using the Focal Environment

David Delahaye
CEDRIC/CNAM, Paris, France
David.Delahaye@cnam.fr

Jean-Frédéric Étienne
CEDRIC/CNAM, Paris, France
etien_je@auditeur.cnam.fr

Véronique Viguié Donzeau-Gouge
CEDRIC/CNAM, Paris, France
donzeau@cnam.fr

***Abstract*— We present the validation of regulations intended to ensure airport security in the framework of civil aviation. In particular, we describe the proofs of correctness/completeness for two standards, one at the international level and the other at the European level, and we show how the properties of the European level refines those of the international level. These models are expressed using the Focal environment, an object-oriented specification and proof system, and the proofs described by means of a declarative-like language are processed by the automated theorem prover Zenon. We show how Zenon appears quite appropriate when dealing with abstract specifications like our case study, but also how it should be controlled to present readable proofs.**

I. Introduction

Many human activities are controlled by regulations and standards. These are usually legal expressions of policy choices, which intend to govern the behavior of individuals/processes in the perspective to ensure some safety or security properties, or to simply promote a fair and equitable behavior among various interacting stake-holders. Regulations can be seen as a set of rules/specifications and a key element to guarantee their effective enforcement is to assess the conformity of the procedures and artifacts they intend to regulate. However, the conformity assessment procedures are worthless if the correctness, completeness and consistency of the specifications are not established.

Standards and recommended practices are usually written in natural language in order to be easily understood and adopted by a large number of stake-holders. Nevertheless, the normative documents are generally of voluminous size, ambiguous and often open to (mis-)interpretation. Moreover, it is very difficult to automatically process natural language documents in search for inconsistencies. When a document is several hundred pages long, it is difficult to ensure that the content of a particular paragraph is not contradicted by some others, which may be several dozen pages from the first one. This is even more problematic when analyzing the effects of changes over the entire regulatory system, especially for normative documents maintained by different standardized committees. All these problems highlight the lack of a formal drafting process and this is where modeling techniques can help. Recent work [1] has shown that there is an increased interest in providing automated and systematic support to reason about regulations due to the growing complexity of safety and security requirements.

In this paper, we report on our experience in building and analyzing the formal models of two standards related to airport security: the first one is the international standard Annex 17 [2] produced by the International Civil Aviation Organization (ICAO), an agency of the United Nations; the second one is the European Directive Doc 2320 [3] produced by the European Civil Aviation Conference (ECAC) and which is supposed to refine the first one at the European level. This formalization was completed using the Focal [4] environment, within the framework of the EDEMOI[1] [5] project. The EDEMOI project aims to integrate and apply several requirements engineering and formal methods techniques to analyze regulation standards in the domain of airport security. The novelty of the methodology developed in this project, resides in the application of techniques, usually reserved for safety-critical software, to the domain of regulations (in which no implementation is expected).

This paper is complementary to the work presented in [6], which details the structure of the formal models produced for the two regulations considered above. In [6], we show that in order to provide an appropriate support for the analysis of the regulations, we have to encode the terminologies, the concepts, as well as the integrity constraints characterizing the environment that they intend to regulate. In addition, the formal models produced make a clear distinction between the security requirements and the ways/means to implement them. In fact, from these models, it should be possible to rigorously assess the conformity of a given implementation w.r.t. the security requirements. In this paper, we focus on the validation part of this work. We first highlight the importance of organizing the regulations as a hierarchy of goals down to specific security requirements. We then show how to reason about the said hierarchy to detect anomalies (or to provide evidence of their absence) and to identify hidden assumptions which may lead to shortcomings when additional explanations are required.

Another motivation of this paper is to introduce the Focal [4] (previously Foc) environment, developed by the Focal team, and to show how this tool is appropriate when dealing with this kind of application. The idea is to assess and validate the design features as well as the reasoning

[1]The EDEMOI project is supported by the French National "Action Concertée Incitative Sécurité Informatique".

support mechanism offered by the Focal specification and proof system. In the models of our case study, amongst others, we essentially use the features of inheritance (refinement) and parameterization (modularity). Regarding the reasoning support, the declarative-like proof language appears to be quite appropriate to describe our proofs naturally, whereas the first-order automated theorem-prover of Focal, called Zenon, provides us an effective help by automatically discharging most of the proofs required by the specification.

The paper is organized as follows: first, we give a brief description of the Focal language with its main structures and features; next, we present the analysis of our case study (i.e. the several standards regulating security in airports and in particular, those we chose to model); finally, we describe the validation part of our models made in Focal, that is to say the different proofs ensuring the correctness/completeness of the regulations considered.

II. The Focal environment

In this section, we present very briefly the Focal environment giving, in particular, the syntax as well as the informal semantics for the specifications and the proofs. This will be useful to understand the formalization and especially the validation of the regulations we consider in this paper (see Section IV). To get a more concrete presentation of Focal, the reader can refer to [6], [7], as well as to the standard library where numerous examples can be found (see Section II-D).

A. What is Focal?

Focal [4], initiated by T. Hardin with R. Rioboo and S. Boulmé, is a language in which it is possible to build applications step by step, going from abstract specifications, called species, to concrete implementations, called collections. These different structures are combined using inheritance and parameterization, inspired by object-oriented programming; in addition, each of these structures is equipped with a carrier set, providing a typical algebraic specification flavor. Moreover, in this language, there is a neat separation between the activities of programming and proving. A compiler was developed by V. Prevosto and is able to produce OCaml [8] code for execution, Coq [9] code[2] for certification, as well as code for documentation (generated by means of structured comments). More recently, D. Doligez provided a first-order automated theorem prover, called Zenon, which helps the user to complete his/her proofs in Focal through a declarative-like proof language. This automated theorem prover can produce pure Coq proofs, which are reinserted in the Coq specifications generated by the Focal compiler and fully verified by Coq.

B. Specification: species and collection

The first major notion of the Focal language is the structure of *species*, which corresponds to the highest level of abstraction in a specification. A species can roughly be seen as a list of attributes of three kinds: the carrier type, called *representation*, which is the type of the entities that are manipulated by the functions of the species and which can be either abstract or concrete; the functions, which denote the operations allowed on the entities and which can be either *definitions* (when a body is provided) or *declarations* (when only a type is given); the properties, which must be verified by any further implementation of the species and which can be either simply properties (when only the proposition is given) or theorems (when a proof is also provided).

More concretely, the general syntax of a species is the following:

```
species <name> =

  rep [= <type>];                 (* abstract/concrete
                                     representation *)

  sig <name> in <type>;           (* declaration *)
  let <name> = <body>;            (* definition *)

  property <name> : <prop>;       (* property *)
  theorem <name> : <prop>         (* theorem *)
  proof : <proof>;

end
```

where <name> is simply a given name, <type> a type expression (mainly typing of core-ML without polymorphism but with inductive types), <body> a function body (mainly core-ML with conditional, pattern-matching and recursion), <prop> a (first-order) proposition and <proof> a proof (expressed in a declarative style and given to Zenon). In the type language, the specific expression **self** refers to the type of the representation and may be used everywhere except when defining a concrete representation.

As said previously, species can be combined using (multiple) inheritance, which works as expected. It is possible to define functions that were previously only declared or to prove properties which had no provided proof. It is also possible to redefine functions previously defined or to reprove properties already proved. However, the representation cannot be redefined and functions as well as properties must keep their respective types and propositions all along the inheritance path. Another way of combining species is to use parameterization. Species can be parameterized either by other species or by entities from species. If the parameter is a species, the parameterized species only has access to the interface of this species, i.e. only its abstract representation, its declarations and its properties. These two features complete the previous syntax definition as follows:

```
species <name> (<name> is <name>,
                <name> in <name>, ...)
       inherits <name>, <name> (<pars>), ... = ...
end
```

where <pars> is a list of <name> and denotes the names which are used as parameters. When the parameter is a species, the keyword is **is**, when it is an entity of a species, the keyword is **in**.

[2]Here, Coq is only used as a proof checker, and not to extract, from provided proofs and using its Curry-Howard isomorphism capability, OCaml programs, which are directly generated from Focal specifications.

The other main notion of the Focal language is the structure of *collection*, which corresponds to the implementation of a species. We will not detail this notion here since it is not used in our formalization (see Section IV). Actually, the airport security regulations considered in this paper are rather abstract and are not expected to be implemented.

C. Certification: proving with Zenon

The certification of a Focal specification is ensured by the possibility of proving properties. To do so, a first-order automated theorem prover (based on the tableau method), called Zenon, helps us to complete the proofs. Basically, there are two ways of building proofs with Zenon: the first one is to provide all the properties (proved or not) and definitions needed by Zenon to build a proof automatically; the second one is to introduce additional auxiliary lemmas to help Zenon find a proof. In the first option, Zenon must be strong enough to find a proof with only the provided properties and definitions; the second option must be considered when Zenon requests some assistance or when the user wants to present his/her proof in a more readable form. In the first option, proofs are described as follows:

```
theorem <name> : <prop>
proof : by <props> def <defs>;
```

where <props> is a list of properties and <defs> a list of definitions.

The proof language of the second option is inspired by a proposition by L. Lamport [10], which is based on a practical and hierarchical structuring of proofs using numeric labels for proof depth:

```
theorem <name> : <prop>
proof :
  <<level>><label> assume <hyps> prove <prop>
  <<level>><label> qed [by <props> def <defs>];
```

where <level> is a natural number, <label> a name and <hyps> a list of hypotheses (of the form <name> : <type> or <prop>). The **assume** ... **prove** expression is used to introduce a new goal to be proved (**assume** provides skolemization); the proof for the new goal is detailed in sub-levels, whereby the numeric label is increased accordingly. The **qed** expression closes a proof level (possibly with the help of the following **by** ... **def** provided by the user).

D. Further information

For additional information regarding Focal and, in particular, for examples of specifications, the reader can refer to [6], [7], as well as to the Focal Web site:

`http://focal.inria.fr/`

which contains the Focal distribution (compiler, Zenon and other tools), the reference manual, a tutorial, some FAQs and also some other references regarding, in particular, Focal's formal semantics (e.g. see S. Boulmé and S. Fechter's PhD theses).

III. Regulation analysis

A. Case study

Airport security controls are governed by a series of standards and recommended practices, whose primary concern is the safeguard of civil aviation against acts of unlawful interference. These normative documents describe the general principles, organization and recommended practices that each stake-holder has to adopt and implement to meet security requirements. The entire regulatory system is organized in an hierarchical way, where each level has its own set of normative documents that are drafted and maintained by different bodies. At the international level, Annex 17 [2] of the International Civil Aviation Organization (ICAO) prescribes a set of preventive security measures and practices to be adopted by each member state. It is refined at the European level by the Doc 2320 [3] produced by the European Civil Aviation Conference (ECAC). At the national level, each contracting state has to establish and maintain a national civil aviation security programme in compliance with international standards and national laws. Finally, at the airport level, the national and international standards are implemented by an airport security programme, which specifies the design and infrastructure-related requirements necessary to establish the security measures.

All these documents are written in natural language and due to their rather voluminous size, it is difficult to manually process the entire regulation in search of inconsistencies. In addition, informal definitions may be interpreted in different ways, and thus can have an adverse impact on the outcome of inspection visits to evaluate the conformity of an airport facility against the prevailing standards. However, these documents have the merit of being rigorously structured. Hence, ensuring their correctness, completeness and consistency while eliminating any ambiguity or misunderstanding is a significant step towards the reinforcement of airport security.

B. Security property analysis

When dealing with regulation modeling, it is essential for the formal models to provide a certain structuring that facilitates the traceability and maintainability of the normative documents. In our case, this structuring should even provide support to analyze the impact of a particular security property over the entire regulatory system. To achieve this purpose, we analyzed the dependencies between the security properties w.r.t. the terminologies and concepts used in the normative documents. Basically, this analysis consists in identifying the fundamental security properties and establishing how they are decomposed into sub-properties.

1) Annex 17: We naturally begun our analysis by considering the Annex 17 (international) standard, which is the highest level of the regulation hierarchy. For the EDEMOI project, we particularly focused on the preventive security measures described in Chapter 4, whose objective is translated into the following security property:

4.1 There are no unauthorized dangerous objects[3] *on board an aircraft.*

Property 4.1 is afterwards treated according to six categories of prevention. These are namely measures relating to access control (A17, 4.2), to aircraft (A17, 4.3), to ordinary passengers and their cabin baggage (A17, 4.4), to hold baggage (A17, 4.5), to cargo and mail (A17, 4.6), and to special categories of passengers (A17, 4.7). From these different categories of prevention, we established that to ensure Property 4.1, the following security properties have to be satisfied:

4.2 Persons (other than passengers) accessing aircraft are trustworthy.

4.4 Ordinary passengers do not have access to unauthorized dangerous objects in cabin.

4.5 Hold baggage loaded into aircraft does not contain any unauthorized dangerous objects.

4.7 Armed passengers admitted on board are eligible, while potentially disruptive passengers who are obliged to travel on commercial flights, cannot put at risk the safety on board.

In Annex 17, these security properties characterize the intent of the prescribed security measures. Activities related to cargo or performed during flight time (mainly 4.3 and 4.6) are not considered here as they fall outside the scope of the case study. Using a sequent-like notation, the following decomposition is therefore obtained for Property 4.1:

$$(4.2), (4.4), (4.5), (4.7) \vdash (4.1)$$

Each property (from 4.2 to 4.7) is also decomposed into sub-properties. For example, from the preventive measures related to hold baggage (4.5), we extract the following security properties (we give the corresponding formalization in Section IV):

4.5.1 Originating hold baggage is screened prior to being loaded into an aircraft.

4.5.2 All screened or checked-in hold baggage must be protected from unauthorized interference.

4.5.3 Hold baggage for passengers who are not on board the aircraft must not be transported, unless when identified as unaccompanied and subjected to additional screening.

4.5.4 Transfer hold baggage is screened prior to being loaded into an aircraft, unless it is established that it has been screened to an appropriate level at the point of origin.

4.5.5 Only hold baggage which has been individually identified as accompanied or unaccompanied, and screened to the appropriate standard is authorized for carriage.

For Property 4.5, the decomposition is then the following:

$$(4.5.1), (4.5.2), (4.5.3), (4.5.4), (4.5.5) \vdash (4.5)$$

Here, the notation used (numbering of Annex 17 for the identified security properties) and the decomposition obtained allow us to establish a certain traceability with the normative documents. In addition, it is obvious that by following this decomposition we can easily analyze the impact of a particular security property over the entire regulatory system, which is particularly helpful when investigating the effects of changes from one amendment to another.

Nevertheless, this kind of decomposition does not necessarily imply that the satisfaction of the sub-properties is sufficient to guarantee the satisfaction of the fundamental ones. For instance, there may be some omissions or hidden assumptions (made during the drafting process) in the regulation. Thus, one way of determining the correctness and completeness of the regulation is to prove the derivability of each decomposition (see Section IV-B).

[3]Here, the term dangerous object is used to denote any dangerous device or weapon that is likely to be introduced on board an aircraft.

2) Doc 2320: When extending the analysis to the Doc 2320 (European) standard, we noticed that the structure of the document is mainly organized according to the different categories of prevention described in Chapter 4 of Annex 17. However, the difference between Doc 2320 and Annex 17 resides in the formulation of the security properties: either new properties are added, or the Annex 17 properties are reformulated (but convey the same information), or made more precise (and sometimes more restrictive), or decomposed into further sub-properties, or simply omitted (in which case, we assume that the international standard still prevails).

For example, in Doc 2320, Property 4.5.1 of Annex 17 is decomposed into two security properties described respectively in Paragraph 5.2.1 and 5.2.2:

5.2.1 Accompanied hold baggage is searched by hand or screened with conventional X-ray equipment before being loaded into an aircraft.

5.2.2 Unaccompanied hold baggage is screened with Explosive Device Detection System (EDS) or is searched by hand supplemented with Trace Detection Equipment.

IV. Validation

A. Model structure

In [6], we shown that to properly capture the meaning of the identified security properties, we also have to model the environment that they intend to regulate, together with the integrity constraints that it entails. The hierarchies determined for the subjects being regulated are represented by a Focal model, where each subject is a species. The representation and functions of each species are left undefined (the modeled regulations are quite abstract). For example, the Focal model obtained for baggage is given in Figure 1 (where each node is a species and each arrow is an inheritance relation s.t. $A \leftarrow B$ means species B inherits from A).

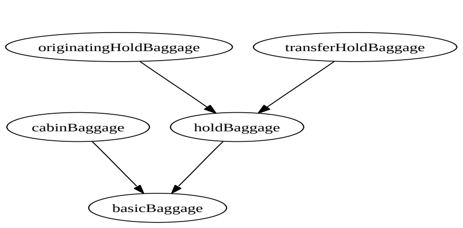

Fig. 1. Hierarchy for baggage.

From Section III-B, it can be observed that each category of prevention targets a specific class of subjects according to their intentions and capabilities. Thus, in order to reflect the decomposition obtained during the regulation analysis phase, we proposed in [6] a formal model where each category of prevention is represented by a species named adequately to ensure traceability. The general model structure obtained for both standards (Annex 17 and Doc 2320) is given in Figure 2, where the Focal specification corresponding to Annex 17 is represented with dashed nodes. The species airsidePersons, ordinaryPassengers, specialPassengers and baggage model the airport environment by introducing the set domain constraints for the identified subjects. They provide the vocabulary necessary for the expression of the security properties. The categories of prevention are afterwards organized hierarchically according to the subjects they regulate and the dependencies between the security properties. For Annex 17, the theorems establishing the validation of the regulation are specified in species annex17. The Doc 2320 specification is obtained by extending the Annex 17 model for each correspondence between the two standards. In addition, refinement proofs are inserted to ensure that the Doc 2320 security properties are not less restrictive than their Annex 17 counterparts. Finally, the validation of the Doc 2320 regulation is established in species doc2320.

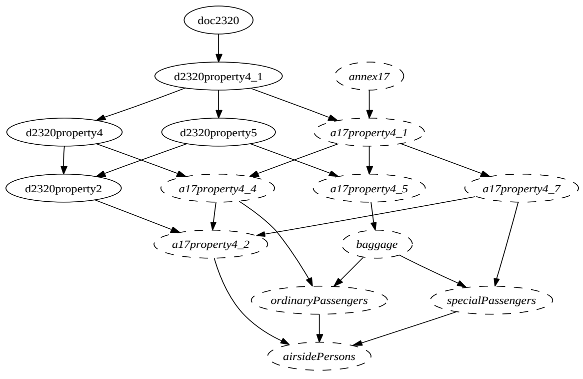

Fig. 2. Structure for Annex 17 and Doc 2320.

B. Correctness and completeness

In this section, we present the different analyses performed on the formal specifications produced in order to establish the correctness and completeness of the regulations considered. In this context, by correctness we mean that the preventive security measures imply the fundamental ones; for example, that Properties 4.2, 4.4, 4.5 and 4.7 imply Property 4.1. By completeness, we mean that the preventive security measures are sufficient to establish the fundamental ones, i.e. no additional assumptions are required to prove the corresponding correctness theorems. In this section, we detail an example showing that the regulations considered are, in a way, not complete. It should also be noted that correctness/completeness theorems do not ensure the absence of contradictions in the regulations (which was not in the objectives of the EDEMOI project), even if we can provide some feasible solutions (see Section IV-C).

1) An example in Annex 17: As an example, we describe the proof of the correctness theorem for Property 4.5, regulating hold baggage. When trying to establish the validity of the decomposition obtained for Property 4.5, we discovered that an assumption has to be made for the proof to be completed. To highlight this hidden assumption, we need to consider how the concerned security properties were formalized:

2) (Property 4.5.1): The security Property 4.5.1, is formalized in species a17property4_5 as follows:

```
property property_4_5_1 : all s in self, all l in ol,
  ol_set!member (l, !originatingHoldLuggage (s)) ->
  ol!loaded (l) -> ol!screened (l);
```

where s represents a particular instance of species a17property4_5 and l an item of originating hold baggage. Here, it can be noticed that this formula follows exactly the text describing Property 4.5.1 in Section III-B. The security property 4.5.4 is formalized in a similar way, except that transfer hold baggage coming from secure destinations may not be subjected to screening before being loaded into an aircraft.

3) (Property 4.5.2): Since the security property 4.5.2 affects both originating and transfer hold baggage, it is broken down into two sub-properties. The following corresponds to the one defined for originating hold baggage:

```
property property_4_5_2a : all s in self, all l in ol,
  ol_set!member (l, !originatingHoldLuggage (s)) ->
    ol!loaded (l) -> ol!secure (l);
```

This property states that if an originating hold baggage is loaded into an aircraft then it is considered to have been protected from unauthorized interference (here represented by the boolean function secure).

4) (Property 4.5.3): The security property 4.5.3 describes the reconciliation process that needs to be performed (when boarding is closed) to ensure that hold baggage is not transported alone, except in special circumstances. The following property corresponds to the formalization obtained for originating hold baggage (a similar property is defined for transfer hold baggage):

```
property property_4_5_3a : all s in self, all a in ac,
  all l in ol, ac_set!member (a, !departureAircraft (s)) ->
  !onboardOriginating (l, a, s) ->
  !boarding_closed (a, s) and
  (!is_unaccompaniedOLuggage (l, a, s) and
  ol!additional_screening (l) or
  !is_accompaniedOLuggage (l, a, s));
```

where a is a departing aircraft and l an item of originating hold baggage. This property describes that originating hold baggage is loaded into an aircraft only when boarding is closed and it is identified as accompanied, or unaccompanied and screened to the appropriate level. The constraint induced by additional screening on unaccompanied hold baggage is specified in species holdbaggage (see Figure 1) as follows:

```
property inv_additional_screening : all hl in self,
  !additional_screening (hl) -> !screened (hl);
```

where hl is an item of hold baggage.

5) (Property 4.5.5): The security property 4.5.5 specifies that hold baggage is authorized for carriage only if there is evidence that it is identified as accompanied or unaccompanied, and screened to the appropriate standard. The following property is obtained for originating hold baggage:

```
property property_4_5_5a : all s in self, all a in ac,
  all l in ol, ac_set!member (a, !departureAircraft (s)) ->
  !onboardOriginating (l, a, s) ->
  !oLuggageMeetsCriteria (l, a, s) ->
  !boarding_closed (a, s) -> ol!authorized_for_carriage (l);
```

where a is a departing aircraft, l an item of originating hold baggage, and where Proposition oLuggageMeetsCriteria encapsulates the verifications mentioned above. A similar property is defined for transfer hold baggage.

6) (Property 4.5): From the above security properties, we can now try to prove the following correctness theorem for Property 4.5:

```
theorem property4_5 : all s in self, all a in ac,
  ac_set!member (a, !departureAircraft (s)) ->
  (all o in do,
     do_set!member (o, !dangerousObjectsInHold (a, s)) ->
     do!is_authorized (o)) and
  (all o in wp,
     wp_set!member (o, !weaponsInHold (a, s)) ->
     wp!is_authorized (o))
```

where s is a particular instance of species a17property4_5, a an aircraft, wp a weapon and do a dangerous object.

Using the declarative-like proof language of Focal (using the underlying automated theorem prover Zenon, see Section II-C), the proof begins by skolemizing and flattening the goal statement as follows:

```
proof:
  <1>1 assume s in self a in ac o in do
            H1: ac_set!member (a, !departureAircraft (s))
            H2: do_set!member (o,
                   !dangerousObjectsInHold (a, s))
         prove do!is_authorized (o) . . .
  <1>2 assume s in self a in ac o in wp
            H1: ac_set!member (a, !departureAircraft (s))
            H2: wp_set!member (o, !weaponsInHold (a, s))
         prove wp!is_authorized (o) . . .
  <1>3 qed;
```

Here, we have to deal with two specific cases. Since the proof is similar for both parts, we only consider the one derived for dangerous objects. In addition, according to the definition of dangerousObjectsInHold, the dangerous object o can either belong to an item of originating or transfer hold baggage. Consequently, we obtain the two following cases:

```
proof:
  <1>1 . . .
    <2>1 assume l in ol
            H3: do_set!member (o,
                   ol!get_dangerousObjects (l))
            H4: ol_set!member (l,
                   !originatingHoldLuggage (s))
            H5: ol!loaded (l)
            H6: ol_set!member (l, olac_set!loadedLuggage
                   (a, !loadedOriginating (s)))
         prove do!is_authorized (o) . . .
    <2>2 . . .
    <2>3 qed by <2>1, <2>2, <1>:H2, do_set!union1,
      ol_set!dangerousObjects1, olac_set!loadedLuggage1,
      olac_set!holdLuggage2, ol_set!subset2, ol_set!equal1,
      tfl_set!dangerousObjects1, tflac_set!loadedLuggage1,
      tfl_set!equal1, tflac_set!holdLuggage2, ol_set!subset2,
      !inv_OLuggageAircraft, !inv_TLuggageAircraft,
      !onboardOLuggage1, !onboardTfLuggage1,
      tfl_set!subset2, def !dangerousObjectsInHold
  <1>2 . . .
  <1>3 qed;
```

Levels <2>1 and <2>2 correspond to the subgoals generated respectively for originating and transfer hold baggage. Level 2<3> specifies how these two subgoals are used to prove Level <1>1 (**by . . . def** clause). Intuitively, the introduction of Hypotheses H3 to H6 in the context of Subgoal <2>1 (similar hypotheses are used for <2>2) results from the fact that if the object o belongs to the set of dangerous objects placed on board an aircraft, then there certainly exists a specific item of originating hold baggage (here identified as l) to which it belongs.

If we now try to prove Subgoal <2>1 w.r.t. the current hypotheses and the security properties defined previously, we notice that the following assumption has to be made for the proof to be completed:

```
property invariant_secure : all hl in self,
 !secure (hl) and !authorized_for_carriage (hl) ->
  (all o in wp,
     wp_set!member (o, !get_weapons (hl)) ->
     wp!is_authorized (o)) and
  (all o in do,
     do_set!member (o, !get_dangerousObjects (hl)) ->
     do!is_authorized (o));
```

This property states that if hold baggage is protected from unauthorized interference and is authorized for carriage then it is considered to only contain dangerous objects/weapons that are authorized. Under this assumption, the proof of Subgoal <2>1 is built as follows:

```
<2>1 assume l in ol
          H3: do_set!member (o, ol!get_dangerousObjects (l))
          H4: ol_set!member (l, !originatingHoldLuggage (s))
          H5: ol!loaded (l)
          H6: ol_set!member (l, olac_set!loadedLuggage
                                 (a, !loadedOriginating (s)))
       prove do!is_authorized (o)
        <3>1 prove
                !oLuggageMeetsCriteria (l, a, s) and
                !boarding_closed (a, s)
              by <2>:H5, <1>:H1, <2>:H4, <2>:H6,
                 !property_4_5_1, !property_4_5_3a
                 def !onboardOriginating,
                 !oLuggageMeetsCriteria
        <3>2 prove
                ol!secure (l) and ol!authorized_for_carriage (l)
              by <2>:H5, <3>1, <1>:H1, <2>:H6,
                 !property_4_5_2a, !property_4_5_5a
                 def !onboardOriginating
        <3>3 qed by <3>2, <2>:H3, ol!invariant_secure
```

As can be seen, Subgoal <2>1 is proved in three steps: at Level <3>1, we show that, by applying H1, H4, H5 and H6 w.r.t. the security properties 4.5.1 and 4.5.3a, we can deduce that the originating hold baggage l is identified as accompanied or unaccompanied, screened to the appropriate level (here represented by oLuggageMeetsCriteria), and that boarding is closed; at Level <3>2, we establish that the originating hold baggage l is considered to be secure and is authorized for carriage, by applying Hypotheses H1, H5 and H6 w.r.t. the security properties 4.5.2a and 4.5.5a, under the assumption of Level <3>1; finally, at Level <3>3, we conclude that the dangerous object o is authorized by applying Hypothesis H3 to Property invariant_secure, under the assumption of Level <3>2. Subgoal <2>2 is proved in a similar way.

By systematically exploring the decomposition obtained for each of the different categories of prevention considered, we managed to identify some other hidden assumptions, which must not be considered as failures of the regulation but more as implicit security requirements. Thus, this validation addresses a certain form of completeness for the regulation, where every security requirement has been made explicit.

7) A refinement of Doc 2320: We consider the example of Section III-B whereby Property 4.5.1 of Annex 17 is intended to be refined by Properties 5.2.1 and 5.2.2 of Doc 2320. This introduces new kinds of theorems (and proofs), that can be called refinement theorems (proofs). These theorems are mainly correctness theorems but are specific in the sense that they ensure a given regulation of a sub-level is not less restrictive than regulations of higher levels. Here, in our example, this means that the Doc 2320 security properties 5.2.1 and 5.2.2 must not invalidate Property 4.5.1 of Annex 17. Hence, we must prove the following theorem:

```
theorem refinement_5_2_1 :
  !d2320_5_2_1 -> !d2320_5_2_2 -> !property_4_5_1
proof : by ..., ol!inv_handSearchXray,
  ol!inv_handSearchTDE_EDS, ol!inv_additionalScreening
  def !d2320_5_1_1, !d2320_5_2_1, !d2320_5_2_2,
  !property_4_5_1, !onboardOriginating;
```

It can be observed that by only providing the appropriate properties and definitions (here, we only provide the main properties and definitions necessary to understand the proof), Zenon manages to discharge the proof obligation automatically. Moreover, as shown below, the proof construction is fairly comprehensible and therefore, does not need to be detailed using the declarative proof language (seen previously with the proof of Property 4.5 of Annex 17). To understand the underlying proof, we need to consider how the concerned security properties are formalized. Properties 5.2.1 and 5.2.2 are respectively formalized as follows:

```
property d2320_5_2_1 : all s in self, all l in ol,
  all a in ac, ac_set!member (a, !departureAircraft (s)) ->
  !onboardOriginating (l, a, s) ->
  !boarding_closed (a, s) ->
  !is_accompaniedOLuggage (l, a, s) ->
  ol!handSearched (l) or ol!xrayed (l);
```

```
property d2320_5_2_2 : all s in self, all l in ol,
  all a in ac, ac_set!member (a, !departureAircraft (s)) ->
  !onboardOriginating (l, a, s) ->
  !boarding_closed (a, s) ->
  !is_unaccompaniedOLuggage (l, a, s) ->
  ol!eds (l) or ol!handSearchedTDE (l);
```

where s represents a particular instance of species d2320property5 (see Figure 2), l an item of originating hold baggage and a an aircraft. Since hand search or conventional X-ray are considered to be traditional screening methods, the following property is introduced in species holdBaggage2320 (which is a refined version of species holdBaggage):

```
property inv_handSearchXray : all hl in self,
  !handSearched (hl) or !xrayed (hl) -> !screened (hl);
```

where hl refers to an item of hold baggage. A similar property (inv_handSearchTDE_EDS) is defined for hold baggage subjected to EDS or hand search with trace detection equipment. However, these are considered to be additional screening techniques (see Property 4.5.3 of Annex 17). Finally, the correlation between Property 4.5.1 and Properties 5.2.1/5.2.2 is established through the use of the following property:

```
property d2320_5_1_1 : all s in self, all a in ac,
  all l in ol, ac_set!member (a, !departureAircraft (s)) ->
  !onboardOriginating (l, a, s) ->
  !boarding_closed (a, s) and
  (!is_unaccompaniedOLuggage (l, a, s) or
  !is_accompaniedOLuggage (l, a, s));
```

It simply states that originating hold baggage is placed on board an aircraft only if it is identified as being accompanied or unaccompanied.

C. Consistency

As said in Section IV-B, the correctness/completeness theorems we identified and proved do not guarantee the absence of contradictions in the regulation. One way to tackle this problem is to try to derive False from the set of security properties and to let Zenon work on it for a while. If the proof succeeds then we have a contradiction, otherwise we can only have a certain level of confidence. This approach may seem rather naive but appears quite pertinent when used to identify the correlation between the several security measures according to specific attack scenarios. The idea is to falsify an existing hypothesis or to add an inconsistent hypothesis and to study its impact over the entire regulation, i.e. where the potential conflicts are located and which security properties are concerned. For example, let us suppose we add the following property to our formalization:

```
property secure_not_screen : all s in self, all l in ol,
  ol_set!member (l, !originatingHoldLuggage (s)) ->
  ol!secure (l) -> not (ol!screened (l));
```

Even if it is not in direct contradiction with any of the properties defined previously, it still introduces an inconsistency in the specification for a given context (see below). Now, let us consider the following facts to model a particular instance of our scenario:

```
property fact1 : all s in self,
  ol_set!member (!my_baggage, !originatingHoldLuggage (s));

property fact2 : ol!loaded (!my_baggage);
```

In this context, Zenon is able to identify the anomaly by successfully proving False (using Properties 4.5.1 and 4.5.2a). In addition, whenever such proof succeeds, Zenon helps the user to locate the source of the inconsistency (the conflicting security properties) by providing the list of properties (and definitions) not used to complete the proof (since the user can provide more properties than necessary). This approach might be coupled with complementary methods, such as deviational techniques [11], in order to cater for external factors (which can still influence the regulation content) using flaw hypotheses to explore security violations.

D. Development

In the validation of this formalization (10,000 lines of Focal code, 150 species, 200 proofs, 2 years to be completed), Zenon allowed us to discharge most of the proof obligations automatically (about 90% of them). Actually, Zenon also succeeded in completing the remaining 10% automatically but beyond the default timeout (set to 3 min in Focal). This tends to show that Zenon is quite appropriate when dealing with abstract specifications (no representation and very few definitions). The development is freely available (sending a mail to the authors) and can be compiled with the latest version of Focal (0.3.1).

V. Conclusion

1) Summary: One way to improve security is to produce high quality standards. The formal models of Annex 17 and Doc 2320 regulations, as well as their validation partially described in this paper, tend to bring an effective solution in the specific framework of airport security. Regarding the validation part, Zenon, the automated theorem prover of the Focal environment, appeared quite appropriate discharging most of our proofs automatically. In particular, it allowed us to complete the correctness/completeness proofs, as well as the refinement proofs resulting from the addition of Doc 2320. It also helped us to identify hidden assumptions, which seemed to be implicit in the corresponding regulation documents. In addition, this prover could be used in each step of the development: in the prototyping phase, providing a set of properties and definitions (to be used by Zenon) to be convinced that a given lemma is correctly formulated; in the finalizing phase, providing more detailed proofs to obtain more readable specifications together with a reasonable compilation time.

2) Related work: Currently, models of the same regulations, by D. Bert and his team, are under development using B [12] in the framework of the EDEMOI project. In the near future, it could be interesting to compare the two formal models (in Focal and B) rigorously in order to understand if and how the specification language but also the reasoning support influence the model itself. Very close to the EDEMOI project is the SAFEE project [13], which aims to use similar techniques for security on board aircraft and during flight time. Regarding similar specifications in Focal, certified implementations of access control models [14] are under development by M. Jaume and C. Morisset.

3) Future work: As said in Section IV-B, we plan to use Zenon as a tool to build attack scenarios which, at least in this context, appear to be quite interesting for official certification authorities. In particular, this will allow us to study the impact of a given security property (or sub-property) over the entire regulation. In the same way, we would like to integrate a test suite into this formalization using an automatic generation procedure (working from a Focal specification) and using stubs for abstract functions (i.e. only declared). Such a procedure is currently work in progress, by C. Dubois and M. Carlier, but is still limited (to universally quantified propositions) and needs to be extended to be applied to our development. We also plan to produce UML diagrams automatically generated from the Focal specifications and which is an effective solution to interact with competent organizations (ICAO, ECAC). Such a tool has been developed by J. F. Étienne but has to be completed to deal with all the features of Focal.

References

[1] R. Laleau and M. Lemoine, Eds., *International Workshop on Regulations Modelling and their Validation and Verifi cation (REMO2V), in conjunction with Conference on Advanced Information Systems Engineering (CAiSE).* Luxembourg (Grand-Duchy of Luxembourg): Presses Universitaires de Namur, June 2006.

[2] T. I. C. A. Organization, *Annex 17 to the Convention on International Civil Aviation, Security - Safeguarding International Civil Aviation against Acts of Unlawful Interference, Amendment 11*, Nov. 2005.

[3] T. E. C. A. Conference, *Regulation (EC) N° 2320/2002 of the European Parliament and of the Council of 16 December 2002 establishing Common Rules in the Field of Civil Aviation Security*, Dec. 2002.

[4] T. Focal. D. Team, Focal, *version 0.3.1*, CNAM/INRIA/LIP6, May 2005, available at: `http://focal.inria.fr/`.

[5] T. EDEMOI. Project, 2003, `http://www-lsr.imag.fr/EDEMOI/`.

[6] D. Delahaye, J.-F. Étienne, and V. Viguié Donzeau-Gouge, "Certifying Airport Security Regulations using the Focal Environment," in *Formal Methods (FM)*, ser. Lecture Notes in Computer Science (LNCS), vol. 4085. Hamilton, Ontario (Canada): Springer, Aug. 2006, pp. 48–63.

[7] C. Dubois, T. Hardin, and V. Viguié Donzeau-Gouge, "Building Certifi ed Components within Focal," in *Symposium on Trends in Functional Programming (TFP)*, vol. 5. Munich (Germany): Intellect (Bristol, UK), Nov. 2004, pp. 33–48.

[8] T. Cristal. Team, Objective Caml, *version 3.09.1*, INRIA, Jan. 2006, Available at: `http://caml.inria.fr/`.

[9] T. Coq. D. Team, Coq, *version 8.0*, INRIA, Jan. 2006, Available at: `http://coq.inria.fr/`.

[10] L. Lamport, "How to Write a Proof," *American Mathematical Monthly*, vol. 102, no. 7, pp. 600–608, Aug. 1995.

[11] T. Srivatanakul, J. A. Clark, and F. Polack, "Effective Security Requirements Analysis: HAZOP and Use Cases," in *Information Security : 7th International Conference*, ser. Lecture Notes in Computer Science (LNCS), vol. 3225. Palo Alto (CA, USA): Springer, Jan. 2004, pp. 416–427.

[12] J. R. Abrial, *The B Book, Assigning Programs to Meanings*. Cambridge (UK): Cambridge University Press, 1996, iSBN 0521496195.

[13] T. SAFEE. Project, 2004, `http://www.safee.reading.ac.uk/`.

[14] M. Jaume and C. Morisset, "Formalisation and Implementation of Access Control Models," in *Information Assurance and Security (IAS), International Conference on Information Technology (ITCC)*. Las Vegas (USA): IEEE CS Press, Apr. 2005, pp. 703–708.