Aligning, Interoperating, and Co-executing Air Traffic Control Rules Across PSOA RuleML and IDP

Marjolein Deryck[1], Theodoros Mitsikas[2], Sofia Almpani[2], Petros Stefaneas[2], Panayiotis Frangos[2], Iakovos Ouranos[3], Harold Boley[4], and Joost Vennekens[1]

[1] KU Leuven, Belgium
{marjolein.deryck; joost.vennekens}[AT]kuleuven.be
[2] National Technical University of Athens, Greece,
mitsikas[AT]central[DOT]ntua[DOT]gr,
salmpani[AT]mail[DOT]ntua[DOT]gr,
petros[AT]math[DOT]ntua[DOT]gr,
pfrangos[AT]central[DOT]ntua[DOT]gr
[3] Hellenic Civil Aviation Authority, Greece,
iouranos[AT]central[DOT]ntua[DOT]gr
[4] University of New Brunswick, Canada
harold[DOT]boley[AT]unb[DOT]ca

Abstract. This paper studies Knowledge Bases (KBs) in PSOA RuleML and IDP, aligning, interoperating, and co-executing them for a use case of Air Traffic Control (ATC) regulations. We focus on the common core of facts and rules in both languages, explaining basic language features. The used knowledge sources are regulations that are specified in (legal) English, and an aircraft data schema. In the modeling process, inconsistencies in both sources were discovered. We present the discovery process utilizing both specification languages, and highlight their unique features. We introduce three extensions to this ATC KB core: 1) While the current PSOA RuleML does not distinguish the ontology separately from the instance level, IDP does. Hence we specify a vocabulary-enriched version of ATC KB in IDP for knowledge validation. 2) While the current IDP uses relational modeling, PSOA also supports graph modeling. Hence we specify a relationally interoperable graph version of ATC KB in PSOA. 3) The KB is extended to include optimization criteria. With this, the determination of an optimal sequence of more than two aircraft is possible.

Keywords: PSOA RuleML · IDP · Interoperation · Knowledge Base · Alignment · Co-execution · Regulations · Air Traffic Control.

1 Introduction

Contributing to cross-fertilizations between, e.g., the Semantic Technologies and Decision Management Communities,[5] in this paper we use the PSOA RuleML

[5] For specific references see http://blog.ruleml.org/post/132677817-decisioncamp-and-ruleml-rr-will-meet-again-in-luxembourg.

version of the ATC KB [13][6] as a starting point for an IDP version, and explore the consequences for both. We compare Knowledge Bases (KBs) in IDP and PSOA RuleML to find how modeling the same knowledge in two languages can help us to improve our specification and to achieve an architecture that combines the best of both systems. Based on regulations and data obtained from [5], the PSOA specification was created. From it, the IDP KB was derived and similarities and differences between both specifications were studied. In the first step, we try to align both systems by choosing from different possibilities, similar ways of modeling. In the process, we discovered that there were not only inconsistencies in the source aircraft characteristics data as discussed in [13], but also in one of the regulations. In the second step we investigate how both systems can be interoperated by translating pieces of knowledge from one source to the other. In the third step the co-execution of both systems is examined. The co-execution allows us to the validate results from both systems. The resulting KB were then expanded: an optimization logic was formalized in IDP, while a perspectival graph version of the KB was created in PSOA. As the systems can be co-executed, an architecture in which the strength of each system is exploited can be envisaged.

Examples of ATC regulations formalization are [12] and [14]. The former presented an overview of a method for formal requirements capture and validation, in the domain of oceanic ATC. The obtained model focused on conflict prediction, while being compliant to the regulations governing aircraft separation in oceanic airspace. The design approach, the specification structure, and some examples of the rules and axioms of the formal specification were provided. Those examples, expressed in many-sorted first-order logic or in the Prolog notation, included rules about conflict prediction and aircraft separation. Supplementary, the model was validated by automated processes, formal reasoning and domain experts. [14] focuses on capturing ATC regulations valid in the airport area. The authors formalized the separation minima mandated by ICAO, FAA, and RECAT regulations in RuleML/POSL form. It formed the foundations for further expansion that focuses on cases of conditional reduced separation minima. It was also the basis for the development of [13], a PSOA RuleML version of ATC KB that in turn, served as the basis for this paper.

The paper is structured as follows. In the next section we introduce IDP and PSOA. Then the use case of Air Traffic Control regulations is introduced in Section 3. The aligned KBs are presented in Section 4, while Section 5 discusses the interoperation and co-execution of the two systems and compares their results. Section 6 discusses inconsistencies found within the regulations, which is followed by the presentation of KBs' extensions in Section 7. Section 8 provides some final conclusions and directions for future work.

2 Knowledge Formalization and Reasoning

In this section we introduce the two specification languages, IDP and PSOA RuleML.

[6] See the PSOA ATC KB sources at `http://users.ntua.gr/mitsikas/ATC_KB/`.

2.1 IDP and the Knowledge-based Paradigm

IDP refers both to a first-order-based language and to the knowledge-based system in which it is used [3]. Knowledge-based systems are systems that adhere to the knowledge-based paradigm (KBP). The governing principle of the KBP is the distinction between domain knowledge *an sich*, and the way this knowledge is used in different use cases [17]. The domain knowledge is formalised and centralised in the knowledge base (KB). Under this paradigm, a KB collects not only simple knowledge (e.g., data in a database), but also complex knowledge that is expressed with definitions, implications, propositions, etc. The aim of the KB is to allow reasoning with this complex knowledge, and support a multitude of inferences that can solve a range of problems. If the KB is complete with regards to the problem domain, this can be done by only selecting or developing the appropriate inference task without changing the KB for a specific case. The advantages of this separation of concerns of knowledge versus problem solving, are the high maintainability of the knowledge base because is only contains descriptive information on the domain; and the flexibility to use this KB in different – often unforeseen – use cases.

The IDP language that is used to create a KB is based on classical first-order logic, but it is enriched with the possibilities to use aggregates, sorts and inductive definitions. The core IDP program typically consists of three parts. The *vocabulary* part contains the concepts that will be used to formulate the domain knowledge. It represents the ontology of the domain. The second part is the *theory* part. It explains how the concepts of a certain vocabulary are linked together using predicate logic. Both rules and constraints can be expressed in the theory. Third, the *structure* over a vocabulary assigns values to some elements of the vocabulary. This includes both the delineation of the domain and the assignment of specific values. Hence, it provides a partial interpretation of the model. Together, these tree parts form the IDP KB, and represent a stand alone piece of knowledge on the problem domain.

To solve specific problems, multiple inference algorithms can be used. Some of the most common algorithms include $Model\ Expansion$, $Propagation$, and $Optimization$.
With a vocabulary V, and theory T and structure S over V, the inference of $Model\ Expansion$ looks for a model. A model gives a total interpretation for (i.e.; assigns a value to all) the elements of V, such that T and S are satisfied.
$Propagation$ is the derival of the value of some unassigned element of V based on the partial interpretation in S and taking into account T. Hence, after propagation, the result is a partial model (i.e.; a model that assigns values to some, but not all elements of V) that satisfies T and S, and is more specific than the original interpretation given in S.
In many use cases, the inference of $Optimization$ is very useful. Whereas model expansion calculates any total model that satisfies the given constraints, Optimization calculates the best model that satisfies the constraints. This is done by the creation of a term, that expresses which criterion should be optimised.

2.2 PSOA RuleML

PSOA RuleML generalizes RIF-BLD and POSL RuleML by a homogeneous integration of table-like relationships and graph-like frames into ***p**ositional-**s**lotted **o**bject-**a**pplicative* (***psoa***) *terms*. The often used *single-tuple independent-slot special case of psoa terms*, oidless or oidful, has these forms [19] (where $n \geq 0$ and $k \geq 0$, of which in ATC KB we only require either $n = 0$ for – oidless – frameships and – oidful – framepoints or $k = 0$ for – oidless – relationships):

$$\textbf{Oidless}: \quad \mathtt{f(t_1 \ldots t_n\ p_1{->}v_1 \ldots p_k{->}v_k)} \tag{1}$$

$$\textbf{Oidful}: \mathtt{o\#f(t_1 \ldots t_n\ p_1{->}v_1 \ldots p_k{->}v_k)} \tag{2}$$

Both (1) and (2) apply a function or predicate `f` (acting as a relator) – in (2) identified by an OID `o` via a membership, `o # f`, of `o` in `f` (acting as a class) – to a tuple of arguments $\mathtt{t_1} \ldots \mathtt{t_n}$ and to a bag of slots $\mathtt{p_j{->}v_j}$, $\mathtt{j} = 1, \ldots, \mathtt{k}$, each pairing a slot name (attribute) $\mathtt{p_j}$ with a slot filler (value) $\mathtt{v_j}$.

Variables in PSOA are '`?`'-prefixed names, e.g., `?x`. The most common atomic formulas are psoa atoms in the form of (1) or (2). Compound formulas can be constructed using the Horn-like subset of first-order logic. A PSOA KB consists of clauses, mostly as ground facts and non-ground rules: While facts are psoa atoms, rules are defined – within `Forall` wrappers – using a Prolog-like *conclusion* `:-` *condition* syntax, where *conclusion* can be a psoa atom and *condition* can be a psoa atom or a prefixed conjunction of psoa atoms.

The reference implementation for deduction in PSOA RuleML is the open-source framework system PSOATransRun.

3 Air Traffic Control Regulations

Collision prevention in Air Traffic Control (ATC) is realized by ensuring a minimum distance between aircraft, a concept also called *separation minimum*. Separation of aircraft serves an additional important role, which is the avoidance of wake turbulence. The separation minimum is defined for a pair of aircraft depending on their wake turbulence category. The current FAA[7] and ICAO[8] regulations categorize aircraft according to the maximum takeoff weight/mass (MTOW/MTOM). MTOW/MTOM represents both the generated wake turbulence from the leading aircraft, as well as how much a following aircraft is affected by the wake turbulence of the leader. Both agencies are in the process of applying a wake turbulence recategorization (RECAT), which recategorize aircraft in six categories, taking into account the wingspan as an additional parameter.

For example, ICAO categories are defined, in (legal) English, as follows [10,11]:

Light – MTOM of 7000 kg or less.
Medium – MTOM of greater than 7000 kg, but less than 136000 kg.

[7] FAA: Federal Aviation Administration. The United States of America national authority that regulates all aspects of civil aviation.
[8] ICAO: International Civil Aviation Organization. A UN specialized agency, for civil aviation.

Heavy – MTOM of 136000 kg or greater.
Super – A separate designation that currently only refers to the Airbus A380 (MTOM 575000 kg, ICAO designation `A388`).

The associated separation minima for flights under instrument flight rules (IFR)[9] are defined in Table 1[10].

Table 1. Current ICAO weight categories and associated separation minima [11]

ICAO separation standards (nautical miles (NM))					
		Follower			
		Super	Heavy	Medium	Light
Leader	Super	MRS	6	7	8
	Heavy	MRS	4	5	6
	Medium	MRS	MRS	MRS	5
	Light	MRS	MRS	MRS	MRS

MRS is the Minimum Radar Separation, which is 3 NM or 2.5 NM depending on operational conditions unrelated to wake turbulence (e.g. visibility, surface) [11].

4 Alignment

KBs represent the knowledge from a knowledge domain. This means that KB languages should be able to represent objects, facts, and relations in this domain. If two KBs are developed for the same knowledge domain, it is expected that they express the same information. Hence, it should be possible to align both. In this section we will discuss the way certain parts of knowledge are represented in both languages.

4.1 Common Core of the KBs

We performed an alignment for all PSOA and IDP constructs used in the ATC KB. Typical parts of this PSOA-IDP alignment are shown below (aircraft-characterizing facts were obtained from [5]):

[9] Separation minima for arrivals and departures for flights on visual flight rules (VFR) are time-based. Additionally, this time-based separation can be applied between arriving IFR flights under certain conditions [7,10].

[10] The minima set out at Table 1 shall be applied when: a) an aircraft is operating directly behind another aircraft at the same altitude or less than 300 m (1 000 ft) below; or b) both aircraft are using the same runway, or parallel runways separated by less than 760 m (2 500 ft); or c) an aircraft is crossing behind another aircraft, at the same altitude or less than 300 m (1 000 ft) below [10].

```
% PSOA KB fragment
Forall ?a ?w (
  :AircraftIcaoCategory(?a icao:Light) :-
    And(?a#:Aircraft(:mtom->?w)
        math:lessEq(?w 7000))
)

Forall ?a ?w (
  :AircraftIcaoCategory(?a icao:Medium) :-
    And(?a#:Aircraft(:mtom->?w)
        math:greaterThan(?w 7000)
        math:lessThan(?w 136000))
)

Forall ?a ?w (
  :AircraftIcaoCategory(?a icao:Heavy) :-
    And(?a#:Aircraft(:mtom->?w)
        math:greaterEq(?w 136000)
        Naf(:AircraftIcaoCategory(?a icao:Super))
    )
)

:AircraftIcaoCategory(a388 icao:Super).
:AircraftIcaoCategory(a38f icao:Super).

%% ICAO Separation example

Forall ?l ?f (
    :icaoSeparation(:leader->?l
                    :follower->?f
                    :miles->8) :-
    And(:AircraftIcaoCategory(?l icao:Super)
        :AircraftIcaoCategory(?f icao:Light))
    )

%% Sample Aircraft Facts %%

be91#:Aircraft( :mtom->4218.41
                :mtow->9300.0
                :wingspan->45.92
                :appSpeed->100.0)

a388#:Aircraft( :mtom->575000.0
                :mtow->1267658.0
                :wingspan->261.65
                :appSpeed->145.0)
```

```
// IDP KB fragment
vocabulary V {
    type Mtom isa int
    type Aircraft isa string
    MTOM(Aircraft,Mtom)
        ... }
theory T:V{
    !a[Aircraft] w[Mtom]:
    AircraftIcaoCategory(a, Light) <=
        MTOM(a,w)
            & w =< 7000.

!a[Aircraft] w[Mtom]:
AircraftIcaoCategory(a, Medium) <=
    MTOM(a,w)
            & 7000 < w
            & w < 136000.

!a[Aircraft] w[Mtom]:
AircraftIcaoCategory(a, Heavy) <=
    MTOM(a,w)
            & 136000 =< w
            & a ~= a388
            & a ~= a38f.

AircraftIcaoCategory("a388", Super).
AircraftIcaoCategory("a38f", Super).

// ICAO Separation Example

!l[Leader],f[Follower]:

IcaoSeparation(l, f) = 8 <=

    AircraftIcaoCategory(l, Super)
    & AircraftIcaoCategory(f, Light).
}

structure S1 : V {
    //specific value assignments:
    Leader = {a388}
    Follower = {be91}
    ...

    //aircraft data
    MTOM = {be91, 4218; a388, 575000}
    MTOW = {be91, 9300; a388, 1267658}
    WingSpan = {be91, 45; a388, 261}
    AppSpeed = {be91, 100; a388, 145}
}
```

Vocabulary In IDP the types/sorts that will be used in the knowledge base, need to be explicitly declared in the *vocabulary*. It binds the use of types in relations that are appropriate for it. When instances of a type are (correctly) used in IDP, the types can be derived by the system, based on the place in which they occur. In PSOA there is no separate signature declaration.

Using the KB These two specification fragments are equivalent, in the sense that they both have the same class of possible worlds: an interpretation (sometimes

also called a structure) W satifsies the PSOA fragment if and only if it satisfies the IDP fragment.

As written above, both PSOA and IDP represent the categorization of aircraft as a set of implications: the `:-` symbol of PSOA and the <= symbol of IDP both denote the material implication of classical first-order logic. In other words, an interpretation W is a model of an implication `F :- G` in PSOA, or of `F <= G` in IDP, if and only if G holds in W or F is false in W. Accordingly, the class of models of the above IDP / PSOA specification is quite large, since every superset of a model is again a model; e.g., there are models in which the same aircraft belongs to all four categories at the same time. However, this is not a problem for PSOA, since this system uses the knowledge-base by means of the inference task of *query answering*, which looks for properties that hold in *all* models of the specification. If we ask a query such as `:AircraftIcaoCategory(be91 ?c)`, we will only get `?c = Light` as an answer: `be91` has this category in all models and while there are also models in which, e.g., `be91` has category `Heavy` in addition to `Light`, there is also a model in which `be91` does not have the category `Heavy` and consequently `?c = Heavy` is not a valid query answer.

In IDP, the inference task is called *deduction* and it produces identical results. However, this is not an idiomatic use of IDP. The knowledge representation philosophy behind IDP is that each KB should be written in such a way that its models correspond one-to-one with possible worlds in reality. The above specification obviously does not have this property and is therefore not recommended usage of IDP. In particular, the above specification works *only* when the inference task of deduction is applied to it. When applying another inference task such as *model expansion* or *optimization*, we may obtain erroneous results in which an aircraft is assigned some category even though it does not satisfy the condition for belonging to this category (such as `be91` belonging to the category `Heavy`).

The more idiomatic way of representing the above knowledge in IDP is by replacing each of the material implications <= by IDP's *definitional implication* symbol <-. Such a definitional implication entails the material implication (i.e., if `F <- G` then also `F <= G`), but in addition, it also implies that F can *only* be true if there is at least one rule of the form `F <- G` such that `G` is true. For sets of rules that have no recursion over negation, this means that attention is restricted to the *least* model of the implications. When recursion over negation is present, the *well-founded model* of the rule set is used. In other words, the definitional IDP specification of the aircraft categorization will only have a single model, in which each aircraft is assigned only that category for which it satisfies the conditions of the rule. The definitonal IDP specification is therefore no longer equivalent to the PSOA specifiation. However, when we apply either model expansion (compute a single model) or deduction (compute facts that are true in all models—but there is only one model in this case) to the definitional IDP specification, we still obtain precisely the same answers as when we query the PSOA specification.

Expressing relations In the aligned KBs above, the purpose is to establish the relation between an aircraft and the ICAO regulation. In Section 6 we discuss

the modeling choice that was made earlier to use aircraft type as an identifier, and the challenges put forward by this. In this part we assume that an aircraft can only be assigned to one category, as this is the case for every specific aircraft.

In both modeling languages it is possible to employ relations in different ways, of which we chose a compatible subset for our KBs:
PSOA allows very general atoms [2], but here uses the *single-tuple independent-slot* special case of psoa terms (Section 2.2). Specializing further, we need atoms that are oidful-slotted (*framepoints*) for the KB facts, oidless-tupled (*relationships*) for the aircraft categorization, and oidless-slotted (*frameships*) for the separation. A perspectival version is discussed in Section 7.2.
IDP allows relations (which can be true or false) and functions (that have exactly one image). A 0-ary relation is a Boolean, a 0-ary function is a constant. Both also have unary and n-ary variants. A function is a special relation, in the sense that a function `f(x)=y` could also be written as a relation `r(x,y)`, with the additional constraint that each argument `x` needs to have exactly one image `y`. As an aircraft can only belong to one category, the use of function represents the actual domain knowledge in the most appropriate way: `AircraftIcaoCategory(Aircraft):Category`. Alternatively, a relation can also be used, which is closer to the modeling in PSOA. The relation is expressed as: `AircraftIcaoCategory(Aircraft, Category)`. A separate constraint can be formulated to express that an aircraft may belong to only one category: $\forall a[Aircraft] : \#c : AircraftIcaoCategory(a, c) = 1$.

Exceptions to the regulations The specification identifies two specific types of aircraft—`a388` and `a38f`—as belonging to the category *super*, even though their weight would normally put them in the category *heavy*. In other words, these two aircraft can be seen as exceptions to the general rule that aircraft with a weight over 136000 are "heavy". In the above IDP specification, we have represented these exceptions by excluding these two aircraft from the rule for "heavy" *by name*. Obviously, this is a poor representation, because it requires us to update both the rule for "heavy" and the rule for "super" if more aircraft are added to the "super" category[11]. In PSOA, we have an appealing alternative in the use of *negation as failure (naf)*: we can write `Naf(:AircraftIcaoCategory(?a icao:Super))` in the body of the rule for "heavy". This atom will hold for any aircraft for which it *cannot be proven* that it belongs to *super* (which will be precisely all those aircraft that are not enumerated as being "super").

IDP does not have negation as failure and we therefore cannot adopt the same representation as long as we are using material implication. However, as discussed before, the idiomatic IDP representation would be to use definitional implications instead. Under this representation, we can simply write `~AircraftIcaoCategory(a, Super)` in the body of the rule for "heavy". The `~` symbol represent simply classical negation, meaning that in any model in

[11] In the ATC domain the regulations are stable and new types of aircraft e.g. in the **Super** category are not currently in active development. Therefore, we do not consider this a major problem.

which *a* is not "super", it will be "heavy". Because we make use of definitional implications, there is only one model, in which `a388` and `a38f` are the only two aircraft that are "super", and therefore this represention is correct. We therefore see that the combination of material implication with negation as failure in PSOA is functionally identical (though not formally equivalent, since the former has many more models than the latter) to the combination of definitional implementation with classical negation in IDP.

Comparing to [13], the newest PSOA version presented here does not need the workaround of the extra slot `SpecialCase`, as PSOATransRun now supports negation as failure.

5 Interoperation and Co-execution

Many of the commonalities and differences between the PSOA and IDP have been discussed in the previous section. In this section, we examine how both systems interoperate and co-execute.

5.1 Syntactic Translation for Interoperation

PSOA is a rule-based system, whereas IDP is a generic constraint-based system (with rules as a special form of constraints). Their interoperation is only applicable to facts and rules. Also because of this, IDP features different reasoning tasks. This means that it might be more advantageous to go beyond the literal translation of the PSOA KB, to find the appropriate notation useful for all inferences.

Proceeding from the aligned KBs from Section 4.1, a partial translation can be realized:

Atoms: For *relationships*, there is a direct PSOA-IDP tuple correspondence. For *framepoints*, a slot name (e.g., `:mtom`) in PSOA is reflected by a *binary relation* (e.g., `MTOM`) in IDP, with the OID as the first argument and the filler as the second argument (the predicate name is already part of IDP's vocabulary declaration). For *frameships*, n-1 slots in PSOA can map to the argument tuple of a function in IDP, and 1 slot to its returned value.

Symbols: Some symbols can be directly translated from one language to the other, e.g. quantifiers (Forall $\leftrightarrow \forall$, Exists $\leftrightarrow \exists$) and implication (:- $\leftrightarrow \Leftarrow$).

Operators: PSOA uses prefix operators, while IDP uses infix operators (And vs. &, Or vs. |). *Externals* and *libraries* in PSOA are also prefixed, while in IDP are infixed (e.g., the comparison `math:lessEq` vs. $=<$).

Rules: These are wrapped into Forall/$\forall$ quantifiers, and built on atoms, possibly within an And/&, for both PSOA (:-) and IDP ($\Leftarrow$).

The interoperation between PSOA RuleML and IDP provides a link between the Semantic Technologies Community (e.g., N3 [1]) and the Decision Management Community (e.g., OMG DMN [15]). For example, it enables the interoperation

path N3→PSOA↔IDP←DMN (for the link N3Basic→PSOA see [18] and for the link IDP←DMN see [4][12]).

5.2 Semantics-Preserving Co-execution

After having performed several experiments with the PSOA version developed from [13], we have also experimented with the new IDP version of the same knowledge base.

For the common core of the KBs presented in Section 4.1 (assuming only two aircraft to conserve space), the answers of PSOA queries are in accordance with the IDP least model, as shown below:

```
% PSOA queries

:AircraftIcaoCategory(?a ?c)

Answer(s):
?a=_be9l ?c=<.../4444#Light>
?a=_a388 ?c=<.../4444#Super>

:icaoSeparation(:leader->a388
                :follower->be9l
                :miles->?d)
Answer(s):
?d=8
```

```
// IDP least model

structure  : V {
  Leader = { "a388" }
  Follower = { "be9l" }
  ...
  AircraftIcaoCategory = { "be9l", Light;
                           "a388", Super}
  ...


  IcaoSeparation = { "a388","be9l"->8 }
}
```

In this example, as in all consistent cases (see Section 6), PSOA and IDP provide semantically compatible results. In general, PSOA/IDP co-execution benefits both systems for the following reasons:

1. We have compared and cross-validated the results from both systems. The inconsistencies that were discovered in the original regulations, using both systems, have been described in Section 6.
2. The top-down processing (backward-reasoning) of PSOATransRun is complementary to the bottom-up processing (forward-reasoning) of the IDP system. Since the ATC KB's required logical expressiveness is on the level of Datalog (function-free Horn logic), both methods are applicable, although there is the expected speed/memory trade-off.
3. Each system can be used for a task it is best suited to. For example, decimal-preserving numeric calculations are not supported by IDP, but are available in PSOA. Therefore, calculations involving decimal numbers are handled by PSOA. On the other hand, as discussed in Section 7.1, IDP uses constraint solving for efficiently optimizing the landing order to obtain the total separation minimum.

[12] The IDP language typically offers more expressivity than DMN decision tables. Current work focusses on an extension to DMN to strenghten the link IDP→DMN.

6 Inconsistencies within Regulations

The process of aligning and co-executing several KBs does not only serve a theoretical purpose. These steps are especially useful in the construction of the KB. The detection of inconsistencies during this research project is an example of the added value of our approach. The KB validation for both PSOA and IDP aims to ensure the completeness (i.e. all aircraft will be categorized) and consistency (i.e. all individual aircraft are categorized in exactly one category for each applicable regulation). It serves a two-fold purpose: First, to ensure that the KB is in accordance with the regulations. Second, to ensure that the regulations and the source dataset are complete and consistent.

The PSOA KB in [13] considers that an aircraft is represented by its ICAO type designator and assigns the latter as an `oid`. This design choice, while efficient when the KB is used as a computational tool where individual aircraft would be handled by a front-end framework, can lead to problems in stand-alone execution: as a specific aircraft type can be assigned in more that one category due to variations (see e.g., [9]), an aircraft `oid` can be categorized in two different categories, as demonstrated by the following PSOA RuleML query:

```
And(:AircraftIcaoCategory(?a ?X) :AircraftIcaoCategory(?a ?Y)
    External(isopl:generic_not_eq(?X ?Y)))
Answer(s):
?a=<http://users.ntua.gr/mitsikas/ATC_KB#b350>
?X=<https://www.icao.int/safety/airnavigation/4444#Light>
?Y=<https://www.icao.int/safety/airnavigation/4444#Medium>

?a=<http://users.ntua.gr/mitsikas/ATC_KB#c207>
?X=<https://www.icao.int/safety/airnavigation/4444#Light>
?Y=<https://www.icao.int/safety/airnavigation/4444#Medium>
```

As explained in [13], the first result is a case where variants of the same type can be categorized in different categories, while the second result is an inconsistency of the source dataset.

Additional validation of the regulations can be realized by using PSOA RuleML or IDP. In [6], the categorization for categories **D** and **F** according to RECAT regulations is defined:

Category D. Aircraft capable of MTOW of less than 300,000 pounds and wingspan greater than 125 ft and less than or equal to 175 ft; or aircraft with wingspan greater than 90 ft and less than or equal to 125 ft.

Category F. Aircraft capable of MTOW of less than 41,000 pounds and wingspan less than or equal to 125 ft, or aircraft capable of MTOW less than 15,500 pounds regardless of wingspan, or a powered sailplane.

According to the above, any aircraft capable of MTOW of less than 41,000 pounds with wingspan greater than 90 ft and less than or equal to 125 ft would be categorized in both **D** and **F** categories.

This inconsistency was discovered by both PSOA and IDP. In PSOA, the addition of the `generic_not_eq` built-in in the PSOATransRun made possible the construction of appropriate non-ground queries, as shown below (some results and prefixes are omitted):

```
And(:AircraftRecatCategory(?a ?X) :AircraftRecatCategory(?a ?Y)
    External(isopl:generic_not_eq(?X ?Y)))
Answer(s):
?a=<.../ATC_KB#dc3>
?X=<...#D> ?Y=<...#F>

?a=<...ATC_KB#dhc4>
...
```

where the obtained answers identified the problem. In [13] the problem was not identified, as such non-ground queries were not possible to construct due to lack of the `generic_not_eq` built-in.

A later revision of the regulations attempted to correct the above problem and can be seen in [8,9]:

Category D. Aircraft capable of MTOW of less than 300,000 pounds and wingspan greater than 125 ft and less than or equal to 175 ft; or aircraft capable of a MTOW greater than 41,000 pounds with a wingspan greater than 90 ft and less than or equal to 125 ft.

This definition leads to an incompleteness for aircraft capable of MTOW of exactly 41,000 pounds with wingspan greater than 90 ft and less than or equal to 125 ft. While an aircraft with the above characteristics did not exist in the KB, the discovery of the incompleteness was made by PSOA by adding witness aircraft representing corner cases.

The behavior of the IDP system for these different inconsistencies depends on the modeling choice that have been made. In general, the more accurate the domain knowledge is represented, the more likely the system will behave as expected with inconsistencies. In this example, the appropriate way to model the categorization, would be to use a function `IcaoAircraftCategory(Aircraft):Category` and the definitional implication that was already discussed in Section 4. The definition assumes a closed world: a definition contains a set of sufficient and necessary conditions. In practice this means that no model can be found and an `unsatisfiable` message will be shown if not every case is defined, independent of the presence of an aircraft of 41,000 pounds. It will act likewise if overlapping categories have been defined. It can be tedious to find the exact inconsistency in a theory. Specific inferences can be used to help identify the exact problem (e.g. `explainunsat` and `printunsatcore`). If we move away from the ideal model, e.g. by defining categories as separate definitions, using the *if and only if* operator, the missing definition will only be discovered if an aircraft of 41,000 pounds is present in the database. Finally, the last way to model the categorization, is the use of material implication. If no category is explicitly assigned to aircraft of 41,000 pounds, any category may be assigned. This is the most dangerous situation, as a random category will be assigned.

While both PSOA and IDP helped to identify the above inconsistency and incompleteness in the regulations, this was through different mechanisms: PSOA RuleML needs the construction of an appropriate query and an example in the KB (a "witness"). If the domain knowledge is appropriately modeled in IDP,

inconsistencies or incompleteness will be found without such a witness. Thus, using both approaches to model safety-critical use cases, can benefit the final KB and can also help to identify such problems in safety-critical regulations.

7 Extensions of the KB

Both KBs are easily expandable, for example if new categories are added in the existing regulations, or if other regulations (e.g. RECAT) should be included in the same KB. Because of the strongtyping of IDP, this does however ask attention to avoid overloading. If other regulations use the same category names, an additional letter or a prefix should be added in IDP to discern between the *Heavy* category of one regulation-set versus the *Heavy* category of the other regulations. The use of prefixes to handle different ICAO-FAA categories with the same name is also recommended in PSOA.

7.1 Optimization of Landing Order

As described in Section 2.1 IDP supports a variety of inferences that can be applied on the KB. An example of an application of this is the calculation of an optimal landing order of a number of given aircraft. There are multiple ways to define the optimal landing order. Typically this is either a time-based optimum (to minimise the time between consecutive aircraft landings) or a distance-based optimum. In the latter case, the purpose is to minimise the total separation distance for a series of aircraft, based on the pairwise separation minima. As we already formalised the pairwaise calculation of the separation minimum, we will optimise the distance based metric.

An approach queue is constructed from a subset of aircraft in the KB, e.g. as follows:

```
Waiting1 = a319; Waiting2 = a388; Waiting3 = b788; Waiting4 = be9l
```

The pairwise separation minima that are calculated with the main program are considered to be given for the example, and can be consulted in Table 2.

Table 2. ICAO separation minima for specific aircraft combinations

ICAO separation minima (nautical miles)					
		Follower			
		a319	a388	b788	be91
Leader	a319	3	3	3	5
	a388	7	3	6	8
	b788	5	3	4	6
	be91	3	3	3	3

We want to come up with an order of aircraft: Leader, Follower1, Follower2, and Follower3. The optimal order of aircraft is the one the minimises the sum of the consecutive separation minima. A term `totalseparation` is created: $sum\{ac : Leader = ac|Follower1 = ac|Follower2 = ac|Follower3 = ac| : Separation(ac, Next(ac))\}$
With the inference `Minimise` a term, in this case `totalseparation`, is minimised. A random order of aircraft could be : *Leader* be9l; *Follower1* b788; *Follower2* a319; *Follower3* a388. The total separation minimum, which is calculated as the sum of consecutive separation minima is 11NM. After minimization, the combination is *Leader* be9l; *Follower1* a319; *Follower2* b788; *Follower3* a388 with a separation minimum of 9.

7.2 Perspectival ATC KB Version

PSOA RuleML explicitly specifies for each descriptor (tuples, slots) whether it is to be interpreted *dependent on* (under the perspective of) the predicate in whose scope it occurs. This perspectivity dimension refines the design space between oidless atoms with a single dependent tuple (relationships) and oidful atoms with only independent slots (framepoints): It also permits atoms with independent tuples and atoms with dependent slots, the latter denoted by "+>" (instead of "->" for independent slots, e.g. used in Section 4.1). This supports advanced data and knowledge representation where, e.g., a slot name can have different fillers each depending on a predicate [2].

For disambiguation of multi-valued slots, the ATC KB was enriched for a perspectival graph version. Examples of perspectival KB facts are shown below (the slot denoting the wake turbulence category, `wtc`, has two fillers, `icao:Super` vs. `faa:Super`, disambiguated by the two perspectivity-providing predicates, `IcaoRegulated` vs. `FaaRegulated`[13]):

```
%% ICAO Wake Turbulence Categories, Super %%
:a388#:IcaoRegulated(wtc+>icao:Super)
:a38f#:IcaoRegulated(wtc+>icao:Super)

%% FAA Wake Turbulence Categories, Super %%
:a388#:FaaRegulated(wtc+>faa:Super)
:a38f#:FaaRegulated(wtc+>faa:Super)
:a225#:FaaRegulated(wtc+>faa:Super)
```

Interoperation between such perspectivally enriched PSOA KBs and IDP KBs would require a dependence-to-independence reduction [2].

8 Conclusions and Future Work

In this paper, we presented the specification in IDP and PSOA of an Air Traffic Control use case. We discussed the alignment of both specifications and the implications of modeling choices that are involved in this. We demonstrated that a partial interoperation is possible for facts and rules. During the process of

[13] For FAA wake turbulence categories see e.g. [7].

constructing and aligning the KBs, some inconsistencies in the original regulations were discovered. The discovery occurred differently for both systems, which points to their respective unique features and to their internal functioning. It also demonstrates the added value of combining two separate systems to formalize the same knowledge. As the systems can be co-executed, the advantages of each system can be exploited from within a combined application. Examples of this are the introduction of optimization, which is only efficient in the constraint-based system IDP, and the disambiguation of slots via perspectivalness, which is only possible in the graph-based system PSOA RuleML.

Future work includes the round-trippable translation between increasing subsets of the two languages. An application can be created in which both systems are connected through an API. Based on the PSOA/IDP cross-fertilization, both systems can be further developed, e.g. by support for a separated vocabulary in PSOA RuleML and for graph modeling in IDP. PSOA and IDP could be aligned for constructs used in additional KBs, ultimately defining the complete intersection of PSOA and IDP constructs. Conversely, additional languages could be aligned for formalizing the ATC KB, ultimately making ATC KB a standard use case. Future extensions to regulations could be easily incorporated into the existing KBs. ATC KB could become a shared resource of a multi-agent environment founded on [16].

Acknowledgments. The author of the Hellenic Civil Aviation Authority wants to thank his colleagues for discussions about the domain-expert knowledge. Any deficiencies are ours and we make this disclaimer: The described work constitutes an informative computational model of ATC regulations and is not intended for use in real aviation environments. The authors of KU Leuven are partially supported by the Flemish Agency for Innovation and Entrepreneurship TETRA project (HBC.2017.0039).

References

1. Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., Hendler, J.: N3Logic: A Logical Framework for the World Wide Web. Theory and Practice of Logic Programming (TPLP) **8**(3) (May 2008)
2. Boley, H., Zou, G.: Perspectival Knowledge in PSOA RuleML: Representation, Model Theory, and Translation. CoRR **abs/1712.02869** (2017), `http://arxiv.org/abs/1712.02869`
3. de Cat, B., Bogaerts, B., Bruynooghe, M., Denecker, M.: Predicate logic as a modelling language: The IDP system. CoRR **abs/1401.6312** (2014), `http://arxiv.org/abs/1401.6312`
4. Deryck, M., Hasic, F., Vanthienen, J., Vennekens, J.: A case-based inquiry into the decision model and notation (DMN) and the knowledge base (KB) paradigm. In: Benzmüller, C., Ricca, F., Parent, X., Roman, D. (eds.) Rules and Reasoning - Second International Joint Conference, RuleML+RR 2018, Luxembourg, September 18-21, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11092, pp. 248–263. Springer (2018). https://doi.org/10.1007/978-3-319-99906-7_17

5. FAA: Aircraft characteristics database. https://www.faa.gov/airports/engineering/aircraft_char_database/, accessed: 2019-05-31
6. FAA: Advisory Circular 90-23G - Aircraft Wake Turbulence (2014)
7. FAA: ORDER JO 7110.65V, Air Traffic Control (2014)
8. FAA: Order JO 7110.659C, Wake Turbulence Recategorization (2016)
9. FAA: Order JO 7360.1C - Aircraft Type Designators (2017)
10. ICAO: Doc 4444-RAC/501, Procedures for Air Navigation Services - Rules of the Air and Air Traffic Services
11. Lang, S., Tittsworth, J., Bryant, W., Wilson, P., Lepadatu, C., Delisi, D., Lai, D., Greene, G.: Progress on an ICAO Wake Turbulence Re-Categorization Effort. AIAA Atmospheric and Space Environments Conference (2010). https://doi.org/10.2514/6.2010-7682
12. McCluskey, T., Porteous, J., Naik, Y., Taylor, C., Jones, S.: A requirements capture method and its use in an air traffic control application. Software: Practice and Experience **25**(1), 47–71 (1995)
13. Mitsikas, T., Almpani, S., Stefaneas, P., Frangos, P., Ouranos, I.: Formalizing Air Traffic Control regulations in PSOA RuleML. In: Proceedings of the Doctoral Consortium and Challenge@ RuleML+ RR 2018 hosted by 2nd International Joint Conference on Rules and Reasoning. vol. 2204. CEUR Workshop Proceedings (2018), http://ceur-ws.org/Vol-2204/paper9.pdf
14. Mitsikas, T., Stefaneas, P., Ouranos, I.: A Rule-Based Approach for Air Traffic Control in the Vicinity of the Airport. In: Algebraic Modeling of Topological and Computational Structures and Applications. pp. 423–438. Springer International Publishing, Cham (2017)
15. Object Management Group (OMG): Decision Model and Notation 1.2. https://www.omg.org/spec/DMN/1.2/ (2019)
16. Valkanas, G., Natsiavas, P., Bassiliades, N.: A collision detection and resolution multi agent approach using utility functions. In: Kefalas, P., Stamatis, D., Douligeris, C. (eds.) 2009 Fourth Balkan Conference in Informatics, BCI 2009, Thessaloniki, Greece, 17-19 September 2009. pp. 3–7. IEEE Computer Society (2009), http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5358991;http://www.computer.org/csdl/proceedings/bci/2009/3783/00/index.html
17. Van Hertum, P., Dasseville, I., Janssens, G., Denecker, M.: The KB paradigm and its application to interactive configuration. Theory and Practice of Logic Programming **17**(1), 91–117 (2017)
18. Zou, G.: Translators for Interoperating and Porting Object-Relational Knowledge. Ph.D. thesis, Faculty of Computer Science, University of New Brunswick (Apr 2018), https://unbscholar.lib.unb.ca/islandora/object/unbscholar%3A9279
19. Zou, G., Boley, H., Wood, D., Lea, K.: Port Clearance Rules in PSOA RuleML: From Controlled-English Regulation to Object-Relational Logic. In: Proceedings of the RuleML+RR 2017 Challenge. vol. 1875. CEUR (Jul 2017), http://ceur-ws.org/Vol-1875/paper6.pdf