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 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 additionally 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 to allow the determination of an optimal sequence of more than two aircraft.

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 Positional-Slotted

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

Object-Applicative (PSOA) RuleML version of the ATC KB [14][6] as a starting point for an IDP (Imperative/Declarative Programming) 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 Air Traffic Control (ATC) regulations and data obtained from [6], the PSOA specification was created. From it, we derive the IDP KB and we study the similarities and differences between both specifications. In the first step, we try to align both systems by choosing similar ways of modeling from different possibilities. In doing this, we discovered that there were not only inconsistencies in the source aircraft characteristics data as discussed in [14], 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 allowing us to validate results from both systems. The resulting KBs 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 [13] and [15]. 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 presented examples are expressed in many-sorted first-order logic or in the Prolog notation, and include rules about conflict prediction and aircraft separation. Supplementary, the model was validated by automated processes, formal reasoning, and domain experts. [15] focuses on capturing ATC regulations valid in the airport area. The authors formalized the separation minima mandated by International Civil Aviation Organization (ICAO), Federal Aviation Administration (FAA), and FAA's "RECAT" regulations in POSL RuleML. It formed the foundations for further expansion that focuses on cases of conditional reduced separation minima. It was the basis for the development of [14], 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 ATC 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 Base Paradigm

The IDP system [3] adheres to the Knowledge Base Paradigm (KBP): it stresses the distinction between domain knowledge *an sich*, and the different ways in which this knowledge can be put into use [18]. The domain knowledge is formalized and centralized in a KB, which collects not only simple knowledge (e.g., data in a database), but also complex knowledge, such as definitions, implications, propositions, etc. One of the advantages of this separation of concerns of knowledge versus problem solving, is the high maintainability of the KB because it only contains descriptive information on the domain. Another advantage is the flexibility to use this KB in different – often unforeseen – use cases.

The IDP system allows KBs to be written in the IDP language, which is based on typed first-order logic, enriched with features such as aggregates and inductive definitions. A KB typically consists of three kinds of components. A *vocabulary* describes the logical symbols (types, constants, functions and predicates) that are used to formulate the domain knowledge. It represents the ontology of the domain. A *structure* for a vocabulary provides an interpretation for each of the symbols of this vocabulary. Finally, a *theory* contains the actual domain knowledge, represented as a set of formulas. A formula can be either a classical first-order logic (FO) formula, or a *rule-based (inductive) definition.* For instance, a theory can contain as a formula the following inductive definition of the transitive closure T of a graph G:

$$\left\{\begin{array}{r}\forall x\ y : T(x,y) \leftarrow G(x,y).\\ \forall x\ y : T(x,y) \leftarrow \exists z : T(x,z) \land T(z,y).\end{array}\right\}$$

The rules in such a definition are built using the *definitional implication* symbol $\leftarrow$, which is to be distinguished from the *material implication* of classical logic, the latter denoted as $\Leftarrow$ in IDP. The formal semantics of such a definition is given by its *well-founded model*, because this coincides with the expected semantics of an inductive definition [4]. The semantics of the FO formulas in a theory is simply given by the standard satisfaction relation $\models$ of classical logic.

The IDP system allows a number of different inference tasks to be performed on a KB. The most common is that of *Model Expansion (MX)*: for a given vocabulary V, and theory T and structure S for some subvocabulary $V' \subseteq V$, MX constructs a structure S' for the entire vocabulary V that extends S (i.e., $\sigma^S = \sigma^{S'}$ for all symbols $\sigma \in V'$) and that is a model of T (i.e., $S' \models T$). Another useful inference task is that of *Optimization*, which selects the most optimal structure (according to some provided criterion) among all the possible model expansions of a given S w.r.t. T. The optimality criterion is provided in the form of a term t which must be minimized, i.e., the solution is the model expansion S' for which the value of $t^{S'}$ of the term t in the structure S' is minimal.

2.2 PSOA RuleML for Graph-Relational Knowledge

PSOA RuleML generalizes RIF-BLD and POSL RuleML by a homogeneous integration of table-like relationships and graph-like frames into ***positional-slotted object-applicative*** (***psoa***) *terms.* The initially used *single-dependent-tuple* *independent-slot special case of psoa terms*, oidless or oidful, has these forms [2,20] (where $n \geq 0$ and $k \geq 0$, of which we focus on either $n = 0$ for – oidless – *frameships* and – oidful – *framepoints* or $k = 0$ for – oidless – *relationships*):

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

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

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

For example, in ATC KB, the term `:be91#:Aircraft(:mtow->9300.0)` is a single-slot framepoint atom, while `:AircraftIcaoCategory(:a388 icao:Super)` is a binary relationship. Both are *ground* atoms, i.e. variableless.

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 then consists of clauses that are ground facts and non-ground rules: while facts are – ground – 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 an `And`-prefixed conjunction of psoa atoms.

The reference implementation for deduction in PSOA RuleML is the open-source framework system PSOATransRun, currently in Version 1.4.2[7].

3 Air Traffic Control Regulations

Collision prevention in ATC is realized by ensuring a minimum distance between aircraft, a concept also called *separation minimum.* Separation of aircraft serves an additional role, which is the avoidance of wake turbulence. The separation minimum is defined for aircraft pairs depending on their wake turbulence category. The current FAA and ICAO regulations categorize aircraft according to their maximum takeoff weight/mass (MTOW/MTOM). MTOW/MTOM represent the wake turbulence of 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 a wake turbulence recategorization (RECAT), which recategorizes aircraft in six categories, taking into account the wingspan as an additional parameter.

For example, ICAO discerns four categories: Light (MTOM of 7000 kg or less), Medium (MTOM of greater than 7000 kg, but less than 136000 kg), Heavy (MTOM of 136000 kg or greater), and Super (a separate designation that currently only refers to the Airbus A380 with MTOM 575000 kg) [11,12]. The associated separation minima for flights under Instrument Flight Rules (IFR)[8] are defined in Table 1[9]. The Minimum Radar Separation (MRS), is 3 NM or 2.5 NM depending on operational conditions unrelated to wake turbulence (e.g. visibility) [12].

[7] `http://psoa.ruleml.org/transrun/1.4.2/local/`

[8] Separation minima for flights on Visual Flight Rules (VFR) are time-based [8,11].

[9] The minima set out at Table 1 shall be applied when e.g. both aircraft are using the same runway, or parallel runways separated by less than 760 m (2 500 ft) [11].

Table 1. Current ICAO weight categories and associated separation minima [12]

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

4 Alignment

KB languages should be able to represent objects, facts, and relations in the knowledge 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 KBs. Typical parts of this PSOA-IDP alignment are shown below (aircraft-characterizing facts were obtained from [6]):

```
% PSOA KB fragment (from                  // IDP KB fragment (from
% users.ntua.gr/mitsikas/                 // gitlab.com/mderyck/atc-kb-idp/)
% ATC_KB/atc_kb-v201906.psoa)
                                          vocabulary V {
                                              type Mtom isa int
                                              type Aircraft isa string
                                              MTOM(Aircraft,Mtom)
                                                  ... }
                                          theory T:V{
Forall ?a ?w (                                !a[Aircraft] w[Mtom]:
  :AircraftIcaoCategory(?a icao:Light) :-     AircraftIcaoCategory(a, Light) <=
    And(?a#:Aircraft(:mtom->?w)                   MTOM(a,w)
        math:lessEq(?w 7000))  )                      & w =< 7000.

Forall ?a ?w (                                !a[Aircraft] w[Mtom]:
  :AircraftIcaoCategory(?a icao:Medium) :-    AircraftIcaoCategory(a, Medium) <=
    And(?a#:Aircraft(:mtom->?w)                   MTOM(a,w)
        math:greaterThan(?w 7000)                         & 7000 < w
        math:lessThan(?w 136000))  )                      & w < 136000.

Forall ?a ?w (                                !a[Aircraft] w[Mtom]:
  :AircraftIcaoCategory(?a icao:Heavy) :-     AircraftIcaoCategory(a, Heavy) <=
    And(?a#:Aircraft(:mtom->?w)                   MTOM(a,w)
        math:greaterEq(?w 136000)                         & 136000 =< w
        not:Naf(:AircraftIcaoCategory(?a icao:Super))     & a ~= a388
    )                                                     & a ~= a38f.
)

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

%% ICAO Separation example                    // ICAO Separation Example

Forall ?l ?f (                                !l[Leader],f[Follower]:
    :icaoSeparation(:leader->?l
                    :follower->?f             IcaoSeparation(l, f) = 8 <=
                    :miles->8) :-
    And(:AircraftIcaoCategory(?l icao:Super)      AircraftIcaoCategory(l, Super)
        :AircraftIcaoCategory(?f icao:Light))     & AircraftIcaoCategory(f, Light).
    )                                         }

%% Sample Aircraft Facts %%               structure S1 : V {
                                              //specific value assignments:
:be91#:Aircraft(:mtom->4218.41                Leader = {a388}
                :mtow->9300.0                 Follower = {be91}
                :wingspan->45.92              ...
                :appSpeed->100.0)
                                              //aircraft data
:a388#:Aircraft(:mtom->575000.0               MTOM = {be91, 4218; a388, 575000}
                :mtow->1267658.0              MTOW = {be91, 9300; a388, 1267658}
                :wingspan->261.65             WingSpan = {be91, 45; a388, 261}
                :appSpeed->145.0)             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 KBs For the alignment we created specifications that are classically equivalent, in the sense that they both have the same class of possible worlds: an interpretation (called a structure in IDP) W satisfies the PSOA fragment P if and only if it satisfies the IDP fragment I: $W \models P \;\Leftrightarrow\; W \models I$.

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.

The existence of these “extra” models is not a problem for PSOA, since this system uses the KB by means of the inference task of *query answering*, which looks for properties that hold in *all* models of the specification.

The IDP system offers the same inference task, allowing the above IDP specification to be used in precisely the same way (and, because of the equivalence of the two specifications, producing identical results). However, this is not an idiomatic use of IDP: IDP adheres to the KBP, which emphasizes that the same KB should be usable by different inference tasks. To guarantee that different inference tasks produce correct results, it is crucial that the KB is constructed in such a way that its models correspond one-to-one to possible worlds in reality. The above specification obviously does not have this property and therefore produces correct results *only* when the inference task of query answering is applied to it. When applying, e.g., the inference task of *model expansion*, we will 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 means of a *definition*, 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. As mentioned above, such a set of definitional implications is interpreted under the well-founded semantics. Since the above specification is not recursive, this means that it is equivalent to its Clark's completion. In other words, it states that an aircraft belongs to a certain category if *and only if* it satisfied the corresponding condition. Therefore, the definitional IDP specification only has a single model, in which each aircraft is assigned a single category, namely that whose condition it satisfies. This specification is therefore considerably stronger than the PSOA specification. However, when we apply either *model expansion* (compute a single model) or *query answering* (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-dependent-tuple independent-slot* special case of psoa terms (cf. 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 dependent-slotted 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 ICAO regulations identify 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*. 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[10]. In PSOA, we have an appealing alternative in the use of *negation as failure (naf)*: we can write `not: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 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 representation 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 [14], 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.

[10] 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.

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 [16]). For example, it enables the interoperation path N3$\rightarrow$PSOA$\leftrightarrow$IDP$\leftarrow$DMN (for the link N3Basic$\rightarrow$PSOA see [19] and for the link IDP$\leftarrow$DMN see [5][11]).

5.2 Semantics-Preserving Co-execution

After having performed several experiments with the PSOA version developed from [14], we have also experimented with the new IDP version of the same KB.

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

```
% PSOA queries                            // IDP least model

                                          structure  : V {
:AircraftIcaoCategory(?a ?c)                Leader = { "a388" }
                                            Follower = { "be9l" }
Answer(s):                                  ...
?a=<...#be9l ?c=<...#Light>                 AircraftIcaoCategory = { "be9l", Light;
?a=<...#a388 ?c=<...#Super>                                          "a388", Super}
                                            ...
:icaoSeparation(:leader->:a388
                :follower->:be9l
                :miles->?d)
Answer(s):
?d=8                                        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 currently 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 a landing queue.

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 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 [14] 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., [10]), 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=<...#b350> ?X=<...#Light> ?Y=<...#Medium>
?a=<...#c207> ?X=<...#Light> ?Y=<...#Medium>
```

[11] The IDP language typically offers more expressivity than DMN decision tables. Current work focuses on an extension to DMN to strengthen the link IDP→DMN.

As explained in [14], 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 [7], 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, appropriate non-ground queries can identify the problem, as shown below:

```
And(:AircraftRecatCategory(?a ?X) :AircraftRecatCategory(?a ?Y)
    External(isopl:generic_not_eq(?X ?Y)))
Answer(s):
?a=<...#dc3> ?X=<...#D> ?Y=<...#F>
?a=<...#dhc4> ...
```

In [14] 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 before PSOA-TransRun Version 1.4.1.

A later revision of the regulations attempted to correct the above problem and can be seen in [9,10]:

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 minimize the time between consecutive aircraft landings) or a distance-based optimum. In the latter case, the purpose is to minimize the total separation distance for a series of aircraft, based on the pairwise separation minima. As we already formalized the pairwise calculation of the separation minimum, we will optimize the distance based metric.

An approach queue is constructed from a subset of aircraft, e.g.; `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, e.g.; the pair leader a319 with follower

a388 has 3 NM as separation minimum. We now want to come up with an order of aircraft: Leader, Follower1, Follower2, and Follower3; that minimizes 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 minimized. 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 Dependent-Slot ATC KB Version

PSOA RuleML explicitly specifies for each descriptor (tuple, slot) whether it is to be interpreted *dependent on* (under the perspective of) the predicate in whose scope it occurs. This dependency 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, for the same OID, a slot name can have different fillers depending on a predicate [2].

For the disambiguation of multi-valued slots, the ATC KB was enriched into a dependent-slot graph version. Examples of dependent-slot KB facts are shown below (the slot denoting the wake turbulence category, `wtc`, has two fillers, `icao:Super` vs. `faa:Super`, disambiguated, for `:a388` and `:a38f`, by the two perspective-providing predicates, `IcaoRegulated` vs. `FaaRegulated`[8]:

```
%% 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 dependence-enriched PSOA KB and IDP KB 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 constructing and aligning the KBs, some inconsistencies in the original regulations were discovered. The discovery process was different 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 their dependence, 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 [17].

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 (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, v3** (2019)
3. de Cat, B., Bogaerts, B., Bruynooghe, M., Denecker, M.: Predicate logic as a modelling language: The IDP system. CoRR **abs/1401.6312** (2014)
4. Denecker, M., Vennekens, J.: The well-founded semantics is the principle of inductive definition, revisited. In: Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning. pp. 1–10 (2014)
5. 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: 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
6. FAA: Aircraft characteristics database. `https://www.faa.gov/airports/engineering/aircraft_char_database/`, accessed: 2019-05-31

7. FAA: Advisory Circular 90-23G - Aircraft Wake Turbulence (2014)
8. FAA: ORDER JO 7110.65V, Air Traffic Control (2014)
9. FAA: Order JO 7110.659C, Wake Turbulence Recategorization (2016)
10. FAA: Order JO 7360.1C - Aircraft Type Designators (2017)
11. ICAO: Doc 4444-RAC/501, Procedures for Air Navigation Services - Rules of the Air and Air Traffic Services
12. 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
13. 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)
14. 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)
15. 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 (2017)
16. Object Management Group (OMG): Decision Model and Notation 1.2. `https://www.omg.org/spec/DMN/1.2/` (2019)
17. Valkanas, G., Natsiavas, P., Bassiliades, N.: A Collision Detection and Resolution Multi Agent Approach Using Utility Functions. In: 2009 Fourth Balkan Conference in Informatics, BCI 2009, Thessaloniki, Greece, 17-19 September 2009. pp. 3–7. IEEE Computer Society (2009)
18. Van Hertum, P., Dasseville, I., Janssens, G., Denecker, M.: The KB paradigm and its application to interactive configuration. Theory and Practice of Logic Programming (TPLP) **17**(1), 91–117 (2017)
19. Zou, G.: Translators for Interoperating and Porting Object-Relational Knowledge. Ph.D. thesis, Faculty of Computer Science, University of New Brunswick (Apr 2018)
20. 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)