Port Clearance Rules in PSOA RuleML: From Controlled-English Regulation to Object-Relational Logic

Gen Zou, Harold Boley, Dylan Wood, Kieran Lea

Faculty of Computer Science,
University of New Brunswick, Fredericton, Canada
{gen[DT]zou, harold[DT]boley, dylan[DT]wood1, kieran[DT]lea}[AT]unb[DT]ca

Abstract. The Decision Management (DM) Community Challenge of March 2016 consisted of creating decision models from ten English Port Clearance Rules inspired by the International Ship and Port Facility Security Code. Based on an analysis of the moderately controlled English rules and current online solutions, we formalized the rules in Positional-Slotted, Object-Applicative (PSOA) RuleML. This resulted in: (1) a reordering, subgrouping, and explanation of the original rules on the specialized decision-model expressiveness level of (deontically contextualized) near-Datalog, non-recursive, near-deterministic, ground-queried, and non-subpredicating rules; (2) an object-relational PSOA RuleML rulebase which was complemented by facts to form a knowledge base queried in PSOATransRun for decision-making. Thus, the DM and logical formalizations get connected, which leads to generalized decision models with Hornlog, recursive, non-deterministic, non-ground-queried, and subpredicating rules.

1 Introduction

The International Rule Challenges have evaluated use cases such as the UServ Product Derby Case Study [1][1], which employs Deliberation RuleML on the NafNegHornlogEq level of expressiveness. Here, we demonstrate a use case for a real-world problem, which employs PSOA RuleML on the near-Datalog level along with its open-source PSOATransRun reference implementation.

The Decision Management (DM) Community[2] has been running Challenges about decision modeling problems[3] since 2014. The DM Community Challenge of March 2016 consisted of creating decision models from the structured text of ten English Port Clearance Rules available online[4], inspired by the International Ship and Port Facility Security Code. They were originally developed for "The Game of Rules" [2][5].

[1] http://deliberation.ruleml.org/1.02/exa/RulebaseCompetition2014/
[2] https://dmcommunity.org
[3] https://dmcommunity.org/challenge/
[4] https://dmcommunity.org/challenge/challenge-march-2016/
[5] http://www.buildingbusinesscapability.com/presentations/2015/2200.pdf

The English of each one of the independently given Port Clearance Rules – as a kind of "legalese"[6] – is moderately controlled, some having a structured 'if' part, which supports their formalization. On the other hand, the textual order of the rules – although logically immaterial – does not reflect their most human-readable arrangement in the created decision models. At the time of this writing, there are eight online solutions[7], some using decision tables or decision trees as their main modeling technique.

Based on an analysis of the English Port Clearance Rules and several solutions, we formalized the rules in Positional-Slotted, Object-Applicative (PSOA) RuleML [3–6][8], complemented them by port clearance facts (data) directly in PSOA RuleML, explored the resulting Knowledge Base (KB) with systematic queries in PSOATransRun [5], and propose generalized decision models.

The paper is organized as follows: Section 2 explains the basics of PSOA RuleML as required for the subsequent sections. Section 3 describes the formalization of the controlled English rules in PSOA RuleML. Section 4 complements the PSOA rules by facts and demonstrates queries. Section 5 concludes the paper. Appendix A provides the PSOA source of the entire Port Clearance KB.

2 PSOA RuleML in a Nutshell

PSOA RuleML is an object-relational Web rule language that generalizes RIF-BLD and POSL by a homogeneous integration of relationships (for composing, e.g., SQL tables from rows) and frames (for composing, e.g., RDF graphs from nodes & outgoing labeled arcs) into ***p**ositional-**s**lotted **o**bject-**a**pplicative* (***psoa***) *terms*, for the often used single-tuple case having these forms ($\mathtt{n} \geq 0$ and $\mathtt{k} \geq 0$):[9]

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

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

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 $\mathtt{p_j\text{->}v_j}$, $\mathtt{j} = 1, \ldots, \mathtt{k}$, each pairing a slot name (attribute) $\mathtt{p_j}$ with a slot filler (value) $\mathtt{v_j}$. An OID `o` typed by the root ("universal", "any", ...) class `f`=`Top` is considered as untyped.

A psoa term can be interpreted as a *psoa expression*, denoting an individual, or a *psoa atom*, denoting a truth value, depending on whether `f` is a function or predicate. A top-level psoa term is always interpreted as an atom. An embedded psoa term is interpreted as an atom if it has the oidful form (2); else, as an expression if it has the oidless form (1). An atomic formula containing an embedded psoa atom can be unnested into a conjunction, as explained in [6].

[6] While these core rules are not meant as a law-interpretation challenge, they are rich enough for development into a legal-informatics use case, as indicated in Section 5.

[7] `https://dmcommunity.org/challenge/challenge-march-2016/#Solutions`

[8] `http://wiki.ruleml.org/index.php/PSOA_RuleML`

[9] We use the all-upper-case "PSOA" as a reference to the language and the all-lower-case "psoa" for its terms. Earlier PSOA papers [3–6] show multi-tuple psoa terms.

Constants include `Top`, numbers, strings, and Internationalized Resource Identifiers (IRIs). An IRI has the full form `<...>` and can be abbreviated using a namespace prefix ending with '`:`', e.g., a full IRI `<http://ex.org/a>` can be abbreviated as `:a` if the KB contains a declaration `Prefix(: <http://ex.org/>)` for the prefix '`:`'. 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 an `at is large and has a validAnd`-prefixed conjunction of psoa atoms.

One PSOA KB can import other PSOA KBs using declarations having the form `Import(<...>)`, where `<...>` is a dereferencable IRI of an imported KB.

The reference implementation of PSOA RuleML is the Prolog instantiation of PSOATransRun [5][10], currently in version 1.3.

3 Formalizing the Port Clearance Rules in PSOA

To arrive at a PSOA RuleML decision model for Port Clearance Rules, the specific object-relational knowledge representation was chosen and the English Port Clearance Rules were formalized in PSOA RuleML presentation syntax (PSOA RuleML/PS).[11] As indicated in Section 1, the original textual rule order calls for a more human-readable version, although this entails that – to maintain the correspondence with the original – the PSOA RuleML model's rule numbers are not consecutive (e.g., the original Rule 4 becomes one of the last rules in PSOA's textual as well as visual forms). Note that the model-theoretic semantics [4,6] of PSOA RuleML – not relying on rule order – is associated with the presentation syntax rather than any decision-model visualization.[12]

The PSOA RuleML decision model for Port Clearance Rules is visualized in Fig. 1, an object-relational And-Or DAG ('And' branches are horizontally cross-linked) with rule names as nodes, e.g. Rule 2, and conclusion predicates as side labels of nodes, e.g. `:MayEnterDutchPortUnloaded`. For the not side-labeled nodes Rule 9, 1&5, and 4, the root-class predicate `Top` is understood, while slot names, e.g. `:size`, and slot fillers, e.g. `:small`, are shown as, respectively, labels of incoming arcs and top labels of the rule nodes (for the slot name `:hull` the filler `:double` does not require any further rule).

The blank, unlabeled node represents the only 'Or' branch in this model, where Rules 8 and 7 are – operationally speaking – 'pre-invoked' via the conclusion predicate `:MeetsSafetyRequirementsUnloaded`, having conditions with

[10] `http://wiki.ruleml.org/index.php/PSOA_RuleML#Prolog_Instantiation`

[11] See Section 5 for hints on the preliminary PSOA RuleML/XML serialization syntax.

[12] While these two notions of "model" are quite unrelated, PSOA's model theory applied to PSOA's Port Clearance Rules formalized in this section provides a semantics for this decision model. Likewise, for any other decision model in PSOA RuleML.

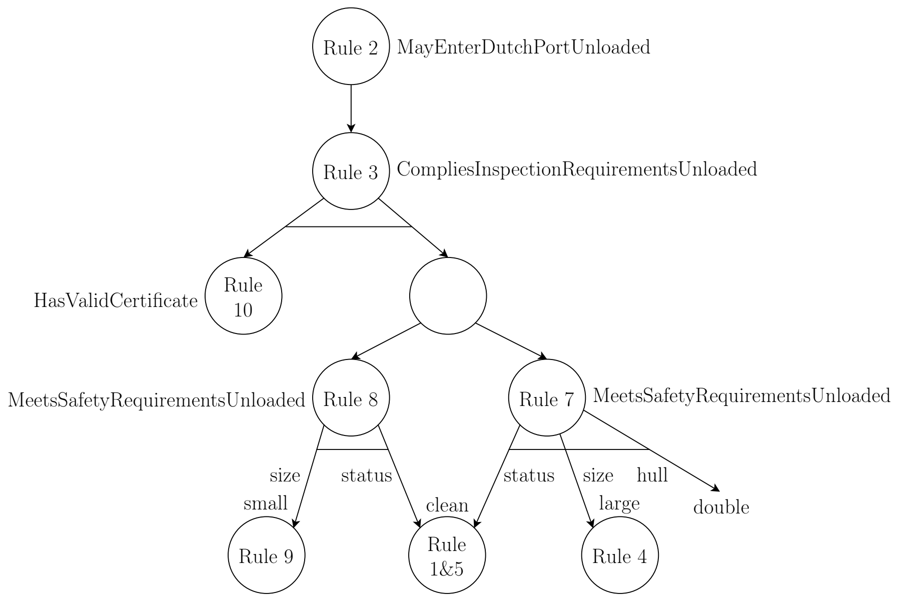
Rule 2
MayEnterDutchPortUnloaded
Rule 3
CompliesInspectionRequirementsUnloaded
HasValidCertificate
Rule 10
MeetsSafetyRequirementsUnloaded
Rule 8
Rule 7
MeetsSafetyRequirementsUnloaded
size
small
status
clean
status
size
large
hull
double
Rule 9
Rule 1&5
Rule 4

Fig. 1. Visualization of PSOA RuleML decision model for Port Clearance Rules.

a first conjunct immediately determining whether the slot `:size` is `:small` or `:large`, so that only either Rule 8 or Rule 7, respectively, can be 'fully invoked', causing *near-deterministic* behavior.

The model is object-relational – combining **frame** (object-centered) and **relational** representations – in that the upper part running to the conclusions of Rules 8 and 7 involves unary relations applied to ships while the lower part from the conditions of Rules 8 and 7 downward involves frames with ship OIDs described by slots. Hence, the upper 'And' branch between Rule 10 and the blank node corresponds to an explicit `And` in PSOA RuleML/PS, while the two lower 'And' branches between Rules 9 and 1&5 as well as between Rule 1&5, Rule 4 and slot `:hull` being `:double` correspond to the implicit conjunction of PSOA RuleML slots[13] (made explicit by *slotribution*, i.e. OID-over-slot distribution [3]).

We now describe the rules, top-down, in five subgroups, referring to the first half of Appendix A for the complete PSOA source.

In the **first subgroup** of rules, the original Rule 2 is the (DAG-)root of the decision model, passing on its ship argument to Rule 3, which conjoins the two requirements for compliance:

2. An unloaded ship may only enter a Dutch port if the ship complies with the requirements of the Inspection for unloaded ships.
3. A ship must comply with the requirements of the Inspection for unloaded ships if the ship complies with all of the following: a) the ship meets the safety

[13] We will later see that, of the conjoined slots, `:size` occurs directly in a `:Ship` frame while `:status` and `:hull` occur in its embedded `:ShipHold` frame.

requirements for unloaded ships; b) the ship has a certificate of registry that is valid.

In PSOA RuleML/PS, the subgroup is formalized as follows:

```
% Main relational rule invokes inspection rule for certificate And safety

% Rule 2
Forall ?s (
  :MayEnterDutchPortUnloaded(?s) :-
    :CompliesInspectionRequirementsUnloaded(?s)
)

% Rule 3
Forall ?s (
  :CompliesInspectionRequirementsUnloaded(?s) :-
    And(:HasValidCertificate(?s)
        :MeetsSafetyRequirementsUnloaded(?s))
)
```

While the original Rule 2's conclusion-side word "may" – encoded in the predicate name `:MayEnterDutchPortUnloaded` – creates a top-level context that can be given a modal – specifically, a deontic – interpretation, the original Rule 3's conclusion-side "must" is not again encoded in the predicate name `:CompliesInspectionRequirementsUnloaded`. Both rules are relational, on the Datalog level of expressiveness (predicates only have a variable, `?s`, no function application, as their argument). The `:Ship`-type test for `?s` is postponed to the later object-centered rules, where `:Ship` becomes a class.

The **second subgroup** consists of just the original Rule 10, which looks up a ship's certificate-validity date and checks whether the current date is before that:

10. A ship's certificate of registry must be considered valid if the date up to which the registration is valid of the certificate of registry is after the current date.

In PSOA RuleML/PS, this subgroup has two formal options (local/fixed date):

```
% Object-relational certificate rule compares ship's registry expiration with current date

% Rule 10
Forall ?s ?d ?e (
  :HasValidCertificate(?s) :-
    And(?s#:Ship(:registryExpirationDate->?e)
%       phys:currentDate(?d)  % Uncomment for local date (deployment)
        :currentDate(?d)      % Uncomment for fixed date (reproducibility)
        phys:lessThanDate(?d ?e))
)
```

As in Rule 3, the "must" is not explicitly encoded here or later on. Rule 10 transits from the relational to the object-centered paradigm: The relational conclusion argument `?s` becomes, in the first condition conjunct, the OID of class

`:Ship` of a query frame with a `:registryExpirationDate` slot, whose filler is a date encoded as a Hornlog-expressiveness-level (constructor-)function application `phys:date(`*year month day*`)` – this depth-1 nesting could be easily eliminated, hence stays on what we refer to as the (Datalog-transformable) *near-Datalog* expressiveness level.[14] The second conjunct queries the current date in that encoding, optionally yielding the local date or a fixed date.[15] The third conjunct then checks `phys:lessThanDate` between these dates.

The **third subgroup** contains the complementary original Rules 8 resp. 7, which – for `:MeetsSafetyRequirementsUnloaded` of small resp. large ships – give two resp. three condition conjuncts, where the b) conjunct is shared (leading to the DAG structure of the decision model visualized in Fig. 1[16]):

8. A ship only meets the safety requirements for small unloaded ships if the ship complies with all of the following: a) the ship is categorized as small; b) the hold of the ship is clean.
7. A ship only meets the safety requirements for large unloaded ships if the ship complies with all of the following: a) the ship is categorized as large; b) the hold of the ship is clean; c) the hold of the ship is double hulled.

In PSOA RuleML/PS, this subgroup has the following formalization:

```
% Object-relational size-switched safety rules check status (small) or status and hull (large)

% Rule 8 (includes disjunct of original Rule 6)
Forall ?s ?h (
  :MeetsSafetyRequirementsUnloaded(?s) :-
    ?s#:Ship(:size->:small
             :hold->?h#:ShipHold(:status->:clean))
)

% Rule 7 (includes disjunct of original Rule 6)
Forall ?s ?h (
  :MeetsSafetyRequirementsUnloaded(?s) :-
    ?s#:Ship(:size->:large
             :hold->?h#:ShipHold(:status->:clean
                                 :hull->:double))
)
```

Rules 8 and 7 each includes a disjunct of the original Rule 6, which uses two intermediate predicates that are not needed for realizing the decision logic. Both

[14] The ternary function `phys:date` becomes unnecessary when dividing the slot `:registryExpirationDate` into three slots `:registryExpirationYear`, `:registryExpirationMonth`, and `:registryExpirationDay`.

[15] While for deployment the local date is computed by the `phys:currentDate` predicate from the physics library, imported via `http://psoa.ruleml.org/lib/phys.psoa`, for (test-suite) reproducibility a fixed date is looked up as a `:currentDate` ground fact.

[16] For layout reasons, in the DAG visualization the b) conjunct is shown as the second 'And' branch of Rule 8 but as the first 'And' branch of Rule 7.

Rules 8 and 7 transit from the relational to the object-centered paradigm with their frame conditions: The relational conclusion argument `?s` becomes the OID of class `:Ship` of a frame with `:size` and `:hold` slots, where the slot value of `:hold` is an embedded frame with OID `?h` of class `:ShipHold` and slots `:status` and `:clean`. The `:hold`-embedded `:ShipHold` frame – corresponding to an embedded `:ShipHold` function application – can be regarded as raising the expressiveness level to Hornlog, but – being only a depth-1 nesting – can be easily unnested, hence stays on the near-Datalog level. The nested frames of Rules 8 and 7 in PSOATransRun will be transformed into conjunctions via unnesting followed by slotribution. For example, this is the unnesting result for Rule 7's condition (the two `?h` occurrences are used for the referenced OID of the extracted frame moved to the top-level and for the referencing OID in the `:hold`-filler position of the main frame):

```
And(
  ?h#:ShipHold(:status->:clean :hull->:double)
  ?s#:Ship(:size->:large :hold->?h)
)
```

The subsequent slotribution result is as follows (all but one of the `?h` and `?s` occurrences just use the root class `Top`):

```
And(
  And(?h#:ShipHold  ?h#Top(:status->:clean)  ?h#Top(:hull->:double))
  And(?s#:Ship      ?s#Top(:size->:large)    ?s#Top(:hold->?h))
)
```

The **fourth subgroup** includes the similar original Rules 9 resp. 4, which determine whether a ship is small resp. large:

9. A ship must be categorized as small if the total length of the ship is less than 80 meters.
4. A ship must be categorized as large if the total length of the ship is at least 80 meters.

In PSOA RuleML/PS, this subgroup's formalization is:

```
% Object-centered (except for math) rules to get qualitative size by thresholding length

% Rule 9
Forall ?s ?l (
  ?s#Top(:size->:small) :-
    And(?s#:Ship(:totalLength->?l)
        math:lessThan(?l 80))
)

% Rule 4
Forall ?s ?l (
  ?s#Top(:size->:large) :-
    And(?s#:Ship(:totalLength->?l)
        math:greaterEq(?l 80))
)
```

Both rules are object-centered except for the relational `math:lessThan` and `math:greaterEq` calls in their second conjuncts.[17] Note that units of measure – here, "meter" and, in Rule 5, "mg dry weight per cm" – are omitted on this near-Datalog level of expressiveness, but could become Hornlog function applications in slot fillers – here, `:m(?l)` and, in Rule 5, `:mgDryWeightPerSqcm(?c)`.

The **fifth subgroup** consists of the original Rules 1 and 5, where the condition of Rule 1 negates the conclusion of Rule 5:

1. The hold of a ship must be considered clean if the hold does not contain remainders of cargo.

5. A ship's hold contains remainders of cargo if the residual cargo measurement is higher than 0.5 mg dry weight per cm^2.

In PSOA RuleML/PS, this subgroup is formalized as one combined Rule 1&5, thus:

```
% Object-centered (except for math) rule to get qualitative status by thresholding residual

% Rule 1&5 (combines Rule 1 and Rule 5)
Forall ?h ?c (
  ?h#Top(:status->:clean) :-
    And(?h#:ShipHold(:residualCargoMeasurement->?c)
        math:lessEq(?c 0.5))
)
```

By propagating the negation into Rule 5's condition, the negation of a `math:greaterThan` call is simplified to a `math:lessEq` call, so that negation is eliminated.[18] The resulting Rule 1&5 is again object-centered except for the relational `math:lessEq`.

4 Enrichment by Port Clearance Facts and Queries

Since the March 2016 DM Community Challenge has introduced only ship rules, we have developed ship facts for systematic testing of the rules in Section 3, as shown in the second half of Appendix A. Each ship fact is a frame having an OID `:ship`$k, k = 1, 2, ...$, and three slots, `:registryExpirationDate`, `:totalLength`, and `:hold`. The value of the `:hold` slot is an embedded frame having an OID `:h`$k, k = 1, 2, ...$, and two slots, `:residualCargoMeasurement` and `:hull`. Following is an example of a ship fact.

[17] The predicates having the `math:` prefix are defined in the imported mathematics library `http://psoa.ruleml.org/lib/math.psoa`. They are shortcuts for external built-in calls in PSOA.

[18] Coincidentally, while for the PSOATransRun 1.3 implementation negation is restricted to numeric disequality (`math:notEq`), in a future PSOATransRun a Negation-as-failure (Naf) primitive could be mapped to the underlying Prolog's Naf.

```
% Ship 7 -  Yes, hold clean and double-hulled
:ship7#:Ship(:registryExpirationDate->phys:date(2020 1 1)
             :totalLength->90
             :hold->:h7#:ShipHold(:residualCargoMeasurement->0.4
                                  :hull->:double))
```

The ship facts are covering all cases with qualitative slot-filler distinctions. Each ship fact comes with a comment on whether the ship instance should be allowed to enter a Dutch port, as of 2017-05-06, as well as the main reason.

In the following, we pose representative copy&paste-ready queries to the KB and demonstrate the answers obtained by PSOATransRun. The queries that answer Port Clearance questions such as for the DM Challenge are ground queries using the top-level predicate `:MayEnterDutchPortUnloaded` applied to specific ship instances, e.g. to `:ship1` and `:ship7`, as shown below.

```
:MayEnterDutchPortUnloaded(:ship1)
No

:MayEnterDutchPortUnloaded(:ship7)
Yes
```

We can also pose a generalized, symbolic-execution-style non-ground query to `:MayEnterDutchPortUnloaded`, using output variable `?w` to deduce ships that may enter a Dutch port. Such non-ground queries can make predicates behave non-deterministically: The multiple (here, four) `?w`-answer bindings are shown as full IRIs expanded from the ':'-prefixed `:ship`*k* abbreviations in the KB.[19]

```
:MayEnterDutchPortUnloaded(?w)
?w=<http://psoa.ruleml.org/usecases/PortClearance#ship14>
?w=<http://psoa.ruleml.org/usecases/PortClearance#ship2>
?w=<http://psoa.ruleml.org/usecases/PortClearance#ship12>
?w=<http://psoa.ruleml.org/usecases/PortClearance#ship7>
?w=<http://psoa.ruleml.org/usecases/PortClearance#ship4>
```

The supporting computations for these top-level decision-making questions can be revealed by specialized object-centered and relational queries. Let us thus explore queries over the KB in a bottom-up traversal of parts of the DAG in Fig. 1, explaining the answers obtained by PSOATransRun.

We start with the object-centered-query portion of this traversal. The following query asks whether `:h7` is a ship hold and it is clean, which can be proved by Rule 1&5 and the `:h7` frame embedded inside the `:ship7` fact.

```
:h7#:ShipHold(:status->:clean)    % Query centered on OID :h7
Yes
```

[19] For queries with variable-binding results, `Yes` answers will be understood. Since multiple variable-binding answers – like KB clauses – are semantically unordered, implementations such as PSOATransRun can choose any enumeration order.

Building on that, a query then asks whether the hold `?h` of `:ship7` is clean. This can be proved by first binding `?h` to (the expanded IRI of) the hold `:h7` of `:ship7` and then check whether `:h7` is clean using the previous (sub)query.

```
:ship7#:Ship(:hold->?h#:ShipHold(:status->:clean))    % Main OID is :ship7
?h=<http://psoa.ruleml.org/usecases/PortClearance#h7>
```

The next query asks whether `:ship7` is a large ship, which can be proved by Rule 4 using its `:totalLength` slot in the `:ship7` fact.

```
:ship7#:Ship(:size->:large)
Yes
```

An extended query then asks whether `:ship7` is a large ship and its hold is clean and double hulled. The size and hold status of `:ship7` have been proved by the previous two (sub)queries while the hold-hull information can be proved directly through the `:ship7` fact.

```
:ship7#:Ship(:size->:large :hold->?h#:ShipHold(:status->:clean :hull->:double))
?h=<http://psoa.ruleml.org/usecases/PortClearance#h7>
```

We now proceed to the relational-query portion of this traversal. The following query asks whether `:ship7` meets the safety requirements. It can be proved by Rule 7 and the previous (sub)query.

```
:MeetsSafetyRequirementsUnloaded(:ship7) % Query with unary safety relator
Yes
```

The sibling query in Fig. 1 asks whether `:ship7` has a valid certificate, which can be proved by Rule 10 based on its `:registryExpirationDate` slot in a fact.

```
:HasValidCertificate(:ship7)                      % As of 2017-05-06
Yes
```

The penultimate query of this traversal asks Whether `:ship7` complies with the requirements of the inspection for unloaded ships. It can be proved by Rule 3 based on the previous two (sub)queries.

```
:CompliesInspectionRequirementsUnloaded(:ship7)   % As of 2017-05-06
Yes
```

The top-level query `:MayEnterDutchPortUnloaded(:ship7)` can now be proved by Rule 2 and the previous (sub)query.

Next we explore some non-ground queries that can extract interesting content from the KB.

The following query asks for the `:size` of `:ship1`.

```
:ship1#:Ship(:size->?z)
?z=<http://psoa.ruleml.org/usecases/PortClearance#small>
```

The subsequent query asks for any large ship whose hold is clean. Here, the stand-alone '?' is an (anonymous) variable whose bindings are not needed.

```
?s#:Ship(:size->:large :hold->?#:ShipHold(:status->:clean))
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship7>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship13>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship10>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship14>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship6>
```

The next query asks for any ship that is large and has a valid certificate.

```
:HasValidCertificate(?s#:Ship(:size->:large))
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship5>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship14>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship6>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship9>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship13>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship7>
```

The final query asks for any ship that is small and meets the safety requirements for unloaded ships.

```
:MeetsSafetyRequirementsUnloaded(?s#:Ship(:size->:small))
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship4>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship1>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship12>
?s=<http://psoa.ruleml.org/usecases/PortClearance#ship2>
```

5 Conclusions

In this paper, we have demonstrated the formalization of moderately controlled English rules and a corresponding decision model as part of a logical KB. This prepares the automation of such formalizations as well as enabling formal query, proof, analysis, and translation. The Port Clearance Rules have already progressed from a DM Challenge to a use case for comparing DM systems. With their formalization in PSOA RuleML, they are also turning into a use case for logical (e.g., object-relational) systems and for the interoperation between both kinds of systems. This PSOA RuleML formalization of a DM Challenge, along with Semantic DMN [7], can thus initiate Semantic Decision Rules.

Since the original Port Clearance Rules were developed independently, the use case provides extra evidence that PSOA RuleML is well-suited to capture real-world problems and PSOATransRun is well-suited for KB development. While this paper uses integrated object-relational modeling, the original English rules are amenable also to purely object-centered modeling and to purely relational modeling, where these paradigms are bridged within the PSOA language [4]. Generalizations for Hornlog, non-deterministic, and non-ground-queried KBs for advanced modeling have already been indicated earlier. Here are examples for the other two generalizations: *Recursion* can realize the transitive closure, `:navigates`, over a `:connects` base predicate, for ships finding an inspection pier.[20] *Subpredicating*, via PSOA's '`##`' infix, can taxonomically differentiate the class `:Ship` into `:ShipUnloaded` vs. `:ShipLoaded` for precise semantic modeling.

Future work includes the interchange of presentation-syntax and (XML-) serialization formats of DM systems such as OpenRules[21] – e.g. via the OMG Decision Model and Notation (DMN)[22] – with PSOA RuleML. For XML-based interchange, PSOA RuleML already has a preliminary serialization, which will be formally defined in Relax NG.[23] One direction of interoperation can build on the translator from DMN decision tables to RuleML [8]. While Fig. 1 visualizes rules only, a proof-explanation facility could be added to the PSOATransRun environment which provides visualization, presentation, and serialization formats for queries reduced, via rules, to facts, building on, e.g., Grailog[24] and VProof$_\mathrm{H}$ [9]. Because of their relevance to (e.g., harbor-security) laws, extensions of the Port Clearance Rules – including for loaded ships – should be of interest, e.g. as part of legal-informatics efforts such as OASIS LegalRuleML[25] and Stanford CodeX[26].

6 Acknowledgements

We thank Jacob Feldman for pointing us to the DM Challenges, for hints about the evolution of The Game of Rules / Port Clearance Rules, and for encouraging better integration of business rule products and semantic rule products. Moreover, we thank the organizers of the International Rule Challenge 2017 for holding this event, and its reviewers for valuable comments.

[20] Further exemplifying non-determinism, in case an inspection pier turns out to be unavailable, backtracking as in the Prolog-targeting PSOATransRun instantiation PSOATransRun[PSOA2Prolog,XSBProlog] can search for other inspection piers.

[21] `http://openrules.com`

[22] `http://www.omg.org/spec/DMN`

[23] `http://wiki.ruleml.org/index.php/PSOA_RuleML#Syntax`

[24] `http://wiki.ruleml.org/index.php/Grailog`

[25] `https://www.oasis-open.org/committees/legalruleml`

[26] `https://law.stanford.edu/codex-the-stanford-center-for-legal-informatics/`

References

1. Tylkowski, M., Müller, M.: Experiences Using Deliberation RuleML 1.01 as Rule Interchange Language: Do Rulebases Need an External Vocabulary? In Patkos, T., Wyner, A.Z., Giurca, A., eds.: Proceedings of the RuleML 2014 Challenge and the RuleML 2014 Doctoral Consortium hosted by the 8th International Web Rule Symposium, Challenge+DC@RuleML 2014, Prague, Czech Republic, August 18-20, 2014. Volume 1211 of CEUR Workshop Proceedings., CEUR-WS.org (2014)
2. Spreeuwenberg, S., Bouvy, C., Zoet, M.: Het spel met de regels. Uitgave van BRPN, ISBN 978-90-815568-2-8 (2013)
3. Boley, H.: A RIF-Style Semantics for RuleML-Integrated Positional-Slotted, Object-Applicative Rules. In: Proc. 5th International Symposium on Rules: Research Based and Industry Focused (RuleML-2011 Europe), Barcelona, Spain. Lecture Notes in Computer Science, Springer (July 2011) 194–211
4. Boley, H.: PSOA RuleML: Integrated Object-Relational Data and Rules. In Faber, W., Paschke, A., eds.: Reasoning Web. Web Logic Rules (RuleML 2015) - 11th Int'l Summer School 2015, Berlin, Germany, July 31- August 4, 2015, Tutorial Lectures. Volume 9203 of LNCS., Springer (2015)
5. Zou, G., Boley, H.: PSOA2Prolog: Object-Relational Rule Interoperation and Implementation by Translation from PSOA RuleML to ISO Prolog. In: Proc. 9th International Web Rule Symposium (RuleML 2015), Berlin, Germany. Lecture Notes in Computer Science, Springer (August 2015)
6. Zou, G., Boley, H.: Minimal Objectification and Maximal Unnesting in PSOA RuleML. In Alferes, J.J., Bertossi, L.E., Governatori, G., Fodor, P., Roman, D., eds.: Rule Technologies. Research, Tools, and Applications - 10th International Symposium, RuleML 2016, Stony Brook, NY, USA, July 6-9, 2016. Proceedings. Volume 9718 of Lecture Notes in Computer Science., Springer (2016) 130–147
7. Calvanese, D., Dumas, M., Maggi, F.M., Montali, M.: Semantic DMN: Formalizing Decision Models with Domain Knowledge. In: Proc. International Joint Conference on Rules and Reasoning (RuleML+RR 2017), London, UK. Volume 10364 of Lecture Notes in Computer Science., Springer (July 2017)
8. Paschke, A., Könnecke, S.: RuleML - DMN Translator. In Athan, T., Giurca, A., Grütter, R., Proctor, M., Teymourian, K., Woensel, W.V., eds.: Supplementary Proceedings of the RuleML 2016 Challenge, Doctoral Consortium and Industry Track hosted by the 10th International Web Rule Symposium, RuleML 2016, New York, USA, July 6-9, 2016. Volume 1620 of CEUR Workshop Proceedings., CEUR-WS.org (2016)
9. Kontopoulos, E., Bassiliades, N., Antoniou, G.: Visualizing Semantic Web Proofs of Defeasible Logic in the DR-DEVICE System. Knowledge-Based Systems **24**(3) (2011) 406–419

A Port Clearance KB in PSOA RuleML

```
Document (

  Prefix(: <http://psoa.ruleml.org/usecases/PortClearance#>)
  Prefix(math: <http://psoa.ruleml.org/lib/math#>)
  Prefix(phys: <http://psoa.ruleml.org/lib/phys#>)

  Import(<http://psoa.ruleml.org/lib/math.psoa>)
  Import(<http://psoa.ruleml.org/lib/phys.psoa>)
```

```
Group (

  % Port Clearance Knowledge Base
  % Version: 2017-06-08

  % Copy&paste-ready KB file in presentation syntax (PSOA RuleML/PS):
  % http://psoa.ruleml.org/usecases/PortClearance/PortClearance.psoa
  % To be complemented by KB file in serialization syntax (PSOA RuleML/XML):
  % http://psoa.ruleml.org/usecases/PortClearance/PortClearance.ruleml

  % PSOA RuleML/PS can be directly used as a PSOATransRun input file:
  % http://wiki.ruleml.org/index.php/PSOA_RuleML#PSOATransRun (Current Release)

  % Reordering, subgrouping, and explaining the rules from
  % https://dmcommunity.org/challenge/challenge-march-2016/

  % Main relational rule invokes inspection rule for certificate And safety

  % Rule 2
  Forall ?s (
    :MayEnterDutchPortUnloaded(?s) :-
      :CompliesInspectionRequirementsUnloaded(?s)
  )

  % Rule 3
  Forall ?s (
    :CompliesInspectionRequirementsUnloaded(?s) :-
      And(:HasValidCertificate(?s)
          :MeetsSafetyRequirementsUnloaded(?s))
  )

  % Object-relational certificate rule compares ship's registry expiration with current date

  % Rule 10
  Forall ?s ?d ?e (
    :HasValidCertificate(?s) :-
      And(?s#:Ship(:registryExpirationDate->?e)
  %       phys:currentDate(?d)  % Uncomment for local date (deployment)
          :currentDate(?d)      % Uncomment for fixed date (reproducibility)
          phys:lessThanDate(?d ?e))
  )

  % Object-relational size-switched safety rules check status (small) or status and hull (large)

  % Rule 8 (includes disjunct of original Rule 6)
  Forall ?s ?h (
    :MeetsSafetyRequirementsUnloaded(?s) :-
      ?s#:Ship(:size->:small
               :hold->?h#:ShipHold(:status->:clean))
  )

  % Rule 7 (includes disjunct of original Rule 6)
  Forall ?s ?h (
    :MeetsSafetyRequirementsUnloaded(?s) :-
      ?s#:Ship(:size->:large
               :hold->?h#:ShipHold(:status->:clean
                                   :hull->:double))
  )

  % Object-centered (except for math) rules to get qualitative size by thresholding length

  % Rule 9
  Forall ?s ?l (
    ?s#Top(:size->:small) :-
      And(?s#:Ship(:totalLength->?l)
          math:lessThan(?l 80))
  )
```

```
% Rule 4
Forall ?s ?l (
  ?s#Top(:size->:large) :-
    And(?s#:Ship(:totalLength->?l)
        math:greaterEq(?l 80))
)

% Object-centered (except for math) rule to get qualitative status by thresholding residual

% Rule 1&5 (combines Rule 1 and Rule 5)
Forall ?h ?c (
  ?h#Top(:status->:clean) :-
    And(?h#:ShipHold(:residualCargoMeasurement->?c)
        math:lessEq(?c 0.5))
)


:currentDate(phys:date(2017 5 6))   % Uncomment for fixed date (reproducibility)


% Ship facts (No or Yes refer to answers for queries, as of 2017-05-06, with :ship1, :ship2, ... as arguments)

% Facts covering all cases with qualitative slot-filler distinctions
% Explanatory comments for Yes answers focus on the most relevant slots

% Distinction for :registryExpirationDate

% Ship 1 - No, registry has expired
:ship1#:Ship(:registryExpirationDate->phys:date(2017 5 1)
             :totalLength->20
             :hold->:h1#:ShipHold(:residualCargoMeasurement->0.2
                                  :hull->:single))

% Ship 2 - Yes, registry is valid
:ship2#:Ship(:registryExpirationDate->phys:date(2017 10 1)
             :totalLength->20
             :hold->:h2#:ShipHold(:residualCargoMeasurement->0.2
                                  :hull->:single))

% Distinction for :residualCargoMeasurement

% Ship 3 - No, hold not clean
:ship3#:Ship(:registryExpirationDate->phys:date(2020 1 1)
             :totalLength->70
             :hold->:h3#:ShipHold(:residualCargoMeasurement->0.6
                                  :hull->:single))

% Ship 4 - Yes, hold clean (qualitatively the same as for Ship 2)
:ship4#:Ship(:registryExpirationDate->phys:date(2020 1 1 )
             :totalLength->70
             :hold->:h4#:ShipHold(:residualCargoMeasurement->0.4
                                  :hull->:single))

% Distinctions for :residualCargoMeasurement and :hull

% Ship 5 - No, hold not clean
:ship5#:Ship(:registryExpirationDate->phys:date(2020 1 1)
             :totalLength->90
             :hold->:h5#:ShipHold(:residualCargoMeasurement->0.6
                                  :hull->:double))

% Ship 6 - No, size large yet hold single-hulled
:ship6#:Ship(:registryExpirationDate->phys:date(2020 1 1)
             :totalLength->90
             :hold->:h6#:ShipHold(:residualCargoMeasurement->0.4
                                  :hull->:single))
    % Ship 7 - Yes, hold clean and double-hulled
    :ship7#:Ship(:registryExpirationDate->phys:date(2020 1 1)
                 :totalLength->90
                 :hold->:h7#:ShipHold(:residualCargoMeasurement->0.4
                                      :hull->:double))


    % Facts with multiple reasons for No or Yes

    % Three reasons for No

    % Ship 8 - No, registry expired, hold not clean, and size large yet hold single-hulled
    :ship8#:Ship(:registryExpirationDate->phys:date(2017 1 1)
                 :totalLength->90
                 :hold->:h8#:ShipHold(:residualCargoMeasurement->0.9
                                      :hull->:single))

    % Two reasons for No

    % Ship 9 - No, hold not clean and size large yet hold single-hulled
    :ship9#:Ship(:registryExpirationDate->phys:date(2018 1 1)
                 :totalLength->90
                 :hold->:h9#:ShipHold(:residualCargoMeasurement->0.9
                                      :hull->:single))

    % Ship 10 - No, registry expired and size large yet hold single-hulled
    :ship10#:Ship(:registryExpirationDate->phys:date(2017 1 1)
                  :totalLength->90
                  :hold->:h10#:ShipHold(:residualCargoMeasurement->0.2
                                        :hull->:single))

    % Ship 11 - No, registry expired and hold not clean
    :ship11#:Ship(:registryExpirationDate->phys:date(2017 1 1)
                  :totalLength->90
                  :hold->:h11#:ShipHold(:residualCargoMeasurement->0.9
                                        :hull->:double))

    % Two reasons for Yes

    % Ship 12 - Yes, size small nevertheless hold double-hulled
    :ship12#:Ship(:registryExpirationDate->phys:date(2020 1 1)
                  :totalLength->60
                  :hold->:h12#:ShipHold(:residualCargoMeasurement->0.1
                                        :hull->:double))


    % Facts probing special cases

    % Ship 13 - No, large ship must have some (a double) hull
    :ship13#:Ship(:registryExpirationDate->phys:date(2020 1 1)
                  :totalLength->120
                  :hold->:h13#:ShipHold(:residualCargoMeasurement->0.2))

    % Ship 14 - Yes, date, length, and measurement are at the threshold
    :ship14#:Ship(:registryExpirationDate->phys:date(2017 5 7)
                  :totalLength->80
                  :hold->:h14#:ShipHold(:residualCargoMeasurement->0.5
                                        :hull->:double))

  )

)
```