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

11th International Rule Challenge, RuleML+RR 2017
July 12-15, 2017

Outline

Outline

Background

- Decision Management (DM) Community has been running Challenges about decision modeling problems since 2014
- The DM Challenge of March 2016 consisted of creating decision models from the structured text of English Port Clearance Rules, available online

Port Clearance Rules

Jacob Feldman *pointed us to this DM Challenge on The Game of Rules / Port Clearance Rules*

- Decide whether **a ship can enter** a Dutch port on a certain date
- Ten English rules inspired by the international Ship and Port Facility Security Code, originally developed by **Silvie Spreeuwenberg et al.** for "The Game of Rules"
- The English of each one of these independently given rules is **moderately controlled**, some having a structured 'if' part
- We **formalized** the rules in PSOA RuleML, added facts (data) directly in PSOA, **queried** result in PSOATransRun, and propose **generalized** decision models

Generalized Decision Models: Non-ground, Non-deterministic, Horn Logic

Ground term No variables (inside)

Non-ground term At least one variable (inside)

Deterministic No predicate occurs more than once in rule conclusions

Non-deterministic At least one predicate occurs more than once in rule conclusions

Datalog No (constructor) function applications

Horn logic At least one (constructor) function application

PSOA RuleML in a Nutshell (1)

- Novel object/frame-relational rule language generalizing **relationships** (e.g., in LP) and **frames** (e.g., in RDF) into **p**ositional-**s**lotted **o**bject-**a**pplicative (**psoa**) terms

  Single-tuple case, where "#" means "member of":

  Oidless : $f(t_1 \ldots t_n \;\; p_1 \texttt{->} v_1 \ldots p_k \texttt{->} v_k)$
  Oidful : $o\#f(t_1 \ldots t_n \;\; p_1 \texttt{->} v_1 \ldots p_k \texttt{->} v_k)$

- Oidless psoa terms are interpreted as **atoms** (i.e., predicate applications) on the top-level and as **expressions** (i.e., function applications) when embedded in another term
- Oidful psoa terms are interpreted as atoms both on the top-level and when embedded
- Embedded oidful atoms can be extracted via **unnesting**

PSOA RuleML in a Nutshell (2)

- Special cases of psoa atoms

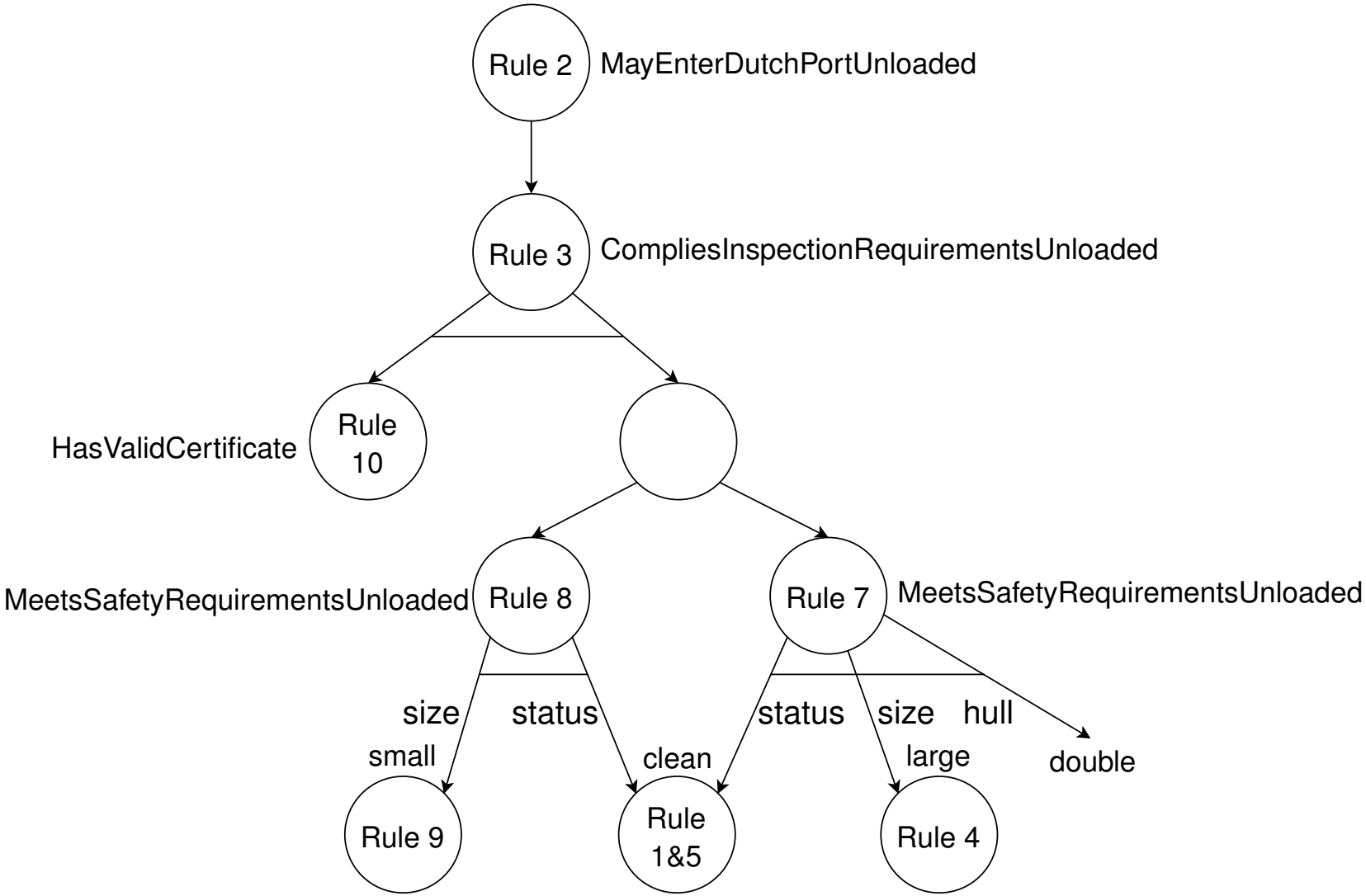
  Relationships: $\texttt{f(t}_1 \ldots \texttt{t}_n\texttt{)}$
  Frames: $\texttt{o\#f(p}_1\texttt{->v}_1 \ldots \texttt{p}_k\texttt{->v}_k\texttt{)}$

- PSOA RuleML syntax
  - Constants include `Top`, numbers, strings, and Internationalized Resource Identifiers (IRIs)
  - A full IRI, e.g., `<http://ex.org/a>`, can be abbreviated using a namespace prefix ending with '`:`', e.g., `:a`, if the KB contains a declaration `Prefix(: <http://ex.org/>)` for the prefix '`:`'
  - Variables in PSOA are '`?`'-prefixed names, e.g., `?x`
  - A PSOA KB consists of clauses, mostly as ground facts (psoa atoms) and non-ground rules (*conclusion* `:-` *condition* with `Forall` wrappers)
- Reference implementation:
  Prolog instantiation of PSOATransRun

Outline

Visualization of PSOA's Formal Decision Model (1)

- An object-relational And-Or DAG with rule names as nodes and conclusion predicates as side labels of nodes



Visualization of PSOA's Formal Decision Model (2)

- For the not side-labeled nodes, the root-class predicate `Top` is understood, while slot names are shown as 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 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 in that the upper part running to the conclusions of Rules 8 and 7 involves unary relations applied to ships while the lower part involves frames with ship OIDs described by slots

Subgroup 1 (1)

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

```
% 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))
)
```

Subgroup 1 (2)

- Both rules are relational, on the Datalog level of expressiveness
- `:Ship`-type test for `?s` is postponed to the later object-centered rules, where `:Ship` becomes a class

Subgroup 2 (1)

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.

```
% 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))
)
```

Subgroup 2 (2)

- Rule 10 transits from the relational to the object-centered paradigm
- Relational conclusion argument `?s`, in the first condition conjunct, becomes the OID of class `:Ship` of a frame
- Filler of a `:registryExpirationDate` slot is a date encoded as a Hornlog-expressiveness-level function application `phys:date(`*year month day*`)` – this depth-1 nesting could be easily eliminated, hence stays on the (Datalog-transformable) *near-Datalog* expressiveness level
- Second conjunct queries the current date in the above encoding, optionally yielding the local date for deployment (using `phys:currentDate` from the physics library) or a fixed date (using `:currentDate`) for reproducibility
- Third conjunct checks `phys:lessThanDate` between these dates

Subgroup 3 (1)

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.

```
% 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))
)
```

Subgroup 3 (2)

- Rules 8 and 7 each includes a disjunct of the original Rule 6, which uses intermediate predicates that are not needed for realizing the decision logic
- Both rules again transit from the relational to the object-centered paradigm: relational conclusion argument `?s` becomes the OID of class `:Ship` of condition frame
- Slot filler of `:hold` is an embedded 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

Unnesting and Slotribution

Condition of Rule 7:

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

After unnesting:

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

After slotribution:

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

Subgroup 4

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.

```
% 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))
)
```

`math:` predicates are from mathematics library

Subgroup 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.

```
% 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))
)
```

Negation is eliminated by propagation into Rule 5's condition, where the negated `math:greaterThan` call is simplified to a `math:lessEq` call

Outline

Facts

- Since the DM Challenge has introduced only ship rules, we have developed ship facts for systematic testing of rules using PSOATransRun[PSOA2Prolog,XSBProlog] 1.3
- Example of ship facts

```
% 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 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))
```

Queries that Answer DM Challenge Questions

- Queries for Port Clearance questions are ground, using top-level predicate `:MayEnterDutchPortUnloaded` applied to specific ship instances

```
:MayEnterDutchPortUnloaded(:ship1)
No

:MayEnterDutchPortUnloaded(:ship7)
Yes
```

- Generalized non-ground query can also be posed

```
: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>
```

Supporting Computation via Specialized Queries (1)

Whether `:h7` is a ship hold and is clean (proved by Rule 1&5 and `:h7` frame embedded inside `:ship7` fact)

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

Whether hold `?h` of `:ship7` is clean (proved by Rule 4 using its `:totalLength` slot in `:ship7` fact)

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

Whether `:ship7` is a large ship (proved by Rule 4 using its `:totalLength` slot in `:ship7` fact)

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

Supporting Computation via Specialized Queries (2)

Whether `:ship7` is a large ship and its hold is clean and double hulled (proved by previous two (sub)queries and `:ship7` fact)

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

Whether `:ship7` meets safety requirements (proved by Rule 7 and previous (sub)query)

```
:MeetsSafetyRequirementsUnloaded(:ship7)
Yes
```

Supporting Computation via Specialized Queries (3)

Whether `:ship7` has valid certificate (proved by Rule 10 based on its `:registryExpirationDate` slot in a fact)

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

Whether `:ship7` complies with requirements of inspection for unloaded ships (proved by Rule 3 based on previous two (sub)queries)

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

Top-level query `:MayEnterDutchPortUnloaded(:ship7)` can now be proved by Rule 2 and previous (sub)query

Non-ground Queries Extracting Interesting Content (1)

A ship's size

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

Any ship that is large and has hold that is clean

```
?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>
```

Non-ground Queries Extracting Interesting Content (2)

Any ship that is large and has 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>
```

Any ship that is small and meets 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>
```

Outline

Conclusions

- Demonstrated formalization of moderately controlled English rules as a decision model being part of a logical KB enabling formal query, proof, analysis, and translation
- English rules formalized as (deontically contextualized) near-Datalog, non-recursive, near-deterministic, ground-queried, and non-subpredicating KB rules
- KB rules complemented by PSOA facts queried in PSOATransRun for decision-making
- Provides extra evidence that PSOA RuleML is well-suited to capture real-world problems and PSOATransRun is well-suited for KB development
- While this KB 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 PSOA

Future Work

- Further generalizations of the use case, e.g. adding *recursion* and *subpredicating*
- Interchange of presentation-syntax and (XML-)serialization formats of DM systems such as OpenRules with PSOA RuleML
- Based on the preliminary serialization of PSOA RuleML, formally define the schema in Relax NG and develop an XML serialization of Port Clearance KB
- Proof-explanation facility could be added to PSOATransRun, providing visualization, presentation, and serialization formats for queries reduced to facts
- 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 and Stanford CodeX