A Hands-on PSOA RuleML Tutorial

Relationship & Framepoint Facts and Rules

Theodoros Mitsikas
National Technical University of Athens | RuleML
mitsikas@central.ntua.gr

2nd Workshop on Rules: Logic and Applications, NTUA, Athens, Greece, 16-17 December 2019

PSOA RuleML

- An object-relational Web rule language
- Integrates various atomic formulas (atoms), from Prolog-like relationships to F-logic-like frames, and introduces new ones, in the systematics of **p**ositional-**s**lotted **o**bject-**a**pplicative (psoa) atoms

- Use cases:
  - Port Clearance Rules, Medical Devices Rules, Air Traffic Control KB, ...

Relationships: Oidless, Single-Tupled, Dependent Atoms

“A purchase of Fido by John from Mary”

`purchase(John Mary Fido)` PSOA RuleML

`purchase(john,mary,fido)` Prolog

PSOA RuleML:

- Predicate arguments are separated by whitespace, not by comma
- `John` etc. are individual constants (variables will be denoted by the ‘?’ prefix)
  - constants include `Top` (the root of the predicate hierarchy), numbers, strings, and Internationalized Resource Identifiers (IRIs)

Relationships: Oidless, Single-Tupled, Dependent Atoms

“A purchase of Fido by John from Mary”

`purchase(John Mary Fido)` PSOA RuleML

- the order of the arguments is significant
- we can have n-ary relationships (here: n=3)
- the argument tuple of a relationship is predicate-dependent

Logic Variables

Logic variables are indicated by a '?' prefix:

```
purchase(?b ?s ?i)
```

(Fact) Queries

Ground (i.e., with no variables) queries:

```
purchase(John Mary Fido)     % Yes
```

Non-ground (i.e., with at least one variable) queries:

```
purchase(?b ?s ?i) % ?b=_John ?s=_Mary ?i=_Fido (explicit local constants)

purchase(?b ?s)    % No (there can be no bindings)

?p(John Mary Fido) % ?p=_purchase (predicate variable)
```

Framepoints: Oidful, Slotted, Independent Atoms

“A purchase, transaction200, with buyer John, seller Mary, and item Fido”

```
transaction200#purchase(buyer->John seller->Mary item->Fido)
```

- uses slot names ('explicit roles') `buyer`, `seller`, and `item`
- hash infix, "`#`", types the Object Identifier (OID) `transaction200` with its predicate (i.e., indicates membership)
- independent-arrow infix, "`->`", pairs a predicate-independent slot name with its filler
- ordering between slots is not important
- framepoint atoms build a Directed Labeled Graph with predicate-typed nodes

Tuple/Slot-combining PSOA Atoms

The atom

```
transaction300#purchase(John Mary item->Fido)
```

is oidful, tupled+slotted

(Ground) Rule

"John is liable for Fido if John purchases Fido from Mary"

```
liability(John Fido) :-
  purchase(John Mary Fido)
```

(Non-ground) Rule

"A buyer is liable for an item if the buyer purchases the item from a seller"

```
Forall ?b ?s ?i (
 liability(?b ?i) :-
   purchase(?b ?s ?i)
)
```

(Non-ground) Rule

```
Forall ?b ?s ?i ?t (
 liabilityID(?t)#liability(bearer->?b item->?i) :-
   ?t#purchase(buyer->?b seller->?s item->?i)
)
```

PSOA Hybrid Rules

Relationship conclusion, framepoint condition non-ground rule:

```
Forall ?b ?s ?i ?t (
 liability(?b ?i) :-
   ?t#purchase(buyer->?b seller->?s item->?i)
)
```

Conjunctions, Disjunctions

```
RuleML(
 Assert(

  Forall ?Hu ?Wi ?Ch (
   family(husb->?Hu wife->?Wi child->?Ch) :-  % Head (conclusion) is OIDless slotted
    And(
        % Body (condition) conjoins two 2-ary relationships
        married(?Hu ?Wi) kid(?Wi ?Ch))
    )
  married(Joe Sue)     % 2-ary married relationship fact
  kid(Sue Pete)        % 2-ary kid relationship fact
 )
)
```

Deductive PSOA Queries

```
% KB
transaction200#purchase(
            buyer->John seller->Mary item->Fido)

Forall ?b ?s ?i ?t (
 liabilityID(?t)#liability(bearer->?b item->?i) :-
              ?t#purchase(buyer->?b seller->?s
              item->?i)
)
```

```
liability(bearer->?b item->?i)     % ?b=_John ?i=_Fido

?o#liability(bearer->?b item->?i) % Extra binding:
                                    ?o=_liabilityID(_transaction200)
```

Advanced PSOA Queries

```
% KB
transaction200#purchase(
            buyer->John seller->Mary item->Fido)

Forall ?b ?s ?i ?t (
 liabilityID(?t)#liability(bearer->?b item->?i) :-
              ?t#purchase(buyer->?b seller->?s
              item->?i)
)
```

```
transaction200#purchase(buyer->John seller->Mary)     % Yes
Or( ?t#purchase(buyer->Theodore) ?t#purchase(buyer->John) )
                                              %?t=_transaction200

transaction200#purchase                  % Yes
transaction200#Top(buyer->John)          % Yes
liabilityID(?t)#purchase(buyer->?b) % No (“buyer” slot describes
                                           OID constant transaction200)
```

Live Demo

- Using PSOATransRun: the reference PSOA RuleML reasoner
- PSOATransRun maps knowledge bases and queries
  in PSOA RuleML presentation syntax to TPTP or Prolog
  - runtime options allow us to see, e.g., the Prolog translation!

Some (Further) Advanced Features of PSOA RuleML and PSOATransRun

- Built-in mathematical predicates and functions, libraries
- Dependent slots and independent tuples
- Subclasses
- Static translation
- N3/Turtle import
- Graph modeling

Further Reading

PSOA RuleML Wiki page:

- http://wiki.ruleml.org/index.php/PSOA_RuleML#Presentation_Preview
- http://wiki.ruleml.org/index.php/PSOA_RuleML#Examples
- http://wiki.ruleml.org/index.php/PSOA_RuleML#References

Learn PSOA RuleML - a resource page on PSOA syntax, (query) semantics, and tools: http://psoa.ruleml.org/learn

Join the Open-source Project

- Develop use cases
  wiki.ruleml.org/index.php/PSOA_RuleML#Use_Cases

- Contribute to PSOATransRun development
  wiki.ruleml.org/index.php/PSOATransRun_Development_Agenda