A Hands-on PSOA RuleML Tutorial

Relationship & Framepoint Facts and Rules

Theodoros Mitsikas
National Technical University of Athens | RuleML
mitsikas[AT]central[DOT]ntua[DOT]gr

4th International Joint Conference on Rules and Reasoning,
Virtual, June 29th - July 1st, 2020

Updated for Skype “PSOA RuleML Topics” Presentation on: September 8th, 2020
Available at: http://ruleml.org/talks/TheodorosMitsikas-PSOARuleMLTutorial-RuleMLRR2020.pdf
Upgraded from: 2nd Workshop on Rules: Logic and Applications (RulesLogApps 2019)

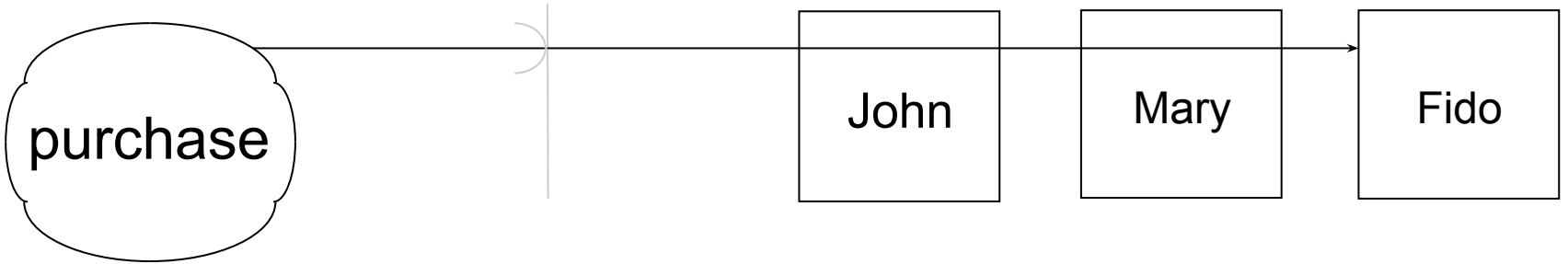
PSOA RuleML

- A **graph/object-relational** Web rule language for asserting plus querying **node/object**-describing *key-value pairs* and **relational**-style table rows in a uniform manner: generalizes Knowledge Graphs to PSOA KGs (http://wiki.ruleml.org/index.php/PSOA_KGs:_RuleML_Technical_Group_on_PSOA_Knowledge_Graphs)
- Integrates spectrum of atomic formulas (atoms), arriving at F-logic-like **framepoints** from Prolog-like **relationships**, with blended ones in between, in a systematics of **p**ositional-**s**lotted **o**bject-**a**pplicative (psoa) atoms
- Use cases of Knowledge Bases (KBs):
  - Port Clearance Rules, Medical Devices Rules, Air Traffic Control KB, ...

Relationships: From English & RCL to Visualization (Grailog)

English: "A purchase involving John, Mary, and Fido"
RCL (RuleML Controlled Language): "a purchase of John, Mary, thru Fido"

Grailog (1 directed hyperarc applying predicate to 3-argument tuple):



Relationships: Oidless, Single-Tupled, Dependent Atoms

`purchase(John Mary Fido)` PSOA RuleML

`purchase(john,mary,fido)` Prolog

PSOA RuleML:

- Predicate arguments are separated by whitespace, not commas
- `John`, ... are individual constants (variables denoted by "?" prefix)
  - constants also include numbers, strings, Internationalized Resource Identifiers (IRIs), and `Top` (the root of the predicate hierarchy)

Relationships: Oidless, Single-Tupled, Dependent Atoms

`purchase(John Mary Fido)` PSOA RuleML

- we don't need to provide an Object IDentifier (OID)
- the order of the arguments matters in one (implicit) n-tuple
- used for n-ary relationships (here: n=3)
- the argument tuple of a relationship is predicate-dependent (predicate-scope-sensitive)
- this talk focuses function-free (Datalog-like) expressiveness

(Fact-retrieving) Queries

Ground queries (no variables):

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

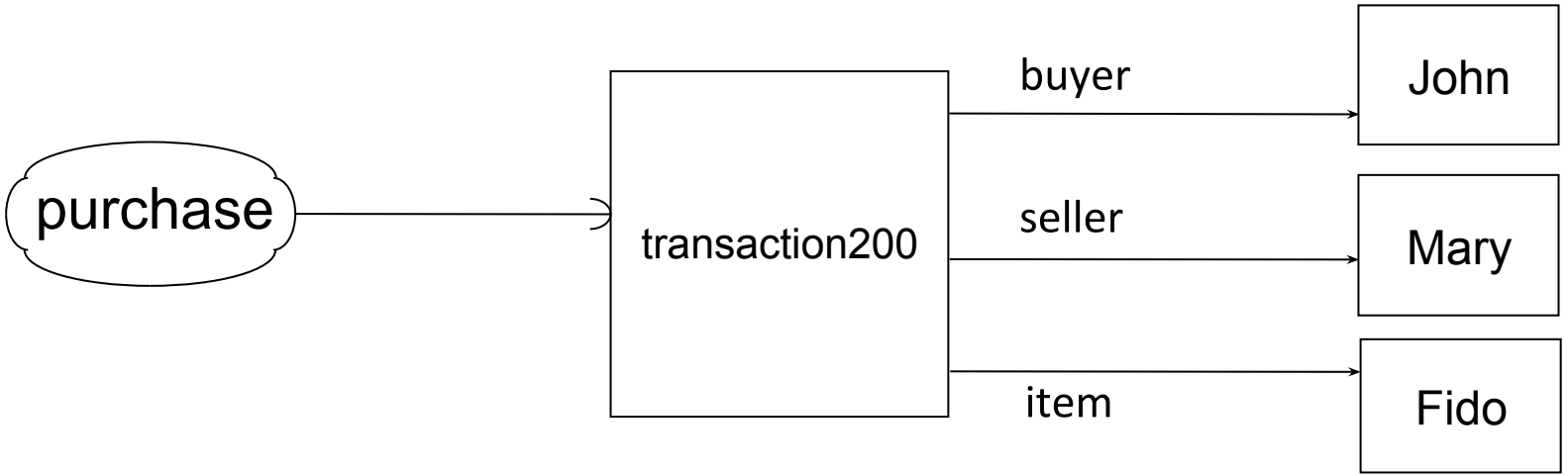
Non-ground queries (at least one "?" variable, bound to "_"-prefixed explicit local constants):

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

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

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

Framepoints: From Engl.-like RCL to Visualization (Grailog)

"transaction200, a purchase with buyer John, seller Mary, plus item Fido"

Grailog (OID with 3 slot-name-labeled arcs and 1 "has element" arc):



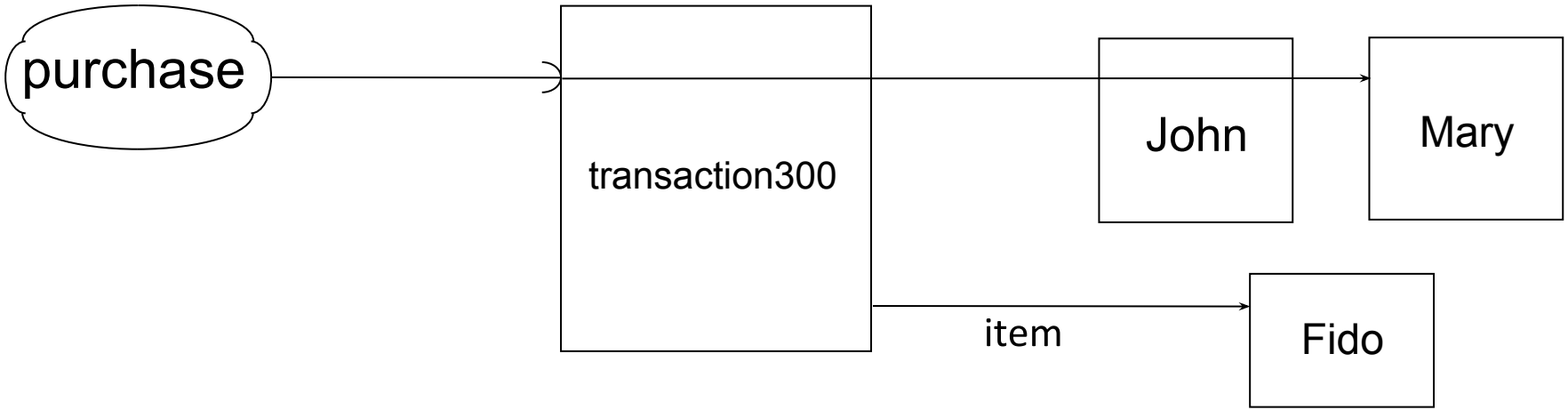
Framepoints: Oidful, Slotted, Independent Atoms

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

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

Atoms Blending Relationships and Framepoints:
From Keyword (RCL) to Visual (Grailog) Syntax

"transaction300, a purchase of John thru Mary, with item Fido"



Tuple/Slot-combining PSOA Atoms in Systematics from Relationships to Framepoints

The blended atom

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

is oidful (`transaction300`) as well as dependently tupled (`John Mary`) and independently slotted (`item->Fido`)

(Ground) Rule over Relationships

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

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

As in Prolog, the " :-" (read: "if") indicates rules

(Non-ground) Rule over Relationships

"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)
)
```

Hybrid Rule over Relationships and Framepoints

Non-ground relationship-conclusion framepoint-condition rule:

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

Rule over Framepoints with Coupled OIDs

```
Forall ?b ?s ?i ?t (

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

(Rule-applying) Oidless/Oidful Queries

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

Forall ?b ?s ?i ?t (
 dutyFunc(?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=_dutyFunc(_transaction200)
```

Explore More Oidful Queries

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

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

```
transaction200#purchase(buyer->?b seller->?s) % ?b=_John ?s=_Mary
transaction200#Top(buyer->?b)                 % ?b=_John
transaction200#purchase                       % Yes
dutyFunc(?t)#liability(?r->John)     % ?r=_bearer ?t=_transaction200
Or( ?t#purchase(buyer->Theodore) ?t#purchase(buyer->John) )
                                     %            ?t=_transaction200
```

Live Demo

- Using PSOATransRun: the reference PSOA RuleML reasoner
- PSOATransRun translates graph/object-relational KBs and queries from PSOA RuleML presentation syntax to relational languages

- Available online: https://psoademo-chatty-cat.eu-gb.mybluemix.net
  - ‘ready-to-use’ demo version translating to TPTP
- Available for download: http://wiki.ruleml.org/index.php/PSOA_RuleML#Prolog_Instantiation
  - runtime options allow to see, e.g., the TPTP or Prolog translation results!

Some Advanced Features of PSOA RuleML and PSOATransRun

- Built-ins and libraries for math & physics predicates/functions
- Dependent slots (e.g., `item+>Fido`) as well as explicit dependent and independent tuples (`+[...]` and `–[...]`)
- Subpredicates (e.g., `bargain##purchase`)
- Embedded (oidless/oidful) atoms
- RDF import (N3/Turtle)
- Negation-as-failure (Naf) for relationships; explored for framepoints, …

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