Perspectival Knowledge in PSOA RuleML 1.0: Representation, Model Theory, and Translation

Harold Boley, Gen Zou

Faculty of Computer Science, University of New Brunswick, Canada

Standards Session, RuleML+RR 2017
July 12-15, 2017

Update: October 15, 2019

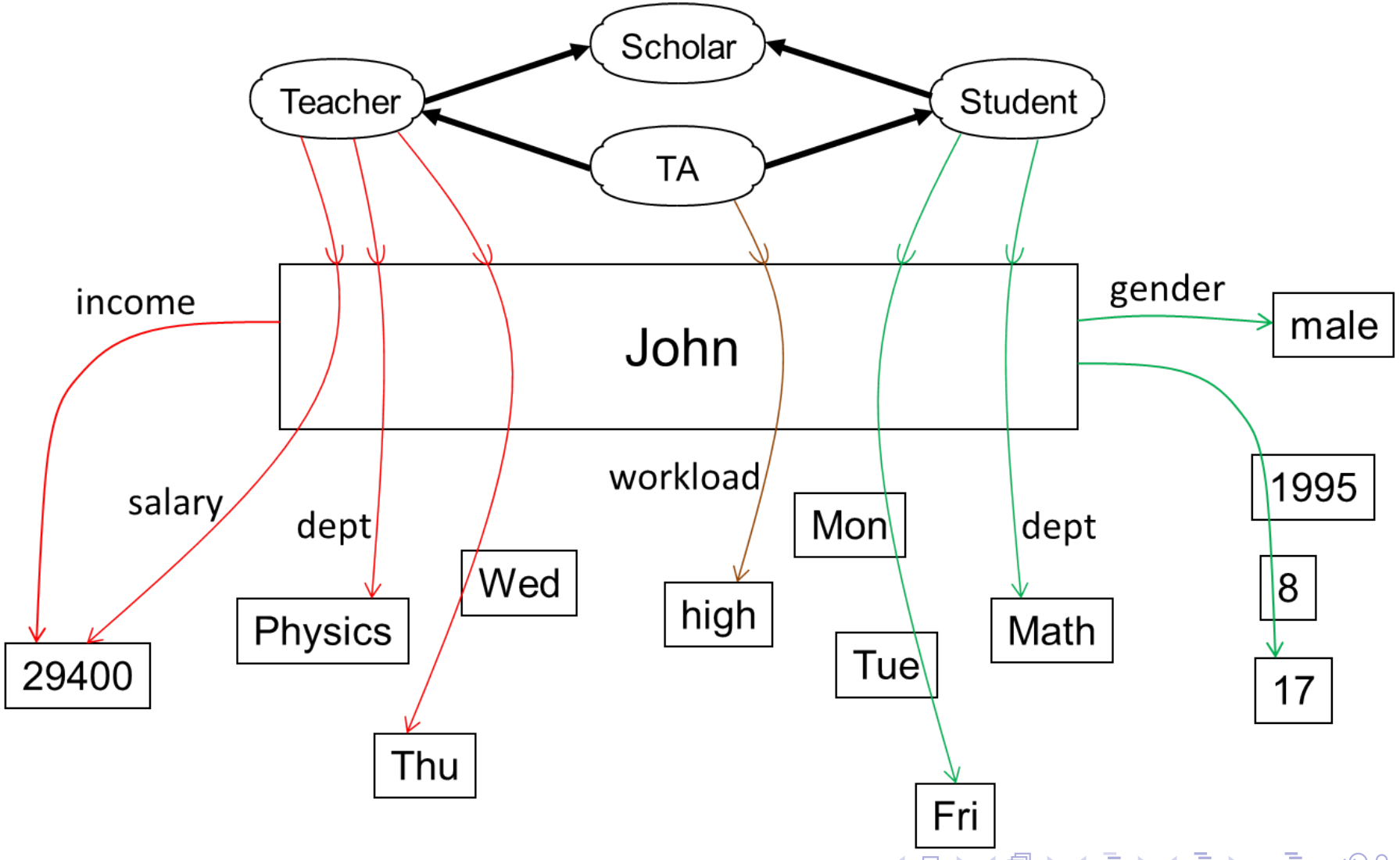
Introduction: PSOA RuleML's Third Dimension

- **P**ositional-**S**lotted **O**bject-**A**pplicative (**PSOA**) RuleML permits atom to apply predicate – possibly identified by Object IDentifier (OID) typed by predicate – to bag of tupled descriptors, each representing argument sequence, and to bag of slotted descriptors, each representing attribute-value pair
- Dimensions of oidless-vs.-oidful predicate applications & tupled-vs.-slotted descriptors augmented by 3rd dimension: (predicate-) independent-vs.-dependent descriptors, hence psoa atoms

Introduction: AI KBs with Context and Perspective

- In advanced Artificial Intelligence (AI) Knowledge Bases (KBs), notions like “context” and “perspective” are becoming increasingly relevant
  - While *context* mechanism allows to partition clauses of KB,
  - introduction of *perspective* allows to describe same OID differently via multiple clause conclusions – e.g., atoms employed as facts – having different predicates

Rich TA in PSOA: Individual OID John Described Independently and Under the Perspectives of (Dependent on) Predicates Teacher, TA, Student



Rich TA in PSOA: Visualization Syntax Revealed

- Upper part shows diamond-shaped taxonomy of four predicates – `Scholar`, `Teacher`, `Student`, and `TA` – connected by heavy arcs understood to be implicitly labeled with `subClassOf`, where `TA` – directly below both `Teacher` and `Student` – exemplifies multiple inheritance
- Lower part uses three predicates for dependent hyperarc arrows starting with predicate labelnode, e.g. `Teacher`, pointing to OID, `John`, and cutting through further nodes before pointing to the last node. Optional labels on these hyperarcs, e.g. `dept`, are slot names, thus distinguishing slot from tuple hyperarcs. E.g., `Teacher` hyperarcs indicate, from right to left, that – dependent on perspective of `Teacher` – `John` is described by (length-2 tuple for) `Wed` followed by `Thu`, is in `dep`(artmen)`t` of `Physics`, and has `salary` of `29400`. On extreme left, labeled arc, starting directly at OID, records `John`'s (total) `income` (also) as `29400` – independently of, e.g., `Teacher`, `TA`, and `Student` perspectives

Rich TA in PSOA: “as a” Entails “is a”

- Since `John` is represented as OID node pointed to and cut through by hyperarcs starting with three different predicates – `Teacher`, `TA`, and `Student` – he is involved under these different perspectives
- The “pointing to” also entails multi-membership of `John` in three predicates, here acting as classes
- Abbreviating “under the perspective of” to “as a”, we say **“as a” entails “is a”**, where “is a” of Semantic Nets is often called “is member of” on Semantic Web

Rich TA in PSOA: Presentation Syntax for KB of Atomic Ground (Variable-less) Clauses

% (KB2)

```
Teacher##Scholar    Student##Scholar
TA##Teacher         TA##Student
John#TA(workload+>high)
John#Teacher(+[Wed Thu]
             dept+>Physics
             salary+>29400
             income->29400)
John#Student(+[Mon Tue Fri]
             -[1995 8 17]
             dept+>Math
             gender->male)
```

Rich TA in PSOA: Presentation Syntax Explained (1)

- In `(KB2)`'s upper four clauses, representing TA-diamond taxonomy part of visualization, "`##`" infix indicates `subClassOf` relation
- In lower three clauses, representing rest as **p**ositional-**s**lotted **o**bject-**a**pplicative (**psoa**) atoms asserted as ground facts,
  the following notation is employed

Rich TA in PSOA: Presentation Syntax Explained (2)

- Psoa atoms use three predicates for dependent hyperarc arrows starting with predicate labelnode, e.g. `Teacher`, pointing to OID, `John`, and cutting through further nodes before pointing to last node
- Optional labels on these hyperarcs, e.g. `dept`, are slot names, thus distinguishing slot from tuple hyperarcs

Rich TA in PSOA: Presentation Syntax Explained (3)

- E.g., `Teacher` hyperarcs indicate, from right to left, that – under perspective of `Teacher` – `John` is described by (length-2 tuple for) `Wed` followed by `Thu`, is in `dep`(artmen)`t` of `Physics`, and has `salary` of `29400`
- On extreme left, labeled arc, starting directly at OID, records `John`'s (total) `income` (also) as `29400` – independently of, e.g., `Teacher`, `TA`, and `Student` perspectives

Rich TA in PSOA: Presentation Syntax Explained (4)

Dual prefix characters "+" vs. "−" uniformly used for, respectively, dependent vs. independent descriptors, leading to four kinds of descriptors (exemplified with descriptors of `(KB2)`'s `Student` atom):

- For tuples, "+" vs. "−" used prior to opening square brackets, yielding syntaxes `+[...]` vs. `-[...]`, e.g. `+[Mon Tue Fri]` vs. `-[1995 8 17]`
- For slots, "+" vs. "−" used as shafts of infix arrows, yielding syntaxes `...+>...` vs. `...->...`, e.g. `dept+>Math` vs. `gender->male`

Workload Example: Rules Can Use All of the above Descriptors in their Conditions and Conclusions

```
Forall ?o ?ht ?hs (                          % (R1)
 ?o#TA(workload+>high)
 :-
 And(
   ?o#Teacher(coursehours+>?ht)
   External(
     pred:numeric-greater-than(?ht 10))      % ?ht>10
   ?o#Student(coursehours+>?hs)
   External(
     pred:numeric-greater-than(?hs 18)))     % ?hs>18
    )
```

Workload Example: Rule Explained

- Rule conclusion deduces – for any OID `?o` that is member of `TA` – `TA`-dependent slot `workload+>high` from condition doing arithmetic threshold comparisons for `Teacher`-dependent slot `coursehours+>?ht` and `Student`-dependent slot `coursehours+>?hs`
- Three `?o` occurrences refer to same individual, but under different perspectives
- Rule thus augments each quantitative-condition-satisfying OID `?o` with dependent qualitative `workload` slot

Workload Example: Rule Applied (1)

Assuming that `(KB2)`'s `Teacher/Student` descriptors for `John` are augmented by corresponding dependent quantitative `coursehours` slots,

```
John#Teacher(... coursehours+>12 ...)
John#Student(... coursehours+>20 ...)
```

rule used to answer dependent-slot query

```
John#TA(workload+>high)
```

by unifying with conclusion, followed by retrieval of `Teacher/Student`-dependent `coursehours` `12/20` in first/third conditions and "`>`"-comparing them with thresholds `10/18` in second/fourth conditions

Workload Example: Rule Applied (2)

Similarly, rule will make dependent-slot non-ground (variable-containing) query

`?who#TA(workload+>?level)`

succeed, with bindings `?who` = `John` and `?level` = `high`

Presentation Syntax: Template for Psoa Terms

Four lines of subsequences for four kinds of descriptors, where superscripts indicate subterms that are part of dependent ($^+$) vs. independent ($^-$) descriptors, and right-slot, right-independent normal form is assumed:

$$\begin{aligned}
&\texttt{o\#f(}+\texttt{[}t^+_{1,1}\ \ldots\ t^+_{1,n^+_1}\texttt{]}\ \ldots\ +\texttt{[}t^+_{m^+,1}\ \ldots\ t^+_{m^+,n^+_{m^+}}\texttt{]}\\
&\quad -\texttt{[}t^-_{1,1}\ \ldots\ t^-_{1,n^-_1}\texttt{]}\ \ldots\ -\texttt{[}t^-_{m^-,1}\ \ldots\ t^-_{m^-,n^-_{m^-}}\texttt{]}\\
&\quad p^+_1\texttt{+>}v^+_1\ \ldots\ p^+_{k^+}\texttt{+>}v^+_{k^+}\\
&\quad p^-_1\texttt{->}v^-_1\ \ldots\ p^-_{k^-}\texttt{->}v^-_{k^-}\texttt{)}
\end{aligned}$$

Presentation Syntax: Principles

- Employs document root `RuleML`, rather than earlier `Document`, and `Assert`, rather than earlier `Group`, complementing it with `Query`
- Refines all descriptors for ("`DI`"-)distinction of Dependent vs. Independent tuples (`TUPLEDI`) and slots (`SLOTDI`)
- Reflects use of (a) oidless and oidful psoa terms as `Atom`s in/as `FORMULA`s, (b) oidful `Atom`s (for unnesting, leaving behind the OID term) as `TERM`s in `Atom`s and `Expr`essions, as well as (c) oidless psoa terms as `Expr`essions
- Revises `CLAUSE`, `Implies`, and `HEAD` productions for closure under objectification and slotribution/tupribution

Presentation Syntax: EBNF for Rule Language

```
RuleML ::= 'RuleML' '(' Base? Prefix* Import*
                                (Assert | Query)* ')'
Base ::= 'Base' '(' ANGLEBRACKIRI ')'
Prefix ::= 'Prefix' '(' Name ANGLEBRACKIRI ')'
Import ::= 'Import' '(' ANGLEBRACKIRI PROFILE? ')'
Assert ::= 'Assert' '(' (RULE | Assert)* ')'
Query ::= 'Query' '(' FORMULA ')'
RULE ::= ('Forall' Var+ '(' CLAUSE ')') | CLAUSE
CLAUSE ::= Implies | HEAD
Implies ::= HEAD ':-' FORMULA
HEAD ::= ATOMIC | 'Exists' Var+ '(' HEAD ')' |
                              'And' '(' HEAD* ')'
PROFILE ::= ANGLEBRACKIRI
```

Presentation Syntax: EBNF for Condition Language

```
FORMULA ::= 'And'/'Or' '(' FORMULA* ')' |
            'Exists' Var+ '(' FORMULA ')' |
            ATOMIC | 'External' '(' Atom ')'
ATOMIC ::= Atom | Equal | Subclass
Atom ::= ATOMOIDLESS | ATOMOIDFUL
ATOMOIDLESS/FUL ::= PSOAOIDLESS/FUL
Equal ::= TERM'='TERM    Subclass ::= TERM'##'TERM
PSOA ::= PSOAOIDLESS | PSOAOIDFUL
PSOAOIDLESS ::= TERM '('(TERM*|TUPLEDI*)SLOTDI* ')'
PSOAOIDFUL ::= TERM '#' PSOAOIDLESS
TUPLEDI ::= ('+' | '-') '[' TERM* ']'
SLOTDI ::= TERM ('+>' | '->') TERM
TERM ::= Const | Var | ATOMOIDFUL | Expr |
Expr ::= PSOAOIDLESS        'External' '(' Expr ')'
Const ::= '"' UNICODESTRING '"^^' SYMSPACE |
Var ::= '?' PN_LOCAL?                    CONSTSHORT
SYMSPACE ::= ANGLEBRACKIRI | CURIE
```

Serialization Syntax: Principles

- PSOA RuleML 1.0/XML serialization syntax extends Hornlog RuleML 1.02/XML
- XML serialization syntax of PSOA RuleML 1.0 can be derived from presentation syntax
- Mainly differs from presentation syntax in being “striped”, alternating between edges (absent from presentation syntax) & Nodes
- For (dependent and independent) descriptor-defining EBNF-grammar productions of presentation syntax (reproduced – slightly modified – with “P(resentation):” label), we give EBNF-like productions of serialization syntax (introduced with “X(ML):” label)

Serialization Syntax: Condition Language Descriptors (Presentation to Serialization)

```
P: TUPLEDI ::= '+' '['TERM* ']' | '-' '['TERM* ']'
X: TUPLEDI ::= tupdep | tup    % Different edges
X: tupdep ::= Tuple            % lead into same
X: tup ::= Tuple               % Tuple Node

P: SLOTDI ::= TERM '+>' TERM | TERM '->' TERM
X: SLOTDI ::= slotdep | slot   % Different edges
X: slotdep ::= TERM TERM       % lead into same
X: slot ::= TERM TERM          % pair of TERM Nodes
```

Serialization Syntax: Template for Psoa Terms

General case of psoa terms in serialization syntax can be instantiated for atoms as follows (decorated letters t, p, and v are understood here to stand for serialized terms, properties, i.e. slot names, and values, i.e. slot fillers):

<Atom>
<oid><Ind>o</Ind></oid><op><Rel>f</Rel></op>
<tupdep><Tuple>$t^+_{1,1} \ldots t^+_{1,n^+_1}$</Tuple></tupdep> . . .
<tupdep><Tuple>$t^+_{m^+,1} \ldots t^+_{m^+,n^+_{m^+}}$</Tuple></tupdep>
<tup><Tuple>$t^-_{1,1} \ldots t^-_{1,n^-_1}$</Tuple></tup> . . .
<tup><Tuple>$t^-_{m^-,1} \ldots t^-_{m^-,n^-_{m^-}}$</Tuple></tup>
<slotdep>$p^+_1\ v^+_1$</slotdep> . . . <slotdep>$p^+_{k^+}\ v^+_{k^+}$</slotdep>
<slot>$p^-_1\ v^-_1$</slot> . . . <slot>$p^-_{k^-}\ v^-_{k^-}$</slot>
</Atom>

Serialization Syntax: Rich TA Facts (1)

Psoa-atom facts of Rich TA example (KB2) in presentation syntax result in this serialization:

```
<Atom>
   <oid><Ind>John</Ind></oid><op><Rel>TA</Rel></op>
   <slotdep><Ind>workload</Ind><Ind>high</Ind></slotdep>
</Atom>

<Atom>
   <oid><Ind>John</Ind></oid><op><Rel>Teacher</Rel></op>
   <tupdep><Tuple><Ind>Wed</Ind><Ind>Thu</Ind></Tuple></tupdep>
   <slotdep><Ind>dept</Ind><Ind>Physics</Ind></slotdep>
   <slotdep><Ind>salary</Ind><Ind>29400</Ind></slotdep>
   <slot><Ind>income</Ind><Ind>29400</Ind></slot>
</Atom>
```

Serialization Syntax: Rich TA Facts (2)

```
<Atom>
    <oid><Ind>John</Ind></oid><op><Rel>Student</Rel></op>
    <tupdep>
        <Tuple><Ind>Mon</Ind><Ind>Tue</Ind><Ind>Fri</Ind></Tuple>
    </tupdep>
    <tup>
        <Tuple><Ind>1995</Ind><Ind>8</Ind><Ind>17</Ind></Tuple>
    </tup>
    <slotdep><Ind>dept</Ind><Ind>Math</Ind></slotdep>
    <slot><Ind>gender</Ind><Ind>male</Ind></slot>
</Atom>
```

Transformations: Slotribution/Tupribution

To incorporate dependent descriptors, slotribution/tupribution is revised to replace every oidful psoa atom having general form

$$\begin{aligned}&\texttt{o\#f(}+\texttt{[}t^+_{1,1}\ \ldots\ t^+_{1,n^+_1}\texttt{]}\ \ldots\ +\texttt{[}t^+_{m^+,1}\ \ldots\ t^+_{m^+,n^+_{m^+}}\texttt{]}\\ &\quad -\texttt{[}t^-_{1,1}\ \ldots\ t^-_{1,n^-_1}\texttt{]}\ \ldots\ -\texttt{[}t^-_{m^-,1}\ \ldots\ t^-_{m^-,n^-_{m^-}}\texttt{]}\\ &\quad p^+_1\texttt{+>}v^+_1\ \ldots\ p^+_{k^+}\texttt{+>}v^+_{k^+}\\ &\quad p^-_1\texttt{->}v^-_1\ \ldots\ p^-_{k^-}\texttt{->}v^-_{k^-}\texttt{)}\end{aligned}$$

with the conjunction

$$\begin{aligned}&\texttt{And(o\#f}\\ &\quad \texttt{o\#f(}+\texttt{[}t^+_{1,1}\ \ldots\ t^+_{1,n^+_1}\texttt{])}\ \ldots\ \texttt{o\#f(}+\texttt{[}t^+_{m^+,1}\ \ldots\ t^+_{m^+,n^+_{m^+}}\texttt{])}\\ &\quad \texttt{o\#Top(}-\texttt{[}t^-_{m^-,1}\ \ldots\ t^-_{m^-,n^-_{m^-}}\texttt{])}\ \ldots\ \texttt{o\#Top(}-\texttt{[}t^-_{m^-,1}\ \ldots\ t^-_{m^-,n^-_{m^-}}\texttt{])}\\ &\quad \texttt{o\#f(}p^+_1\texttt{+>}v^+_1\texttt{)}\ \ldots\ \texttt{o\#f(}p^+_{k^+}\texttt{+>}v^+_{k^+}\texttt{)}\\ &\quad \texttt{o\#Top(}p^-_1\texttt{->}v^-_1\texttt{)}\ \ldots\ \texttt{o\#Top(}p^-_{k^-}\texttt{->}v^-_{k^-}\texttt{))}\end{aligned}$$

Conversion to Relational-only Logic

Slotribution/tupribution-yielded conjuncts are converted to relational-only logic using reserved predicates `tupterm`, `prdtupterm`, `sloterm`, and `prdsloterm` for independent tuple terms, dependent tuple terms, independent slot terms, and dependent slot terms, respectively, as shown in table where ρ denotes recursive mapping from PSOA to Prolog, TPTP, etc.

Psoa Atoms	Prolog and TPTP Atoms
o#f	memterm(ρ(o), ρ(f))
o#Top(-[t_1...t_n])	tupterm(ρ(o), ρ(t_1), ..., ρ(t_n))
o#f(+[t_1...t_n])	prdtupterm(ρ(o), ρ(f), ρ(t_1), ..., ρ(t_n))
o#Top(p->v)	sloterm(ρ(o), ρ(p), ρ(v))
o#f(p+>v)	prdsloterm(ρ(o), ρ(f), ρ(p), ρ(v))

Conclusions

- Model-theoretic semantics uses 'built-in' slotribution/tupribution
- PSOATransRun implementation realizes this in the transformation chain of an efficient Java/ANTLR-based open-source system:

  http://wiki.ruleml.org/index.php/PSOA_RuleML#PSOATransRun

- Schema for serialization syntax is being standardized in Relax NG:

  http://wiki.ruleml.org/index.php/PSOA_RuleML#Syntaxes