RuleML-Integrated Positional-Slotted, Object-Applicative Terms and Rules with a RIF-Style First-Order Model-Theoretic Semantics

Seminar at the School of Computing Science, Simon Fraser University, Vancouver, 8 August 2011

Harold Boley

Institute for Information Technology, National Research Council;
Faculty of Computer Science, University of New Brunswick, Canada

Motivation: Linked Object-and-Rule Interchange

- Linking is characteristic for modern object and rule languages on the (Social Semantic) Web
- Employ IRIs – Internationalized Resource Identifiers – as OIDs – Object IDentifiers – (which can also occur as slot fillers), slot keys, type identifiers, and document identifiers for object & rule bases (modules)
- Linked objects and rules can thus be regarded as generalized linked data
- Therefore, object & rule language such as RuleML usable for exchange of information ranging from linked data sets to linked Horn/F-logic knowledge bases to linked first-order and higher-order logic documents

Introduction: Two IT Paradigms

Knowledge representation & problem solving in

- AI
- the (Semantic) Web
- IT at large

can be

1. Relational (and logic-based):
   FOL, Horn, LP
2. Object-oriented (and frame-based):
   CLOS, RDF, N3

Combined approaches:

- Description Logics (DLs)
- Object-Oriented Databases (OODBs) /
  Deductive Object-Oriented Databases (DOODs)
- Object-oriented logic languages:
  LIFE and Frame logic (F-logic)
- W3C Rule Interchange Format (RIF):
  - Semantics based on F-logic
  - Serialization syntax based on RuleML

Introduction: Object-Extended Semantics

- F-logic and RIF extend first-order model-theoretic semantics for **objects (frames)**
- Added separately from **function and predicate applications to arguments**
- Resulting complexity of object-extended semantics can be reduced by **integrating objects with applications**

Introduction: Object-Extended Semantics (Cont'd)

- Integration based on **positional-slotted, object-applicative** rules of POSL and RuleML
- **F-logic's model-theoretic semantics in the style of RIF** is also the **starting point** of our **integrated semantics**
- Permits **applications with optional object identifiers** and, orthogonally, **arguments that are positional or slotted**
- Structured by these **independent dimensions** of defining features, language **constructs can be freely combined**

Introduction: Psoa Terms and Rules

- RuleML-2011 paper formalizes **p**ositional-**s**lotted, **o**bject-**a**pplicative (***psoa***) terms and rules
- Psoa term applies **function or predicate** symbol, possibly **instantiated by object**, to zero or more **positional or slotted (named)** arguments
- For a psoa term as **atomic formula**, predicate symbol is **class (type) of object** as well as **relation between arguments**, which describe object
- Each **argument** of a psoa term can be psoa term applying **function** symbol

- Psoa terms that apply a predicate symbol (as a relation) to *positional arguments* can be employed to make factual assertions

- An example, in simplified RIF (presentation) syntax, is term `married(Joe Sue)` for binary predicate `married` applied to `Joe` and `Sue`, where positional (left-to-right) order can be used to identify husband, as 1*st* argument, and wife, as 2*nd* argument

- Psoa terms that apply a predicate symbol (as a class) to *slotted arguments* correspond to typed attribute-value descriptions
- An example is psoa term `family(husb->Joe wife->Sue)` or `family(wife->Sue husb->Joe)` for `family`-typed attribute-value pairs (slots) {<`husb`,`Joe`>, <`wife`,`Sue`>}
  - Easily extended with further slots, e.g. by adding children, as in `family(husb->Joe wife->Sue child->Pete)`

Introduction: Distinctions in Psoa Taxonomy (Cont'd)

- Usually, slotted terms describe an object symbol, i.e. an object identifier (OID), maintaining object identity even when slots of their descriptions are added or deleted
  - This leads to (typed) frames in the sense of F-logic
- E.g., using RIF's membership syntax #, OID `inst1` in class `family` is describable by `inst1#family(husb->Joe wife->Sue)`, `inst1#family(husb->Joe wife->Sue child->Pete)`, etc. Psoa terms can also specialize to class membership terms, e.g. `inst1#family()`, abridged `inst1#family`, represents `inst1` $\in$ `family`

Introduction: Slotted and Positional OID Description

- Like OID-describing slotted terms constitute a (multi-slot) ‘frame’, positional terms that describe an object constitute a (single-tuple) ‘shelf’, similar to a (one-dimensional) array describing its name
- Thus, `family`’s `husb` and `wife` slots can be positionalized as in earlier `married` example: `inst1#family(Joe Sue)` describes `inst1` with tuple `[Joe Sue]`

Introduction: Positional-Slotted OID Description

- *Combined positional-slotted* psoa terms are allowed, similarly as in XML elements (tuple$\rightsquigarrow$subelements, slots$\rightsquigarrow$attributes), e.g. describing an object, as in RDF descriptions (object$\rightsquigarrow$subject, slots$\rightsquigarrow$properties)
- For example, `inst1#family(Joe Sue child->Pete)` describes `inst1` with two positional and one slotted argument

Introduction: Positional and Slotted OID Description

- On the other hand, positional `married` example could be made slotted, leading to

  `married(husb->Joe wife->Sue),`

  and even be used to describe a (marriage) object: positionally, as in

  `inst2#married(Joe Sue),`

  or slotted, as in

  `inst2#married(husb->Joe wife->Sue)`

- A **frame without explicit class** is semantically treated as **typing its object with root class** $\top$ (syntactically, `Top`)
- For example, (untyped) frame `inst3[color->red shape->diamond]` in square-bracketed F-logic/RIF syntax is equivalent to our parenthesized `inst3#Top(color->red shape->diamond)`

Introduction: Atom Objectification

- An **atomic formula without OID** is treated as **having implicit OID**
- An OID-less application is *objectified* by syntactic transformation: *The OID of a ground fact is new constant generated by 'new local constant' (stand-alone _); the OID of non-ground fact or atomic formula in rule conclusion,* `f(...)`*, is new, existentially scoped variable* `?i`*, leading to* `Exists ?i (?i#f(...))`*; the OID of other atomic formulas is new variable generated by 'anonymous variable' (stand-alone* `?`*)*
- Objectification allows compatible semantics for an atom constructed as RIF-like slotted (named-argument) term and corresponding frame, solving issue with named-argument terms:

  http://lists.w3.org/Archives/Public/public-rif-wg/2008Jul/0000.html

- For example, slotted-fact assertion `family(husb->Joe wife->Sue)` is syntactically objectified to assertion `_#family(husb->Joe wife->Sue)`, and — if `_1` is first new constant from `_1`, `_2`, ... — to `_1#family(husb->Joe wife->Sue)`
- This typed frame, then, is semantically *slotributed* to `_1#family(husb->Joe)` and `_1#family(wife->Sue)`

- Query `family(husb->Joe)` is syntactically objectified to query `?#family(husb->Joe)`, i.e. — if `?1` is first new variable in `?1`, `?2`, ... — to `?1#family(husb->Joe)`
- Posed against fact, it succeeds with first slot, unifying `?1` with `_1`
- Slotribution ('slot distribution') avoids POSL's 'rest-slot' variables: frame's OID 'distributes' over slots of a description

Introduction: Psoa Rules

- Rules can be defined on top of psoa terms in a natural manner
- A rule derives (a conjunction of possibly existentially scoped) conclusion psoa atoms from (a formula of) premise psoa atoms
- Consider example with rule deriving `family` frames

Introduction: Psoa Rules Exemplified

Example (Rule-defined anonymous family frame)

`Group` is used to collect a rule and two facts. `Forall` quantifier declares orginal universal argument variables and generated universal OID variables `?2`, `?3`, `?4`. Infix `:-` separates conclusion from premises of rule, which derives anonymous/existential `family` frame from `married` relation `And` from `kid` relation of `husb` `Or` `wife` (the left-hand side is objectified on the right).

```
Group (
 Forall ?Hu ?Wi ?Ch (

  family(husb->?Hu wife->?Wi child->?Ch):-
   And(married(?Hu ?Wi)
       Or(kid(?Hu ?Ch) kid(?Wi ?Ch))) )
 married(Joe Sue)
 kid(Sue Pete)
      )
```

```
Group (
 Forall ?Hu ?Wi ?Ch ?2 ?3 ?4 (
  Exists ?1 (
   ?1#family(husb->?Hu wife->?Wi child->?Ch)) :-
    And(?2#married(?Hu ?Wi)
        Or(?3#kid(?Hu ?Ch) ?4#kid(?Wi ?Ch))) )
 _1#married(Joe Sue)
 _2#kid(Sue Pete)
      )
```

Semantically, example is modeled by predicate extensions corresponding to following set of ground facts (the subdomain of individuals $\boldsymbol{D}_{\text{ind}}$ is to be defined):

{*o#family(husb->Joe wife->Sue child->Pete)* } ∪

{*_1#married(Joe Sue)*, *_2#kid(Sue Pete)*}, where $o \in \boldsymbol{D}_{\text{ind}}$.

Introduction: PSOA RuleML Presentation

PSOA RuleML is defined here as a language incorporating this integration:

- PSOA RuleML's human-readable presentation syntax
- PSOA RuleML's model-theoretic semantics
- Conclusion and future work

Presentation Syntax: Alphabet

Definition (Alphabet)

The ***alphabet*** of the presentation language of PSOA RuleML consists of the following disjoint sets:

- A countably infinite set of ***constant symbols*** `Const` (including the root class `Top` ∈ `Const` and the positive-integer-enumerated local constants `_1`, `_2`, `...` ∈ `Const` as well as individual, function, and predicate symbols)
- A countably infinite set of ***variable symbols*** `Var` (including the positive-integer-enumerated variables `?1`, `?2`, `...` ∈ `Var`)
- The connective symbols `And`, `Or`, and `:-`
- The quantifiers `Exists` and `Forall`
- The symbols `=`, `#`, `##`, `->`, `External`, `Import`, `Prefix`, and `Base`
- The symbols `Group` and `Document`

Presentation Syntax: Alphabet (Cont'd)

Definition (Alphabet, Cont'd)

Constants have form `"literal"^^symspace`, where `literal` is a sequence of Unicode characters and `symspace` is an identifier for a symbol space. E.g., `"_123"^^rif:local`. Constants use shortcuts defined in RIF-DTB, including underscored `_literal` (e.g., `_123`) for above form with `symspace` specialized to `rif:local`. `Top` is a new shortcut for root class constant `"Top"^^psoa:global` in PSOA RuleML's global symbol space

Anonymous variables are written as a stand-alone question mark symbol (`?`); named variables, as Unicode strings preceded with question mark symbol

Symbols `=` and `##` are used in formulas that define equality and subclass relationships. The symbols `#` and `->` are used in positional-slotted, object-applicative formulas for class memberships and slots, respectively. Symbol `External` indicates that an atomic formula or a function term is defined externally (e.g., a built-in) and symbols `Prefix` and `Base` enable abridged representations of IRIs (Internationalized Resource Identifiers) □

Presentation Syntax: Terms

In this definition, *base term* means a simple term, an anonymous psoa term (i.e., an anonymous frame term, single-tuple psoa term, or multi-tuple psoa term), or a term of the form `External(t)`, where `t` is an anonymous psoa term. Anonymous term can be *deobjectified* (by omitting main `?#`) if its re-objectification results in old term (i.e., re-introduces `?#`).

Definition (Term)

1. *Constants and variables*. If $\mathtt{t} \in \mathtt{Const}$ or $\mathtt{t} \in \mathtt{Var}$ then `t` is a ***simple term***
2. *Equality terms*. $\mathtt{t} = \mathtt{s}$ is an ***equality term*** if `t`, `s` are base terms
3. *Subclass terms*. `t##s` is a ***subclass term*** if `t`, `s` are base terms
4. *Positional-slotted, object-applicative terms*.
   $\mathtt{o\#f([t_{1,1}...t_{1,n_1}]...[t_{m,1}...t_{m,n_m}]\ p_1\text{->}v_1...p_k\text{->}v_k)}$
   is a ***positional-slotted, object-applicative (psoa) term*** if
   $\mathtt{f} \in \mathtt{Const}$ and $\mathtt{o}, \mathtt{t_{1,1}}, ..., \mathtt{t_{1,n_1}}, \ldots, \mathtt{t_{m,1}}, ..., \mathtt{t_{m,n_m}}, \mathtt{p_1}, ..., \mathtt{p_k}, \mathtt{v_1}, ..., \mathtt{v_k}, \mathtt{m} \geq 0, \mathtt{k} \geq 0$, are base terms

Presentation Syntax: Terms (Cont'd)

Definition (Term, Cont'd)

Psoa terms can be specialized in the following way

- For `m = 0` they become ***(typed or untyped) frame terms*** $\texttt{o\#f(p}_1\texttt{->v}_1\ \texttt{...}\ \texttt{p}_k\texttt{->v}_k\texttt{)}$. We consider two overlapping subcases
  - For `k = 0` they become ***class membership terms*** `o#f()`, abridged to `o#f`, corresponding to those in F-logic and RIF
  - For $\texttt{k} \geq \texttt{0}$ they can be further specialized in two ways, which can be orthogonally combined
    - For `o` being the anonymous variable `?`, they become ***anonymous frame terms (slotted terms)*** $\texttt{?\#f(p}_1\texttt{->v}_1\ \texttt{...}\ \texttt{p}_k\texttt{->v}_k\texttt{)}$, deobjectified $\texttt{f(p}_1\texttt{->v}_1\ \texttt{...}\ \texttt{p}_k\texttt{->v}_k\texttt{)}$, corresponding to *terms with named arguments* in RIF
    - For `f` being the root class `Top`, they become ***untyped frame terms*** $\texttt{o\#Top(p}_1\texttt{->v}_1\ \texttt{...}\ \texttt{p}_k\texttt{->v}_k\texttt{)}$ corresponding to *frames* in abridged form $\texttt{o[p}_1\texttt{->v}_1\ \texttt{...}\ \texttt{p}_k\texttt{->v}_k\texttt{]}$ of F-logic and RIF, where square brackets are used instead of round parentheses

Presentation Syntax: Terms (Cont'd)

Definition (Term, Cont'd)

- For `m = 1` psoa terms become ***single-tuple psoa terms*** $\texttt{o\#f([t}_{1,1}$ ... $\texttt{t}_{1,n_1}\texttt{] p}_1\texttt{->v}_1 \texttt{ ... p}_k\texttt{->v}_k\texttt{)}$, abridged to $\texttt{o\#f(t}_{1,1}$ ... $\texttt{t}_{1,n_1}\texttt{ p}_1\texttt{->v}_1\texttt{ ... p}_k\texttt{->v}_k\texttt{)}$

  These can be further specialized in two ways, which can be orthogonally combined:

  - For `o` being the anonymous variable `?`, they become ***anonymous single-tuple psoa terms*** $\texttt{?\#f(t}_{1,1}$ ... $\texttt{t}_{1,n_1}\texttt{ p}_1\texttt{->v}_1\texttt{...p}_k\texttt{->v}_k\texttt{)}$, deobjectified $\texttt{f(t}_{1,1}$ ... $\texttt{t}_{1,n_1}\texttt{ p}_1\texttt{->v}_1\texttt{...p}_k\texttt{->v}_k\texttt{)}$. These can be further specialized:
    - For `k = 0`, they become ***positional terms*** $\texttt{?\#f(t}_{1,1}$ ... $\texttt{t}_{1,n_1}\texttt{)}$, deobjectified $\texttt{f(t}_{1,1}$ ... $\texttt{t}_{1,n_1}\texttt{)}$, corresponding to the usual terms and atomic formulas of classical first-order logic
  - For `f` being the root class `Top`, they become ***untyped single-tuple psoa terms*** $\texttt{o\#Top(t}_{1,1}$ ... $\texttt{t}_{1,n_1}\texttt{ p}_1\texttt{->v}_1\texttt{ ... p}_k\texttt{->v}_k\texttt{)}$. These can be further specialized:
    - For `k = 0`, they become ***untyped single-tuple shelf terms*** $\texttt{o\#Top(t}_{1,1}$ ... $\texttt{t}_{1,n_1}\texttt{)}$ describing object `o` with positional arguments $\texttt{t}_{1,1}$, ..., $\texttt{t}_{1,n_1}$

Presentation Syntax: Terms (Cont'd)

Definition (Term, Cont'd)

5. *Externally defined terms.* If `t` is anonymous psoa term then `External(t)` is an ***externally defined term*** External terms represent built-in function or predicate invocations as well as "procedurally attached" function or predicate invocations. Procedural attachments are often provided by rule-based systems, and external terms constitute a way of supporting them in PSOA RuleML □

Presentation Syntax: Multiple Tuples

- The notion of psoa term is generalized here from allowing a single tuple, as in POSL, to allowing a bag (multi-set) of tuples
- Together with ‘tupribution’, this accommodates for distributed positional descriptions of the same OID
- For multiple tuples (m>1) each tuple is enclosed by square brackets, which can be omitted for a single tuple (m=1)
- Special case $\mathtt{n}_1 = \ldots = \mathtt{n}_\mathtt{m}$ used to describe a distributed object with ‘homogeneous’ equal-length tuples of a relation: OID names extension of relation’s tuples

Presentation Syntax: Argument Names

- Argument names of psoa terms, $\mathtt{p}_1$, ..., $\mathtt{p}_n$, are base terms, e.g. constants or variables
- Since psoa terms include anonymous frames (slotted terms), this generalizes RIF, where corresponding named-argument terms must use argument names from set `ArgNames`, to reduce unification complexity
- PSOA RuleML could emulate such special treatment of slotted terms by reserving `ArgNames`-style subset of `Const`
- But, since PSOA RuleML's slotted terms via objectification become frames, they can be queried by slotribution rather than unification

Presentation Syntax: Formulas

Any psoa term `f(...)` or `o#f(...)`, with `f` a predicate symbol and $\circ$ a simple term (constant or variable), is an ***atomic formula***. Likewise, externally defined term `External(`φ`)`, where φ is atomic formula, equality, and subclass term. Simple terms are *not* formulas

Generalized formulas built from atomic formulas by connectives

Definition (Formula, Condition Language)

A ***formula*** can have one of the following forms:

1. *Atomic*: An atomic formula is also a formula
2. *Condition formula*: A ***condition formula*** is either an atomic formula or a formula that has one of the following forms:

Presentation Syntax: Formulas (Cont'd)

Definition (Formula, Condition Language, Cont'd)

2. - *Conjunction*: If φ_1, ..., φ_n, $\mathtt{n} \geq \mathtt{0}$, are condition formulas then so is `And(`φ_1 `...` φ_n`)`, called a ***conjunctive*** formula. As special case, `And()` is allowed and treated as tautology, i.e., formula that is always true
   - *Disjunction*: If φ_1, ..., φ_n, $\mathtt{n} \geq \mathtt{0}$, are condition formulas then so is `Or(`φ_1 `...` φ_n`)`, called a ***disjunctive*** formula. As special case, `Or()` is considered as contradiction, i.e., formula that is always false
   - *Existentials*: If φ is a condition formula and $\mathtt{?V}_1$, ..., $\mathtt{?V}_n$, $\mathtt{n>0}$, are distinct variables then `Exists ?V`$_1$ `...` `?V`$_n$`(`φ`)` is an ***existential*** formula

Condition formulas are intended as premises of rules

Presentation Syntax: Formulas (Cont'd)

Definition (Formula, Rule Language)

3. *Rule implication*: φ `:-` ψ is a formula, called ***rule implication***, if:
   - φ is a head formula or a *conjunction* of head formulas, where a head formula is an atomic formula or an *existentially* scoped atomic formula,
   - ψ is a condition formula, and
   - none of the atomic formulas in φ is an externally defined term (i.e., term of the form `External(...)`)
4. *Universal rule*: If φ is a rule implication and $?V_1$, ..., $?V_n$, $n>0$, distinct variables then `Forall ?V`$_1$ `...` `?V`$_n$`(`φ`)` is a *universal rule* formula. It is required that all *free* variables in φ occur among variables `?V`$_1$ `...` `?V`$_n$ in quantification part. Generally, an occurrence of variable `?v` is *free* in φ if it is not inside subformula of φ of the form `Exists ?v (`ψ`)` and ψ is a formula. Universal rules are also referred to as *PSOA RuleML rules*.

Presentation Syntax: Formulas (Cont'd)

Definition (Formula, Rule Language, Cont'd)

5. *Universal fact*: If φ is an atomic formula and $\mathtt{?V_1}$, ..., $\mathtt{?V_n}$, $\mathtt{n>0}$, distinct variables then $\mathtt{Forall\ ?V_1\ ...\ ?V_n(}\varphi\mathtt{)}$ is a *universal fact* formula, provided that all free variables in φ occur among variables $\mathtt{?V_1\ ...\ ?V_n}$.
   Universal facts are treated as rules without premises
6. *Group*: If φ_1, ..., φ_n are PSOA RuleML rules, universal facts, variable-free rule implications or atomic formulas, *or* groups then $\mathtt{Group(}\varphi_1\ ...\ \varphi_n\mathtt{)}$ is a *group formula*.
   Group formulas represent sets of rules and facts
7. *Document*: An expression of the form $\mathtt{Document(}$*directive*$_1$ ... *directive*$_n$ $\Gamma\mathtt{)}$ is a *PSOA RuleML document formula*, if
   - Γ is an optional group formula; it is called the group formula *associated* with the document
   - *directive*$_1$, ..., *directive*$_n$ is optional sequence of *directives*. Can be *base directive*, *prefix directive* or *import directive* □

Presentation Syntax: Well-formed Formulas

- Not all formulas or documents are well-formed in PSOA RuleML
- Well-formedness restriction similar to standard first-order logic: required that no constant appear in more than one **context**
- No constant symbol can occur within same document as individual or (plain or external) function or predicate in different places

Presentation Syntax: Condition Language

Example (PSOA RuleML conditions)

This example shows conditions that are composed of psoa terms (`"Opticks"` is shortcut for `"Opticks"^^xs:string`).

```
Prefix(bks  <http://eg.com/books#>)
Prefix(auth <http://eg.com/authors#>)
Prefix(cts  <http://eg.com/cities#>)
Prefix(cpt  <http://eg.com/concepts#>)
```

Formula that uses an anonymous psoa (positional term):

```
  ?#cpt:book(auth:Newton "Opticks")
```

Deobjectified version:

```
  cpt:book(auth:Newton "Opticks")
```

Formula that uses an anonymous psoa (slotted term):

```
  ?#cpt:book(cpt:author->auth:Newton cpt:title->"Opticks")
```

Deobjectified version:

```
  cpt:book(cpt:author->auth:Newton cpt:title->"Opticks")
```

Formula that uses a named psoa (typed frame):

```
  bks:opt1#cpt:book(cpt:author->auth:Newton cpt:title->"Opticks")
```

Formula that uses a named psoa (untyped frame):

```
  bks:opt1#Top(cpt:author->auth:Newton cpt:title->"Opticks")
```

Deobjectified version of a formula that uses an anonymous psoa (multi-tuple term):

```
  cpt:book([auth:Newton "Opticks"] [cts:London "1704"^^xs:integer])
```

Deobjectified version of a formula that uses an anonymous psoa (positional-slotted term):

```
  cpt:book(auth:Newton "Opticks"
           cpt:place->cts:London
           cpt:year->"1704"^^xs:integer)
```

Presentation Syntax: Rule Language

Example (PSOA RuleML business rule)

Adapts business rule from POSL logistics use case. Ternary `reciship` conclusion represents `reci`procal `ship`pings, at total cost (as single positional argument), between `source` and `dest`ination (as two slotted arguments). First two premises apply 4-ary `shipment` relation that uses anonymous cargo and named cost variables as two positional arguments, as well as `reciship`'s slotted arguments (in both 'directions').
Third premise is `External`-wrapped `numeric-add` RIF-DTB built-in applied on right-hand side of equality to sum up `shipment` costs for total. With the two facts, `?cost = ?57.0`.

```
Prefix(cpt  <http://eg.com/concepts#>)
Prefix(mus  <http://eg.com/museums#>)
Prefix(func <http://www.w3.org/2007/rif-builtin-function#>)
Prefix(xs   <http://www.w3.org/2001/XMLSchema#>)
Group (
  Forall ?cost ?cost1 ?cost2 ?A ?B (
    cpt:reciship(?cost cpt:source->?A cpt:dest->?B) :-
      And(cpt:shipment(? ?cost1 cpt:source->?A cpt:dest->?B)
          cpt:shipment(? ?cost2 cpt:source->?B cpt:dest->?A)
          ?cost = External(func:numeric-add(?cost1 ?cost2)) )
                                            )
  shipment("PC"^^xs:string "47.5"^^xs:float
           cpt:source->mus:BostonMoS cpt:dest->mus:LondonSciM)
  shipment("PDA"^^xs:string "9.5"^^xs:float
           cpt:source->mus:LondonSciM cpt:dest->mus:BostonMoS)
      )
```

Presentation Syntax: Rule Language (Cont'd)

Example (PSOA RuleML business rule, Cont'd)

The rule can be objectified as follows (`External`s are not being transformed):

```
Forall ?cost ?cost1 ?cost2 ?A ?B ?2 ?3 (
  Exists ?1 (?1#cpt:reciship(?cost cpt:source->?A cpt:dest->?B)) :-
    And(?2#cpt:shipment(? ?cost1 cpt:source->?A cpt:dest->?B)
        ?3#cpt:shipment(? ?cost2 cpt:source->?B cpt:dest->?A)
        ?cost = External(func:numeric-add(?cost1 ?cost2)) )
                                               )
```

Further, it can be tupributed and slotributed (actually done by the semantics):

```
Forall ?cost ?cost1 ?cost2 ?A ?B ?2 ?3 (
  Exists ?1 (And(?1#cpt:reciship(?cost)
                 ?1#cpt:reciship(cpt:source->?A)
                 ?1#cpt:reciship(cpt:dest->?B))) :-
    And(?2#cpt:shipment(? ?cost1)
        ?2#cpt:shipment(cpt:source->?A)
        ?2#cpt:shipment(cpt:dest->?B)
        ?3#cpt:shipment(? ?cost2)
        ?3#cpt:shipment(cpt:source->?B)
        ?3#cpt:shipment(cpt:dest->?A)
        ?cost = External(func:numeric-add(?cost1 ?cost2)) )
                                               )
```

Semantics: RIF-Style PSOA RuleML

PSOA RuleML semantics in style of RIF-BLD — more general than what would be required:

- Ensure that the semantics stay comparable
- Future RIF logic dialect could be specified to cater for PSOA
  - E.g., via an updated RIF-FLD

Semantics: Constant Generator

- In given document, *new local constant generator*, written as stand-alone `_`, denotes first new local constant `_`i, $i \geq 1$, from the sequence `_1`, `_2`, . . . that does not already occur in that document
- For each document we assume OID-less psoa terms have undergone objectification, whose head existentials make PSOA RuleML non-Horn

- Use ***TV*** as set of semantic truth values {**t**,**f**}
- Truth valuation of PSOA RuleML formulas will be defined as mapping $TVal_{\mathcal{I}}$ in two steps:

  1. Mapping ***I*** generically bundles various mappings from semantic structure, $\mathcal{I}$;
     I maps formula to element of domain ***D***
  2. Mapping $\boldsymbol{I}_{\text{truth}}$ takes such a domain element to ***TV***

  This indirectness allows HiLog-like generality

Definition (Semantic structure)

A ***semantic structure***, $\mathcal{I}$, is a tuple of the form
<$\boldsymbol{TV}$, $\boldsymbol{DTS}$, $\boldsymbol{D}$, $\boldsymbol{D}_{\text{ind}}$, $\boldsymbol{D}_{\text{func}}$, $\boldsymbol{I}_{\text{C}}$, $\boldsymbol{I}_{\text{V}}$, $\boldsymbol{I}_{\text{psoa}}$, $\boldsymbol{I}_{\text{sub}}$, $\boldsymbol{I}_{=}$, $\boldsymbol{I}_{\text{external}}$, $\boldsymbol{I}_{\text{truth}}$>

Here $\boldsymbol{D}$ is a non-empty set of elements called the ***domain*** of $\mathcal{I}$, and $\boldsymbol{D}_{\text{ind}}$, $\boldsymbol{D}_{\text{func}}$ are nonempty subsets of $\boldsymbol{D}$

The domain must contain at least the root class: $\top \in \boldsymbol{D}$

$\boldsymbol{D}_{\text{ind}}$ is used to interpret elements of `Const` acting as individuals
$\boldsymbol{D}_{\text{func}}$ is used to interpret constants acting as function symbols

As before, `Const` denotes set of all constant symbols and `Var` set of all variable symbols

$\boldsymbol{DTS}$ denotes set of identifiers for primitive datatypes

Semantics: Semantic Structures (Cont'd)

Definition (Semantic structure, Cont'd)

The other components of $\mathcal{I}$ are *total* mappings defined thus:

1. $\boldsymbol{I}_{\mathrm{C}}$ maps $\mathtt{Const}$ to $\boldsymbol{D}$.
   This mapping interprets constant symbols.
   In addition, it is required that:
   - If a constant, $\mathtt{c} \in \mathtt{Const}$, is an *individual* then it is required that $\boldsymbol{I}_{\mathrm{C}}(\mathtt{c}) \in \boldsymbol{D}_{\mathrm{ind}}$
   - If $\mathtt{c} \in \mathtt{Const}$ is a *function symbol* then it is required that $\boldsymbol{I}_{\mathrm{C}}(\mathtt{c}) \in \boldsymbol{D}_{\mathrm{func}}$
   - It is required that $\boldsymbol{I}_{\mathrm{C}}(\mathtt{Top}) = \top$
2. $\boldsymbol{I}_{\mathrm{V}}$ maps $\mathtt{Var}$ to $\boldsymbol{D}_{\mathrm{ind}}$. Mapping interprets variable symbols

Semantics: Semantic Structures (Cont'd)

Definition (Semantic structure, Cont'd)

3. $\boldsymbol{I}_{\text{psoa}}$ maps $\boldsymbol{D}$ to total functions $\boldsymbol{D}_{\text{ind}} \times \texttt{SetOfFiniteBags}(\boldsymbol{D}^{*}_{\text{ind}}) \times \texttt{SetOfFiniteBags}(\boldsymbol{D}_{\text{ind}} \times \boldsymbol{D}_{\text{ind}}) \to \boldsymbol{D}$. Interprets psoa terms, combining positional, slotted, and frame terms, as well as class memberships. Argument $\texttt{d} \in \boldsymbol{D}$ of $\boldsymbol{I}_{\text{psoa}}$ represents function or predicate symbol of positional terms and slotted terms, and object class of frame terms, as well as class of memberships. Element $\texttt{o} \in \boldsymbol{D}_{\text{ind}}$ represents object of class $\texttt{d}$, which is described with two bags.
   - Finite bag of finite tuples $\{\texttt{<}\mathtt{t}_{1,1}, ..., \mathtt{t}_{1,n_1}\texttt{>}, ..., \texttt{<}\mathtt{t}_{m,1}, ..., \mathtt{t}_{m,n_m}\texttt{>}\} \in \texttt{SetOfFiniteBags}(\boldsymbol{D}^{*}_{\text{ind}})$, possibly empty, represents positional information. $\boldsymbol{D}^{*}_{\text{ind}}$ is set of all finite tuples over the domain $\boldsymbol{D}_{\text{ind}}$. Bags are used since order of tuples in a psoa term is immaterial and tuples may repeat
   - Finite bag of attribute-value pairs $\{\texttt{<a1,v1>}, ..., \texttt{<ak,vk>}\} \in \texttt{SetOfFiniteBags}(\boldsymbol{D}_{\text{ind}} \times \boldsymbol{D}_{\text{ind}})$, possibly empty, represents slotted information. Bags, since order of attribute-value pairs in a psoa term is immaterial and pairs may repeat

Semantics: Semantic Structures (Cont'd)

Definition (Semantic structure, Cont'd)

3. In addition:
   - If $\mathtt{d} \in \boldsymbol{D}_{\text{func}}$ then $\boldsymbol{I}_{\text{psoa}}(\mathtt{d})$ must be a ($\boldsymbol{D}_{\text{ind}}$-valued) function $\boldsymbol{D}_{\text{ind}} \times \mathtt{SetOfFiniteBags}(\boldsymbol{D}^{*}_{\text{ind}}) \times \mathtt{SetOfFiniteBags}(\boldsymbol{D}_{\text{ind}} \times \boldsymbol{D}_{\text{ind}}) \rightarrow \boldsymbol{D}_{\text{ind}}$
   - Implies that when a function symbol is applied to arguments that are individual objects then result is also individual object
4. $\boldsymbol{I}_{\text{sub}}$ gives meaning to the subclass relationship. It is a total mapping of the form $\boldsymbol{D}_{\text{func}} \times \boldsymbol{D}_{\text{func}} \rightarrow \boldsymbol{D}$
5. $\boldsymbol{I}_{=}$ is a mapping of the form $\boldsymbol{D}_{\text{ind}} \times \boldsymbol{D}_{\text{ind}} \rightarrow \boldsymbol{D}$. Gives meaning to the equality operator
6. $\boldsymbol{I}_{\text{external}}$ is a mapping to give meaning to `External` terms. Maps external symbols in `Const` to fixed functions
7. $\boldsymbol{I}_{\text{truth}}$ is a mapping of the form $\boldsymbol{D} \rightarrow \boldsymbol{TV}$. Used to define truth valuation for formulas

Semantics: Semantic Structures (Cont'd)

Definition (Semantic structure, Cont'd)

Generic mapping from terms to $\boldsymbol{D}$, denoted by $\boldsymbol{I}$

- $\boldsymbol{I}(\mathtt{k}) = \boldsymbol{I}_C(\mathtt{k})$, if `k` is a symbol in `Const`
- $\boldsymbol{I}(\mathtt{?v}) = \boldsymbol{I}_V(\mathtt{?v})$, if `?v` is a variable in `Var`
- $\boldsymbol{I}(\ \mathtt{o\#f([t_{1,1}}\ \ldots\ \mathtt{t_{1,n_1}]}\ \ldots\ \mathtt{[t_{m,1}}\ \ldots\ \mathtt{t_{m,n_m}]\ a_1\text{->}v_1}\ \ldots\ \mathtt{a_k\text{->}v_k)}\ )$
  $= \boldsymbol{I}_{psoa}(\boldsymbol{I}(\mathtt{f}))(\boldsymbol{I}(\mathtt{o}), \{<\boldsymbol{I}(\mathtt{t_{1,1}}), \ldots, \boldsymbol{I}(\mathtt{t_{1,n_1}})>, \ldots, <\boldsymbol{I}(\mathtt{t_{m,1}}), \ldots, \boldsymbol{I}(\mathtt{t_{m,n_m}})>\},$
  $\{<\boldsymbol{I}(\mathtt{a_1}),\boldsymbol{I}(\mathtt{v_1})>, \ldots, <\boldsymbol{I}(\mathtt{a_k}),\boldsymbol{I}(\mathtt{v_k})>\}\ )$
  Again {...} denote *bags* of tuples and attribute-value pairs.
- $\boldsymbol{I}(\mathtt{c1\#\#c2}) = \boldsymbol{I}_{sub}(\boldsymbol{I}(\mathtt{c1}), \boldsymbol{I}(\mathtt{c2}))$
- $\boldsymbol{I}(\mathtt{x=y}) = \boldsymbol{I}_{=}(\boldsymbol{I}(x), \boldsymbol{I}(y))$
- $\boldsymbol{I}(\mathtt{External(p(s_1\ldots s_n))}) = \boldsymbol{I}_{external}(\mathtt{p})(\boldsymbol{I}(\mathtt{s_1}), \ldots, \boldsymbol{I}(\mathtt{s_n}))$

Semantics: Method of Formula Interpretation

- Define mapping, $\mathit{TVal}_{\mathcal{I}}$, from set of all non-document formulas to $\boldsymbol{TV}$
- For atomic formula ϕ, $\mathit{TVal}_{\mathcal{I}}(\phi)$ defined essentially as $\boldsymbol{I}_{\text{truth}}(\boldsymbol{I}(\phi)))$
- Recall that $\boldsymbol{I}(\phi)$ is just an element of domain $\boldsymbol{D}$ and $\boldsymbol{I}_{\text{truth}}$ maps $\boldsymbol{D}$ to truth values in $\boldsymbol{TV}$
- Might surprise, since normally mapping $\boldsymbol{I}$ defined only for terms that occur as arguments to predicates, not for atomic formulas. Similarly, truth valuations usually defined via mappings from instantiated formulas to $\boldsymbol{TV}$, not from interpretation domain $\boldsymbol{D}$ to $\boldsymbol{TV}$
- HiLog-style definition inherited from RIF-FLD and equivalent to a standard one for first-order languages such as RIF-BLD and PSOA RuleML — but enables future higher-order features

Semantics: Interpretation of Formulas

Definition (Truth valuation)

Truth valuation for well-formed formulas in PSOA RuleML determined using function $TVal_{\mathcal{I}}$:

1. *Equality*: $TVal_{\mathcal{I}}(\mathtt{x} = \mathtt{y}) = \boldsymbol{I}_{\text{truth}}(\boldsymbol{I}(\mathtt{x} = \mathtt{y}))$.
   - Required that $\boldsymbol{I}_{\text{truth}}(\boldsymbol{I}(\mathtt{x} = \mathtt{y})) = \mathbf{t}$ if $\boldsymbol{I}(\mathtt{x}) = \boldsymbol{I}(\mathtt{y})$ and that $\boldsymbol{I}_{\text{truth}}(\boldsymbol{I}(\mathtt{x} = \mathtt{y})) = \mathbf{f}$ otherwise
   - This can also be expressed as $TVal_{\mathcal{I}}(\mathtt{x} = \mathtt{y}) = \mathbf{t}$ if and only if $\boldsymbol{I}(\text{x}) = \boldsymbol{I}(\text{y})$
2. *Subclass*: $TVal_{\mathcal{I}}(\mathtt{sc}\ \#\#\ \mathtt{c1}) = \boldsymbol{I}_{\text{truth}}(\boldsymbol{I}(\mathtt{sc}\ \#\#\ \mathtt{c1}))$.
   In particular, for root class, $\mathtt{Top}$, and all $\mathtt{sc} \in \boldsymbol{D}$, $TVal_{\mathcal{I}}(\mathtt{sc}\ \#\#\ \mathtt{Top}) = \mathbf{t}$.
   To ensure that $\#\#$ is transitive, i.e., $\mathtt{c1}\ \#\#\ \mathtt{c2}$ and $\mathtt{c2}\ \#\#\ \mathtt{c3}$ imply $\mathtt{c1}\ \#\#\ \mathtt{c3}$, the following is required:
   - For all $\mathtt{c1}, \mathtt{c2}, \mathtt{c3} \in \boldsymbol{D}$, if $TVal_{\mathcal{I}}(\mathtt{c1}\ \#\#\ \mathtt{c2}) = TVal_{\mathcal{I}}(\mathtt{c2}\ \#\#\ \mathtt{c3}) = \mathbf{t}$ then $TVal_{\mathcal{I}}(\mathtt{c1}\ \#\#\ \mathtt{c3}) = \mathbf{t}$

Semantics: Interpretation of Formulas (Cont'd)

Definition (Truth valuation, Cont'd)

3. *Psoa formula*:
$TVal_{\mathcal{I}}(\texttt{o\#f([t}_{1,1}\ldots\texttt{t}_{1,n_1}\texttt{]}\ldots\texttt{[t}_{m,1}\ldots\texttt{t}_{m,n_m}\texttt{] a}_1\texttt{->v}_1\ldots\texttt{a}_k\texttt{->v}_k\texttt{))} =$
$\boldsymbol{I}_{\text{truth}}(\boldsymbol{I}(\texttt{o\#f([t}_{1,1}\ldots\texttt{t}_{1,n_1}\texttt{]}\ldots\texttt{[t}_{m,1}\ldots\texttt{t}_{m,n_m}\texttt{] a}_1\texttt{->v}_1\ldots\texttt{a}_k\texttt{->v}_k\texttt{)}))$.
The formula consists of an object-typing membership, a bag of tuples representing a conjunction of all the object-centered tuples (*tupribution*), and a bag of slots representing a conjunction of all the object-centered slots (*slotribution*). Hence use restriction, where $\texttt{m}\geq\texttt{0}$ and $\texttt{k}\geq\texttt{0}$:
   - $TVal_{\mathcal{I}}(\texttt{o\#f([t}_{1,1}\ldots\texttt{t}_{1,n_1}\texttt{]}\ldots\texttt{[t}_{m,1}\ldots\texttt{t}_{m,n_m}\texttt{] a}_1\texttt{->v}_1\ldots\texttt{a}_k\texttt{->v}_k\texttt{))} = \mathbf{t}$
     if and only if
     $TVal_{\mathcal{I}}(\texttt{o \# f}) =$
     $TVal_{\mathcal{I}}(\texttt{o\#Top([t}_{1,1}\ldots\texttt{t}_{1,n_1}\texttt{]))} = \ldots = TVal_{\mathcal{I}}(\texttt{o\#Top([t}_{m,1}\ldots\texttt{t}_{m,n_m}\texttt{]))} =$
     $TVal_{\mathcal{I}}(\texttt{o\#Top(a}_1\texttt{->v}_1\texttt{))} = \ldots = TVal_{\mathcal{I}}(\texttt{o\#Top(a}_k\texttt{->v}_k\texttt{))} =$
     $\mathbf{t}$

Semantics: Interpretation of Formulas (Cont'd)

Definition (Truth valuation, Cont'd)

3. - Observe that on right-hand side of "if and only if" there are $\mathtt{1+m+k}$ subformulas splitting left-hand side into an object membership, $\mathtt{m}$ object-centered positional formulas, each associating the object with a tuple, and $\mathtt{k}$ object-centered slotted formulas, i.e. 'triples', each associating object with attribute-value pair. All parts on both sides of "if and only if" are centered on object $\mathtt{o}$, which connects subformulas on right-hand side (first subformula providing $\mathtt{o}$-member class $\mathtt{f}$, remaining $\mathtt{m+k}$ ones using root class $\mathtt{Top}$)

   For root class, $\mathtt{Top}$, and all $\mathtt{o} \in \boldsymbol{D}$, $TVal_{\mathcal{I}}(\mathtt{o}\,\#\,\mathtt{Top}) = \mathbf{t}$.
   To ensure that all members of subclass are also members of its superclasses, i.e., $\mathtt{o}\,\#\,\mathtt{f}$ and $\mathtt{f}\,\#\#\,\mathtt{g}$ imply $\mathtt{o}\,\#\,\mathtt{g}$, the following restriction is imposed:

   - For all $\mathtt{o}, \mathtt{f}, \mathtt{g} \in \boldsymbol{D}$, if $TVal_{\mathcal{I}}(\mathtt{o}\,\#\,\mathtt{f}) = TVal_{\mathcal{I}}(\mathtt{f}\,\#\#\,\mathtt{g}) = \mathbf{t}$ then $TVal_{\mathcal{I}}(\mathtt{o}\,\#\,\mathtt{g}) = \mathbf{t}$

Semantics: Interpretation of Formulas (Cont'd)

Definition (Truth valuation, Cont'd)

4. *Externally defined atomic formula*:
   $TVal_{\mathcal{I}}(\texttt{External(t)}) = \boldsymbol{I}_{\text{truth}}(\boldsymbol{I}_{\text{external}}(\texttt{t}))$
5. *Conjunction*: $TVal_{\mathcal{I}}(\texttt{And(}c_1 \ldots c_n\texttt{)}) = \mathbf{t}$
   if and only if $TVal_{\mathcal{I}}(c_1) = \ldots = TVal_{\mathcal{I}}(c_n) = \mathbf{t}$.
   Otherwise, $TVal_{\mathcal{I}}(\texttt{And(}c_1 \ldots c_n\texttt{)}) = \mathbf{f}$.
   Empty conjunction becomes tautology: $TVal_{\mathcal{I}}(\texttt{And()}) = \mathbf{t}$
6. *Disjunction*: $TVal_{\mathcal{I}}(\texttt{Or(}c_1 \ldots c_n\texttt{)}) = \mathbf{f}$
   if and only if $TVal_{\mathcal{I}}(c_1) = \ldots = TVal_{\mathcal{I}}(c_n) = \mathbf{f}$.
   Otherwise, $TVal_{\mathcal{I}}(\texttt{Or(}c_1 \ldots c_n\texttt{)}) = \mathbf{t}$.
   Empty disjunction becomes contradiction: $TVal_{\mathcal{I}}(\texttt{Or()}) = \mathbf{f}$

Semantics: Interpretation of Formulas (Cont'd)

Definition (Truth valuation, Cont'd)

7. *Quantification*:
   - $TVal_{\mathcal{I}}(\texttt{Exists ?v}_1 \ldots \texttt{?v}_n \; (\varphi)) = \mathbf{t}$
     if and only if for some $\mathcal{I}^*$, described below, $TVal_{\mathcal{I}^*}(\varphi) = \mathbf{t}$
   - $TVal_{\mathcal{I}}(\texttt{Forall ?v}_1 \ldots \texttt{?v}_n \; (\varphi)) = \mathbf{t}$
     if and only if for every $\mathcal{I}^*$, described below, $TVal_{\mathcal{I}^*}(\varphi) = \mathbf{t}$

Here $\mathcal{I}^*$ is a semantic structure of the form <$\boldsymbol{TV}$, $\boldsymbol{DTS}$, $\boldsymbol{D}$, $\boldsymbol{D}_{\text{ind}}$, $\boldsymbol{D}_{\text{func}}$, $\boldsymbol{I}_{\text{C}}$, $\boldsymbol{I}^*_{\text{V}}$, $\boldsymbol{I}_{\text{psoa}}$, $\boldsymbol{I}_{\text{sub}}$, $\boldsymbol{I}_{=}$, $\boldsymbol{I}_{\text{external}}$, $\boldsymbol{I}_{\text{truth}}$>, which is exactly like $\mathcal{I}$, except that mapping $\boldsymbol{I}^*_{\text{V}}$, is used instead of $\boldsymbol{I}_{\text{V}}$. $\boldsymbol{I}^*_{\text{V}}$ is defined to coincide with $\boldsymbol{I}_{\text{V}}$ on all variables except, possibly, on $\texttt{?v}_1,\ldots,\texttt{?v}_n$

Semantics: Interpretation of Formulas (Cont'd)

Definition (Truth valuation, Cont'd)

8. *Rule implication*:
   - $TVal_{\mathcal{I}}(conclusion$:- $condition)$ = **t**, if either $TVal_{\mathcal{I}}(conclusion)$ = **t** or $TVal_{\mathcal{I}}(condition)$ = **f**
   - $TVal_{\mathcal{I}}(conclusion$:- $condition)$ = **f** otherwise
9. *Groups of rules*:
   If Γ is a group formula of the form `Group(`φ_1 `...` φ_n`)` then
   - $TVal_{\mathcal{I}}(\Gamma)$ = **t** if and only if $TVal_{\mathcal{I}}(\varphi_1) = \ldots = TVal_{\mathcal{I}}(\varphi_n)$ = **t**
   - $TVal_{\mathcal{I}}(\Gamma)$ = **f** otherwise

   In other words, rule groups are treated as conjunctions □

Conclusion: From Semantics to Implementations

- W3C's RIF-BLD has provided a **reference semantics** for extensions, and for continued efforts, as described here
- Flora 2, OO jDREW, and other engines could be adapted for our PSOA RuleML semantics
- A subset of PSOA RuleML with single-tuple psoa terms has already been prototyped in OO jDREW
- Project with Alexandre Riazanov is **implementing** PSOA RuleML in Vampire Prime via TPTP

Conclusion: Encodings and Alignments

- Future work on psoa terms includes encoding (multi-)slots and slotribution as (multi-)tuples and tupribution
- Conversely, tuples could be encoded as multi-list values of a `tuple` slot
- Web ontologies, especially taxonomies, in OWL 2, RDF Schema, etc. could be reused for PSOA RuleML's OID type systems by alignments rooted in their classes `owl:Thing`, `rdfs:Resource`, etc. and in `Top`

Conclusion: Horn

- Further efforts concern Horn rules
- Notice introductory example is not Horn in that there is a head existential after objectification
- To address this issue, it can be modified as follows

Conclusion: Psoa Rules Made Horn

Example (Rule-extended named family frame)

Horn version of introductory example retrieves `family` frame with named OID variable in premise and uses its binding to extend that frame in conclusion (left: given; right: objectified).

```
Group (
  Forall ?Hu ?Wi ?Ch ?o (
    ?o#family(husb->?Hu
              wife->?Wi
              child->?Ch)      :-
      And(?o#family(husb->?Hu
                    wife->?Wi)
          Or(kid(?Hu ?Ch)
             kid(?Wi ?Ch))) )
  inst4#family(husb->Joe
               wife->Sue)
  kid(Sue Pete)
      )
```

```
Group (
  Forall ?Hu ?Wi ?Ch ?o ?1 ?2 (
    ?o#family(husb->?Hu
              wife->?Wi
              child->?Ch)      :-
      And(?o#family(husb->?Hu
                    wife->?Wi)
          Or(?1#kid(?Hu ?Ch)
             ?2#kid(?Wi ?Ch))) )
  inst4#family(husb->Joe
               wife->Sue)
  _1#kid(Sue Pete)
      )
```

$\rightsquigarrow$ Simpler semantics corresponding to this set of ground facts:

{*inst4#family(husb->Joe wife->Sue child->Pete)*, *_1#kid(Sue Pete)*}