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
Knowledge representation & problem solving in

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

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
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
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
RuleML-2011 paper formalizes positional-slotted, object-applicative (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 *position*al 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 1st argument, and wife, as 2nd 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

\[
\text{family}(\text{husb} \rightarrow \text{Joe} \ \text{wife} \rightarrow \text{Sue}) \text{ or } \text{family}(\text{wife} \rightarrow \text{Sue} \ \text{husb} \rightarrow \text{Joe})
\]

for *family*-typed attribute-value pairs (slots)

\[
\{<\text{husb}, \text{Joe}>, <\text{wife}, \text{Sue}>\}
\]

Easily extended with further slots, e.g. by adding children, as in

\[
\text{family}(\text{husb} \rightarrow \text{Joe} \ \text{wife} \rightarrow \text{Sue} \ \text{child} \rightarrow \text{Pete})
\]
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 $\text{inst1}$ in class $\text{family}$ is describable by $\text{inst1#family(husb->Joe wife->Sue)}$, $\text{inst1#family(husb->Joe wife->Sue child->Pete)}$, etc. Psoa terms can also specialize to class membership terms, e.g. $\text{inst1#family()}$, abridged $\text{inst1#family}$, represents $\text{inst1 }\in \text{family}$.
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 husband and wife slots can be positionalized as in earlier married example: inst1#family(Joe Sue) describes inst1 with tuple [Joe Sue]
Combined positional-slotted psoa terms are allowed, similarly as in XML elements (tuple $\mapsto$ subelements, slots $\mapsto$ attributes), e.g. describing an object, as in RDF descriptions (object $\mapsto$ subject, slots $\mapsto$ properties)

For example, \texttt{inst1#family(Joe Sue child$\rightarrow$Pete)} describes \texttt{inst1} with two positional and one slotted argument
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, $\text{Top}$).

For example, (untyped) frame $\text{inst3}[\text{color->red shape->diamond}]$ in square-bracketed F-logic/RIF syntax is equivalent to our parenthesized $\text{inst3}\#\text{Top(color->red shape->diamond)}$. 
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, $f(\ldots)$, is new, existentially scoped variable $?i$, leading to $\text{Exists } ?i \ (?i#f(\ldots))$; 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 \texttt{family(husb->Joe)} is syntactically objectified to query \texttt{\#family(husb->Joe)}, i.e.
— if \texttt{?1} is first new variable in \texttt{?1, ?2, ...} — to \\
\texttt{?1#family(husb->Joe)}

Posed against fact, it succeeds with first slot, unifying \texttt{?1} with \texttt{_1}

Slotribution (‘slot distribution’) avoids POSL’s ‘rest-slot’ variables: frame’s OID ‘distributes’ over slots of a description
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
Example (Rule-defined anonymous family frame)

Group is used to collect a rule and two facts. For all 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)))  
)  
moved(Joe Sue)  
kid(Sue Pete)  
)

Group (  
  Exists ?1 (  
    family(husb->?Hu wife->?Wi child->?Ch)?1:-  
    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 $D_{ind}$ is to be defined):

$$\{ o#family(husb->Joe wife->Sue child->Pete) \} \cup \{ _1#married(Joe Sue), _2#kid(Sue Pete) \}$$

where $o \in D_{ind}$. 
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
Definition (Alphabet)

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

- A countably infinite set of **constant symbols** $\text{Const}$ (including the root class $\text{Top} \in \text{Const}$ and the positive-integer-enumerated local constants $\_1, \_2, \ldots \in \text{Const}$ as well as individual, function, and predicate symbols)
- A countably infinite set of **variable symbols** $\text{Var}$ (including the positive-integer-enumerated variables $?1, $?2, \ldots \in \text{Var}$)
- The connective symbols $\text{And}$, $\text{Or}$, and $\text{ :- }$
- The quantifiers $\text{Exists}$ and $\text{Forall}$
- The symbols $\text{=}$, $\#$, $\#\#$, $\rightarrow$, $\text{External}$, $\text{Import}$, $\text{Prefix}$, and $\text{Base}$
- The symbols $\text{Group}$ and $\text{Document}$
Definition (Alphabet, Cont’d)

Constants have form "literal" \textsuperscript{symspace}, where literal is a sequence of Unicode characters and symspace is an identifier for a symbol space. E.g., "\_123" \textsuperscript{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" \textsuperscript{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 \rightarrow are used in positional-slotted, object-applicative formulas for class memberships and slots, respectively. Symbol Extern\al 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).
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 $\text{External}(t)$, 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 $t \in \text{Const}$ or $t \in \text{Var}$ then $t$ is a *simple term*

2. **Equality terms.** $t = s$ 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.**
   
   $o#f([t_{1,1} \ldots t_{1,n_1}] \ldots [t_{m,1} \ldots t_{m,n_m}] \ p_1 \rightarrow v_1 \ldots p_k \rightarrow v_k)$
   
   is a *positional-slotted, object-applicative (psoa) term* if
   
   $f \in \text{Const}$ and $o$, $t_{1,1}$, ..., $t_{1,n_1}$, ..., $t_{m,1}$, ..., $t_{m,n_m}$,
   
   $p_1$, ..., $p_k$, $v_1$, ..., $v_k$, $m \geq 0$, $k \geq 0$, are base terms
Psoa terms can be specialized in the following way:

- For $m = 0$ they become **(typed or untyped) frame terms** $\circ \# f(p_1 \rightarrow v_1 \ldots p_k \rightarrow v_k)$. We consider two overlapping subcases:
  - For $k = 0$ they become **class membership terms** $\circ \# f()$, abridged to $\circ \# f$, corresponding to those in F-logic and RIF.
  - For $k \geq 0$ they can be further specialized in two ways, which can be orthogonally combined:
    - For $\circ$ being the anonymous variable $\?$, they become **anonymous frame terms (slotted terms)** $\? \# f(p_1 \rightarrow v_1 \ldots p_k \rightarrow v_k)$, deobjectified $f(p_1 \rightarrow v_1 \ldots p_k \rightarrow v_k)$, corresponding to **terms with named arguments** in RIF.
    - For $f$ being the root class $\text{Top}$, they become **untyped frame terms** $\circ \# \text{Top}(p_1 \rightarrow v_1 \ldots p_k \rightarrow v_k)$ corresponding to **frames** in abridged form $\circ [p_1 \rightarrow v_1 \ldots p_k \rightarrow v_k]$ of F-logic and RIF, where square brackets are used instead of round parentheses.
For $m = 1$ psoa terms become **single-tuple psoa terms**

$$o\#f([t_{1,1} \ldots t_{1,n_1}] p_1 \rightarrow v_1 \ldots p_k \rightarrow v_k),$$

abridged to

$$o\#f(t_{1,1} \ldots t_{1,n_1} p_1 \rightarrow v_1 \ldots p_k \rightarrow v_k).$$

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**

  $$?\#f(t_{1,1} \ldots t_{1,n_1} p_1 \rightarrow v_1 \ldots p_k \rightarrow v_k),$$

  deobjectified $f(t_{1,1} \ldots t_{1,n_1} p_1 \rightarrow v_1 \ldots p_k \rightarrow v_k).$ These can be further specialized:

  - For $k = 0$, they become **positional terms**

    $$?\#f(t_{1,1} \ldots t_{1,n_1}),$$

    deobjectified $f(t_{1,1} \ldots t_{1,n_1})$, corresponding to the usual terms and atomic formulas of classical first-order logic

- For $f$ being the root class $\text{Top}$, they become **untyped single-tuple psoa terms**

  $$o\#\text{Top}(t_{1,1} \ldots t_{1,n_1} p_1 \rightarrow v_1 \ldots p_k \rightarrow v_k).$$

  These can be further specialized:

  - For $k = 0$, they become **untyped single-tuple shelf terms**

    $$o\#\text{Top}(t_{1,1} \ldots t_{1,n_1})$$

    describing object $o$ with positional arguments $t_{1,1}, \ldots, t_{1,n_1}$. 
Externally defined terms. If $t$ is anonymous psoa term
then $\text{External}(t)$ 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.
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 $n_1 = \ldots = n_m$ used to describe a distributed object with ‘homogeneous’ equal-length tuples of a relation: OID names extension of relation’s tuples.
Argument names of psoa terms, \( p_1, \ldots, p_n \), 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 \( \text{ArgNames} \), to reduce unification complexity.

PSOA RuleML could emulate such special treatment of slotted terms by reserving \( \text{ArgNames-style subset of Const} \).

But, since PSOA RuleML’s slotted terms via objectification become frames, they can be queried by slotribution rather than unification.
Any psoa term $f(...)$ or $o \# f(...)$, with $f$ a predicate symbol and $o$ a simple term (constant or variable), is an **atomic formula**. Likewise, externally defined term $\text{External}(\varphi)$, where $\varphi$ 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:
**Definition (Formula, Condition Language, Cont’d)**

- **Conjunction**: If $\varphi_1, ..., \varphi_n, n \geq 0$, are condition formulas then so is $\text{And}(\varphi_1 \ldots \varphi_n)$, called a *conjunctive* formula. As special case, $\text{And}()$ is allowed and treated as tautology, i.e., formula that is always true.

- **Disjunction**: If $\varphi_1, ..., \varphi_n, n \geq 0$, are condition formulas then so is $\text{Or}(\varphi_1 \ldots \varphi_n)$, called a *disjunctive* formula. As special case, $\text{Or}()$ is considered as contradiction, i.e., formula that is always false.

- **Existentials**: If $\varphi$ is a condition formula and $?V_1, ..., ?V_n, n>0$, are distinct variables then $\text{Exists } ?V_1 \ldots ?V_n(\varphi)$ is an *existential* formula.

Condition formulas are intended as premises of rules.
Rule implication: $\varphi : \neg \psi$ is a formula, called **rule implication**, if:

- $\varphi$ is a head formula or a conjunction of head formulas, where a head formula is an atomic formula or an existentially scoped atomic formula,
- $\psi$ is a condition formula, and
- none of the atomic formulas in $\varphi$ is an externally defined term (i.e., term of the form $\text{External}(...)$).

Universal rule: If $\varphi$ is a rule implication and $?V_1, \ldots, ?V_n$, $n>0$, distinct variables then $\forall ?V_1 \ldots ?V_n (\varphi)$ is a universal rule formula. It is required that all free variables in $\varphi$ occur among variables $?V_1 \ldots ?V_n$ in quantification part. Generally, an occurrence of variable $?v$ is free in $\varphi$ if it is not inside subformula of $\varphi$ of the form $\exists ?v (\psi)$ and $\psi$ is a formula. Universal rules are also referred to as **PSOA RuleML rules**.
Universal fact: If \( \varphi \) is an atomic formula and \(?V_1, \ldots, ?V_n, n>0, \) distinct variables then \( \text{Forall} \ ?V_1 \ldots \ ?V_n(\varphi) \) is a universal fact formula, provided that all free variables in \( \varphi \) occur among variables \(?V_1 \ldots \ ?V_n. \)

Universal facts are treated as rules without premises.

Group: If \( \varphi_1, \ldots, \varphi_n \) are PSOA RuleML rules, universal facts, variable-free rule implications or atomic formulas, or groups then \( \text{Group}(\varphi_1 \ldots \ \varphi_n) \) is a group formula. Group formulas represent sets of rules and facts.

Document: An expression of the form

\[ \text{Document}(\text{directive}_1 \ldots \ \text{directive}_n \ \Gamma) \]

is a PSOA RuleML document formula, if

- \( \Gamma \) is an optional group formula; it is called the group formula associated with the document
- \( \text{directive}_1, \ldots, \text{directive}_n \) is optional sequence of directives. Can be base directive, prefix directive or import directive
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
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):

\(?#\text{cpt:book(auth:Newton "Opticks")}\)

Deobjectified version:

\(\text{cpt:book(auth:Newton "Opticks")}\)

Formula that uses an anonymous psoa (slotted term):


Deobjectified version:

\(

Formula that uses a named psoa (typed frame):

\(\text{bks:opt1#cpt:book(cpt:author->auth:Newton cpt:title->"Opticks")}\)

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


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

Example (PSOA RuleML business rule)

Adapts business rule from POSL logistics use case. Ternary `reciship` conclusion represents reciprocal shippings, at total cost (as single positional argument), between source and destination (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, \( \text{?cost} = 57.0 \).

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 {
        ?cost = External(func:numeric-add(?cost1 ?cost2))  
    )
  )

  shipment("PC"^^xs:string "47.5"^^xs:float  

  shipment("PDA"^^xs:string "9.5"^^xs:float  
)}
Example (PSOA RuleML business rule, Cont’d)

The rule can be objectified as follows (Externals are not being transformed):

\[
\text{Forall } ?\text{cost } ?\text{cost1 } ?\text{cost2 } ?A ?B ?2 ?3 ( \\
\quad \text{Exists } ?1 ( ?1\#\text{cpt:reciship}(?\text{cost } \text{cpt:source->} ?A \text{ cpt:dest->} ?B)) : - \\
\quad \quad \text{And}( ?2\#\text{cpt:shipment}(? ?\text{cost1 } \text{cpt:source->} ?A \text{ cpt:dest->} ?B) \\
\quad \quad \quad \quad \quad ?3\#\text{cpt:shipment}(? ?\text{cost2 } \text{cpt:source->} ?B \text{ cpt:dest->} ?A) \\
\quad \quad \quad \quad \quad ?\text{cost} = \text{External}(\text{func:numeric-add}(?\text{cost1 } ?\text{cost2})) ) \\
\]

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

\[
\text{Forall } ?\text{cost } ?\text{cost1 } ?\text{cost2 } ?A ?B ?2 ?3 ( \\
\quad \text{Exists } ?1 ( ?1\#\text{cpt:reciship}(?\text{cost}) \\
\quad \quad \quad ?1\#\text{cpt:reciship}(\text{cpt:source->} ?A) \\
\quad \quad \quad ?1\#\text{cpt:reciship}(\text{cpt:dest->} ?B))) : - \\
\quad \text{And}( ?2\#\text{cpt:shipment}(? ?\text{cost1}) \\
\quad \quad \quad ?2\#\text{cpt:shipment}(\text{cpt:source->} ?A) \\
\quad \quad \quad ?2\#\text{cpt:shipment}(\text{cpt:dest->} ?B) \\
\quad \quad \quad ?3\#\text{cpt:shipment}(? ?\text{cost2}) \\
\quad \quad \quad \quad \quad ?3\#\text{cpt:shipment}(\text{cpt:source->} ?B) \\
\quad \quad \quad \quad \quad \quad ?3\#\text{cpt:shipment}(\text{cpt:dest->} ?A) \\
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{cost} = \text{External}(\text{func:numeric-add}(?\text{cost1 } ?\text{cost2})) ) \\
\]
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
In given document, *new local constant* generator, written as stand-alone \( _i \), denotes first new local constant \( _i, i \geq 1 \), from the sequence \( _1, _2, \ldots \) 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.
Semantics: Truth Values and Valuation

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

This indirectness allows HiLog-like generality
A **semantic structure**, $\mathcal{I}$, is a tuple of the form $<TV, DTS, D, D_{\text{ind}}, D_{\text{func}}, I_C, I_V, I_{\text{psoa}}, I_{\text{sub}}, I =, I_{\text{external}}, I_{\text{truth}}>$

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

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

$D_{\text{ind}}$ is used to interpret elements of $\text{Const}$ acting as individuals

$D_{\text{func}}$ is used to interpret constants acting as function symbols

As before, $\text{Const}$ denotes set of all constant symbols and $\text{Var}$ set of all variable symbols

$DTS$ denotes set of identifiers for primitive datatypes
The other components of $\mathcal{I}$ are total mappings defined thus:

1. $I_C$ maps $\text{Const}$ to $D$. This mapping interprets constant symbols. In addition, it is required that:
   - If a constant, $c \in \text{Const}$, is an \textit{individual} then it is required that $I_C(c) \in D_{\text{ind}}$.
   - If $c \in \text{Const}$ is a \textit{function symbol} then it is required that $I_C(c) \in D_{\text{func}}$.
   - It is required that $I_C(\text{Top}) = \top$.

2. $I_V$ maps $\text{Var}$ to $D_{\text{ind}}$. Mapping interprets variable symbols.
Definition (Semantic structure, Cont’d)

\( I_{\text{psoa}} \) maps \( D \) to total functions \( D_{\text{ind}} \times \text{SetOfFiniteBags}(D^*_{\text{ind}}) \times \text{SetOfFiniteBags}(D_{\text{ind}} \times D_{\text{ind}}) \rightarrow D \). Interprets psoa terms, combining positional, slotted, and frame terms, as well as class memberships. Argument \( d \in D \) of \( I_{\text{psoa}} \) represents function or predicate symbol of positional terms and slotted terms, and object class of frame terms, as well as class of memberships. Element \( o \in D_{\text{ind}} \) represents object of class \( d \), which is described with two bags.

- Finite bag of finite tuples \( \{<t_{1,1}, ..., t_{1,n_1}>, ..., <t_{m,1}, ..., t_{m,n_m}>\} \in \text{SetOfFiniteBags}(D^*_{\text{ind}}) \) , possibly empty, represents positional information. \( D^*_{\text{ind}} \) is set of all finite tuples over the domain \( D_{\text{ind}} \). Bags are used since order of tuples in a psoa term is immaterial and tuples may repeat.

- Finite bag of attribute-value pairs \( \{<a_1, v_1>, ..., <a_k, v_k>\} \in \text{SetOfFiniteBags}(D_{\text{ind}} \times D_{\text{ind}}) \) , possibly empty, represents slotted information. Bags, since order of attribute-value pairs in a psoa term is immaterial and pairs may repeat.
In addition:

3. If $d \in D_{\text{func}}$ then $I_{\text{psoa}}(d)$ must be a ($D_{\text{ind}}$-valued) function $D_{\text{ind}} \times \text{SetOfFiniteBags}(D^*_{\text{ind}}) \times \text{SetOfFiniteBags}(D_{\text{ind}} \times D_{\text{ind}}) \rightarrow D_{\text{ind}}$

4. $I_{\text{sub}}$ gives meaning to the subclass relationship. It is a total mapping of the form $D_{\text{func}} \times D_{\text{func}} \rightarrow D$

5. $I_{=} \text{ is a mapping of the form } D_{\text{ind}} \times D_{\text{ind}} \rightarrow D$. Gives meaning to the equality operator

6. $I_{\text{external}} \text{ is a mapping to give meaning to External terms. Maps external symbols in } \text{Const} \text{ to fixed functions}$

7. $I_{\text{truth}} \text{ is a mapping of the form } D \rightarrow TV$. Used to define truth valuation for formulas
Generic mapping from terms to $D$, denoted by $I$

- $I(k) = I_{C}(k)$, if $k$ is a symbol in $\text{Const}$
- $I(\?v) = I_{V}(\?v)$, if $\?v$ is a variable in $\text{Var}$
- $I(\circ \# f([t_{1,1} \ldots t_{1,n_{1}}] \ldots [t_{m,1} \ldots t_{m,n_{m}}] a_{1} \rightarrow v_{1} \ldots a_{k} \rightarrow v_{k})) = I_{\text{psoa}}(I(f))(I(\circ), \{<I(t_{1,1}), \ldots, I(t_{1,n_{1}})>, \ldots, <I(t_{m,1}), \ldots, I(t_{m,n_{m}})>, <I(a_{1}), I(v_{1})>, \ldots, <I(a_{k}), I(v_{k})>\})$
- $I(c_{1} \# c_{2}) = I_{\text{sub}}(I(c_{1}), I(c_{2}))$
- $I(x = y) = I_{=} (I(x), I(y))$
- $I(\text{External}(p(s_{1} \ldots s_{n}))) = I_{\text{external}}(p)(I(s_{1}), \ldots, I(s_{n}))$

Again {...} denote bags of tuples and attribute-value pairs.
Define mapping, $TVal_\mathcal{I}$, from set of all non-document formulas to $TV$

For atomic formula $\phi$, $TVal_\mathcal{I}(\phi)$ defined essentially as $I_{\text{truth}}(I(\phi))$

Recall that $I(\phi)$ is just an element of domain $D$ and $I_{\text{truth}}$ maps $D$ to truth values in $TV$

Might surprise, since normally mapping $I$ 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 $TV$, not from interpretation domain $D$ to $TV$

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**: $TVal_\mathcal{I}(x = y) = I_{\text{truth}}(I(x = y))$.
   - Required that $I_{\text{truth}}(I(x = y)) = t$ if $I(x) = I(y)$
   - and that $I_{\text{truth}}(I(x = y)) = f$ otherwise
   - This can also be expressed as $TVal_\mathcal{I}(x = y) = t$
     if and only if $I(x) = I(y)$

2. **Subclass**: $TVal_\mathcal{I}(sc \#\# c1) = I_{\text{truth}}(I(sc \#\# c1))$.
   - In particular, for root class, Top, and all $sc \in D$, $TVal_\mathcal{I}(sc \#\# \text{Top}) = t$.
   - To ensure that $\#\#$ is transitive, i.e., $c1 \#\# c2$ and $c2 \#\# c3$
     imply $c1 \#\# c3$, the following is required:
   - For all $c1, c2, c3 \in D$, if $TVal_\mathcal{I}(c1 \#\# c2) = t$
     and $TVal_\mathcal{I}(c2 \#\# c3) = t$ then $TVal_\mathcal{I}(c1 \#\# c3) = t$
Definition (Truth valuation, Cont’d)

**Psoa formula:**

\[ TVal_\mathcal{I}(\circ\#f([t_{1,1}...t_{1,n_1}]...[t_{m,1}...t_{m,n_m}] a_1->v_1...a_k->v_k)) = I_{truth}(I(\circ\#f([t_{1,1}...t_{1,n_1}]...[t_{m,1}...t_{m,n_m}] a_1->v_1...a_k->v_k))). \]

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 \( m \geq 0 \) and \( k \geq 0 \):

- \[ TVal_\mathcal{I}(\circ\#f([t_{1,1}...t_{1,n_1}]...[t_{m,1}...t_{m,n_m}] a_1->v_1...a_k->v_k)) = t \]
  if and only if

- \[ TVal_\mathcal{I}(\circ\#Top([t_{1,1}...t_{1,n_1}])) = ... = TVal_\mathcal{I}(\circ\#Top([t_{m,1}...t_{m,n_m}])) = TVal_\mathcal{I}(\circ\#Top(a_1->v_1)) = ... = TVal_\mathcal{I}(\circ\#Top(a_k->v_k)) = t \]
Observe that on right-hand side of “if and only if” there are $1+m+k$ subformulas splitting left-hand side into an object membership, $m$ object-centered positional formulas, each associating the object with a tuple, and $k$ 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 $o$, which connects subformulas on right-hand side (first subformula providing $o$-member class $f$, remaining $m+k$ ones using root class $\text{Top}$).

For root class, $\text{Top}$, and all $o \in D$, $TVal_{\mathcal{I}}(o \ # \ \text{Top}) = t$.

To ensure that all members of subclass are also members of its superclasses, i.e., $o \ # f$ and $f \ ## g$ imply $o \ # g$, the following restriction is imposed:

For all $o, f, g \in D$, if $TVal_{\mathcal{I}}(o \ # f) = TVal_{\mathcal{I}}(f \ ## g) = t$ then $TVal_{\mathcal{I}}(o \ # g) = t$.
Externally defined atomic formula:
\[ TVal_{\mathcal{I}}(\text{External}(t)) = I_{\text{truth}}(I_{\text{external}}(t)) \]

Conjunction: \[ TVal_{\mathcal{I}}(\text{And}(c_1 \ldots c_n)) = t \]
if and only if \[ TVal_{\mathcal{I}}(c_1) = \ldots = TVal_{\mathcal{I}}(c_n) = t. \]
Otherwise, \[ TVal_{\mathcal{I}}(\text{And}(c_1 \ldots c_n)) = f. \]
Empty conjunction becomes tautology: \[ TVal_{\mathcal{I}}(\text{And}()) = t \]

Disjunction: \[ TVal_{\mathcal{I}}(\text{Or}(c_1 \ldots c_n)) = f \]
if and only if \[ TVal_{\mathcal{I}}(c_1) = \ldots = TVal_{\mathcal{I}}(c_n) = f. \]
Otherwise, \[ TVal_{\mathcal{I}}(\text{Or}(c_1 \ldots c_n)) = t. \]
Empty disjunction becomes contradiction: \[ TVal_{\mathcal{I}}(\text{Or}()) = f \]
Definition (Truth valuation, Cont’d)

**Quantification:**

- \( TVal_I(\text{Exists } ?v_1 \ldots ?v_n (\phi)) = t \)
  if and only if for some \( I^* \), described below, \( TVal_{I^*}(\phi) = t \)

- \( TVal_I(\text{Forall } ?v_1 \ldots ?v_n (\phi)) = t \)
  if and only if for every \( I^* \), described below, \( TVal_{I^*}(\phi) = t \)

Here \( I^* \) is a semantic structure of the form \(<TV, DTS, D, D_{ind}, D_{func}, I_C, I^*_V, I_{psoa}, I_{sub}, I_-, I_{external}, I_{truth}>\), which is exactly like \( I \), except that mapping \( I^*_V \), is used instead of \( I_V \). \( I^*_V \) is defined to coincide with \( I_V \) on all variables except, possibly, on \( ?v_1, \ldots, ?v_n \).
Definition (Truth valuation, Cont’d)

8 Rule implication:
- \( TVal_I(\text{conclusion} :- \text{condition}) = t \), if either
  \( TVal_I(\text{conclusion}) = t \) or \( TVal_I(\text{condition}) = f \)
- \( TVal_I(\text{conclusion} :- \text{condition}) = f \) otherwise

9 Groups of rules:
If \( \Gamma \) is a group formula of the form \( \text{Group}(\varphi_1 \ldots \varphi_n) \) then
- \( TVal_I(\Gamma) = t \) if and only if \( TVal_I(\varphi_1) = \ldots = TVal_I(\varphi_n) = t \)
- \( TVal_I(\Gamma) = f \) otherwise

In other words, rule groups are treated as conjunctions
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.
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.
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) 
)

⇝

Simpler semantics corresponding to this set of ground facts:

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