RuleML for Object-Relational Knowledge Representation on the Web

International Joint Conference on Artificial Intelligence, IJCAI 2011 Best Papers from Sister Conferences Track, Barcelona, Spain, July 19-22, 2011

Harold Boley

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

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

- 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**

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

Introduction: Distinctions in Psoa Taxonomy

- 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)`

- 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]`

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

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)`

- 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)* } $\cup$

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

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

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.* `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.*
   $\mathtt{o\#f([t_{1,1}...t_{1,n_1}]...[t_{m,1}...t_{m,n_m}]\ p_1\text{->}v_1...p_k\text{->}v_k)}$
   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)

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

  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*** $\texttt{?\#f(t}_{1,1} \ldots \texttt{t}_{1,n_1}\texttt{ p}_1\texttt{->v}_1 \ldots \texttt{p}_k\texttt{->v}_k\texttt{)}$, deobjectified $\texttt{f(t}_{1,1} \ldots \texttt{t}_{1,n_1}\texttt{ p}_1\texttt{->v}_1 \ldots \texttt{p}_k\texttt{->v}_k\texttt{)}$. These can be further specialized:
    - For `k = 0`, they become ***positional terms*** $\texttt{?\#f(t}_{1,1} \ldots \texttt{t}_{1,n_1}\texttt{)}$, deobjectified $\texttt{f(t}_{1,1} \ldots \texttt{t}_{1,n_1}\texttt{)}$, 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*** $\texttt{o\#Top(t}_{1,1} \ldots \texttt{t}_{1,n_1}\texttt{ p}_1\texttt{->v}_1 \; \ldots \; \texttt{p}_k\texttt{->v}_k\texttt{)}$. These can be further specialized:
    - For `k = 0`, they become ***untyped single-tuple shelf terms*** $\texttt{o\#Top(t}_{1,1} \ldots \texttt{t}_{1,n_1}\texttt{)}$ describing object `o` with positional arguments $\texttt{t}_{1,1}, \ldots, \texttt{t}_{1,n_1}$

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*.

- 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

Semantics: Semantic Structures

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

DTS denotes set of identifiers for primitive datatypes

Semantics: Semantic Structures (Cont'd)

Definition (Semantic structure, Cont'd)

3. $\boldsymbol{I}_{\text{psoa}}$ maps $\boldsymbol{D}$ to total functions $\boldsymbol{D}_{\text{ind}} \times \texttt{SetOfFiniteBags}(\boldsymbol{D^*}_{\text{ind}}) \times \texttt{SetOfFiniteBags}(\boldsymbol{D}_{\text{ind}} \times \boldsymbol{D}_{\text{ind}}) \rightarrow \boldsymbol{D}$. Interprets psoa terms, combining positional, slotted, and frame terms, as well as class memberships. Argument $\mathtt{d} \in \boldsymbol{D}$ 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 $\mathtt{o} \in \boldsymbol{D}_{\text{ind}}$ represents object of class $\mathtt{d}$, which is described with two bags.
   - Finite bag of finite tuples $\{<\mathtt{t}_{1,1}, ..., \mathtt{t}_{1,n_1}>, ..., <\mathtt{t}_{m,1}, ..., \mathtt{t}_{m,n_m}>\} \in \texttt{SetOfFiniteBags}(\boldsymbol{D^*}_{\text{ind}})$, 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 $\{<\mathtt{a1}, \mathtt{v1}>, ..., <\mathtt{ak}, \mathtt{vk}>\} \in \texttt{SetOfFiniteBags}(\boldsymbol{D}_{\text{ind}} \times \boldsymbol{D}_{\text{ind}})$, 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)

Generic mapping from terms to ***D***, denoted by ***I***

- $\boldsymbol{I}(\texttt{k}) = \boldsymbol{I}_C(\texttt{k})$, if `k` is a symbol in `Const`
- $\boldsymbol{I}(\texttt{?v}) = \boldsymbol{I}_V(\texttt{?v})$, if `?v` is a variable in `Var`
- $\boldsymbol{I}(\ \texttt{o\#f([t}_{1,1}\ \ldots\ \texttt{t}_{1,n_1}\texttt{]}\ \ldots\ \texttt{[t}_{m,1}\ \ldots\ \texttt{t}_{m,n_m}\texttt{]}\ \texttt{a}_1\texttt{->v}_1\ \ldots\ \texttt{a}_k\texttt{->v}_k\texttt{)}\ )$
  $= \boldsymbol{I}_{psoa}(\boldsymbol{I}(\texttt{f}))(\boldsymbol{I}(\texttt{o}), \{<\boldsymbol{I}(\texttt{t}_{1,1}), \ldots, \boldsymbol{I}(\texttt{t}_{1,n_1})>, \ldots, <\boldsymbol{I}(\texttt{t}_{m,1}), \ldots, \boldsymbol{I}(\texttt{t}_{m,n_m})>\},$
  $\{<\boldsymbol{I}(\texttt{a}_1),\boldsymbol{I}(\texttt{v}_1)>, \ldots, <\boldsymbol{I}(\texttt{a}_k),\boldsymbol{I}(\texttt{v}_k)>\}\ )$
  Again {...} denote *bags* of tuples and attribute-value pairs.
- $\boldsymbol{I}(\texttt{c1\#\#c2}) = \boldsymbol{I}_{sub}(\boldsymbol{I}(\texttt{c1}), \boldsymbol{I}(\texttt{c2}))$
- $\boldsymbol{I}(\texttt{x=y}) = \boldsymbol{I}_{=}(\boldsymbol{I}(\mathrm{x}), \boldsymbol{I}(\mathrm{y}))$
- $\boldsymbol{I}(\texttt{External(p(s}_1\texttt{...s}_n\texttt{)))} = \boldsymbol{I}_{external}(\texttt{p})(\boldsymbol{I}(\texttt{s}_1), \ldots, \boldsymbol{I}(\texttt{s}_n))$

Semantics: Method of Formula Interpretation

- Define mapping, $\mathit{TVal}_{\mathcal{I}}$, from set of all non-document formulas to ***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 ***D*** and $\boldsymbol{I}_{\text{truth}}$ maps ***D*** to truth values in ***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 $TVal_{\mathcal{I}}$:

3. *Psoa formula*:
   $TVal_{\mathcal{I}}(\texttt{o\#f([t}_{1,1}\texttt{...t}_{1,n_1}\texttt{]...[t}_{m,1}\texttt{...t}_{m,n_m}\texttt{] a}_1\texttt{->v}_1\texttt{...a}_k\texttt{->v}_k\texttt{)}) =$
   $\boldsymbol{I}_{\mathrm{truth}}(\boldsymbol{I}(\texttt{o\#f([t}_{1,1}\texttt{...t}_{1,n_1}\texttt{]...[t}_{m,1}\texttt{...t}_{m,n_m}\texttt{] a}_1\texttt{->v}_1\texttt{...a}_k\texttt{->v}_k\texttt{)})).$
   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 $\texttt{k}\geq\texttt{0}$:

   - $TVal_{\mathcal{I}}(\texttt{o\#f([t}_{1,1}\texttt{...t}_{1,n_1}\texttt{]...[t}_{m,1}\texttt{...t}_{m,n_m}\texttt{] a}_1\texttt{->v}_1\texttt{...a}_k\texttt{->v}_k\texttt{)}) = \mathbf{t}$
     if and only if
     $TVal_{\mathcal{I}}(\texttt{o \# f}) =$
     $TVal_{\mathcal{I}}(\texttt{o\#Top([t}_{1,1}\texttt{...t}_{1,n_1}\texttt{])})=...=TVal_{\mathcal{I}}(\texttt{o\#Top([t}_{m,1}\texttt{...t}_{m,n_m}\texttt{])}) =$
     $TVal_{\mathcal{I}}(\texttt{o\#Top(a}_1\texttt{->v}_1\texttt{)}) = \ldots = TVal_{\mathcal{I}}(\texttt{o\#Top(a}_k\texttt{->v}_k\texttt{)}) =$
     $\mathbf{t}$

Semantics: Interpretation of Formulas (Cont'd)

Definition (Truth valuation, Cont'd)

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

In other words, rule groups are treated as conjunctions $\square$

Conclusion: From Semantics to Implementations

- W3C's RIF-BLD has provided a **reference semantics** for extensions, and for continued efforts, as described here
- Project with Alexandre Riazanov is **implementing** PSOA RuleML in Vampire Prime via TPTP

- 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)
      )
```

⇝ Simpler semantics corresponding to this set of ground facts:

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