

Integrating Positional and Slotted Knowledge on the Semantic Web

Harold Boley

Semantic Web Laboratory

Institute for Information Technology – e-Business, National Research Council of Canada

Fredericton, NB, E3B 9W4, Canada

<http://www.cs.unb.ca/~boley>

CASCON Workshop on Applications of Automated Reasoning, Toronto, October 7th, 2004

Revised for: XML Clearinghouse Colloquium, Institut für Informatik, FU Berlin, March 15th, 2005

'Human-Oriented' POSL ↔ 'Machine-Oriented' RuleML

- ▷ POSL integrates **p**ositional and **s**lotted knowledge for humans (e.g.: Prolog's positional and F-logic's slotted knowledge)
- ▷ OO RuleML marks up this knowledge for machines
- ▷ POSL ↔ OO RuleML translators in OO jDREW and as servlets:
 - ▷ **Parser:** `http://www.ruleml.org:8080/converters/servlet/AsciiToRuleML`
 - ▷ **Generator:** `http://www.ruleml.org:8080/converters/servlet/RuleMLToAscii`

Advantages of 'Human-Oriented' Web Knowledge Syntax

- ▷ Allow knowledge shorthand, presentation, and (even) exchange
- ▷ Study expressive classes and formal semantics (cf. OWL)
- ▷ Develop knowledge bases and parse into XML markup (cf. N3):
 - ▷ Parser reads for XML-aware tools
 - ▷ Generator prints for stack-limited humans
- ▷ Use case **New Brunswick Business Knowledge Base (NBBizKB):**

<http://www.scs.carleton.ca/~diis/arise/Presentations/HaroldBoley.pdf>

Semantic Web Language Design Space

- ▷ Object-centered instance descriptions via binary properties (RDF)
- ▷ Taxonomies over classes and properties (RDFS)
- ▷ Class-forming operations and class/property axioms (OWL DL)
- ▷ Derivation, integrity, transformation, and reaction rules (RuleML)

Integrations of Semantic Web Languages

- ▷ Object-centered descriptions plus rules (N3, OO RuleML)
- ▷ Description logic plus rules (Description Logic Programs, SWRL)

↪ Web information integration

E.g.: Mapping object-centered representations to positional ones

Orthogonal, Integrated Design for POSL

- ▷ Orthogonal ('decoupled') dimensions for systematic language development
- ▷ Incorporate above notions so they can be used and revised independently

Prolog and F-logic Integrated in POSL

- ▷ Both predated the (Semantic) Web, yet have been very useful for it
 - ▷ Prolog: Positional language based on Horn logic with facts and rules
 - ▷ F-logic: Slotted language with object-centered descriptions and rules
- ▷ Concise ASCII syntaxes, elegant semantics, and decent computational properties
- ▷ Often needed conjointly in the XML&RDF Web

Positional Notations

- ▷ Ordered sequences of possibly repeating objects
- ▷ In logics used for the arguments to n-ary relations
- ▷ E.g.: `shipment` relation with ordered arguments `cargo`, `price`, `source`, and `destination`
- ▷ POSL uses Prolog-like syntax, e.g. for ground facts:

```
shipment (PC, 47.5, BostonMoS, LondonSciM) .
```

```
shipment (PDA, 9.5, LondonSciM, BostonMoS) .
```


Slotted Notations

- ▷ Unordered sets of attribute-value pairs
- ▷ In frame logics used for molecular formulas
- ▷ E.g.: `shipment` relation as slotted frame, with unordered *slot names* such as `cargo`
- ▷ POSL uses F-logic-inspired syntax, obtaining these facts:

```
shipment (cargo->PC;price->47.5;source->BostonMoS;dest->LondonSciM) .  
shipment (cargo->PDA;price->9.5;source->LondonSciM;dest->BostonMoS) .
```

Positional-Slotted Notations

- ▷ Ordered and unordered combined
- ▷ In Lisp used for some functions
- ▷ E.g.: `shipment` relation with two positional arguments, `cargo` and `price`, and two slots, `source` and `destination`
- ▷ POSL uses Prolog/F-logic-combining syntax, obtaining these facts:

```
shipment (PC, 47.5; source->BostonMoS; dest->LondonSciM) .
```

```
shipment (PDA, 9.5; source->LondonSciM; dest->BostonMoS) .
```

Complex Terms and Plexes

- ▷ All three notations are possible for any complex term (*cterm*)
- ▷ *plex* regarded as special case of a constructorless *cterm*
- ▷ E.g.: Pair of stakeholders (“[...]” for constructor applications):

<i>notation</i>	<i>cterm</i>	<i>plex</i>
positional	stakepair[MM, SS]	[MM, SS]
slotted	stakepair[owner->MM; shipper->SS]	[owner->MM; shipper->SS]
positional-slotted	stakepair[MM; shipper->SS]	[MM; shipper->SS]

Non-Ground Formulas for the Three Notations

▷ Variable arguments interpreted as: universally (existentially) quantified in facts (queries)

▷ Variables can be named (prefix “?”) or anonymous (stand-alone “?”)

▷ E.g.: Non-ground query of earlier positional shipment ground fact:

`shipment(PC, ?, BostonMoS, ?goal)`

`succeeds, binding ?goal to LondonSciM`

Rest Arguments – Basics

- ▷ Rests permitted for (*polyadic*) atoms
 - ▷ One for positional arguments, one for slotted arguments
- ▷ Positional arguments separated from positional rest by “|”
- ▷ Slotted arguments separated from slotted rest by “!”
- ▷ Rest itself normally a variable, for varying number of arguments
- ▷ ‘Fixed-arity/polyadic’ is orthogonal to ‘positional/slotted’

Rest Arguments – Anonymous

- ▷ Anonymous variable usable as positional or slotted “don’t care” rest
- ▷ Slotted “don’t care” rest “!?” makes option from F-logic’s convention: to tolerate arbitrary *excess slots* in either formula (e.g., a fact), having slot names not used by any slot of the other (“!?”-)formula (e.g., a query), for unification

Rest Arguments – Examples (I)

For the earlier slotted PC-shipment fact

```
shipment (cargo->PC;price->47.5;source->BostonMoS;dest->LondonSciM) .
```

▷ the query

```
shipment (cargo->?what;price->?;source->BostonMoS;dest->?goal)
```

succeeds, binding ?what to PC and ?goal to LondonSciM

▷ However, the query

```
shipment (owner->?who;cargo->?;price->?;source->BostonMoS;dest->?)
```

fails because of its excess slot named owner

Rest Arguments – Examples (II)

Similarly, for the earlier slotted PC-shipment fact

```
shipment (cargo->PC;price->47.5;source->BostonMoS;dest->LondonSciM) .
```

▷ the query

```
shipment (cargo->?what;source->BostonMoS;dest->?goal)
```

fails because of the fact's excess slot named `price`

▷ On the other hand, the query

```
shipment (cargo->?what;source->BostonMoS;dest->?goal!?)
```

again succeeds with initial bindings, since slotted “rest doesn't care”,

“!?”, unifies `price` slot (independent of where it occurs in fact)

Rest Arguments – Examples (III)

- ▷ Conversely, earlier fact would tolerate excess query slots such as in above `owner` query after making it non-ground via anonymous rest:
`shipment (cargo->PC;price->47.5;source->BostonMoS;dest->LondonSciM!?) .`
- ▷ If query also contains anonymous rest, both it and the fact can contain excess slots, as in
`shipment (owner->?who;cargo->?what;source->BostonMoS;dest->?goal!?)`
which succeeds with initial bindings, since query rest unifies fact's `price` slot and fact rest unifies query's `owner` slot, leaving variable `?who` free, and querier agnostic about the owner

Rest Arguments – Novelty



- ▷ If anonymous rest slots are employed in all formulas, effect of F-logic's implicit rest variables is obtained
- ▷ More precise, “!”-free slotted formulas can enforce more restricted unifications where needed

Rest Arguments – Unification

- ▷ “|” and “!” rests can follow after zero or more fixed positional and slotted arguments
- ▷ Unify the zero or more remaining arguments
- ▷ Before being bound to a variable, polyadic rest e_1, \dots, e_Z or $s_1 \rightarrow f_1; \dots; s_Z \rightarrow f_Z$ made into single complex term, namely plex $[e_1, \dots, e_Z]$ or $[s_1 \rightarrow f_1; \dots; s_Z \rightarrow f_Z]$, respectively

Atom and Cterm Syntax Summary

With both kinds of rests, these are the most general (non-normal) forms of positional-slotted atoms and cterms (for normal forms all slots go to the right):

`r (s1->f1; ...; sL->fL; e1, ..., eM | Ve; sL+1->fL+1; ...; sN->fN! Vf)`

`c [s1->f1; ...; sL->fL; e1, ..., eM | Ve; sL+1->fL+1; ...; sN->fN! Vf]`

Semantics of Atoms and Cterms – Instantiation & Equality

Based on slotted extensions to the positional (here, LP) notions of clause instantiation and ground equality (model-theoretic semantics) as well as unification (proof-theoretic semantics)

- ▷ *Slotted instantiation* recursively walks through fillers of slots, substituting dereferenced values from substitution (environment) for any variables encountered
- ▷ *Slotted ground equality* recursively compares two ground atoms or cterms after lexicographic sorting of slots encountered

Semantics of Atoms and Cterms – Unification

Slotted unification performs sorting, uses the slotted instantiation of variables, and otherwise proceeds left-to-right as for positional unification,

- ▷ pairing up identical slot names before recursively unifying their fillers,
- ▷ while collecting excess slots on each level in the plex value of corresponding slotted rest variable

Positional Rules

- ▷ Horn rules, in POSL written using Prolog-like syntax, but again employing “?”(-prefixed) variables
- ▷ `reciship` example starts as Datalog rule for reciprocal shippings of unspecified cargos at a total cost between two sites:

```
reciship(?cost, ?A, ?B) :-  
    shipment(?, ?cost1, ?A, ?B),  
    shipment(?, ?cost2, ?B, ?A),  
    add(?cost, ?cost1, ?cost2).
```

Variable Typing

- ▷ Types can be defined as RDFS or OWL classes
- ▷ **Use types** Float, Address, **and** Product **in** reciship **rule:**

```
reciship(?cost:Float, ?A:Address, ?B:Address) :-  
    shipment(?:Product, ?cost1:Float, ?A, ?B),  
    shipment(?:Product, ?cost2:Float, ?B, ?A),  
    add(?cost, ?cost1, ?cost2) .
```


Slotted Rules

- ▷ Much like in F-logic (typing could be added, as above)
- ▷ reciship relation with slot names price, site1, and site2. Analogously, add relation with slot names sum, addend1, and addend2:

```
reciship(price->?cost;site1->?A;site2->?B) :-  
    shipment(cargo->?;price->?cost1;source->?A;dest->?B),  
    shipment(cargo->?;price->?cost2;source->?B;dest->?A),  
    add(sum->?cost;addend1->?cost1;addend2->?cost2) .
```

Positional-Slotted Rules

- ▷ Positional and slotted relations *for* conclusion or premises, or positional-slotted relations *within* conclusion or premises
- ▷ `reciship` rule can be positional for conclusion and add premise, and slotted for the `shipment` premises:

```
reciship(?cost, ?A, ?B) :-  
    shipment(cargo->?;price->?cost1;source->?A;dest->?B),  
    shipment(cargo->?;price->?cost2;source->?B;dest->?A),  
    add(?cost, ?cost1, ?cost2).
```

Semantics of (Positional-)Slotted Clause Sets

On top of the earlier semantic basis for atoms and complex terms

- ▷ On clause level, three notations have same interpretation, hence earlier treatment naturally extends to (positional-)slotted generalizations of positional (LP) clauses
- ▷ Further semantic treatment via Herbrand models and resolution proof theory directly follows positional treatment
- ▷ Typing (sorts) can be reduced to unsorted case

Implementation of POSL's (Positional-)Slotted Clauses

- ▷ OO jDREW: Ball05 has realized semantics via adaptation of Java-based jDREW interpreter by Spencer02
- ▷ Freely available via Java Web Start and for JAR download:
www.jdrew.org/oojdrew
- ▷ Begun to adapt sorted indexing techniques to RDFS and OO jDREW

Applications of POSL

- ▷ Product-seeking/advertising trees in the tree-similarity-based AgentMatcher system
- ▷ Music filtering rules in the collaborative system RACOFI Music
- ▷ Business-analysis and data-integration rules in the above-mentioned NBBizKB use case

POSL Webizing

- ▷ POSL language elements can be given URIs: individuals (and constructors), relations, slots, and types
- ▷ Occurrences of the same language element can thus be disambiguated
- ▷ Orthogonal to the positional/slotted distinction

URIs in POSL

- ▷ An (active) URI is enclosed in a pair of angular brackets, `<...>`, following IETF's generic URI syntax
- ▷ Symbolic language element occurrences can be associated with URIs via juxtaposition: *symbol*`<...>`
- ▷ Symbols can also be entirely replaced by URIs
- ▷ Symbols can still be used without URIs

Webized Individuals

- ▷ URIs in place of, or in addition to, individual-constant symbols
- ▷ Can use URI for the intended SpeedShip company's homepage
`<http://sship.com>`
- ▷ employed in place of the individual symbol, as practiced in RDF, N3, and other Web languages (here, first argument of a 5-ary fact):
`shipment (<http://sship.com>, PC, 47.5, BostonMoS, LondonSciM) .`
- ▷ or, associated with it:
`shipment (SpeedShip<http://sship.com>, PC, 47.5, BostonMoS, LondonSciM) .`

Webized Relations

- ▷ URIs in place of, or in addition to, symbolic relation names
- ▷ The 4-ary and 5-ary positional `shipment` relations can be uniquely distinguished via URIs pointing to different signatures:

`shipment<http://transport.org/rels/pos/shipment#4>`

`shipment<http://transport.org/rels/pos/shipment#5>`

Webized Slots

- ▷ URIs in place of, as pioneered by RDF, or in addition to, symbolic slot names
- ▷ `shipment` slots may be drawn from URIs containing fragmentid's `#id` with slot names, except for `charge` fragmentid, for which local slot name `price` is kept:

```
shipment (<http://transport.org/slots/shipment#shipper>->SpeedShip;  
         <http://transport.org/slots/shipment#cargo>->PC;  
         price<http://ebizguide.org/slots#charge>->47.5;  
         <http://trajectory.org/slots/movement#source>->BostonMoS;  
         <http://trajectory.org/slots/movement#dest>->LondonSciM) .
```

Webized Types

- ▷ URI references to an RDFS or OWL class
- ▷ Product type can be associated with a URI for the corresponding OWL class:

```
Product<http://www.daml.org/services/owl-s/1.0/  
ProfileHierarchy.owl#Product>
```

Web-Typed Rule Example

Use `Product<...>` for typing anonymous variable of earlier positional rule, `Float` from XML Schema Datatypes for its `cost`-like variables, and webized `Address` type:

```
reciship(?cost:Float<http://www.w3.org/TR/2001/
          REC-xmlschema-2-20010502/#float>,
         ?A:<http://ebizguide.org/types#Address>,
         ?B:<http://ebizguide.org/types#Address>)
                                                    :-
shipment (?:Product<http://www.daml.org/services/owl-s/1.0/
          ProfileHierarchy.owl#Product>,
         ?cost1:Float<http://www.w3.org/TR/2001/
          REC-xmlschema-2-20010502/#float>,
         ?A, ?B),
                                                    . . . .
```

Anchored POSL Atoms

- ▷ Webizing is also possible for entire atoms, as a way of associating them with names
- ▷ Fact atom can be *anchored* by OID (symbolic name or URI, possibly prefixed by symbolic name)
- ▷ Special ‘zeroth’ argument separated from further arguments by hat infix “^”: *relation(oid^arg₁...arg_N)*

Anchoring Examples

Earlier 4-ary positional and slotted facts (see “%” comments) can now be anchored using variously webized versions of names like `s1` and `s2`:

```
shipment (s1^PC, 47.5, BostonMoS, LondonSciM) .           % positional
shipment (<http://sship.com/event#s2>^PDA, 9.5, LondonSciM, BostonMoS) .
shipment (s1<http://sship.com/event#s1>^                 % slotted
        cargo->PC;price->47.5;
        source->BostonMoS;dest->LondonSciM) .
shipment (<http://sship.com/event#s2>^
        ...)
```

RDF Descriptions as Anchored Facts

- ▷ RDF descriptions can be conceived as anchored slotted POSL facts
- ▷ In the absence of `rdf:type` these facts have null relation
- ▷ For the following comparison assume `shipper slot etc.` determine `shipment relationship`, so no relation is needed

Comparison: RDF Description ...

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://transport.org/slots/shipment#"
  xmlns:p="http://ebizguide.org/slots#"
  xmlns:m="http://trajectory.org/slots/movement#">
  <rdf:Description about="http://sship.com/event#s1">
    <s:shipper rdf:resource="http://sship.com"/>
    <s:cargo>PC</s:cargo>
    <p:charge>47.5</p:charge>
    <m:source rdf:resource="http://www.mos.org/info/contact.html"/>
    <m:dest rdf:resource="http://www.sciencemuseum...location.asp"/>
  </rdf:Description>
</rdf:RDF>
```


Comparison: ... POSL Fact

```
(<http://sship.com/event#s1>^  
<http://transport.org/slots/shipment#shipper>->  
    <http://sship.com>;  
<http://transport.org/slots/shipment#cargo>->PC;  
<http://ebizguide.org/slots#charge>->47.5;  
<http://trajectory.org/slots/movement#source>->  
    <http://www.mos.org/info/contact.html>;  
<http://trajectory.org/slots/movement#dest>->  
    <http://www.sciencemuseum...location.asp>).
```

Symbolic and webized individuals are represented in the same manner here, so that symbolic name like PC can later be replaced by blank node or URI, without changing enclosing slot

RDF Blank Nodes as POSL Skolem Constants

- ▷ Blank nodes are used for OIDs local to current document
- ▷ For example, earlier shipping description can be refined by referring to a local cargo description using blank node identifier `PeterMillerPC`
- ▷ In the following again compare RDF and POSL

Comparison: RDF Blank Node ..

```
<rdf:RDF
  ...
  <rdf:Description about="http://sphen.com/event#s1">
    ...
    <s:cargo rdf:nodeID="PeterMillerPC"/>
    ...
  </rdf:Description>
  <rdf:Description rdf:nodeID="PeterMillerPC">
    <p:value>2500.0</p:value>
    <p:weight>17.5</p:weight>
  </rdf:Description>
</rdf:RDF>
```

Comparison: ... POSL Skolem Constant

```
{  
  (<http://sship.com/event#s1>^  
    ...  
    <http://transport.org/slots/shipment#cargo>->_PeterMillerPC;  
    ...).  
  (_PeterMillerPC^  
    <http://ebizguide.org/slots#value>->2500.0;  
    <http://ebizguide.org/slots#weight>->17.5).  
}
```

Module “{...}” of two facts connected by an existential variable,
in POSL a local Skolem constant (global to clauses), `_PeterMillerPC`

Generating New Skolem Constants

- ▷ Module-scoped, *unique Skolem constants* can be generated by *New Skolem constant* primitive (written as a stand-alone “_”)
- ▷ All occurrences “_”, “_”, ... semantically replaced by fresh constants $_{-1}, _{-2}, \dots$
- ▷ Model theory for (New) Skolem constants in rules has been developed on top of *anonymous-domain-augmented Herbrand universe* by Yang&Kifer03

RDF-Like Rule Example in POSL

Earlier slotted rule modified to query such facts, inferring, as new “_”-anchored atoms, OIDs and aggregated cost of reciprocal shippings (webized slot names abridged using symbolic names):

```
reciship(_^forth->?oid1;back->?oid2;
         price->?cost;site1->?A;site2->?B)
                                     :-
  (?oid1^shipper->;cargo->;price->?cost1;source->?A;dest->?B),
  (?oid2^shipper->;cargo->;price->?cost2;source->?B;dest->?A),
  add(sum->?cost;addend1->?cost1;addend2->?cost2) .
```

Notice that `?oid1/?oid2` variables occur in two roles: to the left of “^”, as proper OIDs, and to the right of “^”, as ordinary data values

Metadata Deduction Rules

- ▷ In bottom-up derivations, “_” of conclusion generates fresh Skolem constants, obtaining facts such as `reciship(_4711^...)`.
- ▷ Such rules can be employed within semantic search engine on RDF/POSL-described metadata for high-precision results
- ▷ E.g.: Priced pairs of Web objects about A-to-B and B-to-A shippings

Conclusions

- ▷ Introduced kernel of positional and slotted notions plus notations for knowledge on Semantic Web
- ▷ Extensions in online POSL document www.ruleml.org/submission/ruleml-shortation.html
- ▷ Current work concerns general POSL treatment of slot cardinalities (cf. exact, min, and max cardinality restrictions in OWL DL)
- ▷ Future research on extending OIDs for general object identity:
From OO rules to OOP-like reaction rules and Web Services