PSOA2Prolog: Object-Relational Rule Interoperation and Implementation by Translation from PSOA RuleML to ISO Prolog

The 9th International Web Rule Symposium, RuleML 2015
August 2-5, 2015

Gen Zou Harold Boley

Faculty of Computer Science,
University of New Brunswick, Fredericton, Canada

Outline

Outline

Introduction

- **Rule paradigms** on the Semantic Web and in AI
  - Relational: FOL, Prolog
  - Object-centered: RDF, N3
  - **Combined**
    - Heterogenous: F-logic, RIF
      Allow atomic formulas in separate relational and object-centered paradigms for data as well as rules, also mixed within same rule
    - **Homogeneous**: PSOA RuleML
      Generalize atomic relational and object-centered formulas into uniform kind of atom

Introduction

- To create a joint interoperation & implementation path for PSOA RuleML, we developed the PSOA2Prolog translator from PSOA RuleML to a subset of the relational ISO Prolog
- PSOA2Prolog is composed of source-to-source normalizer and mapper to pure Prolog (Horn logic) subset of ISO Prolog
- PSOA2Prolog supports KBs and queries employing all major PSOA features but restricting the use of equality to external-function evaluation
- Our PSOATransRun[PSOA2Prolog, XSBProlog] engine for PSOA RuleML combines the PSOA2Prolog translator with the XSB Prolog runtime system

PSOA RuleML – Principles

- Uses **p**ositional-**s**lotted **o**bject-**a**pplicative (**psoa**) terms, permitting a relation application to have an OID – typed by the relation – and, independently, its arguments to be positional or slotted.

  General case (multi-tuple):

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

  Special cases (optional single-tuple brackets and zero-argument parentheses):

  Combined: $o\#f([t_1 \ldots t_n]\ p_1{->}v_1 \ldots p_k{->}v_k)$
  Positional: $o\#f([t_1 \ldots t_n])$
  Slotted: $o\#f(\ p_1{->}v_1 \ldots p_k{->}v_k)$
  Member-only: $o\#f()$

- For an **anonymous psoa term**, without a 'user' OID, *objectification* will introduce a 'system' OID
- Psoa terms without class typing is expressed by `Top`, specifying the root of class hierarchy

PSOA RuleML – Presentation Syntax (PS)

Integrates RIF PS for relations and frames (see above)

As in RIF PS:

- "*oid* is member of *class*" written as "*oid*#*class*" (see above)
- "$class_1$ is subclass of $class_2$" written as "$class_1$ ## $class_2$"
- Local constants and relations prefixed by underscore ('_'); variables prefixed by question mark ('?')

Startup Example (in PS)

KB:

```
Document (
  Group (
    Forall ?X ?Y ?Z ?EX ?EY (
      _startup(?X ?Y _employee->?Z) :-
         And(_cofounders(?X ?Y) _hire(?X ?Z)
             _equity(?X ?EX) _equity(?Y ?EY)
             External(
               pred:numeric-less-than-or-equal(
                 External(func:numeric-add(?EX ?EY)) 100)))
    )

    _cofounders(_Ernie _Tony)          _hire(_Ernie _Kate)
    _equity(_Ernie 50)                 _equity(_Tony 30)
    _startup##_company
  )
)
```

Query: `_startup(?X ?Y _employee->?Z)` `_company(?X ?Y)`
Answer: `?X=_Ernie ?Y=_Tony ?Z=_Kate` `?X=_Ernie ?Y=_Tony`

ISO Prolog

- International standard for the widely used logic programming language Prolog
- We focus on the Horn subset of ISO Prolog, which excludes extended features like Negation as failure
- Horn ISO subset is also a subset of XSB Prolog

Outline

Normalization

- Objectification
- Skolemization
- Slotribution and Tupribution
- Flattening
- Rule Splitting

Objectification

- Introduce an OID for each OID-less atomic psoa formula `p(...)`

  For a formula in the KB:
  - Ground fact
    `_`i`#p(...)`, `_`i is a **new constant**, $\mathtt{i} = 1, 2, \ldots$
  - Non-ground fact or in rule conclusion
    `Exists` *`?j`* `(?j#p(...))`
  - In rule premise
    `?j``#p(...)`, where *`?j`* is a **new variable** in the universal scope of the enclosing rule

  For a formula in the query: `Exists` *`?j`* `(?j#p(...))`

Startup Example (After Objectification)

KB:

```
Document (
  Group (
    Forall ?X ?Y ?Z ?EX ?EY ?2 ?3 ?4 ?5 (
      Exists ?1 (
        ?1#_startup(?X ?Y _employee->?Z)) :-
          And(?2#_cofounders(?X ?Y) ?3#_hire(?X ?Z)
              ?4#_equity(?X ?EX) ?5#_equity(?Y ?EY)
              External(
                pred:numeric-less-than-or-equal(
                  External(func:numeric-add(?EX ?EY)) 100)))
    )
    _1#_cofounders(_Ernie _Tony)      _2#_hire(_Ernie _Kate)
    _3#_equity(_Ernie 50)             _4#_equity(_Tony 30)
    _startup##_company
  )
)
```

Objectified Query:

```
Exists ?1 (?1#_company(?X ?Y))
```

Skolemization

- Specialized FOL Skolemization is employed to eliminate existentials in rule conclusions or facts, which are not allowed in Prolog
- Replace each existential formula `Exists ?X (`σ`)` in a rule conclusion or a fact with σ[`?X/_skolem`k`(?`v_1 `...` `?`v_m`)`], where each occurrence of `?X` in σ becomes a Skolem function `_skolem`k applied to all universally quantified variables `?`v_1 `...` `?`v_m from the clause's quantifier prefix
- **New skolem function name** `_skolem`k ($k = 1, 2, \ldots$) generated for each existential variable

Startup Example (After Skolemization)

Skolemized KB:

```
Document (
  Group (
    Forall ?X ?Y ?Z ?EX ?EY ?2 ?3 ?4 ?5 (
      _skolem1(?X ?Y ?Z ?EX ?EY ?2 ?3 ?4 ?5)
          #_startup(?X ?Y _employee->?Z) :-
          And(?2#_cofounders(?X ?Y) ?3#_hire(?X ?Z)
              ?4#_equity(?X ?EX) ?5#_equity(?Y ?EY)
              External(
                pred:numeric-less-than-or-equal(
                  External(func:numeric-add(?EX ?EY)) 100)))
    )
    _1#_cofounders(_Ernie _Tony)     _2#_hire(_Ernie _Kate)
    _3#_equity(_Ernie 50)            _4#_equity(_Tony 30)
    _startup##_company
  )
)
```

Query:

```
Exists ?1 (?1#_company(?X ?Y))
```

Slotribution and Tupribution

- Rewrite each psoa formula

  $o\#f([t_{1,1} \ldots t_{1,n_1}] \ldots [t_{m,1} \ldots t_{m,n_m}]$
  $p_1 \texttt{->} v_1 \ldots p_k \texttt{->} v_k)$

  into a conjunction

  $\texttt{And}(o\#f$
  $o\#\texttt{Top}(t_{1,1} \ldots t_{1,n_1}) \ldots o\#\texttt{Top}(t_{m,1} \ldots t_{m,n_m})$
  $o\#\texttt{Top}(p_1 \texttt{->} v_1) \ldots o\#\texttt{Top}(p_k \texttt{->} v_k))$

Startup Example (After Slotribution and Tupribution)

```
Document(
  Group (
    Forall ?X ?Y ?Z ?EX ?EY ?2 ?3 ?4 ?5 (
         And(_skolem1(?X ?Y ?Z ?EX ?EY ?2 ?3 ?4 ?5)#_startup
             _skolem1(...)#Top(?X ?Y)
             _skolem1(...)#Top(_employee->?Z)) :-
         And(?2#_cofounders ?2#Top(?X ?Y)  ?3#_hire ?3#Top(?X ?Z)
             ?4#_equity  ?4#Top(?X ?EX) ?5#_equity ?5#Top(?Y ?EY)
             External(
               pred:numeric-less-than-or-equal(
                 External(func:numeric-add(?EX ?EY)) 100)))
    )

    _1#_cofounders          _1#Top(_Ernie _Tony)
    _2#_hire                _2#Top(_Ernie _Kate)
    _3#_equity              _3#Top(_Ernie 50)
    _4#_equity              _4#Top(_Tony 30)
    _startup##_company
  )
)
```

Slotributed/Tupributed Query:

```
Exists ?1 (And(?1#_company ?1#Top(?X ?Y)))
```

Flattening

- Extract each embedded interpreted function application as a separate equality
- Each atomic formula φ (in a rule premise or a query) that embeds an external function application ψ, which is not on the top level of an equality, is replaced with `And(?i=`ψ $\varphi[\psi/$`?i])`, where `?i` is the first variable in `?1`, `?2`, ... that does not occur in the enclosing rule

Startup Example (After Flattening)

KB:

```
Document(
  Group (
    Forall ?X ?Y ?Z ?EX ?EY ?2 ?3 ?4 ?5 ?6 (
         And(_skolem1(?X ?Y ?Z ?EX ?EY ?2 ?3 ?4 ?5)#_startup
             _skolem1(...)#Top(?X ?Y)
             _skolem1(...)#Top(_employee->?Z)) :-
         And(?2#_cofounders ?2#Top(?X ?Y)  ?3#_hire ?3#Top(?X ?Z)
             ?4#_equity  ?4#Top(?X ?EX) ?5#_equity ?5#Top(?Y ?EY)
             And(?6=External(func:numeric-add(?EX ?EY))
                 External(pred:numeric-less-than-or-equal(?6 100))))
    )

    _1#_cofounders        _1#Top(_Ernie _Tony)
    _2#_hire              _2#Top(_Ernie _Kate)
    _3#_equity            _3#Top(_Ernie 50)
    _4#_equity            _4#Top(_Tony 30)
    _startup##_company
  )
)
```

Query:

```
Exists ?1 (And(?1#_company ?1#Top(?X ?Y)))
```

Rule Splitting

- Remove conjunctions in rule conclusions, which are not supported in Prolog
- Each rule with an *n*-ary conjunction in the conclusion

  $\texttt{Forall } ?v_1 \ \ldots \ ?v_m \ (\texttt{And}(\varphi_1 \ \ldots \ \varphi_n) \ \texttt{:-} \ \varphi')$

  is split into *n* rules

  $\texttt{Forall } ?v_1 \ \ldots \ ?v_m \ (\varphi_1 \ \texttt{:-} \ \varphi')$

  $\ldots$

  $\texttt{Forall } ?v_1 \ \ldots \ ?v_m \ (\varphi_n \ \texttt{:-} \ \varphi')$

Startup Example (After Rule Splitting)

KB:

```
Document(
  Group (
    Forall ?X ?Y ?Z ?EX ?EY ?2 ?3 ?4 ?5 ?6 (
      _skolem1(?X ?Y ?Z ?EX ?EY ?2 ?3 ?4 ?5)#_startup :-
        And(?2#_cofounders ?2#Top(?X ?Y)  ?3#_hire ?3#Top(?X ?Z)
            ?4#_equity  ?4#Top(?X ?EX) ?5#_equity ?5#Top(?Y ?EY)
            And(?6=External(func:numeric-add(?EX ?EY))
                External(pred:numeric-less-than-or-equal(?6 100)))
    )
    Forall ?X ?Y ?Z ?EX ?EY ?2 ?3 ?4 ?5 ?6 (
      _skolem1(...)#Top(?X ?Y) :- And(...)
    )
    Forall ?X ?Y ?Z ?EX ?EY ?2 ?3 ?4 ?5 ?6 (
      _skolem1(...)#Top(_employee->?Z) :- And(...)
    )
    _1#_cofounders          _1#Top(_Ernie _Tony)
    _2#_hire                _2#Top(_Ernie _Kate)
    _3#_equity              _3#Top(_Ernie 50)
    _4#_equity              _4#Top(_Tony 30)
    _startup##_company
  )
)
```

Outline

Constants and Variables

Use recursive mapping function denoted by ρ_{psoa}

- Constants
  - If c is a number, $\rho_{psoa}(c)$ is the corresponding Prolog number
  - If c is an arithmetic built-in, $\rho_{psoa}(c)$ is the corresponding Prolog built-in
  - Otherwise, $\rho_{psoa}(c)$ is the single-quoted version of c
- Variables
  For a '?'-prefixed variable v, $\rho_{psoa}(v)$ replaces '?' with the upper-case letter 'Q' (Question mark)

Central PSOA Constructs

Table: Mapping from PSOA/PS constructs to Prolog constructs

PSOA/PS Constructs	Prolog Constructs
o#Top(t$_1$...t$_k$)	tupterm(ρ_{psoa}(o), ρ_{psoa}(t$_1$) ... ρ_{psoa}(t$_k$))
o#Top(p->v)	sloterm(ρ_{psoa}(o), ρ_{psoa}(p), ρ_{psoa}(v))
o#c()	memterm(ρ_{psoa}(o), ρ_{psoa}(c))
f(t$_1$...t$_k$)	ρ_{psoa}(f)(ρ_{psoa}(t$_1$), ..., ρ_{psoa}(t$_k$))
And(f$_1$ ... f$_n$)	(ρ_{psoa}(f$_1$) , ... , ρ_{psoa}(f$_n$))
Or(f$_1$ ... f$_n$)	(ρ_{psoa}(f$_1$) ; ... ; ρ_{psoa}(f$_n$))
Exists ?v$_1$ ... ?v$_m$ (φ)	$\rho_{psoa}(\varphi)$
Forall ?v$_1$ ... ?v$_m$ (φ)	$\rho_{psoa}(\varphi)$
φ :- ψ	$\rho_{psoa}(\varphi)$:- $\rho_{psoa}(\psi)$.
?v=External(f(t$_1$...t$_k$))	is(ρ_{psoa}(?v), ρ_{psoa}(f)(ρ_{psoa}(t$_1$), ..., ρ_{psoa}(t$_k$)))
c1##c2	memterm(X,ρ_{psoa}(c2)) :- memterm(X,ρ_{psoa}(c1)).

Mapped Startup Example

Translated KB:

```
memterm('_skolem1'(QX,QY,QZ,QEX,QEY,Q2,Q3,Q4,Q5),'_startup') :-
 (((memterm(Q2,'_cofounders'),tupterm(Q2,QX,QY)),
   (memterm(Q3,'_hire'),tupterm(Q3,QX,QZ)),
   (memterm(Q4,'_equity'),tupterm(Q4,QX,QEX)),
   (memterm(Q5,'_equity'),tupterm(Q5,QY,QEY)),
   (is(Q0,'+'(QEX,QEY)),'=<'(Q0,100)))).
tupterm('_skolem1'(...),QX,QY) :- (...).
sloterm('_skolem1'(...),'_employee',QX) :- (...).
memterm('_1','_cofounders').   tupterm('_1','_Ernie','_Tony').
memterm('_2','_hire').         tupterm('_2','_Ernie','_Kate').
memterm('_3','_equity').       tupterm('_3','_Ernie',50).
memterm('_4','_equity').       tupterm('_4','_Tony',30).
memterm(X,'_company') :- memterm(X,'_startup').
```

Translated Query:

```
(memterm(Q1,'_company'),tupterm(Q1,QX,QY)).
```

Answer in Prolog:

```
Q1='_skolem1'(...), QX='_Ernie', QY='_Tony'
```

Outline

Realization

- We implemented PSOA2Prolog in Java using ANTLR v3
- Translation chain starts with parsing PSOA/PS input into Abstract Syntax Trees (ASTs), then performs normalization steps on AST representations of PSOA sources, and finally maps the AST representation into ISO Prolog syntax
- PSOA2Prolog and XSB Prolog are composed into the instantiation PSOATransRun[PSOA2Prolog, XSBProlog] of our PSOATransRun framework

Evaluation

- Compared with previous TPTP instantiation PSOATransRun[PSOA2TPTP,VampirePrime]
- First evaluation: a test suite of 30 test cases and 90 queries, covering all PSOA features that we have implemented
- Each test case consists of one KB, multiple queries and user-provided answers to each query
- Prolog instantiation passed all 30 test cases, while the TPTP instantiation failed on 10 test cases which contain features that it currently does not support: external functions, subclass formulas, and IRI constants

Evaluation

- Second evaluation: size-parameterized test cases `chain(`k`)` for performance testing
- Each call of `chain(`k`)` generates a KB consisting of one fact `_r0(_a1 _a2 _a3)` and k rules of the form
  `Forall ?X ?Y ?Z (`
  `_r`i`(?X ?Y ?Z) :- _r`$i-1$`(?X ?Y ?Z)`
  `)`, $i = 1, \ldots, k$
- The query is `_r`k`(?X ?Y ?Z)`, which has one answer
  `?X=_a1 ?Y=_a2 ?Z=_a3`
- We measured the average query execution time of the two instantiations on ten test cases, starting with $k = 20$ and increasing in steps of 20 rules until reaching $k = 200$

Evaluation

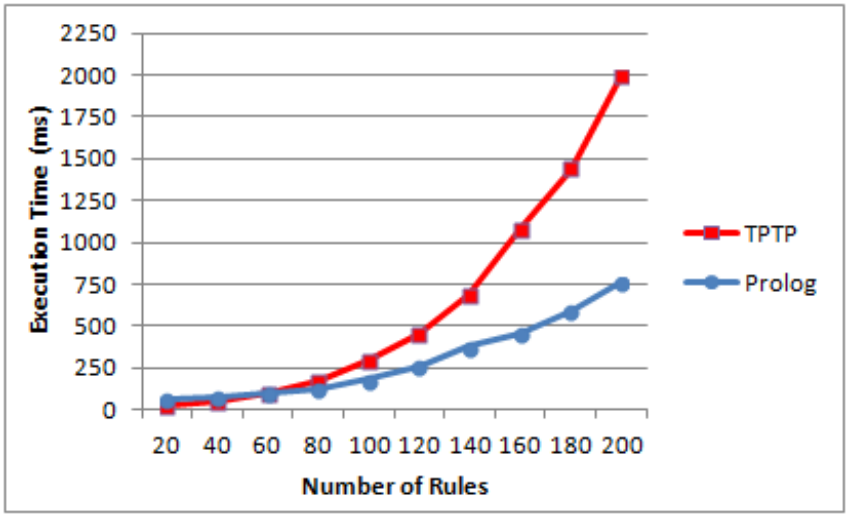

Figure: Performance of PSOATransRun instantiations on ten `chain(`k`)` test cases

Outline

Summary

- We realized a PSOA2Prolog translator as a five-step source-to-source normalizer followed by a mapper to a pure (Horn) subset of ISO Prolog
- PSOATransRun[PSOA2Prolog, XSBProlog] composition provides efficient query answering for PSOA RuleML
- It outperforms our earlier PSOATransRun instantiation targeting TPTP on large test cases

Future Work

- Detailed comparisons regarding the functionality and performance of comparable subsets of PSOA RuleML and (1) 'native' ISO Prolog and (2) F-logic
- Exploring further relational translation targets besides TPTP and Prolog
- Realizing translations between PSOA RuleML and object-centered languages such as N3
- Studying subsets with interesting properties, e.g. function-free PSOA RuleML, and its connection to Datalog$^{\pm}$
- Versions of PSOA2Prolog that implement equality for user-defined functions and translate to an expanded set of built-ins