Interpreting SWRL Rules in RDF Graphs

WLFM 2005, 19 July 2005

Jing Mei[1], Harold Boley[2]
[1]Peking University, China
[2]National Research Council of Canada

Contents

1 SWRL (Semantic Web Rule Language)

- SWRL ≅ OWL + Unary/Binary Datalog RuleML

  $H_1 \wedge \ldots \wedge H_n \leftarrow B_1 \wedge \ldots \wedge B_m$

  consequent ← antecedent

where H_i and B_j are atoms of the form C(u) or P(u,v)

C = OWL class:	A |	(atomic concept)
	T |	(universal concept)
	$\bot$ |	(bottom concept)
	$\neg D$ |	(complementOf)
	$C_1 \wedge \ldots \wedge C_n$ |	(intersectionOf)
	$C_1 \vee \ldots \vee C_n$ |	(unionOf)
	$\exists P.D$ |	(someValuesFrom)
	$\forall P.D$ |	(allValuesFrom)
	$=nP$, $\leq nP$, $\geq nP$	(cardinality)

P = OWL property, having its property as:

[Symmetric], [Functional], [InverseFunctional], [Transitive], [inverseOf]

u, v = OWL constants or SWRL variables

- Undecidable

1.1 SWRL in Abstract Syntax

- A SWRL document is an OWL ontology extended with rule axioms

```
axiom        ::=    rule
rule         ::=    'Implies(' [ URIreference ] { annotation } antecedent consequent ')'
antecedent ::=      'Antecedent(' { atom } ')'
consequent ::=      'Consequent(' { atom } ')'
atom         ::=    description '(' i-object ')'
                    | dataRange '(' d-object ')'
                    | individualvaluedPropertyID '(' i-object i-object ')'
                    | datavaluedPropertyID '(' i-object d-object ')'
                    | sameAs '(' i-object i-object ')'
                    | differentFrom '(' i-object i-object ')'
                    | builtIn '(' builtinID { d-object } ')'
builtinID    ::=    URIreference
i-object     ::=    i-variable | individualID
d-object     ::=    d-variable | dataLiteral
i-variable   ::=    'I-variable(' URIreference ')'
d-variable   ::=    'D-variable(' URIreference ')'
```

- Example of a rule Abs-hasUncle:

hasUncle(x, z) ← hasParent(x, y), hasBrother(y, z)

```
Implies(foo:r
  Antecedent(
    foo:hasParent(I-variable(foo:x)  I-variable(foo:y))
    foo:hasBrother(I-variable(foo:y)  I-variable(foo:z))
  )
  Consequent(
    foo:hasUncle(I-variable(foo:x)  I-variable(foo:z))
  )
)
```

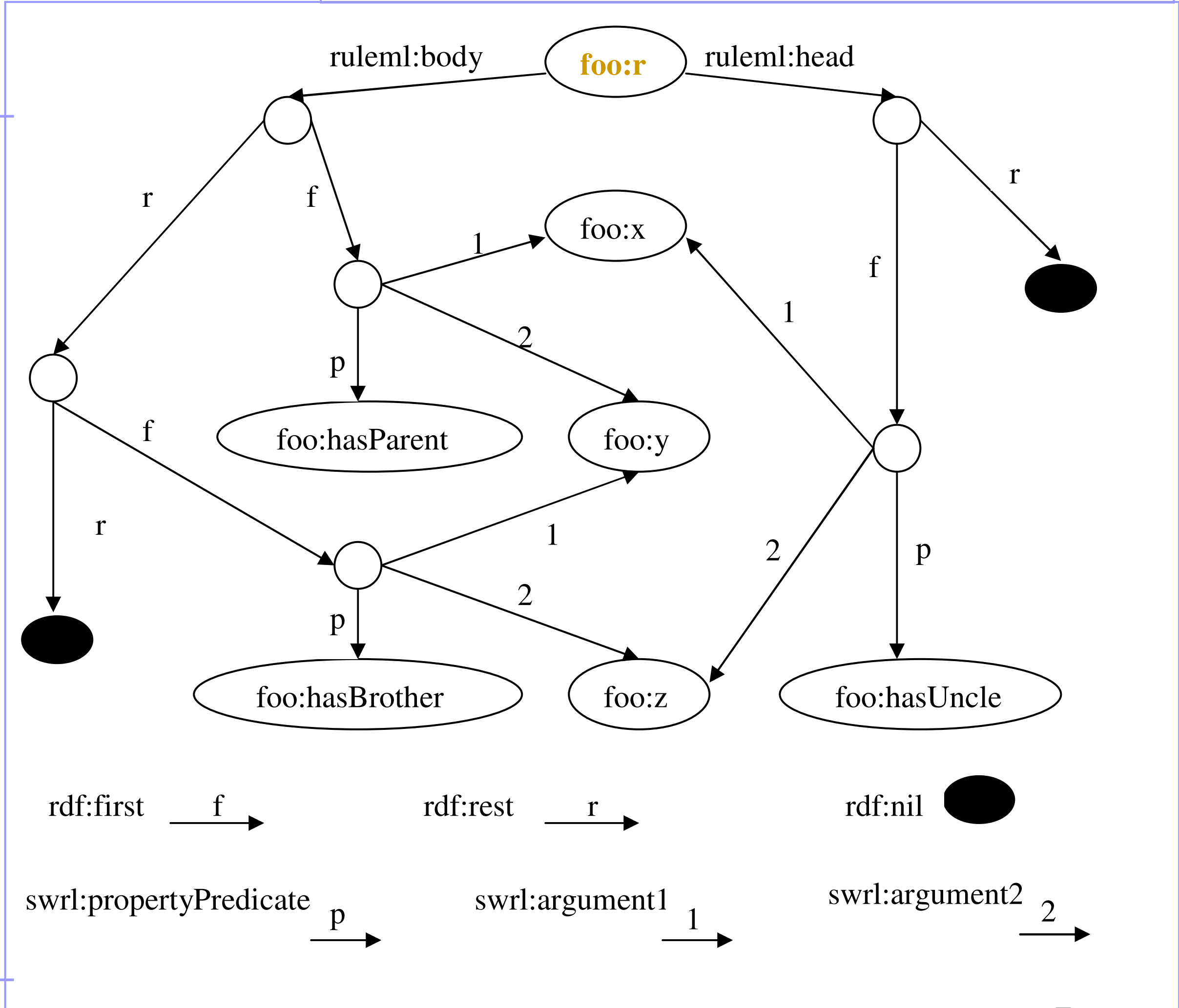
1.2 SWRL in RDF Syntax

- RDF triples: “subject predicate object .”

hasParent(x, y), hasBrother(y, z) → hasUncle(x, z)

Example of a rule Def-hasUncle:

subject	predicate	object
foo:r	ruleml:body	_:b
foo:r	ruleml:head	_:h
_:b	rdf:first	_:ap
_:b	rdf:rest	_:l
_:l	rdf:first	_:ab
_:l	rdf:rest	rdf:nil
_:ap	swrl:propertyPredicate	foo:hasParent
_:ap	swrl:argument1	foo:x
_:ap	swrl:argument2	foo:y
_:ab	swrl:propertyPredicate	foo:hasBrother
_:ab	swrl:argument1	foo:y
_:apb	swrl:argument2	foo:z
_:h	rdf:first	_:au
_:h	rdf:rest	rdf:nil
_:au	swrl:propertyPredicate	foo:hasUncle
_:au	swrl:argument1	foo:x
_:au	swrl:argument2	foo:z



2 Interpreting SWRL Rules in RDF Graphs

- A transformer T*: abstract syntax ⇨ RDF triples
- An RDF-compatible interpretation I: RDF resources ⇨ the domain of I
- SWRL Full: meta-modeling, as RDF(S) or OWL Full does
- SWRL non-Full: a separation of the domain of discourse into disjoint parts
- SWRL DL-safe: each variable in a rule ⇨ an explicitly named constant
- Implementation:
  an RDF database (such as Sesame) + Bottom-up Datalog engine
- Related work:
  - KAON2: DL ⇨ disjunctive Datalog
  - Hoolet: DL & rules ⇨ first-order formulas
  - SWRLJessTab: Racer (for DL) + Jess (for rules)

2.1 Transformer

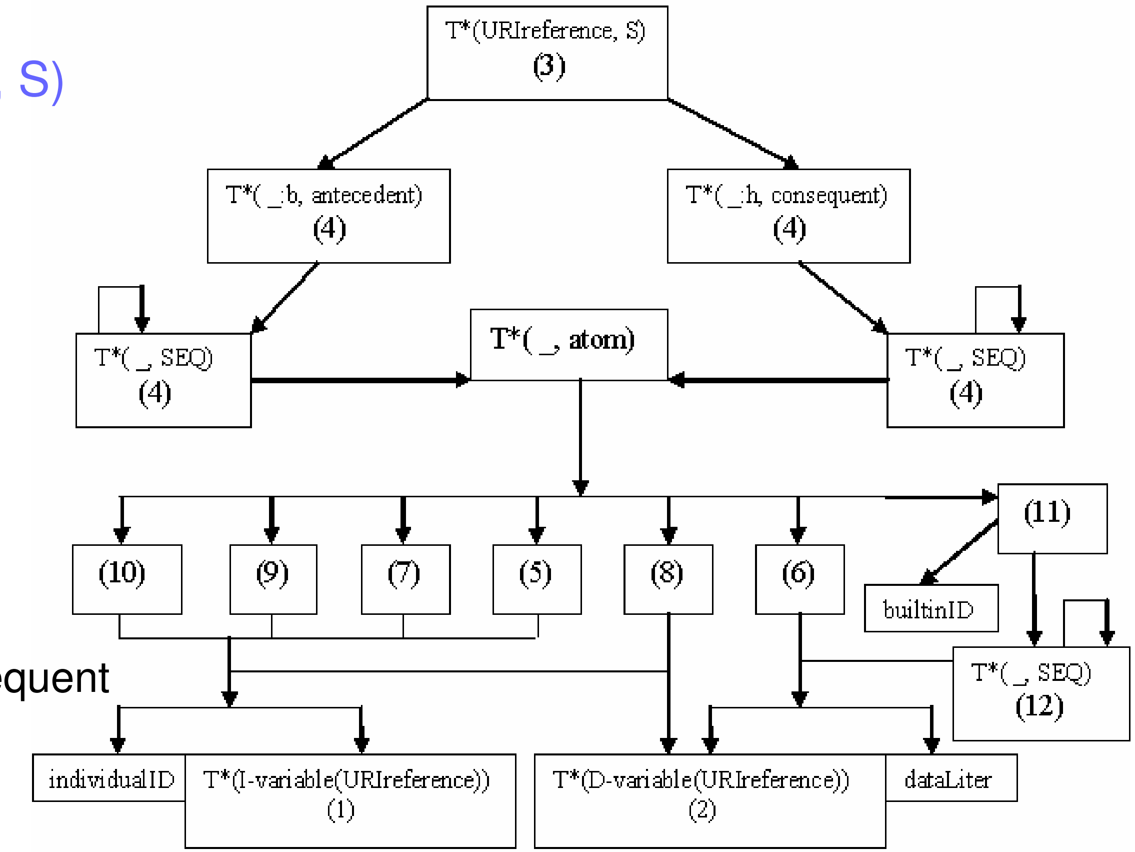
Top-level call: T*(URIreference, S)

S = Implies([uri]

Antecedent(antecedent)
Consequent(consequent))

with a fixed uri = URIreference

By induction on the structure of S, the transformer T* is proved correct



Binary T*(uri, S) returns the uri:
(3) for rules
(4) for sequences of antecedent/consequent
(5) for ClassAtom
(6) for DataRangeAtom
(7) for IndividualPropertyAtom
(8) for DatavaluedPropertyAtom
(9) for SameIndividualAtom
(10) for DifferentIndividualsAtom
(11) for BuiltinAtom
(12) for sequences of built-in objects

Unary T*(S) returns the URIreference:
(1) for individual variables
(2) for data-literal variables
or T*(S) returns itself for
S = individualID, dataLiteralID, builtinID

2.2 Interpretation

- R_I: the domain of discourse or universe
- S_I: URIreference ⇨ R_I
- L_I: typed literal ⇨ R_I
- EXT_I: the extension of an RDF property
- $CEXT_I$: the extension of an RDFS class

// rule, variable, atom, etc.

- IRR = $CEXT_I$ (S_I (ruleml:Implies))
- IRV = $CEXT_I$ (S_I (swrl:Variables))
- IRA = $CEXT_I$ (S_I (swrl:Atom))
- IRB = $CEXT_I$ (S_I (swrl:Builtin))

// constant

- IOT = $CEXT_I$ (S_I (owl:Thing))
- LV_I = $CEXT_I$ (S_I (rdfs:Literal))

// unary predicate

- IOC = $CEXT_I$ (S_I (owl:Class))
- IDC = $CEXT_I$ (S_I (rdfs:Datatype))

// binary predicate

- IOOP = $CEXT_I$ (S_I (owl:ObjectProperty))
- IODP = $CEXT_I$ (S_I (owl:DatatypeProperty))

Semantic conditions for atoms & variables

if	then
$< a,p > \in EXT_I(S_I(\text{swrl:classPredicate}))$	$a \in CEXT_I(S_I(\text{swrl:ClassAtom})) \subseteq IRA$
$< a,i > \in EXT_I(S_I(\text{swrl:argument1}))$	$p \in IOC,\ i \in IOT \cup IRV$
$< a,p > \in EXT_I(S_I(\text{swrl:dataRange}))$	$a \in CEXT_I(S_I(\text{swrl:DataRangeAtom})) \subseteq IRA$
$< a,d > \in EXT_I(S_I(\text{swrl:argument1}))$	$p \in IDC,\ d \in LV_I \cup IRV$
$< a,p > \in EXT_I(S_I(\text{swrl:propertyPredicate}))$	$a \in CEXT_I(S_I(\text{swrl:IndividualPropertyAtom})) \subseteq IRA$
$< a,u > \in EXT_I(S_I(\text{swrl:argument1}))$	$p \in IOOP,\ u \in IOT \cup IRV,\ v \in IOT \cup IRV$; OR
$< a,v > \in EXT_I(S_I(\text{swrl:argument2}))$	$a \in CEXT_I(S_I(\text{swrl:DatavaluedPropertyAtom})) \subseteq IRA$
	$p \in IODP,\ u \in IOT \cup IRV,\ v \in LV_I \cup IRV$
$< a,b > \in EXT_I(S_I(\text{swrl:builtin}))$	$a \in CEXT_I(S_I(\text{swrl:BuiltinAtom})) \subseteq IRA$
$< a,v > \in EXT_I(S_I(\text{swrl:arguments}))$	v is a sequence of $v_1, ..., v_l$ over $LV_I \cup IRV$
	$b \in IRB,\ < v_1, ..., v_l > \in D(b)$
$a \in CEXT_I(S_I(\text{swrl:SameIndividualAtom}))$	$< a, S_I(\text{owl:sameAs}) > \in EXT_I(S_I(\text{swrl:propertyPredicate}))$
$a \in CEXT_I(S_I(\text{swrl:DifferentIndividualsAtom}))$	$< a, S_I(\text{owl:differentFrom}) > \in EXT_I(S_I(\text{swrl:propertyPredicate}))$

2.2 Interpretation (cont'd)

- M_R: IRR $\longrightarrow \{<v_1, \ldots, v_t> \mid$
  $v_k \in$ IRV, $1 \le k \le t$ and $v_i \ne v_j$, $1 \le i < j \le t$ where $0 \le t \le |$IRV$|$ }

- M_A: IRV $\cup LV_I \cup$ IOT $\longrightarrow N \cup \{0\}$

From an argument in an atom to its position (for variables) or zero (for constants)

- M_B: $N \times$ IRV $\longrightarrow LV_I \cup$ IOT

Binding the variable at position k to a certain constant

- M_B: $\{0\} \times LV_I \longrightarrow LV_I$

Mapping a literal to itself

- M_B: $\{0\} \times$ IOT $\longrightarrow$ IOT

Mapping an individual to itself

- "Implies":

if <r, h> $\in$ EXT_I (S_I (ruleml:head)) and <r, b> $\in$ EXT_I (S_I (ruleml:body)),
then r $\in$ IRR, h is not empty,
 and h is a sequence of $h_1, \ldots, h_n$ over IRA,
 and b is a sequence of $b_1, \ldots, b_m$ over IRA,
 for each possible binding M_B (k, v_k) $\in LV_I \cup$ IOT
 where M_R (r) = <$v_1, \ldots, v_t$>, $1 \le k \le t$, $0 \le t \le |$IRV$|$,
 if b is empty or $1 \le j \le m$, b_j is true, then $1 \le i \le n$, h_i is true

2.2 Interpretation (cont'd)

- The definition for an atom being true:

Definition: $a \in$IRA is true iff

(1) $< a, p > \in \text{EXT}_I(\text{S}_I(\text{swrl:classPredicate}))$ and $< a, i > \in \text{EXT}_I(\text{S}_I(\text{swrl:argument1}))$ and $\text{M}_B(\text{M}_A(i), i) \in \text{CEXT}_I(p)$

(2) $< a, p > \in \text{EXT}_I(\text{S}_I(\text{swrl:dataRange}))$ and $< a, d > \in \text{EXT}_I(\text{S}_I(\text{swrl:argument1}))$ and $\text{M}_B(\text{M}_A(d), d) \in \text{CEXT}_I(p)$

(3) $< a, p > \in \text{EXT}_I(\text{S}_I(\text{swrl:propertyPredicate}))$ and $< a, u > \in \text{EXT}_I(\text{S}_I(\text{swrl:argument1}))$ and $< a, v > \in \text{EXT}_I(\text{S}_I(\text{swrl:argument2}))$ and $< \text{M}_B(\text{M}_A(u), u), \text{M}_B(\text{M}_A(v), v) > \in \text{EXT}_I(p))$

- The substitution for SLD resolution:

  $\sigma = \{v \rightarrow c \mid v \in \text{IRV}, c = \text{M}_B(\text{M}_A(v), v) \in \text{LV}_I \cup \text{IOT}\}$

2.2 Interpretation (cont'd)

Example of
Def-hasUncle
Def-hasNiece

A binding established in a rule is undone for other rules

Rule:

Def-hasUncle: hasUncle(x,z) ← hasParent(x,y), hasBrother(y,z)

Def-hasNiece: hasNiece(y,z) ← hasSibling(y,x), hasDaughter(x,z)

Facts:

<mj *hasParent* mdg> <mdg *hasBrother* mdq>

<mdq *hasSibling* mdg> <mdg *hasDaughter* mj>

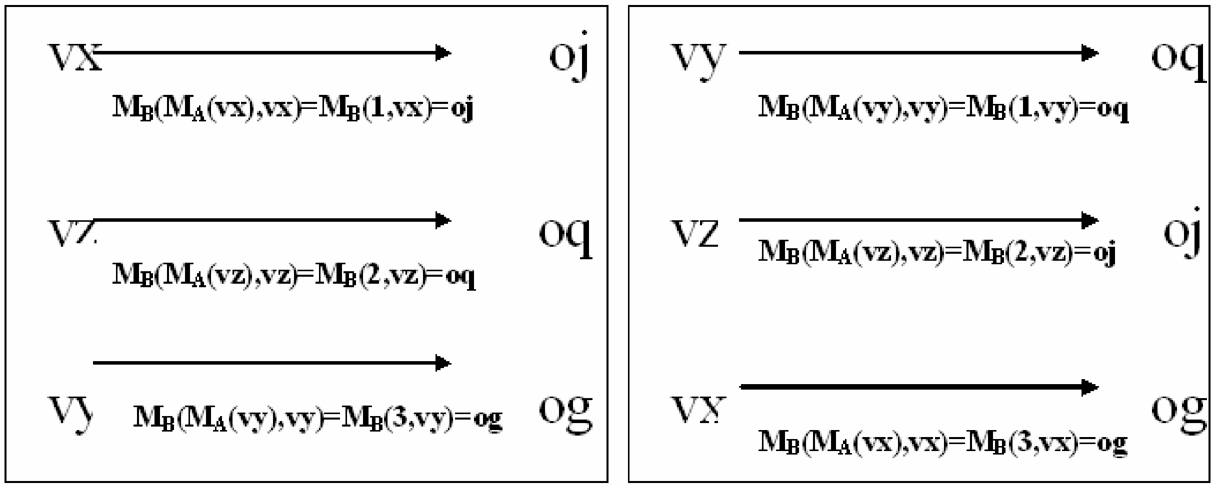
Mapping:

$ru=S_I(\text{Def-hasUncle}) \in IRR$ $rn=S_I(\text{Def-hasNiece}) \in IRR$

$vx=S_I(x) \in IRV$ $vy=S_I(y) \in IRV$ $vz=S_I(z) \in IRV$

$oj=S_I(mj) \in IOT$ $og=S_I(mdg) \in IOT$ $oq=S_I(mdq) \in IOT$

$M_R(ru)=< vx, vz, vy >$ $M_R(rn)=< vy, vz, vx >$



Def-hasUncle Def-hasNiece

2.3 SWRL Full and non-Full

- SWRL Full: meta-modeling, as RDF(S) or OWL Full does
  - $CEXT_I(S_I(\text{owl:Thing}))$ = $CEXT_I(S_I(\text{rdfs:Resource}))$
  - $CEXT_I(S_I(\text{owl:ObjectProperty}))$ = $CEXT_I(S_I(\text{rdf:Property}))$
  - $CEXT_I(S_I(\text{owl:Class}))$ = $CEXT_I(S_I(\text{rdfs:Class}))$
- SWRL non-Full: a separation of the domain of discourse into disjoint parts
  - LV*I*, IOT, IOC, IDC, IOOP, IODP, IOAP, IOXP, IL, IX, IRR, IRV, IRA and IRB are all pairwise disjoint
- An example: the OWL primitive semantic condition
  - Instances of OWL classes are OWL individuals
  - <foo:x rdfs:subClassOf owl:Thing> ← <foo:x rdf:type owl:Class>
  - Implies(Antecedent(owl:Class(I-variable(foo:x)))
    Consequent(rdfs:subClassOf(I-variable(foo:x) owl:Thing))
)
  - $v = M_B(M_A(S_I(\text{foo:x})), S_I(\text{foo:x})) = M_B(1, S_I(\text{foo:x})) \in IOT$
  - $c = M_B(M_A(S_I(\text{owl:Thing})), S_I(\text{owl:Thing})) = M_B(0, S_I(\text{owl:Thing})) = S_I(\text{owl:Thing}) \in IOT$
  - Implies:
    - if $\mathbf{v} \in CEXT_I(S_I(\text{owl:Class})) = IOC$
    - then $<\mathbf{v}, \mathbf{c}> \in EXT_I(S_I(\text{rdfs:subClassOf}))$
    - i.e., $CEXT_I(\mathbf{v}) \subseteq CEXT_I(\mathbf{c}) = CEXT_I(S_I(\textbf{owl:Thing})) = IOT$
  - Impossible for SWRL non-Full, because $v \in IOT$ and $v \in IOC$ and $CEXT_I(v) \in IOT$

2.4 SWRL DL-safe Rules

- DL-safe rules
  - a decidable combination of OWL-DL with rules (cf. [1])
  - Definition: each variable occurs in a non-DL-atom in the rule body
  - Making DL-safe:
    - $A_0 \leftarrow A_1, \ldots, A_m$ (*)
    - $A_0 \leftarrow A_1, \ldots, A_m, O(x_1), \ldots, O(x_n)$ (**)
    - adding special non-DL-atoms O(x) to body of a rule r, for any variable x occurring in r
    - adding a fact O(a) to the KB for each explicitly named individual a in KB
- SWRL DL-safe rules
  - $CR = \{a \mid O(a)\}$: a snapshot of the closed resources from R_I
  - $CEXT_I (S_I (O)) = CR$
  - $M_B: N \times IRV \rightarrow (LV_I \cup IOT) \cap CR$
  - $M_B(M_A(v), v) \in CR = CEXT_I (S_I (O))$ s.t. O(v) is true, for any variable v in r

[1] Boris Motik and Ulrike Sattler and Rudi Studer: Query Answering for OWL-DL with Rules. In Proceedings of ISWC 2004.

2.5 Implementation

- Bottom-up Datalog Engine + RDF database (such as Sesame)
  - Sesame: decomposing SWRL document into RDF graph
  - Datalog: simulating bottom-up SLD triple engine, with fixpoint operator
- Reasoning support for RDF(S) + SWRL rules
  - Sesame: RDF Schema inferencing and querying
  - Datalog: recursive rules like
    - hasDescendent(x,y) ← hasParent(y,x)
    - hasDescendent(y,z) ← hasParent(x,y), hasDescendent(x,z)

2.5 Implementation (cont'd)

- Use case: family.swrl ([2] with some modifications)
  - OWL ontology
    - Class: Person, Man, Woman, Child, Parent, etc.
    - Property: hasChild, hasParent, hasUncle, etc.
    - Individual: 10 Man and 10 Woman
  - SWRL rules
    - 15 rules: Def-hasUncle, Def-Sibling, Def-hasDescendent, etc.
  - Result
    - Input: 12 assertions of "hasChild"
    - Output: 24 hasParent, 4 hasUncle, 14 hasSibling, 54 hasDescendent, etc.

[2] Christine Golbreich: Combining Rule and Ontology Reasoners for the Semantic Web. In Proceedings of RuleML 2004.

3 Conclusion

- An RDF-compatible model-theoretic semantics for SWRL
  - SWRL interpretations
  - SWRL Full interpretations
  - SWRL non-Full interpretations
  - SWRL DL-safe rules
- A bottom-up Datalog engine for SWRL rules as RDF triples
  - On top of RDF graphs
  - SWRL engine for rules in RDF syntax
- Limitation
  - Only partial support for OWL reasoning
- Ongoing work
  - More efficient algorithms for combining OWL and Datalog

Thanks!