ARISTOTLE UNIVERSITY OF THESSALONIKI

SCHOOL OF INFORMATICS AUTH

Intelligent Systems

SWRL2SPIN

Converting SWRL to SPIN

Nick Bassiliades
Intelligent Systems Lab, School of Informatics, Aristotle Univ. of Thessaloniki, Greece

What?

- SWRL2SPIN is a prototype tool built in SWI-Prolog
- Takes as input an OWL ontology with a (flat) SWRL rule base
- Converts SWRL rules into SPIN rules in the same ontology
  - Rule conditions are analyzed
  - SPIN rules are linked to ontology classes (OO flavor of SPIN)
  - Condition elements (could be) re-ordered for optimized evaluation

Why?

- To prolong the life of existing SWRL rule-based ontology applications by converting to SPIN
- SWRL combines OWL with Horn Logic rules of the RuleML family
  - Supported by Protégé, rule engines (Jess, Drools) and ontology reasoners (Pellet)
  - Very popular choice for developing rule-based applications on top of OWL
  - Difficult to become a W3C standard; reach out to industrial world
- SPIN has become a de-facto industry standard to represent SPARQL rules and constraints
  - Builds on acceptance of SPARQL
  - SHACL W3C standard is a legitimate successor of SPIN (strongly influenced)

How?

- Conversion at the RDF vocabulary level
  - SWRL RDF representation → SPIN RDF vocabulary
- Also textual SPIN representation is generated
  - This could be (has been) exploited for SWRL-to-SHACL-rules conversion

```
Student(?s) ∧ attends(?s,?c) ∧ isTaughtBy(?c,?f) →
                                                knows(?s,?f)
```

```
CONSTRUCT {
    ?s :knows ?f .
}
WHERE {
    ?s rdf:type :Student .
    ?s :attends ?c .
    ?c :isTaughtBy ?f .
}
```

Correspondence between SWRL and SPIN constructs (I)

SWRL	SPIN
swrl:Imp	sp:Construct
swrl:head	sp:templates
swrl:body	sp:where
swrl:ClassAtom **swrl:classPredicate <Class>** **swrl:argument1 <Arg>**	sp:subject <Arg> sp:predicate rdf:type sp:object <Class>
Optionally, to avoid expensive RDFS/OWL reasoning	sp:subject <Arg> sp:path <rdf:type/rdfs:subClassOf*> sp:object <Class>
swrl:IndividualPropertyAtom **swrl:propertyPredicate <Prop>** **swrl:argument1 <Arg1>** **swrl:argument2 <Arg2>**	sp:subject <Arg1> sp:predicate <Prop> sp:object <Arg2>
swrl:DifferentIndividualsAtom **swrl:propertyPredicate <Prop>** **swrl:argument1 <Arg1>** **swrl:argument2 <Arg2>**	sp:subject <Arg1> sp:predicate <Prop> sp:object <Arg2>

Correspondence between SWRL and SPIN constructs (II)

SWRL	SPIN
swrl:SameIndividualAtom swrl:argument1 <Arg1> swrl:argument2 <Arg2>	sp:subject <Arg1> sp:predicate owl:sameAs sp:object <Arg2>
swrl:DifferentIndividualsAtom swrl:argument1 <Arg1> swrl:argument2 <Arg2>	sp:subject <Arg1> sp:predicate owl:differentFrom sp:object <Arg2>
swrl:BuiltinAtom swrl:builtin <Fun> swrl:arguments <Args>	*Customized translation*
swrl:Variable <Var>	sp:varName “<Var>”
<Value> ^^ <DataType>	<Value> ^^ <DataType>
<Individual>	<Individual>

Embedding SPIN rules in Classes

```
CONSTRUCT {
    ?x :knows ?z .
}
WHERE {
    ?x rdf:type :Student .
    ?x :attends ?y .
    ?y :isTaughtBy ?z .
}
```

```
CONSTRUCT {     # @Student
      ?this  :knows ?z .
}
WHERE {
      ?this  :attends ?y .
      ?y  :isTaughtBy ?z .
}
```

```
CONSTRUCT {     # @Course
      ?x  :knows ?z .
}
WHERE {
      ?x rdf:type  :Student .
      ?x  :attends ?this .
      ?this  :isTaughtBy ?z .
}
```

Optimizing SPIN rules

```
CONSTRUCT {     # @Course
      ?x   :knows ?z .
}
WHERE {
      ?x rdf:type   :Student .
      ?x   :attends ?this .
      ?this   :isTaughtBy ?z .
}
```

```
CONSTRUCT {     # @Course
      ?x   :knows ?z .
}
WHERE {
      ?this   :isTaughtBy ?z .
      ?x   :attends ?this .
      ?x rdf:type   :Student .
}
```

Implementing SWRL builtins

- 41 built-ins supported (out of 78)
  - Mostly: Comparisons, Mathematics, Strings and Lists
  - Date, Time and Duration: only swrlb:date
- Conversion of builtins falls into 10 categories:
  - Filter: binary filter, filter function, complex filter
  - Bind: associative infix assign, binary infix assign, unary assign, assign function, complex assign
  - Other: magic property, complex expression

Built-in examples (1/2)

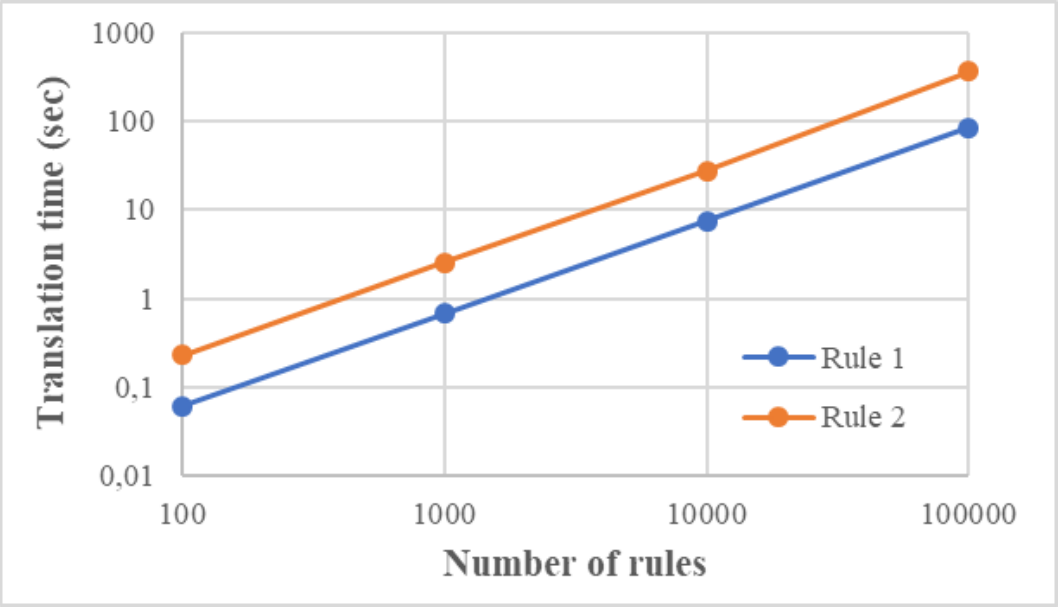
- Binary filter
  - greaterThan(?x,?y) → FILTER (?x > ?y)
- Associative infix assign
  - add(?y,?x1,?x2,…,?xn) → BIND ((((?x1 + ?x2) +…) + ?xn) AS ?y)
- Filter function
  - endsWith(?x,?y) → FILTER STRENDS(?x, ?y)
- Assign function
  - stringLength(?y,?x) → BIND (STRLEN(?x) AS ?y)

Built-in examples (2/2)

- Complex assign
  - integerDivide(?z,?x,?y) → BIND (spif:cast(?x / ?y, xsd:integer) AS ?z)
- Complex filter
  - stringEqualIgnoreCase(?s1,?s2) → FILTER (LCASE(?s1) = LCASE(?s2))
- Complex expression
  - member(?e,?list) → ?list (rdf:rest)*/rdf:first ?e .
- Magic property
  - tokenize(?x,?y,?z) → ?x spif:split (?y ?z) .

Evaluation

Rule translation time scalability



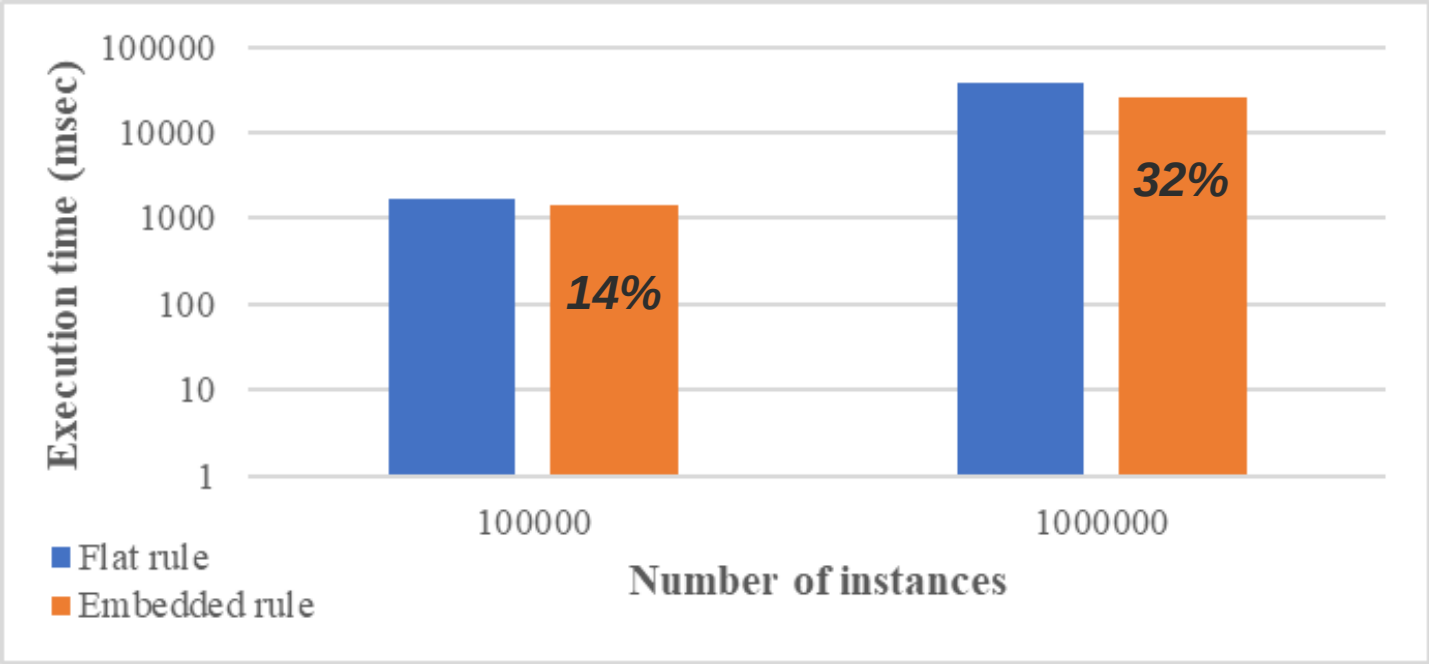
Rule1:

Student(?s) ∧ attends(?s,?c) ∧ isTaughtBy(?c,?f) → knows(?s,?f)

Rule 2:

Student(?x)∧ attends(?x,?y)∧ isTaughtBy(?y,?z) ∧ firstName(?z,?f) ∧ lastName(?z,?l) ∧ swrlb:stringConcat(?fn,?f, " ", ?l) -> knowsName(?x, ?fn)

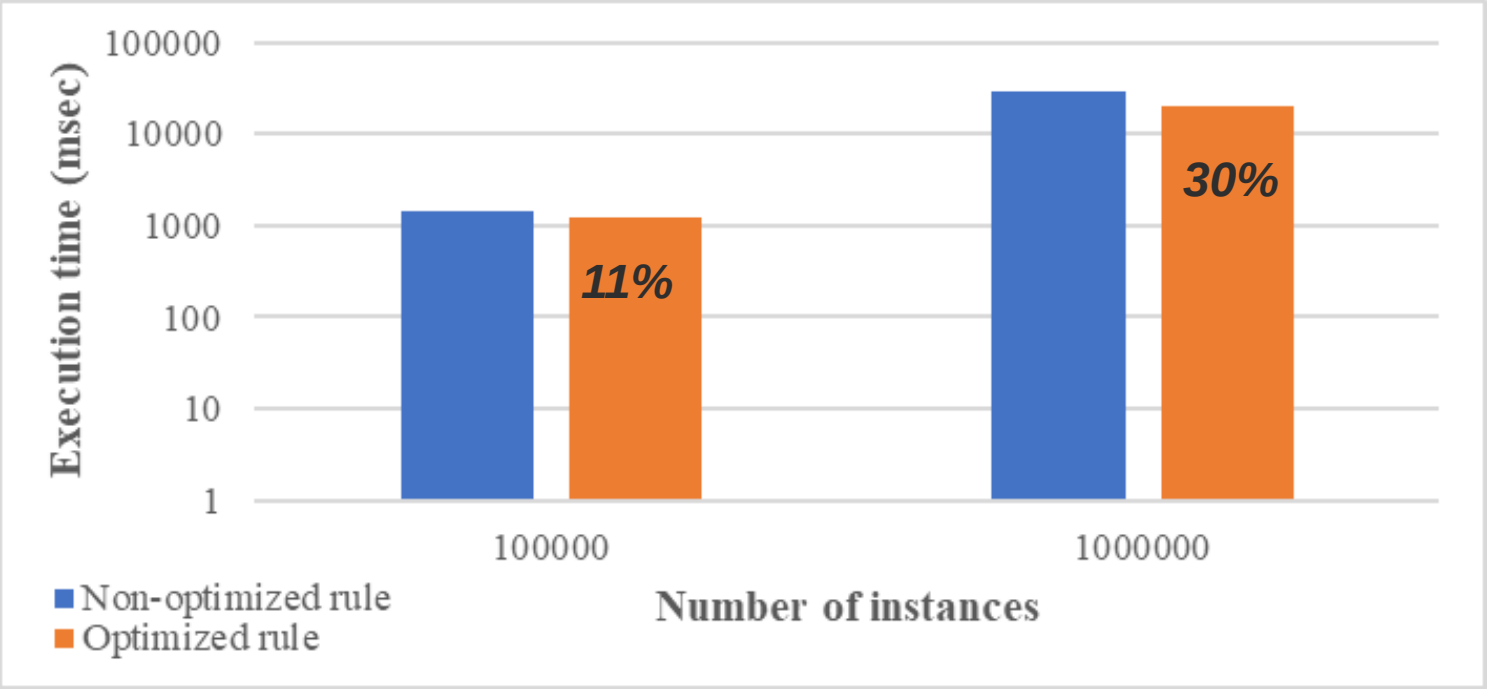
Evaluation
Flat SPIN rule vs. rule embedded in a class




```
# Flat rule
CONSTRUCT {
     ?x :knows ?z .
}
WHERE {
     ?x rdf:type :Student .
     ?x :attends ?y .
     ?y :isTaughtBy ?z .
}
```

```
# Embedded rule
CONSTRUCT { # @Student
     ?this  :knows ?z .
}
WHERE {
     ?this  :attends ?y .
     ?y  :isTaughtBy ?z .
}
```

Evaluation
Non-optimized SPIN rule vs. optimized rule




```
# non-optimized rule
CONSTRUCT { # @Course
     ?x :knows ?z .
}
WHERE {
     ?x rdf:type :Student .
     ?x :attends ?this .
     ?this :isTaughtBy ?z .
}
```

```
# optimized rule
CONSTRUCT { # @Course
     ?x :knows ?z .
}
WHERE {
     ?this :isTaughtBy ?z .
     ?x :attends ?this .
     ?x rdf:type :Student .
}
```

Conclusions

- SWRL2SPIN: A tool to transform SWRL to SPIN rules
- Based on RDF vocabulary translation AND text SPIN syntax
- Rules embedded in classes (instead of flat) and optimized (re-ordering atoms in the condition)
- Initial evaluation shows promising results
- Looking for large SWRL rule bases for proper evaluation
- Porting tool to translate SWRL to SHACL SPARQL rules

SWRL2SPIN Code: https://github.com/nbassili/SWRL2SPIN

SWRL2SPIN full Technical Report: https://arxiv.org/abs/1801.09061

Thank you!