Protégé-OWL SWRLQueryTab

Martin O'Connor
Stanford Medical Informatics, Stanford University, Stanford, CA, USA

Abstract The SWRLQueryTab is a plug-in to Protege-OWL that provides a graphical interface to work with SWRL rules that contain query built-ins. These built-ins that can be used in SWRL rules to query OWL ontologies and effectively turn SWRL into a query language. They provide SQL-like operations to format knowledge retrieved from an OWL ontology. The resulting query language does not alter SWRL's semantics and uses the standard presentation syntax supported by the Protégé-OWL SWRL Editor.

Basic SWRL Queries

Assume we have a simple ontology with classes `Person`, which has subclasses `Male` and `Female` with associated functional properties `hasAge` and `hasName`, and a class `Car`, that can be associated with individual of class `Person` through a `hasCar` property.

Given this ontology we can, for example, write a SWRL query [1] to extract all known persons in an ontology whose age is less than 25, together with their ages:

Person(?p) ^ hasAge(?p, ?age) ^ swrlb:lessThan(?age, 25) -> query:select(?p, ?age)

To list all the cars owned by each person, we can write:

Person(?p) ^ hasCar(?p, ?c) -> query:select(?p, ?c)

This query will return pairs of individuals and their cars. Assuming `hasCar` is a non functional property, multiple pairs would be displayed for each individual - one pair for each car that they own.

Queries and Rules

Rules with query built-ins are not independent of other SWRL rules in an ontology - they operate in conjunction with those rules as part of a rule base. Queries can thus be used to retrieve knowledge inferred by other rules.

Assume, for example, that we define the person subclass `Adult` and attach the functional property `hasAge` to persons and write the following rule to classify persons older than 17 as adults:

Person(?p) ^ hasAge(?p, ?age) ^ swrlb:greaterThan(?age, 17) -> Adult(?p)

We can then write a query to list all the adults in an ontology as follows:

Adult(?p) -> query:select(?p)

While, one could write a single rule that simply retrieved all persons older than 17 years old, the use of an intermediate class assists readability and permits reuse of the intermediate concept in other rules. While subqueries are not possible in SWRL queries, these intermediate concepts can be used to provide an effective equivalent.

Counting

Basic counting is also supported by the query built-in library. A built-in called `count` provides this functionality. It takes a single argument. If we wished to, say, get a count of the number of cars owned by individuals in an ontology, we could write:

Person(?p) ^ hasCar(?p, ?c) -> query:select(?p) ^ query:count(?c)

This query would return a list of individuals and counts, with one row for each individual together with a count of the number of cars that they own. Individuals that have no cars would not be matched by this query.

A similar query to count the number of cars owned by persons in an ontology can be written as:

Person(?p) ^ hasCar(?p, ?c) -> query:count(?c)

Aggregation

Basic aggregation is also supported. Four built-ins called `min`, `max`, `sum`, and `avg` provide this functionality. Aggregation built-ins take a single argument which must represent a numeric type. For example, a query to return the average age of persons in an ontology (for which an age is known) can be written as:

Person(?p) ^ hasAge(?p, ?age) -> query:avg(?age)

Similarly, a query to return the maximum age of a person in an ontology can be written as:

Person(?p) ^ hasAge(?p, ?age) -> query:max(?age)

Any numeric variable not passed to a select built-in can be aggregated. Variables that have already been passed to a select built-in cannot be aggregated - an error will be generated by the query library if an attempt is made to use them in this way.

Ordering of Results

Results can be ordered using the `orderBy` and `orderByDescending` built-ins. For example, to extend the earlier query that returns a count of the number of cars owned by each person to order the results by each person's name, we can write:

Person(?p) ^ hasName(?p, ?name) ^ hasCar(?p, ?c) -> query:select(?name) ^ query:count(?c) ^ query:orderBy(?name)

The `orderBy` and `orderByDescending` built-ins take one or more variables as arguments. All such arguments must have been used in a `select`, `count`, or aggregate built-in in the same query.

So, for example, we can rewrite the previous query to order the results but the number of cars owned by each person in descending order as follows:

Person(?p) ^ hasName(?p, ?name) ^ hasCar(?p, ?c) -> query:select(?name) ^ query:count(?c) ^ query:orderByDescending(?c)

Interoperation with other Built-In Libraries

Query built-ins can be used freely with other SWRL built-in libraries [2]. For example, to retrieve the name of all males in an ontology and prepend the title "Mr." to each name, we can use core SWRL Built-in [3] `stringConcat` built-in:

Person(?p) ^ Male(?p) ^ hasName(?p, ?name) ^ swrlb:stringConcat(?fullname, "Mr. ", ?name) -> query:select(?fullname)

Using the TBox Query Library [4] with query built-ins allows the writing of queries that ask questions about the classes and properties in an OWL ontology. For example, a query that returns all the direct subclasses of the `Person` class can be written:

tbox:isDirectSubClassOf(?subClass, Person) - > query:select(?subClass)

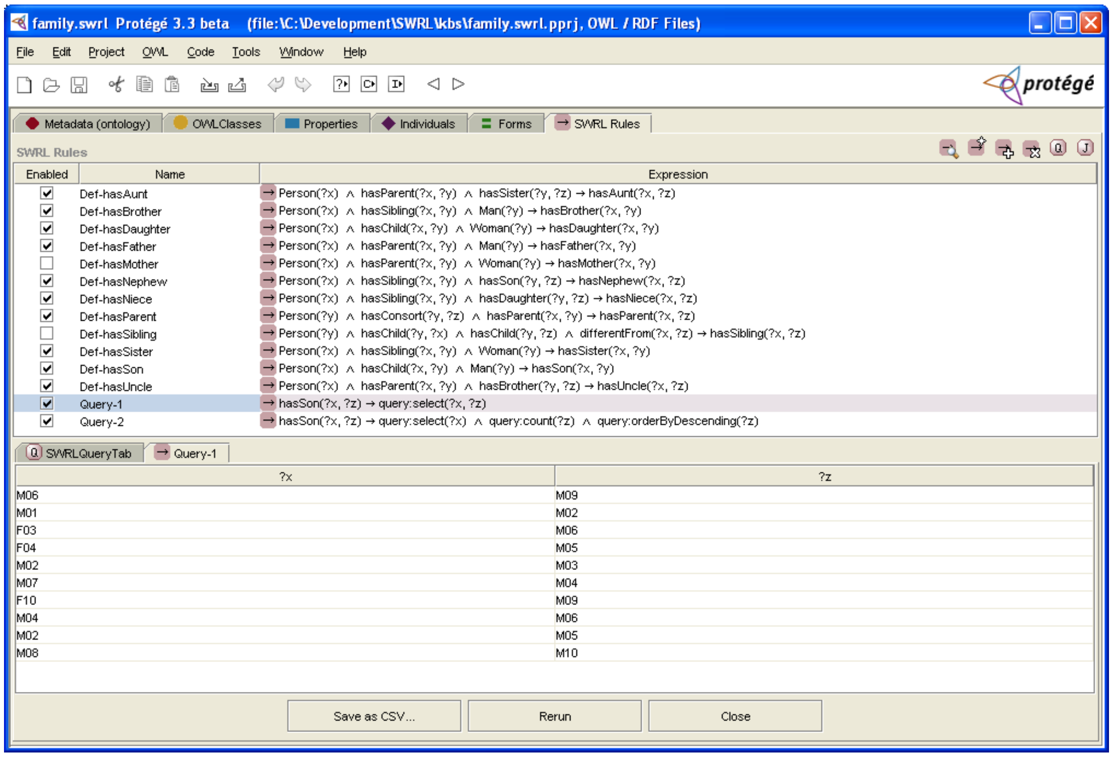
To retrieve all properties (both direct and indirect) of the `hasName` property, one can write:

tbox:isSubPropertyOf(?supProperty, hasName) - > query:select(?subProperty)

The ability to freely use built-ins in a query provides a means of continuously expanding the power of the query language.

SWRLQueryTab

The SWRLQueryTab [5] provides a convenient way to visualize the results of these SWRL queries. It has a control sub-tab that can be used to control the execution of SWRL rules containing query built-ins. A query can be selected from the rule table in the Protege-OWL SWRL Editor and executed to display results of the query. Users can navigate to that sub-tab to review the results displayed in tabular form.


family.swrl Protégé 3.3 beta (file:\C:\Development\SWRL\kbs\family.swrl.pprj, OWL / RDF Files)
File Edit Project OWL Code Tools Window Help
protégé
Metadata (ontology) OWLClasses Properties Individuals Forms SWRL Rules
SWRL Rules
SWRLQueryTab Query-1
Save as CSV... Rerun Close

Enabled	Name	Expression
✓	Def-hasAunt	Person(?x) ∧ hasParent(?x, ?y) ∧ hasSister(?y, ?z) → hasAunt(?x, ?z)
✓	Def-hasBrother	Person(?x) ∧ hasSibling(?x, ?y) ∧ Man(?y) → hasBrother(?x, ?y)
✓	Def-hasDaughter	Person(?x) ∧ hasChild(?x, ?y) ∧ Woman(?y) → hasDaughter(?x, ?y)
✓	Def-hasFather	Person(?x) ∧ hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
	Def-hasMother	Person(?x) ∧ hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
✓	Def-hasNephew	Person(?x) ∧ hasSibling(?x, ?y) ∧ hasSon(?y, ?z) → hasNephew(?x, ?z)
✓	Def-hasNiece	Person(?x) ∧ hasSibling(?x, ?y) ∧ hasDaughter(?y, ?z) → hasNiece(?x, ?z)
✓	Def-hasParent	Person(?y) ∧ hasConsort(?y, ?z) ∧ hasParent(?x, ?y) → hasParent(?x, ?z)
	Def-hasSibling	Person(?y) ∧ hasChild(?y, ?x) ∧ hasChild(?y, ?z) ∧ differentFrom(?x, ?z) → hasSibling(?x, ?z)
✓	Def-hasSister	Person(?x) ∧ hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
✓	Def-hasSon	Person(?x) ∧ hasChild(?x, ?y) ∧ Man(?y) → hasSon(?x, ?y)
✓	Def-hasUncle	Person(?x) ∧ hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)
✓	Query-1	hasSon(?x, ?z) → query:select(?x, ?z)
✓	Query-2	hasSon(?x, ?z) → query:select(?x) ∧ query:count(?z) ∧ query:orderByDescending(?z)

?x	?z
M06	M09
M01	M02
F03	M06
F04	M05
M02	M03
M07	M04
F10	M09
M04	M06
M02	M05
M08	M10

SWRLQueryAPI

The SWRLQueryAPI [6] provides a JDBC-like Java interface to retrieve the result of SWRL rules containing query built-ins. The interface Result [7] defines methods for processing query results. Results for a particular query can be retrieved from a SWRL rule engine bridge [8] by using the method `getQueryResult` supplied with the name of the SWRL rule containing the query:

```
Result result = bridge.getQueryResult("Query-1");
```

Rows in a `Result` object can be iterated through using the `hasNext` and `next` methods. The contents of each column in a row can then be retrieved using appropriate accessor method for the four possible types. For example, if we wish to process the results of the earlier query that extracts all persons under the age of 25 from our ontology, we can write:

```
while (result.hasNext()) {
  System.out.println("Person: " + result.getDatatypeValue("?p"));
  System.out.println("Age: " + result.getDatatypeValue("?age"));

  result.next();
} // while
```

Installation

The SWRLQueryTab is contained in the standard Protege-OWL 3.3 distribution and does not need to be downloaded separately. It currently requires the Jess rule engine [9], which must be downloaded separately. Jess is not open source and a license is required for its use. However, this license is free for academic users. The Jess rule engine is contained in a Java JAR called jess.jar, which is contained in the standard Jess distribution. This JAR must be copied to the Protege-OWL plugins subdirectory in the Protege installation directory (i.e., the ./plugins/edu.stanford.smi.protegex.owl/ subdirectory of the Protege installation directory). Protege-OWL will automatically load this JAR file on startup if it is present in its plugins directory. If this JAR file is not present, the SWRLQueryTab will display an error when it is activated.

References

[1] SWRL Query Built-Ins: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLQueryBuiltIns
[2] SWRLTab Built-in Libraries: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTabBuiltInLibraries
[3] Core SWRL Built-in Library: http://protege.cim3.net/cgi-bin/wiki.pl?CoreSWRLBuiltIns
[4] TBox Built-in Library: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTBoxBuiltIns
[5] SWRLQueryTab: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLQueryTab
[6] SWRLQueryAPI: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLQueryAPI
[7] Result http://protege.stanford.edu/javadoc/edu/stanford/smi/protegex/owl/swrl/bridge/query/Result.html
[8] SWRL Rule Engine Bridge: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLRuleEngineBridgeFAQ
[9] Jess: http://herzberg.ca.sandia.gov/jess/