

The Semantic Web Rule Language

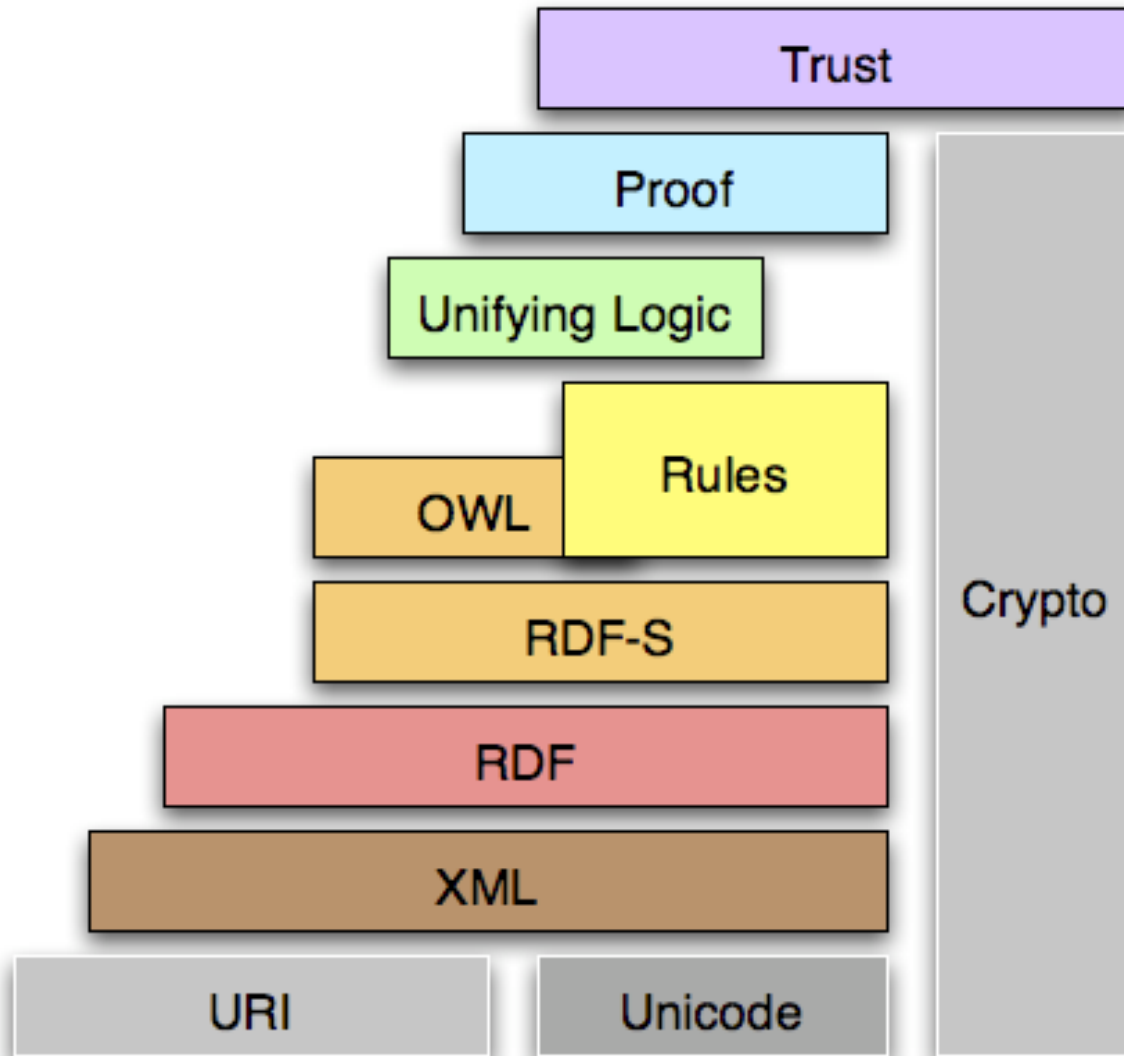
Martin O'Connor
Stanford Center for Biomedical Informatics Research,
Stanford University



Talk Outline

- Rules and the Semantic Web
- Basic SWRL Rules
- SWRL's Semantics
- SWRLTab: a Protégé-OWL development environment for SWRL
- SQWRL: a SWRL-based OWL query language

Semantic Web Stack



Rule-based Systems are common in many domains

- Engineering: Diagnosis rules
- Commerce: Business rules
- Law: Legal reasoning
- Medicine: Eligibility, Compliance
- Internet: Access authentication

Rule Markup (RuleML) Initiative

- Effort to standardize inference rules.
- RuleML is a markup language for publishing and sharing rule bases on the World Wide Web.
- Focus is on rule interoperation between industry standards.
- RuleML builds a hierarchy of rule sublanguages upon XML, RDF, and OWL, e.g., SWRL

What is SWRL?

- SWRL is an acronym for Semantic Web Rule Language.
- SWRL is intended to be the rule language of the Semantic Web.
- SWRL includes a high-level abstract syntax for Horn-like rules.
- All rules are expressed in terms of OWL concepts (classes, properties, individuals).
- Language FAQ:
 - <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ>

SWRL Characteristics

- W3C Submission in 2004:
<http://www.w3.org/Submission/SWRL/>
- Rules saved as part of ontology
- Increasing tool support: Bossam, R2ML, Hoolet, Pellet, KAON2, RacerPro, SWRLTab
- Can work with reasoners

Example SWRL Rule: Reclassification

Man(?m) → Person(?m)

Possible in OWL - some rules are OWL syntactic sugar

Example SWRL Rule: property value assignment

$\text{Person}(\text{?p}) \wedge \text{hasSibling}(\text{?p}, \text{?s}) \wedge \text{Man}(\text{?s})$
 $\rightarrow \text{hasBrother}(\text{?p}, \text{?s})$

Example SWRL Rule: property value assignment

$\text{hasParent}(\text{?x}, \text{?y}) \wedge \text{hasBrother}(\text{?y}, \text{?z})$
 $\rightarrow \text{hasUncle}(\text{?x}, \text{?z})$

Not possible in OWL 1.0 - some rules are *not* OWL syntactic sugar

Example SWRL Rule with Named Individuals: Has brother

Person(Fred) ^ hasSibling(Fred, ?s) ^ Man(?s)
→ hasBrother(Fred, ?s)

Example SWRL Rule with Literals and Built-ins: is adult?

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

Built-ins dramatically increase expressivity - most rules are not
OWL syntactic sugar

Example SWRL Rule with String Built-ins

Person(?p) ^ hasNumber(?p, ?number)
^ **swrlb:startsWith**(?number, "+") →
hasInternationalNumber(?p, true)

Example SWRL Rule with Built-in Argument Binding

Person(?p) ^ hasSalaryInPounds(?p, ?pounds) ^
swrlb:multiply(?dollars, ?pounds, 2.0) →
hasSalaryInDollars(?p, ?dollars)

Example SWRL Rule with Built-in Argument Binding II

Person(?p) ^ hasSalaryInPounds(?p, ?pounds) ^
swrlb:multiply(2.0, ?pounds, ?dollars) ->
hasSalaryInDollars(?p, ?dollars)

Arguments can bind in any position - though generally an implementation will support binding of only the first argument

Can define new Built-in Libraries

- **Temporal built-ins:**
 - `temporal:before("1999-11-01T10:00", "2000-02-01T11:12:12.000")`
 - `temporal:duration(2, "1999-11-01", "2001-02-01", temporal:Years)`
- **TBox built-ins:**
 - `tbox:isDatatypeProperty(?p)`
 - `tbox:isDirectSubPropertyOf(?sp, ?p)`
- **Mathematical built-ins:**
 - `swrlm:eval(?circumference, "2 * pi * r", ?r)`

SWRLTab Built-in Libraries

The screenshot shows a Mozilla Firefox browser window with the address bar displaying `http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTabBuiltInLibraries`. The page title is "ProtegeWiki: SWRLTab Built In Libraries". The main heading is "SWRLTabBuiltInLibraries". Below the heading, there are navigation links: "WikiHomePage", "RecentChanges", and "Page Index". On the right side, there is a user profile for "MartinOConnor" with links for "preferences" and "logout".

The main content area contains a paragraph: "A number of built-in libraries are provided by the [SWRLTab](#). These include: (8AA)"

- **Core SWRL Built-Ins Library** Contains implementations for the core built-ins defined by the [SWRL Submission](#). It is documented [here](#). (8AB)
- **SQWRL Built-In Library** Contains a set of built-ins that extend SWRL to SQWRL. It is documented [here](#). (8XB)
- **Temporal Built-Ins Library** Defines a set of built-ins that can be used to perform temporal operations. It is documented [here](#). (995)
- **ABox Built-Ins Library** Defines built-ins that can be used to query an ABox. It is documented [here](#). (8LB)
- **TBox Built-Ins Library** Defines built-ins that can be used to query a TBox. It is documented [here](#). (8LB)
- **Mathematical Expressions Built-Ins Library** Defines built-ins that can be used to evaluate complex mathematical expressions in SWRL rules. It is documented [here](#). (8LB)
- **XML Built-Ins Library** Defines built-ins that can be used to query XML documents. It is documented [here](#). (A00)
- **Extensions Built-ins Library** Defines some experimental built-ins that can be used to increase the expressivity of SWRL. It is documented [here](#). (8LC)

New SWRL built-in libraries can be defined by developers using the [SWRLBuiltInBridge](#). (8AE)

At the bottom of the page, there are links for "Edit text of this page" and "View other revisions", and a note "Last edited October 26, 2007 17:20 (diff)".

On the right side, there is a "Your Visited Pages" section listing: SWRLTabBuiltInLibraries, SWRLLanguageFAQ, SWRLTab, SWRLTabMathematicalBuiltIns, SWRLTabXMLBuiltIns, SWRLBoxBuiltIns, SWRLABoxBuiltIns, and SWRLTemporalBuiltIns. Below this is a "View Backlinks" button and a "Search" box.

The browser's status bar at the bottom shows "Done" and a search bar with the text "class de".

Example SWRL Rule with OWL Class Expressions

$(\text{hasChild } \geq 1)(?x) \rightarrow \text{Parent}(?x)$

This does not say: all individuals with a child are parents

It says: all individuals that are members of an OWL class with the associated restriction that its hasChild property has a minimum cardinality of one

Example SWRL Rule with Inferred OWL Class Expressions

Parent(?x) \rightarrow (hasChild \geq 1)(?x)

Arbitrary OWL class expressions are allowed

Expression syntax may vary, though Manchester Syntax common

SWRL Semantics

- Based on OWL-DL
- Has a formal semantics
- Complements OWL and fully semantically compatible
- More expressive yet at expense of decidability
- Use OWL if extra expressiveness not required (possible exception: querying)

OWL Class Expressions and the Open World Assumption

$(\text{hasChild } \geq 1)(?x) \rightarrow \text{Parent}(?x)$

This does not say: all individuals with a child are parents

It says: all individuals that are members of the OWL class with the associated restriction that its hasChild property has a minimum cardinality of one

Individuals with no **known** children may be classified as parents

SWRL and Open World Semantics: sameAs, differentFrom

Publication(?p) ^ hasAuthor(?p, ?y) ^
hasAuthor(?p, ?z) ^ **differentFrom**(?y, ?z)
→ cooperatedWith(?y, ?z)

Like OWL, SWRL does *not* adopt the **unique name assumption**

Individuals must also be **explicitly stated to be different** (using, for example, owl:allDifferents restriction)

SWRL is Monotonic: does not Support Negated Atoms

Person(?p) ^ **not** hasCar(?p, ?c) →
CarlessPerson(?p)

Not possible - language does not support negation here

Potential invalidation - what if a person later gets a car?

SWRL is Monotonic: retraction (or modification) not supported

Person(?p) ^ hasAge(?p,?age) ^
swrlb:add(?newage, ?age, 1)
→ hasAge(?p, ?newage)

Incorrect - will run forever and attempt to assign an infinite number of values to hasAge property

Potential invalidation - essentially attempted retraction

SWRL is Monotonic: counting not supported

Publication(?p) ^ hasAuthor(?p,?a) ^
<has exactly one hasAuthor value in
current ontology>

→ SingleAuthorPublication(?p)

Not expressible - open world applies

Potential invalidation - what if author is added later?

SWRL is Monotonic: counting not supported II

Publication(?p) ^ (hasAuthor = 1)(?p)
→ SingleAuthorPublication(?p)

Closure - though best expressed in OWL in this case

SWRLTab

- A Protégé-OWL development environment for working with SWRL rules
- Supports editing and execution of rules
- Extension mechanisms to work with third-party rule engines
- Mechanisms for users to define built-in method libraries
- Supports querying of ontologies

SWRLTab Wiki : <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab>

SWRLTab

[WikiHomePage](#) | [RecentChanges](#) | [Page Index](#)

MartinOConnor ([preferences](#) | [logout](#))

The SWRLTab is a development environment for working with [SWRL rules](#) in [Protege-OWL](#). It supports the editing and execution of SWRL rules. It provides a set of libraries that can be used in rules, including libraries to interoperate with XML documents, and spreadsheets, and libraries with mathematical, string, RDFS, and temporal operators. A SWRL-based OWL query language called SQWRL is also provided. [\(A3E\)](#)

An introduction to the SWRL language can be found [here](#). An introduction SQWRL can be found [here](#). [\(9EQ\)](#)

The SWRLTab has several software components: [\(6GY\)](#)

- **SWRL Editor** The editor supports editing and saving of SWRL rules in an OWL ontology. See the [SWRL Editor FAQ](#) for more details. [\(8GS\)](#)
- **SWRL Built-in Libraries** A number of built-in libraries are provided by the SWRLTab. These include an implementation of the core SWRL built-ins defined in the [SWRL Submission](#) and built-ins for querying OWL ontologies. The libraries are documented [here](#). [\(8D1\)](#)
- **SQWRL Query Tab** The query tab provides a graphical interface to display the results of SQWRL queries. It is documented [here](#). [\(8XC\)](#)
- **SQWRL Query API** This API provides a JDBC-like Java interface to retrieve the result of SQWRL queries. It is documented [here](#). [\(8XD\)](#)
- **SWRL Built-in Bridge** [SWRL built-ins](#) are user-defined predicates that can be used in SWRL rules. The SWRLTab has a subcomponent called the [built-in bridge](#) that provides a mechanism to define Java implementations of SWRL built-ins. These implementations can then be dynamically loaded by the bridge and invoked from a rule engine. [\(6H2\)](#)
- **SWRL Bridge** The bridge provides the infrastructure necessary to incorporate rule engines into Protege-OWL to execute SWRL rules. See the [SWRL Rule Engine Bridge FAQ](#) for more details. The hope is that bridges for other rule engines will be developed by the Protege-OWL community and than an array of inference mechanism will become available for executing SWRL rules. [\(6H1\)](#)
- **SWRL Jess Bridge** A bridge for the Jess rule engine is provided in the Protege-OWL distribution. A user interface called the SWRLJessTab is also provided to interact with this bridge. [\(8P4\)](#)
- **SWRL Factory** The factory provides high-level Java APIs that support the creation and modification of SWRL rules in an OWL ontology. This API can be used by developers who wish to work with SWRL rules in their applications. See the [SWRL Factory FAQ](#) for more details. [\(6H0\)](#)

The SWRLTab is designed to work in Protege-OWL only. However, partial interoperation with Jena is also supported. See [here](#) for details. Work has proceeded to port it to Protege 4. [\(A6A\)](#)

Your Visited Pages

- SWRLTab
- SWRLTabSyntax
- SWRLLanguageFAQ
- SWRLTabXMLBuiltIns
- SWRLTabBuiltInLibraries
- SWRLTemporalBuiltIns
- SWRLEditorFAQ
- SWRLBoxBuiltIns

View Backlinks

Search

Done

What is the SWRL Editor?

- The SWRL Editor is an extension to Protégé-OWL that permits the interactive editing of SWRL rules.
- The editor can be used to create SWRL rules, edit existing SWRL rules, and read and write SWRL rules.
- It is accessible as a tab within Protégé-OWL.

family.swrl Protégé 3.3 beta (file:\C:\Development\SWRL\kbs\family.swrl.pprj, OWL / ...

File Edit Project OWL Code Tools Window Help

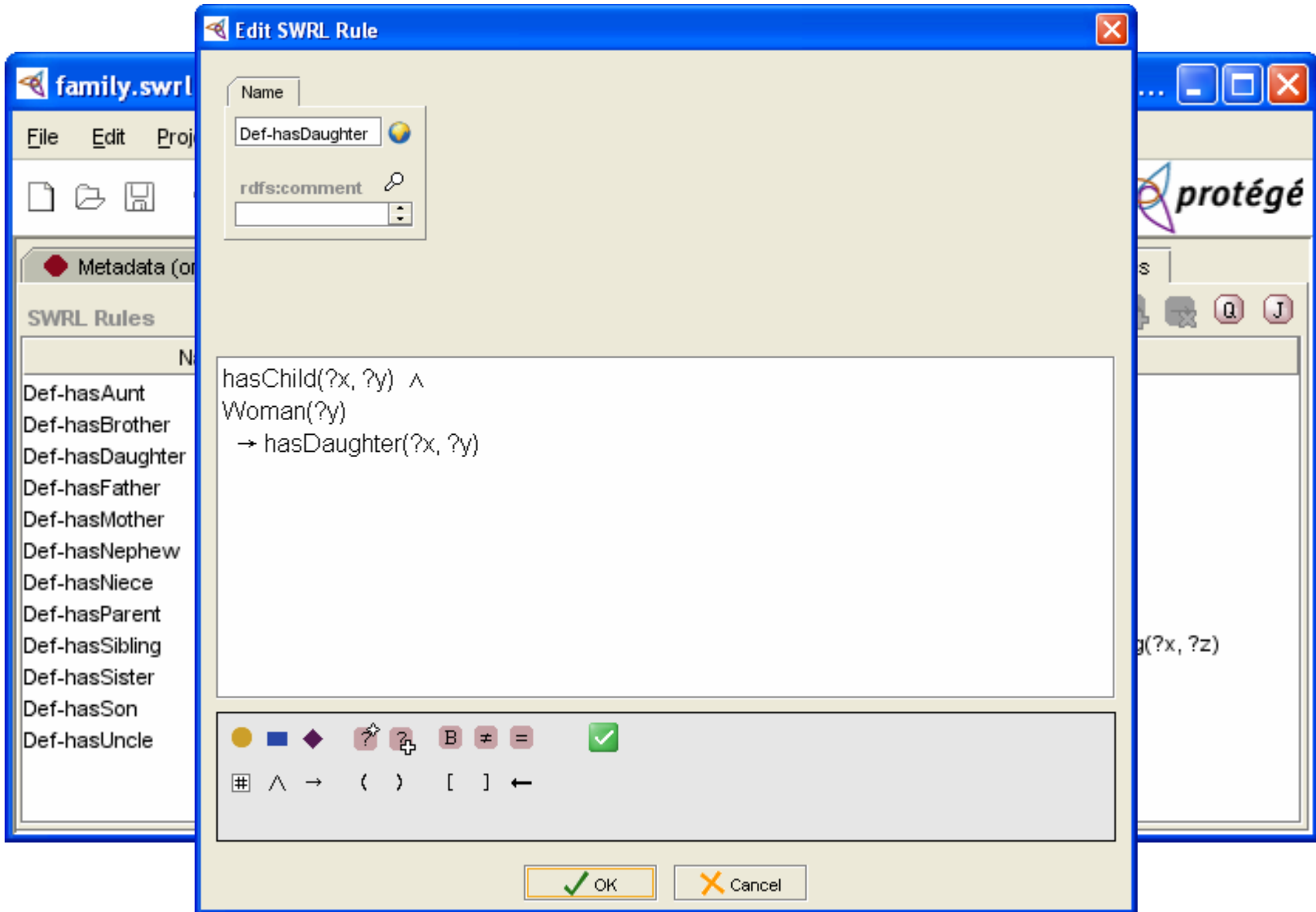


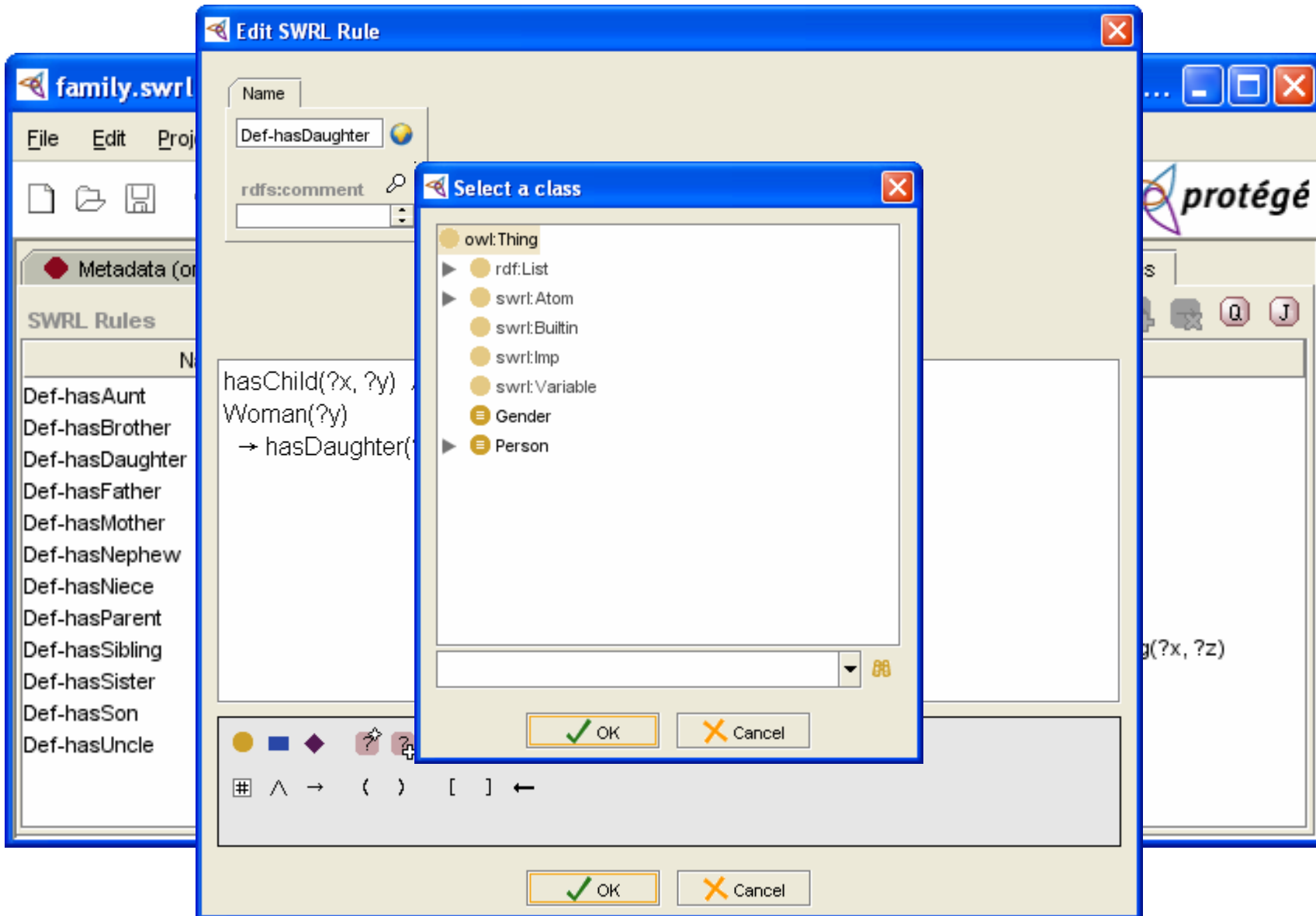
Metadata (ontology) OWLClasses Properties Individuals Forms SWRL Rules

SWRL Rules



Name	Expression
Def-hasAunt	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{hasSister}(?y, ?z) \rightarrow \text{hasAunt}(?x, ?z)$
Def-hasBrother	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasBrother}(?x, ?y)$
Def-hasDaughter	$\rightarrow \text{hasChild}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasDaughter}(?x, ?y)$
Def-hasFather	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasFather}(?x, ?y)$
Def-hasMother	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasMother}(?x, ?y)$
Def-hasNephew	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{hasSon}(?y, ?z) \rightarrow \text{hasNephew}(?x, ?z)$
Def-hasNiece	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{hasDaughter}(?y, ?z) \rightarrow \text{hasNiece}(?x, ?z)$
Def-hasParent	$\rightarrow \text{hasConsort}(?y, ?z) \wedge \text{hasParent}(?x, ?y) \rightarrow \text{hasParent}(?x, ?z)$
Def-hasSibling	$\rightarrow \text{hasChild}(?y, ?x) \wedge \text{hasChild}(?y, ?z) \wedge \text{differentFrom}(?x, ?z) \rightarrow \text{hasSibling}(?x, ?z)$
Def-hasSister	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasSister}(?x, ?y)$
Def-hasSon	$\rightarrow \text{hasChild}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasSon}(?x, ?y)$
Def-hasUncle	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{hasBrother}(?y, ?z) \rightarrow \text{hasUncle}(?x, ?z)$





family.swrl P

File Edit Project

Metadata (onto)

SWRL Rules

Name
Def-hasAunt
Def-hasBrother
Def-hasDaughter
Def-hasFather
Def-hasMother
Def-hasNephew
Def-hasNiece
Def-hasParent
Def-hasSibling
Def-hasSister
Def-hasSon
Def-hasUncle

Edit SWRL Rule

Name: def-hasDaughter

rdfs:comment: []

hasChild(?x, ?y) ^ Woman(?y) → hasDaughter(?x, ?y) ^ swrlb:

- swrlb:abs
- swrlb:add
- swrlb:addDayTimeDurations
- swrlb:addDayTimeDurationToDate
- swrlb:addDayTimeDurationToDateTime
- swrlb:addDayTimeDurationToTime
- swrlb:addYearMonthDurations
- swrlb:addYearMonthDurationToDate

OK Cancel

protégé

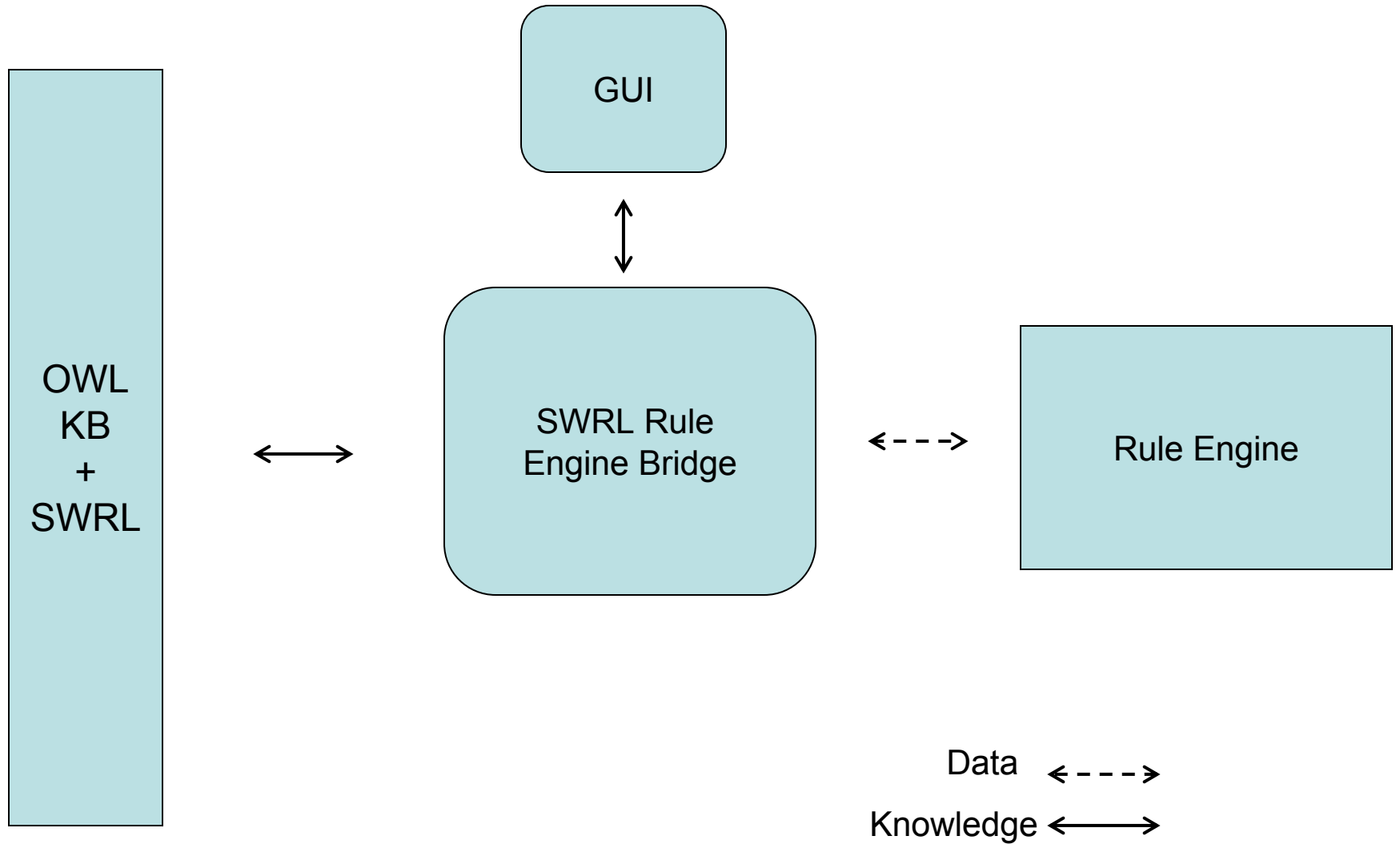
Rules

ling(?x, ?z)

Executing SWRL Rules

- SWRL is a language specification
- Well-defined semantics
- Developers must implement engine
- Or map to existing rule engines
- Hence, a bridge...

SWRL Rule Engine Bridge



SWRL Rule Engine Bridge

- Given an OWL knowledge base it will extract SWRL rules and relevant OWL knowledge.
- Also provides an API to assert inferred knowledge.
- Knowledge (and rules) are described in non Protégé-OWL API-specific way.
- These can then be mapped to a rule-engine specific rule and knowledge format.
- This mapping is developer's responsibility.

We used the SWRL Bridge to Integrate Jess Rule Engine with Protégé-OWL

- Jess is a Java-based rule engine.
- Jess system consists of a rule base, fact base, and an execution engine.
- Available free to academic users, for a small fee to non-academic users
- Has been used in Protégé-based tools, e.g., JessTab.



SWRL Rules



Name	Expression
Def-hasAunt	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{hasSister}(?y, ?z) \rightarrow \text{hasAunt}(?x, ?z)$
Def-hasBrother	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasBrother}(?x, ?y)$
Def-hasDaughter	$\rightarrow \text{hasChild}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasDaughter}(?x, ?y)$
Def-hasFather	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasFather}(?x, ?y)$
Def-hasMother	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasMother}(?x, ?y)$
Def-hasNephew	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{hasSon}(?y, ?z) \rightarrow \text{hasNephew}(?x, ?z)$
Def-hasNiece	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{hasDaughter}(?y, ?z) \rightarrow \text{hasNiece}(?x, ?z)$
Def-hasParent	$\rightarrow \text{hasConsort}(?y, ?z) \wedge \text{hasParent}(?x, ?y) \rightarrow \text{hasParent}(?x, ?z)$
Def-hasSibling	$\rightarrow \text{hasChild}(?y, ?x) \wedge \text{hasChild}(?y, ?z) \wedge \text{differentFrom}(?x, ?z) \rightarrow \text{hasSibling}(?x, ?z)$
Def-hasSister	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasSister}(?x, ?y)$
Def-hasSon	$\rightarrow \text{hasChild}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasSon}(?x, ?y)$
Def-hasUncle	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{hasBrother}(?y, ?z) \rightarrow \text{hasUncle}(?x, ?z)$

SWRLJessTab

See <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessTab> for SWRLJessTab documentation.

Press the "OWL+SWRL->Jess" button to transfer SWRL rules and relevant OWL knowledge to Jess.
Press the "Run Jess" button to run the Jess rule engine.
Press the "Jess->OWL" button to transfer the inferred Jess knowledge to OWL knowledge.

IMPORTANT: With the exception of owl:sameAs, owl:differentFrom and owl:allDifferent, owl:equivalentProperty, and owl:equivalentClass, the Jess rule engine is currently ignoring OWL restrictions. To ensure consistency, a reasoner should be run on an OWL knowledge base before SWRL rules and OWL knowledge are transferred to Jess. Also, if inferred knowledge from Jess is inserted back into an OWL knowledge base, a reasoner should again be executed to ensure that the new knowledge does not conflict with OWL restrictions in that knowledge base.

cf. <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLRuleEngineBridgeFAQ#nid6QL> for details.

OWL+SWRL->Jess Run Jess Jess->OWL



SWRL Rules

Name	Expression
Def-hasAunt	→ hasParent(?x, ?y) ∧ hasSister(?y, ?z) → hasAunt(?x, ?z)
Def-hasBrother	→ hasSibling(?x, ?y) ∧ Man(?y) → hasBrother(?x, ?y)
Def-hasDaughter	→ hasChild(?x, ?y) ∧ Woman(?y) → hasDaughter(?x, ?y)
Def-hasFather	→ hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
Def-hasMother	→ hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
Def-hasNephew	→ hasSibling(?x, ?y) ∧ hasSon(?y, ?z) → hasNephew(?x, ?z)
Def-hasNiece	→ hasSibling(?x, ?y) ∧ hasDaughter(?y, ?z) → hasNiece(?x, ?z)
Def-hasParent	→ hasConsort(?y, ?z) ∧ hasParent(?x, ?y) → hasParent(?x, ?z)
Def-hasSibling	→ hasChild(?y, ?x) ∧ hasChild(?y, ?z) ∧ differentFrom(?x, ?z) → hasSibling(?x, ?z)
Def-hasSister	→ hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
Def-hasSon	→ hasChild(?x, ?y) ∧ Man(?y) → hasSon(?x, ?y)
Def-hasUncle	→ hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)

SWRLJessTab

See <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessTab> for SWRLJessTab documentation.

Press the "OWL+SWRL->Jess" button to transfer SWRL rules and relevant OWL knowledge to Jess.

Press the "Run Jess" button to run the Jess rule engine.

Press the "Jess->OWL" button to transfer the inferred Jess knowledge to OWL knowledge.

IMPORTANT: With the exception of owl:sameAs, owl:differentFrom and owl:allDifferent, owl:equivalentProperty, and owl:equivalentClass, the Jess rule engine is currently ignoring OWL restrictions. To ensure consistency, a reasoner should be run on an OWL knowledge base before SWRL rules and OWL knowledge are transferred to Jess. Also, if inferred knowledge from Jess is inserted back into an OWL knowledge base, a reasoner should again be executed to ensure that the new knowledge does not conflict with OWL restrictions in that knowledge base.

cf. <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLRuleEngineBridgeFAQ#nid6QL> for details.

SWRL Rules

Name	Expression
Def-hasAunt	→ hasParent(?x, ?y) ∧ hasSister(?y, ?z) → hasAunt(?x, ?z)
Def-hasBrother	→ hasSibling(?x, ?y) ∧ Man(?y) → hasBrother(?x, ?y)
Def-hasDaughter	→ hasChild(?x, ?y) ∧ Woman(?y) → hasDaughter(?x, ?y)
Def-hasFather	→ hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
Def-hasMother	→ hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
Def-hasNephew	→ hasSibling(?x, ?y) ∧ hasSon(?y, ?z) → hasNephew(?x, ?z)
Def-hasNiece	→ hasSibling(?x, ?y) ∧ hasDaughter(?y, ?z) → hasNiece(?x, ?z)
Def-hasParent	→ hasConsort(?y, ?z) ∧ hasParent(?x, ?y) → hasParent(?x, ?z)
Def-hasSibling	→ hasChild(?y, ?x) ∧ hasChild(?y, ?z) ∧ differentFrom(?x, ?z) → hasSibling(?x, ?z)
Def-hasSister	→ hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
Def-hasSon	→ hasChild(?x, ?y) ∧ Man(?y) → hasSon(?x, ?y)
Def-hasUncle	→ hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)

Jess Rules Panel

```
(defrule Def-hasUncle (hasParent ?x ?y) (hasBrother ?y ?z) => (assert (hasUncle ?x ?z)) (assertOWLProperty hasUncle ?x ?z) )
(defrule Def-hasSon (hasChild ?x ?y) (Man (name ?y)) => (assert (hasSon ?x ?y)) (assertOWLProperty hasSon ?x ?y) )
(defrule Def-hasDaughter (hasChild ?x ?y) (Woman (name ?y)) => (assert (hasDaughter ?x ?y)) (assertOWLProperty hasDaughter ?x ?y) )
(defrule Def-hasNephew (hasSibling ?x ?y) (hasSon ?y ?z) => (assert (hasNephew ?x ?z)) (assertOWLProperty hasNephew ?x ?z) )
(defrule Def-hasNiece (hasSibling ?x ?y) (hasDaughter ?y ?z) => (assert (hasNiece ?x ?z)) (assertOWLProperty hasNiece ?x ?z) )
(defrule Def-hasBrother (hasSibling ?x ?y) (Man (name ?y)) => (assert (hasBrother ?x ?y)) (assertOWLProperty hasBrother ?x ?y) )
(defrule Def-hasAunt (hasParent ?x ?y) (hasSister ?y ?z) => (assert (hasAunt ?x ?z)) (assertOWLProperty hasAunt ?x ?z) )
(defrule Def-hasMother (hasParent ?x ?y) (Woman (name ?y)) => (assert (hasMother ?x ?y)) (assertOWLProperty hasMother ?x ?y) )
(defrule Def-hasParent (hasConsort ?y ?z) (hasParent ?x ?y) => (assert (hasParent ?x ?z)) (assertOWLProperty hasParent ?x ?z) )
(defrule Def-hasFather (hasParent ?x ?y) (Man (name ?y)) => (assert (hasFather ?x ?y)) (assertOWLProperty hasFather ?x ?y) )
(defrule Def-hasSister (hasSibling ?x ?y) (Woman (name ?y)) => (assert (hasSister ?x ?y)) (assertOWLProperty hasSister ?x ?y) )
(defrule Def-hasSibling (hasChild ?y ?x) (hasChild ?y ?z) (differentFrom ?x ?z) => (assert (hasSibling ?x ?z)) (assertOWLProperty hasSibling ?x ?z) )
```


family.swrl Protégé 3.3 beta (file:IC:\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

Name	Expression
Def-hasAunt	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{hasSister}(?y, ?z) \rightarrow \text{hasAunt}(?x, ?z)$
Def-hasBrother	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasBrother}(?x, ?y)$
Def-hasDaughter	$\rightarrow \text{hasChild}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasDaughter}(?x, ?y)$
Def-hasFather	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasFather}(?x, ?y)$
Def-hasMother	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasMother}(?x, ?y)$
Def-hasNephew	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{hasSon}(?y, ?z) \rightarrow \text{hasNephew}(?x, ?z)$
Def-hasNiece	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{hasDaughter}(?y, ?z) \rightarrow \text{hasNiece}(?x, ?z)$
Def-hasParent	$\rightarrow \text{hasConsort}(?y, ?z) \wedge \text{hasParent}(?x, ?y) \rightarrow \text{hasParent}(?x, ?z)$
Def-hasSibling	$\rightarrow \text{hasChild}(?y, ?x) \wedge \text{hasChild}(?y, ?z) \wedge \text{differentFrom}(?x, ?z) \rightarrow \text{hasSibling}(?x, ?z)$
Def-hasSister	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasSister}(?x, ?y)$
Def-hasSon	$\rightarrow \text{hasChild}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasSon}(?x, ?y)$
Def-hasUncle	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{hasBrother}(?y, ?z) \rightarrow \text{hasUncle}(?x, ?z)$

Jess Control Rules Classes Properties Individuals Restrictions Asserted Individuals Asserted Properties

Jess Class Definitions

- (deftemplate Nephew extends Relative)
- (deftemplate Son extends Child)
- (deftemplate owl:Thing (slot name))
- (deftemplate Relative extends Person)
- (deftemplate Sibling extends Person)
- (deftemplate Aunt extends Relative)
- (deftemplate Person extends owl:Thing)
- (deftemplate Mother extends Parent)
- (deftemplate Niece extends Relative)
- (deftemplate Daugther extends Child)
- (deftemplate Father extends Parent)
- (deftemplate Parent extends Person)
- (deftemplate Sister extends Sibling)
- (deftemplate Brother extends Sibling)
- (deftemplate Child extends Person)
- (deftemplate Uncle extends Relative)
- (deftemplate Woman extends Person)
- (deftemplate Man extends Person)



SWRL Rules

Name	Expression
Def-hasAunt	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{hasSister}(?y, ?z) \rightarrow \text{hasAunt}(?x, ?z)$
Def-hasBrother	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasBrother}(?x, ?y)$
Def-hasDaughter	$\rightarrow \text{hasChild}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasDaughter}(?x, ?y)$
Def-hasFather	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasFather}(?x, ?y)$
Def-hasMother	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasMother}(?x, ?y)$
Def-hasNephew	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{hasSon}(?y, ?z) \rightarrow \text{hasNephew}(?x, ?z)$
Def-hasNiece	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{hasDaughter}(?y, ?z) \rightarrow \text{hasNiece}(?x, ?z)$
Def-hasParent	$\rightarrow \text{hasConsort}(?y, ?z) \wedge \text{hasParent}(?x, ?y) \rightarrow \text{hasParent}(?x, ?z)$
Def-hasSibling	$\rightarrow \text{hasChild}(?y, ?x) \wedge \text{hasChild}(?y, ?z) \wedge \text{differentFrom}(?x, ?z) \rightarrow \text{hasSibling}(?x, ?z)$
Def-hasSister	$\rightarrow \text{hasSibling}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasSister}(?x, ?y)$
Def-hasSon	$\rightarrow \text{hasChild}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasSon}(?x, ?y)$
Def-hasUncle	$\rightarrow \text{hasParent}(?x, ?y) \wedge \text{hasBrother}(?y, ?z) \rightarrow \text{hasUncle}(?x, ?z)$

SWRLJessTab

See <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessTab> for SWRLJessTab documentation.

Press the "OWL+SWRL->Jess" button to transfer SWRL rules and relevant OWL knowledge to Jess.
Press the "Run Jess" button to run the Jess rule engine.
Press the "Jess->OWL" button to transfer the inferred Jess knowledge to OWL knowledge.

IMPORTANT: With the exception of owl:sameAs, owl:differentFrom and owl:allDifferent, owl:equivalentProperty, and owl:equivalentClass, the Jess rule engine is currently ignoring OWL restrictions. To ensure consistency, a reasoner should be run on an OWL knowledge base before SWRL rules and OWL knowledge are transferred to Jess. Also, if inferred knowledge from Jess is inserted back into an OWL knowledge base, a reasoner should again be executed to ensure that the new knowledge does not conflict with OWL restrictions in that knowledge base.

cf. <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLRuleEngineBridgeFAQ#nid6QL> for details.

OWL+SWRL->Jess Run Jess Jess->OWL

Outstanding Issues

- SWRL Bridge does not know about all OWL restrictions:
 - Contradictions with rules possible!
 - Consistency must be assured by the user incrementally running a reasoner.
 - Hard problem to solve in general.
- Integrated reasoner and rule engine would be ideal.
- Current solution with Pellet, though only with core built-in libraries.

SWRLTab Java APIs

- The SWRLTab provides APIs for all components
- These APIs are accessible to all OWL Protégé-OWL developers.
- Third party software can use these APIs to work directly with SWRL rules and integrate rules into their applications
- Fully documented in SWRLTab Wiki

SWRL and Querying

- SWRL is a rule language, not a query language
- However, a rule antecedent can be viewed as a pattern matching specification, i.e., a query
- With built-ins, language compliant query extensions are possible
- Hence: SQWRL (Semantic Query-Enhanced Web Rule Language; pronounced *squirrel*)

Example SQWRL Query

Person(?p) ^ hasAge(?p,?age)
^ swrlb:greaterThan(?age,17)
→ **sqwrl:select**(?p, ?age)

Ordering Query Results

Person(?p) ^ hasAge(?p,?age)
^ swrlb:greaterThan(?age,17)
→ sqwrl:select(?p, ?age) ^
sqwrl:orderBy(?age)

Counting Query Results

Person(?p) ^ hasCar(?p,?car)

→ sqwrl:select(?p) ^
sqwrl:count(?car)

Important: no way of asserting count in ontology!

Count all Owned Cars in Ontology

Person(?p) ^ hasCar(?p, ?c) →
sqwrl:count(?c)

Count all Cars in Ontology

`Car(?c) → sqwrl:count(?c)`

Aggregation Queries: average age of persons in ontology

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

Also: `sqwrl:max`, `sqwrl:min`, `sqwrl:sum`

Queries and Rules Can Interact

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

Adult(?a) → **sqwrl:select(?a)**

Example SWRL Query with OWL Restrictions

$(\text{hasChild} \geq 1)(?x) \rightarrow \text{sqwrl:select}(?x)$

SQWRL can act as a DL query language

All Built-ins can be used in Queries

tbox:isDirectSubClassOf(?subClass, Person)

-> sqwrl:select(?subClass)

tbox:isSubPropertyOf(?supProperty, hasName)

-> sqwrl:select(?subProperty)

Note: use of property and class names as built-in arguments is not OWL DL

Important: these built-ins should be used in queries only – inference with them would definitely not be OWL DL

SQWRL Language FAQ

ProtegeWiki: SQWRL - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL

Most Visited

Google sqwrl

ProtegeWiki: SQWRL Mozilla Firefox Start Page Mozilla Firefox Start Page Mozilla Firefox Start Page Mozilla Firefox Start Page

SQWRL

WikiHomePage | RecentChanges | Page Index

MartinOConnor (preferences | logout)

SQWRL (Semantic Query-Enhanced Web Rule Language; pronounced *squirrel*) is a [SWRL](#)-based query language that can be used to query OWL ontologies. SQWRL provides SQL-like operations to format knowledge retrieved from an OWL ontology. SQWRL does not alter SWRL's semantics and uses the standard [SWRL](#) presentation syntax supported by the [SWRLTab](#). (A3M)

SQWRL is defined using a library of [SWRL built-ins](#) that effectively turns SWRL into a query language. These built-ins are defined in the [SQWRL Ontology](#). It has the default namespace prefix `sqwrl`. A copy of this file can be found the standard Protege-OWL repositories, and can be imported through the 'Import Ontology' option in the Metadata tab. (A10)

This document assumes that the reader is already familiar with SWRL. An introduction can be found [here](#). (A02)

Basic Queries (A11)

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 the `hasCar` property. (A12)

Here, for example, is a simple SQWRL query to extract all known persons in an ontology whose age is less than 25, together with their ages: (A13)

- Person(?p) ^ hasAge(?p, ?a) ^ swrlb:lessThan(?a, 25) -> sqwrl:select(?p, ?a) (A14)

This query will return pairs of individuals and ages. (A15)

To list all the cars owned by each person, we can write: (A16)

- Person(?p) ^ hasCar(?p, ?c) -> sqwrl:select(?p, ?c) (A17)

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. (A18)

The `select` operator accepts a variable number of arguments. Literal values can also be passed to it.

Done

Your Visited Pages

- SQWRL
- SQWRLQueryTab
- SQWRLQueryAPI
- SWRLTabBuiltinLibraries
- SWRLLanguageFAQ
- SWRLTab
- SWRLTabMathematicalBuiltIns

View Backlinks

Search

SQWRLTab

- Available as part of Protégé-OWL
SWRLTab in current Protégé-3.4 beta
- Graphical interface to execute queries
- Low-level JDBC-like API for use in
embedded applications
- Can use any existing rule engine back end

family.sqwr1 Protégé 3.4 (file:IC:\Development\SWRL\kbs\family.sqwr1.pprj, OWL / RDF Files)

File Edit Project OWL Reasoning Code Tools Window Help

Metadata(ontology) OWLClasses Properties Individuals Forms **SWRL Rules**

SWRL Rules

Enabled	Name	Expression
<input checked="" type="checkbox"/>	Def-hasAunt	→ Person(?x) ∧ hasParent(?x, ?y) ∧ hasSister(?y, ?z) → hasAunt(?x, ?z)
<input checked="" type="checkbox"/>	Def-hasBrother	→ Person(?x) ∧ hasSibling(?x, ?y) ∧ Man(?y) → hasBrother(?x, ?y)
<input checked="" type="checkbox"/>	Def-hasDaughter	→ Person(?x) ∧ hasChild(?x, ?y) ∧ Woman(?y) → hasDaughter(?x, ?y)
<input checked="" type="checkbox"/>	Def-hasFather	→ Person(?x) ∧ hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
<input checked="" type="checkbox"/>	Def-hasMother	→ Person(?x) ∧ hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
<input checked="" type="checkbox"/>	Def-hasNephew	→ Person(?x) ∧ hasSibling(?x, ?y) ∧ hasSon(?y, ?z) → hasNephew(?x, ?z)
<input checked="" type="checkbox"/>	Def-hasNiece	→ Person(?x) ∧ hasSibling(?x, ?y) ∧ hasDaughter(?y, ?z) → hasNiece(?x, ?z)
<input checked="" type="checkbox"/>	Def-hasParent	→ Person(?y) ∧ hasConsort(?y, ?z) ∧ hasParent(?x, ?y) → hasParent(?x, ?z)
<input checked="" type="checkbox"/>	Def-hasSibling	→ Person(?y) ∧ hasChild(?y, ?x) ∧ hasChild(?y, ?z) ∧ differentFrom(?x, ?z) → hasSibling(?x, ?z)
<input checked="" type="checkbox"/>	Def-hasSister	→ Person(?x) ∧ hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
<input checked="" type="checkbox"/>	Def-hasSon	→ Person(?x) ∧ hasChild(?x, ?y) ∧ Man(?y) → hasSon(?x, ?y)
<input checked="" type="checkbox"/>	Def-hasUncle	→ Person(?x) ∧ hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)
<input checked="" type="checkbox"/>	Query-1	→ hasSon(?x, ?z) → sqwrl:select(?x, ?z)
<input checked="" type="checkbox"/>	Query-2	→ hasSon(?x, ?z) → sqwrl:select(?x) ∧ sqwrl:count(?z) ∧ sqwrl:orderByDescending(?z)
<input checked="" type="checkbox"/>	Query-3	→ Person(?p) → sqwrl:select(?p)
<input checked="" type="checkbox"/>	Query-4	→ Man(?m) → sqwrl:select(?m)
<input checked="" type="checkbox"/>	Query-5	→ Woman(?w) → sqwrl:count(?w)
<input checked="" type="checkbox"/>	Query-6	→ Woman(?w) ∧ hasChild(?w, ?c) → sqwrl:select(?w) ∧ sqwrl:count(?w)

SQWRLQueryTab

See <http://protege.cim3.net/cgi-bin/wiki.pl?SQWRLQueryTab> for documentation.

Executing queries in this tab does not modify the ontology.

Select a SQWRL query from the list above and press the 'Run' button.

If the selected query generates a result, the result will appear in a new sub tab.

Run

family.sqwr1 Protégé 3.4 (file:IC:\Development\SWRL\kbs\family.sqwr1.pprj, OWL / RDF Files)

File Edit Project OWL Reasoning Code Tools Window Help

protégé

Metadata(ontology) OWLClass

SWRL Rule

Enabled	Name	
<input checked="" type="checkbox"/>	Def-hasAunt	→ Perso
<input checked="" type="checkbox"/>	Def-hasBrother	→ Perso
<input checked="" type="checkbox"/>	Def-hasDaughter	→ Perso
<input checked="" type="checkbox"/>	Def-hasFather	→ Perso
<input checked="" type="checkbox"/>	Def-hasMother	→ Perso
<input checked="" type="checkbox"/>	Def-hasNephew	→ Perso
<input checked="" type="checkbox"/>	Def-hasNiece	→ Perso
<input checked="" type="checkbox"/>	Def-hasParent	→ Perso
<input checked="" type="checkbox"/>	Def-hasSibling	→ Perso
<input checked="" type="checkbox"/>	Def-hasSister	→ Perso
<input checked="" type="checkbox"/>	Def-hasSon	→ Perso
<input checked="" type="checkbox"/>	Def-hasUncle	→ Perso
<input checked="" type="checkbox"/>	Query-1	→ hasS
<input checked="" type="checkbox"/>	Query-2	→ hasS
<input checked="" type="checkbox"/>	Query-3	→ Perso
<input checked="" type="checkbox"/>	Query-4	→ Man(
<input checked="" type="checkbox"/>	Query-5	→ Wom
<input checked="" type="checkbox"/>	Query-6	→ Wom

SQLQueryTab

See <http://protege.cim3.net/cgi-bin/wiki.pl?>

Executing queries in this tab does not mod

Select a SQWRL query from the list above

If the selected query generates a result, th

SWRL Rule

Name	Comment
<input type="text" value="http://a.com/ontology#Query-3"/>	

SWRL Rule

```
Person(?p)
→ sqwrl:select(?p)
```

● ■ ◆ ? + B ≠ = ✓

^ → () [] ←

Run

SWRL Rules

Enabled	Name	Expression
<input checked="" type="checkbox"/>	Def-hasAunt	$\rightarrow \text{Person}(?x) \wedge \text{hasParent}(?x, ?y) \wedge \text{hasSister}(?y, ?z) \rightarrow \text{hasAunt}(?x, ?z)$
<input checked="" type="checkbox"/>	Def-hasBrother	$\rightarrow \text{Person}(?x) \wedge \text{hasSibling}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasBrother}(?x, ?y)$
<input checked="" type="checkbox"/>	Def-hasDaughter	$\rightarrow \text{Person}(?x) \wedge \text{hasChild}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasDaughter}(?x, ?y)$
<input checked="" type="checkbox"/>	Def-hasFather	$\rightarrow \text{Person}(?x) \wedge \text{hasParent}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasFather}(?x, ?y)$
<input checked="" type="checkbox"/>	Def-hasMother	$\rightarrow \text{Person}(?x) \wedge \text{hasParent}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasMother}(?x, ?y)$
<input checked="" type="checkbox"/>	Def-hasNephew	$\rightarrow \text{Person}(?x) \wedge \text{hasSibling}(?x, ?y) \wedge \text{hasSon}(?y, ?z) \rightarrow \text{hasNephew}(?x, ?z)$
<input checked="" type="checkbox"/>	Def-hasNiece	$\rightarrow \text{Person}(?x) \wedge \text{hasSibling}(?x, ?y) \wedge \text{hasDaughter}(?y, ?z) \rightarrow \text{hasNiece}(?x, ?z)$
<input checked="" type="checkbox"/>	Def-hasParent	$\rightarrow \text{Person}(?y) \wedge \text{hasConsort}(?y, ?z) \wedge \text{hasParent}(?x, ?y) \rightarrow \text{hasParent}(?x, ?z)$
<input checked="" type="checkbox"/>	Def-hasSibling	$\rightarrow \text{Person}(?y) \wedge \text{hasChild}(?y, ?x) \wedge \text{hasChild}(?y, ?z) \wedge \text{differentFrom}(?x, ?z) \rightarrow \text{hasSibling}(?x, ?z)$
<input checked="" type="checkbox"/>	Def-hasSister	$\rightarrow \text{Person}(?x) \wedge \text{hasSibling}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasSister}(?x, ?y)$
<input checked="" type="checkbox"/>	Def-hasSon	$\rightarrow \text{Person}(?x) \wedge \text{hasChild}(?x, ?y) \wedge \text{Man}(?y) \rightarrow \text{hasSon}(?x, ?y)$
<input checked="" type="checkbox"/>	Def-hasUncle	$\rightarrow \text{Person}(?x) \wedge \text{hasParent}(?x, ?y) \wedge \text{hasBrother}(?y, ?z) \rightarrow \text{hasUncle}(?x, ?z)$
<input checked="" type="checkbox"/>	Query-1	$\rightarrow \text{hasSon}(?x, ?z) \rightarrow \text{sqwrl:select}(?x, ?z)$
<input checked="" type="checkbox"/>	Query-2	$\rightarrow \text{hasSon}(?x, ?z) \rightarrow \text{sqwrl:select}(?x) \wedge \text{sqwrl:count}(?z) \wedge \text{sqwrl:orderByDescending}(?z)$
<input checked="" type="checkbox"/>	Query-3	$\rightarrow \text{Person}(?p) \rightarrow \text{sqwrl:select}(?p)$
<input checked="" type="checkbox"/>	Query-4	$\rightarrow \text{Man}(?m) \rightarrow \text{sqwrl:select}(?m)$
<input checked="" type="checkbox"/>	Query-5	$\rightarrow \text{Woman}(?w) \rightarrow \text{sqwrl:count}(?w)$
<input checked="" type="checkbox"/>	Query-6	$\rightarrow \text{Woman}(?w) \wedge \text{hasChild}(?w, ?c) \rightarrow \text{sqwrl:select}(?w) \wedge \text{sqwrl:count}(?w)$

?p

M06
F10
F05
F09
M05
M10
F06
M02
M09
M04
F01
F02
F07
M03
M08
F04

SWRLTab Wiki : <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab>

SWRLTab

[WikiHomePage](#) | [RecentChanges](#) | [Page Index](#)

MartinOConnor ([preferences](#) | [logout](#))

The SWRLTab is a development environment for working with [SWRL](#) rules in [Protege-OWL](#). It supports the editing and execution of SWRL rules. It provides a set of libraries that can be used in rules, including libraries to interoperate with XML documents, and spreadsheets, and libraries with mathematical, string, RDFS, and temporal operators. A SWRL-based OWL query language called SQWRL is also provided. [\(A3E\)](#)

An introduction to the SWRL language can be found [here](#). An introduction SQWRL can be found [here](#). [\(9EQ\)](#)

The SWRLTab has several software components: [\(6GY\)](#)

- **SWRL Editor** The editor supports editing and saving of SWRL rules in an OWL ontology. See the [SWRL Editor FAQ](#) for more details. [\(8GS\)](#)
- **SWRL Built-in Libraries** A number of built-in libraries are provided by the SWRLTab. These include an implementation of the core SWRL built-ins defined in the [SWRL Submission](#) and built-ins for querying OWL ontologies. The libraries are documented [here](#). [\(8D1\)](#)
- **SQWRL Query Tab** The query tab provides a graphical interface to display the results of SQWRL queries. It is documented [here](#). [\(8XC\)](#)
- **SQWRL Query API** This API provides a JDBC-like Java interface to retrieve the result of SQWRL queries. It is documented [here](#). [\(8XD\)](#)
- **SWRL Built-in Bridge** [SWRL built-ins](#) are user-defined predicates that can be used in SWRL rules. The SWRLTab has a subcomponent called the [built-in bridge](#) that provides a mechanism to define Java implementations of SWRL built-ins. These implementations can then be dynamically loaded by the bridge and invoked from a rule engine. [\(6H2\)](#)
- **SWRL Bridge** The bridge provides the infrastructure necessary to incorporate rule engines into Protege-OWL to execute SWRL rules. See the [SWRL Rule Engine Bridge FAQ](#) for more details. The hope is that bridges for other rule engines will be developed by the Protege-OWL community and than an array of inference mechanism will become available for executing SWRL rules. [\(6H1\)](#)
- **SWRL Jess Bridge** A bridge for the Jess rule engine is provided in the Protege-OWL distribution. A user interface called the SWRLJessTab is also provided to interact with this bridge. [\(8P4\)](#)
- **SWRL Factory** The factory provides high-level Java APIs that support the creation and modification of SWRL rules in an OWL ontology. This API can be used by developers who wish to work with SWRL rules in their applications. See the [SWRL Factory FAQ](#) for more details. [\(6H0\)](#)

The SWRLTab is designed to work in Protege-OWL only. However, partial interoperation with Jena is also supported. See [here](#) for details. Work has proceeded to port it to Protege 4. [\(A6A\)](#)

Your Visited Pages

- SWRLTab
- SWRLTabSyntax
- SWRLLanguageFAQ
- SWRLTabXMLBuiltIns
- SWRLTabBuiltInLibraries
- SWRLTemporalBuiltIns
- SWRLEditorFAQ
- SWRLBoxBuiltIns

View Backlinks

Search

Done