

# SWRLTab: A Development Environment for working with SWRL Rules In Protégé-OWL

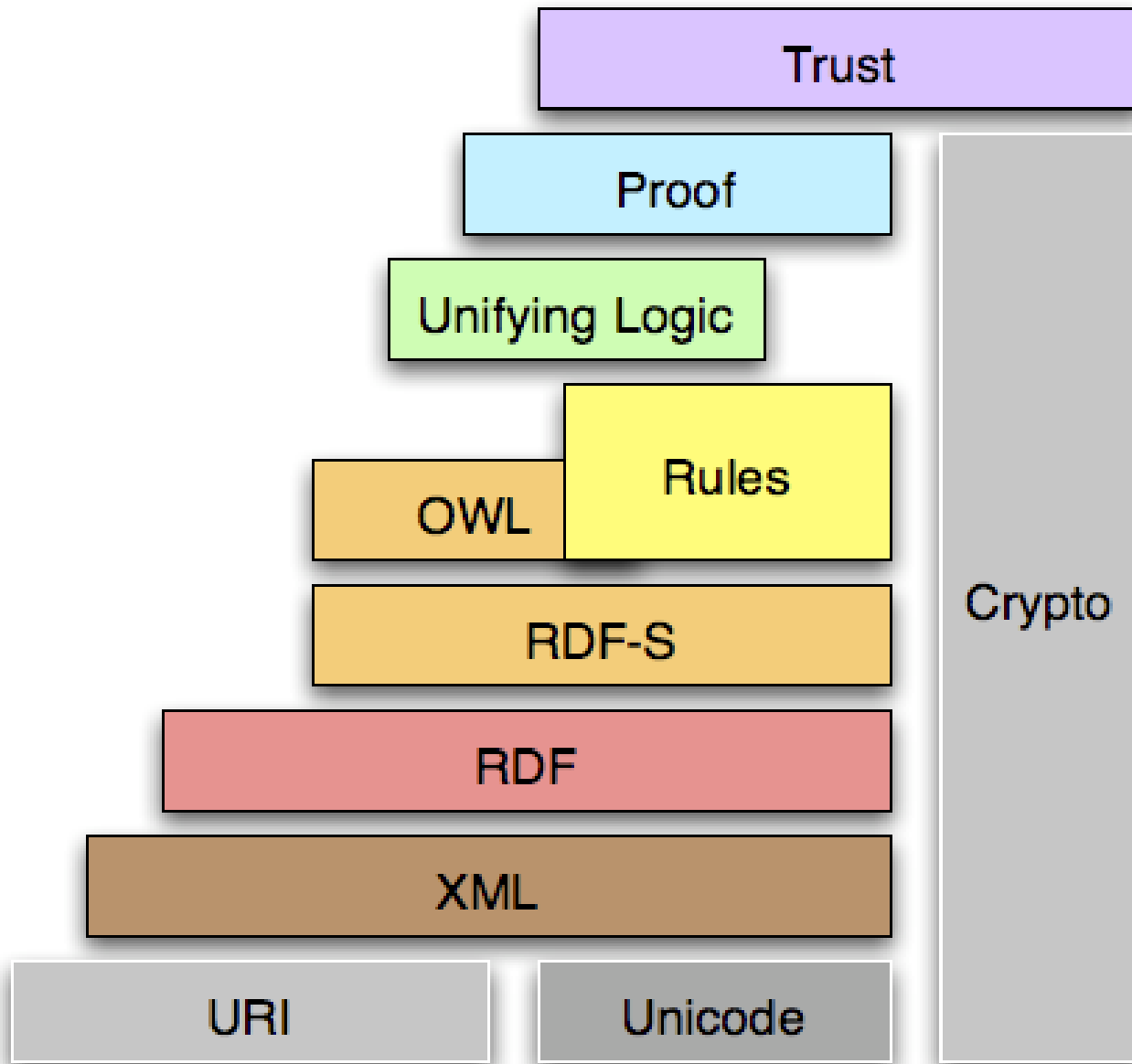
Martin O'Connor  
Stanford Medical Informatics, Stanford University



# Talk Outline

- Introduction to SWRL
- Using SWRL as an OWL query language
- SWRLTab: a Protégé-OWL development environment for SWRL

# 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 interoperability 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: Has uncle

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



# 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)

# 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)

# Example SWRL Rule with OWL Restrictions

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

# Example SWRL Rule with Inferred OWL Restrictions

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

# SWRL and Open World Semantics: sameAs, differentFrom

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



# SWRL is monotonic: does not Support Negated Atoms

$\text{Person}(?p) \wedge \text{not hasCar}(?p, ?c) \rightarrow$   
 $\text{CarlessPerson}(?p)$

**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)

# 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? 20

# SWRL is Monotonic: counting not supported II

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

**Closure:** though best expressed in OWL

# 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)

# 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

## Example SWRL Query

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



# Ordering Query Results

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

# Counting Query Results

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

→ query:select(?p) ^

query:count(?car)

**Important:** no way of asserting count in ontology

# Count all Owned Cars in Ontology

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

# Count all Cars in Ontology

`Car(?c) → query:count(?c)`

# Aggregation Queries: average age of persons in ontology

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

Also: `query:max`, `query:min`, `query:sum`

# Queries and Rules Can Interact

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

Adult(?a) → query:select(?a)

# Example SWRL Query with OWL Restrictions

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

# All Built-ins can be used in Queries

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

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

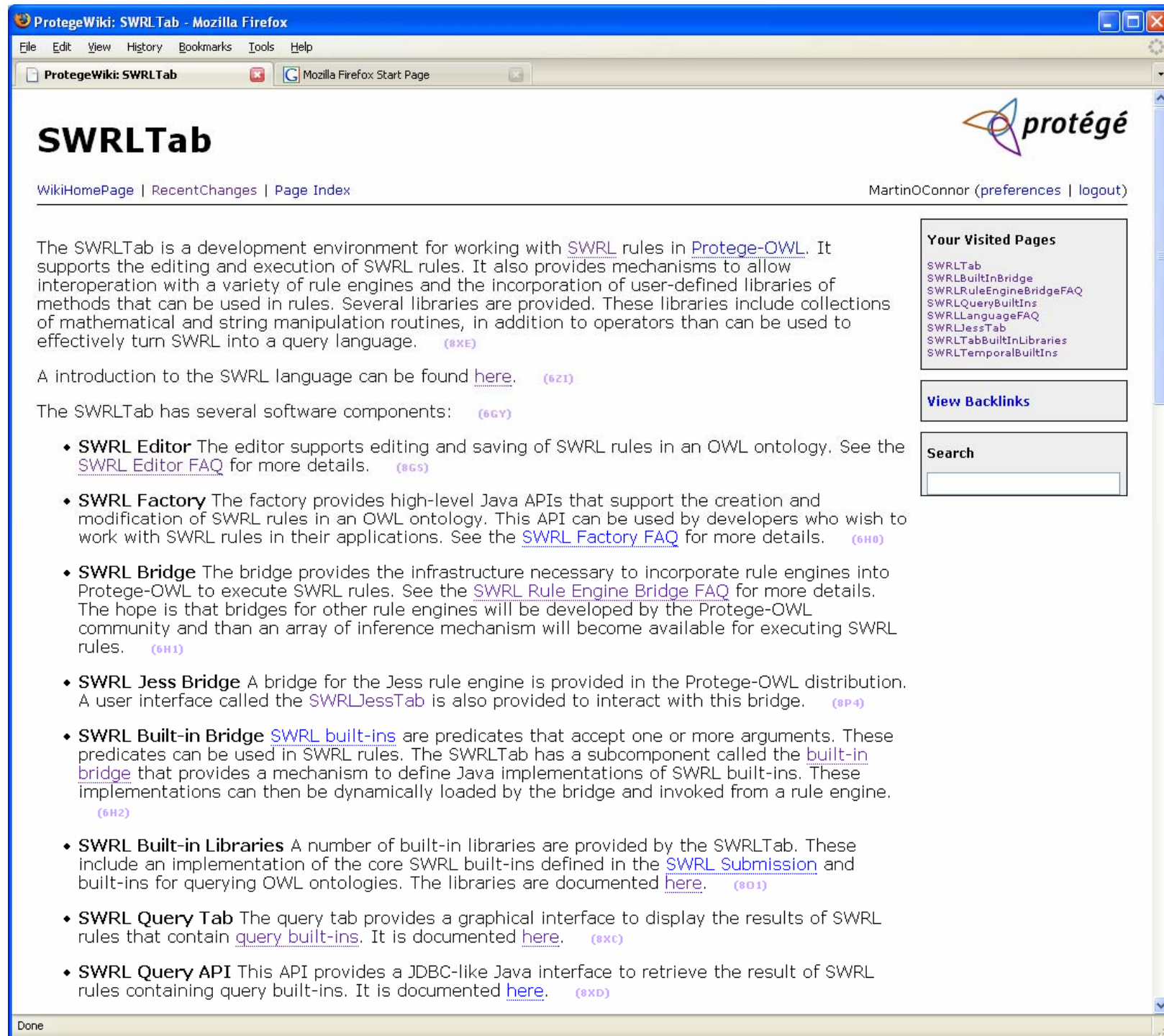
**Note:** use of property and class names as built-in arguments in not OWL DL

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



# 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



The screenshot shows a Mozilla Firefox browser window displaying the ProtegeWiki page for SWRLTab. The browser's address bar shows the URL <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab>. The page title is "SWRLTab" and the Protege logo is visible in the top right corner. The page content includes a navigation menu with links for "WikiHomePage", "RecentChanges", and "Page Index". The user "MartinOConnor" is logged in, with links for "preferences" and "logout". The main text describes the SWRLTab as a development environment for working with SWRL rules in Protege-OWL, supporting editing and execution of SWRL rules and providing mechanisms for interoperability with various rule engines. A list of software components follows, including the SWRL Editor, SWRL Factory, SWRL Bridge, SWRL Jess Bridge, SWRL Built-in Bridge, SWRL Built-in Libraries, SWRL Query Tab, and SWRL Query API. On the right side, there are sections for "Your Visited Pages" (listing various SWRLTab-related pages), "View Backlinks", and a "Search" box.

**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 also provides mechanisms to allow interoperation with a variety of rule engines and the incorporation of user-defined libraries of methods that can be used in rules. Several libraries are provided. These libraries include collections of mathematical and string manipulation routines, in addition to operators than can be used to effectively turn SWRL into a query language. [\(8XE\)](#)

A introduction to the SWRL language can be found [here](#). [\(6Z1\)](#)

The SWRLTab has several software components: [\(6G7\)](#)

- **SWRL Editor** The editor supports editing and saving of SWRL rules in an OWL ontology. See the [SWRL Editor FAQ](#) for more details. [\(8G5\)](#)
- **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. [\(6H0\)](#)
- **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 Built-in Bridge** [SWRL built-ins](#) are predicates that accept one or more arguments. These predicates 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 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](#). [\(801\)](#)
- **SWRL Query Tab** The query tab provides a graphical interface to display the results of SWRL rules that contain [query built-ins](#). It is documented [here](#). [\(8XC\)](#)
- **SWRL Query API** This API provides a JDBC-like Java interface to retrieve the result of SWRL rules containing query built-ins. It is documented [here](#). [\(8XD\)](#)

**Your Visited Pages**

- [SWRLTab](#)
- [SWRLBuiltInBridge](#)
- [SWRLRuleEngineBridgeFAQ](#)
- [SWRLQueryBuiltIns](#)
- [SWRLLanguageFAQ](#)
- [SWRLJessTab](#)
- [SWRLTabBuiltInLibraries](#)
- [SWRLTemporalBuiltIns](#)

**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 / ...)

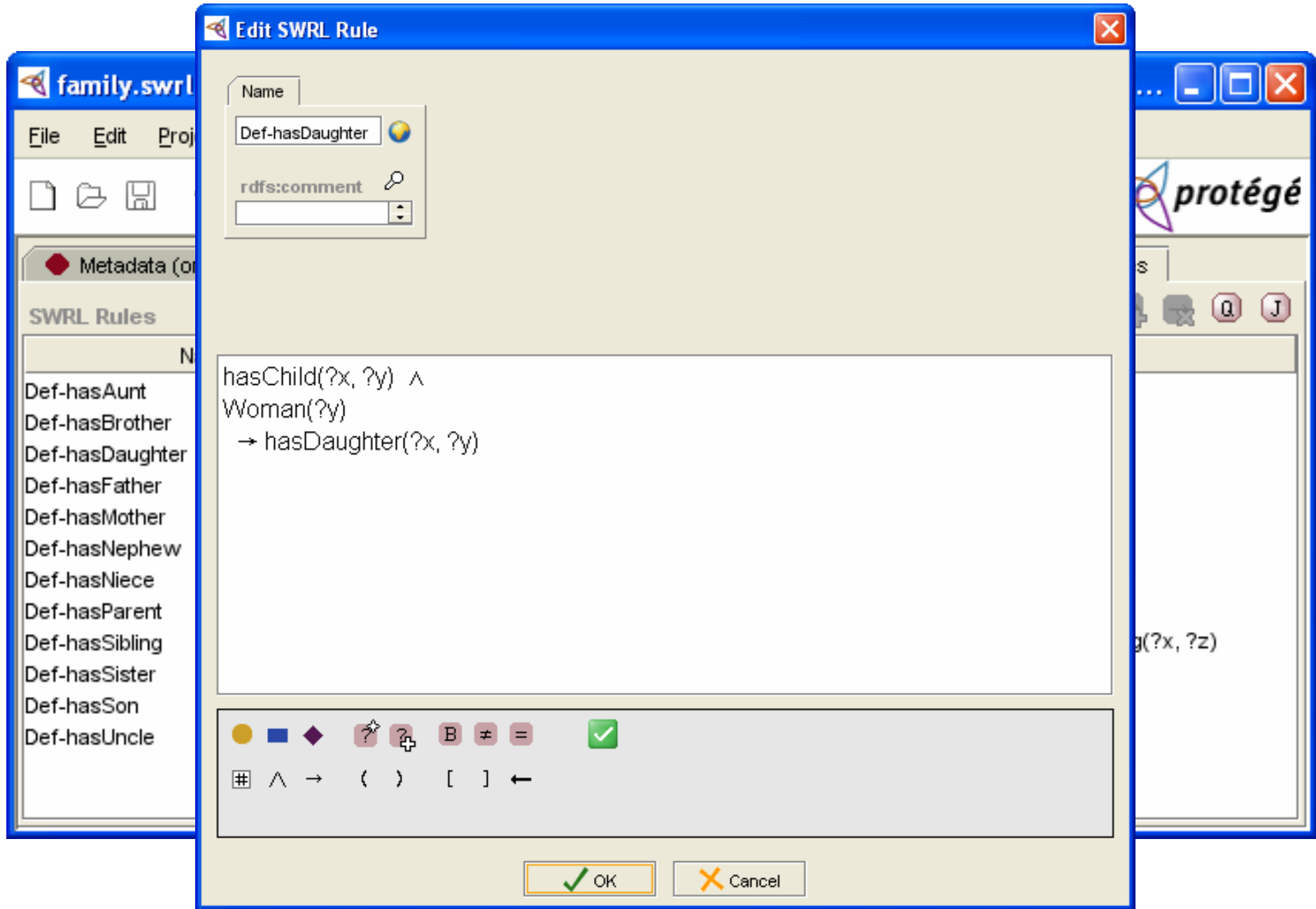
File Edit Project OWL Code Tools Window Help

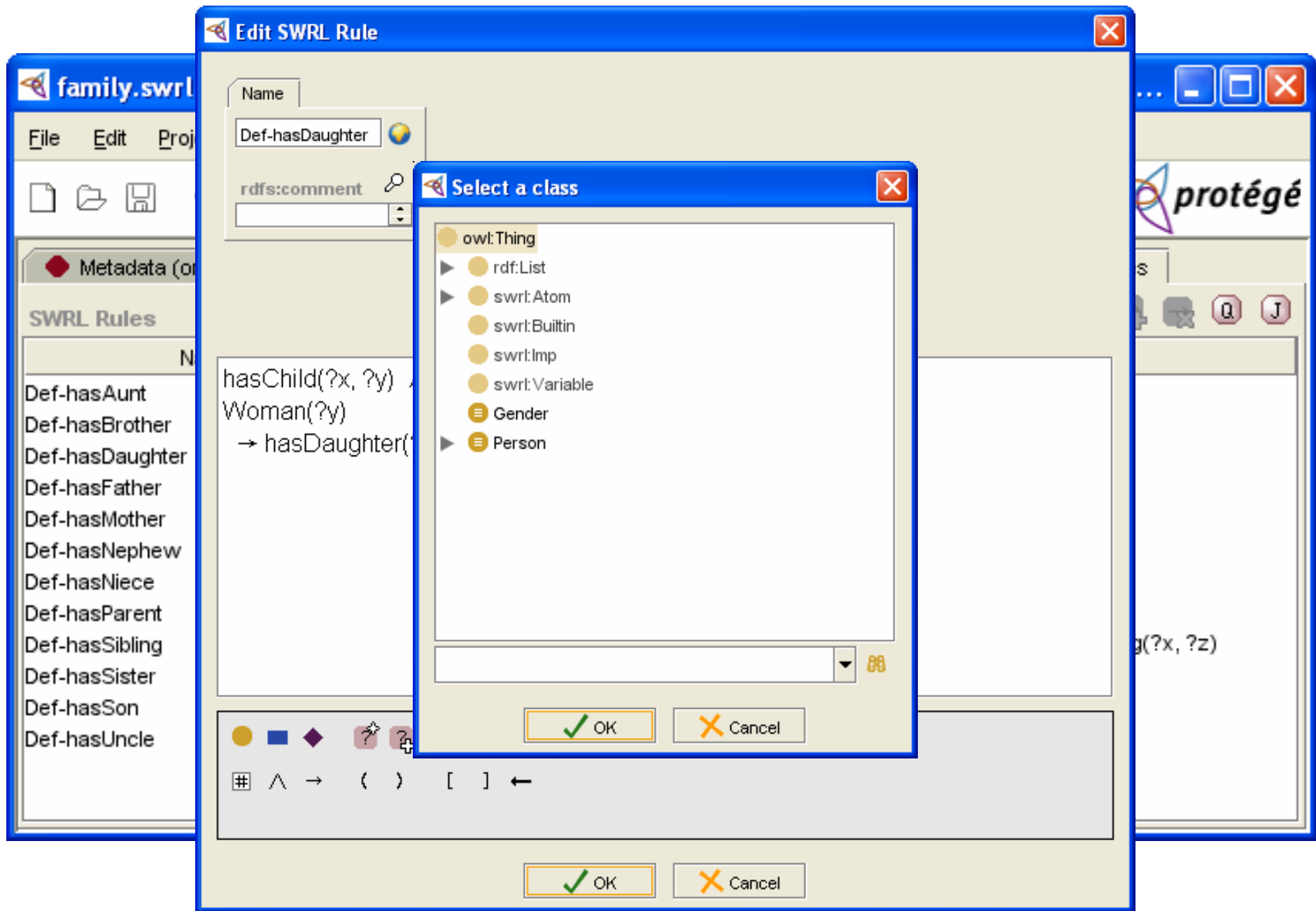
protégé

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





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

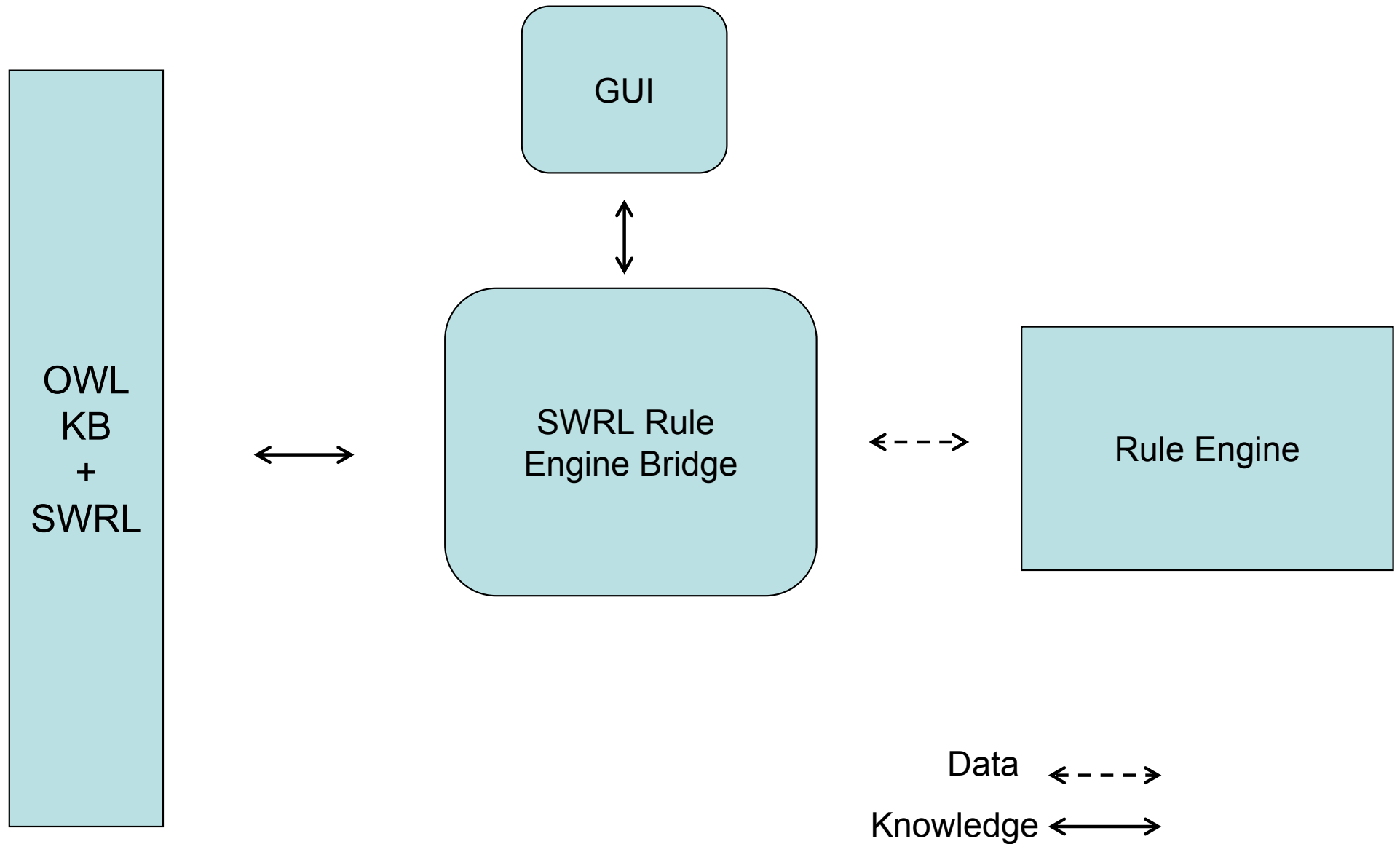
ling(?x, ?z)

# 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.

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	→ 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 Control Rules Classes Properties Individuals Restrictions Asserted Individuals Asserted Properties

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

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

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

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 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	→ 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 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 Daughter 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)

```

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	→ 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 Control → Rules → Classes → Properties → Individuals → Restrictions → Asserted Individuals → Asserted Properties

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.
- Possible solution with Pellet, KAON2.

# SWRL Built-in Bridge

- SWRL provides mechanisms to add user-defined predicates, e.g.,  
Person(?p) ^ hasAge(?p, ?age) ^  
**swrlb:greaterThanOrEqualTo**(?age, 18) -> Adult(?p)
- These built-ins could be implemented by each rule engine.
- However, the SWRL Bridge provides a dynamic loading mechanism for Java-defined built-ins.
- Can be used by any rule engine implementation.

# Defining a Built-in in Protégé-OWL

- Describe library of built-ins in OWL using definition of **swrl:Builtin** provided by SWRL ontology.
- Provide Java implementation of built-ins and wrap in JAR file.
- Load built-in definition ontology in Protégé-OWL. Put JAR in plugins directory.
- Built-in bridge will make run-time links.

# Example: defining `stringEqualIgnoreCase` from Core SWRL Built-ins Library

- Core SWRL built-ins defined by:
  - <http://www.w3.org/2003/11/swrlb>
- Provides commonly needed built-ins, e.g., add, subtract, string manipulation, etc.
- Normally aliased as 'swrlb'.
- Contains definition for `stringEqualIgnoreCase`

# Example Implementation Class for Core SWRL Built-in Methods

```
package edu.stanford.smi.protege.owl.swrl.bridge.builtins.swrlb;  
  
import edu.stanford.smi.protege.owl.swrl.bridge.builtins.*;  
import edu.stanford.smi.protege.owl.swrl.bridge.exceptions.*;  
  
public class SWRLBuiltInLibraryImpl extends SWRLBuiltInLibrary  
{  
    public SWRLBuiltInMethodsImpl() { ...}  
    public void reset() {...}  
  
    public boolean stringEqualIgnoreCase(List arguments) throws BuiltInException { ... }  
    ....  
} // SWRLBuiltInLibraryImpl
```

# Example Implementation for Built-in swrlb:stringEquallgnoreCase

```
public boolean stringEquallgnoreCase(List<Argument> arguments) throws BuiltInException
{
    SWRLBuiltInUtil.checkNumberOfArgumentsEqualTo(2, arguments.size());

    String argument1 = SWRLBuiltInUtil.getArgumentAsString(1, arguments);
    String argument2 = SWRLBuiltInUtil.getArgumentAsString(2, arguments);

    return argument1.equalsIgnoreCase(argument2);
} // stringEquallgnoreCase
```

# Invocation from Rule Engine


- Use of `swrlb:stringEqualIgnoreCase` in rule should cause automatic invocation.
- SWRL rule engine bridge has an invocation method.
- Takes built-in name and arguments and performs method resolution, loading, and invocation.
- Efficiency a consideration: some methods should probably be implemented natively by rule engine, e.g., `add`, `subtract`, etc.

ProtegeWiki: SWRLQuery Built Ins - Mozilla Firefox

File Edit View History Bookmarks Tools Help

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

ProtegeWiki: SWRLQuery Built Ins Mozilla Firefox Start Page Mozilla Firefox Start Page Mozilla Firefox Start Page



# SWRLQueryBuiltIns

[WikiHomePage](#) | [RecentChanges](#) | [Page Index](#) MartinOConnor ([preferences](#) | [logout](#))

The SWRL Query Built-In Library is one of the [SWRLTabBuiltInLibraries](#). It defines a set of built-ins that can be used in SWRL rules to query OWL ontologies. The built-ins in this library can be used to effectively turn SWRL into a query language. They provide SQL-like operations retrieve knowledge from an OWL ontology. The resulting query language does not alter SWRL's semantics and uses the standard [presentation syntax](#) supported by the SWRLTab. (8V8)

These built-ins are defined in the [SWRL Query Ontology](#). It has the default namespace prefix `query`. 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. Java implementations for these built-ins are also included in the Protege-OWL 3.3 beta, build XXX distribution. (8V7)

## Basic Queries (8U1)

Assume we have a simple ontology with classes `Person`, which has subclasses `Male` and `Female` with associated properties `hasAge` and `hasName`, and a class `car`, that can be associated with individual of class `Person` through the `hasCar` property. (8WY)

Here, for example, is a simple SWRL query to extract all known persons in an ontology whose age is less than 25, together with their ages: (85W)

- `Person(?p) ^ hasAge(?p, ?a) ^ swrlb:lessThan(?a, 25) -> query:select(?p, ?a)` (85X)

**Your Visited Pages**

- [SWRLQueryBuiltIns](#)
- [SWRLLanguageFAQ](#)
- [SWRLTabBuiltInLibraries](#)
- [SWRLTab](#)
- [SWRLBuiltInBridge](#)
- [SWRLFactoryFAQ](#)
- [SWRLRuleEngineBridgeFAQ](#)
- [SWRLJesTab](#)

**View Backlinks**

**Search**

Done





## 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)
Query-name	→ Man(?x) ∧ name(?x, ?n) → query:select(?x, ?n)
Query-child	→ Man(?man) ∧ name(?man, ?n1) ∧ hasChild(?man, ?child) ∧ name(?child, ?n2) → query:select(?n1, ?n2)
Query-age	→ Person(?p) ∧ name(?p, ?n) → query:select(?p, ?n)

## SWRLQueryTab

## SWRLQueryTab

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

Executing rules in this tab does not modify the ontology.

Select a rule with query built-ins from the list above and press the 'Run Rules' button.  
If the selected rule generates a result, the result will appear in a new tab.

Run Rules

Run all SWRL rules



Metadata (ontology) OWLClasses

SWRL Rules

Name	
Def-hasAunt	→ hasParent(?x, ?y)
Def-hasBrother	→ hasSibling(?x, ?y)
Def-hasDaughter	→ hasChild(?x, ?y)
Def-hasFather	→ hasParent(?x, ?y)
Def-hasMother	→ hasParent(?x, ?y)
Def-hasNephew	→ hasSibling(?x, ?y)
Def-hasNiece	→ hasSibling(?x, ?y)
Def-hasParent	→ hasConsort(?x, ?y)
Def-hasSibling	→ hasChild(?x, ?y)
Def-hasSister	→ hasSibling(?x, ?y)
Def-hasSon	→ hasChild(?x, ?y)
Def-hasUncle	→ hasParent(?x, ?y)
Query-name	→ Man(?x) ^
Query-child	→ Man(?man) ^
Query-age	→ Person(?p)

Edit SWRL Rule

Name

Query-child

rdfs:comment

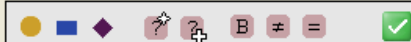


Annotations

Property	Value	Lang
rdfs:comment		

```

Man(?man) ^
name(?man, ?n1) ^
hasChild(?man, ?child) ^
name(?child, ?n2)
→ query:select(?n1, ?n2)
    
```



⊕ ^ → ( ) [ ] ←



?n2)



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)
Query-name	→ Man(?x) ∧ name(?x, ?n) → query:select(?x, ?n)
Query-child	→ Man(?man) ∧ name(?man, ?n1) ∧ hasChild(?man, ?child) ∧ name(?child, ?n2) → query:select(?n1, ?n2)
Query-age	→ Person(?p) ∧ name(?p, ?n) → query:select(?p, ?n)

?n1	?n2
Adam	Michael
Adam	Anna
Johon	Phillipe
George	Surrey
Bill	Elizabeth
Jimmy	Jack
Adam	George
Bill	Catherine
Tom	Ronald
Phillipe	Eva
Phillipe	Tom
Bill	Adam

Run Rules

Close

Save as CSV...

# SWRLQueryTab

- Available as part of Protégé-OWL SWRLTab in current Protégé-3.3.1
- Low-level JDBC-like API for use in embedded applications
- Can use any existing rule engine back end

# Other Built-in Libraries

The screenshot shows a Mozilla Firefox browser window with the title "ProtegeWiki: SWRLTab Built In Libraries - Mozilla Firefox". The address bar contains the URL "http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTabBuiltInLibraries". The page content includes a navigation menu with "WikiHomePage", "RecentChanges", and "Page Index". The main heading is "SWRLTabBuiltInLibraries" with the Protege logo to its right. Below the heading, there is a user profile for "MartinOConnor" with links for "preferences" and "logout". The main text states: "A number of built-in libraries are provided by the [SWRLTab](#). These include: (8AA)" followed by a bulleted list of libraries: "Core SWRL Built-Ins Library", "Query Built-In Library", "ABox Built-Ins Library", "TBox Built-Ins Library", and "Extensions Built-ins Library". On the right side, there is a "Your Visited Pages" sidebar with a list of links, a "View Backlinks" button, and a "Search" input field. At the bottom, there are links for "Edit text of this page" and "View other revisions", along with an "RSS" icon and a "Done" status bar.


ProtegeWiki: SWRLTab Built In Libraries - Mozilla Firefox

File Edit View History Bookmarks Tools Help

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

ProtegeWiki: SWRLTab Built In Li... Mozilla Firefox Start Page Mozilla Firefox Start Page Mozilla Firefox Start Page

## SWRLTabBuiltInLibraries



WikiHomePage | RecentChanges | Page Index

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

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)
- **Query Built-In Library** Defines a set of built-ins that can be used in SWRL rules to query OWL ontologies. It is documented [here](#). (8XB)
- **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)
- **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 using the [SWRLBuiltInBridge](#). (8AE)

[Edit text of this page](#) | [View other revisions](#)  
Last edited March 9, 2007 16:09 ([diff](#))

[RSS](#)

Done

**Your Visited Pages**

- SWRLTabBuiltInLibraries
- SWRLTab
- SWRLBuiltInBridge
- SWRLFactoryFAQ
- SWRLRuleEngineBridgeFAQ
- SWRLQueryBuiltIns
- SWRLLanguageFAQ
- SWRLJessTab

[View Backlinks](#)

**Search**

# 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

# Future Plans

- Port to Protégé 4
- Integrated reasoner/inference support, most likely with Pellet
- Dynamic relational-OWL mapping for inferencing and querying (static already available with Datamaster)
- SQWRL ('squirrel'): enhanced query support – negation, disjunction