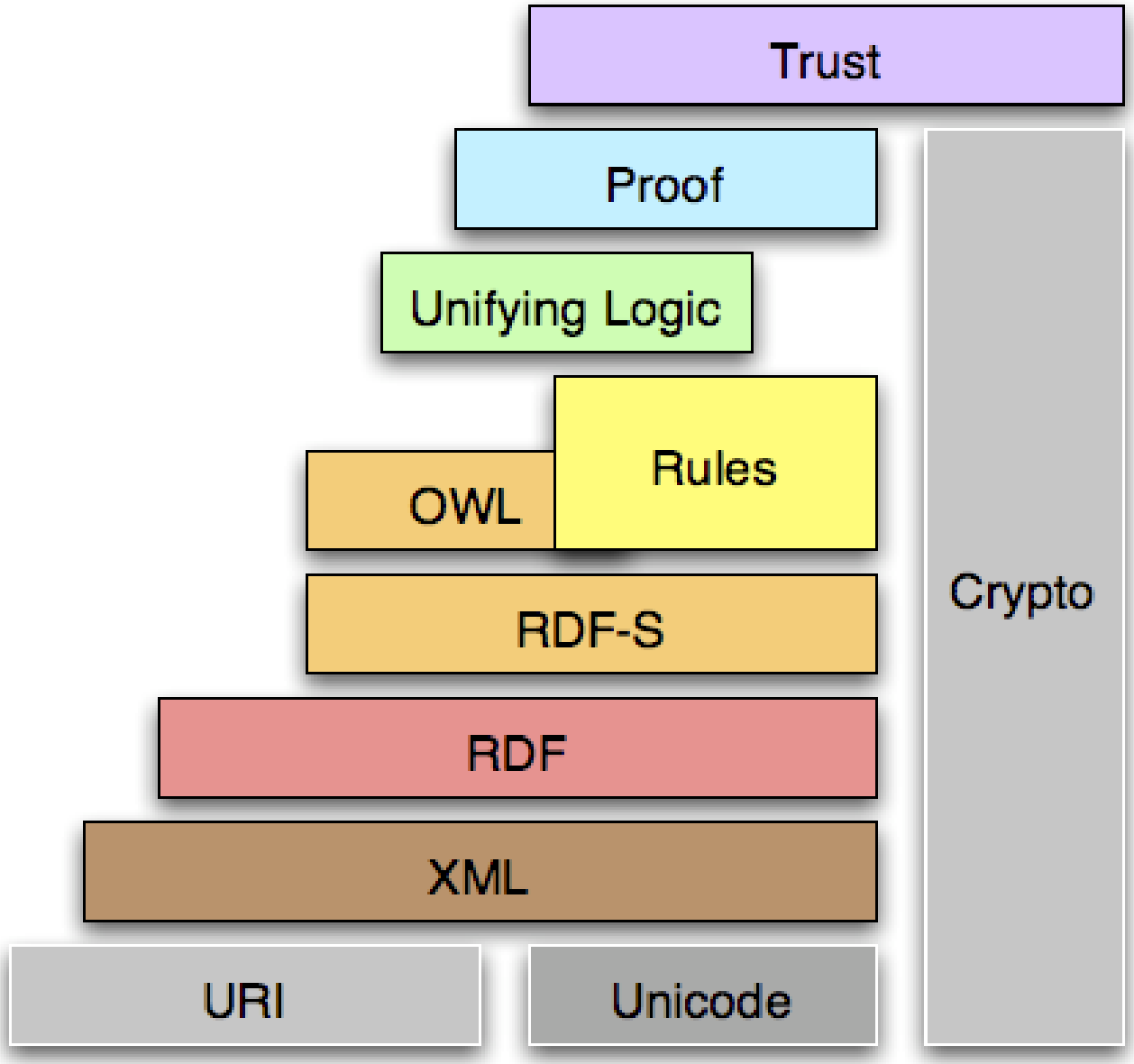
SWRLTab: A Development Environment for working with SWRL Rules In Protégé-OWL

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 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: Has uncle

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

(hasChild >= 1)(?x) → Parent(?x)

Example SWRL Rule with Inferred OWL Restrictions

Parent(?x) → (hasChild >= 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

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)

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

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

$$\text{Person}(?p) \wedge \text{hasCar}(?p,?car) \rightarrow \text{query:select}(?p) \wedge \text{query:count}(?car)$$

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

(hasChild >= 1)(?x) → query:select(?x)

All Built-ins can be used in Queries

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

tbox: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

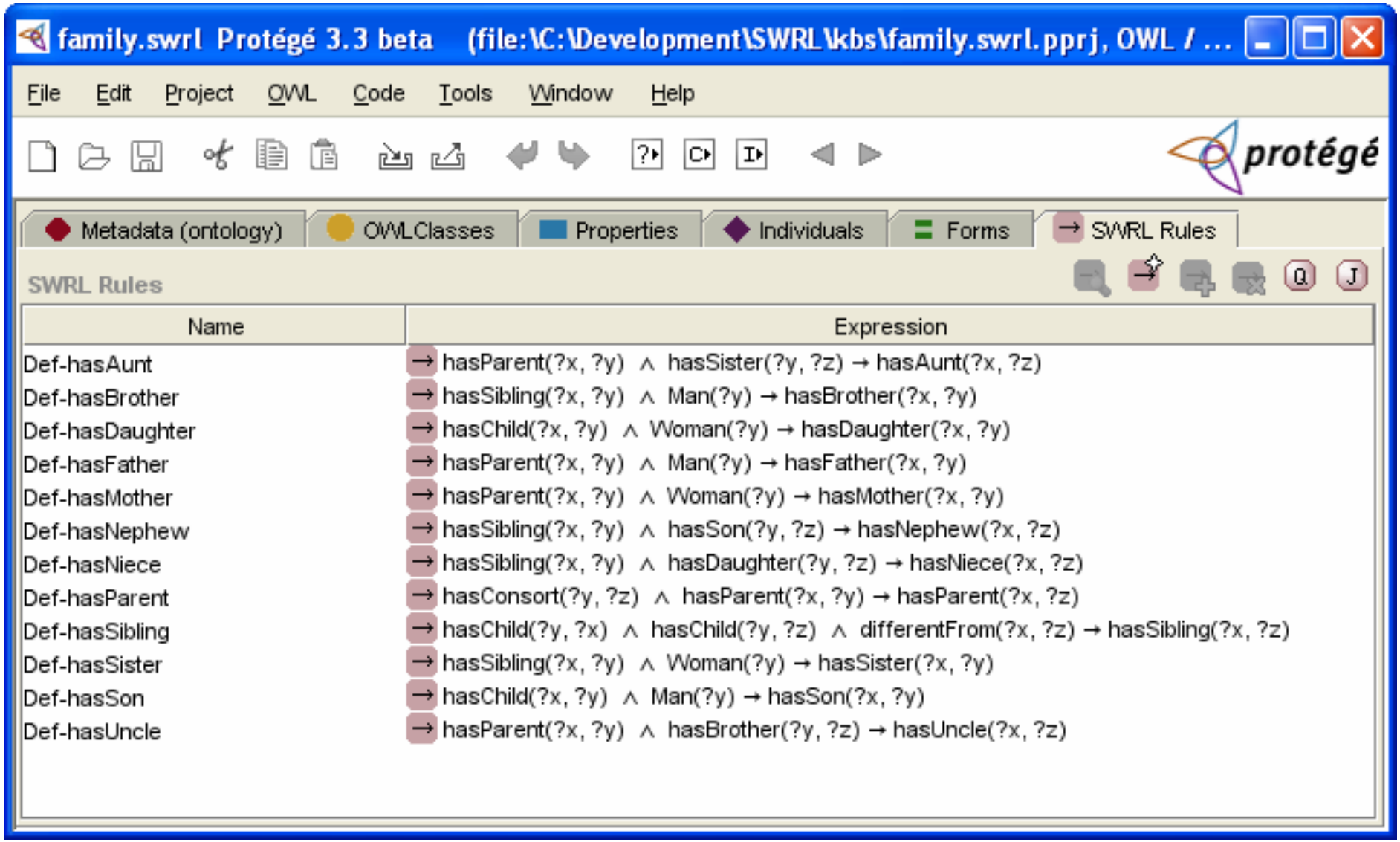
SWRLTab

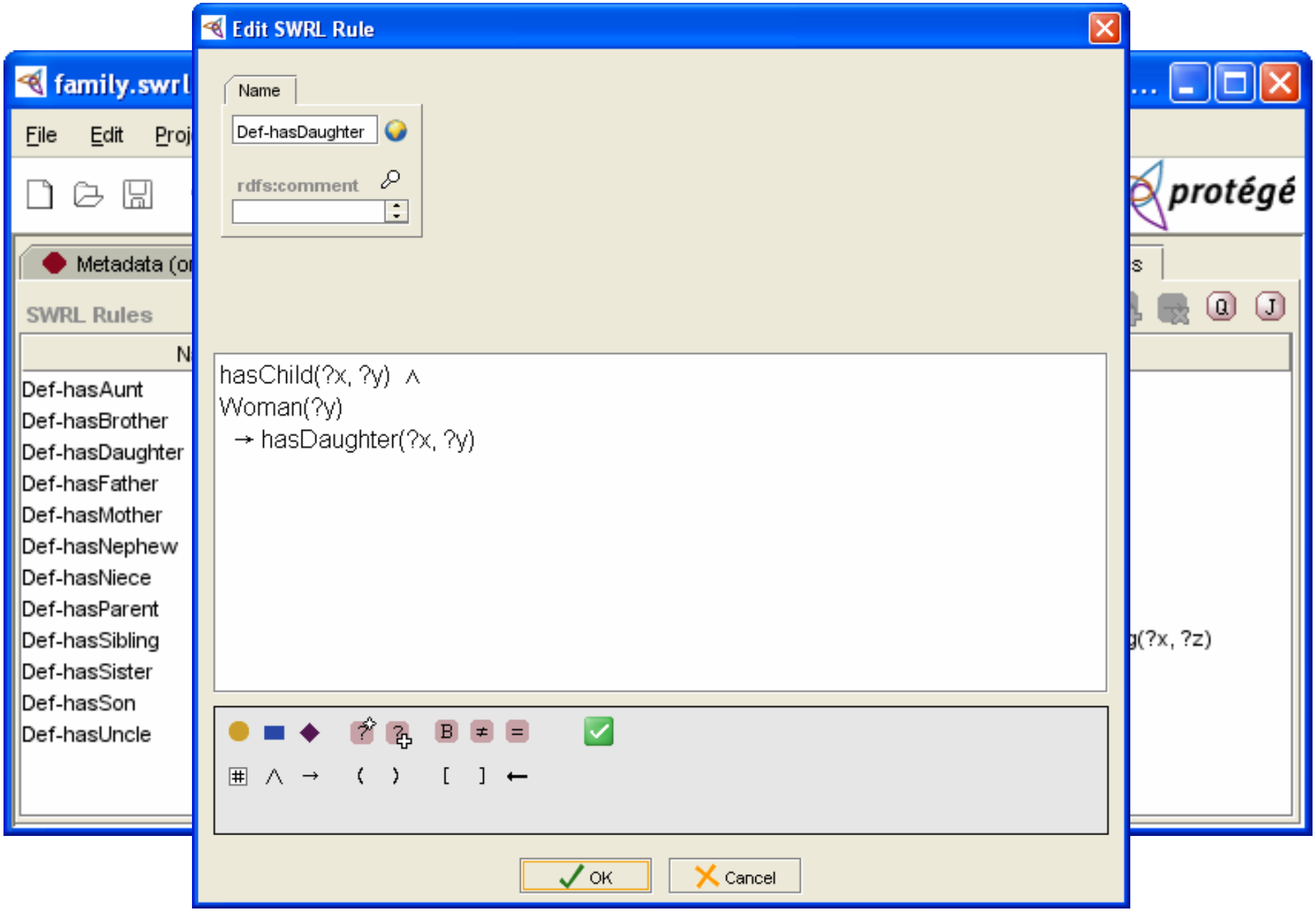
- A Protégé-OWL development environment for working with SWRL rules
- Supports editing and execution of rules
- Extension mechanisms to work with third-party rule engines
- Mechanisms for users to define built-in method libraries
- Supports querying of ontologies

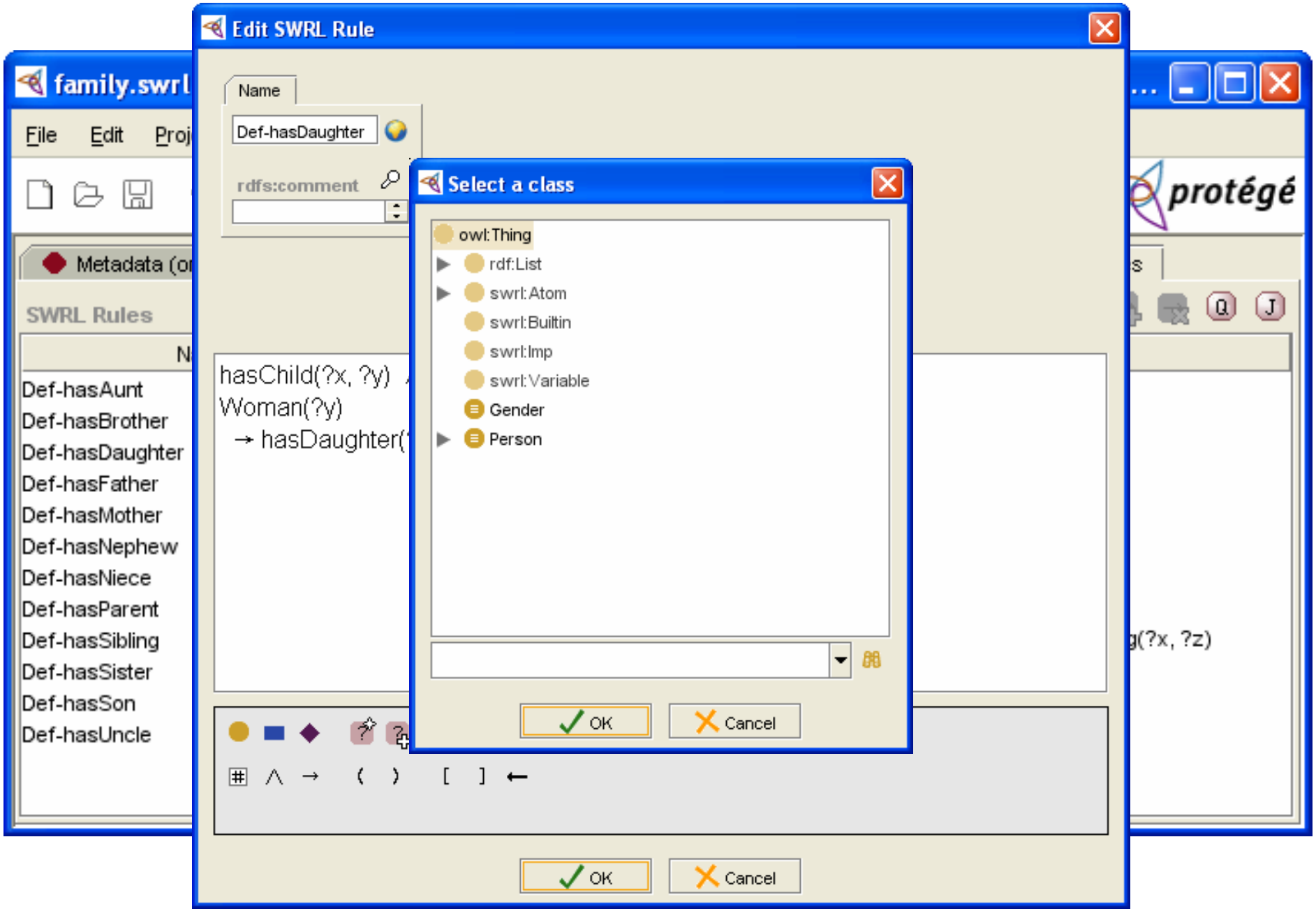
SWRLTab Wiki : http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab

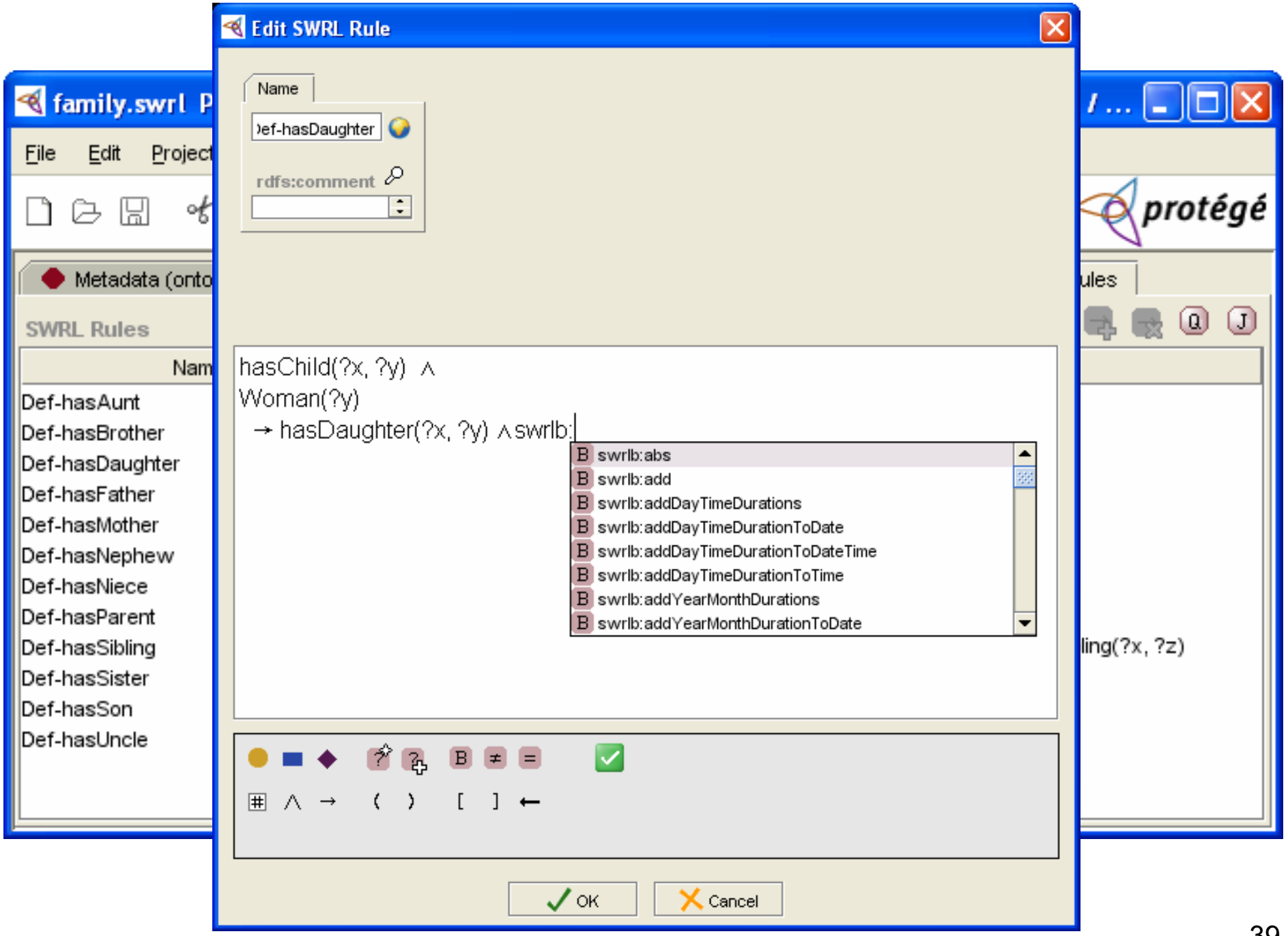
What is the SWRL Editor?

- 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 → 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)


family.swrl
File
Edit
Metadata (or
SWRL Rules
Def-hasAunt
Def-hasBrother
Def-hasDaughter
Def-hasFather
Def-hasMother
Def-hasNephew
Def-hasNiece
Def-hasParent
Def-hasSibling
Def-hasSister
Def-hasSon
Def-hasUncle
protégé
Edit SWRL Rule
Name
Def-hasDaughter
rdfs:comment
hasChild(?x, ?y) ∧
Woman(?y)
→ hasDaughter(?x, ?y)
OK
Cancel

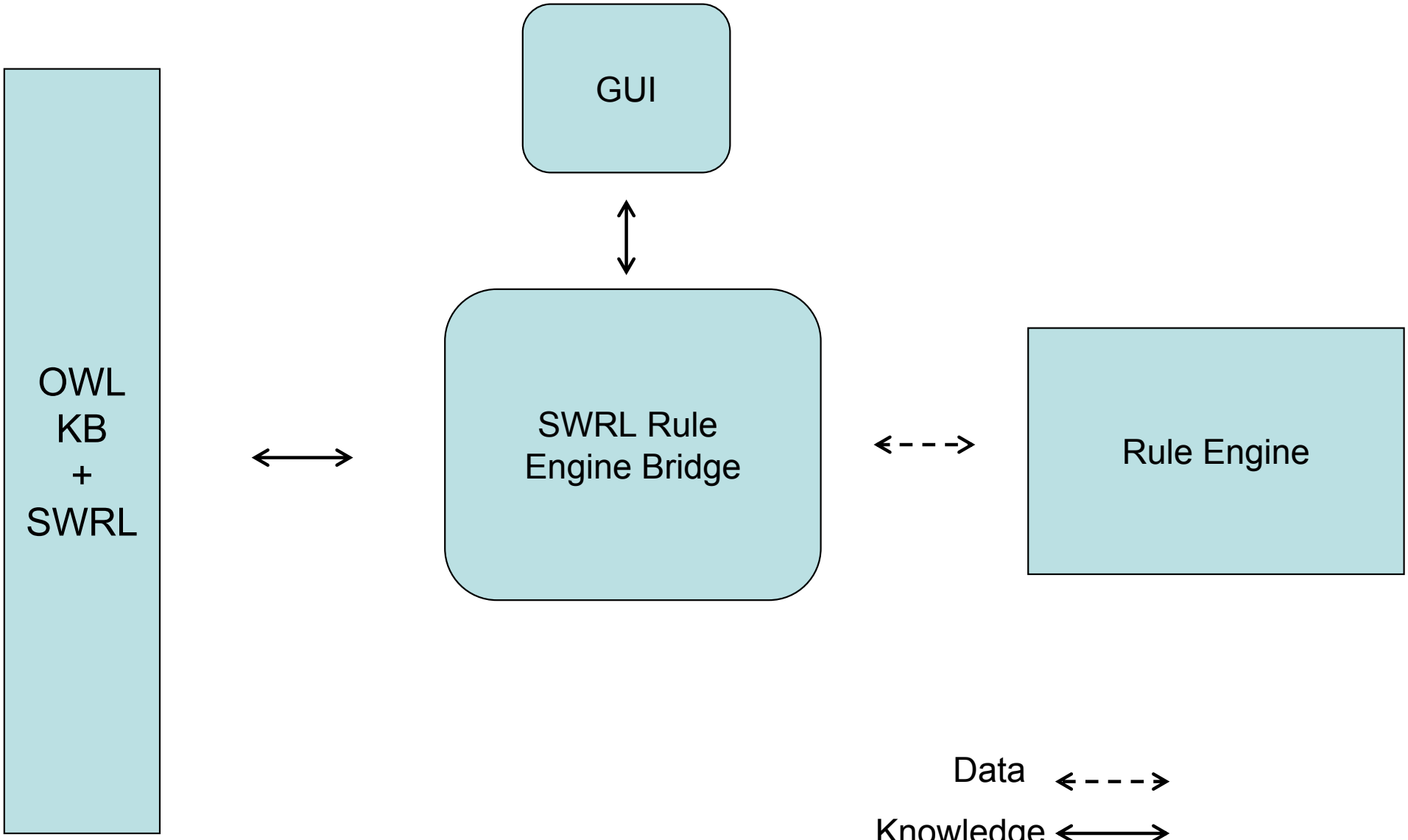

family.swrl
File
Edit
Proj
Metadata (or
SWRL Rules
Def-hasAunt
Def-hasBrother
Def-hasDaughter
Def-hasFather
Def-hasMother
Def-hasNephew
Def-hasNiece
Def-hasParent
Def-hasSibling
Def-hasSister
Def-hasSon
Def-hasUncle
Edit SWRL Rule
Name
Def-hasDaughter
rdfs:comment
hasChild(?x, ?y)
Woman(?y)
→ hasDaughter(
Select a class
owl:Thing
rdf:List
swrl:Atom
swrl:Builtin
swrl:Imp
swrl:Variable
Gender
Person
OK
Cancel
OK
Cancel
protégé
g(?x, ?z)


family.swrl P
File
Edit
Project
Metadata (onto
SWRL Rules
Nam
Def-hasAunt
Def-hasBrother
Def-hasDaughter
Def-hasFather
Def-hasMother
Def-hasNephew
Def-hasNiece
Def-hasParent
Def-hasSibling
Def-hasSister
Def-hasSon
Def-hasUncle
protégé
ules
ling(?x, ?z)
Edit SWRL Rule
Name
)ef-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

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

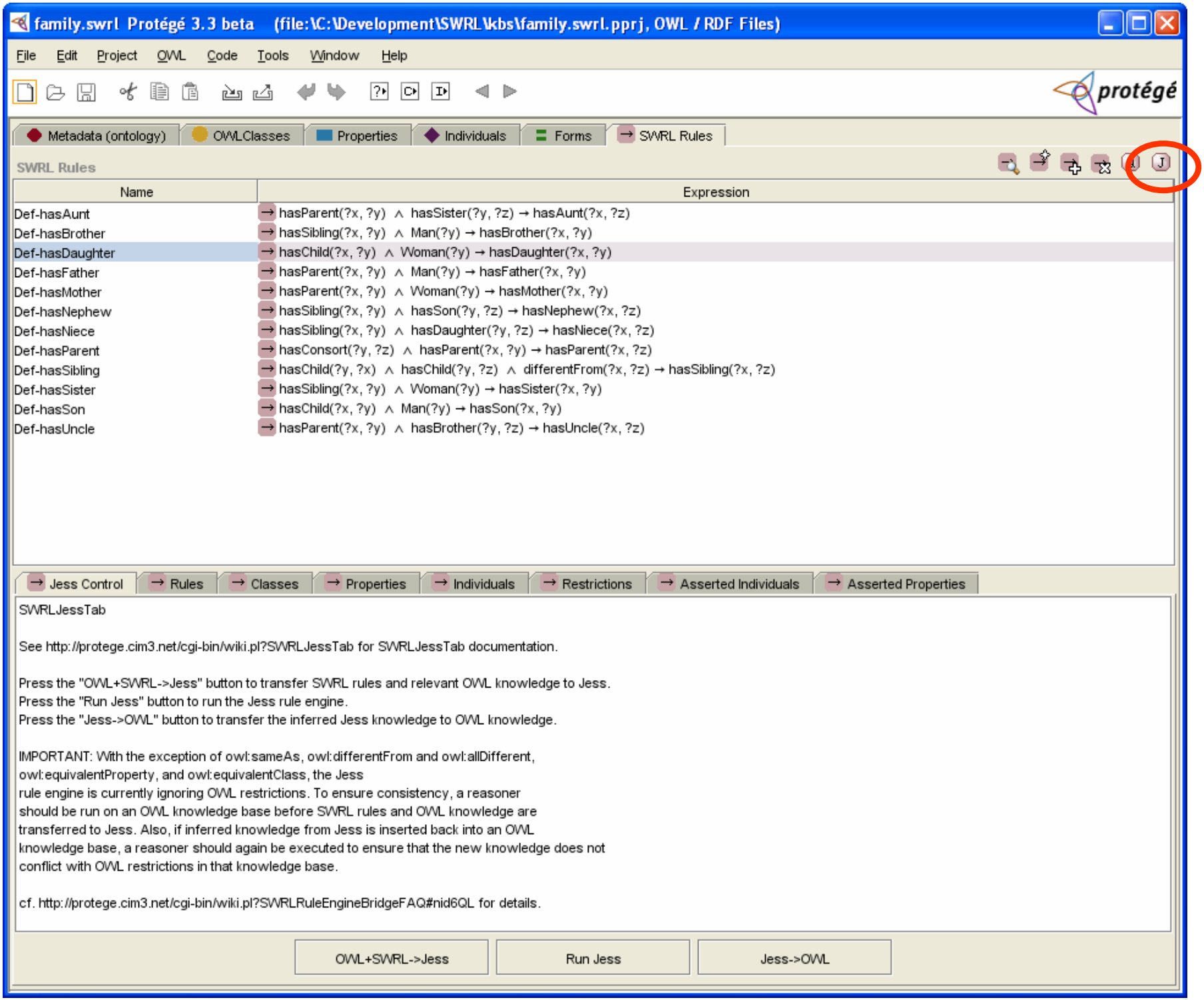


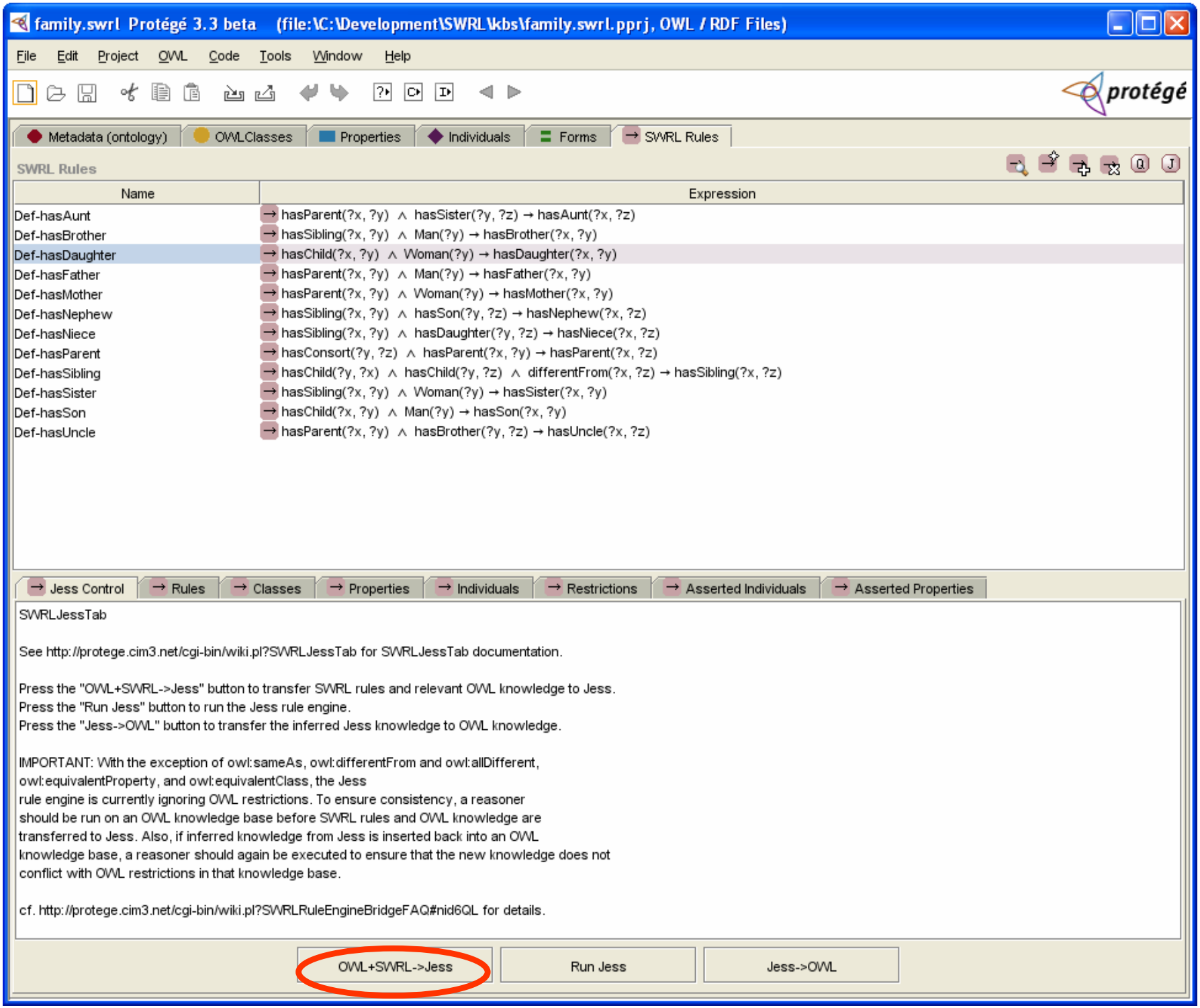
SWRL Rule Engine Bridge

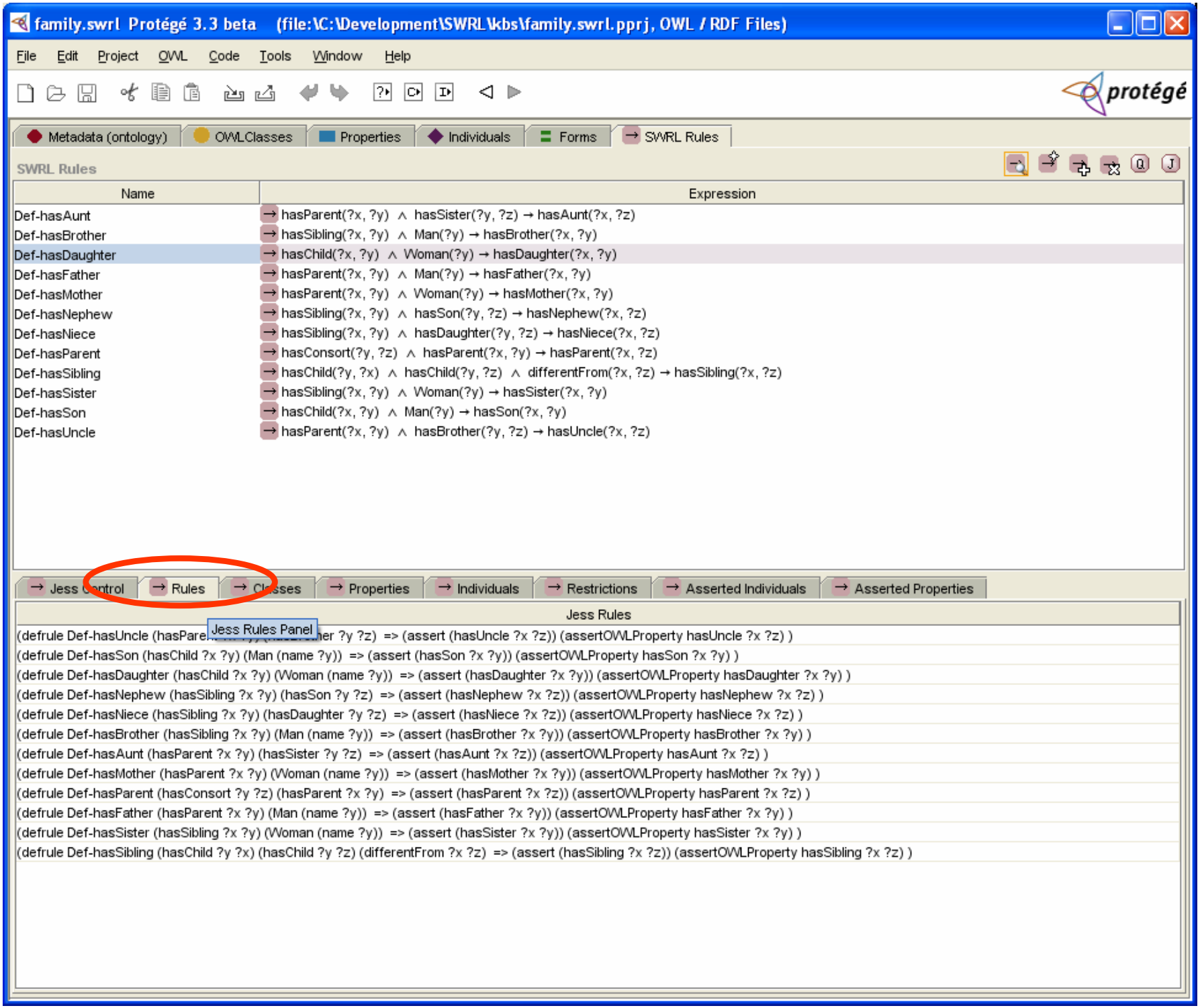
- Given an OWL knowledge base it will extract SWRL rules and relevant OWL knowledge.
- Also provides an API to assert inferred knowledge.
- Knowledge (and rules) are described in non Protégé-OWL API-specific way.
- These can then be mapped to a rule-engine specific rule and knowledge format.
- This mapping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

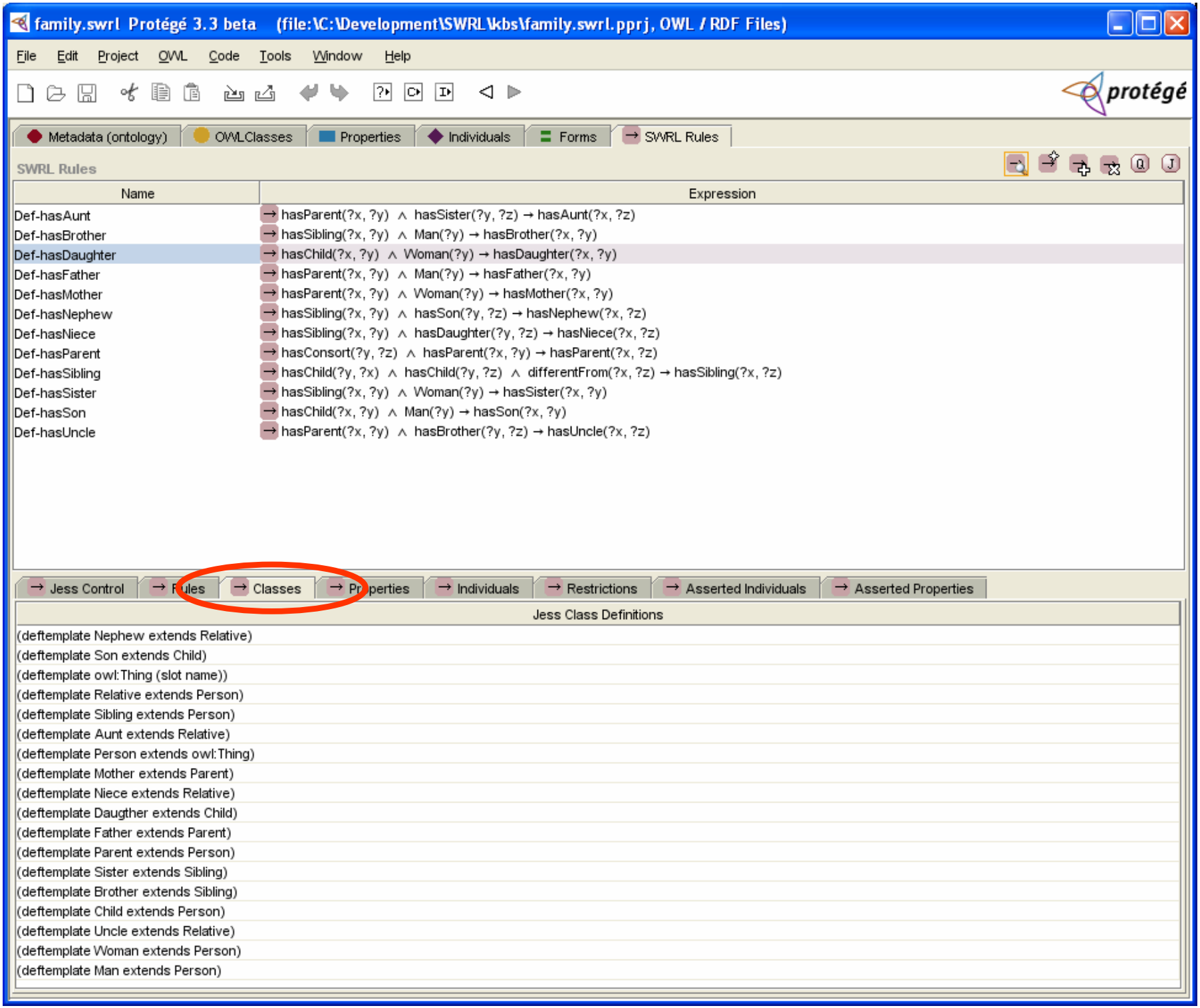
We used the SWRL Bridge to Integrate Jess Rule Engine with Protégé-OWL

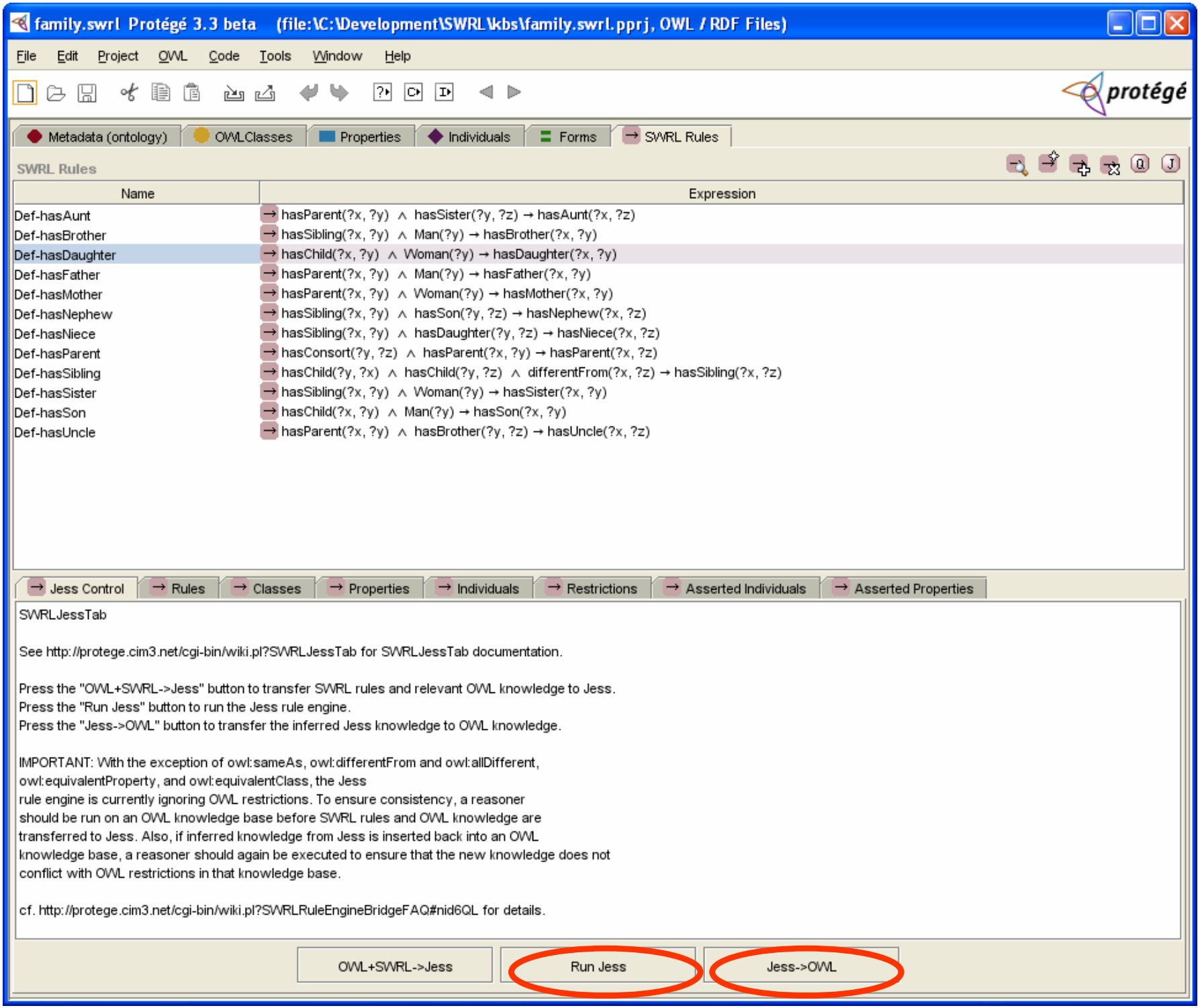
- Jess is a Java-based rule engine.
- Jess system consists of a rule base, fact base, and an execution engine.
- Available free to academic users, for a small fee to non-academic users
- Has been used in Protégé-based tools, e.g., JessTab.


family.swrl Protégé 3.3 beta (file:\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
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:\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
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:\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
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 Properties Individuals Restrictions Asserted Individuals Asserted Properties
Jess Rules Panel
Jess Rules
(defrule Def-hasUncle (hasPare ?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:\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
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 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)


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
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:greaterThanOrEqual(?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

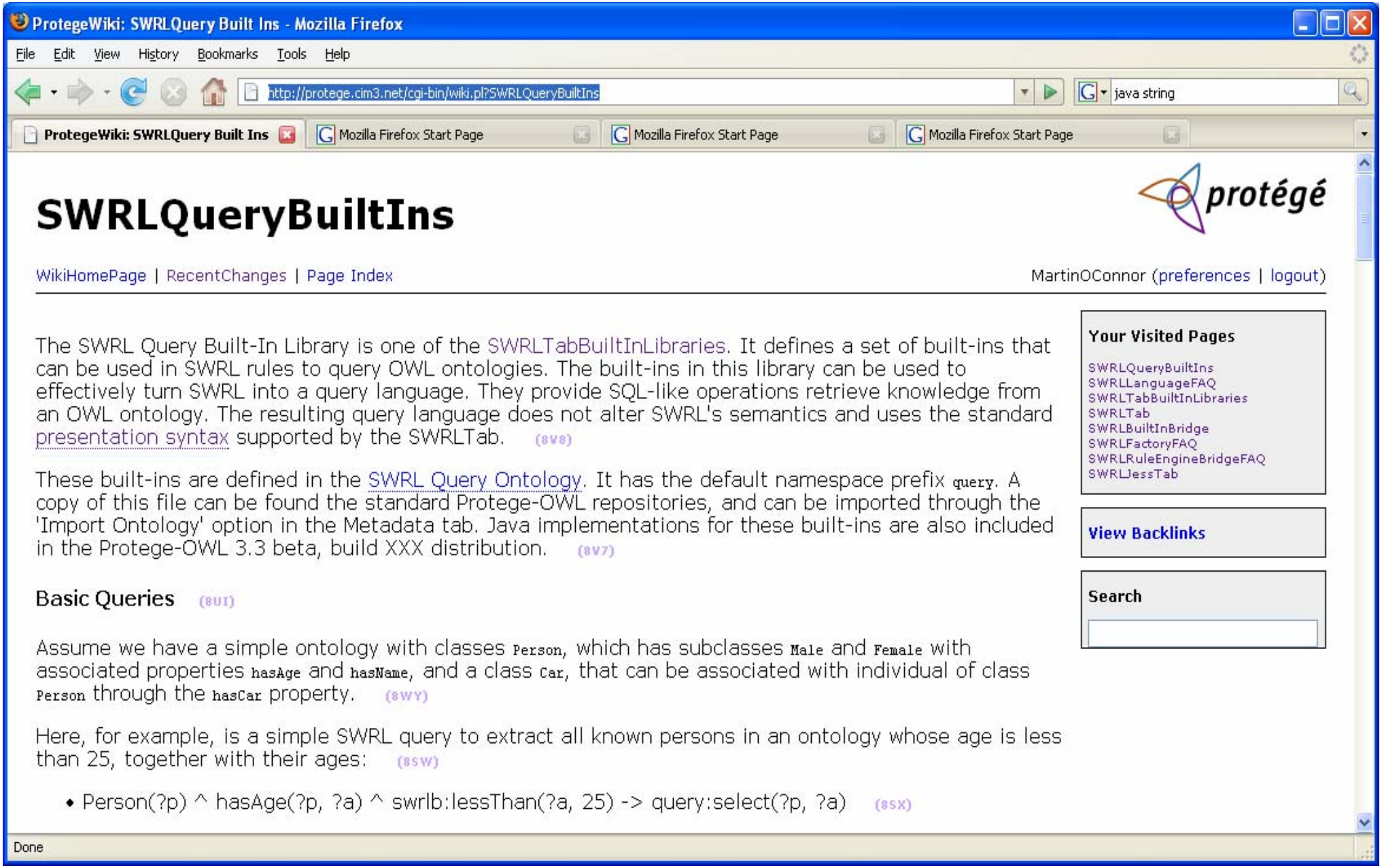
```
package edu.stanford.smi.protegex.owl.swrl.bridge.builtins.swrlb;

import edu.stanford.smi.protegex.owl.swrl.bridge.builtins.*;
import edu.stanford.smi.protegex.owl.swrl.bridge.exceptions.*;

public class SWRLBuiltInLibraryImpl extends SWRLBuiltInLibrary
{
  public SWRLBuiltInMethodsImpl() { …}
  public void reset() {…}

  public boolean stringEqualIgnoreCase(List arguments) throws BuiltInException { ... }
....
} // SWRLBuiltInLibraryImpl
```

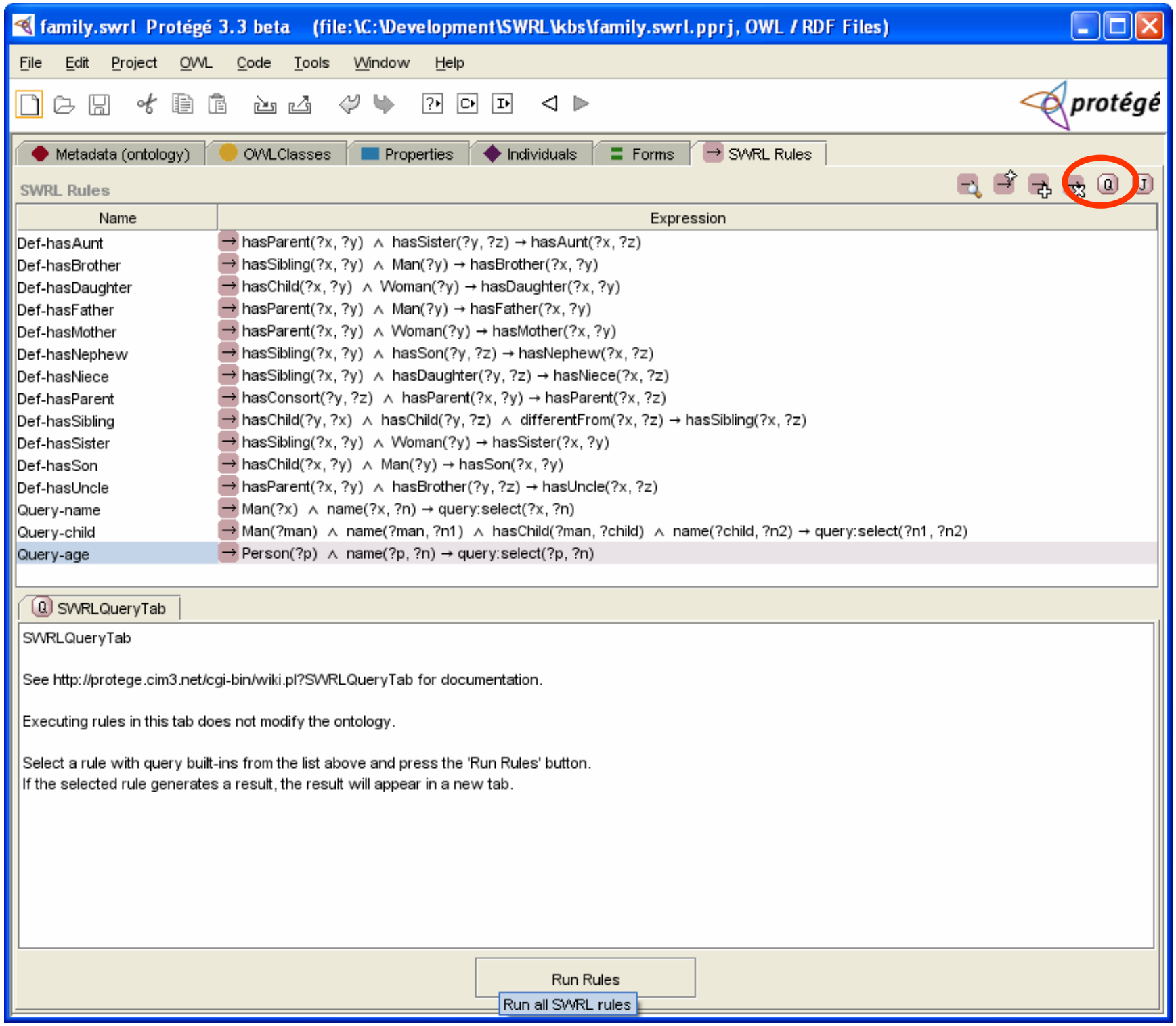
Example Implementation for Built-in swrlb:stringEqualIgnoreCase

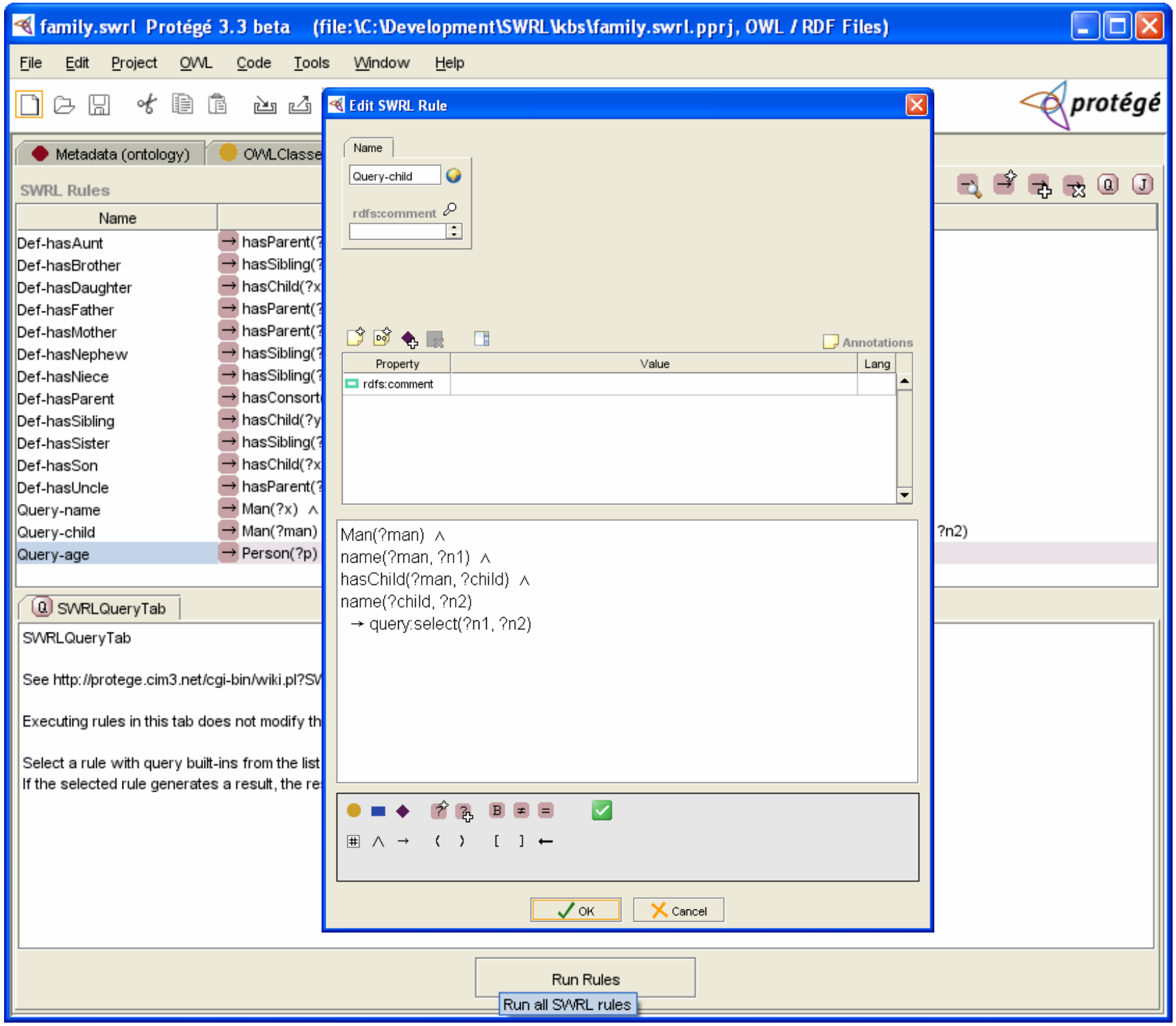
```
public boolean stringEqualIgnoreCase(List<Argument> arguments) throws BuiltInException
{
  SWRLBuiltInUtil.checkNumberOfArgumentsEqualTo(2, arguments.size());

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

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

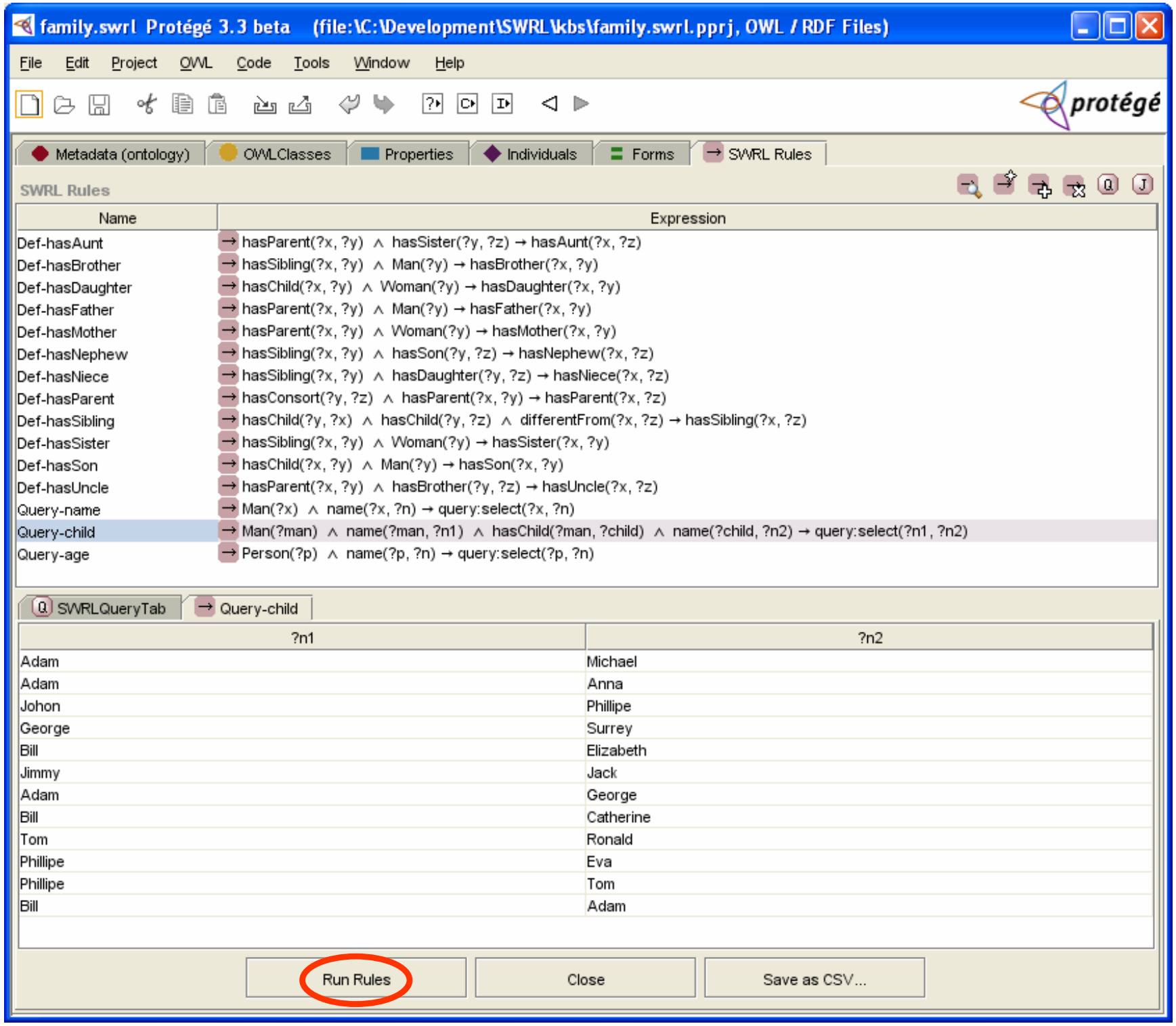
Invocation from Rule Engine

- 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
java string
ProtegeWiki: SWRLQuery Built Ins
Mozilla Firefox Start Page
Mozilla Firefox Start Page
Mozilla Firefox Start Page
SWRLQueryBuiltIns
protégé
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 (8UI)
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: (8SW)
• Person(?p) ^ hasAge(?p, ?a) ^ swrlb:lessThan(?a, 25) -> query:select(?p, ?a) (8SX)
Your Visited Pages
SWRLQueryBuiltIns
SWRLLanguageFAQ
SWRLTabBuiltInLibraries
SWRLTab
SWRLBuiltInBridge
SWRLFactoryFAQ
SWRLRuleEngineBridgeFAQ
SWRLJessTab
View Backlinks
Search
Done


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
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


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)
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
Query-name
Query-child
Query-age
SWRLQueryTab
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)
OK
Cancel
Run Rules
Run all SWRL rules


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
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
Query-child
?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

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