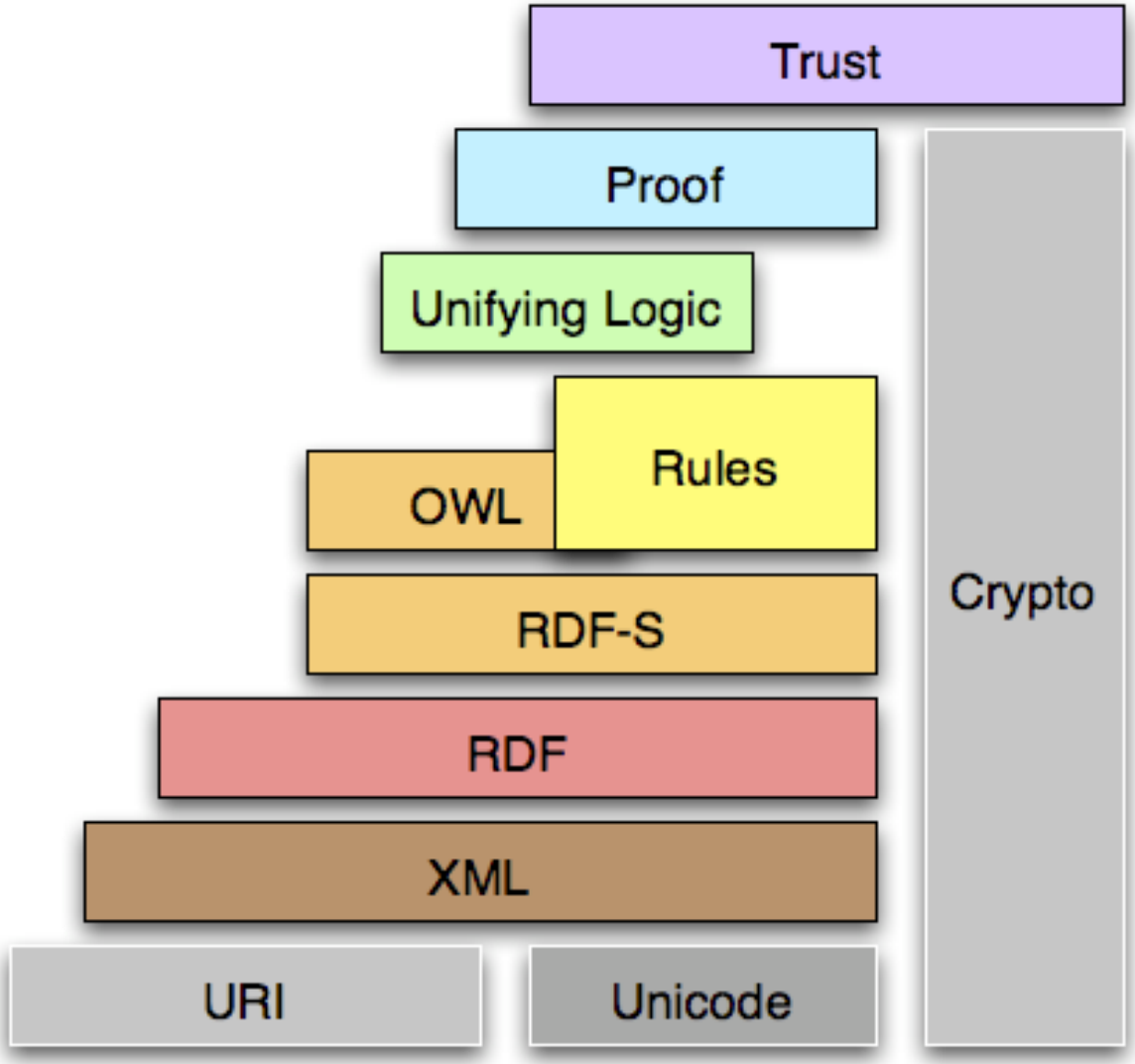
The Semantic Web Rule Language

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 langauge

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

$$\mathrm{Man}(?m) \rightarrow \mathrm{Person}(?m)$$

Possible in OWL - some rules are OWL syntactic sugar

Example SWRL Rule: property value assignment

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

Example SWRL Rule: property value assignment

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

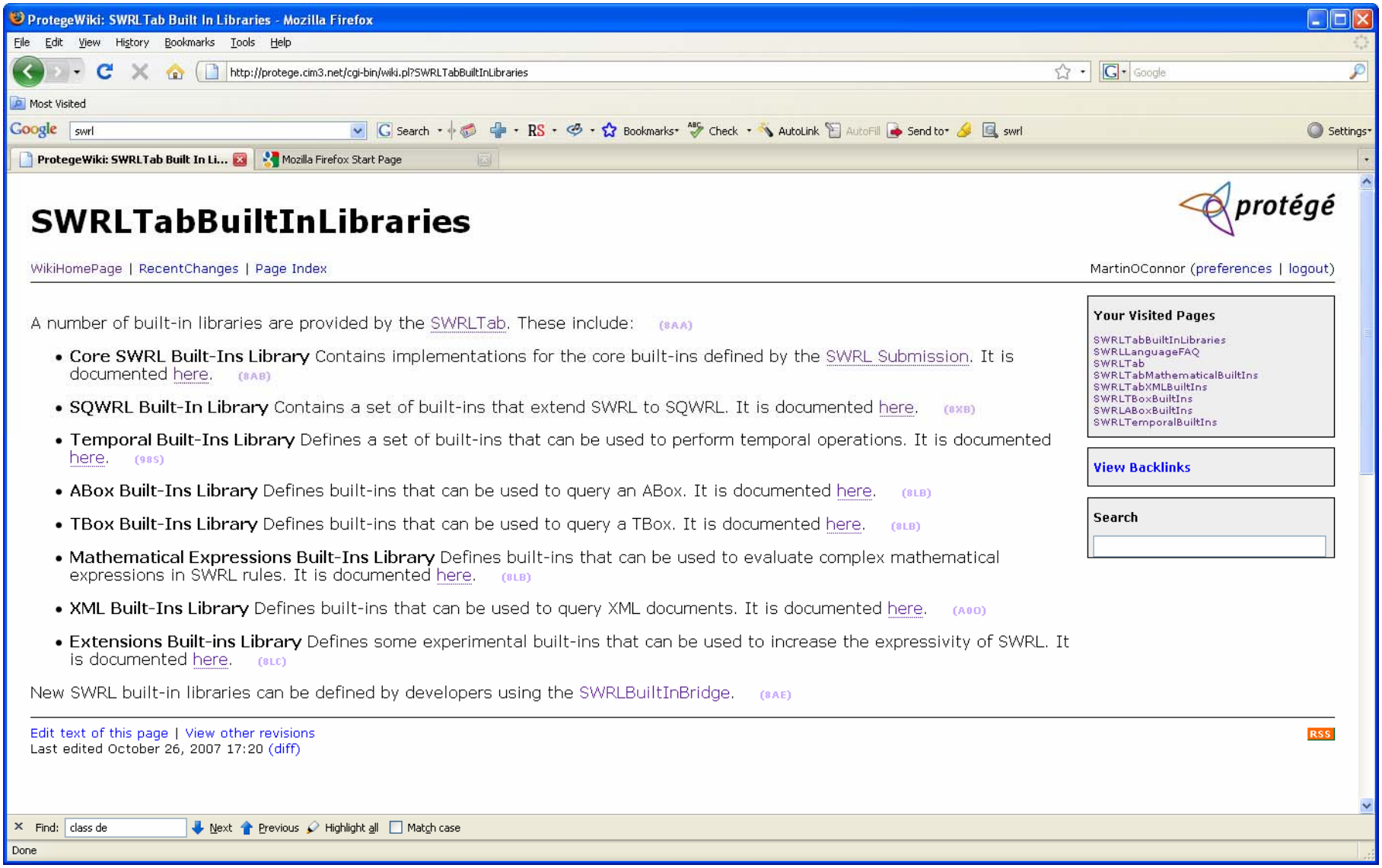
Person(?p) ^ hasSalaryInPounds(?p, ?pounds) ^
swrlb:multiply(2.0, ?pounds, ?dollars) ->
hasSalaryInDollars(?p, ?dollars)

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



Example SWRL Rule with OWL Class Expressions

(hasChild >= 1)(?x) → 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) → (hasChild >= 1)(?x)

Arbitrary OWL class expressions are allowed

Expression syntax may very, 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

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

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)

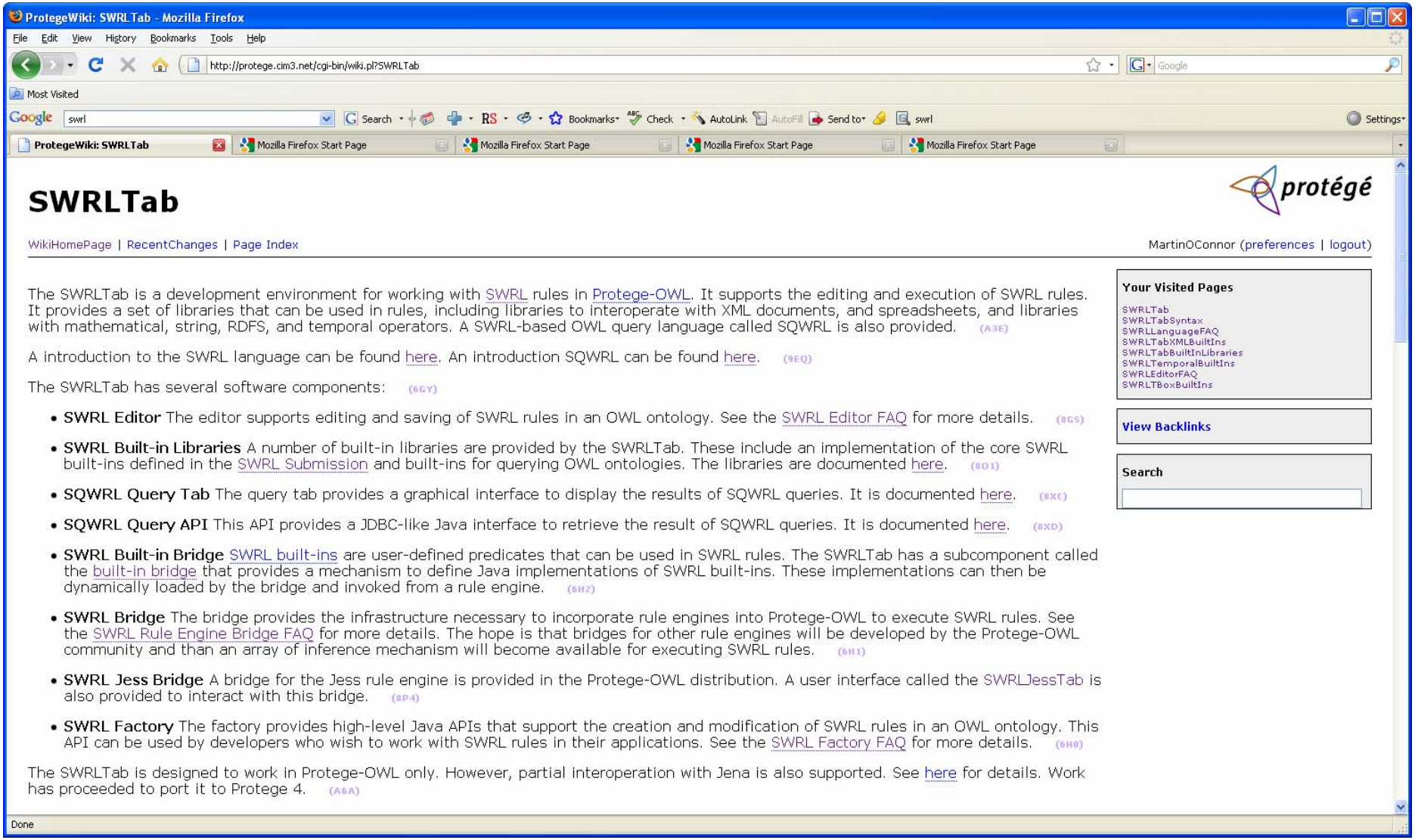
→ SingleAuthorPublication(?p)

Closure - though best expressed in OWL in this case

SWRLTab

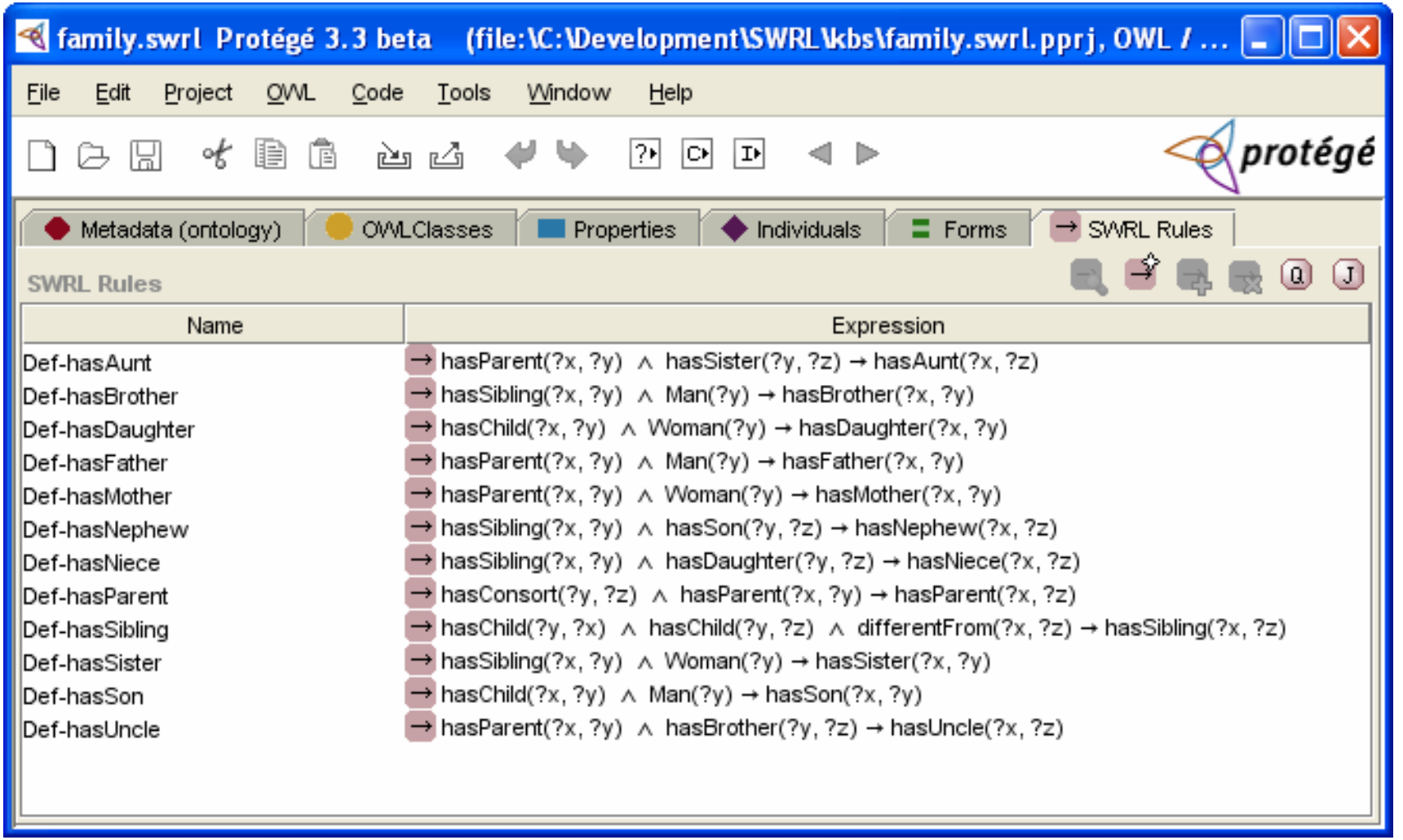
- A Protégé-OWL development environment for working with SWRL rules
- Supports editing and execution of rules
- Extension mechanisms to work with third-party rule engines
- Mechanisms for users to define built-in method libraries
- Supports querying of ontologies

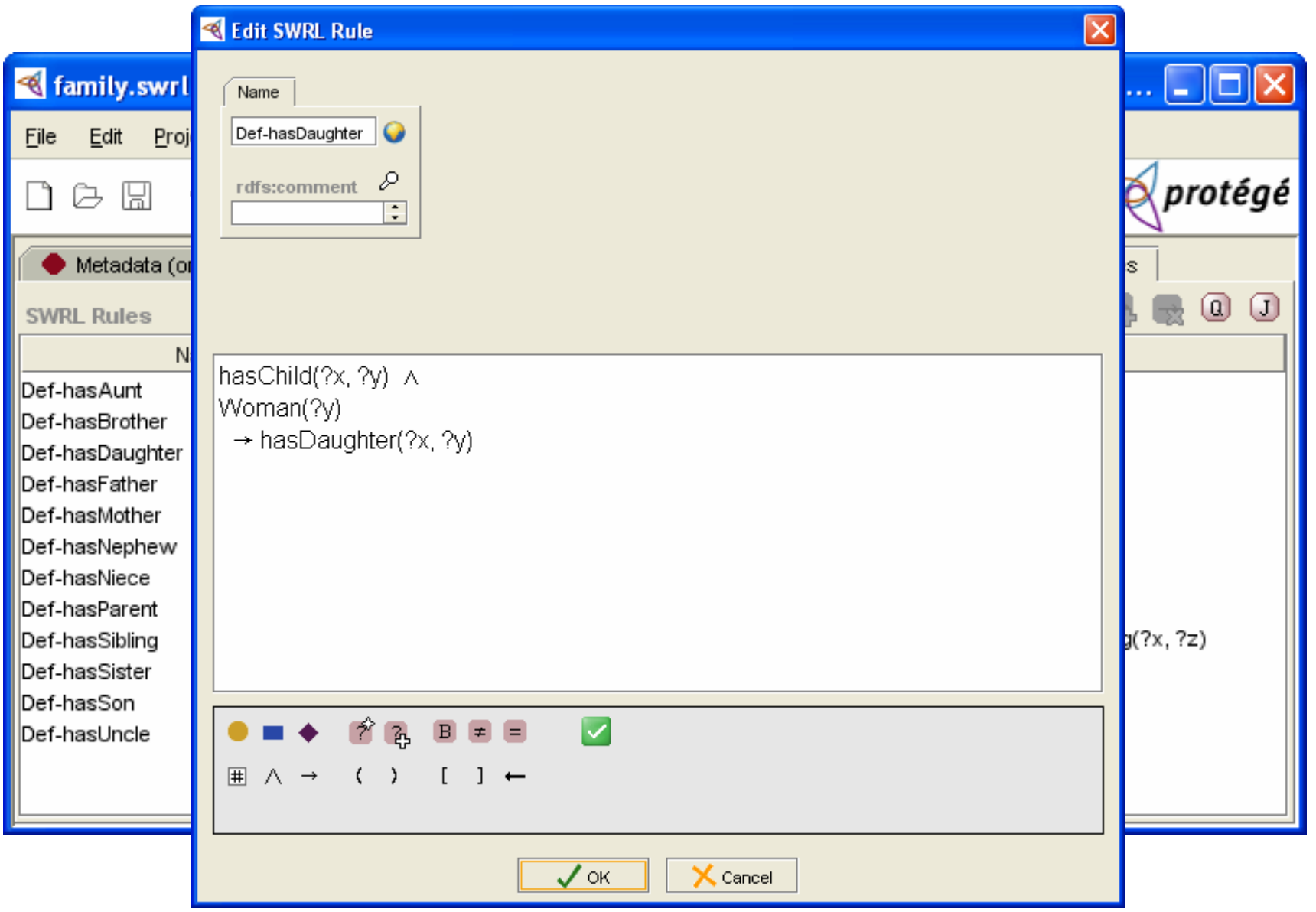
SWRLTab Wiki : http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab

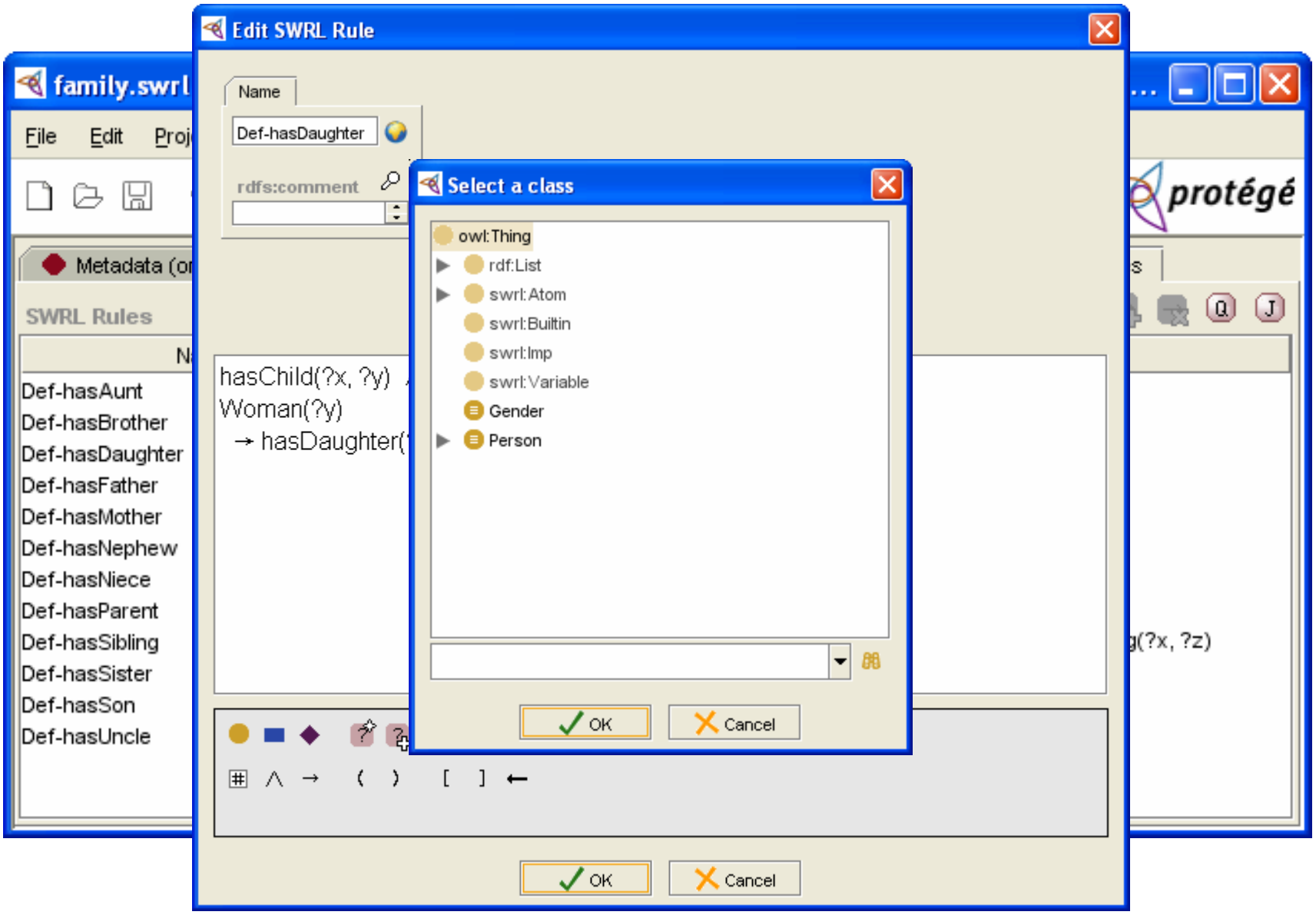


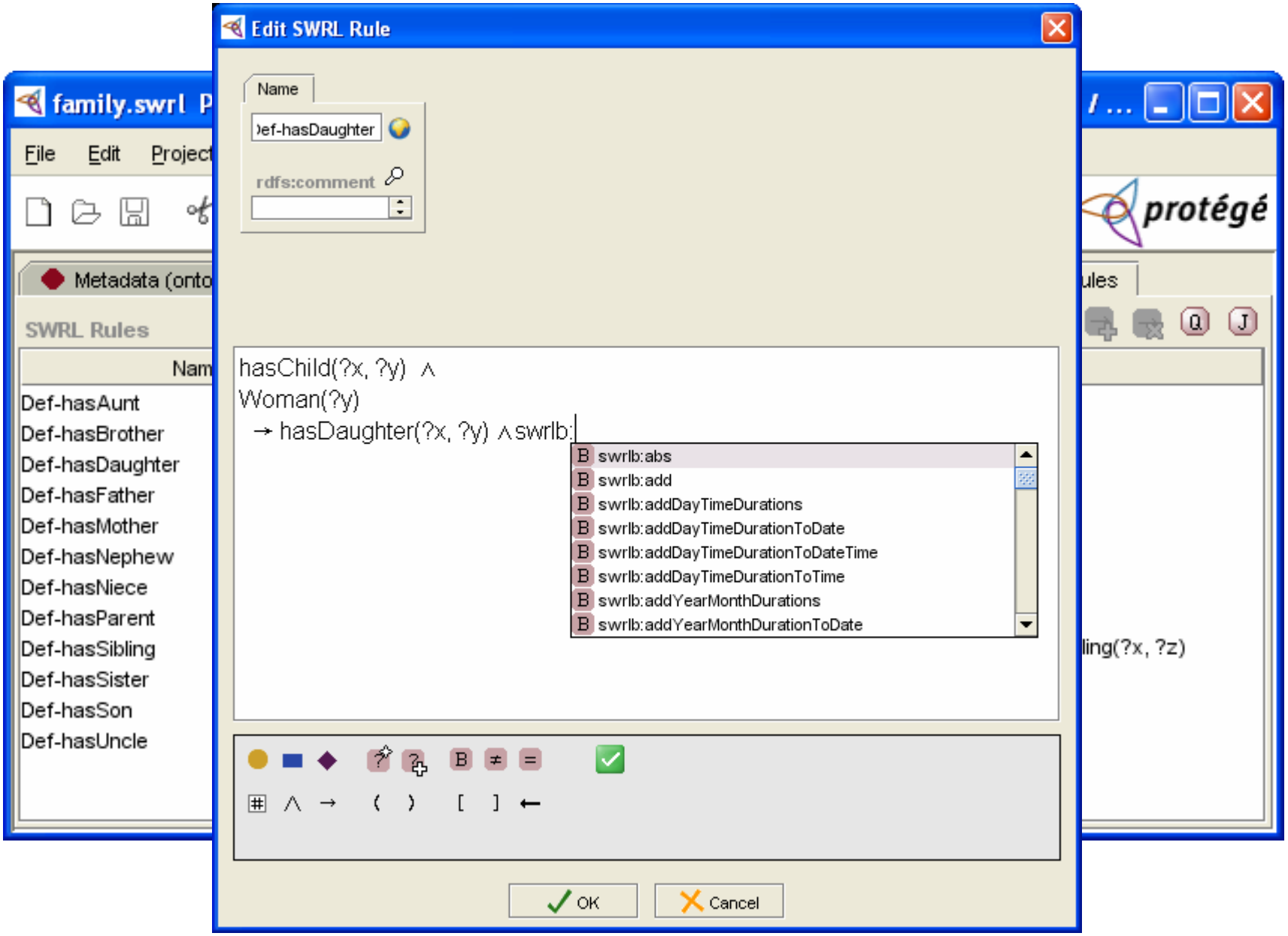
What is the SWRL Editor?

- 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
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(?x, ?y)
OK
Cancel
protégé
g(?x, ?z)

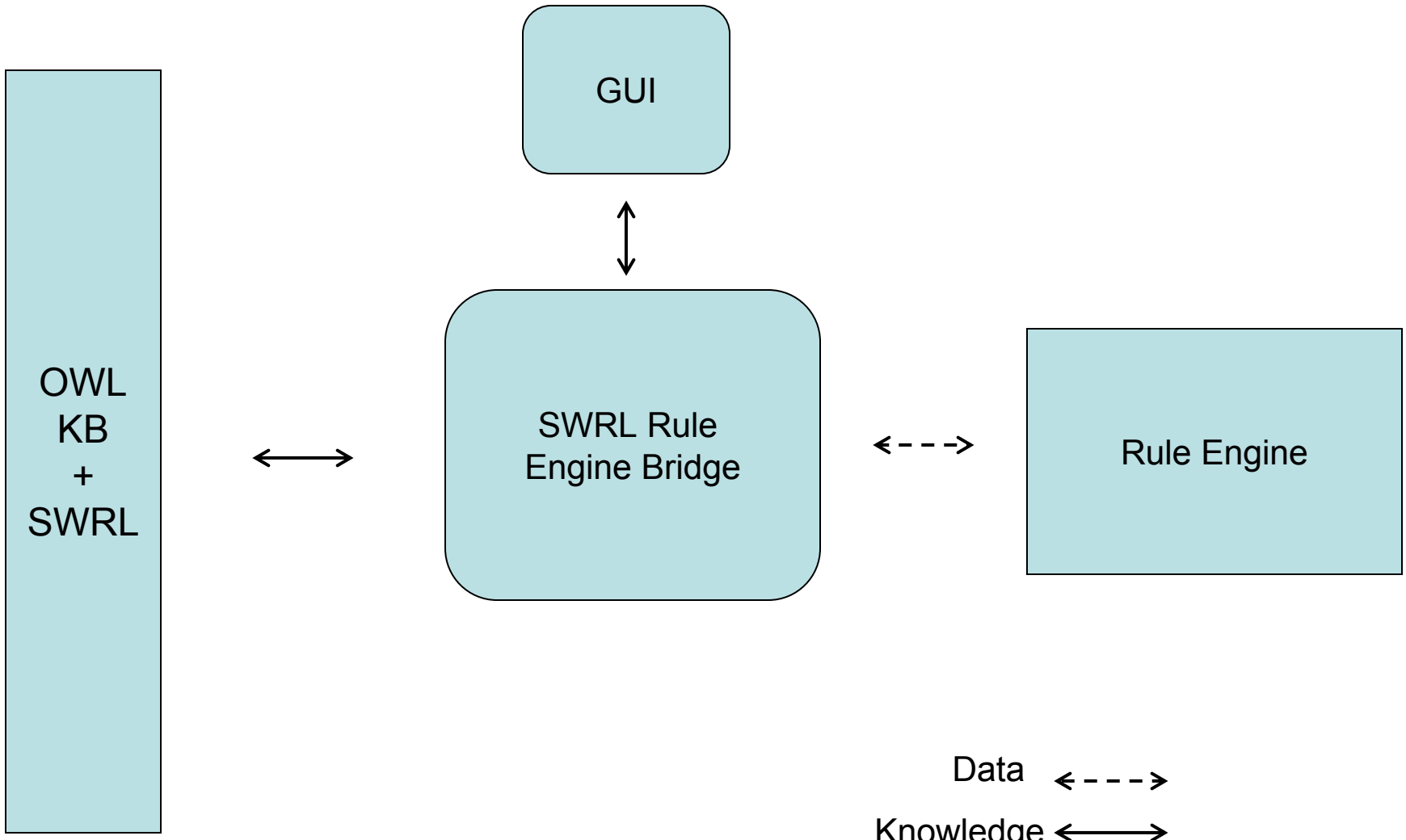

family.swrl
File
Edit
Proj
Metadata (or
SWRL Rules
Def-hasAunt
Def-hasBrother
Def-hasDaughter
Def-hasFather
Def-hasMother
Def-hasNephew
Def-hasNiece
Def-hasParent
Def-hasSibling
Def-hasSister
Def-hasSon
Def-hasUncle
Edit SWRL Rule
Name
Def-hasDaughter
rdfs:comment
hasChild(?x, ?y)
Woman(?y)
→ hasDaughter(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

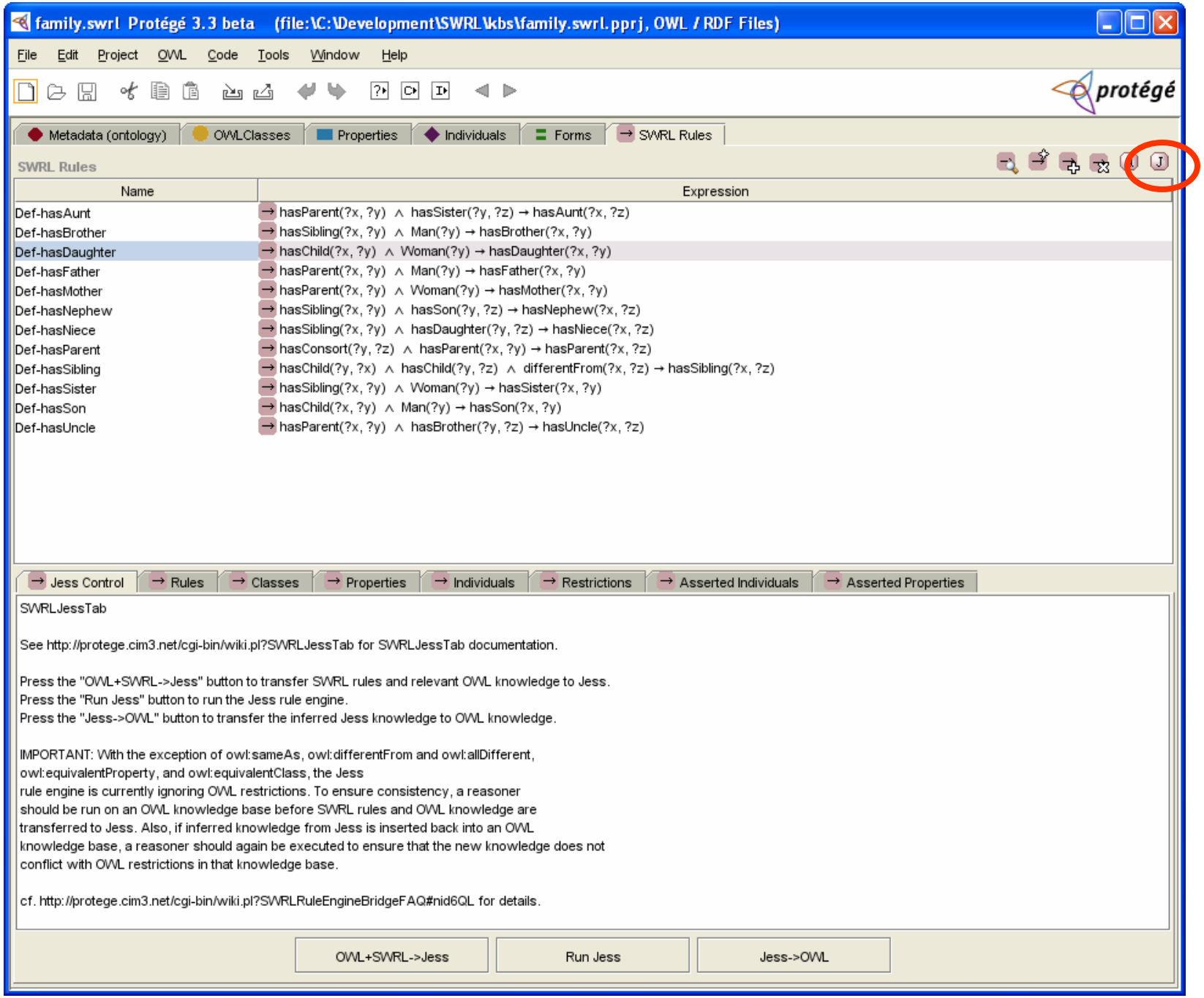


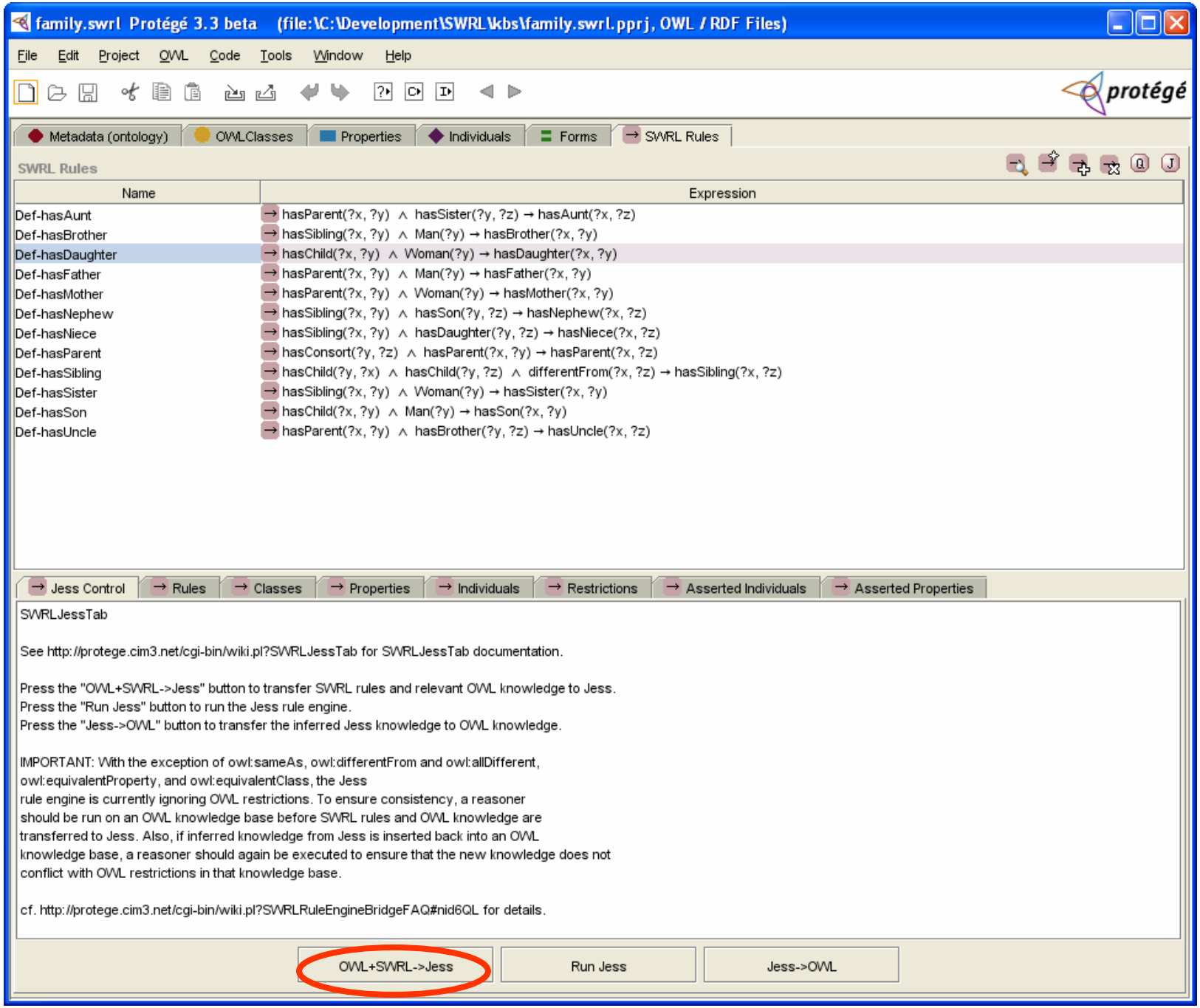
SWRL Rule Engine Bridge

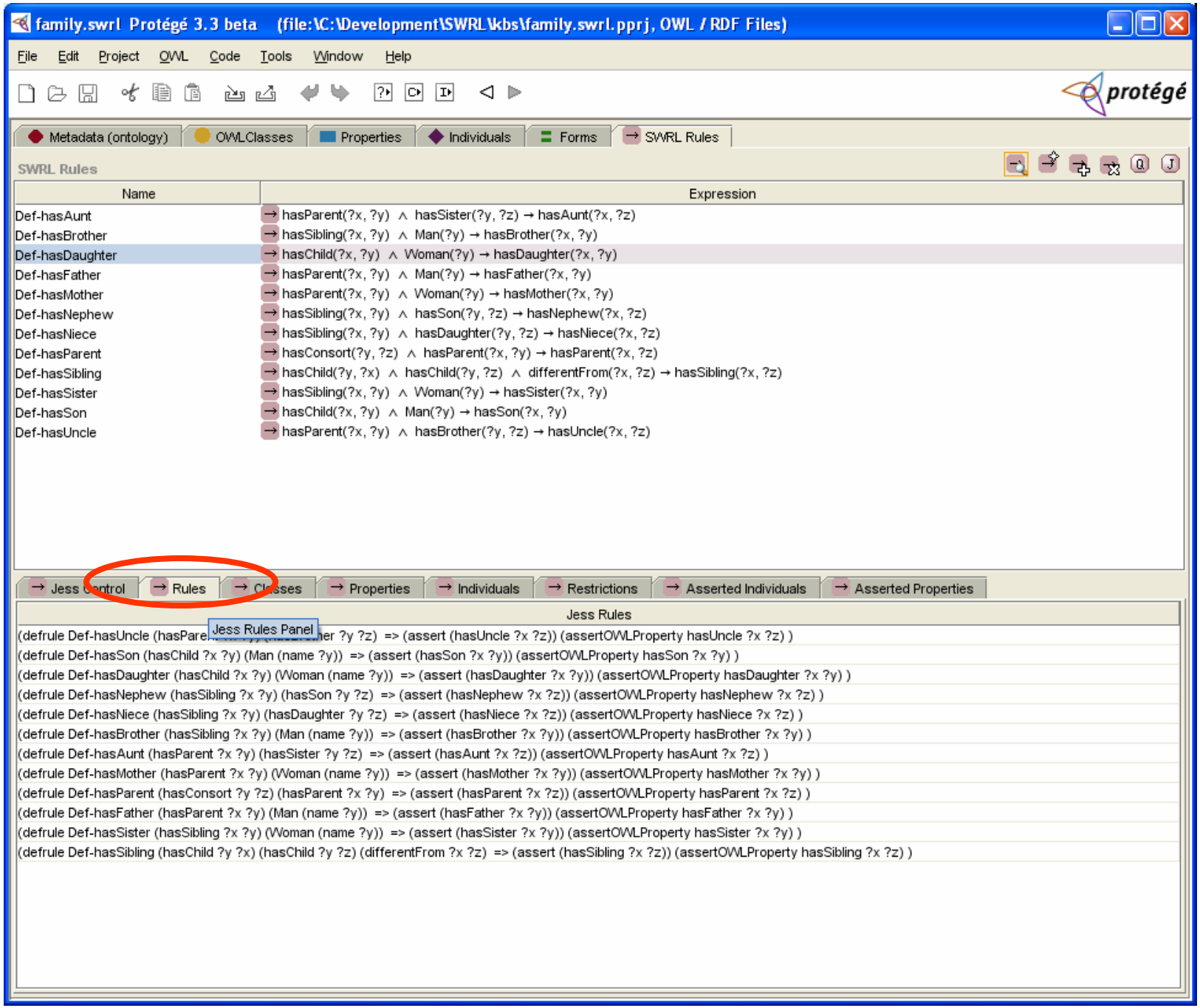
- Given an OWL knowledge base it will extract SWRL rules and relevant OWL knowledge.
- Also provides an API to assert inferred knowledge.
- Knowledge (and rules) are described in non Protégé-OWL API-specific way.
- These can then be mapped to a rule-engine specific rule and knowledge format.
- This mapping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

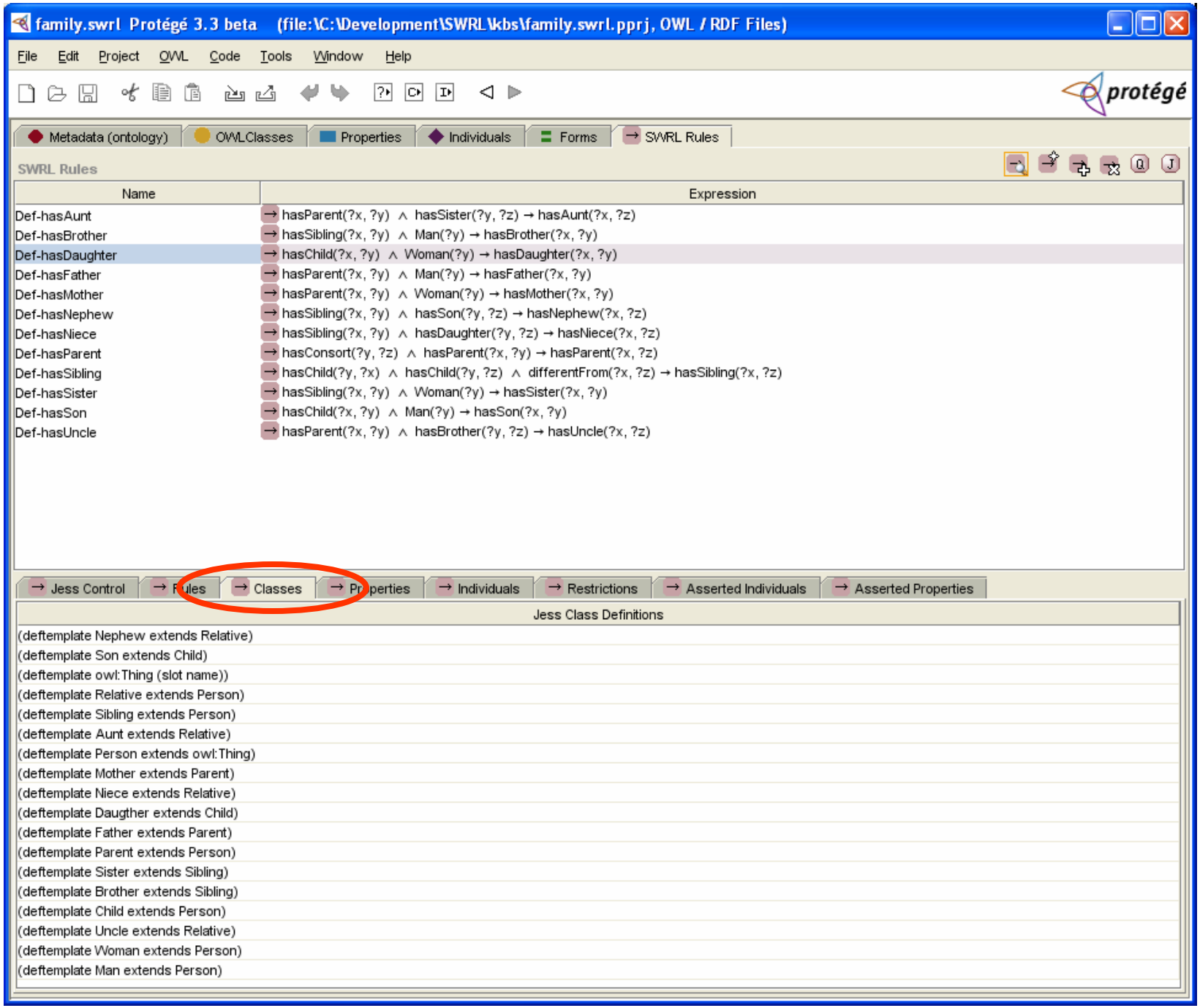
We used the SWRL Bridge to Integrate Jess Rule Engine with Protégé-OWL

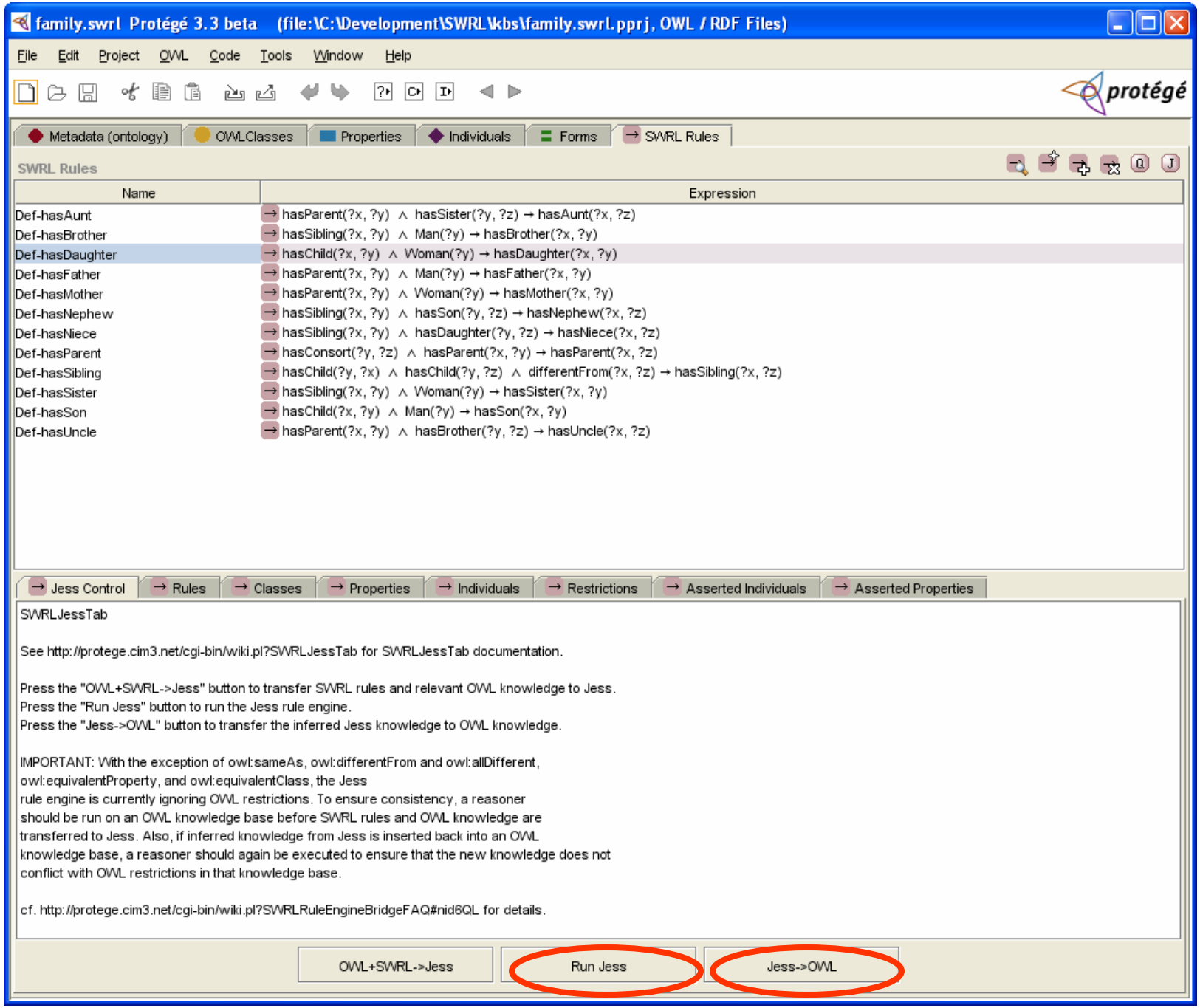
- Jess is a Java-based rule engine.
- Jess system consists of a rule base, fact base, and an execution engine.
- Available free to academic users, for a small fee to non-academic users
- Has been used in Protégé-based tools, e.g., JessTab.


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 Classes Properties Individuals Restrictions Asserted Individuals Asserted Properties
Jess Rules Panel
Jess Rules
(defrule Def-hasUncle (hasPare
her ?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.
- 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

(hasChild >= 1)(?x) → 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)

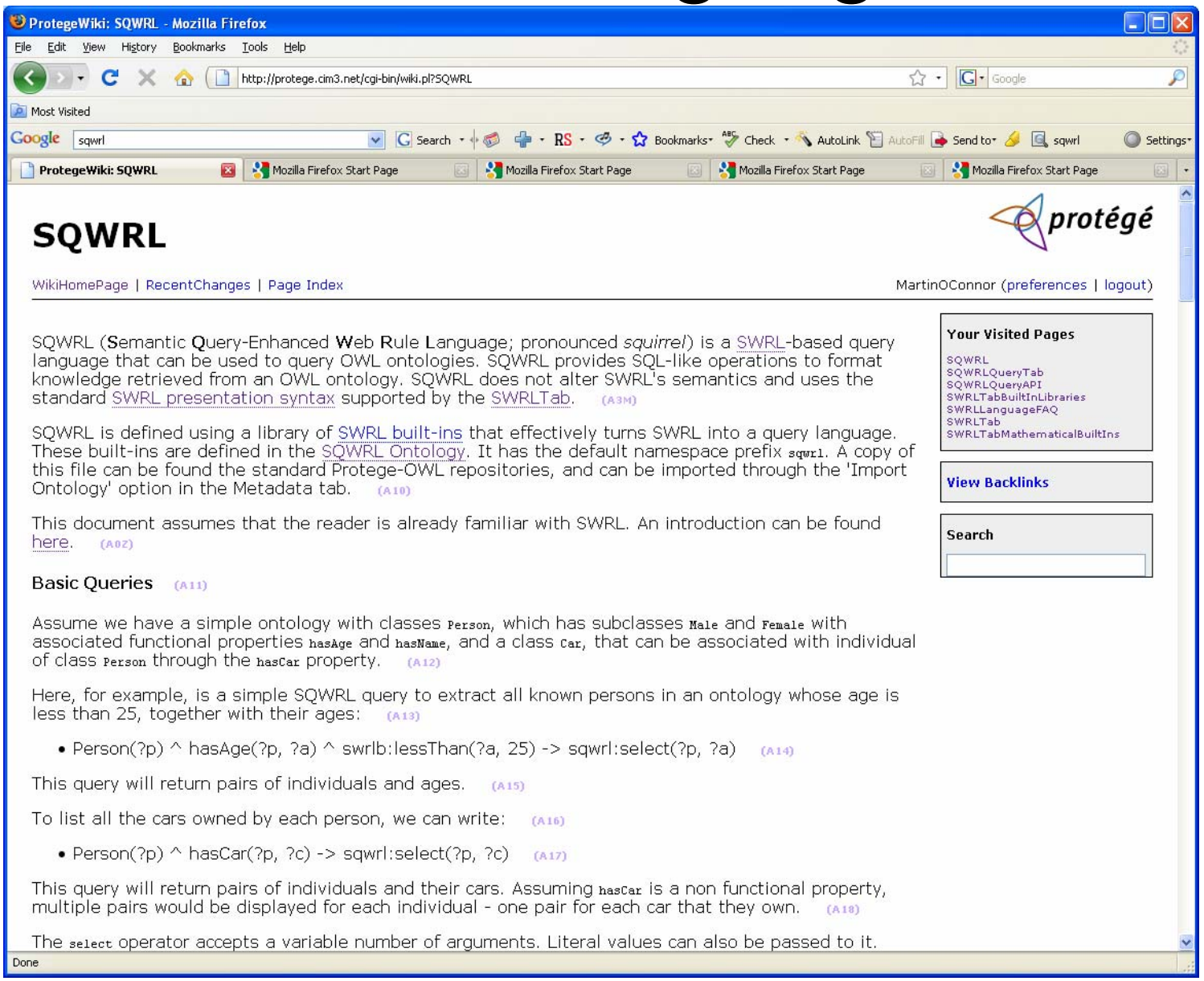
tbox:isSubPropertyOf(?supProperty, hasName)

-> sqwrl:select(?subProperty)

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

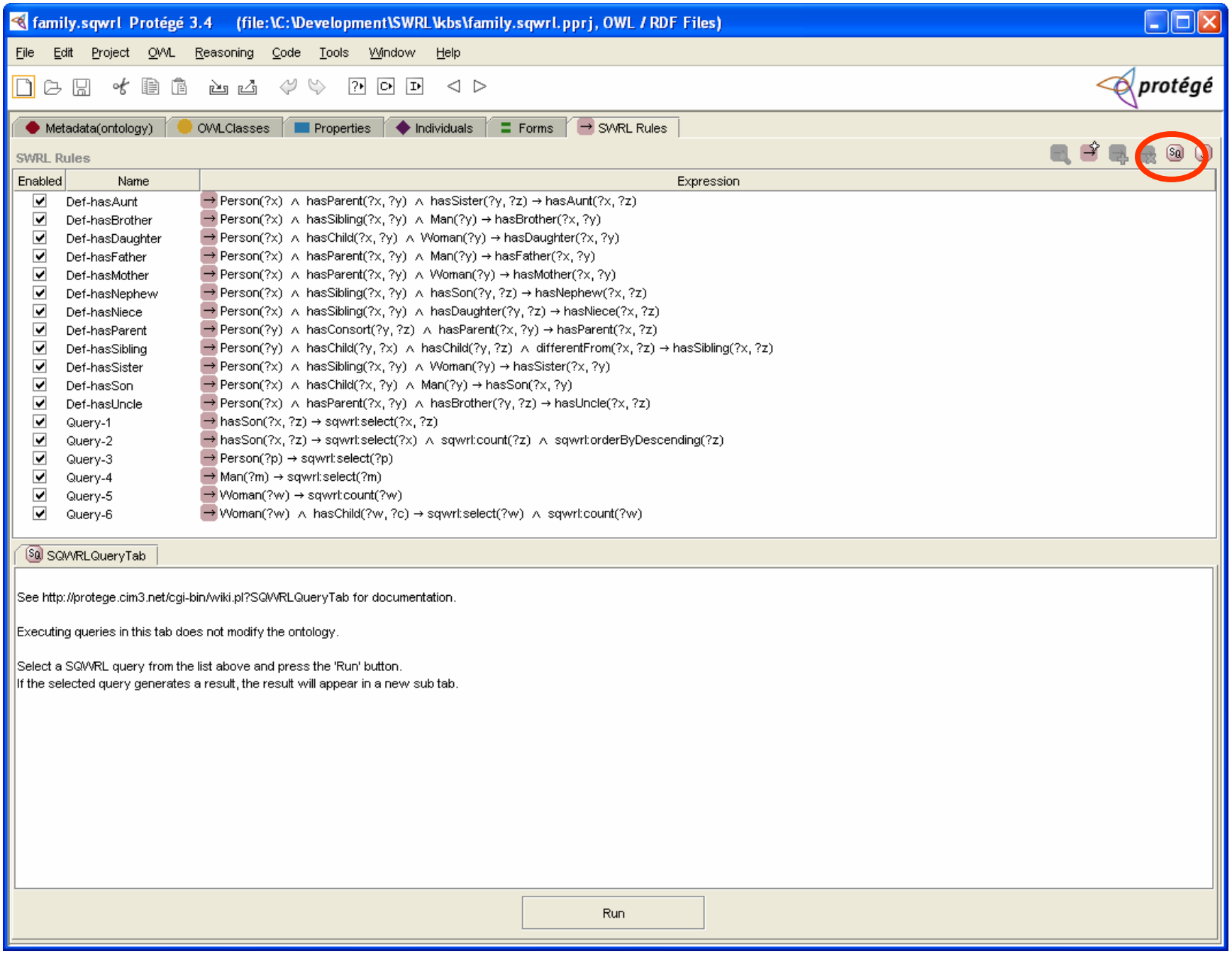
Important: these built-ins should be used in queries only – inference with them would definitely not be OWL DL

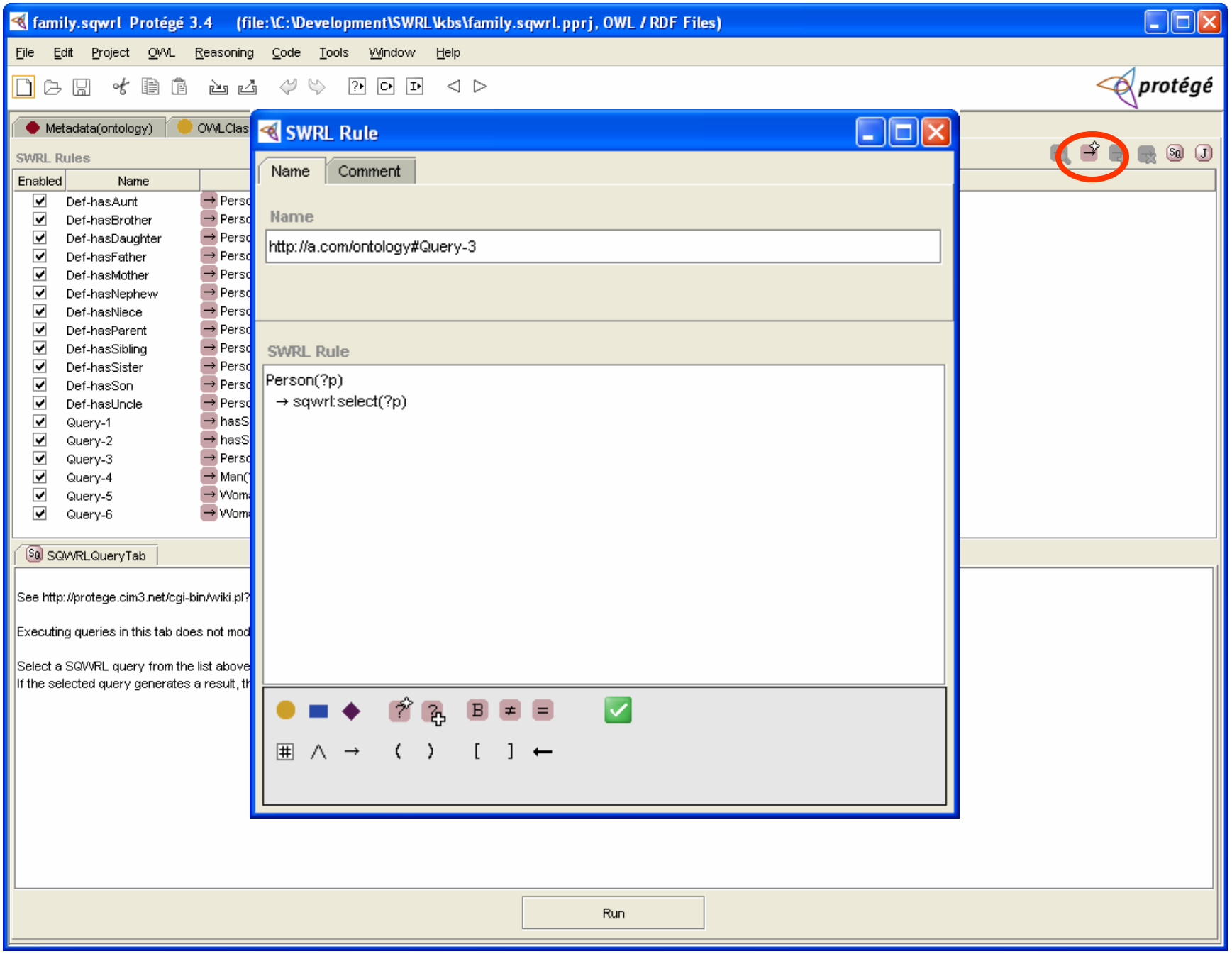
SQWRL Language FAQ

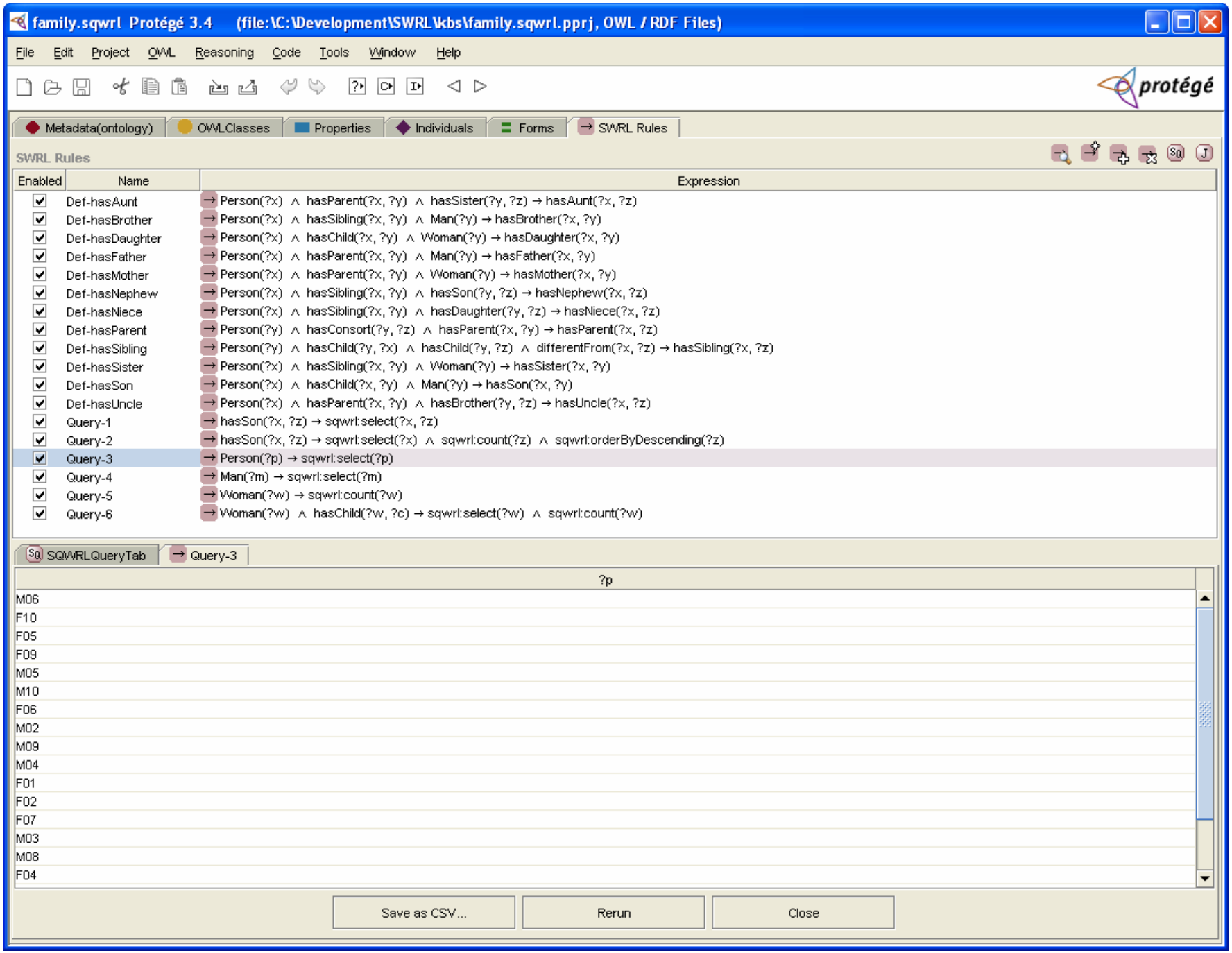


SQWRLTab

- 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.sqwrl Protégé 3.4 (file:\C:\Development\SWRL\kbs\family.sqwrl.pprj, OWL / RDF Files)
File Edit Project OWL Reasoning Code Tools Window Help
protégé
Metadata(ontology) OWLClasses Properties Individuals Forms SWRL Rules
SWRL Rules
Enabled Name Expression
Def-hasAunt Person(?x) ∧ hasParent(?x, ?y) ∧ hasSister(?y, ?z) → hasAunt(?x, ?z)
Def-hasBrother Person(?x) ∧ hasSibling(?x, ?y) ∧ Man(?y) → hasBrother(?x, ?y)
Def-hasDaughter Person(?x) ∧ hasChild(?x, ?y) ∧ Woman(?y) → hasDaughter(?x, ?y)
Def-hasFather Person(?x) ∧ hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
Def-hasMother Person(?x) ∧ hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
Def-hasNephew Person(?x) ∧ hasSibling(?x, ?y) ∧ hasSon(?y, ?z) → hasNephew(?x, ?z)
Def-hasNiece Person(?x) ∧ hasSibling(?x, ?y) ∧ hasDaughter(?y, ?z) → hasNiece(?x, ?z)
Def-hasParent Person(?y) ∧ hasConsort(?y, ?z) ∧ hasParent(?x, ?y) → hasParent(?x, ?z)
Def-hasSibling Person(?y) ∧ hasChild(?y, ?x) ∧ hasChild(?y, ?z) ∧ differentFrom(?x, ?z) → hasSibling(?x, ?z)
Def-hasSister Person(?x) ∧ hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
Def-hasSon Person(?x) ∧ hasChild(?x, ?y) ∧ Man(?y) → hasSon(?x, ?y)
Def-hasUncle Person(?x) ∧ hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)
Query-1 hasSon(?x, ?z) → sqwrl:select(?x, ?z)
Query-2 hasSon(?x, ?z) → sqwrl:select(?x) ∧ sqwrl:count(?z) ∧ sqwrl:orderByDescending(?z)
Query-3 Person(?p) → sqwrl:select(?p)
Query-4 Man(?m) → sqwrl:select(?m)
Query-5 Woman(?w) → sqwrl:count(?w)
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.sqwrl Protégé 3.4 (file:\C:\Development\SWRL\kbs\family.sqwrl.pprj, OWL / RDF Files)
File Edit Project OWL Reasoning Code Tools Window Help
protégé
Metadata(ontology)
SWRL Rules
Enabled
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-1
Query-2
Query-3
Query-4
Query-5
Query-6
SQWRLQueryTab
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
Run
SWRL Rule
Name
Comment
Name
http://a.com/ontology#Query-3
SWRL Rule
Person(?p)
→ sqwrl:select(?p)


family.sqwrl Protégé 3.4 (file:\C:\Development\SWRL\kbs\family.sqwrl.pprj, OWL / RDF Files)
File Edit Project OWL Reasoning Code Tools Window Help
protégé
Metadata(ontology) OWLClasses Properties Individuals Forms SWRL Rules
SWRL Rules
Enabled Name Expression
Def-hasAunt Person(?x) ∧ hasParent(?x, ?y) ∧ hasSister(?y, ?z) → hasAunt(?x, ?z)
Def-hasBrother Person(?x) ∧ hasSibling(?x, ?y) ∧ Man(?y) → hasBrother(?x, ?y)
Def-hasDaughter Person(?x) ∧ hasChild(?x, ?y) ∧ Woman(?y) → hasDaughter(?x, ?y)
Def-hasFather Person(?x) ∧ hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
Def-hasMother Person(?x) ∧ hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
Def-hasNephew Person(?x) ∧ hasSibling(?x, ?y) ∧ hasSon(?y, ?z) → hasNephew(?x, ?z)
Def-hasNiece Person(?x) ∧ hasSibling(?x, ?y) ∧ hasDaughter(?y, ?z) → hasNiece(?x, ?z)
Def-hasParent Person(?y) ∧ hasConsort(?y, ?z) ∧ hasParent(?x, ?y) → hasParent(?x, ?z)
Def-hasSibling Person(?y) ∧ hasChild(?y, ?x) ∧ hasChild(?y, ?z) ∧ differentFrom(?x, ?z) → hasSibling(?x, ?z)
Def-hasSister Person(?x) ∧ hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
Def-hasSon Person(?x) ∧ hasChild(?x, ?y) ∧ Man(?y) → hasSon(?x, ?y)
Def-hasUncle Person(?x) ∧ hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)
Query-1 hasSon(?x, ?z) → sqwrl:select(?x, ?z)
Query-2 hasSon(?x, ?z) → sqwrl:select(?x) ∧ sqwrl:count(?z) ∧ sqwrl:orderByDescending(?z)
Query-3 Person(?p) → sqwrl:select(?p)
Query-4 Man(?m) → sqwrl:select(?m)
Query-5 Woman(?w) → sqwrl:count(?w)
Query-6 Woman(?w) ∧ hasChild(?w, ?c) → sqwrl:select(?w) ∧ sqwrl:count(?w)
SQWRLQueryTab Query-3
?p
M06
F10
F05
F09
M05
M10
F06
M02
M09
M04
F01
F02
F07
M03
M08
F04
Save as CSV... Rerun Close

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