A Mechanism to Define and Execute **SWRL Built-ins** in Protégé-OWL

Martin O'Connor
Stanford Medical Informatics, Stanford University

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)

Example SWRL Rule: Has brother

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

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)

protégé

SWRLTab

The SWRLTab is an extension to Protege-OWL that supports editing and execution of SWRL rules. It provides a graphical editor to create and modify SWRL rules in an OWL knowledge base. It also provides extension mechanisms to support the execution of SWRL rules with a variety of rule engines. At present, the Jess rule engine is supported. (6GX)

The SWRLTab has four main software components: (6GY)

- **SWRL Editor:** The editor supports editing and saving of SWRL rules in an OWL knowledge base. See the SWRL Editor FAQ for more details. An introduction to the SWRL language can be found here. (6ZI)
- **SWRL Factory:** The factory provides high-level Java APIs that support the creation and modification of SWRL rules in an OWL knowledge base. 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. 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. 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 Built-in Bridge:** SWRL built-ins are predicates that accept one or more arguments. These predicates can be used in SWRL rules to support the definition of arbitrary user-defined built-ins, which can then be used in 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)

Your Visited Pages

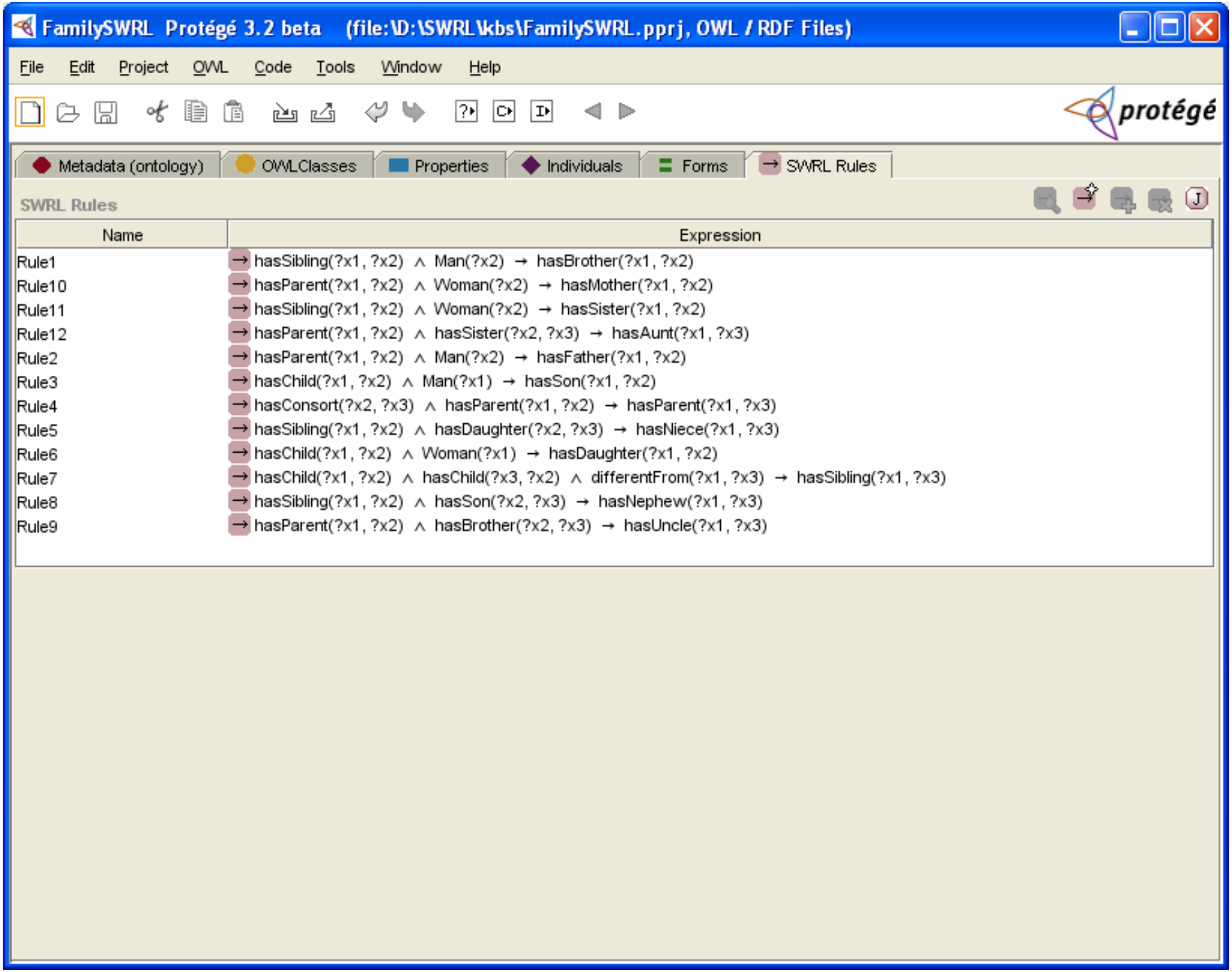
SWRLTab

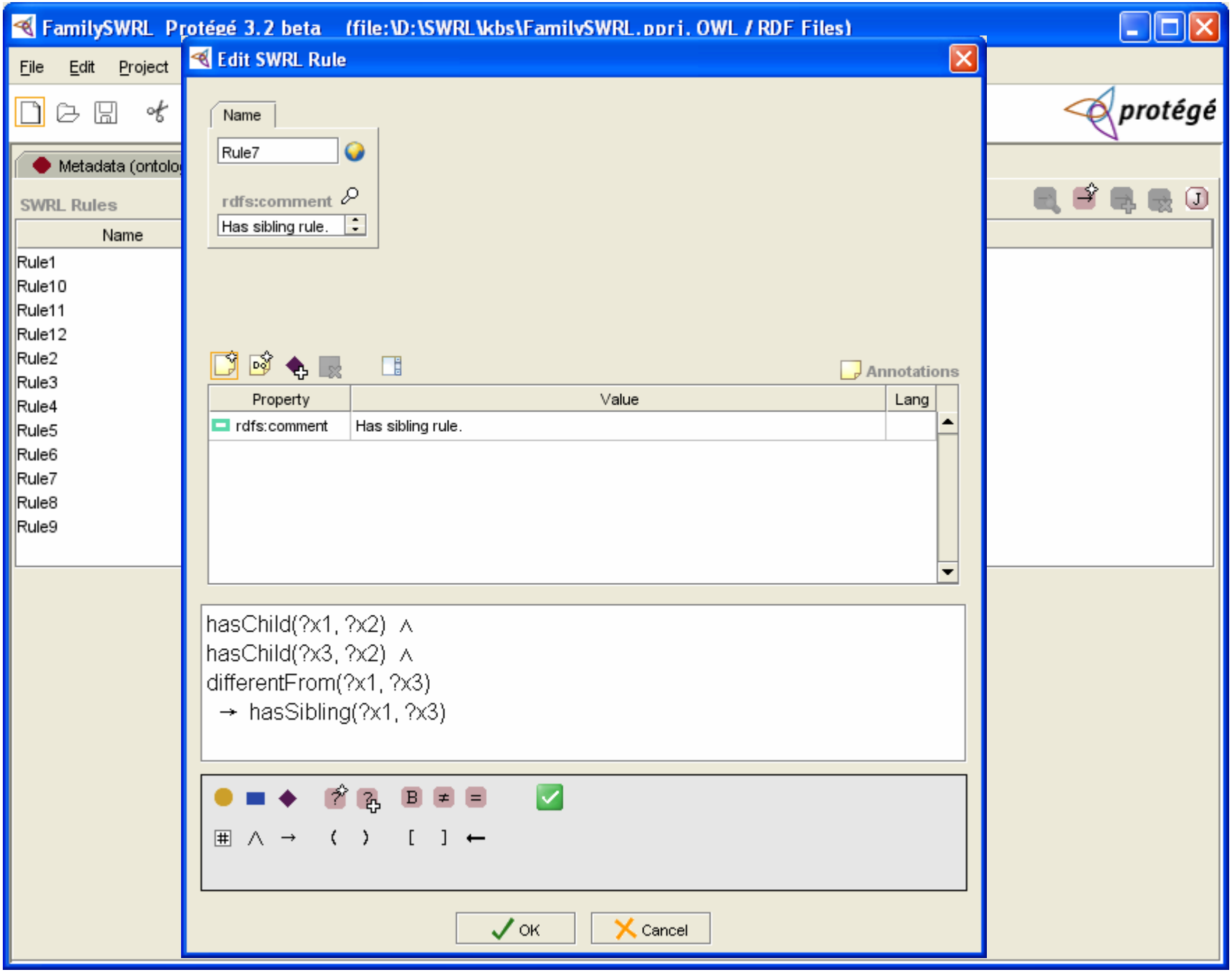
View Backlinks

Search

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.


FamilySWRL Protégé 3.2 beta (file:\D:\SWRL\kbs\FamilySWRL.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
Rule1 hasSibling(?x1, ?x2) ∧ Man(?x2) → hasBrother(?x1, ?x2)
Rule10 hasParent(?x1, ?x2) ∧ Woman(?x2) → hasMother(?x1, ?x2)
Rule11 hasSibling(?x1, ?x2) ∧ Woman(?x2) → hasSister(?x1, ?x2)
Rule12 hasParent(?x1, ?x2) ∧ hasSister(?x2, ?x3) → hasAunt(?x1, ?x3)
Rule2 hasParent(?x1, ?x2) ∧ Man(?x2) → hasFather(?x1, ?x2)
Rule3 hasChild(?x1, ?x2) ∧ Man(?x1) → hasSon(?x1, ?x2)
Rule4 hasConsort(?x2, ?x3) ∧ hasParent(?x1, ?x2) → hasParent(?x1, ?x3)
Rule5 hasSibling(?x1, ?x2) ∧ hasDaughter(?x2, ?x3) → hasNiece(?x1, ?x3)
Rule6 hasChild(?x1, ?x2) ∧ Woman(?x1) → hasDaughter(?x1, ?x2)
Rule7 hasChild(?x1, ?x2) ∧ hasChild(?x3, ?x2) ∧ differentFrom(?x1, ?x3) → hasSibling(?x1, ?x3)
Rule8 hasSibling(?x1, ?x2) ∧ hasSon(?x2, ?x3) → hasNephew(?x1, ?x3)
Rule9 hasParent(?x1, ?x2) ∧ hasBrother(?x2, ?x3) → hasUncle(?x1, ?x3)


FamilySWRL Protégé 3.2 beta (file:\D:\SWRL\kbs\FamilySWRL.pprj, OWL / RDF Files)
File
Edit
Project
protégé
Metadata (ontolo
SWRL Rules
Name
Rule1
Rule10
Rule11
Rule12
Rule2
Rule3
Rule4
Rule5
Rule6
Rule7
Rule8
Rule9
Edit SWRL Rule
Name
Rule7
rdfs:comment
Has sibling rule.
Annotations
Property
Value
Lang
rdfs:comment
Has sibling rule.
hasChild(?x1, ?x2) ∧
hasChild(?x3, ?x2) ∧
differentFrom(?x1, ?x3)
→ hasSibling(?x1, ?x3)
OK
Cancel

SWRL Factory API

- The SWRL API provides a mechanism to create and manipulate SWRL rules in an OWL knowledge base.
- This API is used by the SWRL Editor. However, it is accessible to all OWL Plugin developers.
- Third party software can use this API to work directly with SWRL rules, e.g., new SWRL editor or third-party rule engine developers.
- Fully documented in SWRLTab Wiki.

SWRL Rule Engine Bridge

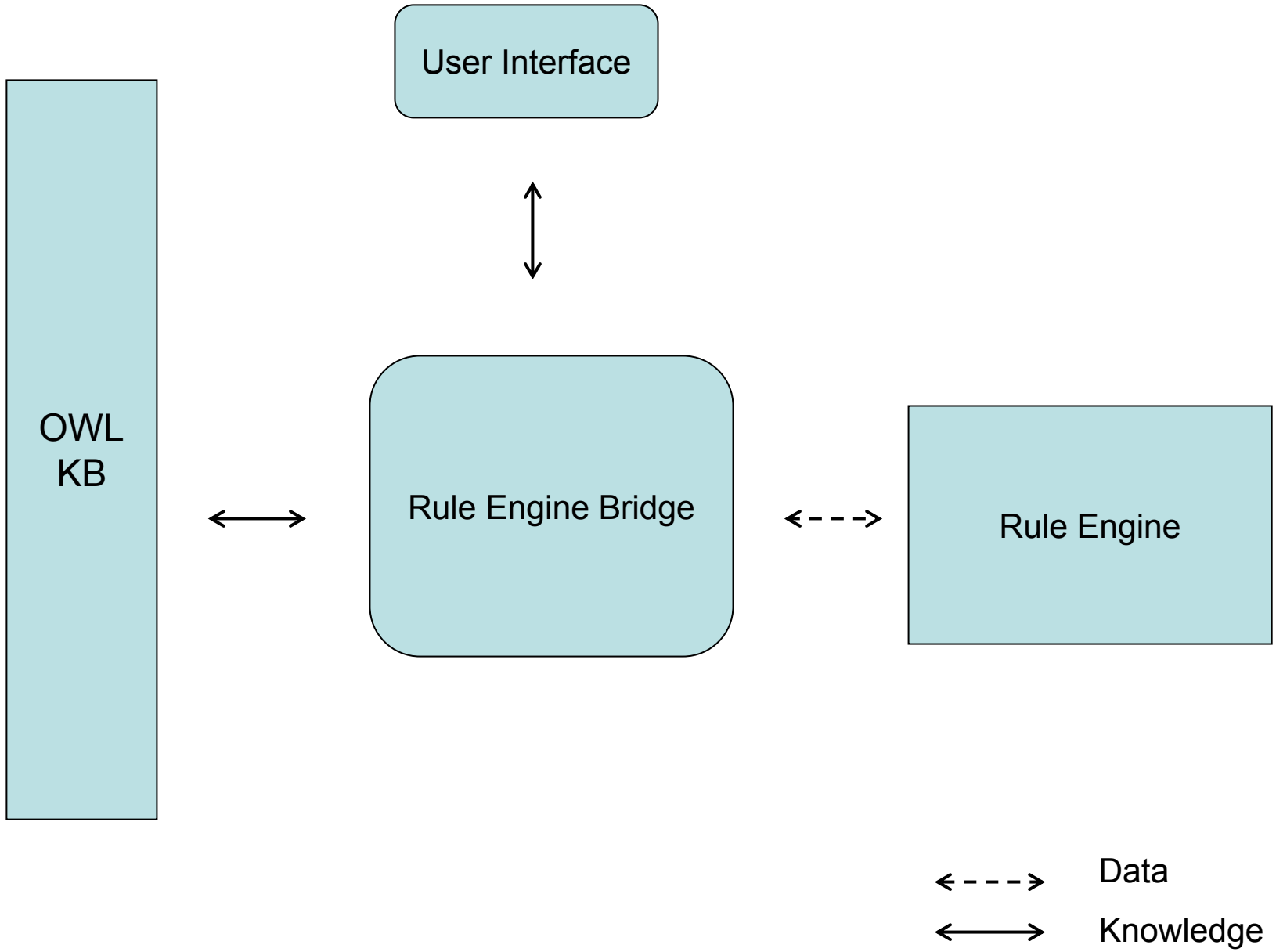
- Given an OWL knowledge base it will extract SWRL rules and relevant OWL knowledge.
- Also provides an API to assert inferred knowledge.
- Knowledge (and rules) are described in non Protégé-OWL API-specific way.
- These can then be mapped to a rule-engine specific rule and knowledge format.
- This mapping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

Rule Engine Interaction with SWRL Rules

- Before mapping, extracting relevant OWL knowledge for inference is an important optimization.
- Not all knowledge needs to be extracted.
- Required knowledge can be determined rules.
- For example, the rule: Man(Fred) ^ Man(?y) ^ hasParent(Fred, ?y) ^ hasBrother(?y,?z) -> hasUncle(Fred, ?z) requires:
  - The individual named **Fred**
  - All individuals of class **Man** and subclasses
  - Fred's **hasParent** properties and subproperties.
  - All individuals with the **hasBrother** property and subproperties.

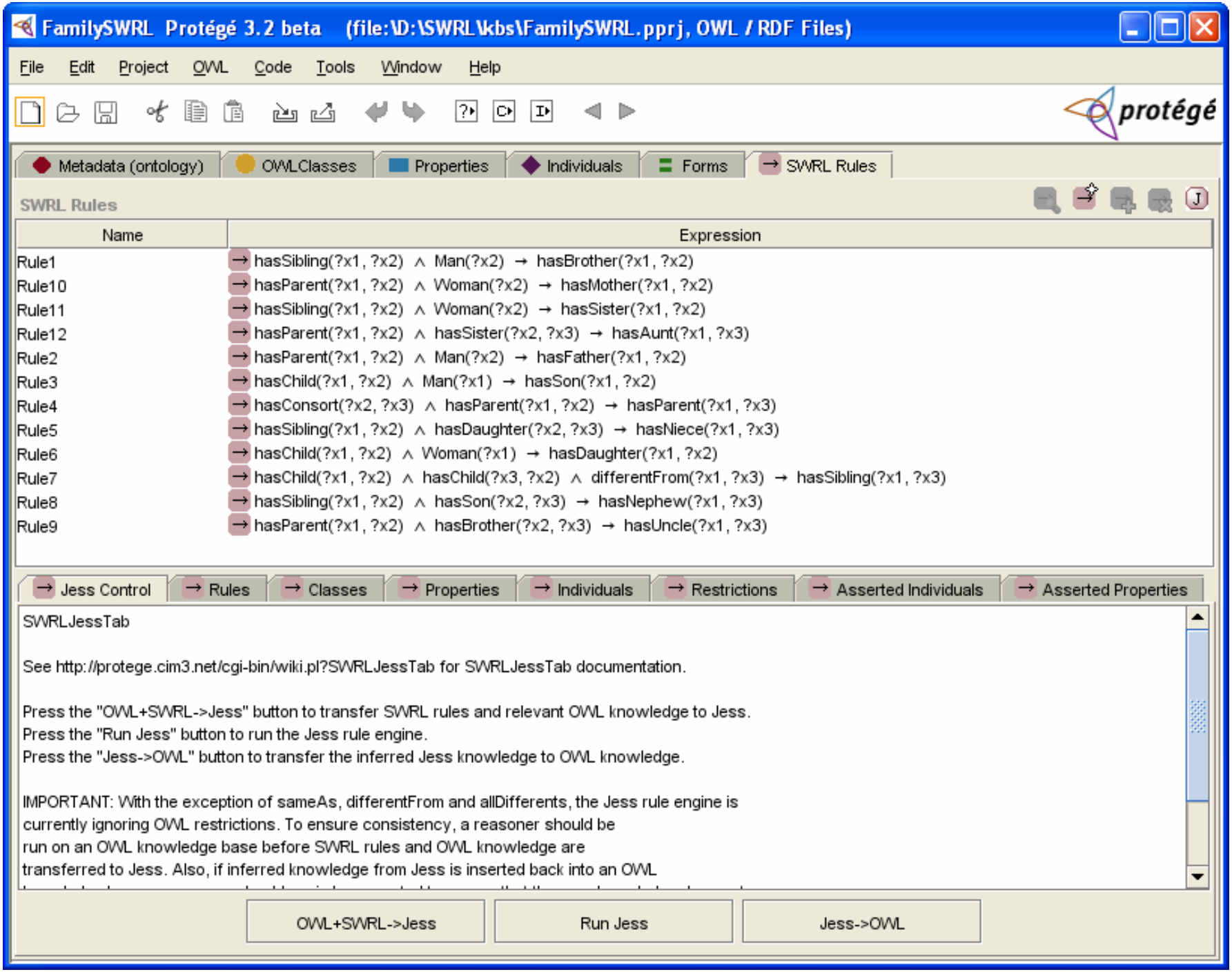
High-level Steps to Integrate Rule Engine with Protégé-OWL

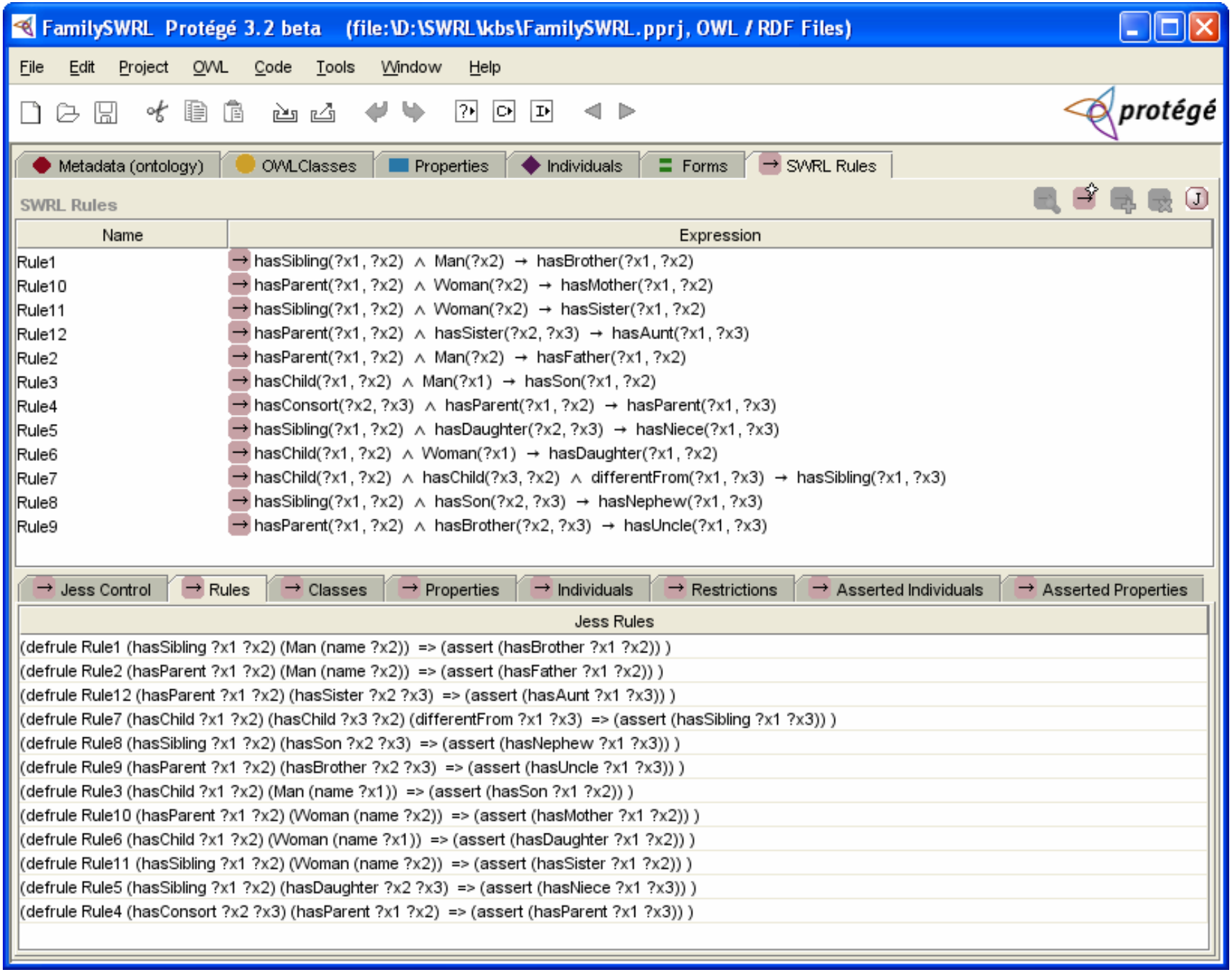
- Use SWRL API to get all rules in knowledge base.
- Use OWL API to get all relevant OWL knowledge.
- Map OWL knowledge to rule engine knowledge.
- Perform inference!
- Map created rule engine knowledge to OWL.
- Use OWL API to put new information into OWL knowledge base.
- Also: GUI real estate is usually required.
- Other issues: integrity checking.

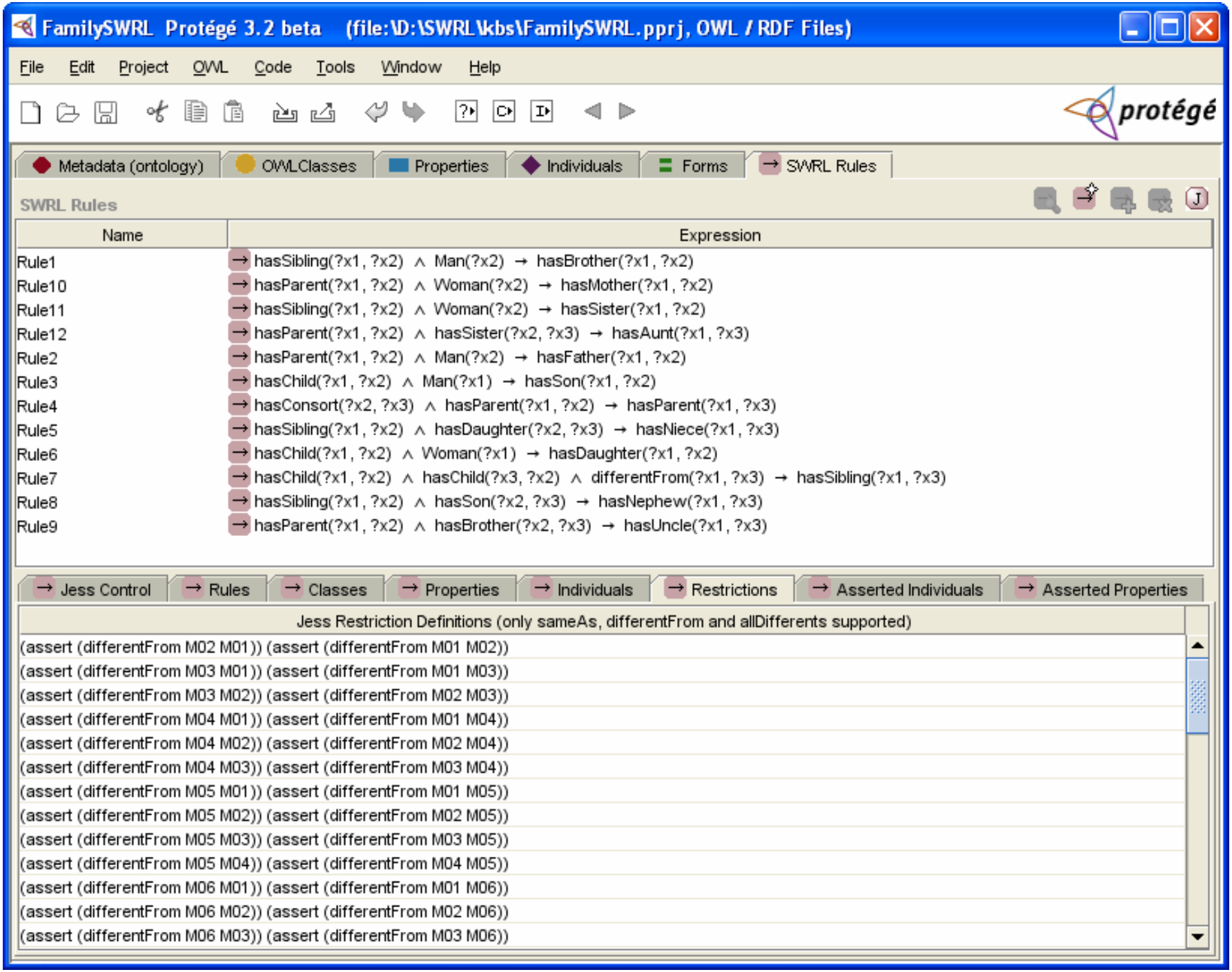

User Interface
OWL
KB
Rule Engine Bridge
Rule Engine
Data
Knowledge

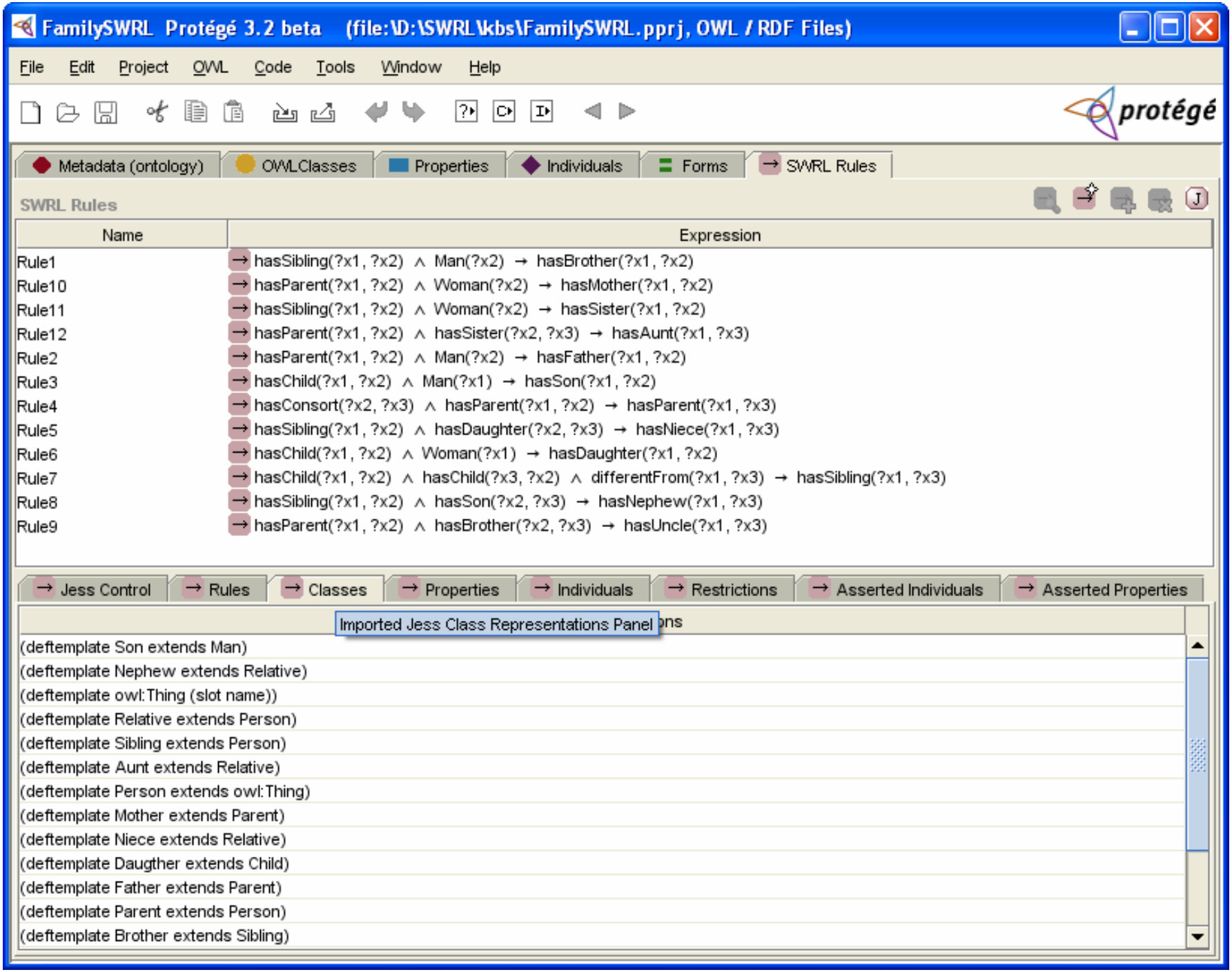
We used the SWRL Bridge to Integrate Jess Rule Engine with Protégé-OWL

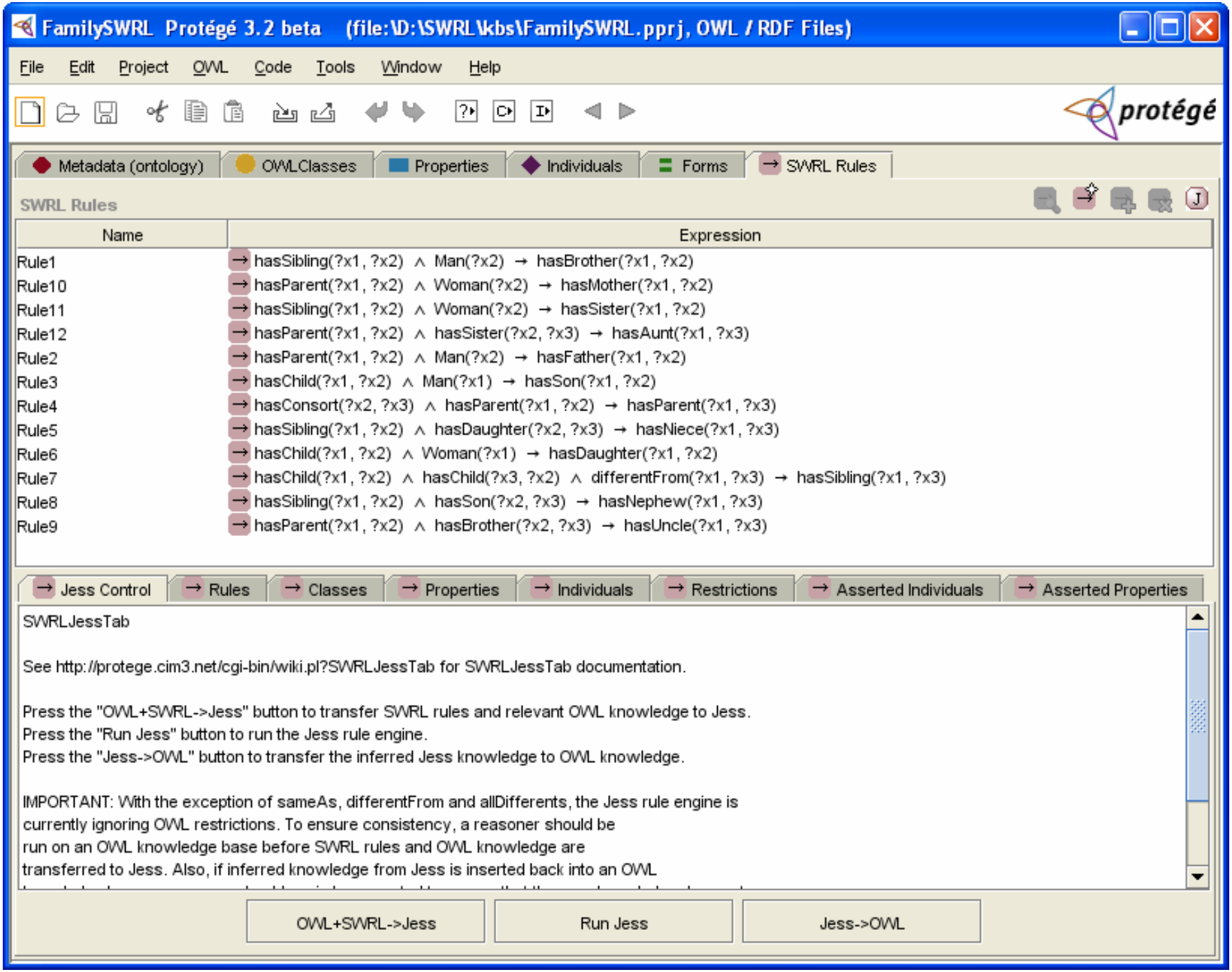
- Jess is a Java-based rule engine.
- Jess system consists of a rule base, fact base, and an execution engine.
- Available free to academic users, for a small fee to non-academic users
- Has been used in Protégé-based tools, e.g., JessTab.

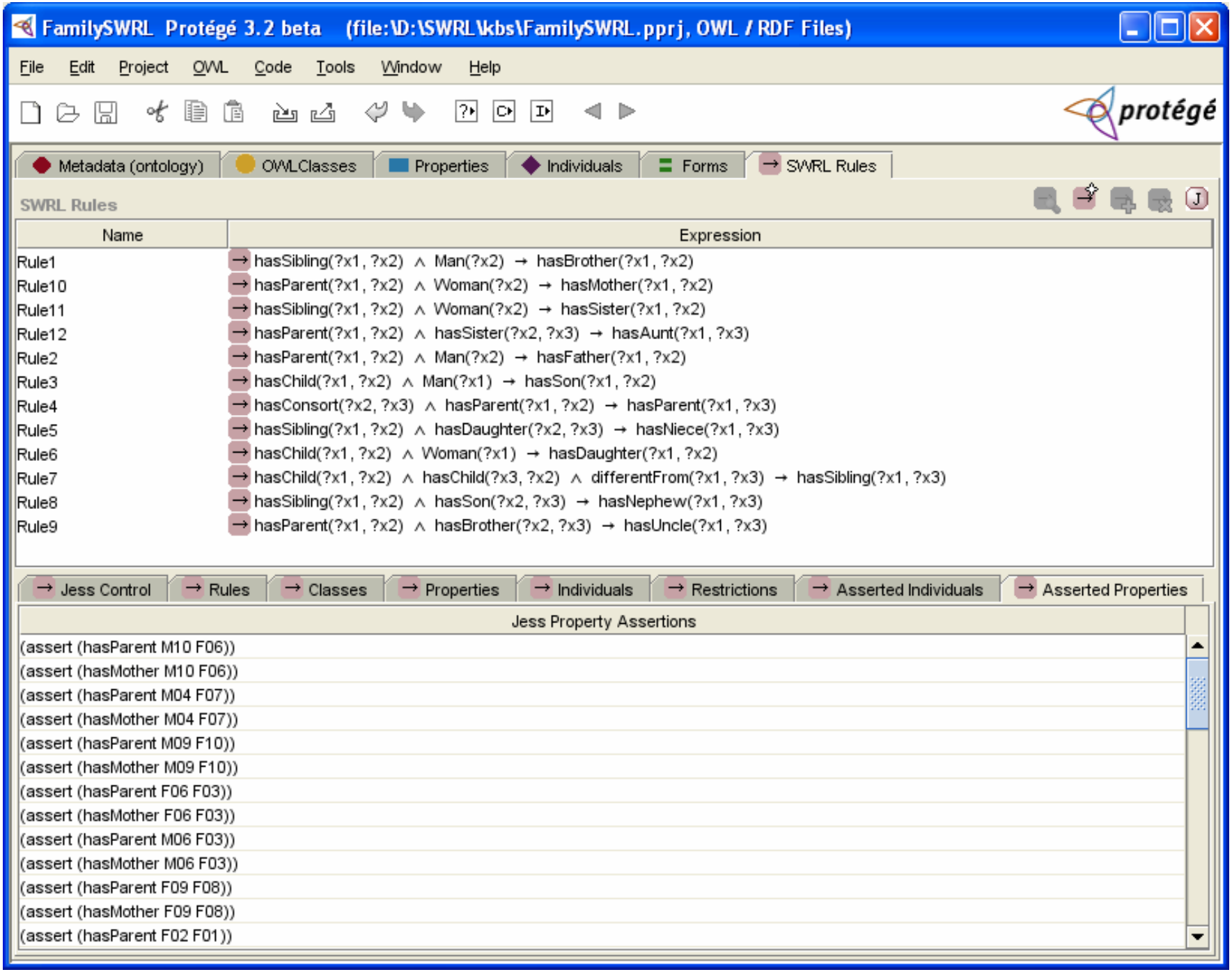

FamilySWRL Protégé 3.2 beta (file:\D:\SWRL\kbs\FamilySWRL.pprj, OWL / RDF Files)
File Edit Project OWL Code Tools Window Help
protégé
Metadata (ontology) OWLClasses Properties Individuals Forms SWRL Rules
SWRL Rules
Name Expression
Rule1 hasSibling(?x1, ?x2) ∧ Man(?x2) → hasBrother(?x1, ?x2)
Rule10 hasParent(?x1, ?x2) ∧ Woman(?x2) → hasMother(?x1, ?x2)
Rule11 hasSibling(?x1, ?x2) ∧ Woman(?x2) → hasSister(?x1, ?x2)
Rule12 hasParent(?x1, ?x2) ∧ hasSister(?x2, ?x3) → hasAunt(?x1, ?x3)
Rule2 hasParent(?x1, ?x2) ∧ Man(?x2) → hasFather(?x1, ?x2)
Rule3 hasChild(?x1, ?x2) ∧ Man(?x1) → hasSon(?x1, ?x2)
Rule4 hasConsort(?x2, ?x3) ∧ hasParent(?x1, ?x2) → hasParent(?x1, ?x3)
Rule5 hasSibling(?x1, ?x2) ∧ hasDaughter(?x2, ?x3) → hasNiece(?x1, ?x3)
Rule6 hasChild(?x1, ?x2) ∧ Woman(?x1) → hasDaughter(?x1, ?x2)
Rule7 hasChild(?x1, ?x2) ∧ hasChild(?x3, ?x2) ∧ differentFrom(?x1, ?x3) → hasSibling(?x1, ?x3)
Rule8 hasSibling(?x1, ?x2) ∧ hasSon(?x2, ?x3) → hasNephew(?x1, ?x3)
Rule9 hasParent(?x1, ?x2) ∧ hasBrother(?x2, ?x3) → hasUncle(?x1, ?x3)
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 sameAs, differentFrom and allDifferents, 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
OWL+SWRL->Jess
Run Jess
Jess->OWL


FamilySWRL Protégé 3.2 beta (file:\D:\SWRL\kbs\FamilySWRL.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
Rule1 hasSibling(?x1, ?x2) ∧ Man(?x2) → hasBrother(?x1, ?x2)
Rule10 hasParent(?x1, ?x2) ∧ Woman(?x2) → hasMother(?x1, ?x2)
Rule11 hasSibling(?x1, ?x2) ∧ Woman(?x2) → hasSister(?x1, ?x2)
Rule12 hasParent(?x1, ?x2) ∧ hasSister(?x2, ?x3) → hasAunt(?x1, ?x3)
Rule2 hasParent(?x1, ?x2) ∧ Man(?x2) → hasFather(?x1, ?x2)
Rule3 hasChild(?x1, ?x2) ∧ Man(?x1) → hasSon(?x1, ?x2)
Rule4 hasConsort(?x2, ?x3) ∧ hasParent(?x1, ?x2) → hasParent(?x1, ?x3)
Rule5 hasSibling(?x1, ?x2) ∧ hasDaughter(?x2, ?x3) → hasNiece(?x1, ?x3)
Rule6 hasChild(?x1, ?x2) ∧ Woman(?x1) → hasDaughter(?x1, ?x2)
Rule7 hasChild(?x1, ?x2) ∧ hasChild(?x3, ?x2) ∧ differentFrom(?x1, ?x3) → hasSibling(?x1, ?x3)
Rule8 hasSibling(?x1, ?x2) ∧ hasSon(?x2, ?x3) → hasNephew(?x1, ?x3)
Rule9 hasParent(?x1, ?x2) ∧ hasBrother(?x2, ?x3) → hasUncle(?x1, ?x3)
Jess Control Rules Classes Properties Individuals Restrictions Asserted Individuals Asserted Properties
Jess Rules
(defrule Rule1 (hasSibling ?x1 ?x2) (Man (name ?x2)) => (assert (hasBrother ?x1 ?x2)))
(defrule Rule2 (hasParent ?x1 ?x2) (Man (name ?x2)) => (assert (hasFather ?x1 ?x2)))
(defrule Rule12 (hasParent ?x1 ?x2) (hasSister ?x2 ?x3) => (assert (hasAunt ?x1 ?x3)))
(defrule Rule7 (hasChild ?x1 ?x2) (hasChild ?x3 ?x2) (differentFrom ?x1 ?x3) => (assert (hasSibling ?x1 ?x3)))
(defrule Rule8 (hasSibling ?x1 ?x2) (hasSon ?x2 ?x3) => (assert (hasNephew ?x1 ?x3)))
(defrule Rule9 (hasParent ?x1 ?x2) (hasBrother ?x2 ?x3) => (assert (hasUncle ?x1 ?x3)))
(defrule Rule3 (hasChild ?x1 ?x2) (Man (name ?x1)) => (assert (hasSon ?x1 ?x2)))
(defrule Rule10 (hasParent ?x1 ?x2) (Woman (name ?x2)) => (assert (hasMother ?x1 ?x2)))
(defrule Rule6 (hasChild ?x1 ?x2) (Woman (name ?x1)) => (assert (hasDaughter ?x1 ?x2)))
(defrule Rule11 (hasSibling ?x1 ?x2) (Woman (name ?x2)) => (assert (hasSister ?x1 ?x2)))
(defrule Rule5 (hasSibling ?x1 ?x2) (hasDaughter ?x2 ?x3) => (assert (hasNiece ?x1 ?x3)))
(defrule Rule4 (hasConsort ?x2 ?x3) (hasParent ?x1 ?x2) => (assert (hasParent ?x1 ?x3)))


FamilySWRL Protégé 3.2 beta (file:\D:\SWRL\kbs\FamilySWRL.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
Rule1 hasSibling(?x1, ?x2) ∧ Man(?x2) → hasBrother(?x1, ?x2)
Rule10 hasParent(?x1, ?x2) ∧ Woman(?x2) → hasMother(?x1, ?x2)
Rule11 hasSibling(?x1, ?x2) ∧ Woman(?x2) → hasSister(?x1, ?x2)
Rule12 hasParent(?x1, ?x2) ∧ hasSister(?x2, ?x3) → hasAunt(?x1, ?x3)
Rule2 hasParent(?x1, ?x2) ∧ Man(?x2) → hasFather(?x1, ?x2)
Rule3 hasChild(?x1, ?x2) ∧ Man(?x1) → hasSon(?x1, ?x2)
Rule4 hasConsort(?x2, ?x3) ∧ hasParent(?x1, ?x2) → hasParent(?x1, ?x3)
Rule5 hasSibling(?x1, ?x2) ∧ hasDaughter(?x2, ?x3) → hasNiece(?x1, ?x3)
Rule6 hasChild(?x1, ?x2) ∧ Woman(?x1) → hasDaughter(?x1, ?x2)
Rule7 hasChild(?x1, ?x2) ∧ hasChild(?x3, ?x2) ∧ differentFrom(?x1, ?x3) → hasSibling(?x1, ?x3)
Rule8 hasSibling(?x1, ?x2) ∧ hasSon(?x2, ?x3) → hasNephew(?x1, ?x3)
Rule9 hasParent(?x1, ?x2) ∧ hasBrother(?x2, ?x3) → hasUncle(?x1, ?x3)
Jess Control Rules Classes Properties Individuals Restrictions Asserted Individuals Asserted Properties
Jess Restriction Definitions (only sameAs, differentFrom and allDifferents supported)
(assert (differentFrom M02 M01)) (assert (differentFrom M01 M02))
(assert (differentFrom M03 M01)) (assert (differentFrom M01 M03))
(assert (differentFrom M03 M02)) (assert (differentFrom M02 M03))
(assert (differentFrom M04 M01)) (assert (differentFrom M01 M04))
(assert (differentFrom M04 M02)) (assert (differentFrom M02 M04))
(assert (differentFrom M04 M03)) (assert (differentFrom M03 M04))
(assert (differentFrom M05 M01)) (assert (differentFrom M01 M05))
(assert (differentFrom M05 M02)) (assert (differentFrom M02 M05))
(assert (differentFrom M05 M03)) (assert (differentFrom M03 M05))
(assert (differentFrom M05 M04)) (assert (differentFrom M04 M05))
(assert (differentFrom M06 M01)) (assert (differentFrom M01 M06))
(assert (differentFrom M06 M02)) (assert (differentFrom M02 M06))
(assert (differentFrom M06 M03)) (assert (differentFrom M03 M06))


FamilySWRL Protégé 3.2 beta (file:\D:\SWRL\kbs\FamilySWRL.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
Rule1 hasSibling(?x1, ?x2) ∧ Man(?x2) → hasBrother(?x1, ?x2)
Rule10 hasParent(?x1, ?x2) ∧ Woman(?x2) → hasMother(?x1, ?x2)
Rule11 hasSibling(?x1, ?x2) ∧ Woman(?x2) → hasSister(?x1, ?x2)
Rule12 hasParent(?x1, ?x2) ∧ hasSister(?x2, ?x3) → hasAunt(?x1, ?x3)
Rule2 hasParent(?x1, ?x2) ∧ Man(?x2) → hasFather(?x1, ?x2)
Rule3 hasChild(?x1, ?x2) ∧ Man(?x1) → hasSon(?x1, ?x2)
Rule4 hasConsort(?x2, ?x3) ∧ hasParent(?x1, ?x2) → hasParent(?x1, ?x3)
Rule5 hasSibling(?x1, ?x2) ∧ hasDaughter(?x2, ?x3) → hasNiece(?x1, ?x3)
Rule6 hasChild(?x1, ?x2) ∧ Woman(?x1) → hasDaughter(?x1, ?x2)
Rule7 hasChild(?x1, ?x2) ∧ hasChild(?x3, ?x2) ∧ differentFrom(?x1, ?x3) → hasSibling(?x1, ?x3)
Rule8 hasSibling(?x1, ?x2) ∧ hasSon(?x2, ?x3) → hasNephew(?x1, ?x3)
Rule9 hasParent(?x1, ?x2) ∧ hasBrother(?x2, ?x3) → hasUncle(?x1, ?x3)
Jess Control Rules Classes Properties Individuals Restrictions Asserted Individuals Asserted Properties
Imported Jess Class Representations Panel
(deftemplate Son extends Man)
(deftemplate Nephew extends Relative)
(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 Brother extends Sibling)


FamilySWRL Protégé 3.2 beta (file:\D:\SWRL\kbs\FamilySWRL.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
Rule1 hasSibling(?x1, ?x2) ∧ Man(?x2) → hasBrother(?x1, ?x2)
Rule10 hasParent(?x1, ?x2) ∧ Woman(?x2) → hasMother(?x1, ?x2)
Rule11 hasSibling(?x1, ?x2) ∧ Woman(?x2) → hasSister(?x1, ?x2)
Rule12 hasParent(?x1, ?x2) ∧ hasSister(?x2, ?x3) → hasAunt(?x1, ?x3)
Rule2 hasParent(?x1, ?x2) ∧ Man(?x2) → hasFather(?x1, ?x2)
Rule3 hasChild(?x1, ?x2) ∧ Man(?x1) → hasSon(?x1, ?x2)
Rule4 hasConsort(?x2, ?x3) ∧ hasParent(?x1, ?x2) → hasParent(?x1, ?x3)
Rule5 hasSibling(?x1, ?x2) ∧ hasDaughter(?x2, ?x3) → hasNiece(?x1, ?x3)
Rule6 hasChild(?x1, ?x2) ∧ Woman(?x1) → hasDaughter(?x1, ?x2)
Rule7 hasChild(?x1, ?x2) ∧ hasChild(?x3, ?x2) ∧ differentFrom(?x1, ?x3) → hasSibling(?x1, ?x3)
Rule8 hasSibling(?x1, ?x2) ∧ hasSon(?x2, ?x3) → hasNephew(?x1, ?x3)
Rule9 hasParent(?x1, ?x2) ∧ hasBrother(?x2, ?x3) → hasUncle(?x1, ?x3)
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 sameAs, differentFrom and allDifferents, 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
OWL+SWRL->Jess
Run Jess
Jess->OWL


FamilySWRL Protégé 3.2 beta (file:\D:\SWRL\kbs\FamilySWRL.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
Rule1 hasSibling(?x1, ?x2) ∧ Man(?x2) → hasBrother(?x1, ?x2)
Rule10 hasParent(?x1, ?x2) ∧ Woman(?x2) → hasMother(?x1, ?x2)
Rule11 hasSibling(?x1, ?x2) ∧ Woman(?x2) → hasSister(?x1, ?x2)
Rule12 hasParent(?x1, ?x2) ∧ hasSister(?x2, ?x3) → hasAunt(?x1, ?x3)
Rule2 hasParent(?x1, ?x2) ∧ Man(?x2) → hasFather(?x1, ?x2)
Rule3 hasChild(?x1, ?x2) ∧ Man(?x1) → hasSon(?x1, ?x2)
Rule4 hasConsort(?x2, ?x3) ∧ hasParent(?x1, ?x2) → hasParent(?x1, ?x3)
Rule5 hasSibling(?x1, ?x2) ∧ hasDaughter(?x2, ?x3) → hasNiece(?x1, ?x3)
Rule6 hasChild(?x1, ?x2) ∧ Woman(?x1) → hasDaughter(?x1, ?x2)
Rule7 hasChild(?x1, ?x2) ∧ hasChild(?x3, ?x2) ∧ differentFrom(?x1, ?x3) → hasSibling(?x1, ?x3)
Rule8 hasSibling(?x1, ?x2) ∧ hasSon(?x2, ?x3) → hasNephew(?x1, ?x3)
Rule9 hasParent(?x1, ?x2) ∧ hasBrother(?x2, ?x3) → hasUncle(?x1, ?x3)
Jess Control Rules Classes Properties Individuals Restrictions Asserted Individuals Asserted Properties
Jess Property Assertions
(assert (hasParent M10 F06))
(assert (hasMother M10 F06))
(assert (hasParent M04 F07))
(assert (hasMother M04 F07))
(assert (hasParent M09 F10))
(assert (hasMother M09 F10))
(assert (hasParent F06 F03))
(assert (hasMother F06 F03))
(assert (hasParent M06 F03))
(assert (hasMother M06 F03))
(assert (hasParent F09 F08))
(assert (hasMother F09 F08))
(assert (hasParent F02 F01))

Outstanding Issues

- SWRL Bridge does not know about all OWL constraints.
  - Contradictions with rules possible!
  - Consistency must be assured by the user running a reasoner.
  - Hard problem to solve in general.
- Integrated reasoner and rule engine would be ideal.
- Possible solution with KAON2.

SWRL Built-in Bridge

- SWRL provides mechanisms to add user-defined predicates, e.g.,
  - hasDOB(?x, ?y) ^ temporal:before(?y, ‘1997’)…
  - hasDOB(?x, ?y) ^ temporal:equals(?y, ‘2000’)…
- 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 an OWL file using definition of swrl:Builtin provided by SWRL ontology
- Provide Java implementation of built-ins and wrap in JAR file
- Load built-in definition file in Protégé-OWL
- Put JAR file in Protégé-OWL 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 SWRLBuiltInMethodsImpl implements SWRLBuiltInMethods
{
  public boolean stringEqualIgnoreCase(List arguments) throws BuiltInException { ... }
....
} // SWRLBuiltInMethodsImpl
```

Example Implementation for Built-in swrlb:stringEqualIgnoreCase

```
private static String SWRLB_SEIC = "stringEqualIgnoreCase";

public boolean stringEqualIgnoreCase(List arguments) throws BuiltInException
{
  SWRLBuiltInUtil.checkNumberOfArgumentsEqualTo(SWRLB_SEIC, 2, arguments.size());

  String argument1 = SWRLBuiltInUtil.getArgumentAsAString(SWRLB_SEIC, 1, arguments);
  String argument2 = SWRLBuiltInUtil.getArgumentAsAString(SWRLB_SEIC, 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 naively by rule engine, e,g., add, subtract, etc.

Example Built-in Library: Temporal Operators

- SWRL has limited temporal support
- OWL and SWRL suffer similar limitations to the relational model:
  - no temporal model
  - limited temporal operators in SWRL
- Hence we have developed a
  - temporal ontology
  - temporal extensions to SWRL

Temporal Extensions

- The temporal ontology provides a standard mechanism for representing temporal data
- The temporal operators provide a rich set of temporal operators, e.g.,
  - before
  - after
  - during

Example SWRL Rule: Constraints

On days that both immunotherapy and omalzumab are administered, omalzumab must be injected 60 minutes after immunotherapy.

Patient(?p) ^
hasExtendedEvent(?p, ?eevent1) ^ hasExtendedEvent(?p, ?eevent2) ^
temporal:hasValue(?eevent1, ?event1) ^ temporal:hasValidTime(?eevent1, ?event1VT) ^
temporal:hasTime(?event1VT, ?event1Time) ^ temporal:hasValue(?eevent2, ?event2) ^
temporal:hasValidTime(?eevent2, ?event2VT) ^ temporal:hasTime(?event2VT, ?event2Time) ^
hasVisit(?event1, ?v1) ^ hasVisit(?event2, ?v2) ^
hasActivity(?event1, ?a1) ^ hasName(?a1, "Omalizumab") ^
hasActivity(?event2, ?a2) ^ hasName(?a2, "Immunotherapy") ^
temporal:before(?event2Time, ?event1Time) ^
temporal:durationMinutesLessThan(60, ?event2Time, ?event1Time)
-> NonConformingPatient(?p)

Conclusion: Developers Needed!

- SWRLTab is open source.
- Well documented in Wiki:
  - http://protege.cim3.net/cgi-bin/wiki.pl?SWRLEditorFAQ
  - http://protege.cim3.net/cgi-bin/wiki.pl?SWRLFactoryFAQ
  - http://protege.cim3.net/cgi-bin/wiki.pl?SWRLRuleEngineBridgeFAQ
  - http://protege.cim3.net/cgi-bin/wiki.pl?SWRLBuiltInBridge
- New rule engines could be integrated
- Could be used to wrap existing method libraries as built-ins