Rule System Interoperability on the Semantic Web with SWRL

Martin O'Connor[1], Holger Knublauch[1], Samson Tu[1], Benjamin Grosof[2], Mike Dean[3], William Grosso[4], Mark Musen[1]

[1]Stanford Medical Informatics, Stanford CA,
[2]Sloan School of Management, MIT, Cambridge MA
[3]BBN Technologies, Ann Arbor MI
[4]Echopass Corp., San Francisco CA

What is SWRL?

- SWRL is an acronym for Semantic Web Rule Language.

- SWRL is based on OWL: all rules are expressed in terms of OWL concepts (classes, properties, individuals, literals…).

- SWRL includes a high-level abstract syntax for Horn-like rules.

Example SWRL Rule: Has uncle

hasParent(?x, ?y) ^ hasBrother(?y, ?z) -> hasUncle(?x, ?z)

Example SWRL Rule: Constraints

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

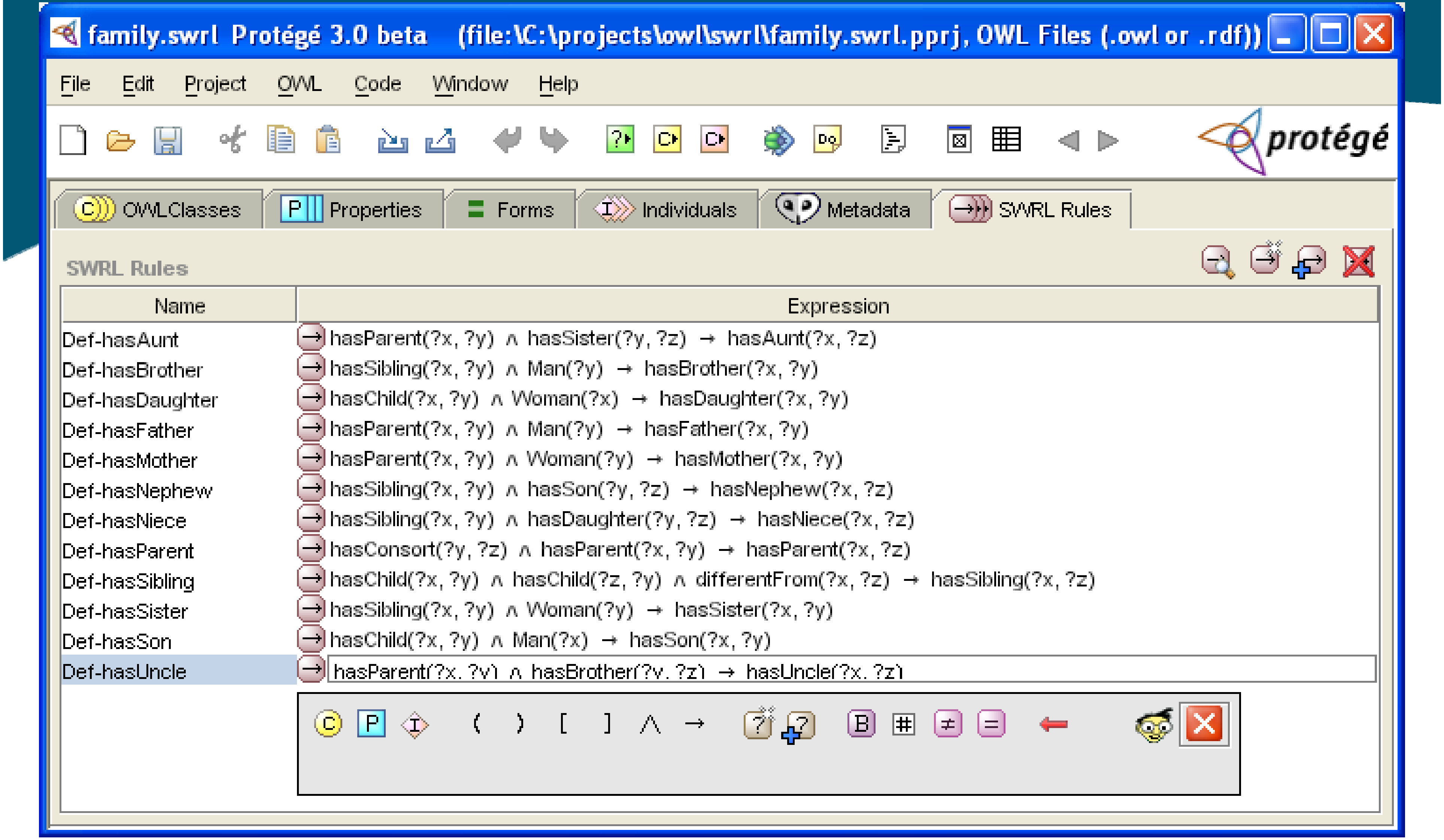
```
Patient(?p) ^
hasExtendedEvent(?p, ?eevent1) ^ hasExtendedEvent(?p, ?eevent2) ^
temporal:hasValue(?eevent1, ?event1) ^ temporal:hasValidTime(?eevent1, ?event1VT) ^
temporal:hasTime(?event1VT, ?event1Time) ^ temporal:hasValue(?eevent2, ?event2) ^
temporal:hasValidTime(?eevent2, ?event2VT) ^ temporal:hasTime(?event2VT,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

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.
- Provides Java APIs to allow interoperation with third-party inference engines.

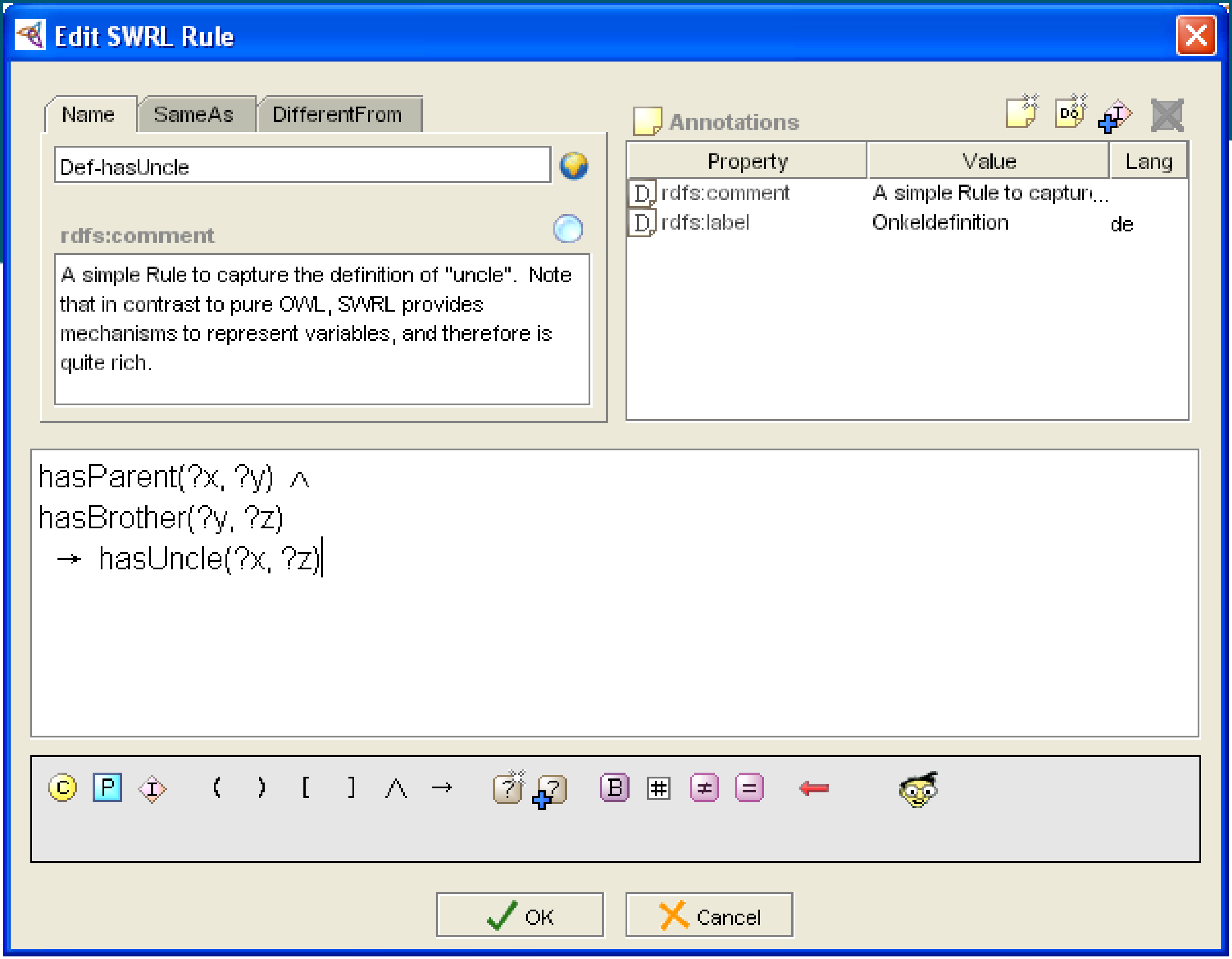
The SWRL Editor

- The SWRL Editor is included as part of Protégé-OWL.
- It is accessible as a tab within Protégé-OWL.
- This tab should be visible for all OWL knowledge bases that import the SWRL Ontology:
  - http://www.daml.org/rules/proposal/swrl.owl


family.swrl Protégé 3.0 beta (file:\C:\projects\owl\swrl\family.swrl.pprj, OWL Files (.owl or .rdf))
File
Edit
Project
OWL
Code
Window
Help
protégé
OWLClasses
Properties
Forms
Individuals
Metadata
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(?x) → 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(?x, ?y) ∧ hasChild(?z, ?y) ∧ differentFrom(?x, ?z) → hasSibling(?x, ?z)
Def-hasSister	hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
Def-hasSon	hasChild(?x, ?y) ∧ Man(?x) → hasSon(?x, ?y)
Def-hasUncle	hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)



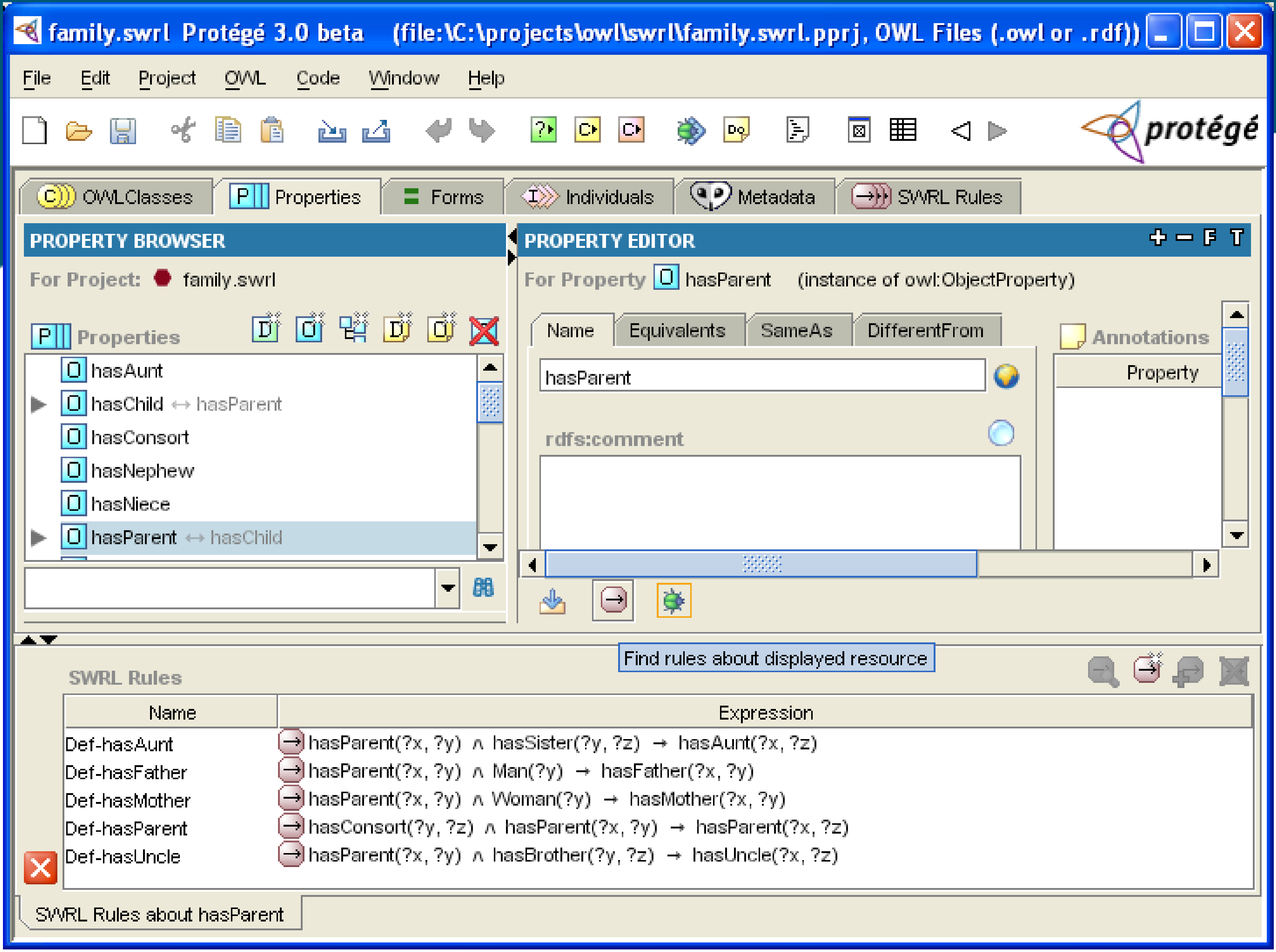
Def-hasUncle

rdfs:comment

A simple Rule to capture the definition of "uncle". Note that in contrast to pure OWL, SWRL provides mechanisms to represent variables, and therefore is quite rich.

Property	Value	Lang
D rdfs:comment	A simple Rule to captur...	
D rdfs:label	Onkeldefinition	de

```
hasParent(?x, ?y) ∧
hasBrother(?y, ?z)
 → hasUncle(?x, ?z)
```



For Project: ● family.swrl

For Property hasParent (instance of owl:ObjectProperty)

- hasAunt
- hasChild ↔ hasParent
- hasConsort
- hasNephew
- hasNiece
- hasParent ↔ hasChild

Name: hasParent

Name	Expression
Def-hasAunt	hasParent(?x, ?y) ∧ hasSister(?y, ?z) → hasAunt(?x, ?z)
Def-hasFather	hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
Def-hasMother	hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
Def-hasParent	hasConsort(?y, ?z) ∧ hasParent(?x, ?y) → hasParent(?x, ?z)
Def-hasUncle	hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)

What checking does the SWRL Editor do?

- Only syntactically valid rules can be saved.
- The SWRL editor will only allow saving of rules relating to currently loaded OWL entities.
- Basic semantic checking, e.g., no variables can be used in a rule consequent that were not referred to in the antecedent
- However, no elaborate sanity checking is performed, e.g., rule could contradict OWL constraints

How are SWRL Rules Saved?

- SWRL rules are saved as OWL individuals with their associated OWL file.
- Classes that describe this ontology are contained in SWRL Ontology:
  - http://www.daml.org/rules/proposal/swrl.orl
- These classes include:
  - swrl:Imp – represents a single SWRL rule
  - swrl:Atom – represents a single rule atom
  - swrl:AtomList – represent a list of atoms
- Other rule engines can use these rules, e.g., SweetRules.

Interacting with SWRL Rules in Protégé-OWL

- Via files – SWRL rules are stored in standard format.
- 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.
  - FAQ: http://protege.stanford.edu/plugins/owl/swrl/SWLFactory.html

Adding a Third Party Rule Engine

- SWRL Editor has been available as part of Protégé-OWL for a year.
- Is open source (like Protégé-OWL itself).
- Initially had no inference capabilities.
- We then integrated the Jess rule engine with Protégé-OWL to perform inference with SWRL rules.

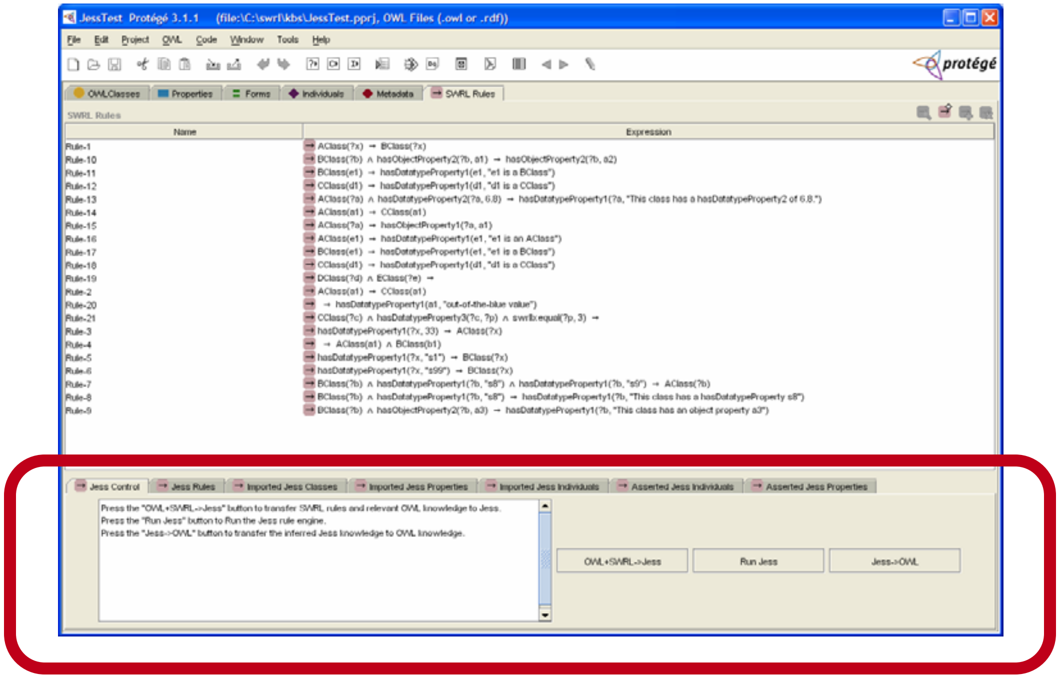
High-level Steps to Integrate Rule Engine with Protégé-OWL

- 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.

GUI Interaction with SWRL Rules in Protégé-OWL

Two choices for GUI interaction:

- Protégé-OWL plugin mechanism
- SWRL Editor plugin mechanism



JessTest Protégé 3.1.1 (file:\C:\swrl\kbs\JessTest.pprj, OWL Files (.owl or .rdf))
File Edit Project OWL Code Window Tools Help
protégé
OWLClasses Properties Forms Individuals Metadata SWRL Rules
SWRL Rules

Name	Expression
Rule-1	AClass(?x) → BClass(?x)
Rule-10	BClass(?b) ∧ hasObjectProperty2(?b, a1) → hasObjectProperty2(?b, a2)
Rule-11	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-12	CClass(d1) → hasDatatypeProperty1(d1, "d1 is a CClass")
Rule-13	AClass(?a) ∧ hasDatatypeProperty2(?a, 6.8) → hasDatatypeProperty1(?a, "This class has a hasDatatypeProperty2 of 6.8.")
Rule-14	AClass(a1) → CClass(a1)
Rule-15	AClass(?a) → hasObjectProperty1(?a, a1)
Rule-16	AClass(e1) → hasDatatypeProperty1(e1, "e1 is an AClass")
Rule-17	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-18	CClass(d1) → hasDatatypeProperty1(d1, "d1 is a CClass")
Rule-19	DClass(?d) ∧ EClass(?e) →
Rule-2	AClass(a1) → CClass(a1)
Rule-20	→ hasDatatypeProperty1(a1, "out-of-the-blue value")
Rule-21	CClass(?c) ∧ hasDatatypeProperty3(?c, ?p) ∧ swrlb:equal(?p, 3) →
Rule-3	hasDatatypeProperty1(?x, 33) → AClass(?x)
Rule-4	→ AClass(a1) ∧ BClass(b1)
Rule-5	hasDatatypeProperty1(?x, "s1") → BClass(?x)
Rule-6	hasDatatypeProperty1(?x, "s99") → BClass(?x)
Rule-7	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") ∧ hasDatatypeProperty1(?b, "s9") → AClass(?b)
Rule-8	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") → hasDatatypeProperty1(?b, "This class has a hasDatatypeProperty s8")
Rule-9	BClass(?b) ∧ hasObjectProperty2(?b, a3) → hasDatatypeProperty1(?b, "This class has an object property a3")

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.

Rule Engine Interaction with SWRL Rules in Protégé-OWL

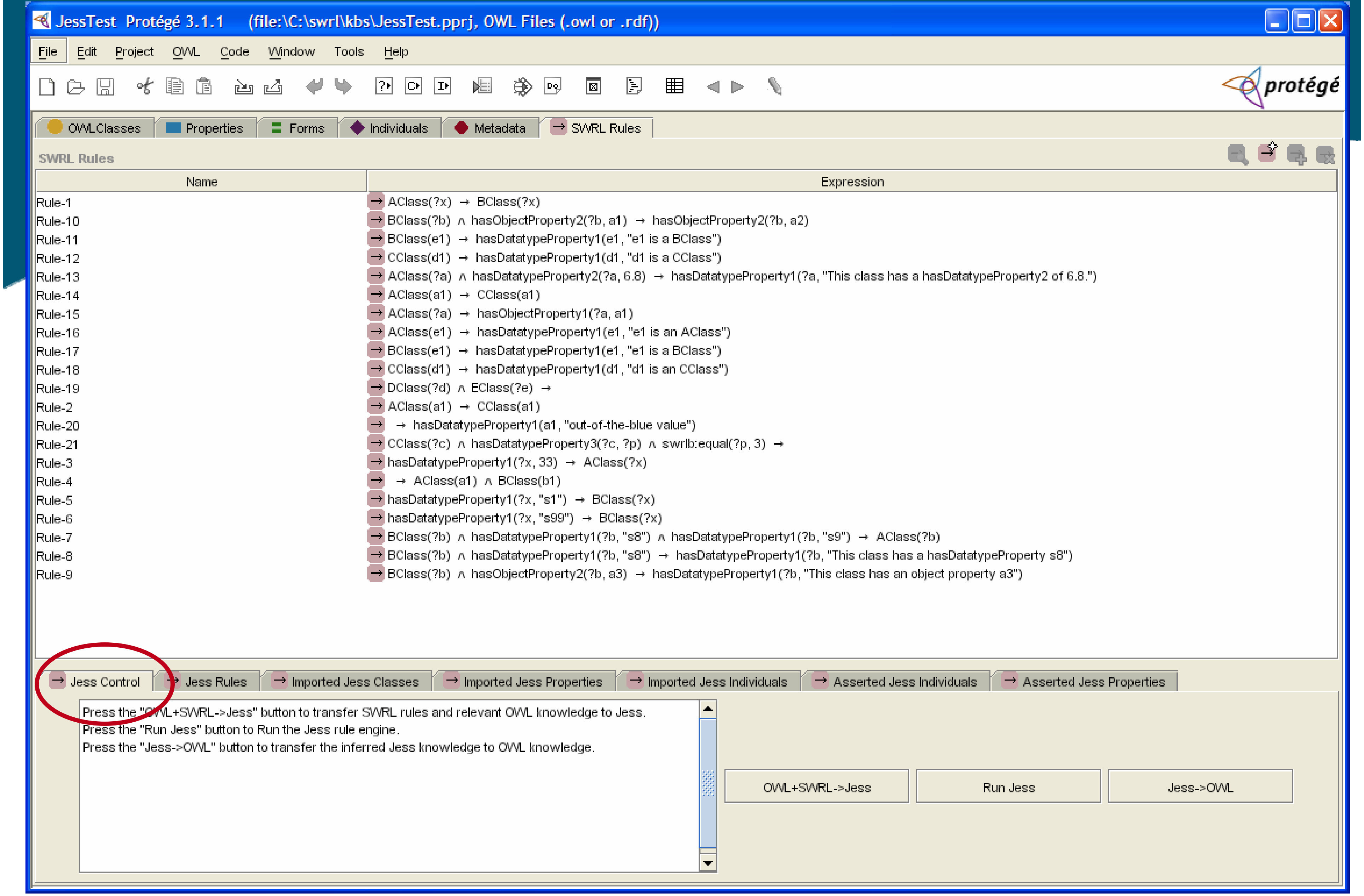
- Before mapping, extracting relevant OWL knowledge for inference is an important optimization.
- Not all knowledge needs to be extracted.
- Required knowledge can be determined from each rule.
- For example, the rule: Man(Fred) ^ Man(?y) ^ hasParent(Fred, ?y) ^ hasBrother(?y,?z) -> hasUncle(Fred, ?z) 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.

Protégé-OWL Provides a SWRL Bridge API

- 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 developers's responsibility.

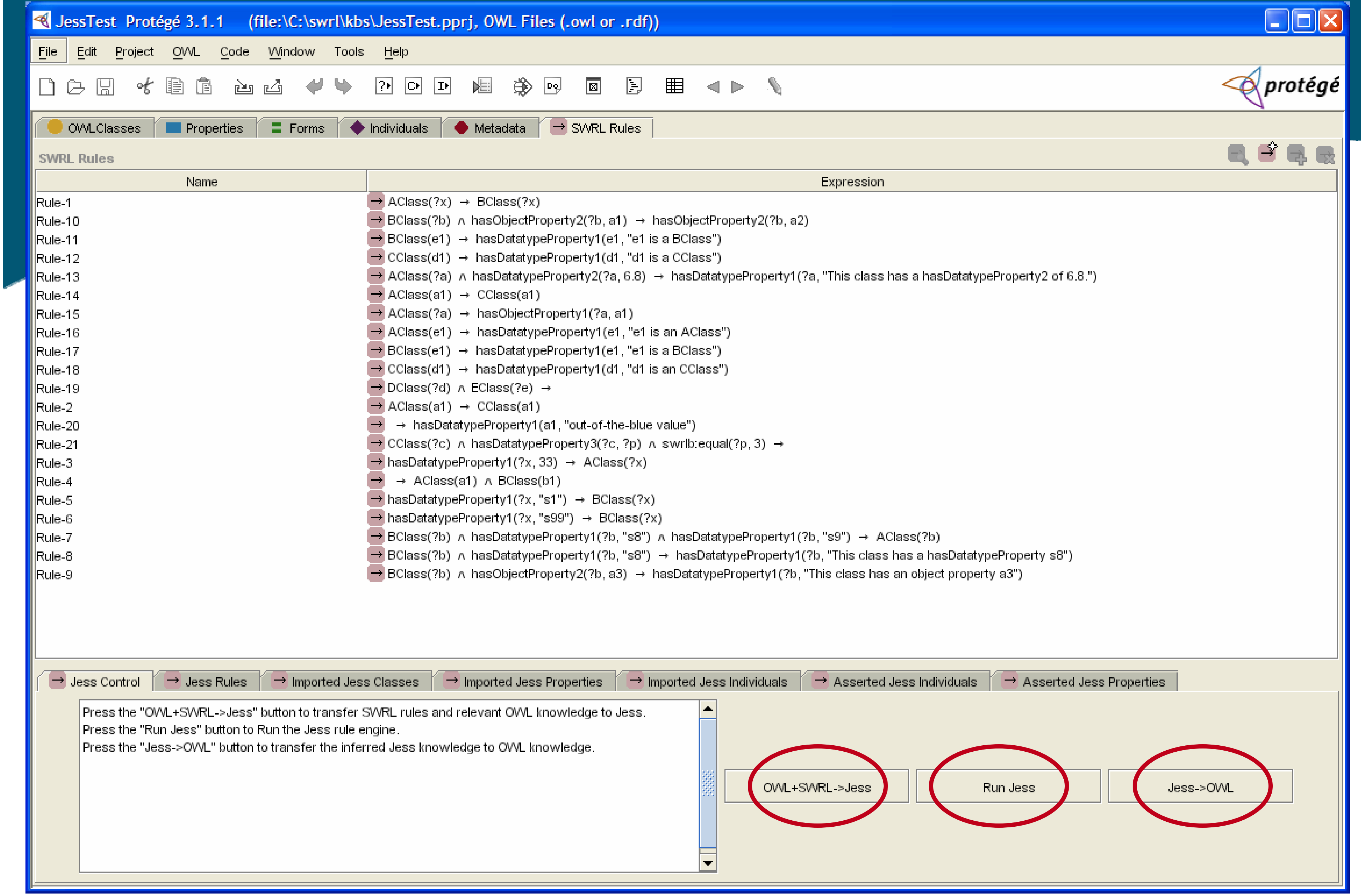
We used SWRL Bridge to Integrate Jess Rule Engine with Protégé-OWL

- 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., SWRLJessTab, SweetJess, JessTab.



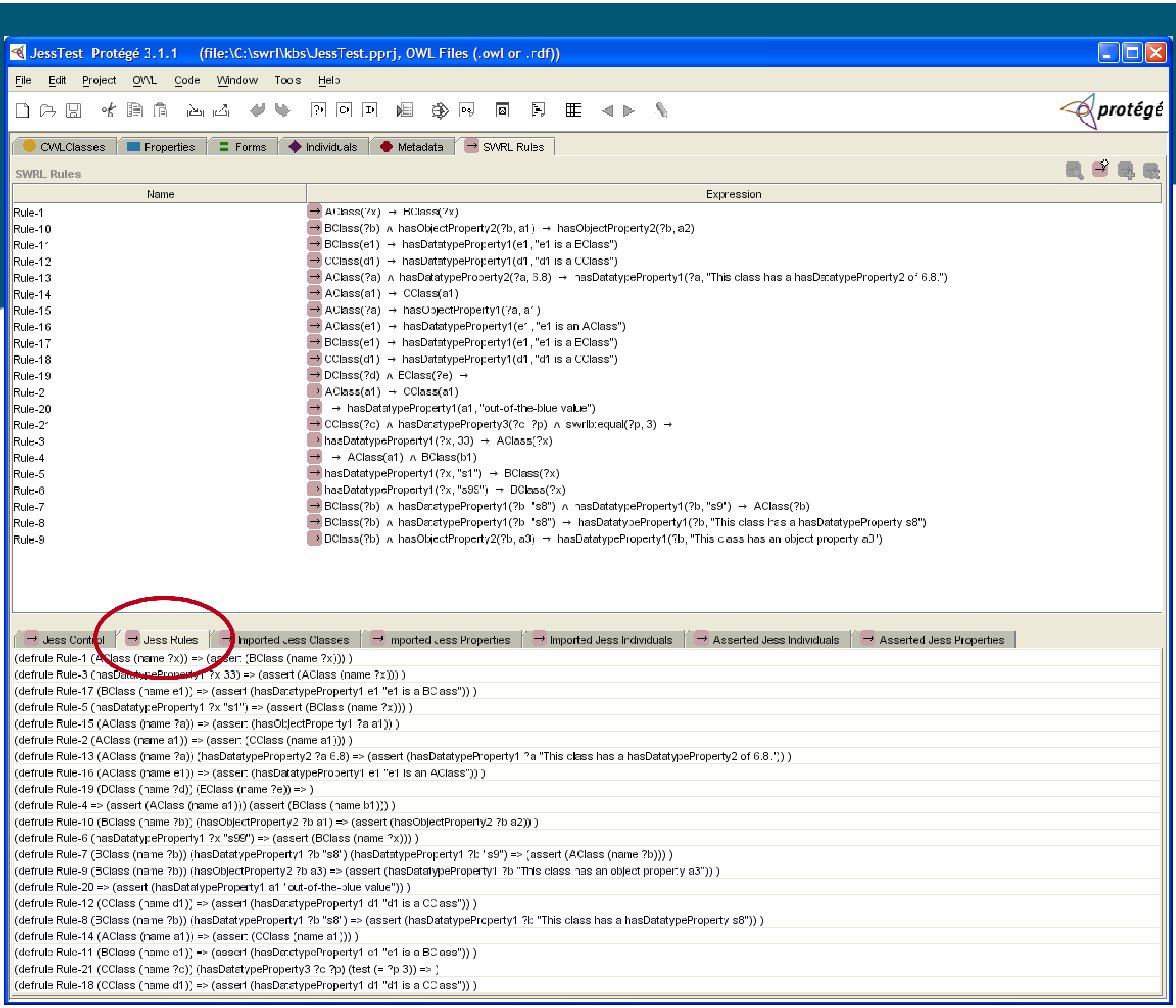
Name	Expression
Rule-1	AClass(?x) → BClass(?x)
Rule-10	BClass(?b) ∧ hasObjectProperty2(?b, a1) → hasObjectProperty2(?b, a2)
Rule-11	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-12	CClass(d1) → hasDatatypeProperty1(d1, "d1 is a CClass")
Rule-13	AClass(?a) ∧ hasDatatypeProperty2(?a, 6.8) → hasDatatypeProperty1(?a, "This class has a hasDatatypeProperty2 of 6.8.")
Rule-14	AClass(a1) → CClass(a1)
Rule-15	AClass(?a) → hasObjectProperty1(?a, a1)
Rule-16	AClass(e1) → hasDatatypeProperty1(e1, "e1 is an AClass")
Rule-17	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-18	CClass(d1) → hasDatatypeProperty1(d1, "d1 is an CClass")
Rule-19	DClass(?d) ∧ EClass(?e) →
Rule-2	AClass(a1) → CClass(a1)
Rule-20	→ hasDatatypeProperty1(a1, "out-of-the-blue value")
Rule-21	CClass(?c) ∧ hasDatatypeProperty3(?c, ?p) ∧ swrlb:equal(?p, 3) →
Rule-3	hasDatatypeProperty1(?x, 33) → AClass(?x)
Rule-4	→ AClass(a1) ∧ BClass(b1)
Rule-5	hasDatatypeProperty1(?x, "s1") → BClass(?x)
Rule-6	hasDatatypeProperty1(?x, "s99") → BClass(?x)
Rule-7	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") ∧ hasDatatypeProperty1(?b, "s9") → AClass(?b)
Rule-8	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") → hasDatatypeProperty1(?b, "This class has a hasDatatypeProperty s8")
Rule-9	BClass(?b) ∧ hasObjectProperty2(?b, a3) → hasDatatypeProperty1(?b, "This class has an object property a3")

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.



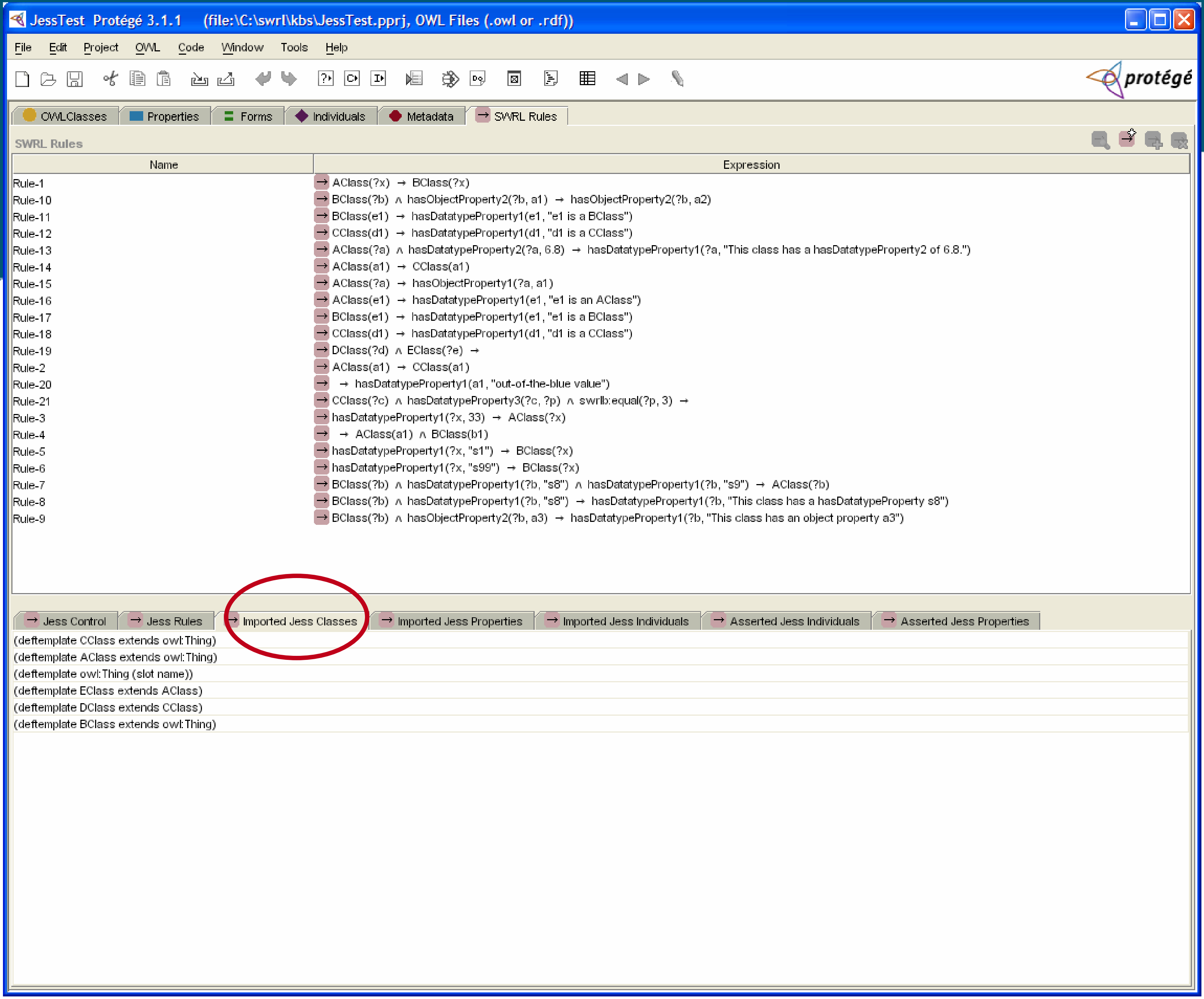
Name	Expression
Rule-1	AClass(?x) → BClass(?x)
Rule-10	BClass(?b) ∧ hasObjectProperty2(?b, a1) → hasObjectProperty2(?b, a2)
Rule-11	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-12	CClass(d1) → hasDatatypeProperty1(d1, "d1 is a CClass")
Rule-13	AClass(?a) ∧ hasDatatypeProperty2(?a, 6.8) → hasDatatypeProperty1(?a, "This class has a hasDatatypeProperty2 of 6.8.")
Rule-14	AClass(a1) → CClass(a1)
Rule-15	AClass(?a) → hasObjectProperty1(?a, a1)
Rule-16	AClass(e1) → hasDatatypeProperty1(e1, "e1 is an AClass")
Rule-17	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-18	CClass(d1) → hasDatatypeProperty1(d1, "d1 is an CClass")
Rule-19	DClass(?d) ∧ EClass(?e) →
Rule-2	AClass(a1) → CClass(a1)
Rule-20	→ hasDatatypeProperty1(a1, "out-of-the-blue value")
Rule-21	CClass(?c) ∧ hasDatatypeProperty3(?c, ?p) ∧ swrlb:equal(?p, 3) →
Rule-3	hasDatatypeProperty1(?x, 33) → AClass(?x)
Rule-4	→ AClass(a1) ∧ BClass(b1)
Rule-5	hasDatatypeProperty1(?x, "s1") → BClass(?x)
Rule-6	hasDatatypeProperty1(?x, "s99") → BClass(?x)
Rule-7	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") ∧ hasDatatypeProperty1(?b, "s9") → AClass(?b)
Rule-8	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") → hasDatatypeProperty1(?b, "This class has a hasDatatypeProperty s8")
Rule-9	BClass(?b) ∧ hasObjectProperty2(?b, a3) → hasDatatypeProperty1(?b, "This class has an object property a3")

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.



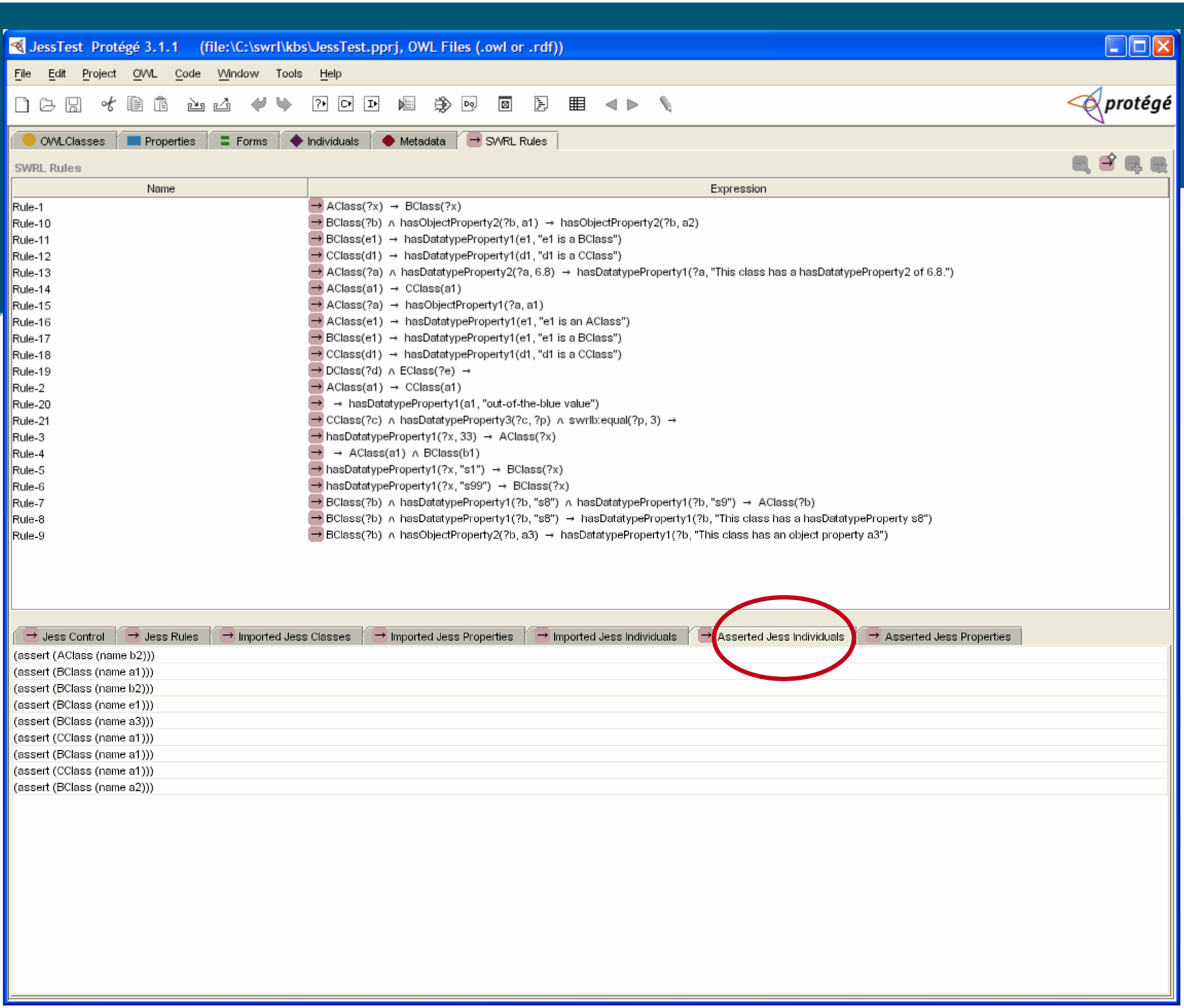
Name	Expression
Rule-1	AClass(?x) → BClass(?x)
Rule-10	BClass(?b) ∧ hasObjectProperty2(?b, a1) → hasObjectProperty2(?b, a2)
Rule-11	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-12	CClass(d1) → hasDatatypeProperty1(d1, "d1 is a CClass")
Rule-13	AClass(?a) ∧ hasDatatypeProperty2(?a, 6.8) → hasDatatypeProperty1(?a, "This class has a hasDatatypeProperty2 of 6.8.")
Rule-14	AClass(a1) → CClass(a1)
Rule-15	AClass(?a) → hasObjectProperty1(?a, a1)
Rule-16	AClass(e1) → hasDatatypeProperty1(e1, "e1 is an AClass")
Rule-17	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-18	CClass(d1) → hasDatatypeProperty1(d1, "d1 is a CClass")
Rule-19	DClass(?d) ∧ EClass(?e) →
Rule-2	AClass(a1) → CClass(a1)
Rule-20	→ hasDatatypeProperty1(a1, "out-of-the-blue value")
Rule-21	CClass(?c) ∧ hasDatatypeProperty3(?c, ?p) ∧ swrlb:equal(?p, 3) →
Rule-3	hasDatatypeProperty1(?x, 33) → AClass(?x)
Rule-4	→ AClass(a1) ∧ BClass(b1)
Rule-5	hasDatatypeProperty1(?x, "s1") → BClass(?x)
Rule-6	hasDatatypeProperty1(?x, "s99") → BClass(?x)
Rule-7	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") ∧ hasDatatypeProperty1(?b, "s9") → AClass(?b)
Rule-8	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") → hasDatatypeProperty1(?b, "This class has a hasDatatypeProperty s8")
Rule-9	BClass(?b) ∧ hasObjectProperty2(?b, a3) → hasDatatypeProperty1(?b, "This class has an object property a3")

```
(defrule Rule-1 (AClass (name ?x)) => (assert (BClass (name ?x))) )
(defrule Rule-3 (hasDatatypeProperty1 ?x 33) => (assert (AClass (name ?x))) )
(defrule Rule-17 (BClass (name e1)) => (assert (hasDatatypeProperty1 e1 "e1 is a BClass")) )
(defrule Rule-5 (hasDatatypeProperty1 ?x "s1") => (assert (BClass (name ?x))) )
(defrule Rule-15 (AClass (name ?a)) => (assert (hasObjectProperty1 ?a a1)) )
(defrule Rule-2 (AClass (name a1)) => (assert (CClass (name a1))) )
(defrule Rule-13 (AClass (name ?a)) (hasDatatypeProperty2 ?a 6.8) => (assert (hasDatatypeProperty1 ?a "This class has a hasDatatypeProperty2 of 6.8.")) )
(defrule Rule-16 (AClass (name e1)) => (assert (hasDatatypeProperty1 e1 "e1 is an AClass")) )
(defrule Rule-19 (DClass (name ?d)) (EClass (name ?e)) => )
(defrule Rule-4 => (assert (AClass (name a1))) (assert (BClass (name b1))) )
(defrule Rule-10 (BClass (name ?b)) (hasObjectProperty2 ?b a1) => (assert (hasObjectProperty2 ?b a2)) )
(defrule Rule-6 (hasDatatypeProperty1 ?x "s99") => (assert (BClass (name ?x))) )
(defrule Rule-7 (BClass (name ?b)) (hasDatatypeProperty1 ?b "s8") (hasDatatypeProperty1 ?b "s9") => (assert (AClass (name ?b))) )
(defrule Rule-9 (BClass (name ?b)) (hasObjectProperty2 ?b a3) => (assert (hasDatatypeProperty1 ?b "This class has an object property a3")) )
(defrule Rule-20 => (assert (hasDatatypeProperty1 a1 "out-of-the-blue value")) )
(defrule Rule-12 (CClass (name d1)) => (assert (hasDatatypeProperty1 d1 "d1 is a CClass")) )
(defrule Rule-8 (BClass (name ?b)) (hasDatatypeProperty1 ?b "s8") => (assert (hasDatatypeProperty1 ?b "This class has a hasDatatypeProperty s8")) )
(defrule Rule-14 (AClass (name a1)) => (assert (CClass (name a1))) )
(defrule Rule-11 (BClass (name e1)) => (assert (hasDatatypeProperty1 e1 "e1 is a BClass")) )
(defrule Rule-21 (CClass (name ?c)) (hasDatatypeProperty3 ?c ?p) (test (= ?p 3)) => )
(defrule Rule-18 (CClass (name d1)) => (assert (hasDatatypeProperty1 d1 "d1 is a CClass")) )
```



Name	Expression
Rule-1	AClass(?x) → BClass(?x)
Rule-10	BClass(?b) ∧ hasObjectProperty2(?b, a1) → hasObjectProperty2(?b, a2)
Rule-11	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-12	CClass(d1) → hasDatatypeProperty1(d1, "d1 is a CClass")
Rule-13	AClass(?a) ∧ hasDatatypeProperty2(?a, 6.8) → hasDatatypeProperty1(?a, "This class has a hasDatatypeProperty2 of 6.8.")
Rule-14	AClass(a1) → CClass(a1)
Rule-15	AClass(?a) → hasObjectProperty1(?a, a1)
Rule-16	AClass(e1) → hasDatatypeProperty1(e1, "e1 is an AClass")
Rule-17	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-18	CClass(d1) → hasDatatypeProperty1(d1, "d1 is a CClass")
Rule-19	DClass(?d) ∧ EClass(?e) →
Rule-2	AClass(a1) → CClass(a1)
Rule-20	→ hasDatatypeProperty1(a1, "out-of-the-blue value")
Rule-21	CClass(?c) ∧ hasDatatypeProperty3(?c, ?p) ∧ swrlb:equal(?p, 3) →
Rule-3	hasDatatypeProperty1(?x, 33) → AClass(?x)
Rule-4	→ AClass(a1) ∧ BClass(b1)
Rule-5	hasDatatypeProperty1(?x, "s1") → BClass(?x)
Rule-6	hasDatatypeProperty1(?x, "s99") → BClass(?x)
Rule-7	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") ∧ hasDatatypeProperty1(?b, "s9") → AClass(?b)
Rule-8	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") → hasDatatypeProperty1(?b, "This class has a hasDatatypeProperty s8")
Rule-9	BClass(?b) ∧ hasObjectProperty2(?b, a3) → hasDatatypeProperty1(?b, "This class has an object property a3")

```
(deftemplate CClass extends owl:Thing)
(deftemplate AClass extends owl:Thing)
(deftemplate owl:Thing (slot name))
(deftemplate EClass extends AClass)
(deftemplate DClass extends CClass)
(deftemplate BClass extends owl:Thing)
```



Name	Expression
Rule-1	AClass(?x) → BClass(?x)
Rule-10	BClass(?b) ∧ hasObjectProperty2(?b, a1) → hasObjectProperty2(?b, a2)
Rule-11	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-12	CClass(d1) → hasDatatypeProperty1(d1, "d1 is a CClass")
Rule-13	AClass(?a) ∧ hasDatatypeProperty2(?a, 6.8) → hasDatatypeProperty1(?a, "This class has a hasDatatypeProperty2 of 6.8.")
Rule-14	AClass(a1) → CClass(a1)
Rule-15	AClass(?a) → hasObjectProperty1(?a, a1)
Rule-16	AClass(e1) → hasDatatypeProperty1(e1, "e1 is an AClass")
Rule-17	BClass(e1) → hasDatatypeProperty1(e1, "e1 is a BClass")
Rule-18	CClass(d1) → hasDatatypeProperty1(d1, "d1 is a CClass")
Rule-19	DClass(?d) ∧ EClass(?e) →
Rule-2	AClass(a1) → CClass(a1)
Rule-20	→ hasDatatypeProperty1(a1, "out-of-the-blue value")
Rule-21	CClass(?c) ∧ hasDatatypeProperty3(?c, ?p) ∧ swrlb:equal(?p, 3) →
Rule-3	hasDatatypeProperty1(?x, 33) → AClass(?x)
Rule-4	→ AClass(a1) ∧ BClass(b1)
Rule-5	hasDatatypeProperty1(?x, "s1") → BClass(?x)
Rule-6	hasDatatypeProperty1(?x, "s99") → BClass(?x)
Rule-7	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") ∧ hasDatatypeProperty1(?b, "s9") → AClass(?b)
Rule-8	BClass(?b) ∧ hasDatatypeProperty1(?b, "s8") → hasDatatypeProperty1(?b, "This class has a hasDatatypeProperty s8")
Rule-9	BClass(?b) ∧ hasObjectProperty2(?b, a3) → hasDatatypeProperty1(?b, "This class has an object property a3")

```
(assert (AClass (name b2)))
(assert (BClass (name a1)))
(assert (BClass (name b2)))
(assert (BClass (name e1)))
(assert (BClass (name a3)))
(assert (CClass (name a1)))
(assert (BClass (name a1)))
(assert (CClass (name a1)))
(assert (BClass (name a2)))
```

Outstanding Issues

- Only named classes can be used in SWRL rules.
- SWRL Bridge does not know about all OWL constraints.
  - Contradictions with rules possible!
  - Consistency must be assured by the user.
  - Hard problem to solve in general.

Conclusion: Developers Needed!

- SWRL Editor is open source.
- Well documented. Several FAQs:
  - http://protege.stanford.edu/plugins/owl/swrl/
  - http://protege.stanford.edu/plugins/owl/swrl/SWRLFactory.html
- Support from Protégé-OWL mailing list.
- Protégé-OWL could be used to implement other OWL-based rule languages.