SWRL
Semantic Web Rule Language

Susana R. Novoa
UNIK4710

Overview

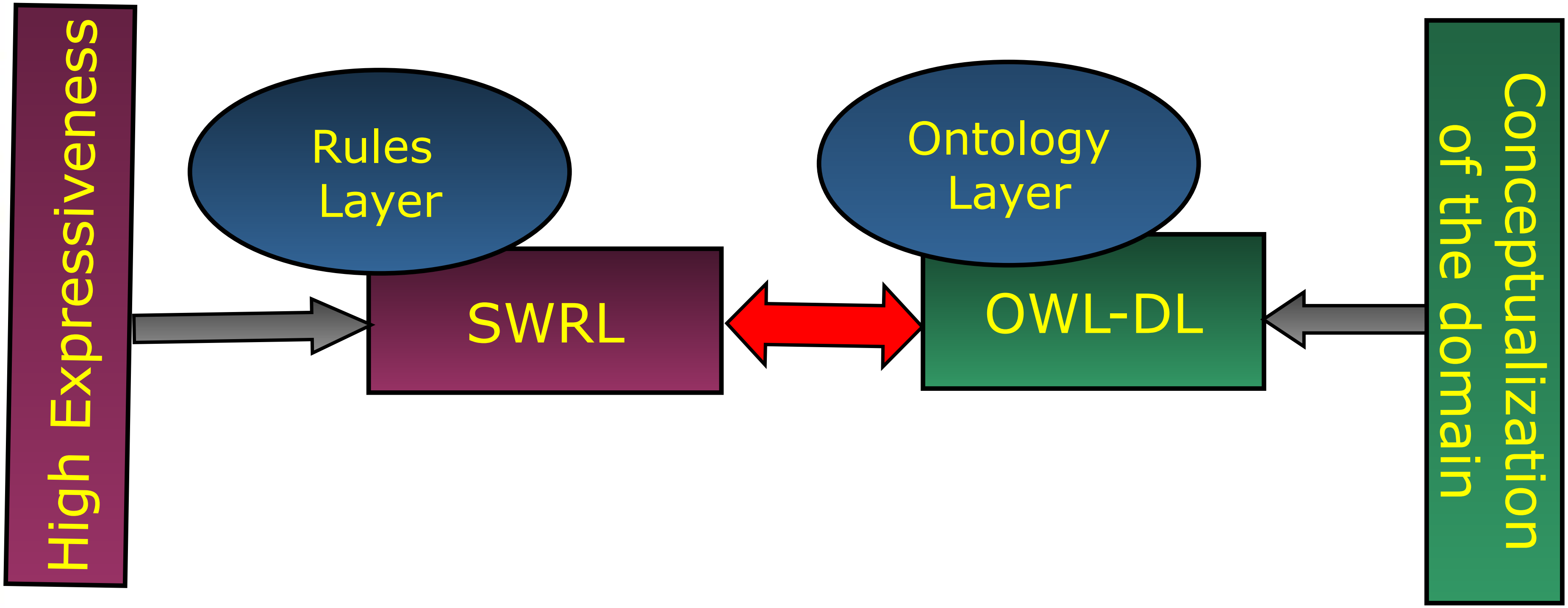
- What is SWRL?
- What is Jess?
    - Installing Jess
    - Creating rules
- SWRL Rules
    - Atom Types
    - SWRLTab
- SQWRL
    - SQWRLTab
- SWRL Resources

What is SWRL?

- SWRL is an acronym for Semantic Web Rule Language.
- SWRL is intended to be the rule language of the Semantic Web.
- All rules are expressed in terms of OWL concepts (classes, properties, individuals).

What is SWRL?

- Ontology languages do not offer the expressiveness we want → Rules do it well.



What is Jess?

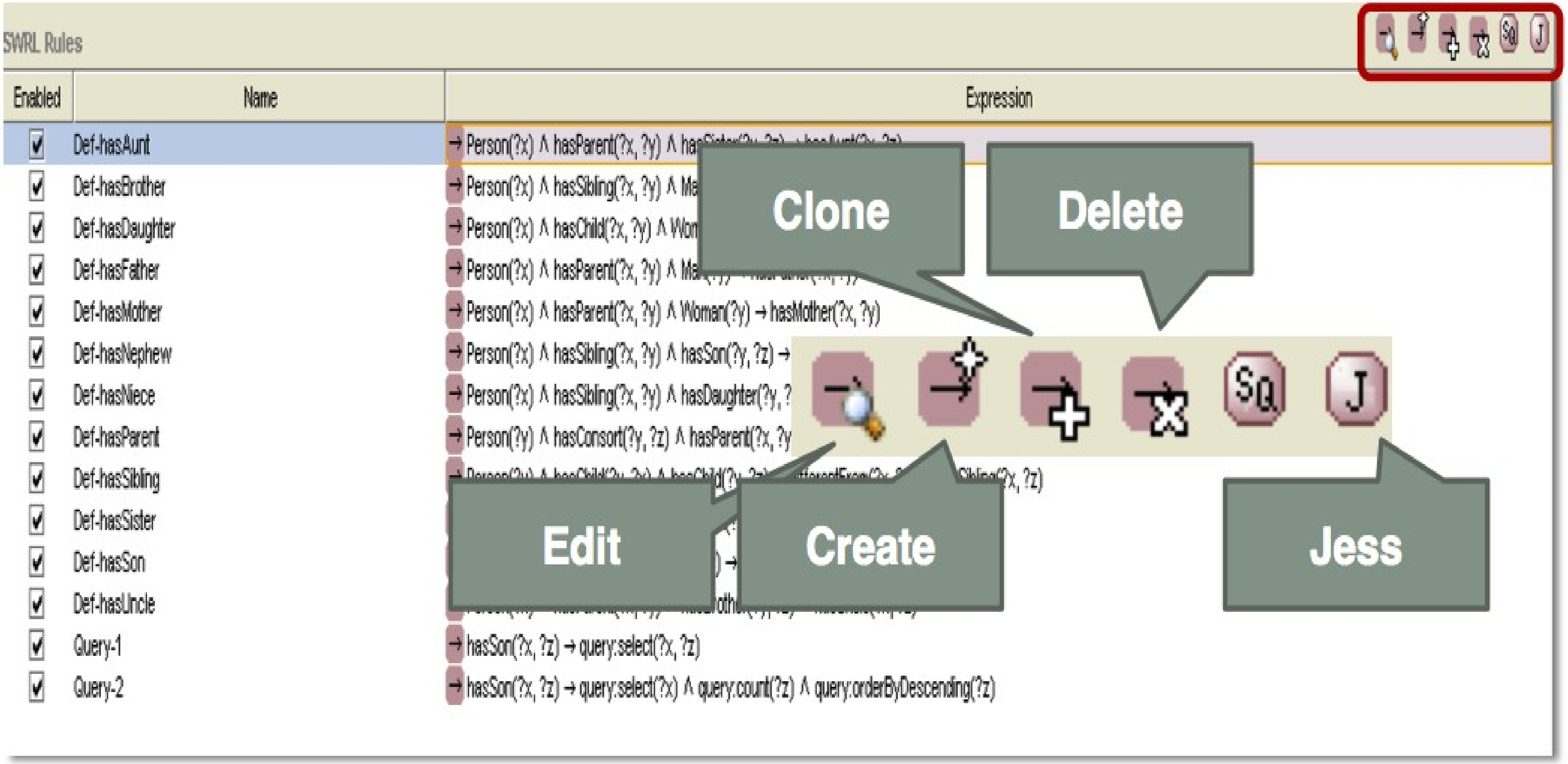
- Jess system consists of a rule base, fact base, and an execution engine.
- Available free to academic users, for a small fee to non-academic users.
- Has been used in Protégé-based tools, e.g., SWRLJessTab, SweetJess, JessTab.

Install Jess

- JESS Download: http://herzberg.ca.sandia.gov/

- SWRL Tab Activation:
  - Unzip Jess70p2.zip
  - Copy Jess70p2\Jess70p2\lib\jess.jar to
  - [Protégé install Folder]/plugins/edu.stanford.smi.protegex.owl/

Creating Rules



SWRL Rules

Enabled	Name	Expression
☑	Def-hasAunt	→ Person(?x) ∧ hasParent(?x, ?y) ∧ ha
☑	Def-hasBrother	→ Person(?x) ∧ hasSibling(?x, ?y) ∧ Ma
☑	Def-hasDaughter	→ Person(?x) ∧ hasChild(?x, ?y) ∧ Wom
☑	Def-hasFather	→ Person(?x) ∧ hasParent(?x, ?y) ∧ Ma
☑	Def-hasMother	→ Person(?x) ∧ hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
☑	Def-hasNephew	→ Person(?x) ∧ hasSibling(?x, ?y) ∧ hasSon(?y, ?z) →
☑	Def-hasNiece	→ Person(?x) ∧ hasSibling(?x, ?y) ∧ hasDaughter(?y, ?
☑	Def-hasParent	→ Person(?y) ∧ hasConsort(?y, ?z) ∧ hasParent(?x, ?y
☑	Def-hasSibling	→ Sibling(?x, ?z)
☑	Def-hasSister	→
☑	Def-hasSon	→
☑	Def-hasUncle	→
☑	Query-1	→ hasSon(?x, ?z) → query:select(?x, ?z)
☑	Query-2	→ hasSon(?x, ?z) → query:select(?x) ∧ query:count(?z) ∧ query:orderByDescending(?z)

Clone

Delete

Edit

Create

Jess

SWRL Rule

- Contains an antecedent part(*body)*, and a consequent (*head)*.

- The body and head consist of positive conjunctions of *atoms*:

Atom ^ Atom …   →   Atom ^ Atom ….

SWRL Rule

An atom is an expression of the form: ***P(arg1 arg2,...)***

- **P** is a predicate symbol (classes, properties...)
- Arguments of the expression: ***arg1, arg2,…*** (individuals, data values or variables)

Example SWRL Rule:

$$\underbrace{\textbf{Person(?p) ^ hasSibling(?p,?s) ^ Man(?s)}}_{\textbf{antecedent}} \rightarrow \underbrace{\textbf{hasBrother(?p,?s)}}_{\textbf{consequent}}$$

Atom Types

SWRL provides seven types of atoms:

- Class Atoms ***owl:Class***
- Individual Property atoms ***owl:ObjectProperty***
- Data Valued Property atoms ***owl:DatatypeProperty***
- Different Individuals atoms
- Same Individual atoms
- Built-in atoms

Class Atom

- Consists of an **OWL named class** or **class expression** and a single argument representing an OWL individual:

Person(?p)

Person (Fred)

- **Person** - OWL named class
- **?p** - variable representing an OWL individual
- **Fred** - name of an OWL individual.

Class Atom Example

- All individual of type Man are also a type of Person:

Man(?p) -> Person(?p)

- Of course, this statement can also be made directly in OWL.

Individual Property Atom

- Consists of an **OWL object property** and two arguments representing OWL individuals:

hasBrother(?x, ?y)

hasSibling(Fred, ?y)

- **hasBrother, hasSibling** - OWL object properties
- **?x and ?y** - variables representing OWL individuals
- **Fred** -name of an OWL individual.

Individual Property Atom Example

- Person with a male sibling has a brother:

Person(?p) ^ hasSibling(?p,?s) ^ Man(?s) -> hasBrother(?p,?s)

- Person and male can be mapped to OWL class called Person with a subclass Man
- The sibling and brother relationships can be expressed using OWL object properties hasSibling and hasBrother with a domain and range of Person.

Data Valued Property Atom

- A data valued property atom consists of an **OWL data property** and two arguments (OWL individual , data value)

hasAge(?x, ?age)

hasHeight(Fred, ?h)

hasAge(?x, 232)

Data Valued Property Atom Example

- All persons that own a car should be classified as drivers

Person(?p) ^ hasCar(?p, true) -> Driver(?p)

- ➔ This rule classifies all car-owner individuals of type Person to also be members of the class Driver.

- Named individuals can be referred directly:

Person(Fred) ^ hasCar(Fred, true) -> Driver(Fred)

- ➔ This rule works with a known individual called Fred in an ontology, and new individual can not be created using this rule.

Different & Same Individuals Atom

- SWRL supports sameAs and differentFrom atoms to determine if individuals refer to the same underlying individual or are distinct, and can use **owl:sameAs**, **owl:allDifferents:**

differentFrom(?x, ?y)

differentFrom(Fred, Joe)

sameAs(?x, ?y)

sameAs(Fred, Freddy)

Different & Same Individuals Atom Example

- If two OWL individuals of type Author cooperate on the same publication that they are collaborators:

Publication(?a) ^ hasAuthor(?x, ?y) ^ hasAuthor(?x, ?z) ^ differentFrom(?y, ?z) -> cooperatedWith(?y, ?z)

Built-In Atom

- A built-in is a predicate that takes one or more arguments and evaluates to true if the arguments satisfy the predicate.
- Core SWRL built-ins are preceded by the namespace qualifier **swrlb**.
- SWRL allows new libraries of built-ins to be defined and used in rules.

Built-In Atom Example

- Person with an age of greater than 17 is an adult:

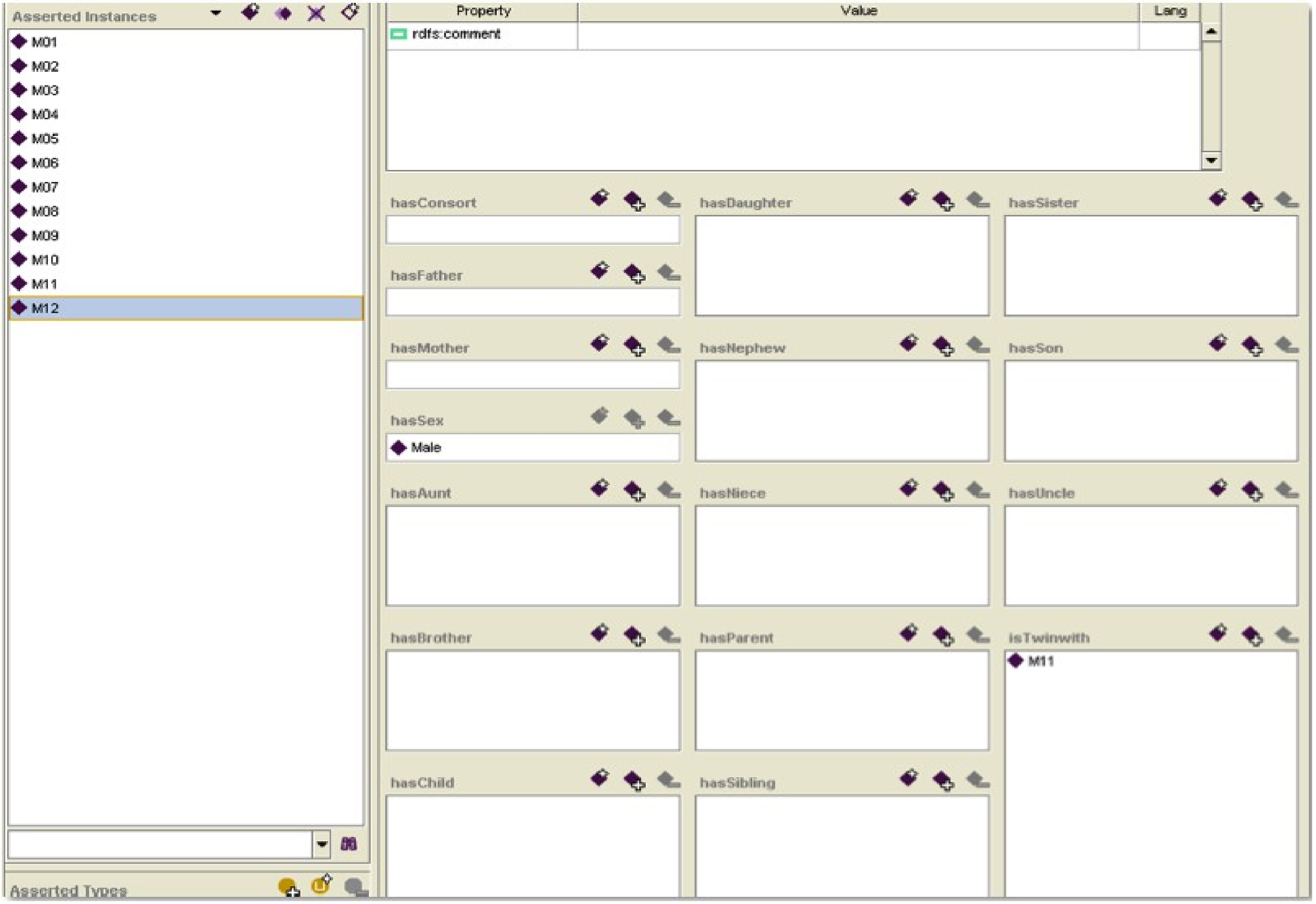
  Person(?p) ^ hasAge(?p, ?age) ^ swrlb:greaterThan(?age, 17) -> Adult(?p)

- Person's telephone number starts with the international access code "+":

  Person(?p)^hasNumber(?p, ?number) ^ swrlb:startsWith(?number, "+") -> hasInternationalNumber(?p,true)

SWRLTab: Displaying Results

Before Jess Reasoning:



Asserted Instances

- M01
- M02
- M03
- M04
- M05
- M06
- M07
- M08
- M09
- M10
- M11
- M12

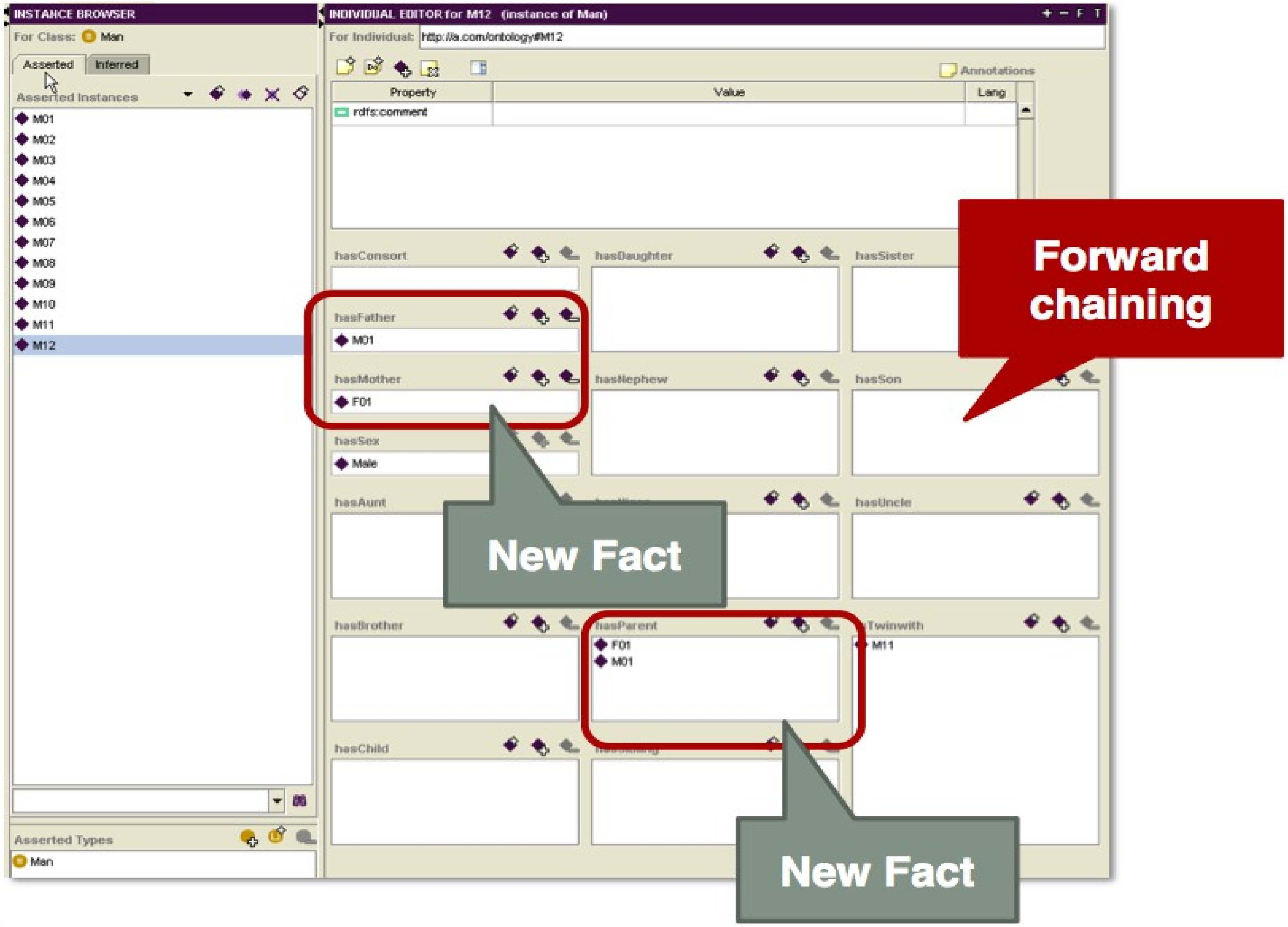
Property	Value	Lang
rdfs:comment		

hasSex: Male

isTwinwith: M11

SWRLTab: Displaying Results

- After Jess Reasoning



INSTANCE BROWSER

For Class: Man

Asserted Instances: M01, M02, M03, M04, M05, M06, M07, M08, M09, M10, M11, M12

Asserted Types: Man

INDIVIDUAL EDITOR for M12 (instance of Man)

For Individual: http://a.com/ontology#M12

Property	Value	Lang
rdfs:comment		

hasFather: M01

hasMother: F01

hasSex: Male

hasParent: F01
M01

TwinWith: M11

Forward chaining

New Fact

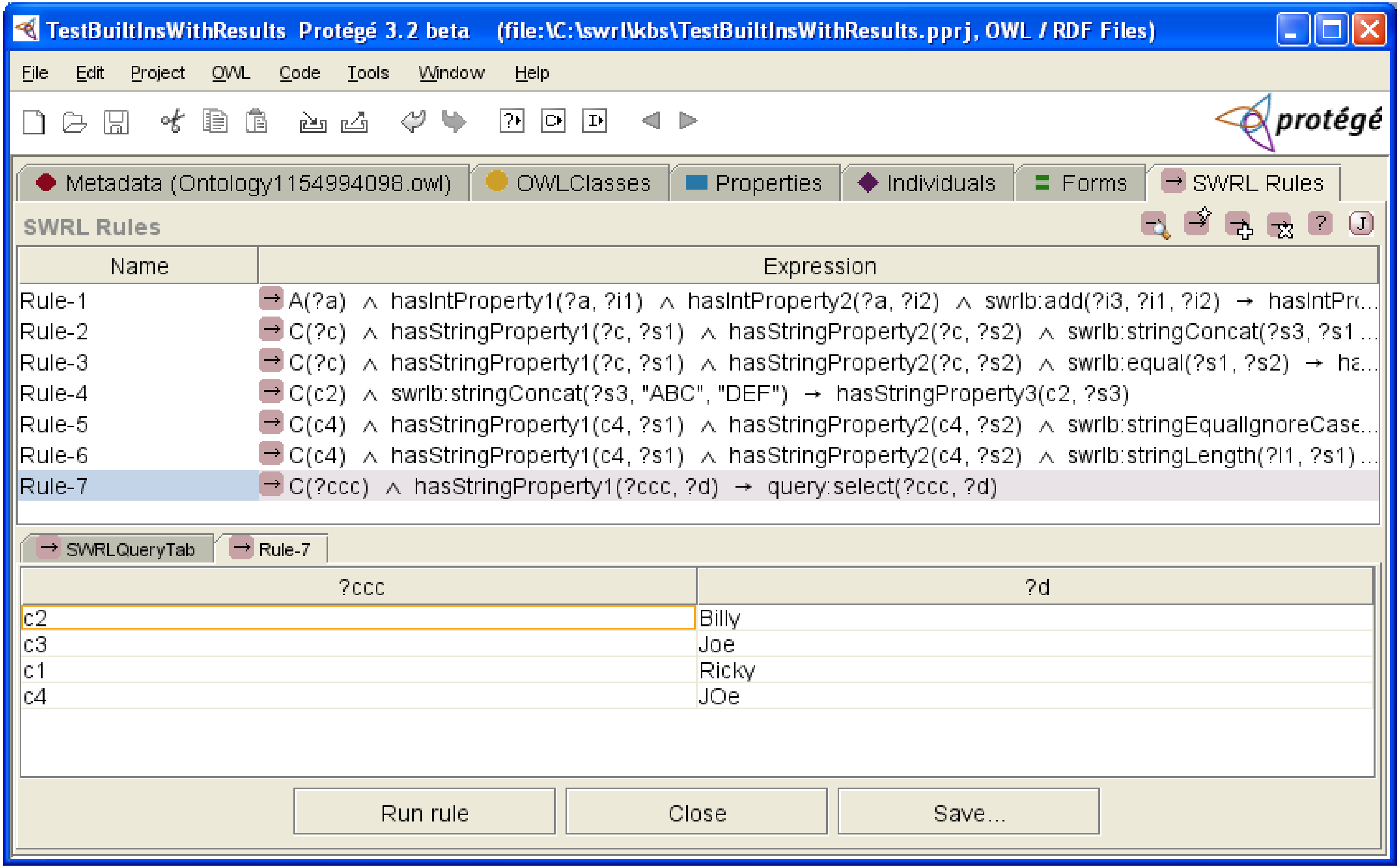
New Fact

SQWRL

- A rule antecedent can be viewed as a pattern matching specification, i.e., a query
- With built-ins, language compliant query extensions are possible.

Person(?p) ^ hasAge(?p, ?age) ^ swrlb:greaterThan(?age, 17) -> swrlq:select(?p) ^ swrlq:orderBy(?age)

SWRLQueryTab: Displaying Results


TestBuiltInsWithResults Protégé 3.2 beta (file:\C:\swrl\kbs\TestBuiltInsWithResults.pprj, OWL / RDF Files)
File
Edit
Project
OWL
Code
Tools
Window
Help
protégé
Metadata (Ontology1154994098.owl)
OWLClasses
Properties
Individuals
Forms
SWRL Rules
SWRL Rules
SWRLQueryTab
Rule-7
Run rule
Close
Save...

Name	Expression
Rule-1	A(?a) ∧ hasIntProperty1(?a, ?i1) ∧ hasIntProperty2(?a, ?i2) ∧ swrlb:add(?i3, ?i1, ?i2) → hasIntPr...
Rule-2	C(?c) ∧ hasStringProperty1(?c, ?s1) ∧ hasStringProperty2(?c, ?s2) ∧ swrlb:stringConcat(?s3, ?s1 ...
Rule-3	C(?c) ∧ hasStringProperty1(?c, ?s1) ∧ hasStringProperty2(?c, ?s2) ∧ swrlb:equal(?s1, ?s2) → ha...
Rule-4	C(c2) ∧ swrlb:stringConcat(?s3, "ABC", "DEF") → hasStringProperty3(c2, ?s3)
Rule-5	C(c4) ∧ hasStringProperty1(c4, ?s1) ∧ hasStringProperty2(c4, ?s2) ∧ swrlb:stringEqualIgnoreCase...
Rule-6	C(c4) ∧ hasStringProperty1(c4, ?s1) ∧ hasStringProperty2(c4, ?s2) ∧ swrlb:stringLength(?l1, ?s1) ...
Rule-7	C(?ccc) ∧ hasStringProperty1(?ccc, ?d) → query:select(?ccc, ?d)

?ccc	?d
c2	Billy
c3	Joe
c1	Ricky
c4	JOe

SWRL Resources

- SWRL Language:
  - Specification: http://www.daml.org/2003/11/swrl/
- SWRL Tab:
  - http://protege.stanford.edu/plugins/owl/swrl/index.html
- SWRL API:
  - http://protege.stanford.edu/plugins/owl/swrl/SWRLFactory.html