

# **SWRL**

## **Semantic Web Rule Language**

*Susana R. Novoa*  
*UNIK4710*

# Overview

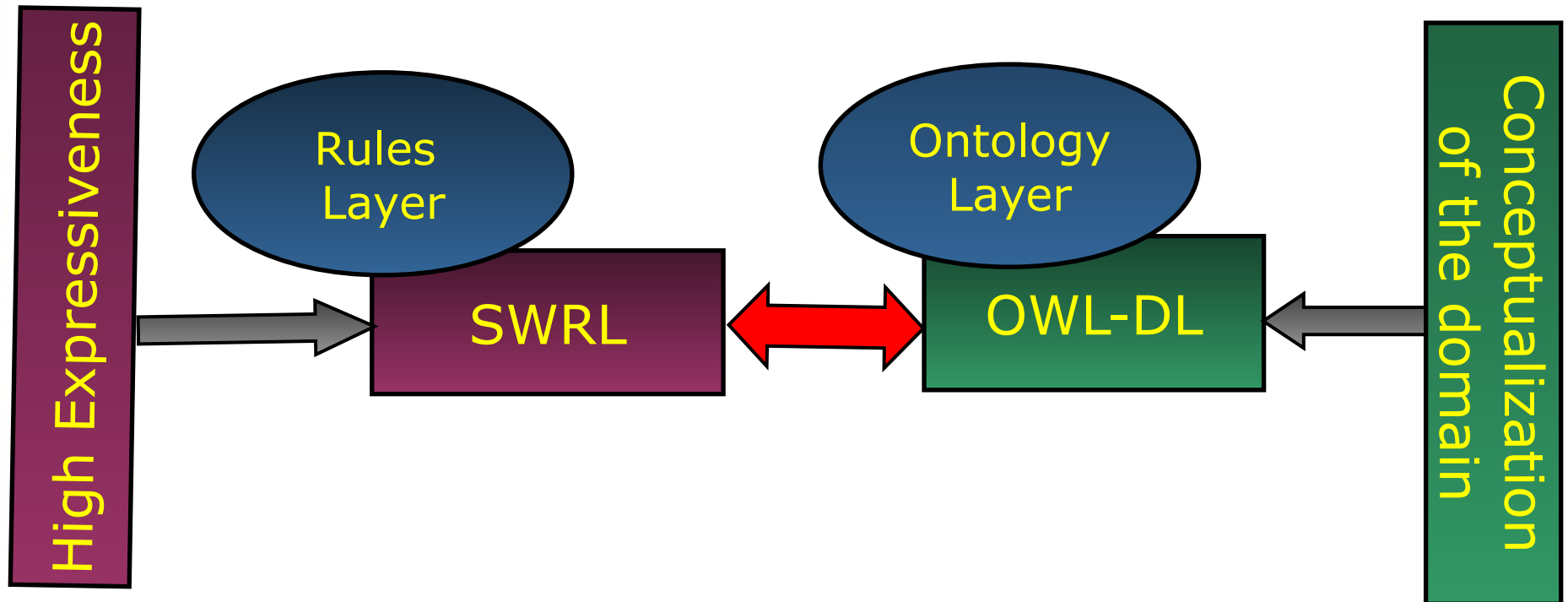
- What is SWRL?
- What is Jess?
  - Installing Jess
  - Creating rules
- SWRL Rules
  - Atom Types
  - SWRLTab
- SQWRL
  - SQWRLTab
- SWRL Resources

# 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.protege.xowl/

# Creating Rules

SWRL Rules

| Enabled                             | Name            | Expression                                                                                                                                                              |
|-------------------------------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <input checked="" type="checkbox"/> | Def-hasAunt     | $\rightarrow \text{Person}(?x) \wedge \text{hasParent}(?x, ?y) \wedge \text{hasSister}(?y, ?z) \wedge \text{hasAunt}(?y, ?z)$                                           |
| <input checked="" type="checkbox"/> | Def-hasBrother  | $\rightarrow \text{Person}(?x) \wedge \text{hasSibling}(?x, ?y) \wedge \text{Male}(?y) \wedge \text{hasBrother}(?x, ?y)$                                                |
| <input checked="" type="checkbox"/> | Def-hasDaughter | $\rightarrow \text{Person}(?x) \wedge \text{hasChild}(?x, ?y) \wedge \text{Woman}(?y) \wedge \text{hasDaughter}(?x, ?y)$                                                |
| <input checked="" type="checkbox"/> | Def-hasFather   | $\rightarrow \text{Person}(?x) \wedge \text{hasParent}(?x, ?y) \wedge \text{Male}(?y) \wedge \text{hasFather}(?x, ?y)$                                                  |
| <input checked="" type="checkbox"/> | Def-hasMother   | $\rightarrow \text{Person}(?x) \wedge \text{hasParent}(?x, ?y) \wedge \text{Woman}(?y) \rightarrow \text{hasMother}(?x, ?y)$                                            |
| <input checked="" type="checkbox"/> | Def-hasNephew   | $\rightarrow \text{Person}(?x) \wedge \text{hasSibling}(?x, ?y) \wedge \text{hasSon}(?y, ?z) \rightarrow \text{hasNephew}(?x, ?z)$                                      |
| <input checked="" type="checkbox"/> | Def-hasNiece    | $\rightarrow \text{Person}(?x) \wedge \text{hasSibling}(?x, ?y) \wedge \text{hasDaughter}(?y, ?z) \rightarrow \text{hasNiece}(?x, ?z)$                                  |
| <input checked="" type="checkbox"/> | Def-hasParent   | $\rightarrow \text{Person}(?y) \wedge \text{hasConsort}(?y, ?z) \wedge \text{hasParent}(?x, ?y) \rightarrow \text{hasParent}(?x, ?z)$                                   |
| <input checked="" type="checkbox"/> | Def-hasSibling  | $\rightarrow \text{Person}(?x) \wedge \text{hasChild}(?x, ?y) \wedge \text{hasChild}(?x, ?z) \wedge \text{differentFrom}(?y, ?z) \rightarrow \text{hasSibling}(?x, ?z)$ |
| <input checked="" type="checkbox"/> | Def-hasSister   | $\rightarrow \text{Person}(?x) \wedge \text{hasSibling}(?x, ?y) \wedge \text{Female}(?y) \wedge \text{hasSister}(?x, ?y)$                                               |
| <input checked="" type="checkbox"/> | Def-hasSon      | $\rightarrow \text{Person}(?x) \wedge \text{hasChild}(?x, ?y) \wedge \text{Male}(?y) \rightarrow \text{hasSon}(?x, ?y)$                                                 |
| <input checked="" type="checkbox"/> | Def-hasUncle    | $\rightarrow \text{Person}(?x) \wedge \text{hasParent}(?x, ?y) \wedge \text{hasBrother}(?y, ?z) \wedge \text{hasUncle}(?x, ?z)$                                         |
| <input checked="" type="checkbox"/> | Query-1         | $\rightarrow \text{hasSon}(?x, ?z) \rightarrow \text{query:select}(?x, ?z)$                                                                                             |
| <input checked="" type="checkbox"/> | Query-2         | $\rightarrow \text{hasSon}(?x, ?z) \rightarrow \text{query:select}(?x) \wedge \text{query:count}(?z) \wedge \text{query:orderByDescending}(?z)$                         |

Clone Delete

Edit Create

Jess

SWRL Rule Editor toolbar icons: Undo, Redo, Add, Delete, Save, Run.

# 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:**

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

antecedent

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:allDifferent**:

*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:

The image shows the SWRLTab interface. On the left, the 'Asserted Instances' panel lists M01 through M12, with M12 selected. The main area displays a table of properties and their values. The 'Property' column lists various relationships, and the 'Value' column shows the corresponding values. The 'Lang' column is empty.

| Property     | Value | Lang |
|--------------|-------|------|
| rdfs:comment |       |      |
| hasConsort   |       |      |
| hasDaughter  |       |      |
| hasSister    |       |      |
| hasFather    |       |      |
| hasMother    |       |      |
| hasNephew    |       |      |
| hasSon       |       |      |
| hasSex       | Male  |      |
| hasAunt      |       |      |
| hasNiece     |       |      |
| hasUncle     |       |      |
| hasBrother   |       |      |
| hasParent    |       |      |
| isTwinwith   | M11   |      |
| hasChild     |       |      |
| hasSibling   |       |      |

# SWRLTab: Displaying Results

- After Jess Reasoning

The screenshot displays the SWRLTab interface after Jess reasoning. The left pane, 'INSTANCE BROWSER', shows a list of instances for the class 'Man', including M01 through M12. The right pane, 'INDIVIDUAL EDITOR for M12', shows the properties of the instance M12. The 'hasFather' and 'hasMother' properties are highlighted with a red box, and the 'hasParent' property is also highlighted with a red box. A red speech bubble labeled 'Forward chaining' points to the 'hasFather' and 'hasMother' properties. Two grey speech bubbles labeled 'New Fact' point to the 'hasParent' property and the 'hasSex' property.

| Property     | Value    | Lang |
|--------------|----------|------|
| rdfs:comment |          |      |
| hasConsort   |          |      |
| hasDaughter  |          |      |
| hasSister    |          |      |
| hasFather    | M01      |      |
| hasMother    | F01      |      |
| hasSex       | Male     |      |
| hasAunt      |          |      |
| hasUncle     |          |      |
| hasBrother   |          |      |
| hasParent    | F01, M01 |      |
| Twinwith     | M11      |      |
| hasChild     |          |      |

# 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

Metadata (Ontology1154994098.owl) OWLClasses Properties Individuals Forms → SWRL Rules

SWRL Rules

| Name   | Expression                                                                                              |
|--------|---------------------------------------------------------------------------------------------------------|
| Rule-1 | → A(?a) ∧ hasIntProperty1(?a, ?i1) ∧ hasIntProperty2(?a, ?i2) ∧ swrlb:add(?i3, ?i1, ?i2) → hasIntPro... |
| 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(?i1, ?s1) ...  |
| Rule-7 | → C(?ccc) ∧ hasStringProperty1(?ccc, ?d) → query.select(?ccc, ?d)                                       |

→ SWRLQueryTab → Rule-7

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

Run rule Close Save...

# 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>