# ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΝΩΣΗΣ ΣΤΟΝ ΠΑΓΚΟΣΜΙΟ ΙΣΤΟ

## SWRL (Semantic Web Rule Language)

Ι. Χατζηλυγερούδης

# Μια γενική ιδέα…

- Η SWRL(Semantic Web Rule Language) ή στα ελληνικά Γλώσσα Κανόνων Σημασιολογικού Ιστού) είναι μια πρόταση για μια γλώσσα αναπαράστασης γνώσης υπό μορφή κανόνων στο Σημασιολογικό Ιστό, που συνδυάζει τις υπογλώσσες της γλώσσας οντολογίας Ιστού OWL (OWL DL και Lite) με εκείνες της γλώσσας σήμανσης κανόνα (μοναδιαίο/δυαδικό Datalog).

# Κανόνες (Rules)

- Ένας κανόνας σε SWRL έχει τη μορφή
  - **$B_1, \ldots, B_n \rightarrow A_1, \ldots, A_m$**
  - Τα κόμματα δείχνουν τη σύζευξη και στις δύο πλευρές
  - **$A_1, \ldots, A_m, B_1, \ldots, B_n$** μπορεί να είναι μορφής **C(x)**, **P(x,y)**, **sameAs(x,y)**, ή **differentFrom(x,y)** όπου το **C** είναι μια περιγραφή της OWL, η **P** μια OWL property , και **x**, **y** είναι μεταβλητές Datalog , σταθερές OWL, ή τιμές δεδομένων

# Ιδιότητες SWRL

- Εάν η  κεφαλή ενός κανόνα έχει περισσότερα από ένα άτομα, ο κανόνας μπορεί να μετασχηματιστεί σε ένα ισοδύναμο σύνολο κανόνων με ένα άτομο στην κεφαλή
- Οι εκφράσεις, όπως οι περιορισμοί, μπορούν να εμφανιστούν στην κεφαλή ή το σώμα ενός κανόνα
- Αυτό το χαρακτηριστικό γνώρισμα προσθέτει τη σημαντική εκφραστική δύναμη στην OWL, αλλά με κόστος την υψηλή τιμή αναποφασιστικότητας

# Σύνταξη SWRL

Η σύνταξη μιας ατομικής έκφρασης στην SWRL ορίζεται ως εξής:

Atom ← C(i) | D(v) | R(i, j) | U(i; v) | builtIn(p, v1,…, vn) | i = j | i ≠ j

- C : κλάση
- R : ιδιότητα αντικειμένου
- i, j : μεταβλητές ονομάτων του αντικειμένου ή μεμονομένα ονόματα μεταβλητών
- $v_1$,… $v_n$ : τύποι μεταβλητών και δεδομένων
- p : Built-in name
- D : Data type
- U : Data type Property

Παράδειγμα κανόνα

hasParent(?x1,?x2) ^ hasBrother(?x2,?x3) → hasUncle(?x1,?x3) )

# Σύνταξη SWRL

has_father(?a,?b) ^ has_married(?b,?c) → female(?c) ^ has_mother(?a,?c)

*Αξίζει να σημειωθεί πως ένας κανόνας με πολλαπλά άτομα στην κεφαλή του ισοδυναμεί με πολλαπλούς κανόνες που έχουν το ίδιο σώμα με τον αρχικό κανόνα και ένα άτομο στην κεφαλή τους*

Ο παραπάνω κανόνας ισοδυναμεί με τους ακόλουθους κανόνες :

has_father(?a,?b) ^ has_married(?b,?c) → female(?c)

και

has_father(?a,?b) ^ has_married(?b,?c) → has_mother(?a,?c)

# Σημείωση

- Ο συνδυασμός κανόνων μαζί με οντολογίες παρέχει μεγάλη εκφραστικότητα, καθώς προσπαθεί, και εν μέρει καταφέρνει, να συνδυάσει τα πλεονεκτήματα της κλασικής λογικής και του λογικού προγραμματισμού.
- Όμως, κάποια βασικά μειονεκτήματα παραμένουν. Θα πρέπει να τονιστεί πως η γλώσσα SWRL αποτελεί επέκταση της OWL και αυτό έχει ως αποτέλεσμα να λειτουργεί σε συνέπεια με την υπόθεση του ανοικτού κόσμου (Open-World Assumption), όπως και η OWL.
- Έτσι, η SWRL δεν υποστηρίζει τελεστή *not*, σε αντίθεση με τα περισσότερα περιβάλλοντα κανόνων που λειτουργούν σε περιβάλλοντα κλειστού κόσμου.
- Τέλος, θα πρέπει να σημειωθεί πως αφού οι κανόνες SWRL

  έχουν μορφή κανόνων-Horn, δεν επιτρέπουν διάζευξη ούτε στο σώμα ούτε στην κεφαλή τους.

# Λεκτική-Λογική Περιγραφή Κανόνα

Εάν ένας καλλιτέχνης x έχει υιοθετήσει μια τεχνοτροπία y και έχει ένα δημιούργημα z τότε το δημιούργημα z είναι της τεχνοτροπίας y.

***Η λεκτική περιγραφή του κανόνα***

***Η λογική περιγραφή του κανόνα***

artist (x) ^ style(y) ^ artist Style (x,y) ^ creator(x,z) → artistfactStyle(z,y)

# RDF Concrete Syntax

```
<swrl:Variable rdf:ID="x"/>
<swrl:Variable rdf:ID="y"/>
<swrl:Variable rdf:ID="z"/>
 <ruleml:Imp>
  <ruleml:body rdf:parseType="Collection">
    <swrl:ClassAtom>
      <swrl:classPredicate rdf:resource="Artist"/>
       <swrl:argument1 rdf:resource="#x" />
    </swrl:ClassAtom>
    <swrl:ClassAtom>
      <swrl:classPredicate rdf:resource="Style"/>
         <swrl:argument1 rdf:resource="#y" />
    </swrl:ClassAtom>
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource =
                                        "artistStyle"/>
       <swrl:argument1 rdf:resource="#x" />
       <swrl:argument2 rdf:resource="#y" />
    </swrl:IndividualPropertyAtom>
```

```
<swrl:IndividualPropertyAtom>
  <swrl:propertyPredicate
                        rdf:resource="creator"/>
     <swrl:argument1 rdf:resource="#x" />
     <swrl:argument2 rdf:resource="#z" />
</swrl:IndividualPropertyAtom>
</ruleml:body>
<ruleml:head rdf:parseType="Collection">
 <swrl:IndividualPropertyAtom>
   <swrl:propertyPredicate
                      rdf:resource="artifactStyle"/>
     <swrl:argument1 rdf:resource="#z" />
     <swrl:argument2 rdf:resource="#y" />
 </swrl:IndividualPropertyAtom>
</ruleml:head>
</ruleml:Imp>
```

# Αναλυτικά

```
<swrl:Variable rdf:ID="x"/>
<swrl:Variable rdf:ID="y"/>
<swrl:Variable rdf:ID="z"/>
```

***Ορισμοί μεταβλητών***

***Ορισμός και αντιστοίχιση κατηγορημάτων του σώματος***
***Συγκεκριμένά:***

artist (x)
style(y)
artist Style (x,y)
creator(x,z)

```
<ruleml:body rdf:parseType="Collection">
<swrl:ClassAtom>
  <swrl:classPredicate rdf:resource="Artist"/>
  <swrl:argument1 rdf:resource="#x" />
</swrl:ClassAtom>
<swrl:ClassAtom>
  <swrl:classPredicate rdf:resource="Style"/>
  <swrl:argument1 rdf:resource="#y" />
</swrl:ClassAtom>
<swrl:IndividualPropertyAtom>
  <swrl:propertyPredicate rdf:resource="artistStyle"/>
  <swrl:argument1 rdf:resource="#x" />
  <swrl:argument2 rdf:resource="#y" />
</swrl:IndividualPropertyAtom>
<swrl:IndividualPropertyAtom>
  <swrl:propertyPredicate rdf:resource="creator"/>
  <swrl:argument1 rdf:resource="#x" />
  <swrl:argument2 rdf:resource="#z" />
</swrl:IndividualPropertyAtom>
</ruleml:body>
```

# Αναλυτικά(2)

```
<ruleml:head rdf:parseType="Collection">
<swrl:IndividualPropertyAtom>
  <swrl:propertyPredicate rdf:resource="artifactStyle"/>
  <swrl:argument1 rdf:resource="#z" />
  <swrl:argument2 rdf:resource="#y" />
</swrl:IndividualPropertyAtom>
</ruleml:head>
```

***Ορισμός και αντιστοίχιση κατηγορημάτος της κεφαλής. Συγκεκριμένα:***
artistfactStyle(z,y)

```
<ruleml:Imp>
…
</ruleml:Imp>
```

***Αυτό το element επιτρέπει να πούμε ότι κάθε σύνδεσμος που ικανοποιεί το body του κανόνα πρέπει επίσης να ικανοποιήσει το head του ίδιου κανόνα.***

# XML Concrete Syntax

```
<swrlx:datarangeAtom>
...
</swrlx:datarangeAtom>
```

***Η περιγραφή σε ένα άτομικό datarange μπορεί να είναι μια ταυτότητα datatype, ή μπορεί να είναι ένα σύνολο literals.***

**Παράδειγμα:**

```
<swrlx:datarangeAtom>
  <owlx:Datatype owlx:name="&xsd;int" />
  <ruleml:var>x1</ruleml:var>
</swrlx:datarangeAtom>
```

```
<ruleml:var>xsd:string</ruleml:var>
```

***Καθορίζει απλά την ύπαρξη μιας μεταβλητής. Αυτό λαμβάνεται από το RuleML namespace.***

**Παράδειγμα:**

```
<ruleml:var>x1</ruleml:var>
```

**XML Concrete Syntax** = Συνδυασμός OWL XML synatx και RuleML XML syntax

# SWRL και Querying: SQWRL

- Η SWRL είναι μια γλώσσα κανόνων, όχι μια γλώσσα διατύπωσης ερωτήσεων
- Εντούτοις, ένας κανόνας προηγουμένως μπορεί να αντιμετωπισθεί ως προδιαγραφή ταιριάσματος σχεδίων, δηλ., μια ερώτηση.
- Με built-ins, γλωσσικές συμμορφώσεις των query extensions είναι δυνατές.
- Έχει αναπτυχθεί μια SWRL-based query γλώσσα που καλείται SQWRL

# Παράδειγμα στην SQWRL Query

Ερώτημα: «Επιστρέψτε όλους τους ενηλίκους σε μια οντολογία»

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

→ sqwrl:select(?p, ?age)

# Άλλη SQWRL Query

«Επιστρέψτε όλους τους ενηλίκους σε μια οντολογία διατεταγμένους με βάση την ηλικία»

Person(?p) ^ hasAge(?p, ?age) ^ swrlb:greaterThan(?age, 17) →
sqwrl:select(?p) ^ sqwrl:orderBy(?age)

# SQWRL ως γλώσσα διατύπωσης ερωτημάτων

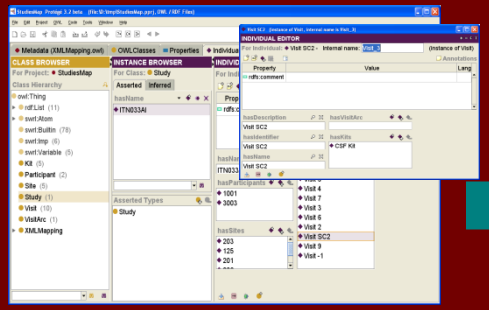
- Καθαρότερη σημασιολογία από την SPARQL
- OWL-based, όχι RDF-based
- Πολύ εκτατή μέσω του built-ins, π.χ., χρονικές ερωτήσεις που χρησιμοποιούν το temporal built-ins

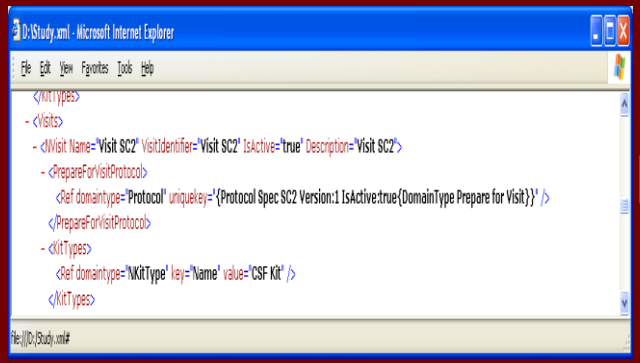
# XML Mapping με την βοήθεια SWRL Mapping Rules

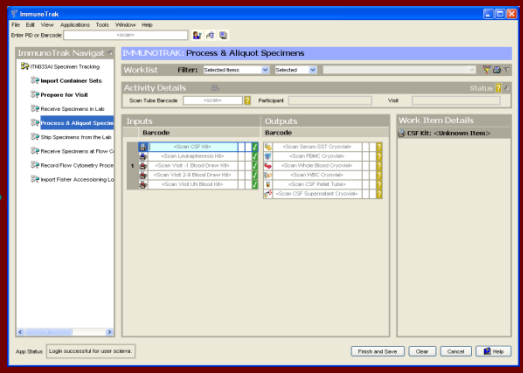
Ontology

XML Document

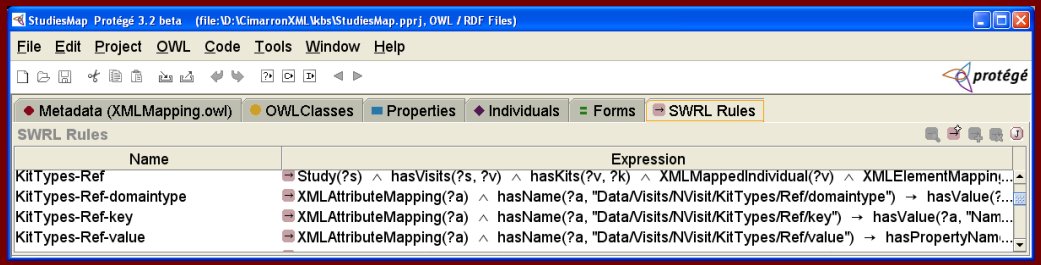
Application





SWRL Mapping Rules



# Μηχανή SWRL: SweetRules v2.0

- SweetRules v2.0 – μηχανή εκτέλεσης κανόνων SWRL
  http://xml.coverpages.org/ni2005-04-25-a.html
- Το SweetRules είναι ένα pluggable σύνολο εργαλείων κανόνων για RuleML και SWRL που χαρακτηρίζουν την διαλειτουργικότητα μεταξύ Prolog, των κανόνων παραγωγής, της OWL, CommonRules, Jena-2, και διάφορων άλλων γλωσσών και inferencing με την άρνηση, τις προτεραιότητες, και τις διαδικαστικές συνδέσεις.

# Άλλες Μηχανές

- swrl2clips
- Hoolet
- VIS use with JESS
- BaseVISor

# APIs

**JAXB SWRL API**

(http://www.daml.org/rules/proposal/jaxb/)

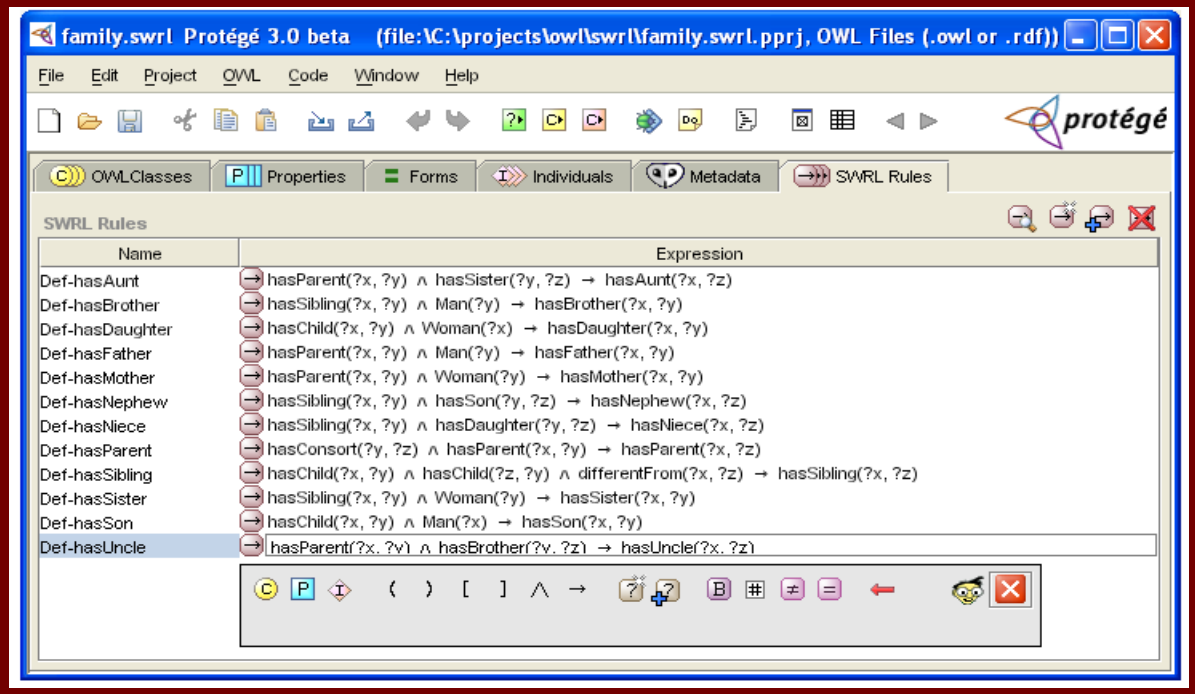
test1.swrlx: sample input for jaxbtest.java

# Εργαλεία σχετικά με την SWRL

- SWRLTab
- SQWRL

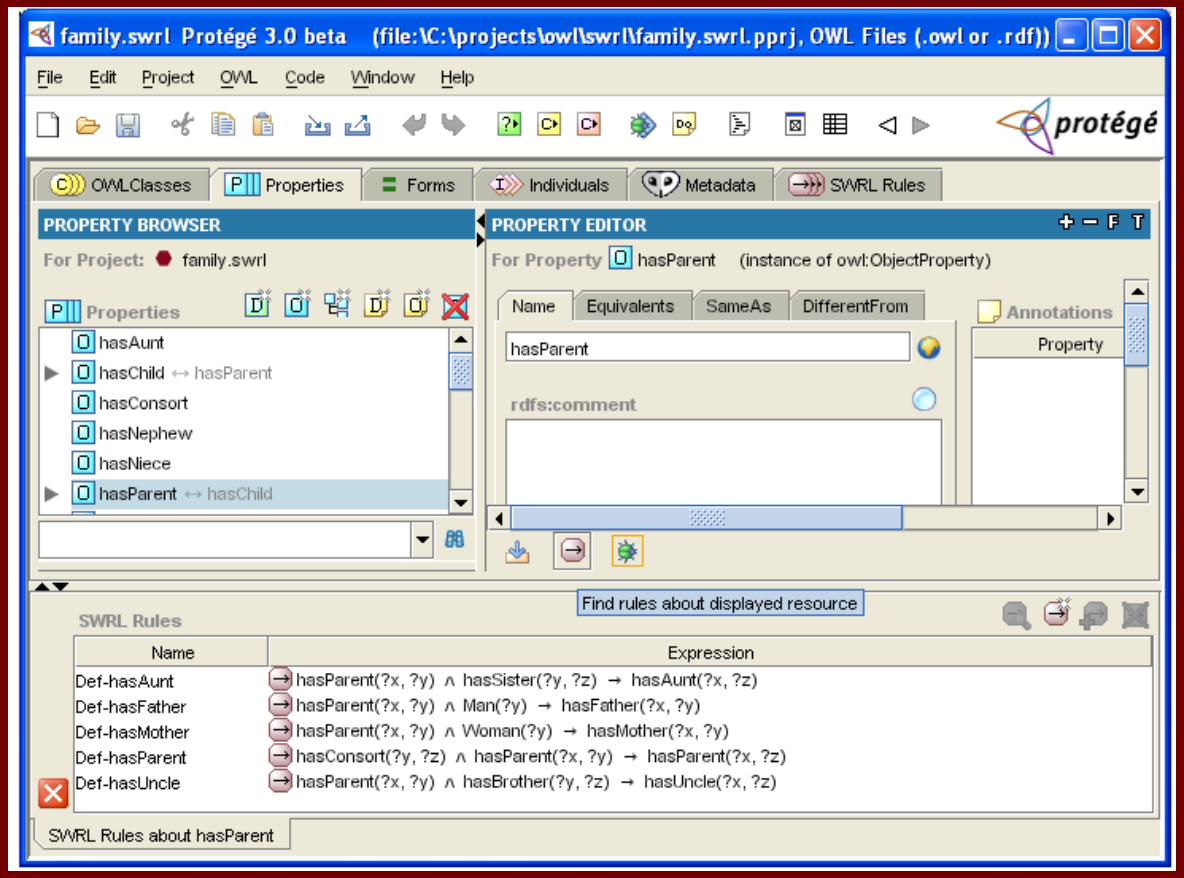
# SWRL Editor

Ο SWRL Editor είναι μία επέκταση του Protege-OWL SWRLTab που υποστηρίζει συγγραφή κανόνων SWRL. Μπορεί να χρησιμοποιηθεί για την δημιουργία SWRL κανόνων, να ανοίξει τους υπάρχοντες SWRL κανόνες, να διαβάσει και να γράψει SWRL κανόνες



# SWRL Editor

Το επόμενο screenshot παρουσιάζει τον κατάλογο κανόνων που περιέχουν μια αναφορά στη hasParent property.



# Μερικοί ακόμη Editors

- VIS RuleVISor
- Protege OWL Plugin, since build 215
- SWeDE