Ingéniérie du WEB Sémantique
Cours 3 : Ajouter des règles à une ontologie : SWRL

Odile PAPINI

POLYTECH
Université d'Aix-Marseille
odile.papini@univ-amu.fr
http://odile.papini.perso.luminy.univ-amu.fr/sources/WEBSEM.html

Plan du cours

Bibliographie I


Olivier Corby and Fabien Gandon and Catherine Faron-Zucker
Le Web sémantique : comment lier les données et les schémas sur le web ?
Dunod, 2012. ISBN : 978-2-10-057294-6.


John Hebeler and Matthew Fisher and Ryan Blace and Andrew Perez-Lopez and Mike Dean
Semantic Web Programming
Wiley, 2009. ISBN : 978-0-470-41801-7


Grigoris Antoniou & Frank van Harmelen
MIT university Press
http ://www.ics.forth.gr/isl/swprimer/presentation.htm
www.titan.be/common/docs/websemantique2007_1.ppt

Bibliographie I

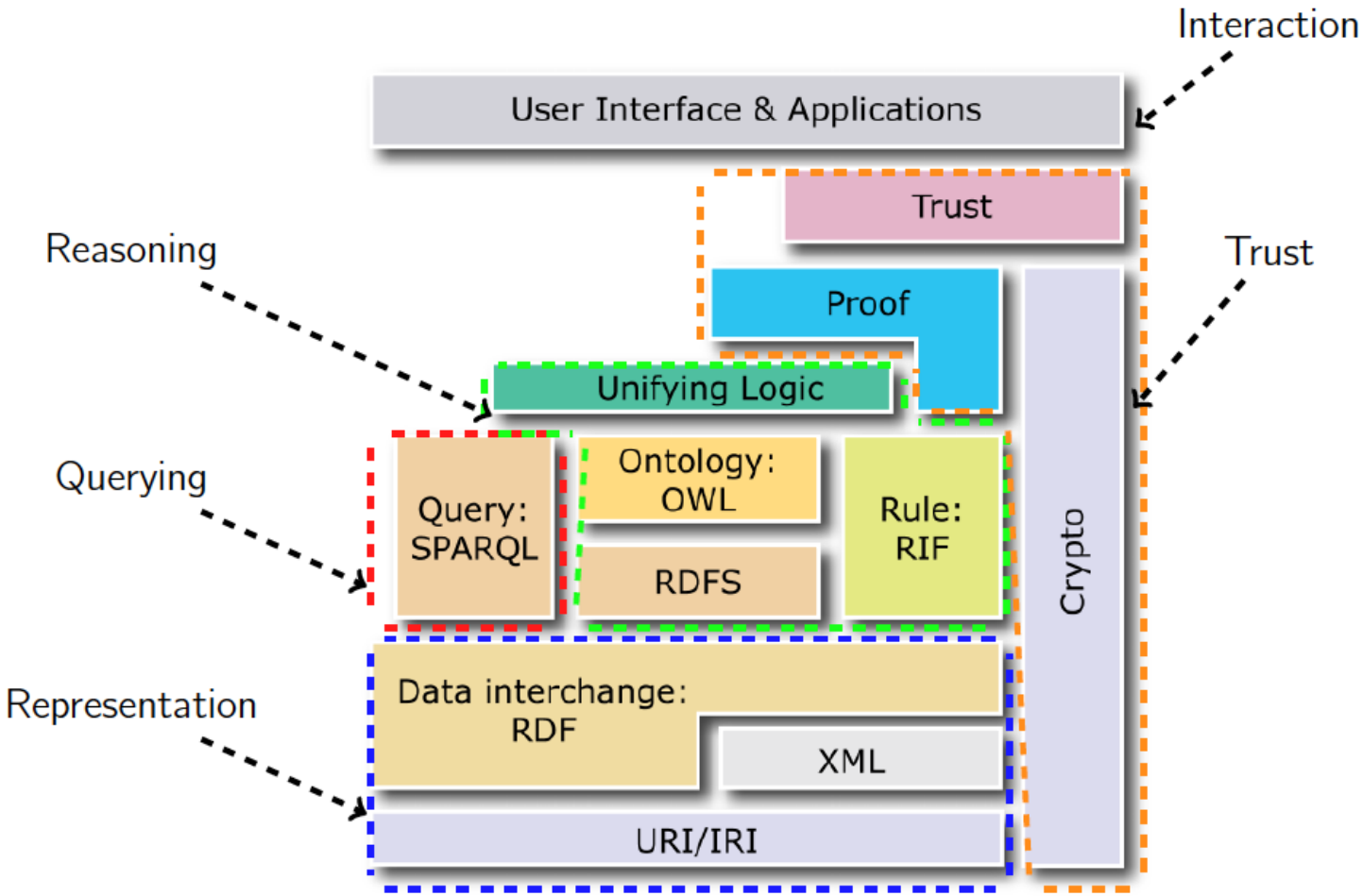

Pascal Hitzler and Markus Krotzsch and Sebastian Rudolph
Foundations of Semantic Web Technologies,
Chapman & Hall/CRC. 2009. ISBN : 9781420090505


John Domingue and Dieter Fensel and James A. Hendler
Handbook of Semantic Web
Springer. 2011. SBN : 978-3-540-92912-3


W3C
`http://www.w3.org/standards/semanticweb/`


W3C
`http://www.w3.org/Submission/SWRL/`

Le Web sémantique : La pile des standards du web sémantique



La pile des standards du web sémantique

- Représentation
  - URI/IRI : Universal Ressource Identifier/International Ressource Identifier
  - XML : Extensible Markup Language
  - RDF : Ressource Description Framework : description des ressources sous forme de graphe à base de triplets
- Raisonnement
  - RDFS : RDF Schema : langage de description de vocabulaire asocié à RDF (description de classes et propriétés)
  - OWL : Ontology Web language : langage de représentation des ontologies
  - RIF : Rule Interchange Format : échange de systèmes à base de règles
- Interrogation
  - SPARQL : Simple Protocol And Rdf Query Language : language d'interrogation de graphe RDF

Règles

Définition

si premisses alors conclusion ou premisses $\rightarrow$ conclusion

- premisse : antécédent (corps), condition d'application de la règle
- conclusion : conséquent (tête)

régles particulières :

- $\rightarrow$ conclusion : fait
  $\top \rightarrow c$

- premisses $\rightarrow$: contrainte
  $p_1 \wedge \cdots \wedge p_n \rightarrow \bot$

Règles

Règles en logique classique

- règle : $p_1 \wedge p_2 \wedge \cdots \wedge p_n \rightarrow c \equiv \neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n \vee c$
- fait : c
- contrainte : $\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n$
- clause de Horn : au plus un littéral positif

Inférence avec des règles

- Ensemble de règles $K = (F, R, C)$
  - F : ensemble de faits, ensemble des premisses $= \emptyset$
  - R : ensemble de règles, ensemble des premisses $\neq \emptyset$ et conclusion $\neq \emptyset$
  - C : ensemble de contraintes, conclusion $= \emptyset$

Inférence : $K \models Q$? avec Q une question (un fait)

- chaînage avant : à partir de K et des premisses des règles
- chaînage arrière : à partir de Q et des conclusions des règles

Inférence : chaînage avant

chaînage dirigé par les données

permet d'ajouter de nouveaux faits en appliquant les règles (production ou saturation)

- méthode déductive
- repose sur Modus Ponens : de A et $A \rightarrow B$ on déduit B
- une règle est applicable si ses premisses sont inclus dans l'ensemble des faits
- appliquer la règle : ajouter sa conclusion à l'ensemble des faits (si elle n'y est pas déjà)

Principe du chaînage avant

appliquer des règles tant que c'est possible ou tant qu'un certain fait n'a pas été obtenu

un algorithme de chaînage avant

```
algorithme FC(K)
données : K = (F, R), résultat : F saturé
début
Atraiter ← F
pour toute r ∈ R faire
Compteur(r) ← nbre_de_premisses(r)
tant que (Atraiter ≠ ∅) faire
    choisir f ∈ Atraiter
    Atraiter ← Atraiter\{f}
    pour toute r ∈ R telle que f ∈ premisse(r) faire
        Compteur(r) ← Compteur(r) − 1
        si Compteur(r) = 0 alors
            si conclusion(r) ∉ F alors
                Atraiter ← Atraiter ∪ {conclusion(r)}
                F ← F ∪ {conclusion(r)}
            finsi
        fin si
    fin pour
fin tant que
fin pour
fin
```

Inférence : chaînage avant

chaînage avant : exemple

$$R = \{$$
$$r_1 : a \wedge b \rightarrow c,$$
$$r_2 : c \wedge d \rightarrow f,$$
$$r_3 : f \wedge b \rightarrow e,$$
$$r_4 : f \wedge a \rightarrow g,$$
$$r_5 : g \wedge f \rightarrow b$$
$$\}$$

$$F = \{a, c, d\}$$

Quel est l'ensemble des conclusions C ?

Inférence : chaînage arrière

chaînage dirigé par les buts

cherche à éffacer le but en utilisant les règles qui l'ont produit

- méthode inductive
- à partir de la conclusion B on cherche la règle $A \rightarrow B$ qui l'a produite et on la remplace par A
- un but est produit par l'application d'une règle s'il est égal à la conclusion de la règle

Principe du chaînage arrière

à partir d'un but rechercher la règle qui l'a produit, le remplacer par la liste des premisses de la règle jusqu'à obtenir une liste vide

un algorithme de chaînage arrière

algorithme **BC**(K, L)
données : $K = (F, R)$, L : liste de buts,
résultat : retourne *VRAI* si la liste est effacée
début
si $L = \emptyset$ alors retourner *VRAI*
sinon
pour toute $r \in K$ telle que *conclusion*$(r) =$ *premier*(L) faire
    $L' \leftarrow$ *concatener premisse*(r) *et reste*(L)
    si **BC**(K, L') alors
        retourner *VRAI*
    fin si
fin pour
retourner *FAUX*
fin si
fin

Inférence : chaînage arrière

chaînage arrière : exemple 1

$$R = \{$$
$$r_1 : b \rightarrow c,$$
$$r_2 : a \wedge d \rightarrow b,$$
$$r_3 : d \rightarrow e,$$
$$\}$$

$$F = \{a, d\}$$

1) Liste de buts = c, e en d'autres termes : $K \models \{c, e\}$?

Inférence : chaînage arrière

chaînage arrière : exemple 2

$$R = \{$$
$$r_1 : a \wedge b \rightarrow c,$$
$$r_2 : c \wedge d \rightarrow f,$$
$$r_3 : f \wedge b \rightarrow e,$$
$$r_4 : f \wedge a \rightarrow g,$$
$$r_5 : g \wedge f \rightarrow b$$
$$\}$$

$$F = \{a, c, d\}$$

1) Liste de buts = g en d'autres termes : $K \models \{g\}$?

Ajouter des règles à une ontologie

Pourquoi ajouter des règles à une ontologie

permettre une plus grande expressivité

- énoncés déclaratifs : proche du langage naturel des utilisateurs
- règles conditionnelles : "*si* $\cdots$ *alors*"
- composition de propriétés

Ajouter des règles à une ontologie : Exemple

Difficulté d'exprimer en logiques de description certaines compositions de propriétés

Exemple 1 :
$Parent(X, Y) \wedge Frere(Z, X) \rightarrow oncle(Z, Y)$

Exemple 2 :
$Personne(X) \wedge aParent(X, Y) \wedge aParent(X, Z) \wedge aEpouse(Y, Z) \rightarrow EnfantDeParentsMaries(X)$

SWRL

SWRL

Semantic Web Rule Language

- SWRL mai 2004
- langage de règles basé sur OWL
- fournit des capacités d'inférences supérieures à OWL seul
- combinaison de OWL DL (ontologies) et de RuleML (règles)

SWRL

règle SWRL

antécédent $\rightarrow$ conséquent (si antécédent alors conséquent)

- antécédent : conjonctions d'atome
- conséquent : conjonction d'atomes
- atome :
  - C(x) où C est un concept (owl :Class) et
  - type de données
  - p(x,y) où p est un rôle (propriété d'objet owl :ObjectProperty) et x, y variables
  - propriété de type de données (owl :dataProperty)
  - sameAs(x,y) et differentFrom(x,y)
  - atome avec prédicat prédéfini "built-in"

x, y : variables, individus OWL, valeurs OWL

noms : RDF URI

SWRL

espace de nommage

espace de nommage : préfixes

Préfixe	URI de l'espace de nommage
rdf	http ://www.w3.org/1999/02/22-rdf-syntax-ns♯
rdfs	http ://www.w3.org/2000/01/rdf-schemas♯
xsd	http ://www.w3.org/2001/XMLSchemas♯
owl	http ://www.w3.org/2002/07/owl♯

SWRL

exemple d'atomes

Personne(?p)

xsd :int(?x)

aEnfant(?x, ?y)

aAge(?x, ?age), aAge(?x, 100)

differentFrom(?x, ?y), sameAs(?x, ?y),

prédicat "built-in" : prédicat prédéfini évalué à vrai si les arguments satisfont le prédicat swrlb :greaterThan(?age,18), permet d'affecter une valeur swrlb :add(?x,2,3),

SWRL

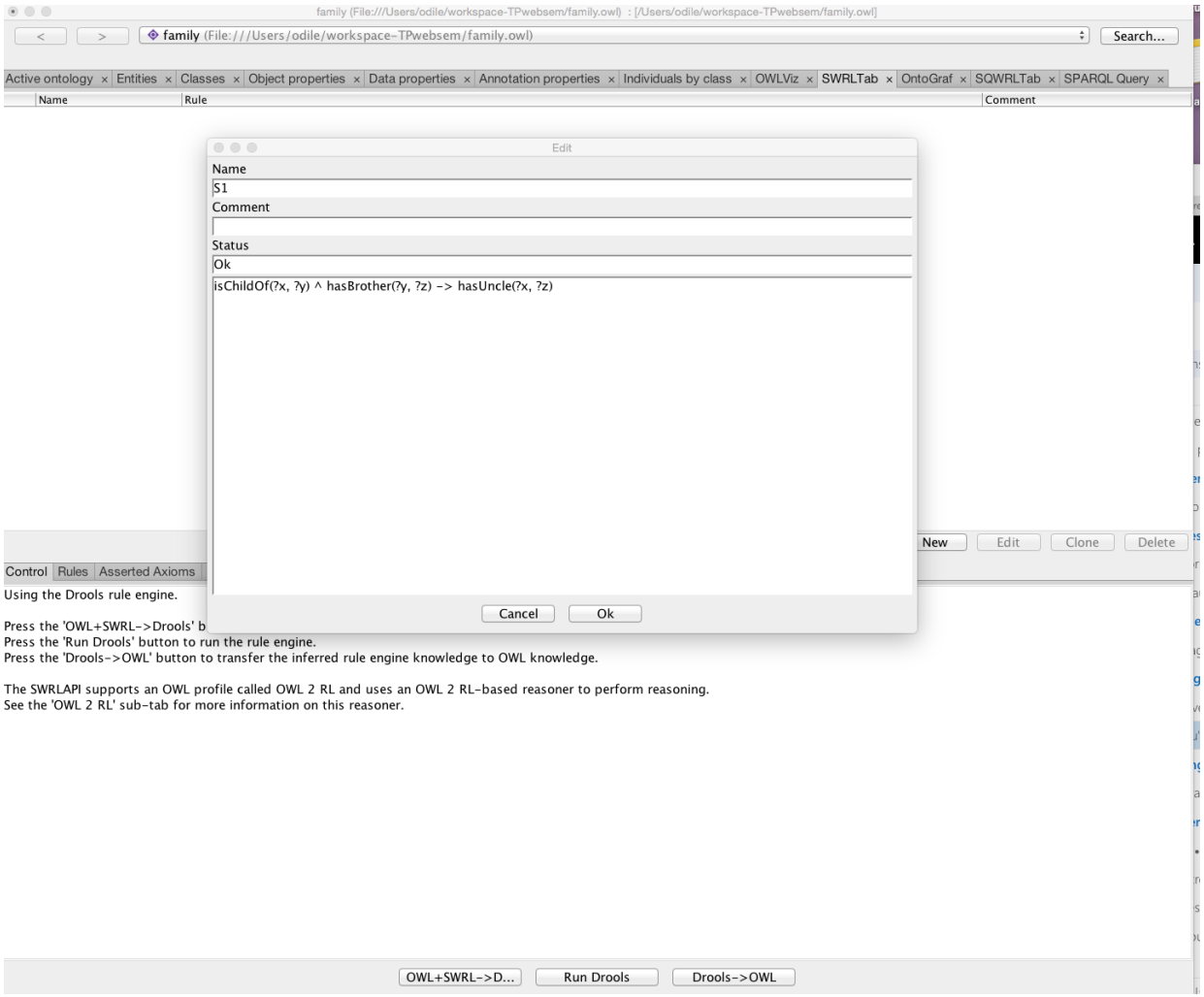
Exemple de règle :

$$hasParent(x, y) \wedge hasSibling(y, z) \wedge hasSexmale(z) \rightarrow hasUncle(x, z)$$

Traduction en SWRL :

```
hasParent(?x, ?y) ^ hasSibling(?y, ?z)  ^ hasSexmale(?z, true) -> hasUncle(?x, ?z)
```

SWRL : exemple de règle sous Protégé



Liens SWRL/OWL

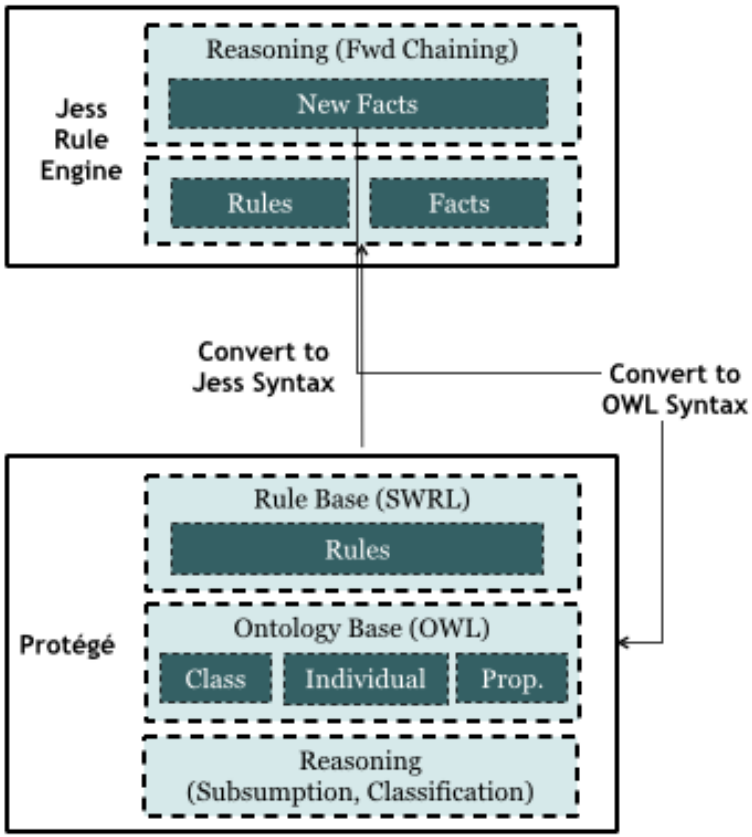


Figure: source : Knut Hinkelmann

SWRL

Base de connaissances SWRL

$K' = (K, R)$ où

- K : une base de connaissances de $SHOIN(D)$
- R : une base (ensemble fini) de règles

Inférence en SWRL

- avec un raisonneur (Pellet)
- chaînage avant

SWRL

syntaxe XML

- Combinaison des langages OWL (syntaxe XML) et RuleML (syntaxe XML)
- espaces de nommage :
  http ://www.w3.org/2003/11/swrlx
  http ://www.w3.org/2003/11/ruleml
  http ://www.w3.org/2003/105/owl-xml
  http ://www.w3.org/2001/XMLSchema

documentation W3C : http ://www.w3.org/Submission/SWRL/

SWRL

Exemple de règle :

$hasParent(x, y) \wedge hasSibling(y, z) \wedge hasSexmale(z) \rightarrow hasUncle(x, z)$

Traduction en SWRL :

```
hasParent(?x, ?y) ^ hasSibling(?y, ?z)  ^ hasSexmale(?z, true) -> hasUncle(?x, ?z)
```

documentation W3C : http ://www.w3.org/Submission/SWRL/

SWRL

syntaxe RDF

- Réutilisation des fonctions prédéfinies de XQuery et XPath
- espace de nommage : http ://www.w3.org/2003/11/swrlb
- fonctions prédéfinies pour :
  - comparaisons
  - booléens
  - mathématiques
  - chaînes de caractères
  - temps

SWRL

syntaxe RDF

Fonctions prédéfinies : comparaisons

- swrlb :equal
- swrlb :notEqual
- swrlb :lessThan
- swrlb :lessThanOrEqual
- swrlb :greaterThan
- swrlb :greaterThanOrEqual

syntaxe RDF

Fonctions prédéfinies : booléens

- swrlb :booleanNot

SWRL

syntaxe RDF

Fonctions prédéfinies : mathématiques

- swrlb :add
- swrlb :sustract
- swrlb :multiply
- swrlb :divide
- swrlb :integerDivide
- swrlb :mod
- swrlb :pow
- swrlb :unaryPlus
- swrlb :unaryMinus
- swrlb :abs
- swrlb :ceiling
- swrlb :floor
- swrlb :round
- swrlb :roundHalfToEven
- swrlb :sin
- swrlb :cos

SWRL

syntaxe RDF

Fonctions prédéfinies : chaînes de caractères

- swrlb :stringEqualIgnoreCase
- swrlb :stringConcat
- swrlb :substring
- swrlb :stringLength
- swrlb :normalizeSpace
- swrlb :upperCase
- swrlb :lowerCase
- swrlb :translate
- swrlb :contains
- swrlb :containsIgnoreCase
- swrlb :startsWith
- swrlb :endsWith
- swrlb :substringBefore
- swrlb :substringAfter
- swrlb :matches
- swrlb :replace

SWRL

syntaxe RDF

Fonctions prédéfinies : temps

- swrlb :yearMonthDuration
- swrlb :dayTimeDuration
- swrlb :dateTime
- swrlb :date
- swrlb :time
- swrlb :addYearMonthDurations
- swrlb :substractYearMonthDurations
- swrlb :multiplyYearMonthDurations
- swrlb :divideYearMonthDurations
- swrlb :addDayTimeDurations
- swrlb :substractDayTimeDurations
- swrlb :multiplyDayTimeDurations
- swrlb :divideDayTimeDurations
- swrlb :substractDates
- swrlb :substractTimes

SWRL

syntaxe RDF

Fonctions prédéfinies : temps

- swrlb :addYearMonthDurationToDateTime
- swrlb :addDayTimeDurationToDateTime
- swrlb :substractYearMonthDurationFromDateTime
- swrlb :substractDayTimeDurationFromDateTime
- swrlb :addYearMonthDurationToDate
- swrlb :addDayTimeDurationToDate
- swrlb :substractYearMonthDurationFromDate
- swrlb :substractDayTimeDurationFromDate
- swrlb :addDayTimeDurationToTime
- swrlb :substractDayTimeDurationFromTime
- swrlb :substractDayTimesYieldingYearMonthDuration
- swrlb :substractDayTimesYieldingDayTimeDuration

SWRL

syntaxe RDF

Fonctions prédéfinies : URI

- swrlb :resolveURI
- swrlb :anyURI

syntaxe RDF

Fonctions prédéfinies : listes

- swrlb :rlistConcat
- swrlb :listIntersection
- swrlb :listSubtraction
- swrlb :member
- swrlb :length
- swrlb :first
- swrlb :rest
- swrlb :sublist
- swrlb :empty

règles en SWRL : exercices

Ecrire les règles suivantes en SWRL avec un ontologie qui comporte

- **les concepts suivants :** Personne, Homme, Femme, GrandMere, Adulte, Auteur, Publication
- **les rôles suivants :** aEnfant, aFille, aFrere, aOncle, aAge, aAgeConducteur, aNumeroTel, aNumeroTelInt, coopereAvec, aSalaireEnEuros, aSalaireEnDollars

1) Les personnes qui ont un enfant de sexe féminin sont des personnes qui ont une fille.
2) Les personne de sexe féminin qui a un petit-enfant est une grand-mère.
3) Les frères des parents d'une personne sont ses oncles.
4) Les personnes âgées de plus de 18 ans sont des adultes.
5) Les personnes âgées de plus de 18 ans et de moins de 80 ans ont l'âge d'être conducteur.
6) Les personnes qui ont un numéro de téléphone qui commence par + ont des numéros de téléphone internationaux
7) Deux auteurs de publications coopèrent.
8) Si le salaire en euros d'une personne est de S alors il est de $S \times 1.20$ en dollars.

SQWRL

SQWRL

Semantic Query-enhanced Web Rule Language

- SQWRL 2009
- interrogation d'ontologie avec des règles
- langage basé sur SWRL

http : *//ceur − ws.org/Vol − 529/owled2009_submission_42.pdf*

https : *//github.com/protegeproject/swrlapi/wiki/SQWRL*

SQWRL

Mécanisme d'interrogation avec SQWRL

Une règle r :

$$\text{antécédent} \rightarrow \text{conséquent}$$

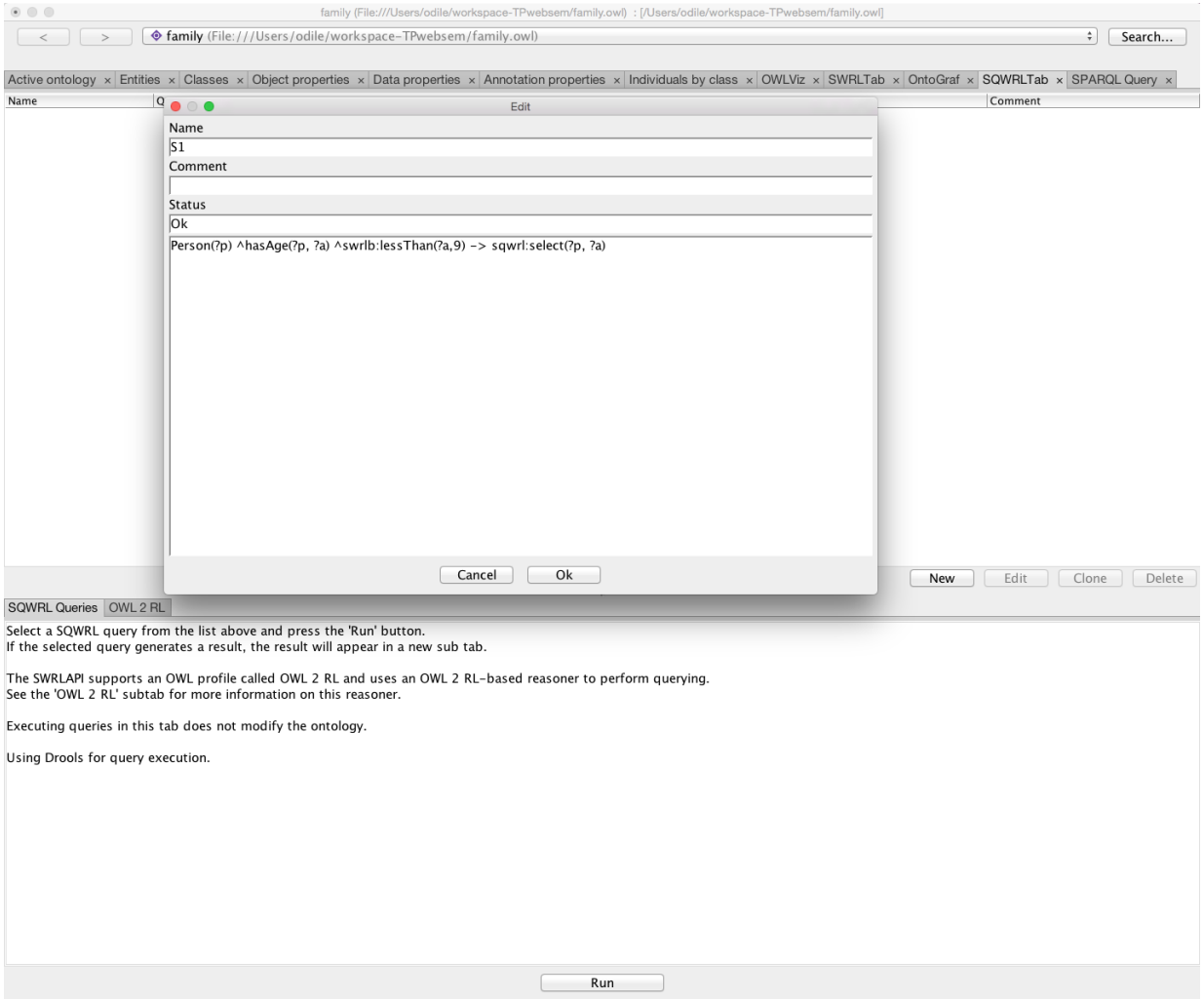
- l'antécédent de r : spécification de la requête
- conséquent de la règle r : recherche selon la spécification

Fonction de base : sqwrl :select

Exemple

```
Personne(?p) ^aAge(?p, ?a) ^swrlb:lessThan(?a,9) -> sqwrl:select(?p, ?a)
```

SQWRL : exemple d'interrogation sous Protégé



SQWRL

Syntaxe

Fonctions prédéfinies : de base

- sqwrl :select
- sqwrl :selectDistinct

Syntaxe

Fonctions prédéfinies : compter

- sqwrl :count
- sqwrl :countDistinct

SQWRL

Syntaxe

Fonctions prédéfinies : aggréger

- sqwrl :min
- sqwrl :max
- sqwrl :sum
- sqwrl :avg

Syntaxe

Fonctions prédéfinies : ordonner

- sqwrl :orderBy
- sqwrl :orderByDescending

SQWRL

Syntaxe

Fonctions prédéfinies : sélectionner un sous-ensemble de résultats (I)

- sqwrl :limit
- sqwrl :firstN
- sqwrl :lastN
- sqwrl :notFirstN
- sqwrl :notLastN
- sqwrl :leastN
- sqwrl :greatestN
- sqwrl :notLeastN
- sqwrl :notGreatestN

SQWRL

Syntaxe

Fonctions prédéfinies : sélectionner un sous-ensemble de résultats (II)

- sqwrl :nth
- sqwrl :notNth
- sqwrl :nthLast
- sqwrl :notNthLast
- sqwrl :nthSlice
- sqwrl :nthLastSlice
- sqwrl :notNthSlice
- sqwrl :notNthLastSlice

SQWRL :exemples

Exemples d'interrogation en SQWRL

```
Personne(?p) ^ aAge(?p, ?a) ^swrlb:lessThan(?a,9) -> sqwrl:select(?p, ?a) ^ orderBy(?a)
```

```
Personne(?p) -> sqwrl:count(?v)
```

```
Personne(?p) ^ aVoiture(?p, ?v)  -> sqwrl:select(?p) ^ sqwrl:count(?v)
```

```
Personne(?p) ^ aAge(?p, ?a)  -> sqwrl:avg(?a)
```

Règle :

```
Personne(?x) ^ aAge(?x, ?a) ^ swrlb:greaterThan(?a, 17) -> Adulte(?x)
```

Requête :

```
Adult(?p) -> sqwrl:select(?p)
```

SQWRL : exercices

Ecrire les requêtes suivantes en SQWRL avec un ontologie qui comporte

- **les concepts suivants :** Personne, Homme, Femme, GrandMere, Adulte, Auteur, Publication
- **les rôles suivants :** aEnfant, aFille, aFrere, aOncle, aAge, aAgeConducteur, aNumeroTel, aNumeroTelInt, coopereAvec, aSalaireEnEuros, aSalaireEnDollars

1) Quel est l'âge de la personne la plus âgées ?
2) Quel est nombre de personnes de moins de 18 ans ?
3) Par personne, quel est le nombre de personnes qui coopèrent avec cette personne ?
4) Quels sont les personnes et les salaires en euros par ordre croissant des salaires ?
5) Quels sont les personnes et les salaires en livre par ordre décroissant des salaires ?
6) Quelles sont les deux premières personnes qui ont l'âge d'être conducteur ?