# Semantic Web Rule Languages

**Tutorial at CSWS2009**
**29th Aug, 2009**

**Jeff Z. Pan, Yuting Zhao,** Stuart Taylor

Department of Computing Science

University of Aberdeen, UK

# Tutorial Overview

- This tutorial will address
  - Why rules are needed in the Semantic Web
  - How does OWL 2 relate to Semantic Web rules
  - How to use ontologies and rules
  - How to reason with ontologies and rules

# Tutorial Outline

- Motivation
- OWL 2 and Semantic Web rules
  - The big picture
- Some technical discussions on Semantic Web rules
- Practical: Hands-on Session

# OWL: Standard Semantic Web Ontology Language

- OWL DL is **decidable**
  - Efficient reasoning engines (for ontologies with reasonable sizes)
  - for standard reasoning tasks (in particular TBox reasoning)
- OWL DL is expressive enough to cover a wide range of applications
  - Semantic Web/Grid, eScience, multimedia, software engineering, medicine, biology, agriculture, geography, space, manufacturing, defence

[First OWLED Workshop, 2005
Photo Credit: Ian Horrocks]

# Why Rules in the Semantic Web

- Expressive power: there are statements that can not be represented by OWL alone
  - Beyond tree / forest shape model
  - Provides expressive query language for ontologies
  - non-monotonic reasoning, with non-classical negation
- People might be more familiar with implementing rule engines then ontology reasoners
- It might be easier for users to write “if … then …” rules than OWL axioms

Therefore, we need both ontologies and rules.

# The Early Days of KR: Rule-Based Formalisms

- Rules provide a natural way of modelling “reason-result” knowledge
- General form of a rule:

  Body ⇒ Head

  – Means “if Body, then Head“
- Example:

hasFather(?x,?y) ⇒ hasChild(?y,?x)

# Example: Why OWL not Enough

- Example: how to represent hasUncle
  - As a class (OWL can represent)

    Class( Uncle complete restriction(

    inverse(hasBrother)  Parent ))

  - As a property (OWL can not represent)

hasParent(?x,?p), hasBrother(?p,?b) ⇒ hasUncle(?x,?b)

ABox: hasBrother(Tom, Tim), hasParent(Mary,Tom)

# A Different Story in OWL 2

- OWL 2 allows property chains

ObjectPropertyChain( P1, …, Pn )

- Therefore the rule

hasParent(?x,?p), hasBrother(?p,?b) ⇒ hasUncle(?x,?b)

can now be represented as

SubObjectPropertyOf(

ObjectPropertyChain(hasParent hasBrother) hasUncle)

Question: What are the impacts of OWL 2 to SW rule languages?

# Tutorial Outline

- Motivation
- OWL 2 and Semantic Web rules
  - The big picture
- More technical discussions on Semantic Web rules
- Practical: Hands-on Session

# Why OWL is Not Enough (or Why OWL 2)

- **Too expensive to reason with**
  - High complexity: NEXPTIME-complete
  - The most lightweight sublanguage OWL-Lite is **NOT** lightweight
  - Some ontologies only use some limited expressive power; e.g. The SNOMED (Systematised Nomenclature of Medicine) ontology
- Not expressive enough; e.g.
  - No user defined datatypes [Pan 2004; Pan and Horrocks, 2005]
  - No metamodeling support [Pan 2004; Pan, Horrocks, Schreiber, 2005]
  - Limited property support [Horrocks et al., 2006]

# What is OWL 2

- A new version of OWL
- Main goals:

1. **To define "profiles" of OWL that are**:

– smaller, easier to implement and deploy

– cover important application areas and are easily understandable to non-expert users

2. To add a few extensions to current OWL that are useful, and is known to be implementable

– user defined datatypes, metamodeling, more property constructors

# New Expressiveness in OWL 2

- New **expressive power on properties**
  - qualified cardinality restrictions, e.g.:

    ObjectMinCardinality(2 hasFriend Scottish)
  - property chain inclusion axioms, e.g.:

    SubObjectPropertyOf(ObjectPropertyChain(parent brother) uncle)
  - local reflexivity restrictions, e.g.:

    ObjectExistsSelf(likes) [for narcissists]
  - reflexive, irreflexive, symmetric, and universal properties, e.g.:

    ReflexiveObjectProperty(hasRelative);
    IrreflexiveObjectProperty(husbandOf)
  - disjoint properties, e.g.:

    DisjointObjectProperties(childOf spouseOf)
  - keys, e.g.:

    HasKey( Person () ( hasSSN ) )

# OWL 2 DL

- $\mathcal{R}$ often used for $\mathcal{ALC}$ extended with property chain inclusion axioms
  - following the notion introduced in $\mathcal{RIQ}$ [Horrocks and Sattler, 2003]
  - including transitive property axioms
- Additional letters indicate other extensions, e.g.:
  - $\mathcal{S}$ for property characteristics (e.g., reflexive and symmetric)
  - $\mathcal{O}$ for **nominals**/singleton classes
  - $\mathcal{I}$ for inverse roles
  - $\mathcal{Q}$ for qualified number restrictions
- property characteristics ($\mathcal{S}$) + $\mathcal{R}$ + nominals ($\mathcal{O}$) + inverse ($\mathcal{I}$) + qualified number restrictions($\mathcal{Q}$) = $\mathcal{SROIQ}$
- $\mathcal{SROIQ}$ [Horrocks et al., 2006] is the basis for OWL 2 DL

# OWL 2 Profiles

- Rationale:
  - Tractable, easier to implement and deploy
  - Tailored to specific reasoning services
- Popular reasoning services
  - TBox reasoning: OWL 2 EL
  - ABox reasoning: OWL 2 RL
  - Query answering: OWL 2 QL
- Specification: http://www.w3.org/TR/2009/CR-owl2-profiles-20090611/

# OWL 2 Reasoners (partial list)

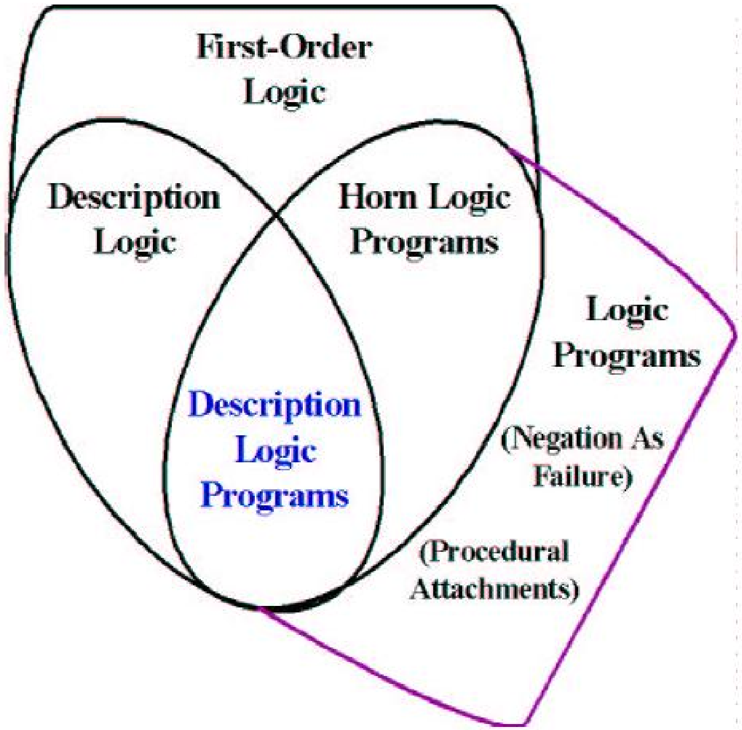
- OWL 2 DL reasoners
  - FaCT++ (Manchester), HermiT (Oxford), Pellet (Clarkparsia)
- OWL 2 EL reasoners
  - CEL (Dresden), REL (Aberdeen)
- OWL 2 RL reasoners
  - OWLRL (Ivan Herman), Jena (HP Labs Bristol, Aberdeen), Oracle 11g OWL Reasoner (Oracle)
- OWL 2 QL reasoners
  - QuOnto (Rome), Quill (Aberdeen)

- See: http://www.w3.org/2007/OWL/wiki/Test_Suite_Status

# Roadmap: OWL 2 Profiles

- Popular reasoning services
  - TBox reasoning: OWL 2 EL
    - see yesterday's tutorial "OWL 2: The Coming Version of OWL" at the Summer School of Logic Foundations of the Semantic Web
  - ABox reasoning: OWL 2 RL
    - most related to this tutorial
  - Query answering: OWL 2 QL
    - see the keynote "Scalable Query Answering over Expressive Ontology Languages" on 31st Aug in the CSWS2009 conference
- Specification: http://www.w3.org/TR/2009/CR-owl2-profiles-20090611/

# OWL 2 and Rules




- Three approaches
  - **OWL 2 RL**: (Explicit) Intersection of OWL 2 and horn rules
  - **DL rules** [Krötzsch et al. 2008]: Internalise some horn rules into OWL 2 axioms
  - "SWRL 2": union of OWL 2 and rules

# OWL 2 RL

- Inspired by Description Logic Programs [Grosof et al., 2003] and pD* [ter Horst, 2005]
  - amenable to implementation using rule-based technologies
- Main idea: avoid the need to infer the existence of individuals not explicitly present in the ontology
  - distinguish subClass expressions from superClass expressions
  - E.g., general existential restrictions can not be used as a superClassExp

# OWL 2 RL Axioms

- Redefine all axioms of the structural specification OWL 2 Specification that refer to class expressions
- Class axioms:
  - Class axioms: SubClassOf (subClassExp superClassExp)
- Domains and ranges
  - ObjectPropertyDomain (ObjectPropertyExp superClassExp)
  - ObjectPropertyRange (ObjectPropertyExp superClassExp)
- Class assertions
  - ClassAssertion (superClassExp individual)
- Specification: http://www.w3.org/TR/2009/CR-owl2-profiles-20090611/

# Examples: OWL 2 RL

- $\Rightarrow C(a), \Rightarrow R(a,b)$
  - $C(a), R(a,b)$
- $C(?x),D(?x) \Rightarrow E(?x)$
  - $C \sqcap D \sqsubseteq E$
- hasParent(?x,?p), hasBrother(?p,?b) ⇒ hasUncle(?x,?b)
  - hasParent ∘ hasBrother ⊑ hasUncle
- $C(?x),R(?x,?y) \Rightarrow D(?x)$
  - $C \sqcap \exists R.T \sqsubseteq D$
- $C(?x),R(?x,?y) \Rightarrow D(?y)$
  - $\exists R^{-}.C \sqsubseteq D$
- $C(?x),R1(?x,?y) \Rightarrow R2(?x,?y)$
  - ?

# Rule Internalisations in OWL 2

- Basic idea: turn classes into properties
- How?
  - By using local reflexivity restrictions ($\exists$R.**Self**)
  - which is beyond OWL 2 RL
- Example
  - C(?x),R1(?x,?y) $\Rightarrow$ R2(?x,?y)
    - C $\equiv \exists$Rc.**Self,** Rc $\circ$ R1 $\sqsubseteq$ R2

# Rule Internalisations in OWL 2 (II)

- How about
  - C(?x),D(?y) ⇒ R(?x,?y)
    - $C \equiv \exists Rc.$**Self**, $D \equiv \exists Rd.$**Self**, $Rc \circ Rd \sqsubseteq R$?
    - Incorrect, since ?x and ?y are unconnected
- Solution: universal property (that connects every pair of individuals)
  - C(?x),D(?y) ⇒ R(?x,?y)
    - $C \equiv \exists Rc.$**Self**, $D \equiv \exists Rd.$**Self**, $Rc \circ U \circ Rd \sqsubseteq R$

# Rule Internalisations in OWL 2 (III)

- To sum up:
  - OWL 2 can internalise many rules
  - Much more than those supported by OWL 2 RL
- But not all of them
  - E.g. those with cycles on the body
  - C(?x),R1(?x,?y),R2(?y,?z), R3(?z,?x) ⇒D(?x)

# Tutorial Outline

- Motivation
- OWL 2 and Semantic Web rules
  - The big picture
- More technical discussions on Semantic Web rules
- Practical: Hands-on Session

# Big picture revisit

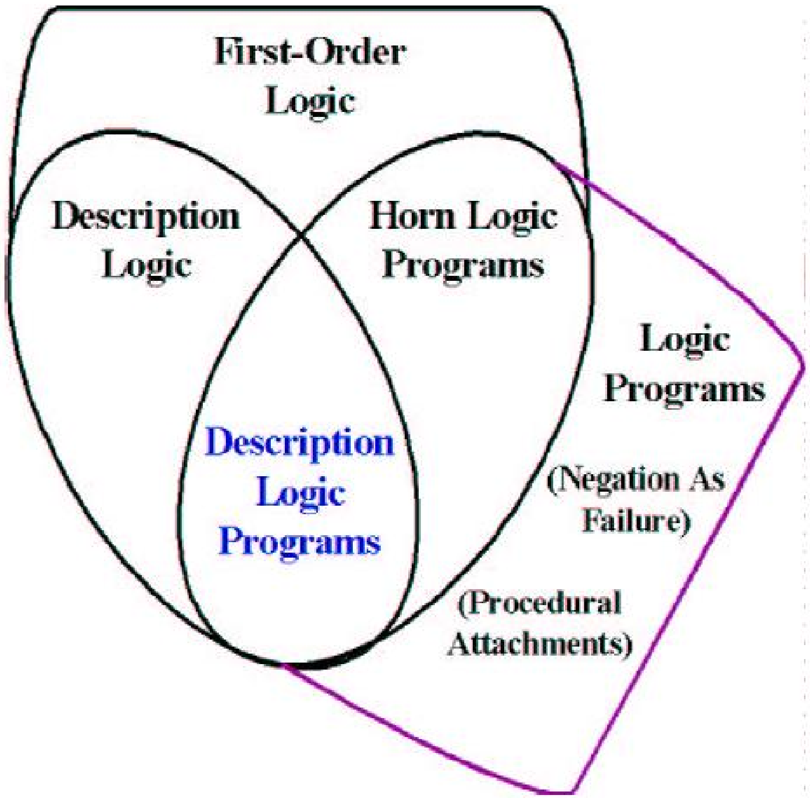
- Two forms of integrations:
  - Homogeneous integration.

    This is where rules are considered an integral part of the knowledge representation formalism used to encode ontologies.
  - Heterogeneous integration.

    This is where rules are not used to model ontologies, but rather used to communicate with ontologies in a more loose fashion. This can take the form of either layering rules on top of ontologies for rule applications, or for querying ontologies.

# Big picture

- Description Logic, Rules, and First Order Logic



- Semantics

Description Logic

Rules

Homogeneous

FOL semantics

FOL semantics

Completion semantics

well-founded semantics

stable model semantics

answer set semantics

Heterogeneous

Semantic Web

# Big picture

- Earlier integrations:
  - CARIN [Levy and Rousset, 1998]
  - AL-log [Donini et al., 1998]
  - Description Logic Programs (DLPs) [Grosof et al., 2003]
  - SWRL [Horrocks et al., 2004]
  - DL-safe rules [Motik et al., 2005]
  - hex-programs [Eiter et al., 2006]
  - HD-rules [Drabent and Maluszynski, 2007]
  - ELP [Krötzsch et al.,, 2008]
  - OWL 2 RL [Patel-Schneider et al., 2008].

# Big picture

Homogeneous
Heterogeneous

- Earlier integrations:
  - CARIN [Levy and Rousset, 1998]
  - AL-log [Donini et al., 1998]
  - Description Logic Programs (DLPs) [Grosof et al., 2003]
  - SWRL [Horrocks et al., 2004]
  - DL-safe rules [Motik et al., 2005]
  - hex-programs [Eiter et al., 2006]
  - HD-rules [Drabent and Maluszynski, 2007]
  - ELP [Krötzsch et al.,, 2008]
  - OWL 2 RL [Patel-Schneider et al., 2008].

# We investigate:

- Earlier integrations:
  - CARIN [Levy and Rousset, 1998]
  - AL-log [Donini et al., 1998]
  - **Description Logic Programs [Grosof et al., 2003]**
  - **SWRL [Horrocks et al., 2004]**
  - **DL-safe rules [Motik et al., 2005]**
  - **hex-programs [Eiter et al., 2006]**
  - HD-rules [Drabent and Maluszynski, 2007]
  - ELP [Krötzsch et al.,, 2008]
  - **OWL 2 RL [Patel-Schneider et al., 2008].**

# Description Logic Programs

- Description Logic Programs [Grosof et al., 2003]
  - an expressive intersection between rule formalisms and DLs
  - Example

    $C \sqsubseteq \forall R.D$ to $C(x), R(x, y) \Rightarrow D(y)$

  - Almost useless in both DL and LP

# DL-safe SWRL

- Idea: KR = OWL DL + Horn rules **?**

- Rules:

antecedent ⇒ consequent

"if antecedent holds, then the consequent also holds."

- Example

parent(?x,?y) , brother(?y,?z) ⇒ uncle(?x,?z)

"the composition of parent and brother properties implies the uncle property "

# DL-safe SWRL

- KR = OWL DL + Rules   is undecidable. Why?

  ↑ DL atoms

- DL-safe rule:
  - "Every variable in the rule must appear in a non-DL atom."
  - It ensures that rule apply only to individuals which are explicitly given in the knowledge base.
  - Herbrand-style way of interpreting them

- Example:

  O(?x), O(?y), O(?z), parent(?x,?y), brother(?y,?z) ) $\Rightarrow$ uncle(?x,?z)

# DL-safe SWRL

- KR = OWL DL + DL-safe Rules   is decidable

- Complexity:
  - exponential time for query answering in

    KB = SHIQ + DL-safe Rules

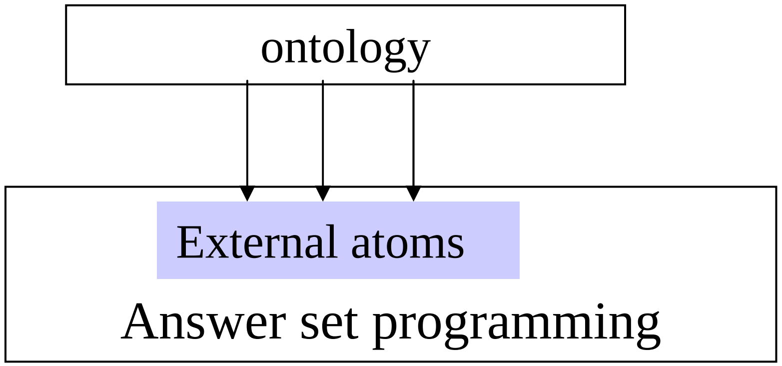
- Systems:
  - KAON2
  - Pellet

# Hex program

- Hex program:
  - A set of rules with negation as failure
  - Load ontology with external atoms
  - Answer set semantics
  - Heterogeneous integration
  - Using external computational source

# Hex program

- Hex program




- External atom

&g[Y1, …, Yn](X1,..,Xm)

&g: external predicate name

Y: input list

X: output list

# Hex program

- Example:

reached(x) :-  &reach[graph1; a](x)

It computes the predicate reached taking values from the predicate &reach, which computes via &reach[edge; a] all the reachable nodes in the graph graph1 from node a.

# Tutorial Outline

- Motivation
- OWL 2 and Semantic Web rules
  - The big picture
- More technical discussions on Semantic Web rules
- Practical: Hands-on Session

Semantic Web

# Practical Preparation

- Java SDK version 1.6
  - http://java.sun.com
- Protégé
  - http://protege.stanford.edu
- Files in the following USB folders
  - SWR-practical

# Part 1: OWL 2 RL

- Check the ontology syntax against the OWL 2 RL Profile using the syntax checker
  - Ontology URL: http://owl.man.ac.uk/2005/07/sssw/people.owl
  - Syntax Checker: http://dipper.csd.abdn.ac.uk:8080/OWL2ProfileChecker/
- There are several axioms which are not valid OWL 2 RL. However, an RL reasoner can perform *incomplete* reasoning over this ontologies.

# Completeness

- This section will give some examples of the sort of entailments can be made by OWL 2 RL reasoners.
  - Open the people.owl ontology in Protégé
  - Open a command prompt or terminal window.
  - Change to the `PracticalOWLandSWRL` directory.

Semantic Web

# Completeness (2)

- Open the file **query_animal-lovers.txt** in a text editor. This query should retrieve all instances of the class animal_lover.
- To run this query with Pellet type:
  - java -jar **PelletSWRL.jar** people.owl < *query_animal-lovers.txt*
- To run this query with Jena type:
  - java -jar **JenaOWL2RL.jar** people.owl < *query_animal-lovers.txt*
- In this case Jena gives incomplete results because of the lack of support for number restrictions in OWL 2 RL.
- Using Protégé, check the Class Description for the concept animal_lover.

# Completeness (3)

- Open the file **query_tabloid-newspapers.txt** in a text editor. This query should retrieve all instances of the class tabloid.
- To run this query with Pellet type:
  - java -jar **PelletSWRL.jar** people.owl < *query_tabloid-newspapers.txt*
- To run this query with Jena type:
  - java -jar **JenaOWL2RL.jar** people.owl < *query_tabloid-newspapers.txt*
- In this case Jena and Pellet give the same results.
- Using Protégé, check which instances are listed for tabloid.
- **The_Sun** is asserted to be an instance of **tabloid**. The **Daily_Mirror** is inferred to be a **tabloid**, since it is **read** by **Mick**, who is a **white_van_man.** The class description for **white_van_man** shows that they *only* **read tabloid** newspapers.
- Universal restrictions *are* supported by OWL 2 RL when used in this way.

# Exercise 1

- Now try **query_things-that-eat-bones.txt** with both reasoners.
- Pellet and Jena’s results differ. Can you explain why an OWL 2 RL reasoner could not find all answers to this query?

# Exercise 2

- Now try **query_white-van-man.txt** with both reasoners.
- Jena returns the correct answers for this query. Which OWL 2 RL axioms could have resulted in the entailment that **Mick** is a **white_van_man** ?

# Part 2: SWRL

- Open the dl-safe.owl ontology with Protégé.
- This ontology contains OWL classes, properties, individuals and SWRL rules.
- If the rules view is not displayed under any of the main tabs:
  - Select: *View* > *Ontology Views* > *Rules*
  - Click on one of the existing panes to display the Rules view.

# SWRL Example

- This rule asserts that a grand child is bad, if it hates another individual:
  - Grandchild(?x) , hates(?x, ?y) -> BadChild(?x)
- Open the file **query_bad-child.txt** in a text editor. This query will retrieve all BadChild individuals entailed by the ontology + rules.
- To run this query with Pellet type:
  - java -jar **PelletSWRL.jar** dl-safe.owl < *query_bad-child.txt*
- Note that OWL 2 RL reasoners do not directly support SWRL rules so we will not use Jena in this section.

# SWRL Example cont.

- The ontology contains an axiom which entails all People are of the class Grandchild, based on the restriction that all Person individuals have a father.
- The axiom responsible is not supported by the Protégé editor:
  - SubClassOf(ObjectSomeValuesFrom(father ObjectSomeValuesFrom(father Person)) Grandchild)
- Finally, for a person to be a BadChild then they must hate another individual. The individual view shows that both Romulus and Cain have a person who they hate, so are therefore instances of BadChild.

# Exercise 3

- Modify the ontology to include a rule that asserts instances of HappyChild.
  - You can add another property to the ontology such as *likes* and assert some of the Person instances to like another individual.
- You can check the entailed instances of HappyChild with Pellet:
  - java -jar **PelletSWRL.jar** dl-safe.owl < *query_happy-child.txt*

# Conclusion

- OWL 2 provide a family of languages with different levels of expressive power and complexity
  - Decidability
  - Tractability
- OWL 2 RL is not the intersection between OWL 2 DL and Horn rules
- Using internalisation, OWL 2 can represent many more rules
- Scalable reasoning services are needed for decidable rule extended ontology languages

# Resources

- W3C OWL WG homepage: http://www.w3.org/2007/OWL/wiki/OWL_Working_Group
- OWL 2 Profile specification: http://www.w3.org/TR/owl2-profiles/
- W3C RIF WG homepage: http://www.w3.org/2007/OWL/wiki/Test_Suite_Status#OWL_2_RL_Test_Cases

Some selected articles:


- M. Krötzsch, S. Rudolph, P. Hitzler. **Description Logic Rules.** In Proc. 18th European Conf. on Artificial Intelligence (ECAI 2008), IOS Press, 2008.