Introduction to Semantic Web Rule Language - SWRL

Bernard ESPINASSE
Aix-Marseille Université (AMU)
Polytech-Marseille
Nov. 2017

• From OWL to SWRL
• SWRL rules
• Examples of use of SWRL rules

Outline

1. Introduction to SWRL:
   • Interest of rules languages
   • Introduction to SWRL
   • SWRL and inference engines
2. SWRL rules
   • SWRL Predicates
   • Main types of SWRL rules
3. Examples of use of SWRL rules
   • Specifications (in Protégé and SWRL syntax)
   • Inferences with rules
4. Limits of SWRL (and OWL2)

References

• Books, articles and reports:
  • W3C, « OWL Web Ontology Language Semantics and Abstract Syntax »
  • ...

• Web W3C:
  • Page du W3C : http://www.w3.org/2004/OWL/
  • Reference : http://www.w3.org/TR/owl-ref/
  • Guide : http://www.w3.org/TR/owl-guide/
  • ...

• Course/tutorials:
  • Tutorial of Martin Kuba, Institute of Computer Science, http://dior.ics.muni.cz/~makub/owl/
  • ...

1. Introduction to SWRL
• Interest of rules languages
• Introduction to SWRL
• SWRL and inference engines
OWL and SWRL (1)

- **OWL language** is not able to express all relations:
  - Ex: it cannot express the relation « child of married parents »
  - -> because there is no way in OWL 2 to express the relation between individuals with which an individual has relations"
  - OWL expressivity can be extended by adding RULES to an ontology

- **A Rule language** is useful for different reasons:
  - **Add expressivity** to OWL: this addition have a price: loss of decidability
  - **Reuse of a set of existent rules**
  - In general it is easier to read and write rule with a rule language: true in general but not always

-> Semantic Web Rule Language - SWRL [Horrocks & al., 2004]

OWL and SWRL (2)

**Semantic Web Rule Language - SWRL** [Horrocks & al., 2004]:

- **SWRL** = acronym for « Semantic Web Rule Language »
- A proposal to combine ontologies and rules:
  - Ontologies : OWL-DL
  - Rules : RuleML (a rule language)
- **SWRL rules** are similar to rules in Prolog or DATALOG languages.
- **SWRL** = OWL-DL + RuleML :
  - OWL-DL: without variable
  - RuleML: use variables
- **Protege OWL editor** supports SWRL rules, as the Pellet and Hermit reasoners.
- Using **OWL API**, SWRL can be used un Java code

SWRL Rules (1)

- **SWRL allows handling of instance** by **variables** (?x, ?y, ?z)
- **SWRL not allows to create concepts and relations**
- **SWRL only allows to add relations** according to the variables values (individuale) and the rule satisfaction
- **SWRL are build according to the following schema**:
  - antecedent -> consequent
- **Antecedent = atoms conjunctions (∧)** and **Consequent = only one atom**
- An atom can be:
  - either an **concept instance**: C_i(z) = unary predicat
  - either an **OWL relation**: R_i(x, y) = binary predicat
  - either an SWRL relations: same-as(?x, ?y) or different-from (?x, ?y)

  Ex:

  $$R_1(x, y) \land \text{different-from}(x, y) \land C_1(z) \land \ldots \rightarrow R_m(x, z)$$

  Antecedent                       Consequent

SWRL Rules (2)

- a **SWRL rule functions according to the satisfiability principle of the antecedent or of the consequent**:
  - If the antecedent and the consequent are **defined**:
    - -> If the antecedent is satisfied then the consequent is also satisfied
  - If the antecedent is **empty**, this corresponds to an satisfied antecedent:
    - -> allows to define facts
  - If the consequent is **empty**, this corresponds to a satisfied consequent:
    - -> the antecedent have not to be satisfied
SWRL Rules (3)

Ex:
- Given in OWL the concept of Uncle defined as:
  
  \[\text{intersectionOf(SubClassOf(Man), isBrotherOf(Father))}\]
- and given the isUncleOf relation
- We know that a person is an uncle, but we do not know of whom: OWL not allows to define the isUncleOf relation representing the fact to be uncle of a person.
- With SWRL it is possible, it permits the description of this isUncleOf relation and rely the concerned instances:

  \[\text{a\Child(\text{x},\text{y}) \land isBrotherOf(\text{z},\text{x}) \rightarrow isUncleOf(\text{z},\text{y})}\]

SWRL Rules and decidability: DL-safe rules

- Arbitrary SWRL rules would lead to undecidability

  \[\rightarrow \text{so only so-called DL-safe rules are implemented in reasoners.}\]

- DL-safe rules are rules applied only to named individuals

- They do not apply to individuals that are not named but are known to exist.

SWRL and Inference Engine (Reasoners)

- Most of inference engines support SWRL:
  - Pellet, Bossam, Hool et, KAON2, RacerPro, R2ML (REWERSE Rule Markup Language) et Sesame.
  - According to 3 approaches:
    - Translate SWRL in first order logic (Hool et, …)
    - Translate OWL-DL in rules and apply an algorithm of forward chaining (Bossam, …)
    - Integrate SWRL rules in the inference engine OWL-DL based on semantic tableau algorithms (Pellet, …)

  However, SWRL current implementations need lot of calculus, they are only usable with small or medium ontologies.

SWRL rule & DL-Safe SWRL rules (1)

- Considering ontology (Turtle syntaxe):

  \[
  \begin{align*}
  \text{BrotherOfJoe} & \text{ a } \text{owl:Class}; \\
  \text{rdfs:subClassOf} & \{ \\
  & \text{a } \text{owl:Restriction}; \\
  & \text{owl:onProperty } :\text{hasBrother}; \\
  & \text{owl:hasValue } :\text{joe} \\
  \} . \\
  :\text{Dick} & \text{ a } \{ \\
  & \text{a } \text{owl:Restriction}; \\
  & \text{owl:onProperty } :\text{hasParent}; \\
  & \text{owl:someValuesFrom } :\text{BrotherOfJoe} \\
  \} . \\
  :\text{Joe} & \text{ a } :\text{Person} . \\
  \end{align*}
  \]

- The first part specify that a brother of Joe (:BrotherOfJoe) has necessarily a brother who is Joe
- The second part specify that Dick has a parent who is a brother of Joe

  \[\rightarrow \text{We would conclude that Joe is the uncle of Dick}\]
Introduction to SWRL

SWRL rule & DL-Safe SWRL rules (2)

- We could write the SWRL rule:
  
  
  
  \(\text{:hasParent}(?x,?y), \text{:hasBrother}(?y,?z)\)
  
  \(\rightarrow \text{:hasUncle}(?x,?z)\)

- If this rule is a normal SWRL rule, we obtain the attended conclusion

- If this rule is a DL-safe SWRL rule, we can apply it only if we can replace variables \(?x, ?y and \(?z by a named instance (named individual)

- But the ontology do not specify the role of \(?y.

- In fact a DL-safe SWRL rule functions as we have a other class :Nom, which instances are all named individuals

- DL-safe SWRL rule is equivalent to a normal SWRL rule as:
  
  
  \(\text{:hasParent}(?x,?y), \text{:hasBrother}(?y,?z), \text{:Nom}(?x), \text{:Nom}(?y), \text{:Nom}(?z)\)
  
  \(\rightarrow \text{:hasUncle}(?x,?z)\)

2. SWRL rules

- SWRL predicates
- Main types of SWRL Rules

SWRL predicates

- SWRL rules can use as predicates: class or property names
- SWRL rules can also use other predicates as:
  - class expressions: arbitrary class expressions, not just named classes
  - property expressions: the only operator available in OWL 2 for creating property expressions is inverse of object property
  - data range restrictions: specifies type of data value, like integer, date, union of some XML Schema types, enumerated type
  - sameIndividual and differentIndividuals: for specifying same and different individuals

and:

- core SWRL built-ins: special predicates defined in SWRL proposal which can manipulate data values, for example to add numbers
- custom SWRL built-ins: you can define your own built-ins using Java code

Main types of SWRL Rules

According to these possible predicates, the main SWRL rules types are:

- SWRL rules with class expressions
- SWRL rules with data range restrictions
- SWRL rules with SWRL core built-ins
- SWRL rules with custom SWRL built-ins (using Java code)
**SWRL Rules Syntaxes**

- **Protege syntax:**
  
  \[
  \text{aChild}(?x,?y) \land \text{isBrotherOf}(?z,?x) \rightarrow \text{isUncleOf}(?z,?y)
  \]

- **SWRL syntax (functional syntax):**

  ```
  DLSafeRule
  Annotation(rdfs:comment "Uncle rule")
  Body
  ClassAtom( :aChild Variable(var:x) Variable(var :y) )
  ClassAtom( :isBrotherOf Variable(var:z) Variable(var :x) )
  Head
  ClassAtom( :isUncleOf Variable(var:z) Variable(var :y))
  ```

**Example of SWRL Rules**

- **with class expressions**

  - **Protege syntax:**
    
    \[
    \text{Person}(?x), \text{hasChild min 1 Person}(?x) \rightarrow \text{Parent}(?x)
    \]

  - **Meaning:** individuals that are in the Person class, and that have at least one property hasChild with an individual from the class Person, must then be in the Parent class (in another words, that persons with at least one child are parents)

  - **SWRL syntax:**
    
    ```
    DLSafeRule
    Annotation(rdfs:comment "Rule with class expression")
    Body
    ClassAtom( :Person Variable(var:p) )
    ClassAtom(ObjectMinCardinality( 1 :hasChild :Person ) Variable(var:x) )
    Head
    ClassAtom( :Parent Variable(var:p) )
    ```

**Example of SWRL Rules with data restrictions**

- **Protege syntax:**

  \[
  \text{Person(?p), integer}[>= 18 , <= 65](?age), \text{hasAge}(?p, ?age) \rightarrow \text{hasDriverAge}(?p, \text{true})
  \]

- **Meaning:** The data range restriction is satisfied when the ?age variable has an integer value between 18 and 65 inclusive

- **In SWRL syntax (functional syntax):**

  ```
  DLSafeRule
  Annotation(rdfs:comment "Rule with data range restriction")
  Body
  ClassAtom(:Person Variable(var:p))
  DataPropertyAtom(:hasAge Variable(var:p) Variable(var:age))
  DataRangeAtom( DatatypeRestriction(xsd:integer xsd:minInclusive "18"^^xsd:integer xsd:maxInclusive "65"^^xsd:integer) Variable(var:age) )
  Head
  DataPropertyAtom( :hasDriverAge Variable(var:p) "true"^^xsd:boolean)
  ```

**3. Examples of use of SWRL Rules**

- **Specifications (in Protégé and SWRL syntax)**

- **Inferences with rules**
Examples of SWRL rules (1)

Ex:

- Valentin is the son of Berny and Sabine
- The symmetric property hasSpouse connects Berny and Sabine:

```
SymmetricObjectProperty(:hasSpouse)
ObjectPropertyAssertion(:hasSpouse :Berny :Sabine)
ObjectPropertyAssertion(:hasParent :Valentin :Berny)
ObjectPropertyAssertion(:hasParent :Valentin :Sabine)
```

Example of SWRL rules (2)

- Meaning: an individual X from the Person class, which has parents Y and Z such that Y has spouse Z, belongs to a new class ChildOfMarriedParents
- Protege syntax:

  ```
  Person(?x), hasParent(?x, ?y), hasParent(?x, ?z), hasSpouse(?y, ?z) -> ChildOfMarriedParents(?x)
  ```

  (In Protege, in Active Ontology onglet, in the Rule window, we add this expression)
- SWRL syntax (Functional syntax):

  ```
  Prefix(var:=<urn:swrl#>)
  Declaration( Class( :ChildOfMarriedParents ) )
  SubClassOf( :ChildOfMarriedParents :Person )
  DLSafeRule( Body( ClassAtom( :Person Variable(var:x)) ObjectPropertyAtom(:hasParent Variable(var:x) Variable(var:y)) ObjectPropertyAtom(:hasParent Variable(var:x) Variable(var:z)) ObjectPropertyAtom(:hasSpouse Variable(var:y) Variable(var:z)) ) Head( ClassAtom( :ChildOfMarriedParents Variable(var:x)) ) )
  ```

Examples of SWRL rules (3)

When we use the reasoner, it infers that Valentin belongs to the class ChildOfMarriedParents, and even explains why:

```
Explanation for: Valentin Type ChildOfMarriedParents
1) Valentin hasParent Berny In ALL other justifications
2) Valentin Type Person In ALL other justifications
3) Valentin hasParent Sabine In ALL other justifications
4) Berny hasSpouse Sabine In NO other justifications
5) Person(?x), hasParent(?x, ?y), hasParent(?x, ?z), hasSpouse(?y, ?z) -> ChildOfMarriedParents(?x) In ALL other justifications
```

Examples of SWRL rules (4)

Given an ontology defining their birthdays:

```
This ontology is written in functional syntax that Protege can load directly
```

This ontology is written in functional syntax that Protege can load directly.
Examples of SWRL rules (5)

The ontology also defines some rules:

Rule 1:

Rule 2:
DLSafeRule(Annotation(rdfs:comment "Rule with core built-in swrlb:greaterThan") Body(ClassAtom(:Person Variable(var:p)) DataPropertyAtom(:hasAge Variable(var:p) Variable(var:age)) BuiltInAtom(swrlb:greaterThan Variable(var:age) "18"^^xsd:integer) Head(ClassAtom(:Adult Variable(var:p)))))

Rule 3:
DLSafeRule(Annotation(rdfs:comment "Rule with data range restriction") Body(ClassAtom(:Person Variable(var:p)) DataPropertyAtom(:hasAge Variable(var:p) Variable(var:age)) DataRangeAtom(DatatypeRestriction(xsd:integer xsd:minInclusive "18"^^xsd:integer xsd:maxInclusive "65"^^xsd:integer) Variable(var:age)) Head(DataPropertyAtom(:hasDriverAge Variable(var:p) "true"^^xsd:boolean))))

Rule 4:

Examples of SWRL rules (6)

In Protégé, these rules are specified according to a variant of Manchester syntax without namespace qualifiers:

- R1 - Person(?p), swrlb:subtract(?age, ?nowyear, ?year), thisYear(?nowyear), bornInYear(?p, ?year) -> hasAge(?p, ?age)
- R2 - Person(?p), hasAge(?p, ?age), swrlb:greaterThan(?age, 18) -> Adult(?p)
- R3 - Person(?p), hasAge(?p, ?age), xsd:integer([>= 18, <= 65])(?age) -> hasDriverAge(?p, true)
- R4 - Person(?x), hasChild min 1 Person(?x) -> Parent(?x)
- R5 - Person(?p), swrlb:date(?date, ?year, ?month, ?day, ?timezone), bornOnDate(?p, ?date) -> bornInYear(?p, ?year)

Examples of SWRL rules (7)

- Protege syntax:
  Person(?x), hasChild min 1 Person(?x) -> Parent(?x)

- Meaning: individuals, that are in the Person class, and that have at least one property hasChild with an individual from the class Person, must then be in the Parent class (persons with at least one child are parents)

- Functional syntax:
  DLSafeRule(Annotation(rdfs:comment "Rule with class expression") Body(ClassAtom(Person Variable(var:x)) ClassAtom(Person Variable(var:x)) ObjectMinCardinality(1 :hasChild :Person) Variable(var:x)) Head(ClassAtom(Person Variable(var:x))))

Examples of SWRL rules (8)

- From Protégé, the reasoner (Pellet) will infer that Berny and Sabine are in the class Parent, as can be seen in the lower right corner:
Examples of SWRL rules (9)

- In Protégé when you click on the question mark (?) on right of the inferred information, the reasoner even provides information how that information was inferred:

4. Limits of SWRL (and OWL2)

- Limits of SWRL
- Limits of SWRL (and OWL2)

Limits of SWRL: expression power

- OWL 1 cannot express the «uncle» relation, which is a chain of relations parent and sibling (children of same parents)

- OWL 2 can express «uncle» relation using property chains, however it still cannot express relations between individuals referenced by properties.

- OWL 2 cannot express the “child of married parents” concept because it cannot express the relationship between parents of the individual

- SWRL can express the concept of “child of married parents”

Limits of SWRL: decidability

- However arbitrary SWRL rules would lead to undecidability

  => so only so-called **DL-safe** rules are implemented in reasoners.

- **DL-safe rules**:
  - are rules applied only to named individuals,
  - they do not apply to individuals that are not named but are known to exist.
Limits of OWL2 (and SWRL)

- Being a fragment of *first order predicate logic*, the OWL2 and SWRL cannot express (Kuba):
  - **fuzzy expressions**: “It *often* rains in autumn”
  - **non-monotonicity**: “Birds fly, penguin is a bird, but penguin *does not fly*”
  - **propositional attitudes**: “Eve *thinks* that 2 is not a prime number” (It is true that she thinks it, but what she thinks is not true.)

- **Modal Logics** could permit:
  - *possibility and necessity* - “It is *possible* that it will rain today.”
  - *epistemic modalities* - “Eve *knows* that 2 is a prime number.”
  - *temporal logic* - “I am *always* hungry.”
  - *deontic logic* - “You *must* do this.”