Chapter 6

Ontologies and Rules

by

Hanne Oustad & Øyvind Selmer

6.1 Intro - What is a rule?

- A rule of any type should consist of at least a premise and a conclusion, and in any situation where the premise applies, the conclusion must also hold.
- FOL, OWL DL, datalog
- Confusing terminology
- Familiar semantics

6.2.1 Introduction to datalog

- Datalog is an implication which contains conjunctions, constant symbols and universally quantified variables
- First developed for making queries to databases.
- Datalog syntax depends on three sets of symbols:
  - ***C****onstant symbols* are used as names to refer to certain elements of the domain of interest
  - ***V****ariables* are used as place holders for (arbitrary) domain elements to which rules might apply
  - ***P****redicate symbols,* or simply *predicates,* are used to denote relations between domain elements. May have one, multiple or none arguments
- The amount of arguments for a predicate is called the *arity* of the predicate
- Every set of datalog rules is based on a *signature:* (C,V,P)

Datalog rules can be applied *recursively*.

example: Vegetarian(x) ^ Fishproduct(y) -> dislikes(x,y)

6.2.1 Introduction to datalog

- Given the signature (C,V,P) we can build datalog rules as follows:
  - A datalog *term* is a constant symbol or a variable
  - A datalog *atom* is a formula of the form $p(t_1,...,t_n)$ given that $p \epsilon P$ is a predicate of arity n, and $t_1,...,t_n$ are terms
- A *datalog* rule is a formula of the form:
  - $\forall x_1 ... \forall x_m.(B_1 \wedge ... \wedge B_k \rightarrow H)$, where $B_1,...,B_k$ and H are datalog atoms, and $x_1,...,x_m$ are exactly the variables that occur within these atoms

Datalog program example:

(1) Vegetarian(x) ^ Fishproduct(y) -> dislikes(x,y)

(2) orderedDish(x,y) ^ dislikes(x,y) -> Unhappy(x)

(3) orderedDish(x,y) -> Dish(y)

(4) dislikes(x,z) ^ Dish(y) ^ contains(y,z) -> dislikes(x,y)

(5) -> Vegetarian(markus)

(6) Happy(x) ^ Unhappy(x) ->

6.3 Combining rules with OWL DL

- Would such a combination be meaningful?
- Interpretations of datalog and DL are closely related
  - constant symbol = individual name
  - unary and binary predicates = class names and role names

- Using combination of datalog rules and description logic ontology:
  - datalog interpretations over a datalog signature.

6.3.3 Description Logic Rules

Which datalog rules could be directly represented as description logic axioms?

- $\mathcal{SROIQ}$(the basis for OWL2 DL) can express significantly more rules than the description logic $\mathcal{SHOIN}$(the basis for OWL DL).

Datalog: Person(x) ^ authorOf(x,y) ^ Book(y) -> Bookauthor(x)
Description logic: $\text{Person} \sqcap \exists \text{authorOf.Book} \sqsubseteq \text{Bookauthor}$

Mixed: (Person $\sqcap \exists$ authorOf.Book)(x) -> Bookauthor(x)

- Rewriting atoms as description logic class expressions is called *rolling-up.*

6.3.3 Description Logic Rules

orderedDish(*x,y*) ^ dislikes(*x,y*) ^ -> unhappy(x).

- Rolling up is only possible if a variable is "reachable" by only a single binary predicate

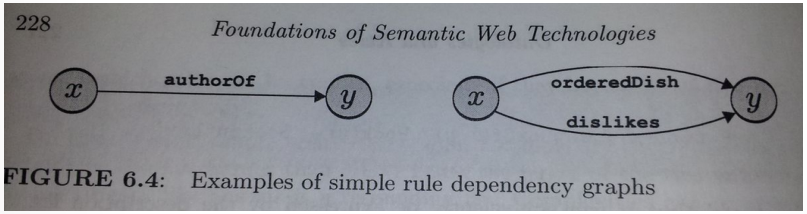

FIGURE 6.4: Examples of simple rule dependency graphs

- if *R(x,y)* is an atom in *B,* then {*R(x,y)*} is a path between *x* and *y.*
- if *p* is a path between *x* and *y,* then *p* is also a path between *y* and *x.*
- if *p* is a path between *x* and *y, q* is a path between *y* and *z,* and no atom occurs both in *p* and in *q,* then *p* U *q* is a path between *x* and *z.*

6.3.3 Description Logic Rules

- Now we can transform a datalog rule into a semantically equal set of axioms of the description logic $\mathcal{SROIQ}$ if the following conditions hold:
  - The rule contains only unary and binary predicates.
  - For any two variables *x* and *y*, there is at most a single path between *x* and *y* in the premise of the rule.

6.3.4 DL-safe Rules

- Based on the idea of limiting the interaction between datalog and description logics.
- We call a datalog atom a *DL-atom* if its predicate is used as a role or class name in a concrete DL knowledge base *K*(all other atoms are called *non-DL-atoms*).
- *B -> H* is DL-safe for *K* if all variables occurring in *B -> H* also occur in a non-DL-atom in the body *B.*
- A set of datalog rules is DL-safe for *K* if all of its rules are DL-safe for *K.*

6.3.4 DL-safe Rules

(1) Vegetarian(x) ^ Fishproduct(y) -> dislikes(x,y)

(2) orderedDish(x,y) ^ dislikes(x,y) -> Unhappy(x)

(3) orderedDish(x,y) -> Dish(y)

(4) dislikes(x,z) ^ Dish(y) contains(y,z) -> dislikes(x,y)

(5) -> Vegetarian(markus)

(6) Happy(x) ^ Unhappy(x) ->

---------- description logic (*K*)----------

(7) orderedDish.ThaiCurry(markus)

(8) ThaiCurry $\sqsubseteq \exists$ contains.FishProduct

- Rule(1) is not DL-safe since *y* is used only in the DL-atom FishProduct(*y*).
- The same goes for rule(3)
- The other rules are DL-safe.

6.4 Rule Interchange Format (RIF)

- RIF is a general framework for exchanging rules of many different kinds
- RIF-Core is the the basic core of the rule languages considered by RIF
- RIF also defines the two dialects: Basic Logic Dialect (BLD) and Production Rule Dialect(PDR)
- RIF-Core is closely related to datalog

RIF-Core example

Reconstructs rule(1) in Fig 6.1:

```
Vegetarian(x)^FishProduct(y) -> dislikes(x,y)
```

RIF-Core document with this rule:

```
Document(
   Prefix(ex http://example.com/)
   Group(
      Forall ?x ?y (
         ex:dislikes(?x ?y) :- And( ex:Vegetarian(?x) ex:FishProduct(?y) )
      )
   )
)
```

RIF-Core input	datalog input
Forall	∀
?x_1.....?x_n (Variables)	x_1.....x_n
head :- body	body -> head
And(content$_1$....content$_n$)	(content$_1$) ^.......^ (content$_n$)

Example of a rule that is not allowed in a well-formed document:

```
Forall ?y(
   ex:dislikes(ex:dislikes ?y) :- ex:FishProduct(?y)
)
```

Frames in RIF

- Frames provide a syntax for specifying property values of objects
- specify data structures in an "object oriented" fashion

Example of a fact encoded in a frame: `book:uri[ex:publishedBy -> crc:uri]`

RDF triples: `subject[predicate -> object]`

Rewrites the frame expression: `a[b->c  b->d  e->f]`

into: `a[b->c]  a[b->d]  a[e->f]`

encoding in datalog: `triple(a,b,c) ^ triple(a,b,d) ^ triple(a,e,f)`

XML Syntax for RIF-Core

- The syntactic form that RIF suggests for encoding rules is based on XML

RIF presentation syntax: input	RIF XML syntax: X(input)
Forall ?x_1.....?x_n (content)	<Forall> <declare> X(?x_1)</declare> ... <declare> X(?x_n)</declare> <formula> X(content)</formula> </Forall>

Combining RIF with OWL DL

- RIF rule bases can be combined with OWL DL

Example of importing OWL DL ontologies into RIF documents:

```
Document(
   Import(
      <location>
      <http://www.w3.org/2007/rif-import-profile#OWL-DL>
   )
   ...
)
```

Combining RIF with RDF(S)

- RIF rules can also be combined with RDF data bases

- Semantics that can be used with RDF(S):
  - Simple Semantics
  - RDF Semantics
  - RDFS Semantics