Logic and Inference: Rules


Jouni Tuominen
Department of Computer Science
jouni.tuominen@aalto.fi
Contents

• Introduction to logic
• Rule languages: Horn logic
• Rules on the Semantic Web
• Ontologies vs. logical rules
• Nonmonotonic rules (on separate slides)
Introduction to logic
The Semantic Web technology stack
"layer cake model"

- Metadata
- Vocabulary/ontology
- Inference/logic

XML
Namespaces
URI
Unicode

Logic framework
Rules
Trust
Proof
Signature
Encryption

RDF M&S
RDF Schema
Ontology

(Tim Berners-Lee)
Newer version of the cake model
The importance of logic

- High-level language for expressing knowledge
- High expressive power
- Well-understood formal semantics
- Precise notion of logical consequence
- Proof systems can automatically derive statements syntactically from a set of premises
- Sound & complete proof systems exist
  - *First order predicate logic*
  - *Not necessarily available for more expressive logics*
- Logic can provide **explanations** for answers
  - *Trace the proof that leads to a logical consequence*
First order predicate logic: syntax

Sentence $\rightarrow$ AtomicSentence
| Sentence Connective Sentence
| Quantifier Variable Sentence
| $\neg$Sentence
| ($\neg$Sentence)

AtomicSentence $\rightarrow$ Predicate(Term, Term, ...)
| Term=Term

Term $\rightarrow$ Function(Term,Term,...)
| Constant
| Variable

Connective $\rightarrow$ $\vee$ | $\wedge$ | $\Rightarrow$ | $\Leftrightarrow$

Quantifier $\rightarrow$ $\exists$ | $\forall$

Constant $\rightarrow$ A | John | Carl

Variable $\rightarrow$ x | y | z | ...

Predicate $\rightarrow$ Brother | Owns | ...

Function $\rightarrow$ father-of | plus | ...

(R. Mooney)
Sentences in First-Order Logic

- An atomic sentence is simply a predicate applied to a set of terms.

  \text{Owns}(\text{John}, \text{Car1})
  \text{Sold}(\text{John}, \text{Car1}, \text{Fred})

  Semantics is True or False depending on the interpretation, i.e. is the predicate true of these arguments.

- The standard propositional connectives (\lor, \land, \Rightarrow) can be used to construct complex sentences:

  \text{Owns}(\text{John}, \text{Car1}) \lor \text{Owns}(\text{Fred, Car1})
  \text{Sold}(\text{John, Car1, Fred}) \Rightarrow \neg \text{Owns}(\text{John, Car1})

  Semantics same as in propositional logic.
Quantifiers

● Allows statements about entire collections of objects rather than having to enumerate the objects by name.

● Universal quantifier: \( \forall x \)
  Asserts that a sentence is true for all values of variable \( x \)

\[
\forall x \text{ Loves}(x, \text{ FOPC})
\forall x \text{ Whale}(x) \Rightarrow \text{ Mammal}(x)
\forall x \text{ Grackles}(x) \Rightarrow \text{ Black}(x)
\forall x (\forall y \text{ Dog}(y) \Rightarrow \text{ Loves}(x,y)) \Rightarrow (\forall z \text{ Cat}(z) \Rightarrow \text{ Hates}(x,z))
\]

● Existential quantifier: \( \exists \)
  Asserts that a sentence is true for at least one value of a variable \( x \)

\[
\exists x \text{ Loves}(x, \text{ FOPC})
\exists x (\text{ Cat}(x) \land \text{ Color}(x, \text{ Black}) \land \text{ Owns}(\text{ Mary}, x))
\exists x (\forall y \text{ Dog}(y) \Rightarrow \text{ Loves}(x,y)) \land (\forall z \text{ Cat}(z) \Rightarrow \text{ Hates}(x,z))
\]
Logical KB

- KB contains general **axioms** describing the relations between predicates and **definitions** of predicates using $\Leftrightarrow$.

\[
\forall x, y \text{ Bachelor}(x) \Leftrightarrow \text{Male}(x) \land \text{Adult}(x) \land \neg \exists y \text{Married}(x,y).
\forall x \text{ Adult}(x) \Leftrightarrow \text{Person}(x) \land \text{Age}(x) \geq 18.
\]

- May also contain specific ground facts.

  Male(Bob), Age(Bob)=21, Married(Bob, Mary)

- Can provide **queries** or **goals** as questions to the KB:

  Adult(Bob)  
  Bachelor(Bob)  

- If query is existentially quantified, would like to return **substitutions** or **binding lists** specifying values for the existential variables that satisfy the query.

\[
\exists x \text{ Adult}(x)  
\exists x \text{ Married}(Bob,x)  
\{x/\text{Bob}\}  
\{x/\text{Mary}\}
\]

\[
\exists x, y \text{ Married}(x,y)  
\{x/\text{Bob}, y/\text{Mary}\}
\]

(R. Mooney)
Semantics of predicate logic

A predicate logic model consists of:
• a domain dom(A), a nonempty set of objects about which the formulas make statements
• an element from the domain for each constant
• a concrete function on dom(A) for every function symbol
• a concrete relation on dom(A) for every predicate

The meanings of the logical connectives ¬, ∨, ∧, →, ∀, ∃ are defined according to their intuitive meaning:
• not, or, and, implies, for all, there is
• We define when a formula is true in a model A, denoted as $A \models \varphi$
• A formula $\varphi$ follows from a set $M$ of formulas if $\varphi$ is true in all models $A$ in which $M$ is true
Why is predicate logic not enough?

Predicate logic is not decidable and not efficient
• There is no effective method to answer whether an arbitrary formula is logically valid

Solution: restriction to a reasonable subset of predicate logic
• Balancing between expressiveness and computational complexity (remember OWL Full vs. OWL DL)

→ Description logics and Horn logic
Description logics (DL)

- Family of formal knowledge representation languages used in ontology modeling
- Describe relations between entities in a domain of interest
  - Concepts (classes), roles (properties), individual names (individuals)
- Knowledge base is divided into TBox, RBox, and ABox
  - **TBox**: terminology (relations between concepts), e.g., “All students are persons”
    - **Student** ⊆ **Person** (concept inclusion)
  - **RBox**: role relationships, e.g., “parentOf is a subrole of ancestorOf”
    - **parentOf** ⊆ **ancestorOf** (role inclusion)
  - **ABox**: assertions about individuals, e.g., “John is a student”, “John is a parent of Lisa”
    - **Student(john)**, **parentOf(lisa, john)**
DL constructors for concepts and roles

• OWL DL is based on the description logic called **SROIQ**
• Concept and role inclusion, concept and role equivalence, Boolean operations, quantification, cardinality restrictions
• Concept expressions $C$, role expressions $R$, and named individuals $N_I$

$$C ::= N_C | (C \cap C) | (C \cup C) | \neg C | \top | \bot | \exists R.C | \forall R.C | \geq n R.C | \leq n R.C | \exists R.Self | \{N_I\}$$

**ABox:**
- $C(N_I)$
- $R(N_I, N_I)$
- $N_I \approx N_I$
- $N_I \not\approx N_I$

**TBox:**
- $C \sqsubseteq C$
- $C \equiv C$

**RBox:**
- $R \sqsubseteq R$
- $R \equiv R$
- $R \circ R \sqsubseteq R$
- $\text{Disjoint}(R, R)$

(Krötzsch et al., 2013)
Rule languages: Horn logic
A rule (clause) has the form: $A_1, \ldots, A_n \rightarrow B$

- $A_1, \ldots, A_n$ (body) is a conjunction of atomic formulas
- $B$ (head) is an atomic formula

There are 2 ways of reading a rule:

- **Deductive rules**: If $A_1, \ldots, A_n$ are known to be true, then $B$ is also true
- **Reactive (procedural) rules**: If the conditions $A_1, \ldots, A_n$ are true, then carry out the action $B$

Examples of Rules

- $\text{male}(X), \text{parent}(P,X), \text{parent}(P,Y), \text{notSame}(X,Y) \rightarrow \text{brother}(X,Y)$
- $\text{female}(X), \text{parent}(P,X), \text{parent}(P,Y), \text{notSame}(X,Y) \rightarrow \text{sister}(X,Y)$
- $\text{brother}(X,P), \text{parent}(P,Y) \rightarrow \text{uncle}(X,Y)$
- $\text{mother}(X,P), \text{parent}(P,Y) \rightarrow \text{grandmother}(X,Y)$
- $\text{parent}(X,Y) \rightarrow \text{ancestor}(X,Y)$
- $\text{ancestor}(X,P), \text{parent}(P,Y) \rightarrow \text{ancestor}(X,Y)$
Facts (rules without a body)
• → male(John)
• → male(Bill)
• → female(Mary)
• → female(Jane)
• → parent(John, Mary)
• → parent(John, Bill)
...

Queries / Goals (as rule bodies)
• parent(John, X), female(X) →
• grandmother(X,Y) →
A query is proved by deriving a conflict from it (proof by contradiction)

- Solutions: value substitutions for variables
  - $X=Mary$;
  - $X=Alice, Y=Jill; X=George, Y=Susan$;

RDF properties can be seen as binary predicates!
Application example: recommendation of similar items in MuseumFinland

Pullonsuoja, 2 kpl:istuva koira

Materiali: väinapullo: last, pullonsuoja: lanka
Valmistaja: Karhulan lastehdas, Tapio Wirkola
Valmistuslaite: väinapullo: tehdasvalmistuinen, pullonsuoja: käsitöitä
Käyttäjä: Eero Kallio
Käyttöpaikka: Etelä-Suomen lääni, Suomi
Asiasta: ALKOHOLIJUOMAT, ELÄINHAHMOT, KORISTE-ESINEET
Mität: pullon pohjan halkaisija 6,5cm, korkeus 22,5cm, pullonsuojuksen korkeus 29,0cm
Museotutkimus: LAHDE HISTORIALLISEN MUSEO

Sama käyttäjä
Eero Kallio:
- Keräilykortti, 14 kpl:erotetunlajien osakkeita
- Kallio, 4 kpl:jalavälineet
- Päättänyt, miehen turtikäsikka, 'tuulka'
- Talvifikina, miehenmusta kuvallinen kovikke
- Talvifikina, miehenkoti kuvallinen kovikke

Samaan aiheeseen liittyviä esineitä
alkoholijuomat:
- kannisteritaskumäki
- kannisteritaskumäki
- kannisteritaskumäki
- vinapullo: Isispullo
- pullo: Isispullo

eläimet:
- kuvakirja, kuvakirja, kangasta
- helistin, puralehu
- maakuvaharjuna, karhuendu
- saastolipesantieppa
- maltavarsut
Rules on the Semantic Web
Many different approaches in use

Rule formats
• RuleML, Rule Interchange Format (RIF), ...

Logic programming using RDF data
• E.g., SWI Prolog

OWL RL
• Rule-based implementation of OWL is possible
• Mixing rules and OWL

Semantic Web Rule Language SWRL
• Certain kind of rich rules can be used in OWL DL

SPARQL-based rules
• SPARQL Inference Notation SPIN
Rule Markup Language RuleML

Standardized XML notation for rules

\[
\text{hasParent}(x1, x2) \land \text{hasBrother}(x2, x3) \Rightarrow \text{hasUncle}(x1, x3)
\]

```
<ruleml:imp>
  <ruleml:_body>
    <swrlx:individualPropertyAtom  swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom  swrlx:property="hasBrother">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom  swrlx:property="hasUncle">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```
Rule Interchange Format RIF

Goals

• To define:
  – First, a shared Core for rule systems
  – Then, application-specific extensions (dialects)

• Rule transformation / exchange between different rules systems

• This way systems can understand each other’s operation logic

Based heavily on RuleML

Latest W3C recommendation on 5.2.2013
RIF dialects

RIF Core
• Common core of all RIF dialects
• Essentially function-free Horn logic (Datalog)
• Syntactic extensions
  – frames (syntactic sugar), IRIs, XML datatypes, built-ins (e.g., for numeric comparison)

RIF Basic Logic Dialect (BLD)
• Essentially Horn logic with equality, based on RIF Core
• Compatibility with RDF and OWL (RL)

RIF Production Rule Dialect (PRD)
• Reactive rules with procedural attachment
• Then part (head) of the rule contains actions
RIF example

Document{
  Prefix(rdf <http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
  Prefix(rdfs <http://www.w3.org/2000/01/rdf-schema#>)
  Prefix(imdbrel <http://example.com/imdbrelations#>)
  Prefix(dbpedia <http://dbpedia.org/ontology>)

  Group{
    Forall ?Actor ?Film ?Role {
      If And(rdf:type(?Actor imdbrel:Actor)
      rdf:type(?Film imdbrel:Film)
      rdf:type(?Role imdbrel:Character)
      imdbrel:playsRole(?Actor ?Role)
      imdbrel:roleInilm(?Role ?Film))
      Then dbpedia:starring(?Film ?Actor)
    }
  }
}
Semantic Web Rule Language SWRL

• Proposed combination of function-free Horn logic and OWL DL
• Rule form: $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$
  - Atom forms: $C(x)$, $P(x,y)$, sameAs($x,y$), differentFrom($x,y$)
    • $C(x)$: OWL description
    • $P$: OWL property
    • $x$ and $y$: individuals, variables, or data values
• Main difficulty: restrictions for $A_i$ and $B_j$ needed for decidability
  - A prominent solution: DL-safe rules
    • Every variable must appear in a non-description logic atom in the rule body ($P(x,y)$ in $A_i$)
• OWL RL = low-end solution, SWRL high-end solution in integrating rules and DLs
SWRL example

OWL cannot express an axiom “a person whose parents are married, is a child of married parents”

- SWRL rule expressed in OWL Functional-style syntax (can also be expressed in other OWL/RDF syntaxes and RuleML):

```
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) )
  )
)
```

(Kuba, 2012)
SPARQL Inference Notation SPIN

**SPIN – SPARQL syntax**
- Proposed format for representing SPARQL in RDF
- Allows storage, maintenance, and sharing of queries
- Schema (RDF specification) in the namespace URI: http://spinrdf.org/sp#

**SPIN – Modeling Vocabulary**
- Format for linking classes with SPIN SPARQL expressions
- Expression applied to all instances of the class (rules, logical constraints)
- Schema (RDF specification) in the namespace URI: http://spinrdf.org/spin#

**Modularization**
- Extending the language: templates, functions, magic properties

E.g., OWL RL can be implemented using SPIN
For example, the SPARQL query

```sparql
# must be at least 18 years old
ASK WHERE {
    ?this my:age ?age .
    FILTER (?age < 18) .
}
```

can be represented by a blank node in the SPIN RDF Syntax in Turtle as

```turtle
[ a sp:Ask ;
    rdfs:comment "must be at least 18 years old"^^xsd:string ;
    sp:where ([
        sp:object sp:_age ;
        sp:predicate my:age ;
        sp:subject spin:_this
    ] [ a sp:Filter ;
        sp:expression
        [ sp:arg1 sp:_age ;
            sp:arg2 18 ;
            a sp:lt
        ]
    ])
]
```
Example of a rule using CONSTRUCT

New triples visible for the next rule (not inserted into data, but added into a special “inferences” graph)

```sparql
ex:Person
  a rdfs:Class ;
  rdfs:label "Person"^^xsd:string ;
  rdfs:subClassOf owl:Thing ;
spin:rule
  [ a sp:Construct ;
    sp:templates { [ sp:object sp:_grandParent ;
      sp:predicate ex:grandParent ;
      sp:subject spin:_this
    ]
    sp:where { [ sp:object spin:_this ;
      sp:predicate ex:child ;
      sp:subject sp:_parent
    } [ sp:object sp:_parent ;
      sp:predicate ex:child ;
      sp:subject sp:_grandParent
    ]
  ] .
```

In textual SPARQL syntax, the above query would read as:

```sparql
CONSTRUCT {
}
WHERE {
  ?parent ex:child ?this .
}
```
Ontologies vs. logical rules
Horn logic vs. description logics

- E.g., how to represent rules in description logics?
- E.g., how to represent cardinality constraints in Horn logic?
Logics of the Semantic Web

HLP = FOL & LP
DLP = DL & HLP

(Antoniou, van Harmelen, 2007)
Description Logic Programs

• Description Logic Programs (DLP) can be considered as the intersection of Horn logic and description logic

• DLP allows to combine advantages of both approaches, e.g.:
  – A modeler may take a DL view, but
  – The implementation may be based on rule technology
Unique Names Assumption UNA

- Resources are different/same if they have different/same identifiers
- UNA made in logic programming & databases but not in logic
- Sometimes makes sense, sometimes not
  - E.g., T. Halonen, Tarja H., 190446-987X, 190446-767D

Closed World Assumption CWA

- If a fact cannot be deduced it is assumed to be false
- CWA made in logic programming & databases but not in logic
- Sometimes makes sense, sometimes not
  - Did it rain in Tokyo yesterday?
    - CWA would answer (possibly) incorrectly no
  - Was there a tsunami in Tokyo yesterday?
    - CWA would answer correctly no
An interoperability problem

Logic programming & databases usually assume
• UNA + CWA

Description logics & theorem proving do not assume
• UNA + CWA

Result: different conclusions are drawn from same premises
• Interoperability is lost
  - Predicate logic is monotonic: if a conclusion can be drawn, it remains valid even if new knowledge becomes available
  - CWA leads to nonmonotonic behaviour: addition of new information can lead to a loss of a consequence

Compromise approaches
The semantics of the Semantic Web is based on different subsets of the first-order predicate logic

- The core of RDF has logical semantics
- OWL is a formal description logic
- Rules are based on logic

Languages can be used more freely for partial reasoning, even though the entire system would not be formally decidable

- By defining one’s own rules for expressions and RDF graphs
- By limiting oneself to simple structures (e.g., the core RDFS)

Challenges of the standardization work

- UNA and CWA assumptions
  - Practice: logic programming and databases vs. Theory: description logics and classical theorem proving
- How to combine description logics and rule-based reasoning
Nonmonotonic rules
(based on the textbook slides
by G. Antoniou and F. van Harmelen:
see separate slides)
Summary

- Horn logic is a subset of predicate logic that allows efficient reasoning, orthogonal to description logics
- Horn logic is the basis of monotonic rules
- Nonmonotonic rules are useful in situations where the available information is incomplete
  - Rules that may be overridden by contrary evidence
  - Priorities are used to resolve some conflicts between rules
- Rules on the semantic web come in many forms using different assumptions
  - Interoperability between different logic systems is difficult
What’s next on this course?

Lectures on Wednesdays at 10.15-12.00 in lecture hall 2534-2535 (TUAS)

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.01.2016</td>
<td>Introduction, WWW, and Semantic Web</td>
</tr>
<tr>
<td>20.01.2016</td>
<td>RDF and RDF Schema</td>
</tr>
<tr>
<td>27.01.2016</td>
<td>Linked Data and SPARQL</td>
</tr>
<tr>
<td>03.02.2016</td>
<td>Ontologies and SKOS</td>
</tr>
<tr>
<td>10.02.2016</td>
<td>Web Ontology Language OWL</td>
</tr>
<tr>
<td>24.02.2016</td>
<td>Logic and Inference: Rules</td>
</tr>
<tr>
<td><strong>02.03.2016</strong></td>
<td><strong>Semantic Web Infrastructure and Applications</strong></td>
</tr>
</tbody>
</table>

Assignment support sessions on Thursdays at 10.15-12.00 in computer class 1521-1522 (TUAS)

<table>
<thead>
<tr>
<th>Date</th>
<th>Support for</th>
</tr>
</thead>
<tbody>
<tr>
<td>28.01.2016</td>
<td>Assignment 1</td>
</tr>
<tr>
<td>04.02.2016</td>
<td>Assignment 1</td>
</tr>
<tr>
<td>11.02.2016</td>
<td>Assignment 2</td>
</tr>
<tr>
<td>25.02.2016</td>
<td>Assignment 2</td>
</tr>
<tr>
<td>03.03.2016</td>
<td>Assignment 2</td>
</tr>
<tr>
<td>10.03.2016</td>
<td>Assignment 3</td>
</tr>
<tr>
<td>17.03.2016</td>
<td>Assignment 3</td>
</tr>
<tr>
<td>24.03.2016</td>
<td>Assignment 3</td>
</tr>
</tbody>
</table>

04.02. DL for Assignment 1
03.03. DL for Assignment 2
24.03. DL for Assignment 3