Aalto University
School of Science

SeCo

Logic and Inference: Rules

ME-E4300 Semantic Web, 20.2.2019

Eero Hyvönen
Aalto University, Semantic Computing Research Group (SeCo) *http://seco.cs.aalto.fi*
University of Helsinki, HELDIG *http://heldig.fi*

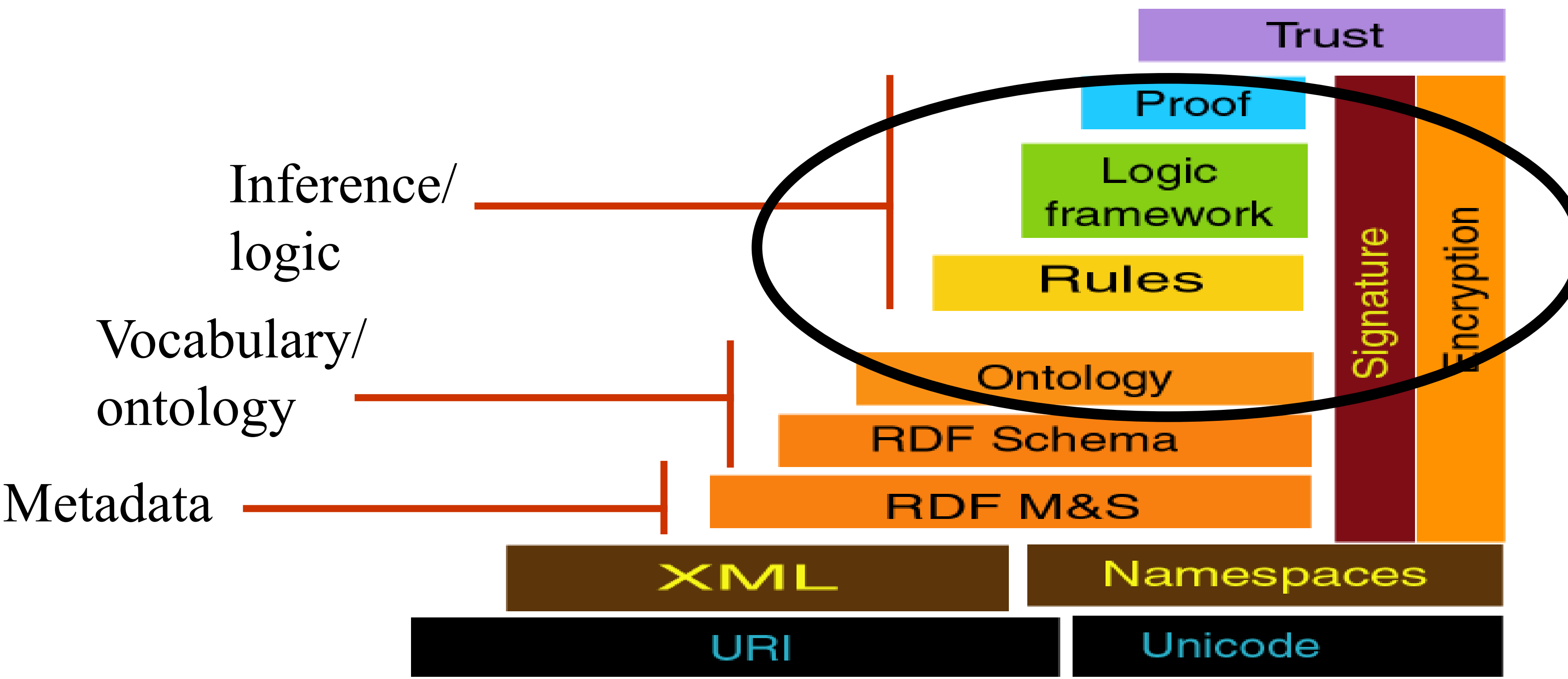
eero.hyvonen@aalto.fi

Contents

- Introduction to logic
- Rule languages: Horn logic
- Rules on the Semantic Web
- Ontologies vs. logical rules
- Nonmonotonic rules (on separate slides)

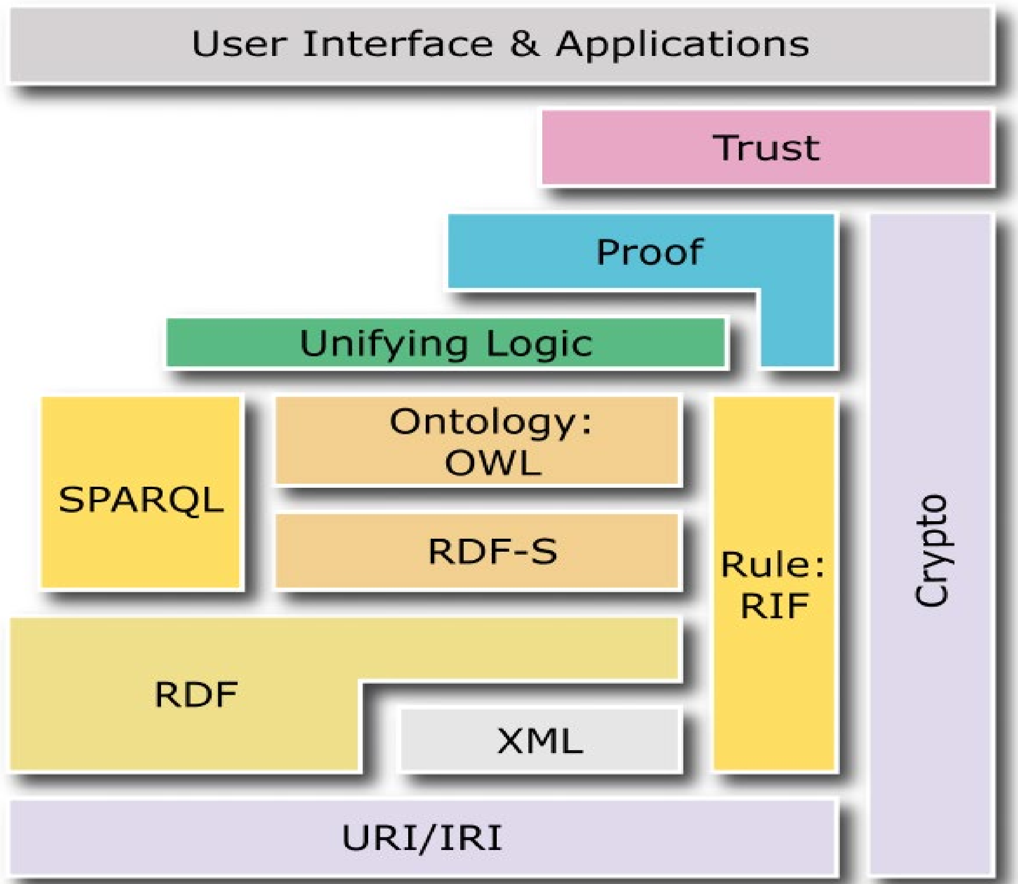
Introduction to logic

The Semantic Web technology stack ”layer cake model” (original)



(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 → AtomicSentence
| Sentence Connective Sentence
| Quantifier Variable Sentence
| ¬Sentence
| (Sentence)

AtomicSentence → Predicate(Term, Term, ...)
| Term=Term

Term → Function(Term,Term,...)
| Constant
| Variable

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

Quanitfier → $\exists$ | $\forall$

Constant → A | John | Car1

Variable → x | y | z |...

Predicate → Brother | Owns | ...

Function → father-of | plus | ...

(R. Mooney)

Sentences in First-Order Logic

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

  Owns(John,Car1)
  Sold(John,Car1,Fred)

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

- The standard propositional connectives ($\vee$ $\neg$ $\wedge$ $\Rightarrow$ $\Leftrightarrow$) can be used to construct complex sentences:

  Owns(John,Car1) $\vee$ Owns(Fred, Car1)
  Sold(John,Car1,Fred) $\Rightarrow$ $\neg$Owns(John, Car1)

  Semantics same as in propositional logic.

(R. Mooney)

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

  $\forall x$ Loves(x, FOPC)
  $\forall x$ Whale(x) $\Rightarrow$ Mammal(x)
  $\forall x$ Grackles(x) $\Rightarrow$ Black(x)
  $\forall x$ ($\forall y$ Dog(y) $\Rightarrow$ Loves(x,y)) $\Rightarrow$ ($\forall z$ Cat(z) $\Rightarrow$ Hates(x,z))

- Existential quantifier: $\exists$
  Asserts that a sentence is true for at least one value of a variable x

  $\exists x$ Loves(x, FOPC)
  $\exists x$(Cat(x) $\wedge$ Color(x,Black) $\wedge$ Owns(Mary,x))
  $\exists x$($\forall y$ Dog(y) $\Rightarrow$ Loves(x,y)) $\wedge$ ($\forall z$ Cat(z) $\Rightarrow$ Hates(x,z))

(R. Mooney)

Logical KB

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

  ∀x,y Bachelor(x) ⇔ Male(x) ∧ Adult(x) ∧ ¬∃yMarried(x,y).
  ∀x Adult(x) ⇔ Person(x) ∧ Age(x) >=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.

  ∃x Adult(x) ?
  {x/Bob}

  ∃x Married(Bob,x) ?
  {x/Mary}

  ∃x,y Married(x,y) ?
  {x/Bob, y/Mary}

(R. Mooney)

Semantics of predicate logic

A predicate logic model (interpretation) 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 |= φ

$\exists x\ Loves(x, FOPC)$
$\exists x(Cat(x) \wedge Color(x,Black) \wedge Owns(Mary,x))$
$\exists x(\forall y\ Dog(y) \Rightarrow Loves(x,y)) \wedge (\forall z\ Cat(z) \Rightarrow Hates(x,z))$

Important concepts

- **Satisfiable φ: If there is a model in which φ is true**
- **Valid φ: If φ is true all models (cf. tautologies)**
- **Logical consequence: φ follows from a set M of formulas if φ 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** $\sqsubseteq$ **Person** (concept inclusion)
  - *RBox: role relationships, e.g., "parentOf is a subrole of ancestorOf"*
    - **parentOf** $\sqsubseteq$ **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 $\mathcal{SROIQ}$
- Concept and role inclusion, concept and role equivalence, Boolean operations, quantification, cardinality restrictions
- Concept expressions **C**, role expressions **R**, and named individuals $\mathbf{N_I}$

$$\mathbf{C} ::= \mathsf{N}_C \mid (\mathbf{C} \sqcap \mathbf{C}) \mid (\mathbf{C} \sqcup \mathbf{C}) \mid \neg\mathbf{C} \mid \top \mid \bot \mid \exists\mathbf{R}.\mathbf{C} \mid \forall\mathbf{R}.\mathbf{C} \mid \geqslant n\,\mathbf{R}.\mathbf{C} \mid \leqslant n\,\mathbf{R}.\mathbf{C} \mid \exists\mathbf{R}.\mathit{Self} \mid \{\mathsf{N}_I\}$$

ABox:	$\mathbf{C}(\mathsf{N}_I)$	$\mathbf{R}(\mathsf{N}_I, \mathsf{N}_I)$	$\mathsf{N}_I \approx \mathsf{N}_I$	$\mathsf{N}_I \not\approx \mathsf{N}_I$
TBox:	$\mathbf{C} \sqsubseteq \mathbf{C}$	$\mathbf{C} \equiv \mathbf{C}$		
RBox:	$\mathbf{R} \sqsubseteq \mathbf{R}$	$\mathbf{R} \equiv \mathbf{R}$	$\mathbf{R} \circ \mathbf{R} \sqsubseteq \mathbf{R}$	$\mathit{Disjoint}(\mathbf{R}, \mathbf{R})$

Cf. Course materials: (Krötzsch et al., 2013)

Rule languages: Horn logic

Horn logic & logic programming

A rule (clause) has the form: A1, . . ., An → B

- **A1, . . ., An** (body) is a conjunction of atomic formulas
- **B** (head) is an atomic formula

There are 2 ways of reading a rule:

- Deductive rules: If **A1,…, An** are known to be true, then **B** is also true
- Reactive (procedural) rules: If the conditions **A1,…, An** are true, then carry out the action **B**

Examples of Rules

- **male(X), parent(P,X), parent(P,Y), notSame(X,Y) → brother(X,Y)**
- **female(X), parent(P,X), parent(P,Y), notSame(X,Y) → sister(X,Y)**
- **brother(X,P), parent(P,Y) → uncle(X,Y)**
- **mother(X,P), parent(P,Y) → grandmother(X,Y)**
- **parent(X,Y) → ancestor(X,Y)**
- **ancestor(X,P), parent(P,Y) → ancestor(X,Y)**

Facts (rules without a body)

- $\rightarrow$ male(John)
- $\rightarrow$ male(Bill)
- $\rightarrow$ female(Mary)
- $\rightarrow$ female(Jane)
- $\rightarrow$ parent(John,Mary)
- $\rightarrow$ parent(John, Bill)

…

Queries / Goals (as rule bodies)

- parent(John, X), female(X) $\rightarrow$
- grandmother(X,Y) $\rightarrow$

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

(<) **Pullonsuojus, 2 kpl:istuva koira** (> Ripustin:henkari, 'Finn Lassie')

Pullonsuojus, 2 kpl:istuva koira

Materiaali: viinapullo: lasi, pulonsuojus: lanka
Valmistaja: Karhulan lasitehdas, Tapio Wirkkala
Valmistusaika: 1962, 1970-l. n.
Valmistustekniikka: viinapullo: tehdasvalmisteinen, pulonsuojus: käsityötä
Käyttäjä: Eero Kallio
Käyttöpaikka: Etelä-Suomen lääni, Suomi
Asiasana: ALKOHOLIJUOMAT, ELÄINHAHMOT, KORISTE-ESINEET
Mitat: pullon pohjan: halkaisija 6,5cm, korkeus 22,5cm, pullonsuojuksen: korkeus 29,0cm
Museokokoelma: LAHDEN HISTORIALLINEN MUSEO
Vastuumuseo: LAHDEN KAUPUNGINMUSEO
Asiasanasto: Lahden kaupunginmuseon sanasto
Esineen numero: LKM:LHM:LHM:ES:95073:154
ID: 95073154

Viinapullo: Alkon Koskenkorvapullo. Lieriömäinen, loivat hartiat. Korkki ja etiketti puuttuvat. Pulonsuojus: istuvan koiran muotoinen pullonsuojus. Muodostuu kahdesta osasta: koiran vartalosta ja päästä. Koiran vartaloon on ommeltu viisi lankatupsua (jalat ja häntä), ylhäällä lankakiristys. Koiran pää on virkattu talouspaperirullasta leikatun lieriön ympärille. Kasvoissa mustat napit silminä, erillinen pieni kuono ja kolme lankatupsua (posket ja päälaella oleva otsatukka).

Esinetyyppi:

Sama käyttäjä

Eero Kallio:

- Keräilykortti, 14 kpl:tuotemainoskortti, erilaisia
- Kulho, 4 kpl:jälkiruokakulho
- Päähine, miehen:turkislakki, 'suikka'
- Taskuliina, miehen:taskuliinan korvike
- Jalkineet, miehen:koripallokengät

Samaan aiheeseen liittyviä esineitä

alkoholijuoma:

- kanisteri:taskumatti
- kanisteri:taskumatti
- kanisteri:taskumatti
- viinipullo:lasipullo
- pullo:lasipullo

eläimet:

- kuvakirja :kuvakirja, kangasta
- helistin :purulelu
- muovikarhu:vinkuva karhulelu
- säästölipas:vanerilipas
- malja:puuvati

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 (part 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

```
hasParent(?x1,?x2) ∧ hasBrother(?x2,?x3) ⇒ hasUncle(?x1,?x3)
```

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#example1"/>
 <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>
```

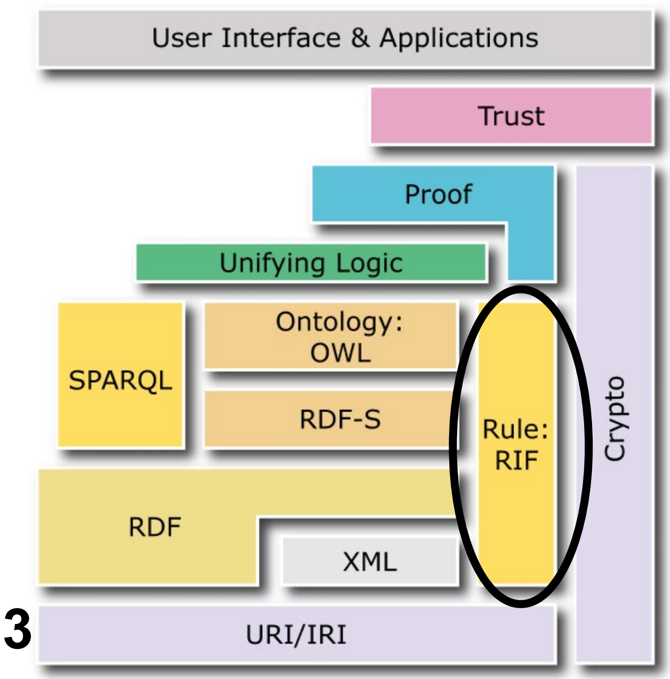
Rule Interchange Format RIF

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: A1, . . ., An → B1,…,Bm
  - *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 Ai and Bj 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 Ai)
- 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

SPIN – SPARQL Syntax

For example, the SPARQL query

```
# 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

```
[ 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)

```
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
                      ])
          ] .
```

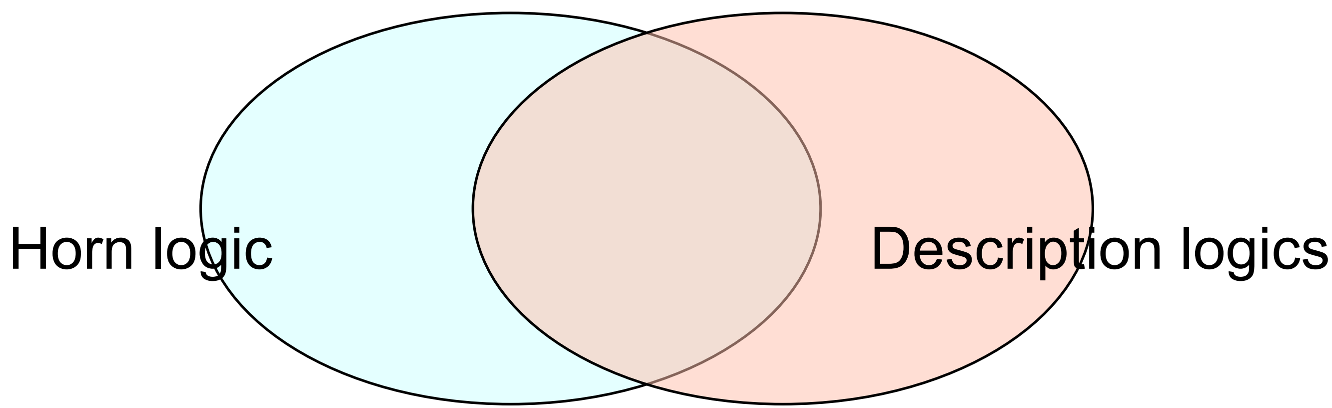
In textual SPARQL syntax, the above query would read as:

```
CONSTRUCT {
    ?this ex:grandParent ?grandParent .
}
WHERE {
    ?parent ex:child ?this .
    ?grandParent ex:child ?parent .
}
```

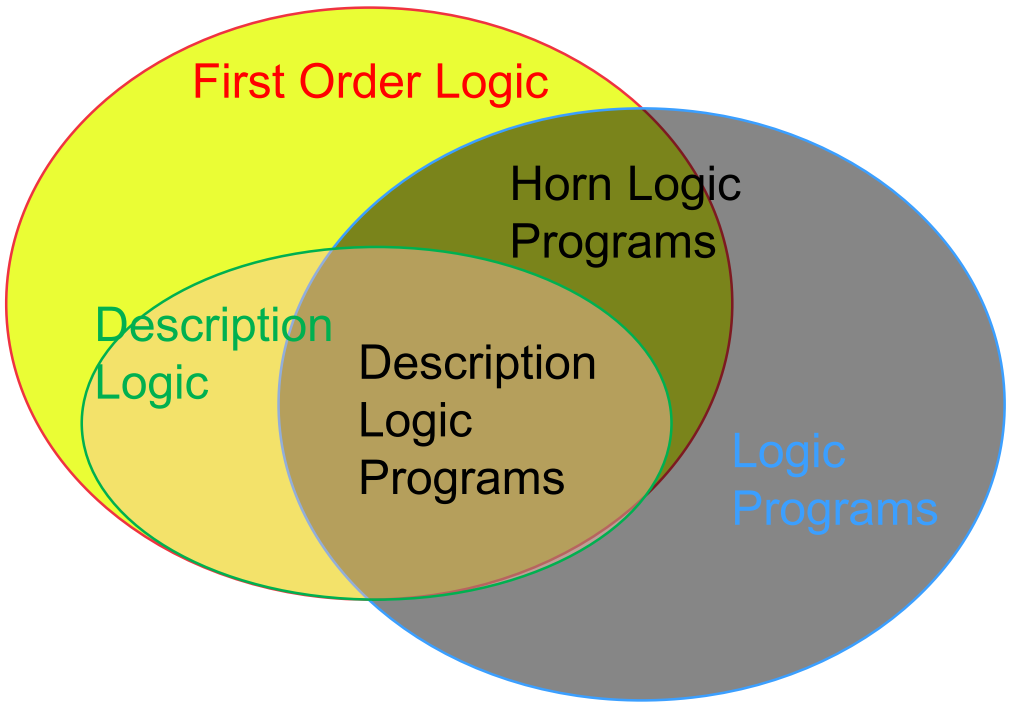
Ontologies vs. logical rules

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*

Two semantic assumptions in logic systems: databases & logic programming vs. pure logic & OWL

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 true it is assumed to be false
  - -> non-monotonic logics!
- CWA made in logic programming & databases but not in logic
- Sometimes makes sense, sometimes not
  - Was there a rain in Tokyo yesterday?
    - *CWA would answer yes/no, if we are informed that there was/was not rain*
    - *CWA would answer no, if we are not informed about rain, which may be not true*

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

Summary: ontology and rule languages

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*