Semantic Web Technologies
The Layer Cake and Beyond

Heiko Paulheim
Previously on Semantic Web Technologies

• What we would like to have:
  \[ \text{daughterOf}(X,Y) \leftarrow \text{childOf}(X,Y) \land \text{Woman}(X) . \]

• Rules are flexible
• There are rules in the Semantic Web, e.g.
  – Semantic Web Rule Language (SWRL)
  – Rule Interchange Format (RIF)
  – Some more

• Some reasoners do (partly) support rules
Semantic Web – Architecture

here be dragons...

Semantic Web Technologies (This lecture)

Technical Foundations

Berners-Lee (2009): Semantic Web and Linked Data
Towards Rules for the Semantic Web

• What we would like to have:
  – daughterOf(X,Y) ← childOf(X,Y) \& Woman(X) .

• OWL only gives an approximation:
SWRL

• Semantic Web Rule Language
  – a rule language for the Semantic Web
  – built to be combined with OWL

• W3C Member Submission (2004)
  – not a standard in a strict sense
  – but widely adopted

• Tool support
  – many reasoners
  – Protégé
SWRL Building Blocks

- Classes are defined as *unary predicates*
  - :Peter a :Person . ↔ Person(Peter)

- Properties are defined as *binary predicates*
  - :Peter :hasMother :Julia . ↔ hasMother(Peter,Mary)
  - :Peter :hatAge 24^^xsd:integer . ↔ hasAge(Peter,24)
SWRL Rules

• Basic form:
  – Head (Consequence) ← Body (Condition)

• Body and head are conjunctions of predicates

• Variables are introduced by ?

• Example:
  – daughterOf(?X,?Y) ← childOf(?X,?Y) ∧ Woman(?X)

• There is no
  – disjunction (logical or)
  – negation
  – unbound variables in the rule head

• ...but there are some ways out
Disjunctions in Rule Body

• There is no disjunction

• Example for disjunction in rule body:
  – Female faculty members are students or staff of the faculty

• Intuitive:
  – `FemaleFacultyMember(?X) ← Woman(?X) ∧ Faculty(?Y) ∧ (worksAt(?X,?Y) ∨ studentAt(?X,?Y))`
Disjunctions in Rule Body

• Solution
  – first step: convert body to disjunctive normal form
    • i.e., disjunction of conjunctions
  – second step: split into individual rules
Disjunctions in Rule Body

• FemaleFacultyMember(?X) ← Woman(?X) ∧ Faculty(?Y) ∧ 
  (worksAt(?X,?Y) ∨ studentAt(?X,?Y))

• turns into
  – FemaleFacultyMember(?X) ←
    (Woman(?X) ∧ Faculty(?Y) ∧ worksAt (?X,?Y)) 
    ∨ (Woman(?X) ∧ Faculty(?Y) ∧ worksAt (?X,?Y))

• ...which turns into
  – FemaleFacultyMember(?X) ←
    Woman(?X) ∧ Faculty(?Y) ∧ worksAt (?X,?Y)
  – FemaleFacultyMember(?X) ←
    Woman(?X) ∧ Faculty(?Y) ∧ studentAt (?X,?Y)
Disjunctions in Rule Head

• Disjunctions in rule head
  – are not so easy to get rid off

• Example
  – Every faculty member is a student or an employee

\[
\text{Student}(?X) \lor \text{Employee}(?X) \leftarrow \text{FacultyMember}(?X)
\]

• On the other hand: what should a reasoner conclude?
  → disjunction in rule head does not make as much sense!
Disjunctions in Rule Head

• SWRL is meant to be used together with OWL
• Idea: build an artificial class for the rule head
  
  StudentOrEmployee owl:unionOf (Student Employee)
  
  StudentOrEmployee(?X) ← FacultyMember(?X)

• This way, we can conclude that ?X is in the union of both classes
  – Further reasoning on other axioms might rule out one option
Negation

• Negation can be simulated with a similar trick

• Example:
  – Creatures living in the water are not human.

• Intuitive:
  – \( \neg \text{Human}(?X) \leftarrow \text{Creature}(?X) \land \text{habitat}(?X,\text{Water}) \)
Simulating Negation

• Again: combining SWRL and OWL
  – NonHuman owl:complementOf Human .

• New Rule:
  – NonHuman(?X) ← Creature(?X) ∧ habitat(?X,Water)

• Now, a reasoner can find a contradiction between
  – :Nemo a :Creature; habitat :Water .
• and
Simulating Negation

- Negation in the rule body:
  \[ \text{FlightlessBird}(?X) \leftarrow \text{Bird}(?X) \land \neg \text{habitat}(?X,\text{Air}) \]

- Define class:
  \[
  \text{notAirHabitat} \equiv \text{owl:Restriction} \cap \text{owl:onProperty} :\text{habitat} \cap \text{owl:allValuesFrom} \left( \text{notAirHabitat} \right) \]

  \[ \text{FlightlessBird}(?X) \leftarrow \text{Bird}(?X) \land \text{notAirHabitat}(?X) \]
Unbound Variables

- All variables appearing in the rule head must also appear in the body
  - those are *bound* variables

- Example: every human has a (human) father
  - `Human(?Y) ∧ hasFather(?X,?Y) ← Human(?X)`

- In that case, the reasoner would have to create *new* instances for Y
  - Possible issue: termination
  - No easy solution in SWRL+OWL
SWRL Extensions and Built-Ins

• Comparison
  - olderThan(?X,?Y) ← hasBirthdate(?X,?BX) ∧ hasBirthdate(?Y,?BY) ∧ swrlb:lessThan(\(?BX,?BY\))

• Arithmetics
  - twiceAsOld(?X,?Y) ← hasAge(?X,?AX) ∧ hasAge(?Y,?AY) ∧ swrlb:multiply(\(\text{?AX,?AY,2}\))

• String operations
  - PeopleWithS(?X) ← hasName(?X,?N) ∧ swrlb:startsWith(?N,"S")
SWRL Extensions and Built-Ins

- Some reasoners also allow for custom built-ins
- E.g., for wiring a reasoner to external systems

```
SELECT id FROM customer ... 
for each(id in results) 
add "http://.../"+id to bindings
```

[Diagram showing relationships between Customer Database, AdapterImpl.java, Reasoner, and the configuration: my:adapter(?x) → Customer(?x)]
SWRL Extensions and Built-Ins

• More use cases for custom built-ins
• Live data
  – Weather
  – Stock exchange
  – Product availability

• Complex computations
  – Trip duration from A to B (e.g., Google Maps API)
  – Simulations and predictions
  – ...
Monotonic Reasoning with SWRL

• Recap: monotonous vs. non-monotonous reasoning
  – monotonous: every consequence derived is true forever
  – non-monotonous: consequences may be revoked

• SWRL ist monotonous
  – i.e., consequences of all rules add up
  – allows for efficient reasoning
  – may lead to contradictions
Safety of Rules

- Termination guarantee of reasoning

- So far
  - no new instances, classes, and properties are generated

- This constrains the set of consequences which can be derived:
  - C*I type assertions
  - I*O*I object property assertions
  - I*D*L datatype property assertions

→ in monotonic reasoning, the reasoner eventually terminates
Safety of Rules

• Consider this example:

```prolog
:Person rdfs:subClassOf [  
a owl:Restriction ;  
owl:onProperty :hasFather ;  
owl:cardinality 1^^xsd:integer ] .
:hasFather rdfs:range :Person .
:Grandchild rdfs:subClassOf :Person .

hasFather(?x,?y) ∧ hasFather(?y,?z) → Grandchild(?x)
```

• Given

```prolog
:Peter a :Person .
```

• Do we derive GrandChild(Peter)?
Safety of Rules

• Possible solution:
  – We know that each person has a father
  – therefore:
    
    \[ \text{Peter} \; \text{:hasFather} \; \text{_:p0} \; \; \text{_:p0} \; \text{:hasFather} \; \text{_:p1} \; \; \text{_:p1} \; \ldots \]
  – and thus
    
    \[ \text{Peter a :Grandchild}. \]

• What is the price of that solution?
  – We allow for the creation of new instances
  – i.e., we sacrifice guaranteed termination
Safety of Rules

• DL safe rules:
  – Variables are only bound to *existing* instances
  – No new instances are created

• Thus, *we cannot* derive
  \[ :\text{Peter} \ a \ :\text{Grandchild} . \]

• Once more: trading off
  – expressivity
  – decidability
Production Rules

- Sometimes, monotonous rules are not desirable
  - consider: if a student passes SWT, his/her credit increases by 6 ECTS

- A first attempt with SWRL + built-ins:
  \[
  \text{Student}(\text{?X}) \land \text{hasPassed}(\text{?X}, :\text{SWT}) \\
  \land \text{hasCredits}(\text{?X},\text{?C}) \land \text{swrlb:add}(\text{?NC},\text{?C},6) \\
  \rightarrow \text{hasCredits}(\text{?X},\text{?NC}) 
  \]
Production Rules

• Consider:

```prolog
:Peter a :Student .
:Peter :hasCredits 26^^xsd:integer .
:Peter :hasPassed :SWT .
```

• After applying the rule:

```prolog
:Peter :hasCredits 32^^xsd:integer .
```

• But rules are monotonous, so the following holds as well:

```prolog
:Peter :hasCredits 26^^xsd:integer .
```

• ...and the reasoner is done yet
Production Rules

• What happens:

:Peter :hasCredits 26^^xsd:integer .
:Peter :hasCredits 32^^xsd:integer .
:Peter :hasCredits 38^^xsd:integer .
:Peter :hasCredits 44^^xsd:integer .
:Peter :hasCredits 50^^xsd:integer .
...

• We need to
  – revoke/overwrite statements
    • in contrast to monotonous reasoning!
  – define new criteria for termination
Rule Interchange Format

- Rule Interchange Format (RIF)
- Unification of
  - Basic Logic Rules (such as SWRL)
  - Production Rules (e.g., JENA rules)
- Standardized by W3C in 2010
Berners-Lee (2009): *Semantic Web and Linked Data*
Other Semantic Web Languages

• What else is out there?

Cardoso (2006): The Semantic Web Vision – Where are We?
Other Semantic Web Languages

• There is a wild mix
  – of old and new languages
  – of different paradigms
  – of sophisticated languages and pure, low-level logic

• We will look at one example of a radically different language
F-Logic

- Main concept: *frames*
  - collection of properties of a class
  - similar to class and database models

<table>
<thead>
<tr>
<th>Person</th>
<th>Mutter (Person)</th>
<th>Vater (Person)</th>
<th>Alter (int)</th>
</tr>
</thead>
<tbody>
<tr>
<td>:Paul</td>
<td>:Martha</td>
<td>:Hans</td>
<td>24</td>
</tr>
<tr>
<td>:Martha</td>
<td>:Johanna</td>
<td>:Karl</td>
<td>47</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
F-Logic: A First Glance

• First observation:
  – relations are bound to class
  – in RDFS/OWL: first class citizens

• Inheritance
  – Relations are inherited to subclasses
  – Domain and range cannot be restricted any further

• Semantics
  – Closed world semantics
  – Negation can be used
F-Logic: Rules

• Almost everything is expressed in rules

• e.g., property chains:

  \[
  \text{uncleOf}(?X, ?Z) :-
  \]

  \[
  \begin{align*}
  & ?X: \text{Man}[\text{parentOf}\rightarrow ?Y] \\
  \end{align*}
  \]

• Datalog-like syntax
• :- is used for implication ←
• Variables are denoted with ?
F-Logic: Quantifiers

• There are extensional and universal quantifiers

• Authors are persons who have written at least one book
  \( ?X:Author \leftarrow ?X:Person \)
  \( \quad \text{AND} \quad \text{(EXIST } ?Y \quad ?Y:Book \quad \text{and} \quad ?X[\text{hasWritten}\rightarrow?Y]) . \)

• A non-author is a person who has not written any book
  \( ?X:NonAuthor \leftarrow ?X:Person \)
  \( \quad \text{AND} \quad \text{NOT(EXIST } ?Y \quad ?Y:Book \quad \text{and} \quad ?X[\text{hasWritten} \rightarrow ?Y]) . \)

• A star author is an author who as only written bestsellers
  \( ?X[\text{isStarAuthor}\rightarrow\text{true}] \leftarrow ?X:Author \quad \text{AND} \)
  \( \quad \text{(FORALL } ?Y \)
  \( \quad \quad \quad \text{(} ?X[\text{hasWritten}\rightarrow?Y] \quad \rightarrow \quad ?Y:Bestseller) \quad ) \quad . \)
F-Logic: Negation

• Negation may have unwanted consequences
• Consider this example:
• \(?X[hates->?Y] \) :-
  


• Assume, the reasoner wants to prove \(?X[likes->Stefan] \) .

• Possible plan:
  
  \(?X[likes->Stefan] \).

  \(?X[knows->Stefan]$ and \(\text{not}(\text{$?X[hates->Stefan]$})) \).

  \(?X[knows->Stefan]$ and \(\text{not}(\text{not}(\text{$?X[likes->Stefan]$ or $?X[doesntCare->Stefan]$})) \).

  \(?X[knows->Stefan]$ and \(\text{not}(\text{not}(\text{$?X[knows->Stefan]$ and ...} \) .
F-Logic: Decidability and Stratification

• F-Logic ontologies with negations can be undecidable
• Underlying problem:
  – Cycles of rules containing negations
• Simplest case
  – \( p(X) :- \neg p(X) \).

• Test: Stratification
  – lat. *Stratum* (pl.: *Strata*): *Layer*
• Divide ontology into layers
• Each predicate is assigned to a layer
  – Classes are treated as unary predicates
Assign a layer $S(p)$ to each predicate $p$

Two conditions must be fulfilled:

- for all rules which have $p$ in their head and a non-negated predicate $q$ in the body:
  \[ S(q) \leq S(p) \]
- for all rules which have $p$ in their head and a negated predicate $q$ in their body
  \[ S(q) < S(p) \]

If such an assignment can be found, the ontology is decidable
F-Logic: Decidability and Stratification

• Simple case:

• We have to ensure
  – S(likes) < S(hates)
  – S(likes) ≤ S(knows)

• For those two rules, we can assign
  – S(likes) = 0
  – S(hates) = 1
  – S(knows) = 0
F-Logic: Decidability and Stratification

- We obtain the following layers

<table>
<thead>
<tr>
<th>Layer 1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Layer 0</th>
</tr>
</thead>
</table>

- Trivial observation
  - For ontologies without negation, one layer is enough!
F-Logic: Decidability and Stratification

• Back to the original example
  
  {?X[hates->?Y] :-

• Can we find a stratification?
• We would need
  – S(likes) < S(hates)
  – S(hates) < S(likes)

• This is not possible!
  → The ontology cannot be stratified, i.e., it is undecidable!
Recap: Russell's Paradox

• A classic paradox by Bertrand Russell, 1918

• In a city, there is exactly one barber who shaves everybody who does not shave themselves.

Who shaves the barber?
F-Logic: Decidability and Stratification

• Russell’s paradox in F-Logic:
  \[\text{theBarber[shaves->?X]} \iff \neg (?X[shaves->?X])\].

• We would need
  \[S(\text{shaves}) < S(\text{shaves})\]

![Error message](image)
Validating Datasets with RDF Shapes

• Ontology reasoning is good for semantic validation
  – but sometimes problematic due to semantic properties
  – i.e., closed world assumption, non unique name assumption

• To validate data quality
  – we want to ensure certain data is there
    • e.g., every person has a name
  – we want to ensure that data is not duplicated
    • e.g., every person has exactly one birth place
  – etc.
Validating Datasets with RDF Shapes

• Example dataset:
  :Mary a :Person .
  :Mary :birthPlace :Mannheim .
  :Mary :birthPlace :Berlin .

• Constraints in OWL:
  Person rdfs:subClassOf [ a owl:Restriction .
    owl:onProperty :name .
    owl:minCardinality 1 . ]
  Person rdfs:subClassOf [ a owl:Restriction .
    owl:onProperty :birthPlace .
    owl:maxCardinality 1 . ]
Shapes Constraint Language (SHACL)

- A W3C Standard since 2017
- For RDF *validation*
- Differences to reasoning
  - Closed world evaluation
  - Counting is possible
  - More fine-grained checks (see later)
Shapes Constraint Language (SHACL)

• Example dataset:
  :Mary a :Person .
  :Mary :birthPlace :Mannheim .
  :Mary :birthPlace :Berlin .

• Constraints in SHACL:
  :PersonShape
    a sh:NodeShape ;
    sh:targetClass :Person ;
    sh:property [
      sh:path :name ;
      sh:minCount 1 ;
      sh:datatype xsd:string ] ;
    sh:property [
      sh:path :birthPlace ;
      sh:maxCount 1 ;
      sh:class :City ] .
Shapes Constraint Languages

• Further possibilities
  – Dependencies between attributes
    • e.g., given name and first name are equivalent
  – Complex expressions involving paths and even SPARQL queries
  – Checking strings against regex patterns (e.g., phone numbers)
  – ...

...
Finale

- One of the current hot topics in Semantic Web research: Embeddings
RDF Embeddings

• Challenge in RDF/OWL etc.:
  – How similar are two entities?
  – e.g., is Mannheim more similar to Karlsruhe than to Heidelberg?

• Application scenarios:
  – Recommender systems
  – Information retrieval
Excursion: word2vec

• Such approaches exist for words
  – aka, *word embeddings*
  – each word becomes a vector in a low-dimensional vector space
  – similar words are close in that vector space
  – semantic relations have a similar direction and length

• allows for arithmetics, e.g., King – Man + Woman = Queen

(Mikolov et al., NAACL HLT, 2013)
word2vec

• General idea: similar words appear in similar contexts
• Training set: sequences from a text corpus
• Training method: neural network
• Training variants:
  – Continuous bag of words (CBOW): predict a word from its context
  – Skip-Gram: predict context from a word
From word2vec to RDF2vec

• Generating sequences from an RDF dataset
  – by starting random walks from each entity

• Example:

• Those are fed into a word2vec training engine

• Variants (Cochez et al., 2017)
  – replace “random” by “semi-random” walk
  – e.g., weight edges by frequency, PageRank, ...
From word2vec to RDF2vec

- Observation: similar properties hold for RDF2vec

Ristoski & Paulheim: RDF2vec: RDF Graph Embeddings for Data Mining, 2016

a) DBpedia vectors
b) Wikidata vectors
TransE and Descendants

• In RDF2vec, relation preservation is a by-product
• TransE: direct modeling
  – Formulates RDF embedding as an optimization problem
  – Find mapping of entities and relations to $\mathbb{R}^n$ so that
    • across all triples $<s,p,o>$
      $\Sigma ||s+p-o||$ is minimized
Limitations of TransE

- Symmetric properties
  - we have to minimize
    \[ \|Barack + \text{spouse} - \text{Michelle}\| \text{ and } \|\text{Michelle} + \text{spouse} - \text{Barack}\| \]
    simultaneously
  - ideally, \(Barack + \text{spouse} = \text{Michelle}\) and \(\text{Michelle} + \text{spouse} = \text{Barack}\)
    - Michelle and Barack become infinitely close
    - spouse becomes 0 vector
Limitations of TransE

• Transitive Properties
  – we have to minimize
    $||Miami + partOf – Florida||$ and $||Florida + partOf – USA||$, but also
    $||Miami + partOf – USA||$
  – ideally, $Miami + partOf = Florida$, $Florida + partOf = USA$, $Miami + partOf = USA$
    • Again: all three become infinitely close
    • $partOf$ becomes 0 vector
Limitations of TransE

- Numerous variants of TransE have been proposed to overcome limitations (e.g., TransH, TransR, TransD, …)
- Plus: embedding approaches based on tensor factorization etc.
A Look Back

• This is where we embarked on our Semantic Web Technologies journey in September:

In the eyes of a human

Dr. Mark Smith
Physician
Main St. 14
Smalltown
Mon-Fri 9-11 am
Wed 3-6 pm

in the eyes of a computer

<html>
  ...
  <b>Dr. Mark Smith</b>
  <i>Physician</i>
  Main St. 14
  Smalltown
  Mon-Fri 9-11 am
  Wed 3-6 pm
  ...
</html>

Print in bold: „hmf298hmmhuds“
Print in italics: „mj2i9ji0“
Print normal: „fdsah
02hfadsh0um2m0adsmf0ihtm
asdfjköfdsa298ndsfmij32mio
lk2mjpoimjiofdpmsajiomjm“
A Look Back

• **Formal Semantics**
  – Every entity has classes and relations to other entities
  – Those are defined in an ontology
  – Humans and computers can interpret those semantics
  – Computers give justification on reasoning results

• **Embeddings**
  – Every entity is an n-dimensional vector
  – We do not know about the meaning of the dimensions
  – Results are often good, but hard to justify
The 2009 Semantic Web Layer Cake
The 2018 Semantic Web Layer Cake

User Interface and Applications

Embeddings

Data Interchange: RDF

Data Interchange: XML

URI

Unicode
Towards Semantic Vector Space Embeddings

[Diagram showing a 2D plot with vectors for various entities such as movies and superheroes, illustrating the semantic space.]
The Holy Grail

• Combine semantics and embeddings
  – e.g., directly create meaningful dimensions
  – e.g., learn interpretation of dimensions a posteriori
  – ...

![Image of the Holy Grail](image-url)
Summary

• OWL and OWL 2 are not the end
  – Rules create more possibilities

• Other (non W3C standard) languages have also been also proposed
  – different semantic paradigms (e.g., F-Logic)
  – different problem setting (e.g., SHACL)

• Recent trend
  – using vector space embeddings
  – challenge: combine interpretable semantics and embeddings
Recommendations for Upcoming Semesters

• Information Retrieval and Web Search (next FSS), Prof. Glavaš

• Web Data Integration (HWS), Prof. Bizer
• Relational Learning (HWS), Prof. Stuckenschmidt
• Text Analytics (HWS), Prof. Glavaš

• Web Mining (FSS 2020), Prof. Ponzetto
Questions?