Knowledge Representation and the Semantic Web – an Ontologician's View

Sebastian Rudolph
ESWC 2018

Crete, 06.06.2018
(1) G(r)eeek Joke ✓
(2) Anecdote
Purpose of the Talk

(P1) Remind the greater ESWC audience of the virtues and usefulness of logic for the Semantic Web (using examples).

(P2) Critically consider the necessary evolution logic-based knowledge representation had and has to undergo to be and remain relevant to SW community.
(P1) Remind the greater ESWC audience of the virtues and usefulness of logic for the Semantic Web (using examples).

(P2) Critically consider the necessary evolution logic-based knowledge representation had and has to undergo to be and remain relevant to SW community.

Hidden agenda:

(H1) Shameless self-advertisement.

**Warning**: This talk is self-centered, opinionated, and subjective.

(H2) Show how cool logicians’ work is.

**Warning**: This talk contains proofs.
(1) Logicians do not believe anything unless it is formally proven.

(2) Logicians love puzzles.

(3) Logicians like to use available tools in unforeseen ways (McGuyverism).

(4) Logicians are lazy. Preferred mode of working: reduce a problem to one already solved.
Gottfried Wilhelm Leibniz (1646-1716)
formal definition of syntax and semantics of logics and deduction calculi establish foundations for automated inferencing
rise of computers and research on symbolic AI and deductive databases pave way toward automated reasoning
Evolution: Mathematical Logic \rightarrow Computational Logic
more and more knowledge is available in interlinked, structured, logically accessible form, providing ground truth about the world.
Evolution: Classical KR → Contemporary KR
Yesterday:

<table>
<thead>
<tr>
<th></th>
<th>classical KR</th>
<th>classical Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td>type of data considered</td>
<td>mainly terminological</td>
<td>mainly assertional</td>
</tr>
<tr>
<td>source of expressivity</td>
<td>ontology language</td>
<td>query language</td>
</tr>
<tr>
<td>tasks considered</td>
<td>satisfiability, axiom entailment</td>
<td>query answering, query containment</td>
</tr>
<tr>
<td>model theory</td>
<td>arbitrary models</td>
<td>finite models</td>
</tr>
</tbody>
</table>
Today:

<table>
<thead>
<tr>
<th></th>
<th>contemporary KR</th>
</tr>
</thead>
<tbody>
<tr>
<td>type of data considered</td>
<td>assertional and terminological</td>
</tr>
<tr>
<td>source of expressivity</td>
<td>ontology language and query language</td>
</tr>
<tr>
<td>tasks considered</td>
<td>query answering and containment over ontologies</td>
</tr>
<tr>
<td>model theory</td>
<td>arbitrary or finite models</td>
</tr>
</tbody>
</table>
The bottlenecks of knowledge representation...
Research Workflow for Logic-based KR

1. Identification of Desirable Novel Logical Features
2. Investigation of Computational Properties
3. Design of Optimized, Practically Efficient Algorithms
4. Development of Intuitive Methodologies for Modeling Supporting
Identification of Desirable Novel Logical Features

Need to interact with “normal people” (i.e., non-logicians)

Development of Intuitive Methodologies for Modeling Supporting
Identification of Desirable Novel Logical Features

Investigation of Computational Properties

Design of Optimized, Practically Efficient Algorithms

Development of Intuitive Methodologies for Modeling Supporting Research Workflow
Identification of Novel Logical Features

**Logical Features could be:**
- Modeling Features (e.g., logical operators)
- Semantic Assumptions
- Query Languages
- Reasoning Tasks

**Typical Questions:**
- Do the features enhance expressivity?
- Can they be simulated with already available features?
- If the latter: at what cost *(succinctness)*?

**How do (non)expressivity proofs look like:**
- If new feature is expressible: easy, just give a construction
- If new feature is not expressible: more tricky (model-theoretic; using properties of underlying logic; ...)
Qualified subproperties encode that one property implies another, provided additional “typing constraints” are satisfied:

„Everybody who is of age and has signed some contract is bound to that contract.“

OfAge hasSigned Contract
Qualified subproperties encode that one property implies another, provided additional “typing constraints” are satisfied:

„Everybody who is of age and has signed some contract is bound to that contract.“
Qualified subproperties encode that one property implies another, provided additional “typing constraints” are satisfied:

„Everybody who is of age and has signed some contract is bound to that contract."

Expressible in First-Order Logic:
\[ \forall x,y. \text{hasSigned}(x,y) \land \text{OfAge}(x) \land \text{Contract}(y) \rightarrow \text{boundTo}(x,y) \]

RDFS? OWL?
Example: Modeling Feature

Yes we can (in OWL)!

„property loops“: OfAge ⊑ ∃ofAge.Self

Contract ⊑ ∃contract.Self

property chain axiom: ofAge ≪ hasSigned ≪ contract ⊑ boundTo
Example: Modeling Feature

Yes we can (in OWL)!

„property loops“: OfAge ⊑ ∃ofAge.Self Contract ⊑ ∃contract.Self

property chain axiom: ofAge ◦ hasSigned ◦ contract ⊑ boundTo
Identification of Desirable Novel Logical Features

Investigation of Computational Properties

Design of Optimized, Practically Efficient Algorithms

Development of Intuitive Methodologies for Modeling Supporting
Identification of Desirable Novel Logical Features

Investigation of Computational Properties

Design of Optimized, Practically Efficient Algorithms

Development of Intuitive Methodologies for Modeling Supporting Research Workflow for Logic-based KR
Investigation of Computational Properties

**Computational Properties:**
- Decidability
- Computational Complexity
- can be fine-tuned: data/query/combined complexity ...
  (important to have more adequate measures for scalability)

**Questions:**
- Can there be a generic algorithm for a certain type of problem?
- How much (time/space) resources will it require (in the worst case) to solve the problem given the size (or some other parameter) of the input?

**How do proofs look like:**
- Reduction from/to known problems
- Decidability/complexity membership: give algorithm
in databases: just one model (the database itself); this is rather easy

in logics: one knowledge base (ontology), many models; not so easy
Problem: Given an OWL knowledge base $\mathcal{K}$ and a simple conjunctive query $q$, is there a match for $q$ in every model of $\mathcal{K}$?

Decision procedure for $\mathcal{K}^2 q$:

- terminates with output YES if $\mathcal{K}^2 q$ and
- terminates with output NO if $\mathcal{K} \not\models q$. 

YES/NO
• no problem: logical consequences in first-order predicate logic (FOL) are (recursively) enumerable
• knowledge bases and conjunctive queries can be expressed in FOL
• input $\mathcal{K}$
• wait for output of $q$
There has to be a model $I$ of $\mathcal{K}$, not containing a match for $q$ (aka “countermodel”).

Problem: there are too many (uncountably many) models.

Solution: identify a representative, countable class of “nice” models.

Candidates: regular, forest-like models.

Question: is this class representative?

to prove this, one shows how an arbitrary countermodel can be transformed into a “nice” one.
subtle model transformation that preserves modelhood and query non-entailment

unraveling -> cautious replacement of critical parts -> collapse
• there are “only” countably many “nice” (that is: regular forest) interpretations, so we can enumerate them one by one
• wait for output of an $I$ with $I^2 K$ and $I^2/q$
...two complementary semi-decision procedures which – if run in parallel – constitute a decision procedure

terminates, if $K^2 q$

terminates, if $K \not\models q$
Many Open Problems...

- Simple Conjunctive Queries for OWL – decidable (complexity unknown)
- Arbitrary CQs? – unknown
- Arbitrary CQs under finite model semantics? – undecidable
A silver bullet for undecidability problems...

Definition 3 (Post Correspondence Problem). Let $\mathbb{P} = \{(g_1, g'_1), \ldots, (g_\mu, g'_\mu)\}$ be an arbitrary finite set of pairs of non-empty strings over the alphabet $\{a, b\}$. A nonempty finite sequence $i_1, \ldots, i_n$ of natural numbers from $\{1, \ldots, \mu\}$ is called a solution sequence of $\mathbb{P}$ if $g_{i_1} \cdots g_{i_n} = g'_{i_1} \cdots g'_{i_n}$. The Post Correspondence Problem (short: PCP) requires to determine if there exists a solution sequence for a given $\mathbb{P}$. 
1  
\[ \text{b} \quad \text{b} \quad \text{b} \quad \text{b} \]  

2  
\[ \text{a} \quad \text{b} \]  

3  
\[ \text{b} \quad \text{b} \quad \text{b} \quad \text{a} \]
1

b
b b b

2

a b
a

3

b b b a
a

2

a b b
2

a

1

b b b
1

b

3

b b b a
3

a
Post Correspondence Problem

\[
\begin{array}{cccccc}
1 & a & b & b & a & b \\
2 & b & b & b & b & b \\
3 & a & a & a & a & a \\
4 & a & b & b & b & b \\
\end{array}
\]
Post Correspondence Problem

a b b a
b b b b a

1 2 3 4

1 b b a
b b b a
1 2 3 4
Post Correspondence Problem

1  2  3  4

| a | b | b | a |
| b | b | a | b |
| b | b | b | b |
| b | a |   |   |

1  2  3  4

| b | b | a | b |
| b | b | b | b |
| b | a |   |   |
• (un)decidability results fragmented and often ad hoc
• few generic principles, no coherent general „theory of (un)decidability“
• many relevant problems open (e.g. full query answering in OWL)
• design of KR formalisms: maximally expressive, “dirty fixes” to avoid undecidability
A Grand Unified Theory of Decidability in Logic-Based Knowledge Representation
Identification of Desirable Novel Logical Features

Investigation of Computational Properties

Design of Optimized, Practically Efficient Algorithms

Development of Intuitive Methodologies for Modeling Supporting
Identification of Desirable Novel Logical Features

Investigation of Computational Properties

Design of Optimized, Practically Efficient Algorithms

Development of Intuitive Methodologies for Modeling Supporting

Research Workflow for Logic-based KR
Options for Algorithm Development:

- DIY from scratch (the hard way)
- Reduction to another problem with tool support

Typical Questions:

- Is the algorithm worst-case optimal?
- Can one (re)use existing, optimized technology?
Example: OWL QL Query Answering

- OWL QL based on DL-Lite family
- typically used for data-intensive scenarios
- also covers RDFS in standard use
Example: OWL QL Query Answering

- Ontology-mediating Query
  - Ontology (OWL QL)
  - Query (Q)

- Data (RDF)

- Standard approach: query rewriting

Example: OWL QL Query

Answering
Knowledge Base = RDF Graph

- but: data and schema knowledge all mixed together
- schema knowledge expressed as data using special vocabulary: `rdfs:subClassOf`, ...
- assumption: RDF Graph not known in advance
Query: Give me all the individuals known to be persons.

first attempt:

```
SELECT ?p
WHERE ?p rdf:type ex:Person
```

does not work for:

```
ex:shakespeare rdf:type ex:Author .
ex:Author rdfs:subClassOf ex:Person .
```
Query: Give me all the individuals known to be persons.

second attempt:

```sparql
SELECT ?p
WHERE { ?p rdf:type ex:Person} UNION
    { ?p rdf:type ?c . ?c rdfs:subClassOf ex:Person}
```

does not work for:

```sparql
ex:shakespeare rdf:type ex:Author .
ex:Author rdfs:subClassOf ex:Artist .
ex:Artist rdfs:subClassOf ex:Person .
```
Query: Give me all the individuals known to be persons.

We need to incorporate arbitrarily long subclass paths.

Use SPARQL 1.1!

```
SELECT ?p
WHERE { ?p rdf:type ?c . ?c rdfs:subClassOf* ex:Person}
```

Is this it?
Query: Give me all the individuals known to be persons.

We need to incorporate arbitrarily long subclass paths. **Use SPARQL 1.1!**

```sparql
SELECT ?p
WHERE {?p rdf:type ?c . ?c rdfs:subClassOf* ex:Person}
```

Is this it? No!
Example: OWL QL Query Answering

rdf:authorOf    rdfs:subPropertyOf  ex:creatorOf .
ex:creatorOf    rdfs:domain        ex:Artist .
ex:Artist       rdfs:subClassOf     ex:Person .
Example: OWL QL Query Answering

Important Observations:

- proof tree is linear
- leaf triples in the proof tree form sort of a chain

→ this holds in general
**Theorem:**
For OWL QL ontologies, each of the following reasoning tasks can be expressed in a single SPARQL 1.1 query:

- Is the ontology consistent?
- Is the class A consistent?
- Does the ontology entail A rdfs:subClassOf B?
- Does the ontology entail R rdfs:subPropertyOf S?
- Does the ontology entail c rdf:type A?
- Does the ontology entail c R d?

Directly extends to schema queries: A,B,R,S can be variables!
SPARQL 1.1 query pattern retrieving all inconsistent classes of an OWL QL ontology in RDF format:

\[
x \left( (sCO \mid eqC \mid ^{eqC} \mid intListMember \mid owl:someValuesFrom \mid \right.
\left. \left( owl:onProperty / \left( INV \mid SpoEqP \right)^* / \left( ^{owl:onProperty} \mid rdfs:domain \mid rdfs:range \right) \right)^* \ ?C . \right)
\left\{ \ ?C \ subClassOf \ owl:Nothing \right\} \ \mathrm{UNION}
\left\{ \ ?C \ subClassOf \ ?D1 \ \left\{ \ ?C \ subClassOf \ ?D2 \right\} \ \mathrm{UNION} \ \mathrm{UNIVCLASS}[?D2] \right\}
\left\{ \ ?D1 \ disjointClasses \ ?D2 \right\} \ \mathrm{UNION}
\left\{ \ ?V \ \mathrm{rdf:type} \ owl:AllDisjointClasses \ . \ twomembers[?V, ?D1, ?D2] \right\}
\right) \ \mathrm{UNION}
\left\{ \ ?C \ (owl:onProperty / \left( INV \mid SpoEqP \right)^* ) \ ?P . \right\}
\left\{ \ ?P \ subPropertyOf \ owl:bottomObjectProperty \right\} \ \mathrm{UNION}
\left\{ \ ?P \ subPropertyOf \ ?Q1 \ \left\{ \ ?P \ subPropertyOf \ ?Q2 \right\} \ \mathrm{UNION} \ \mathrm{UNIVPROPERTY}[?Q2] \right\}
\left\{ \ ?Q1 \ (owl:propertyDisjointWith \mid ^{owl:propertyDisjointWith} ) \ ?Q2 \right\} \ \mathrm{UNION}
\left\{ \ ?V \ \mathrm{rdf:type} \ owl:AllDisjointProperties \ . \ twomembers[?V, ?Q1, ?Q2] \right\}
\right)
\]
**Theorem:** For OWL QL ontologies and for all conjunctive queries, there is a schema-agnostic SPARQL 1.1 rewriting of linear size.

→ More complicated due to non-distinguished variables
→ Use more SPARQL features: VALUES and FILTER
  (some guessing involved)
Every RDF database featuring SPARQL 1.1 queries can be used as an OWL QL reasoner, with full support for conjunctive queries including schema variables.
Every RDF database featuring SPARQL 1.1 queries can be used as an OWL QL reasoner, with full support for conjunctive queries including schema variables.
Identification of Desirable Novel Logical Features

Investigation of Computational Properties

Design of Optimized, Practically Efficient Algorithms

Development of Intuitive Methodologies for Modeling Supporting
Identification of Desirable Novel Logical Features

Investigation of Computational Properties

Design of Optimized, Practically Efficient Algorithms

Development of Intuitive Methodologies for Modeling Supporting
"Normal people" don’t think logically!

Experiments in cognitive psychology: most human only capable of very elementary inferences.

- modus ponens (from "A → B" and "A" follows "B"): 89-100%
- modus tollens (from "A → B" and "not B" follows "not A"): 25-57%

Explicit logical modelling even more demanding than making logical deductions.
Even harder: make sure that specified knowledge is complete (that is, every relevant question can be answered from it).

**knowledge base:**

There are birds.
All birds lay eggs.
Birds are not mammals.
Egg-laying individuals are not livebearing.
There are livebearing mammals.

Are all mammals livebearing?
Even harder: make sure that specified knowledge is complete (that is, every relevant question can be answered from it).

**knowledge base:**

There are birds.
All birds lay eggs.
Birds are not mammals.
Egg-laying individuals are not livebearing.
There are livebearing mammals.

Are all mammals livebearing?

NO! There are egg-laying mammals!
purpose: approve logical statements by presenting to a domain expert based on formal concept analysis core, integrated with reasoner
Modeling Support: Possible World Explorer

```
Pizza ⊨
¬ ∃ hasTopping . T

class expression C(X)

known instances

DomainConcept
   ⊢ Country
      ⊢ hasIngredient . T
      ⊢ hasBase . T

necessary adjuncts

possible adjuncts

known instances
```

Explain the possible world explorer and its features for modeling support.
Absurd situations found in the famous pizza ontology:

- **Class exploration (disjointness axioms)**
  - a country that is a food
  - a country that has a country of origin
  - a food that is the country of origin of something
  - a pizza that is an ingredient of something
  - a pizza that has no topping

- **Role exploration (domain and range axioms)**
  - a country of origin that is not a country
  - something that has a spiciness but that is not some food

- **Further exploration**
  - a vegetarian pizza that has an ingredient that is meat or fish
  - explanation: a vegetarian pizza is defined as not having a meat or fish topping, but this does not exclude that it has a meat or fish ingredient, e.g., in the pizza base (hasIngredient is a super-property of hasTopping)
Further Challenges for KR@SW
Further Challenges I: Noise

*ex falso (sequitur) quodlibet:*

A \[ \rightarrow \neg A \]

B

• from a contradiction, anything can be inferred
• thus any body of knowledge becomes useless by small errors (as far as the consequences are concerned)

\[ \Rightarrow \text{ classical (two-valued) logic is not tolerant against errors} \]

in typical Web scenarios errors / contradictions hardly avoidable:
  • different sources / different perspectives
  • use of information extraction methods (not 100% precise)

Deployment of more "robust" error-tolerant logics dealing with noisy/uncertain/graded knowledge
  • contextualized truth values
  • fuzzy logic
  • paraconsistent logics
  • probabilistic logics (Markov logic)
  • inconsistent query answering
Further Challenges II: Dynamics

• Classical KR setting: Information stored persistently and changes only sporadically. Queries more frequent than updates.

• majority of relevant Web information is very dynamic
• real-time processing essential
• established strategies for indexing and automated inferencing are inappropriate
• new, adequate complexity measures required (dynamic complexity?)
Conclusion & Credo

For logicians:

- more flexibility in response to different practical requirements
- more emphasis on modeling support
- enhance capabilities to deal with noisy data
- take into account that knowledge becomes more and more dynamic

For “other people”:

- have at least one logician in your group and feed them well
- many practical problems when analysed properly have a logical underpinning
- logic in the Semantic Web is not a “closed case”, there are many interesting open problems (and there will be more)