Data, Ontologies, Rules, and the Return of the Blank Node

Markus Krötzsch
Knowledge-Based Systems, TU Dresden

Virtual

23-27 October 2022, Hangzhou, China
Insights that Unite Us

Vocabulary has Meaning
• Identity of "resources"
• Important online (HTTP semantics)
• Important in ontologies (vocabulary)

Knowledge is for Sharing
• Shared conceptualisation
• Declarativity
• Openness & standards

Being Pedantic is OK
• Specifications are valued
• Formal semantics matters
• Caring about details
Insights that Unite Us

Vocabulary has Meaning
- Identity of “resources”
- Important online (HTTP semantics)
- Important in ontologies (vocabulary)
Insights that Unite Us

Vocabulary has Meaning
- Identity of “resources”
- Important online (HTTP semantics)
- Important in ontologies (vocabulary)

Knowledge is for Sharing
- Shared conceptualisation
- Declarativity
- Openness & standards
Insights that Unite Us

**Vocabulary has Meaning**
- Identity of “resources”
- Important online (HTTP semantics)
- Important in ontologies (vocabulary)

**Knowledge is for Sharing**
- Shared conceptualisation
- Declarativity
- Openness & standards

**Being Pedantic is OK**
- Specifications are valued
- Formal semantics matters
- Caring about details
The Triple

- Subject
- Predicate
- Object

Great Advantages
- uniformity & simplicity
- scalability
- structural compatibility
- and the powerful "graph" metaphor

But ... the world is complex
- Other communities prefer less normalisation (RDBs, JSON, ...)
- OWL axioms and ShaCL shapes do not fit into single triples
- Modern Knowledge Graphs are not so simple ...
The Triple

Great Advantages

• uniformity & simplicity
• scalability
• structural compatibility
• and the powerful “graph” metaphor

But... the world is complex

• Other communities prefer less normalisation (RDBs, JSON, ...)  
• OWL axioms and ShaCL shapes do not fit into single triples  
• Modern Knowledge Graphs are not so simple...
The Triple

Great Advantages
- uniformity & simplicity
- scalability
- structural compatibility
- and the powerful “graph” metaphor

But … the world is complex
- Other communities prefer less normalisation (RDBs, JSON, …)
- OWL axioms and ShaCL shapes do not fit into single triples
- Modern Knowledge Graphs are not so simple …
The Knowledge Graph of Wikipedia

- Free, community-built knowledge base
- Large and dynamic:
  - >100,000,000 items (since 19 Oct 2022)
  - >10,000 properties
  - >14.2B Triples in RDF
  - >20,000,000 edits/month
- International, active community
  - >20,000 active users (contributing during a month)
- Many uses

Fig.: Wikidata is celebrating its 10th Birthday on 29th Oct 2022
instance of

human

› 1 reference
Each Wikidata statement turns into several triples

Fig. Sketch of reified statement
Each Wikidata statement turns into several triples

Fig. Sketch of reified statement

**Wikidata SPARQL & RDF usage**

- Official IRI mapping and vocabulary
- Free RDF dumps + LOD exports
- Public live SPARQL endpoint at [https://query.wikidata.org/](https://query.wikidata.org/) (BlazeGraph)

*Key technologies for community and 3rd party users!*
From objects to triples ... and back?

The OWL approach:

• Ontological information may be intertwined with KG
• What reasoning we want may depend on context
• We may need new semantic interpretations for our data
From objects to triples … and back?

The OWL approach:

- Ontological information may be intertwined with KG
- What reasoning we want may depend on context
- We may need new semantic interpretations for our data
From objects to triples ... and back?

The OWL approach:

Great for what it does ... but not always what we need

- Ontological information may be intertwined with KG
- What reasoning we want may depend on context
- We may need new semantic interpretations for our data
New semantic interpretations for data

Wikidata says: “universe — has part(s) of the class — astronomical object”
New semantic interpretations for data

Wikidata says: “universe — has part(s) of the class — astronomical object”
New semantic interpretations for data

Wikidata says: “universe — has part(s) of the class — astronomical object”
Example:

Find all cities

```
SELECT ?city WHERE {
}
```

- Ontological information may be intertwined with KG
- What reasoning we want may depend on context
- We may need new semantic interpretations for our data
**Example:** Find all cities

```
SELECT ?city WHERE {
}
```
Example: Find all cities

```
SELECT ?city WHERE {
}
```

- Ontological information may be intertwined with KG
- What reasoning we want may depend on context
- We may need new semantic interpretations for our data
Example: Find all cities

```
SELECT ?city WHERE {
}
```

- Ontological information may be intertwined with KG
- What reasoning we want may depend on context
- We may need new semantic interpretations for our data
Example: Find all cities

```
SELECT ?city WHERE {
}
```

- Ontological information may be intertwined with KG
- What reasoning we want may depend on context
- We may need new semantic interpretations for our data
Example: Find all cities

SELECT ?city WHERE {
}

- Ontological information may be intertwined with KG ✓
- What reasoning we want may depend on context ✓
- We may need new semantic interpretations for our data ✓
What kind of semantics can we implement in SPARQL?

- RDFS
- OWL QL
- OWL RL
- OWL EL
- OWL 2 DL
SPARQL to the Rescue?

What kind of semantics can we implement in SPARQL?

- RDFS ✓
- OWL QL ✓
- OWL RL ×
- OWL EL ×
- OWL 2 DL ×

Theorem: SPARQL cannot express reasoning tasks that are harder than NLogSpace.
SPARQL to the Rescue?

What kind of semantics can we implement in SPARQL?

- RDFS ✓
- OWL QL ✓
- OWL RL
- OWL EL
- OWL 2 DL

Theorem: SPARQL cannot express reasoning tasks that are harder than NLogSpace.
SPARQL to the Rescue?

What kind of semantics can we implement in SPARQL?

- RDFS ✓
- OWL QL ✓
- OWL RL
- OWL EL
- OWL 2 DL

Fig. SPARQL pattern for finding empty OWL QL classes from [Bischof et al. ISWC 2014]

\[
x (sCO | eqC | ^eqC | intListMember | owl:someValuesFrom | (owl:onProperty / (INV | SpoEQ)^)° "owl:onProperty | rdfs:domain | rdfs:range))° ?C .
{ ?C subClassOf owl:Nothing) UNION
{ ?C subClassOf ?D1 \{ {?C subClassOf ?D2} UNION univClass[?D2] \{ ?D1 disjointClasses ?D2} UNION
{ ?V rdf:type owl:AllDisjointClasses . twoMembers[?V, ?D1, ?D2] } UNION
{ {?C (owl:onProperty / (INV | SpoEQ)^) ?P .
{ {?P subPropertyOf owl:bottomObjectProperty}) UNION
{ {?P subPropertyOf ?Q1 \{ {?P subPropertyOf ?Q2} UNION univProperty[?Q2] \{ ?Q1 (owl:propertyDisjointWith | ~owl:propertyDisjointWith) ?Q2) UNION
}
What kind of semantics can we implement in SPARQL?

- RDFS ✓
- OWL QL ✓
- OWL RL ✗
- OWL EL
- OWL 2 DL

---

**Fig.** SPARQL pattern for finding empty OWL QL classes from [Bischof et al. ISWC 2014]

$$\exists (s\ CO | eqC | \neg eqC | int\ List\ Member | owl:\ someValuesFrom | (owl\ :onProperty / (INV | SpoEq))^+ / (\owl\ :onProperty | rdfs\ :domain | rdfs\ :range))+ ?C . \{ ?C \ sub\ Class\ Of owl\ :Nothing \ UNION \{ ?C \ sub\ Class\ Of ?D1 [\{ ?C \ sub\ Class\ Of ?D2 \} \ UNION \{ ?D1 \ disj\ Ont\ Classes \ ?D2 \} \ UNION \{ ?V \ rdf\ :type \owl\ :All\ Disjoint\ Classes . \ two\ Members[?V, ?D1, ?D2] \} \} \ UNION \{ ?C (owl\ :onProperty / (INV | SpoEq))^+ \ ?P . \{ ?P \ sub\ Property\ Of owl\ :bottom\ Object\ Property \ UNION \{ ?P \ sub\ Property\ Of ?Q1 [\{ ?P \ sub\ Property\ Of ?Q2 \} \ UNION \{ ?Q1 \ owl\ :property\ Disjoint\ With \ ?owl\ :property\ Disjoint\ With \ ?Q2 \} \ UNION \{ ?V \ rdf\ :type \owl\ :All\ Disjoint\ Properties . \ two\ Members[?V, ?Q1, ?Q2] \} \} \} \} \} \}$$
SPARQL to the Rescue?

What kind of semantics can we implement in SPARQL?

- RDFS ✓
- OWL QL ✓
- OWL RL ×
- OWL EL
- OWL 2 DL

**Theorem:**
SPARQL cannot express reasoning tasks that are harder than NLogSpace.

**Fig.** SPARQL pattern for finding empty OWL QL classes from [Bischof et al. ISWC 2014]
What kind of semantics can we implement in SPARQL?

- RDFS ✓
- OWL QL ✓
- OWL RL ×
- OWL EL ×
- OWL 2 DL

Fig. SPARQL pattern for finding empty OWL QL classes from [Bischof et al. ISWC 2014]

```
x (SCO | eqC | ~eqC | INTLISTMEMBER | owl:someValuesFrom |
  {?C subClassOf owl:Nothing} UNION |
  {?C subClassOf ?D1 {?C subClassOf ?D2} UNION UNIVCLASS(?D2)} { |
    {?D1 disjointClasses ?D2} UNION |
    {?V rdf:type owl:AllDisjointClasses . TWOMEMBERS(?V, ?D1, ?D2)} } UNION |
  {?C (owl:onProperty / (INV | SpoEQQ)*} ?P . |
    {?P subPropertyOf owl:bottomObjectProperty} UNION |
    {?P subPropertyOf ?Q1 {?P subPropertyOf ?Q2} UNION UNIVPROPERTY(?Q2)} { |
      {?Q1 (owl:propertyDisjointWith | ~owl:propertyDisjointWith) ?Q2} UNION |
      {?V rdf:type owl:AllDisjointProperties . TWOMEMBERS(?V, ?Q1, ?Q2)} } } )
```

Theorem:
SPARQL cannot express reasoning tasks that are harder than NLogSpace.
SPARQL to the Rescue?

What kind of semantics can we implement in SPARQL?

- RDFS ✓
- OWL QL ✓
- OWL RL ✗
- OWL EL ✗
- OWL 2 DL ✗

Fig. SPARQL pattern for finding empty OWL QL classes from [Bischof et al. ISWC 2014]

Theorem:
SPARQL cannot express reasoning tasks that are harder than NLogSpace.
The Rules Layer on the Cake

Where do we get something more powerful than SPARQL?
The Rules Layer on the Cake

Where do we get something more powerful than SPARQL?
Well, it has always been there . . .
The Rules Layer on the Cake

Where do we get something more powerful than SPARQL?
Well, it has always been there . . .

A Complicated Relationship

- since 2000: **Notation 3 Logic** developed (incl. RDF rules)
- 2004: **SWRL** W3C member submission
- 2005: W3C charters WG for **Rule Interchange Format** (RIF)
- 2009: **OWL RL** becomes W3C Rec
- 2010: RIF enters recommendation
- 2010: 1st **Datalog 2.0 Workshop**
- all this time: SemWeb practitioners use various forms of rules (except RIF)
Datalog: a very simple rule language

- Simple predicate-logic rules:
  \[ \text{rdf:type}(X, Y) \land \text{rdfs:subClassOf}(Y, Z) \rightarrow \text{rdf:type}(X, Z) \]
- Conjunctive query patterns + recursion
- Polynomial time data complexity
- Fast reasoners with RDF support:
  - RDFox [Nenov et al. ISWC 2015],
  - VLog [Carral et al. ISWC 2019]
The Simplest Rule Language

**Datalog: a very simple rule language**

- Simple predicate-logic rules:
  \[
  \text{rdf:type}(X, Y) \land \text{rdfs:subClassOf}(Y, Z) \rightarrow \text{rdf:type}(X, Z)
  \]

- Conjunctive query patterns + recursion
- Polynomial time data complexity
- Fast reasoners with RDF support:
  RDFox [Nenov et al. ISWC 2015],
  VLog [Carral et al. ISWC 2019]

**Nothing complicated**

- No negation
- No disjunction
- No built-ins or filters
OWL with Datalog

- **KG (ontologies)**
- **Rule engine**
- **Results**

<table>
<thead>
<tr>
<th>OWL with Datalog</th>
<th>SPARQL</th>
<th>Datalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDFS</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OWL QL</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OWL RL</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>OWL EL</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>OWL DL</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

- SPARQL
  - [Bischof et al., ISWC 2014] + SPARQL as Datalog
- OWL QL
  - [Krötzsch, ISWC 2012]
- OWL RL
  - [Krötzsch, IJCAI 2011]
- OWL EL
  - Too complex: N2ExpTime > P
- OWL DL
  - Too complex: N2ExpTime > P

Markus Krötzsch, TU Dresden
How complicated are these rules?

\[\text{subClassOf}(?C, ?D1) \land \text{subClassOf}(?C, ?D2) \land \text{isInter}(?D12, ?D1, ?D2) \rightarrow \text{subClassOf}(?C, ?D12)\]
How complicated are these rules?

\[
\text{subClassOf}(\text{C}, \text{D}_1) \land \text{subClassOf}(\text{C}, \text{D}_2) \land \\
\text{isInter}(\text{D}_1, \text{D}_1, \text{D}_2) \rightarrow \text{subClassOf}(\text{C}, \text{D}_{12})
\]

**RDF encoding of** `SubClassOf(A ObjectIntersectionOf(D1 D2))`:

```
A rdfs:subClassOf “D12” owl:intersectionOf n1 rdf:next n2 rdf:next rdf:nil

rdfs:subClassOf

owl:intersectionOf

rdf:next

rdf:next

rdf:next

D1

D2
```
How complicated are these rules?

\[
\text{subClassOf}(\text{C, D1)} \land \text{subClassOf}(\text{C, D2)} \land \\
\text{isInter}(\text{D12, D1, D2)} \rightarrow \text{subClassOf}(\text{C, D12)}
\]

**RDF encoding of** SubClassOf(A ObjectIntersectionOf( D1 D2 )):

```
A rdfs:subClassOf “D12” owl:intersectionOf n1 rdf:next n2 rdf:next rdf:next
D1 rdf:next rdf:next rdf:next
D2

TRIPLE(?D12, owl:intersectionOf, ?L1) \land
TRIPLE(?N1, rdf:next, ?N2) \land TRIPLE(?N2, rdf:next, rdf:nil) \land
TRIPLE(?N1, rdf:first, ?D1) \land TRIPLE(?N2, rdf:first, ?D2) \rightarrow \text{isInter}(\text{D12, D1, D2)}
```

\sim 10 reasoning rules and 36 parsing rules for much of OWL EL [ECAI Tutorial 2020]
So ... what have we accomplished?

- From ontologies to data
- From W3C semantics to user-space semantics
- From dedicated reasoners to ≤50 lines of code

However . . .

- Triples are not enough! [Krötzsch, IJCAI 2011 & ISWC 2012]
- Flexibility has a performance cost (VLog on Snomed: 2min)
- Getting rules right is not easy
So ... what have we accomplished?

- From ontologies to data
- From W3C semantics to user-space semantics
- From dedicated reasoners to <50 lines of code

However ...

- Triples are not enough! [Krötzsch, IJCAI 2011 & ISWC 2012]
- Flexibility has a performance cost (VLog on Snomed: 2min)
- Getting rules right is not easy
Why Datalog is not enough

**Observation 1:** Encoding complex objects in RDF requires auxiliary nodes

**Observation 2:** Datalog can only produce new relationships among existing nodes

```datalog
isInter(?D12, ?D1, ?D2) →
  TRIPLE(?D12, owl:intersectionOf, _:n1) ∧
  TRIPLE(_:n1, rdf:next, _:n2) ∧
  TRIPLE(_:n2, rdf:next, rdf:nil) ∧
  TRIPLE(_:n1, rdf:first, ?D1) ∧
  TRIPLE(_:n2, rdf:first, ?D2)
```
Why Datalog is not enough

**Observation 1:** Encoding complex objects in RDF requires auxiliary nodes
**Observation 2:** Datalog can only produce new relationships among existing nodes

∽ **Major Weakness:** We can parse complex RDF structures but not create them!
Why Datalog is not enough

Observation 1: Encoding complex objects in RDF requires auxiliary nodes
Observation 2: Datalog can only produce new relationships among existing nodes

→ **Major Weakness:** We can parse complex RDF structures but not create them!

**Solution:** Enable value invention in rules

\[
\text{isInter(?D12, ?D1, ?D2)} \rightarrow \text{TRIPLE(?D12, owl:intersectionOf, _:n1)} \land \\
\text{TRIPLE(_,n1, rdf:next, _:n2)} \land \text{TRIPLE(_,n2, rdf:next, rdf:nil)} \land \\
\text{TRIPLE(_,n1, rdf:first, ?D1)} \land \text{TRIPLE(_,n2, rdf:first, ?D2)}
\]

→ “Existential Rules”
Existential Rules: a simple rule language

- Simple predicate-logic rule, like Datalog
- Conjunctive query patterns + recursion + value invention
- Query answering **undecidable**
- Fast reasoners with RDF support: RDFox [Nenov et al. ISWC 2015], VLog [Carral et al. ISWC 2019]

Still nothing complicated

- No negation
- No disjunction
- No built-ins or filters
From Datalog to Existential Rules

Existential Rules: a simple rule language

- Simple predicate-logic rule, like Datalog
- Conjunctive query patterns + recursion + value invention
- Query answering **undecidable**
- Fast reasoners with RDF support: RDFox [Nenov et al. ISWC 2015], VLog [Carral et al. ISWC 2019]

Reasoners recursively apply rules and add new values, hoping for termination

~ Most tool support is for rule sets that “terminate”

Markus Krötzsch, TU Dresden

Data, Ontologies, Rules, and the Return of the Blank Node
The Power of Terminating Existential Rules

Recall:
- Datalog cannot see negative information (no negation)
- Datalog cannot solve non-polynomial tasks

Clear:
- Existential rules cannot see negative information (no negation)
- Terminating existential rules can only solve computable tasks
The Power of Terminating Existential Rules

Recall:
- Datalog cannot see negative information (no negation)
- Datalog cannot solve non-polynomial tasks

Clear:
- Existential rules cannot see negative information (no negation)
- Terminating existential rules can only solve computable tasks

Theorem* [Bourgaux et al., KR 2021]
Terminating existential rules can solve every computable task that does not depend on negative information.

* Terms and conditions apply; see paper
The Return of the Blank Node?

Markus Krötzsch, TU Dresden

Data, Ontologies, Rules, and the Return of the Blank Node
### Summary: User-level specifications for interpreting triples

<table>
<thead>
<tr>
<th>KG (Ontology)</th>
<th>User-level spec.</th>
<th>Rule engine</th>
<th>Results</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SPARQL</th>
<th>Triples-only Datalog</th>
<th>Datalog</th>
<th>Existential Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDFS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OWL QL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OWL RL</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>OWL EL</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>OWL DL</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>
Summary: What really matters

We have split the world into triples, but still struggle to put it back together

- SPARQL is a start, but limited
- Rules are powerful, but subtle
Summary: What really matters

We have split the world into triples, but still struggle to put it back together

- SPARQL is a start, but limited
- Rules are powerful, but subtle

The era of standard semantics has ended . . .

- Real-world complexity eludes single triples and single (OWL) axioms
- Our needs change from context to context
- Managing knowledge requires more than one AI method
Summary: What really matters

We have split the world into triples, but still struggle to put it back together

- SPARQL is a start, but limited
- Rules are powerful, but subtle

The era of standard semantics has ended . . .

- Real-world complexity eludes single triples and single (OWL) axioms
- Our needs change from context to context
- Managing knowledge requires more than one AI method

. . . but declarative, shareable meaning is more relevant than ever.

- Explanation and accountability requires shared concepts
- Declarativity is key to interoperability and scalability
- Knowledge is human, it will not go away
What’s next for the “Semantic Web”?

We have split the world into triples, but still struggle to put it back together. The era of standard semantics has ended . . .

. . . but declarative, shareable meaning is more relevant than ever.

What shall we do about it?
What’s next for the “Semantic Web”?  

We have split the world into triples, but still struggle to put it back together. The era of standard semantics has ended . . .

. . . but declarative, shareable meaning is more relevant than ever.

What shall we do about it?

1. Appreciate what we stand for
What’s next for the “Semantic Web”?

We have split the world into triples, but still struggle to put it back together. The era of standard semantics has ended . . .

. . . but declarative, shareable meaning is more relevant than ever.

What shall we do about it?

1. Appreciate what we stand for
What’s next for the “Semantic Web”?

We have split the world into triples, but still struggle to put it back together.
The era of standard semantics has ended . . .
    . . . but declarative, shareable meaning is more relevant than ever.

What shall we do about it?

1. Appreciate what we stand for
2. Spread our values:
   meaning, sharing, precision
3. Focus on big challenges:
   flexibility, scalability, human collaboration, . . .
4. Find new ways to agree:
   on “standards”, on concepts, on models
5. Engage with communities
   who care and can help
What’s next for the “Semantic Web”?  

We have *split the world into triples*, but still struggle to put it back together. The era of *standard semantics has ended* . . .  

. . . but *declarative, shareable meaning* is more relevant than ever.

“Knowledge is human, it will not go away”


I would like to thank all my co-authors and collaborators on works mentioned here and all other works that contribute to the big vision.

Special thanks are due to Denny Vrandecic and Lydia Pintscher, the past and current leads of Wikidata development (but not featured enough in my reference list).

My ongoing research in Dresden is supported by a number of projects, in particular by DFG in project 389792660 (TRR 248, Center for Perspicuous Systems), by the BMBF under project ScaDS.AI, by BMBF and DAAD in project 57616814 (SECAI: School of Embedded and Composite AI), and by the Center for Advancing Electronics Dresden (cfaed).

We are always looking for new ideas. How about joining us? (Just get in touch, and we will see what can be done :-)