ISWC2022
Hangzhou, China

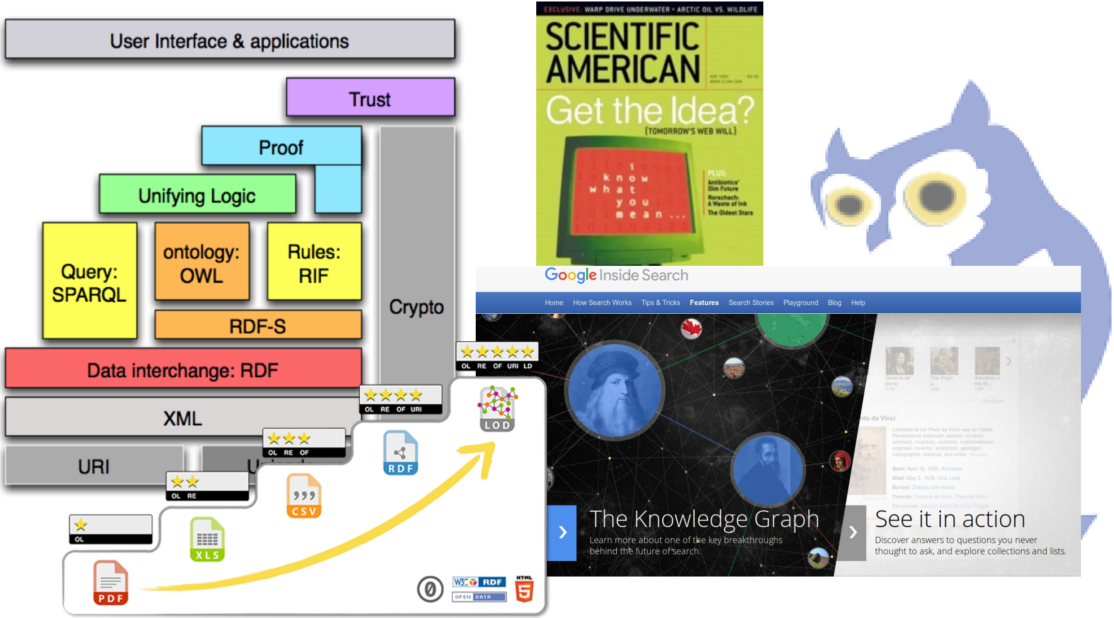
Data, Ontologies, Rules, and the Return of the Blank Node

Markus Krötzsch
Knowledge-Based Systems, TU Dresden

Virtual

23-27 October 2022, Hangzhou, China

TECHNISCHE UNIVERSITÄT DRESDEN


User Interface & applications
Trust
Proof
Unifying Logic
Query: SPARQL
ontology: OWL
Rules: RIF
RDF-S
Crypto
Data interchange: RDF
XML
URI
OL
OL RE
OL RE OF
OL RE OF URI
OL RE OF URI LD
PDF
XLS
CSV
RDF
LOD
W3C RDF
OPEN DATA
HTML
EXCLUSIVE: WARP DRIVE UNDERWATER • ARCTIC OIL VS. WILDLIFE
SCIENTIFIC AMERICAN
Get the Idea?
(TOMORROW'S WEB WILL)
I know what you mean...
Google Inside Search
Home
How Search Works
Tips & Tricks
Features
Search Stories
Playground
Blog
Help
The Knowledge Graph
Learn more about one of the key breakthroughs behind the future of search.
See it in action
Discover answers to questions you never thought to ask, and explore collections and lists.

Insights that Unite Us

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

The Triple

Great Advantages

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

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

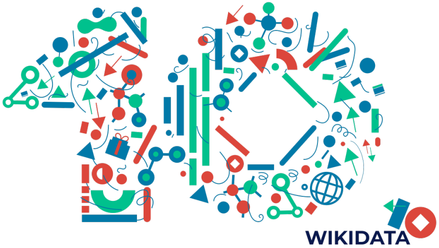


Fig.: Wikidata is celebrating its 10th Birthday on 29th Oct 2022

Tim Berners-Lee (Q80)

British computer scientist

edit

TimBL | Sir Tim Berners-Lee | Timothy John Berners-Lee | TBL | Tim Berners Lee | T. Berners-Lee | T Berners-Lee | Tim Berners-Lee | T.J. Berners-Lee

Tim Berners-Lee (Q80)

British computer scientist

edit

TimBL | Sir Tim Berners-Lee | Timothy John Berners-Lee | TBL | Tim Berners Lee | T. Berners-Lee | T Berners-Lee | Tim Berners-Lee | T.J. Berners-Lee

instance of	human 1 reference	edit

Tim Berners-Lee (Q80)

British computer scientist

edit

TimBL | Sir Tim Berners-Lee | Timothy John Berners-Lee | TBL | Tim Berners Lee | T. Berners-Lee | T Berners-Lee | Tim Berners-Lee | T.J. Berners-Lee

instance of	human 1 reference	edit
employer	CERN start time 1984 end time 1994 position held Fellow 0 references + add reference	edit

Tim Berners-Lee (Q80)

British computer scientist

edit

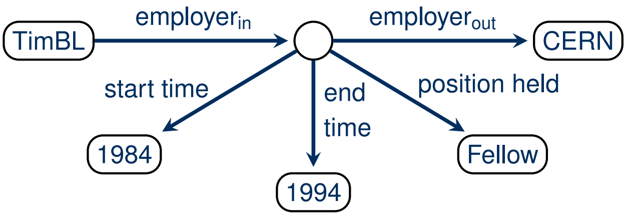
TimBL | Sir Tim Berners-Lee | Timothy John Berners-Lee | TBL | Tim Berners Lee | T. Berners-Lee | T Berners-Lee | Tim Berners-Lee | T.J. Berners-Lee

instance of	human		edit
	1 reference		
employer	CERN		edit
	start time	1984	
	end time	1994	
	position held	Fellow	
	0 references		
			+ add reference
award received	Queen Elizabeth Prize for Engineering		edit
	point in time	2013	
	together with	Robert Kahn	
		Vint Cerf	
		Louis Pouzin	
		Marc Andreessen	
	1 reference		

Wikidata and the Semantic Web

Each Wikidata statement turns into several triples

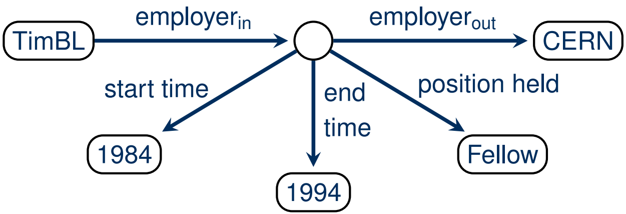
Fig. Sketch of reified statement ►



Wikidata and the Semantic Web

Each Wikidata statement turns into several triples

Fig. Sketch of reified statement ►



Wikidata SPARQL & RDF usage

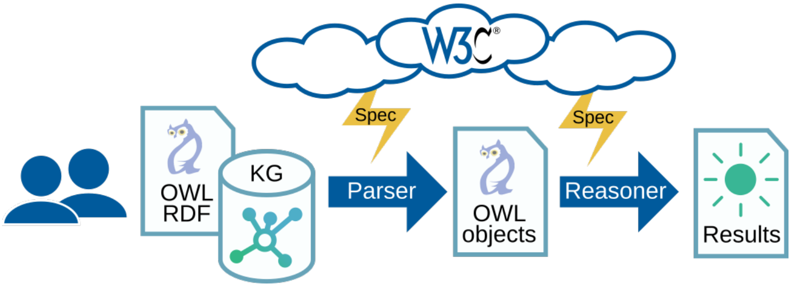
- Official IRI mapping and vocabulary
- Free RDF dumps + LOD exports
- Public live SPARQL endpoint at `https://query.wikidata.org/` (BlazeGraph)

Key technologies for community and 3rd party users!

From objects to triples . . . and back?

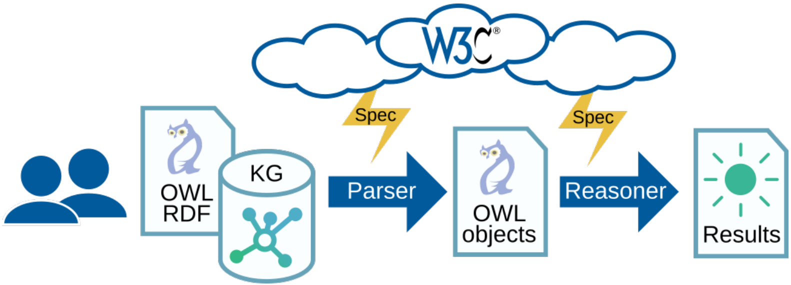
From objects to triples . . . and back?

The OWL approach:



From objects to triples . . . and back?

The OWL approach:



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

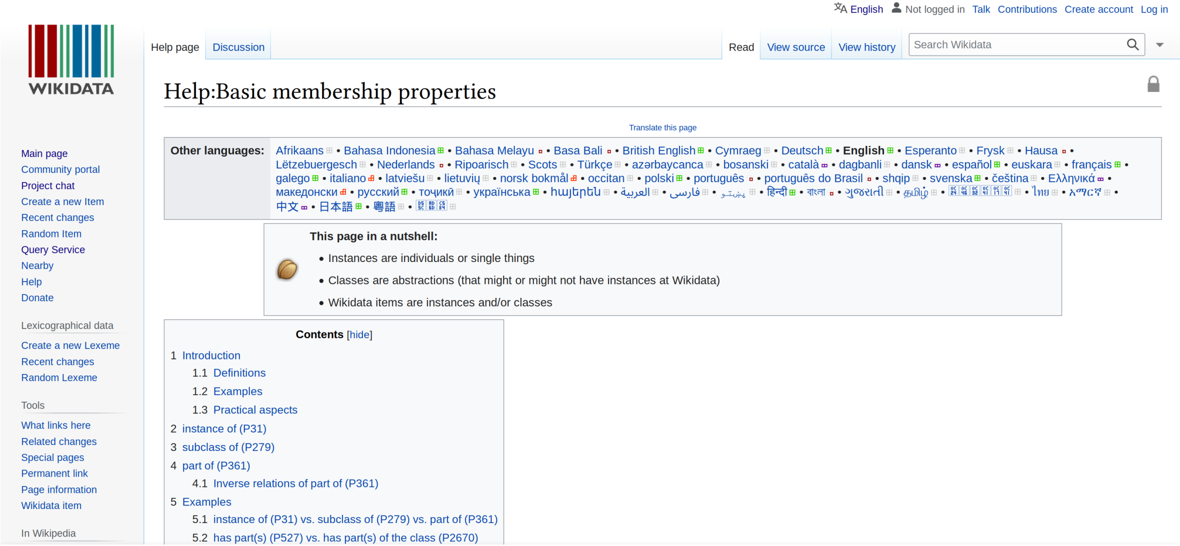
- Ontological information may be intertwined with KG
- What reasoning we want may depend on context
- We may need new semantic interpretations for our data

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”



English Not logged in Talk Contributions Create account Log in

WIKIDATA

Help page | Discussion | Read | View source | View history | Search Wikidata

Help:Basic membership properties

Translate this page

Other languages: Afrikaans • Bahasa Indonesia • Bahasa Melayu • Basa Bali • British English • Cymraeg • Deutsch • **English** • Esperanto • Frysk • Hausa • Lëtzebuergesch • Nederlands • Ripoarisch • Scots • Türkçe • azərbaycanca • bosanski • català • dagbanli • dansk • español • euskara • français • galego • italiano • latviešu • lietuvių • norsk bokmål • occitan • polski • português • português do Brasil • shqip • svenska • čeština • Ελληνικά • македонски • русский • тоҷикӣ • українська • հայերեն • العربية • فارسی • پښتو • हिन्दी • বাংলা • ગુજરાતી • தமிழ் • ไทย • አማርኛ • 中文 • 日本語 • 粵語 •

This page in a nutshell:

- Instances are individuals or single things
- Classes are abstractions (that might or might not have instances at Wikidata)
- Wikidata items are instances and/or classes

Contents [hide]

Main page
Community portal
Project chat
Create a new Item
Recent changes
Random Item
Query Service
Nearby
Help
Donate

Lexicographical data

Create a new Lexeme
Recent changes
Random Lexeme

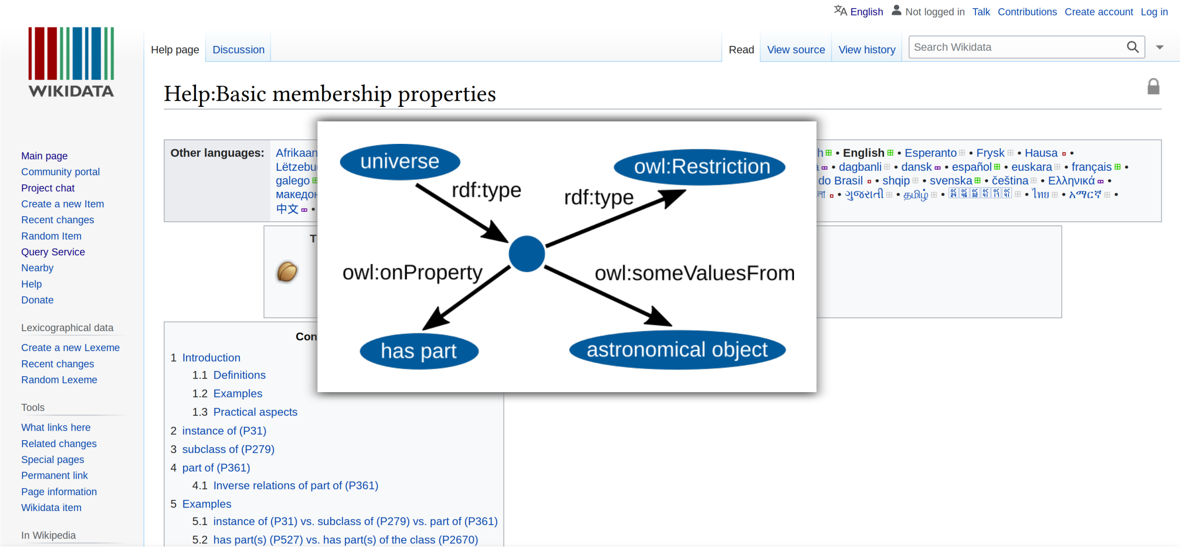
Tools

What links here
Related changes
Special pages
Permanent link
Page information
Wikidata item

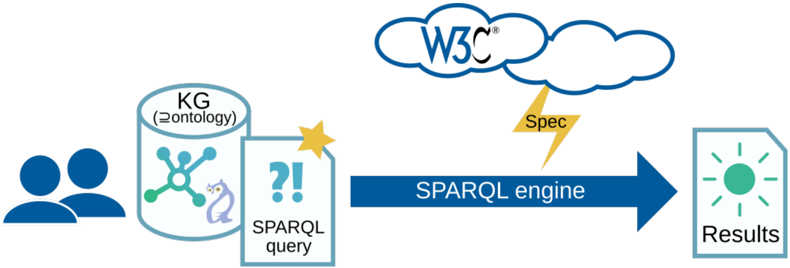
In Wikipedia

New semantic interpretations for data

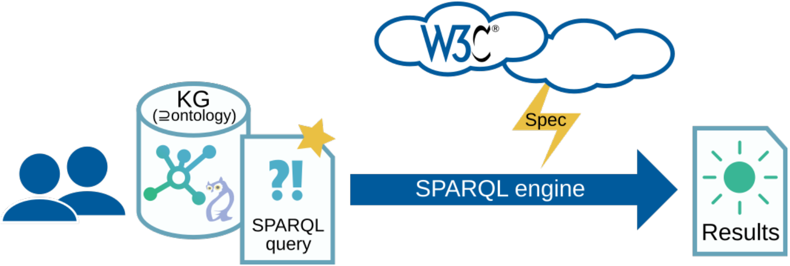
Wikidata says: “universe — has part(s) of the class — astronomical object”



SPARQL to the Rescue



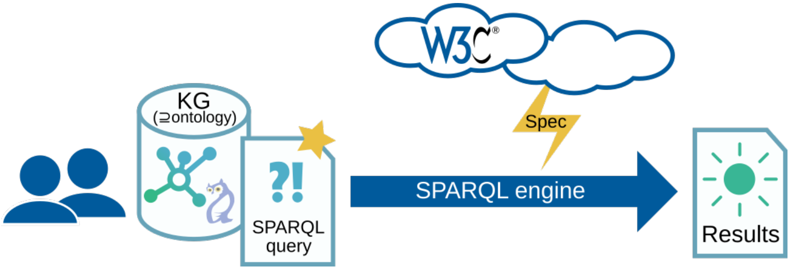
SPARQL to the Rescue



Example: Find all cities

```
SELECT ?city WHERE {
  ?city  rdf:type/rdfs:subClassOf*  <https://example.org/city> .
}
```

SPARQL to the Rescue

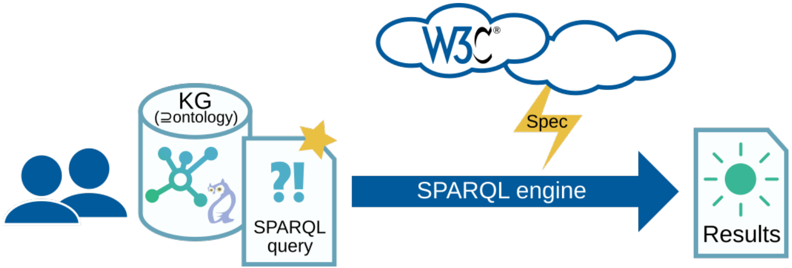


Example: Find all cities

```
SELECT ?city WHERE {
  ?city  rdf:type/rdfs:subClassOf*  <https://example.org/city> .
}
```

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

SPARQL to the Rescue

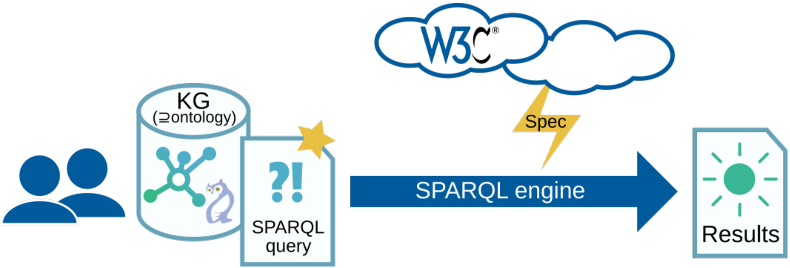


Example: Find all cities

```
SELECT ?city WHERE {
  ?city  rdf:type/rdfs:subClassOf*  <https://example.org/city> .
}
```

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

SPARQL to the Rescue

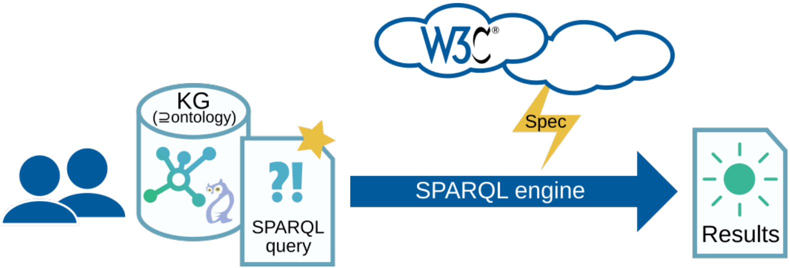


Example: Find all cities

```
SELECT ?city WHERE {
  ?city  rdf:type/rdfs:subClassOf*  <https://example.org/city> .
}
```

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

SPARQL to the Rescue



Example: Find all cities

```
SELECT ?city WHERE {
  ?city  rdf:type/rdfs:subClassOf*  <https://example.org/city> .
}
```

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

SPARQL to the Rescue?

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

SPARQL to the Rescue?

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

```
x (sCO | eqC | ^eqC | intListMember | owl:someValuesFrom |
   (owl:onProperty / (Inv | SpoEqp)* / (^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 | SpoEqp)*) ?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]}
    }
  }}
}
```

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

SPARQL to the Rescue?

What kind of semantics can we implement in SPARQL?

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

```
x (sCO | eqC | ^eqC | intListMember | owl:someValuesFrom |
   (owl:onProperty / (Inv | SpoEqp)* / (^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 | SpoEqp)*) ?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]}
    }
  }}
}
```

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

SPARQL to the Rescue?

What kind of semantics can we implement in SPARQL?

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

```
x (sCO | eqC | ^eqC | intListMember | owl:someValuesFrom |
    (owl:onProperty / (Inv | SpoEqp)* / (^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 | SpoEqp)*) ?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]}
      }
   }}
}
```

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.

SPARQL to the Rescue?

What kind of semantics can we implement in SPARQL?

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

```
x (sCO | eqC | ^eqC | intListMember | owl:someValuesFrom |
    (owl:onProperty / (Inv | SpoEqp)* / (^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 | SpoEqp)*) ?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]}
      }
   }}
}
```

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.

SPARQL to the Rescue?

What kind of semantics can we implement in SPARQL?

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

```
x (sCO | eqC | ^eqC | intListMember | owl:someValuesFrom |
   (owl:onProperty / (Inv | SpoEqp)* / (^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 | SpoEqp)*) ?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]}
    }
  }}
}
```

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.

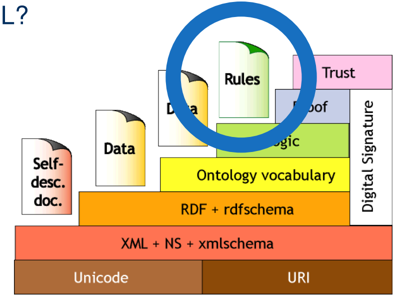
The Rules Layer on the Cake

Where do we get something more powerful than SPARQL?

The Rules Layer on the Cake

Where do we get something more powerful than SPARQL?

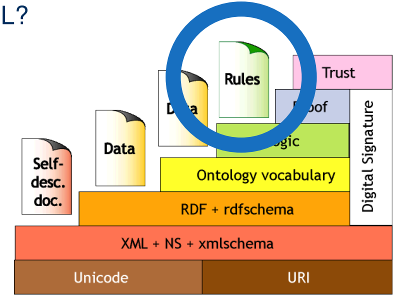
Well, it has always been there . . .



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)

The Simplest Rule Language

Datalog: a very simple rule language

- Simple predicate-logic rules:
  rdf:type$(X, Y) \wedge$ rdfs:subClassOf$(Y, Z) \rightarrow$ 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:
  rdf:type$(X, Y) \wedge$ rdfs:subClassOf$(Y, Z) \rightarrow$ 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

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

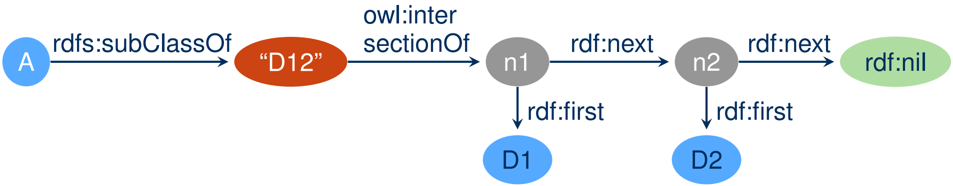
How complicated are these rules?

$$\text{subClassOf}(?C, ?D1) \wedge \text{subClassOf}(?C, ?D2) \wedge \\ \text{isInter}(?D12, ?D1, ?D2) \rightarrow \text{subClassOf}(?C, ?D12)$$

How complicated are these rules?

$$\text{subClassOf}(?C, ?D1) \wedge \text{subClassOf}(?C, ?D2) \wedge \text{isInter}(?D12, ?D1, ?D2) \rightarrow \text{subClassOf}(?C, ?D12)$$

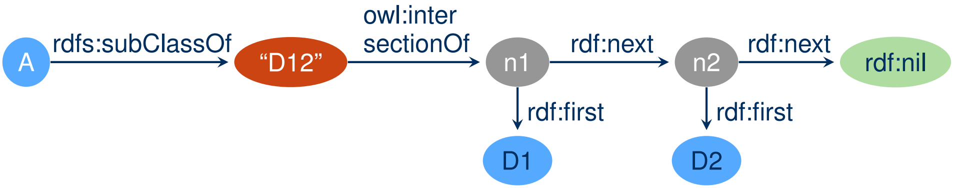
RDF encoding of `SubClassOf(`A` ObjectIntersectionOf( `$D1$` `$D2$` ) )`**:**



How complicated are these rules?

$$\text{subClassOf}(?C, ?D1) \wedge \text{subClassOf}(?C, ?D2) \wedge \text{isInter}(?D12, ?D1, ?D2) \rightarrow \text{subClassOf}(?C, ?D12)$$

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

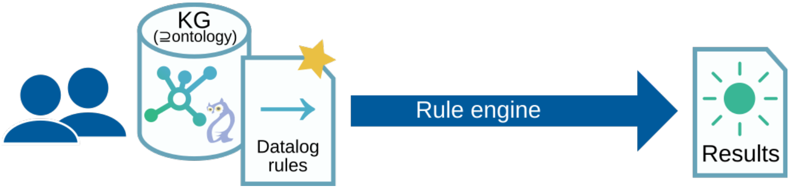


$$\text{TRIPLE}(?D12, \text{owl:intersectionOf}, ?L1) \wedge \text{TRIPLE}(?N1, \text{rdf:next}, ?N2) \wedge \text{TRIPLE}(?N2, \text{rdf:next}, \text{rdf:nil}) \wedge \text{TRIPLE}(?N1, \text{rdf:first}, ?D1) \wedge \text{TRIPLE}(?N2, \text{rdf:first}, ?D2) \rightarrow \text{isInter}(?D12, ?D1, ?D2)$$

$\leadsto$ **10 reasoning rules and 36 parsing rules for much of OWL EL** [ECAI Tutorial 2020]

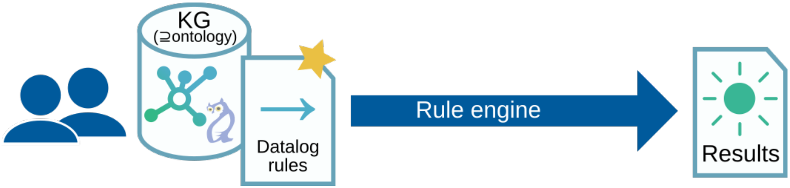
So . . . what have we accomplished?

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



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

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

$\leadsto$ **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

$\leadsto$ **Major Weakness:** We can parse complex RDF structures but not create them!

Solution: Enable value invention in rules

$$\begin{aligned}\mathrm{isInter}(?D12, ?D1, ?D2) \to{} & \mathrm{TRIPLE}(?D12, \mathrm{owl{:}intersectionOf}, \_{:}n1) \land \\ & \mathrm{TRIPLE}(\_{:}n1, \mathrm{rdf{:}next}, \_{:}n2) \land \mathrm{TRIPLE}(\_{:}n2, \mathrm{rdf{:}next}, \mathrm{rdf{:}nil}) \land \\ & \mathrm{TRIPLE}(\_{:}n1, \mathrm{rdf{:}first}, ?D1) \land \mathrm{TRIPLE}(\_{:}n2, \mathrm{rdf{:}first}, ?D2)\end{aligned}$$

$\leadsto$ "Existential Rules"

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]

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]

Still nothing complicated

- No negation
- No disjunction
- No built-ins or filters

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

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

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?

Summary: User-level specifications for interpreting triples

	SPARQL	Triples-only Datalog	Datalog	Existential Rules
RDFS	✓	✓	✓	✓
OWL QL	✓	✓	✓	✓
OWL RL	×	×	✓	✓
OWL EL	×	×	✓	✓
OWL DL	×	×	×	✓

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"?

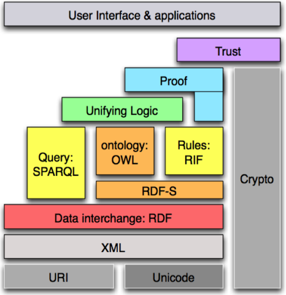
We have split the world into triples, but still struggle to put it back together.

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"?

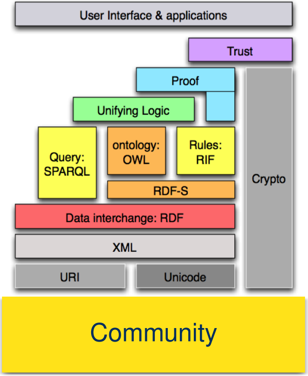
We have split the world into triples, but still struggle to put it back together.

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"?

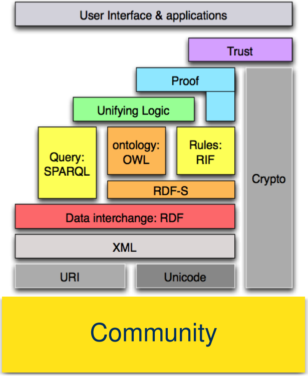
We have split the world into triples, but still struggle to put it back together.

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

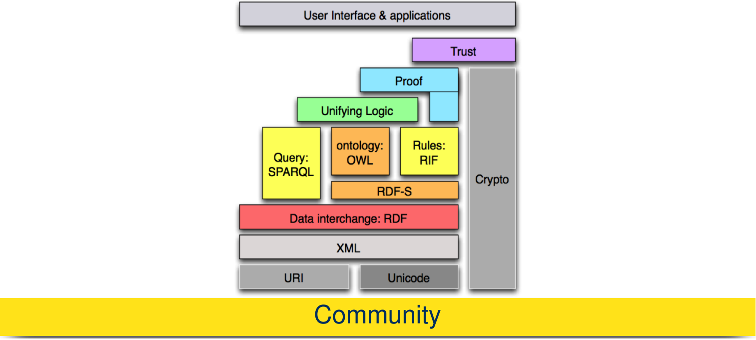


What’s next for the “Semantic Web”?

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”

References All titles are links that lead to a download page

[Krötzsch, IJCAI 2011] Markus Krötzsch: Efficient Rule-Based Inferencing for OWL EL. Proc. 22nd International Joint Conference on Artificial Intelligence (IJCAI'11), 2668–2673, 2011.

[Krötzsch, ISWC 2012] Markus Krötzsch: The Not-So-Easy Task of Computing Class Subsumptions in OWL RL. Proc. 11th International Semantic Web Conference (ISWC'11), 279–294, 2012.

[Bischof et al. ISWC 2014] Stefan Bischof, Markus Krötzsch, Axel Polleres, Sebastian Rudolph: Schema-Agnostic Query Rewriting in SPARQL 1.1. Proc. 13th International Semantic Web Conference (ISWC'14), volume 8796 of LNCS, 584–600, 2014.

[Nenov et al. ISWC 2015] Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, Jay Banerjee: RDFox: A Highly-Scalable RDF Store. Proc. 14th International Semantic Web Conference (ISWC'15) Part II: 3–20, 2015.

[Carral et al. ISWC 2019] David Carral, Irina Dragoste, Larry González, Ceriel Jacobs, Markus Krötzsch, Jacopo Urbani: VLog: A Rule Engine for Knowledge Graphs. Proc. 18th International Semantic Web Conference (ISWC'19) Part II, volume 11779 of LNCS, 2019.

[ECAI Tutorial 2020] David Carral, Markus Krötzsch, Jacopo Urbani: Practical Uses of Existential Rules in Knowledge Representation, tutorial at ECAI 2020.

[Bourgaux et al., KR 2021] Camille Bourgaux, David Carral, Markus Krötzsch, Sebastian Rudolph, Michaël Thomazo: Capturing Homomorphism-Closed Decidable Queries with Existential Rules, Proc. 18th International Conference on Principles of Knowledge Representation and Reasoning (KR'21), 141–150, 2021.

Acknowledgements

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 :-)