Reimagining the Semantic Web – a retrospective

Keynote address @ SWAT4HCLS 2023

Dr. Ora Lassila
Principal Technologist
Amazon Neptune
Reinterpreting the Semantic Web – a retrospective

Keynote address @ SWAT4HCLS 2023

Dr. Ora Lassila
Principal Technologist
Amazon Neptune
Who am I...

Principal Technologist, Amazon Neptune

Some accomplishments re: KR

- co-authored the original W3C RDF specification
- co-authored the seminal article on the Semantic Web
- designed and implemented the frame-based KR subsystem that flew on NASA’s “Deep Space 1” probe past the Asteroid Belt in 1998
- currently: co-chair of the W3C RDF-star WG

Education:

- Ph.D CS, Helsinki University of Technology
A brief history of graphs and ontologies

1730s: Graph theory (Euler)

1950s and onwards: Graphs as the essential underpinning of computer science

1960s: Social networks, “small-world experiment”, Erdős number (Milgram et al)

1960s-1970s: Network databases (CODASYL), semantic networks (Quillian et al)

1970s-1990s: Predicate logic as the foundation of Knowledge Representation (Hayes et al)

1997 and onwards: The Semantic Web, RDF, OWL, etc. (Lassila et al)

Today: Modern knowledge graphs and graph databases

3rd Century BCE: Categories & logic (Aristotle)

1730s: Taxonomical classification of plants and animals (Linnaeus)

1870s: Library classification (Dewey)

1900: Semantics, ontology and logic (Husserl)

1970s-1990s: Predicate logic as the foundation of Knowledge Representation (Hayes et al)
A brief history of graphs and ontologies

1730s: Graph theory (Euler)

1870s: Library classification (Dewey)

1900: Semantics, ontology and logic (Husserl)

1950s and onwards: Graphs as the essential underpinning of computer science

1960s: Social networks, “small-world experiment”, Erdős number (Milgram et al)

1960s-1970s: Network databases (CODASYL), semantic networks (Quillian et al)

1970s-1990s: Predicate logic as the foundation of Knowledge Representation (Hayes et al)

1997 and onwards: The Semantic Web, RDF, OWL, etc. (Lassila et al)

Today: Modern knowledge graphs and graph databases

This is what I want to talk about
Game plan

Original 2001 vision, and what led to it
Afterwards, what worked, what did not
A “re-interpretation” of the Semantic Web idea
The way forward
Semantic Web
Some early history from my viewpoint:

- late 1996: Tim Berners-Lee asks me a fateful question
- 1997: Early brainstorming on metadata and “KR on the Web”, start of the RDF work
- 1999: RDF becomes a W3C Recommendation
- 2000-2001: Tim, Jim Hendler, and me write down our vision for the SW
- early 2000s: DARPA DAML program, proliferation of W3C SW specifications
## Original 2001 Semantic Web vision

<table>
<thead>
<tr>
<th>Content:</th>
<th>Old: Web of <strong>documents</strong> (implicit/hidden semantics)</th>
<th>New: Web of <strong>data</strong> (explicit &amp; accessible semantics)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Humans</td>
<td>Humans &amp; <strong>Agents</strong></td>
</tr>
<tr>
<td>Standards:</td>
<td>Anticipate &amp; standardize everything up front</td>
<td>&quot;<strong>Delayed semantic commitment</strong>&quot;</td>
</tr>
</tbody>
</table>
Semantic Web: a new vision of the Web?

Why?

• original Web facilitated sharing of documents, but not really sharing of data
Semantic Web: a new vision of the Web?

Semantic Web: KR for the Web?

Why?

• metadata
• digital libraries
• better search results
• etc.
Semantic Web: a new vision of the Web?

Semantic Web: KR for the Web?

Semantic Web: KR **using Web technologies**!

**Why?**

- well understood, lots of software support, widely deployed
- “networking friendly” (HTTP goes through firewalls, etc.)
- prevailing mindset of distributed systems
- etc.
Semantic Web: a new vision of the Web?

Semantic Web: KR for the Web?

Semantic Web: KR using Web technologies!

Why?

• well understood, lots of software support, widely deployed
• “networking friendly” (HTTP goes through firewalls, etc.)
• prevailing mindset of distributed systems sharing
• etc.
Semantic Web: KR using Web technologies!

Represents a different take on standardization
- semantics: specify “how to say it”, not “what to say”

The key aspect of the Semantic Web is serendipity
- solution for use cases yet to be articulated
- “delayed semantic commitment”
Original 2001 Semantic Web vision: Retrospective view

**Good:**
- Strong, global IDs
- Graph merging
- Simple schema language *
- Self-describing data  
  - embedded or referenced ontology

**Not so good:**
- Services & publishing
  - SPARQL is pricey to run; no working business model
- SPARQL (success and failure) *
- Upper layers of the “layer cake”  
  - trust, proofs never happened
- No usable composite datatypes *
Original 2001 Semantic Web vision: Retrospective view

**Good:**

- Strong, global IDs
- Graph merging
- Simple schema language *
- Self-describing data
  - embedded or referenced ontology

**SHARING!**
Also, let’s not forget…

The word “semantics” has become much more commonplace
And yet, there seems to be confusion about that it means…
Also, let’s not forget…

The word “semantics” has become much more commonplace
And yet, there seems to be confusion about what it means…

**Semantics:**
defines how data “behaves” and how software can interpret data
Simple schema language? Really?

What about OWL?

- experience shows OWL is difficult for users and poorly understood
- OWA was the right choice, but we also need CWA: enter SHACL

End result: we really have 3 schema languages, and lots of confusion
Why is SPARQL both a success and failure?

SPARQL 1.0 was an utter failure
• I told the WG that they should have support for paths. Their response: “paths are not a use case for graphs” (really).

SPARQL 1.1 is a lot better
• some support for paths
• federated queries
• update

But:
• no path discovery
• developers want nothing to do with SPARQL
Composite datatypes

In RDF, composite datatypes are constructed using the graph structure. Instead, we should have provided abstractions and interfaces.

I think we messed this up…
Code example:

Python code for modifying the items of a (potentially existing) RDF container

```python
def setContainerItems(graph, node, predicate, values, newtype=RDF.Seq):
    if values:
        statements = getContainerStatements(graph, node, predicate)
        if statements:
            container = statements[0][0]
            for statement in statements:
                graph.remove(statement)
        else:
            container = BNode()
            graph.add((node, predicate, container))
            graph.add((container, RDF.type, newtype))
        i = 1
        for value in values:
            graph.add((container, URIRef(makeContainerItemPredicate(i)), value))
            i += 1
    else:
        container = getValue(graph, node, predicate)
        if container:
            graph.remove((node, predicate, container))
            graph.remove((container, None, None))

def getContainerStatements(graph, node, predicate):
    containers = list(graph.objects(node, predicate))
    n = len(containers)
    if n == 1:
        return sorted([statement for statement in graph.triples((containers[0], None, None))
                        if isContainerItemPredicate(statement[1])],
                       key=lambda tr: tr[1])
    elif n == 0:
        return None
    else:
        raise ValueError("Expected only one value for ")
def setContainerItems(graph, node, predicate, values, newtype=RDF.Seq):
    if values:
        statements = getContainerStatements(graph, node, predicate)
        if statements:
            container = statements[0][0]
            for statement in statements:
                graph.remove(statement)
        else:
            container = BNode()
            graph.add((node, predicate, container))
            graph.add((container, RDF.type, newtype))
            i = 1
            for value in values:
                graph.add((container, URIRef(makeContainerItemPredicate(i)), value))
                i += 1
    else:
        container = getValue(graph, node, predicate)
        if container:
            graph.remove((node, predicate, container))
            graph.remove((container, None, None))

def getContainerStatements(graph, node, predicate):
    containers = list(graph.objects(node, predicate))
    n = len(containers)
    if n == 1:
        return sorted([statement for statement in graph.triples((containers[0], None, None))
                        if isContainerItemPredicate(statement[1])],
                       key=lambda tr: tr[1])
    elif n == 0:
        return None
    else:
        raise ValueError("Expected only one value for {0}".format(predicate))

(and this is not all of it...)
Modern Knowledge Graphs

The “new” Semantic Web
The “new” Semantic Web

Life sciences were big early adopters of the original Semantic Web.

Today we see adoption in all kinds of industries:
  • graph databases have entered the mainstream
  • new technologies: Labeled Property Graphs

Now, many new popular use cases:
  • data integration
  • fraud detection, identity resolution, “Customer 360”, etc.

Integration of non-symbolic AI techniques:
  • a broader interpretation of “reasoning”
<table>
<thead>
<tr>
<th></th>
<th>Old Semantic Web:</th>
<th>New Semantic Web:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scope:</strong></td>
<td>WWW</td>
<td>Enterprise</td>
</tr>
<tr>
<td><strong>Actors:</strong></td>
<td>Humans &amp; Agents</td>
<td>Humans &amp; <strong>Cloud Services</strong></td>
</tr>
<tr>
<td><strong>Sources:</strong></td>
<td>WWW (?)</td>
<td>Enterprise data (relational databases, etc.)</td>
</tr>
<tr>
<td><strong>AI:</strong></td>
<td>Symbolic</td>
<td>Symbolic &amp; <strong>non-symbolic</strong></td>
</tr>
</tbody>
</table>
Modern knowledge graphs

Typically organizational context
  • “enterprise knowledge graphs”

Often introduced to clean the mess that exists with data
  • data integration, “democratization of data”
  • organizational upper ontologies

Sometimes these are closed systems
  • (I think this is usually a mistake…)
Labeled Property Graphs

Are closer to the “developer mindset”

Have better programming abstractions and interfaces

Often characterize as having “edge properties”, but this is not all

Support “path finding” (nonexistent in SPARQL)

but...
Labeled Property Graphs

LPGs do not have this:

Strong, global IDs
Graph merging
Simple schema language
Self-describing data
  • embedded or referenced ontology

Note that this is exactly what is so good about the Semantic Web technologies and RDF
Reification?

Note that I did not include RDF reification in what did not work

Reification is an important part of RDF

- the 1st working draft of the original RDF specification (1997-10-02) considered reification absolutely central to RDF
- reification is also widely misunderstood

“Reification” is a scary word for “edge properties”

W3C RDF-star WG (now ongoing) will “fix” reification
A graph is a graph is a graph...?

For knowledge graphs, you typically need what the Semantic Web technologies offer.

Other graph applications often treat the graph as a very large, potentially complex data structure.
A graph is a graph is a graph…?

For knowledge graphs, you typically need what the Semantic Web technologies offer. Other graph applications often treat the graph as a very large, potentially complex data structure.

Graph as a **logical representation** vs. graph as a **data structure**

- **RDF & OWL**
- **LPGs**

“The Rift”
A graph is a graph is a graph...?

Neptune currently supports both RDF and LPGs

- either or, not simultaneously
- customers have to choose, and this often leads to confusion

The Neptune team is working to mitigate the rift: Project OneGraph

The solution is not easy

- RDF-star is a significant step in the right direction

See this for more info: semantic-web-journal.net/system/files/swj3273.pdf
OneGraph

1G: a metamodel that unifies RDF, RDF-star, and LPGs

Each of the existing graph metamodels is a “lower-dimensional projection” of 1G data

Consequently, roundtrips:
- RDF $\rightarrow$ 1G $\rightarrow$ RDF: lossless, but
- 1G $\rightarrow$ RDF $\rightarrow$ 1G: not necessarily lossless
- etc.

The main (practical) challenge is that RDF and LPGs are used differently
OneGraph

Big goal: “graph interoperability” (i.e., no more confusion)

There will be benefits “on both sides”:

• use all of the good features of RDF (and SPARQL) with LPGs, without having to reinvent them
• no more complaints that RDF does not have “edge properties”
• mitigate SPARQL’s lack of path discovery
• Gremlin queries over RDF!
Summary

2001 Semantic Web vision: some things worked, others not so much

Modern knowledge graphs: the new Semantic Web

New technologies, much confusion

OneGraph: unify the graph landscape
Thank you!

Contact:

- ora@amazon.com
- Mastodon: @ora@w3c.social
- Twitter: @oralassila