From: **thomas lörtsch** tl@rat.io
Subject: Position paper for the W3C Workshop on Graph Data, 4-6 March 2019, Berlin
Date: 11 January 2019 at 20:29
To: group-data-ws-pc@w3.org

Hi,

my position paper is a bit longer than you allow. You can however read only the paragraphs titled "tl;dr" at the beginning of each section and skip the rest. Then I'm well within the limits :)

Regards,
Thomas Lörtsch

Position paper for the W3C Workshop on Graph Data, 4-6 March 2019, Berlin

tl;dr
RDF meta modeling requires succinct syntax and sound semantics for both reification of single statements and grouping of statements in graphs.
Some syntactic sugar for statement identifiers like in RDF/XML could get rid of the verbose RDF standard reification quad. A statement identifier combined with a graph would denote that statement occurrence, providing sound and intuitive semantics (even RDF 1.1 datasets provide enough semantics as they unequivocally identify a graph). This would provide the base for semantically sound Property Graph style modeling.
Graphs are a means to group statements and attribute them in certain ways. A mechanism to explicate if a graph name labels or actually names a graph would be sufficient to define the semantics of named graphs and make them not only unequivocally identifiable but also attributable in semantically sound and interoperable ways. This would enable coarse as well as fine grained graph attribution in a wide array of meta modeling scenarios like provenance tracking, access control, documenting viewpoints, managing complex object structures etc (and any combination thereof).
All that's needed to make all this possible are a few in-between additions to RDF: some vocabulary, some semantics, some syntactic sugar - no changes to RDF 1.x, just a semantic extension as provisioned by the RDF 1.x Semantics. They could be published in a W3C Note, implemented with standard RDF triples as a baseline and accompanied by mappings to possible quad and quint based optimizations.

The architecture of the Semantic Web was guided by the idea that only a very simple, even simplistic design could achieve broad adoption but that such a simple design could still go a long way in providing useful and exciting services. Uptake however has been underwhelming ever since and the simplistic design probably is one of the reasons for that. It seems that RDF misses the 80/20 rule of thumb according to which most tasks are relatively easy to accomplish while some rarer cases are still achievable, albeit with more effort.
It was always clear that a unified global information space amenable to shared reasoning was an ideal impossible to achieve in practice, on the open web. Still RDF provides only limited support to establish more controlled subspaces and well defined connections between those and to the globally shared Semantic Web, namely in the form of RDF Standard Reification as defined in RDF 1.0 and, since RDF 1.1, as Named Graphs. RDF Standard Reification lacks sufficient formal semantics and is syntactically verbose while Named Graphs have a more pleasant syntax but no semantics at all.
The result is that any Semantic Web application of reasonable size has to either resort to non standard means or give up on the promises of sound semantics and automated reasoning if it wants to leave the confines of an isolated data silo and savely interact on the open, globally shared Semantic Web.
That lack of sound meta modeling facilities also hinders applications that require recording of meta data, be it for administrative tasks like provenance documentation or to support controlled integration of differing viewpoints, conflicting statements and other variations that are unavoidable on an open system.
The design of the Semantic Web often relies on concepts like "social meaning" to manage (and get around) the endless variations of real life semantics that would be impossible to implement and express in a globally shared information system. Instead it tries to cater for the most sensible defaults. It does however provide very little means to explicate subtle differences when the need arises as manifested by the infamous Http-range 14 discussion on how to discriminate denotation and indication in URI references. RDF can't be blamed for the quagmires that formal semantics can get stuck in so quickly but it should provide more means to explicate them. Otherwise one is in constant danger to fall of the cliff into some semantic wormhole without prior warning when unassumingly hitting to 81% mark.

The Semantic Web needs to step outside of its idealized comfort zone of one shared global information space with well defimed semantics. It has not only to accept but to embrace partitioning, and provide means to express and connect those partitions in well defined ways. I'll argue below that a few and intermediate additions to RDF syntax and semantics could bring out some much needed semantically sound expressivity and should make it much easier to manage and control data flows by clarifying boundaries and more explicitly expressing relations between domains and applications. It should also enable more intuitive modeling styles and reduce the semantic overload of helpers like blank nodes. The net result should be much more expressivity and clarity at the cost of a little more machinery.

Reification

tl;dr
RDF statement reification can be streamlined by providing a statement identifier that doesn't depend on the RDF standard

RDF statement reification can be streamlined by providing a statement identifier that doesn't depend on the RDF standard reification quad but is calculated on demand by an algorithm - e.g. a MD5 hash.
Such a statement identifier would refer to the abstract triple type. Combined with a graph name however it would denote a concrete statement occurrence, e.g.:

```
:someStatementId@someGraphUri
```

Reification was always a neglected topic on the Semantic Web but it is the core of meta modeling and a stepping stone to sound meta modeling on the graph level. Many popular use cases of RDF concentrate on the grouping of statements in graphs but some critically depend on statement level meta data. Also some modeling techniques like attributed graphs that aim to add structure to unwieldy graphs benefit heavily from reification.
The RDF 1.0 Primer precisely describes the problem: RDF reification is meant to describe not an abstract triple type but specific occurrences of a triple to e.g. record provenance information on it. The RDF reification machinery however only provides means to describe the triple type but not the specific situation or location in which the triple occurrence takes place. A graph would be the natural and intuitive way to describe such a place. Some syntactic sugar could replace the verbose RDF standard reification quad, like the otherwise rightfully deprecated RDF/XML did. A standard mechanism like some hashing function to generate identification IDs should be defined.
In recent years some node and property centric approaches to reification like Singleton Properties and Fluents have been proposed. It seems however that while all these proposals are technically equivalent and can be converted to each other automatically, statement level reification still is the most intuitive solution (e.g. it doesn't break expectations regarding node and property stability), has good querying and entailment properties and provides the best base for some more advanced modeling techniques.

Identification

tl;dr
If a URI is meant to denote or indicate a resource depends on its usage in some specific situation. It is a role a URI plays in a certain statement.
A fragment identifier on a statement identifier could succinctly adress such nodes and clarify the roles they play if needed, e.g.:

```
:someStatementId#subject rdf2:identificationSemantics rdf2:Denotation .
```

RDF has a problem with semantically sound identification of resources which stems from ambiguities in natural language. Humans are used to infer from context if a reference is meant to refer to a resource itself or to what the resource describes. This is so common in everyday communication that we are not even aware of the process. On the Semantic Web the necessity to differentiate between denotation and indication however feels tedious and unreasonably burdensome. The solution has been to go with the concept of identification as subsuming both denotation and indication and rely on out-of-band means to express the exact intention if the need arises (which luckily is not too often). Denotation and indication can be understood as roles that a resource can play in some statement, so concise statement reification would provide a means to express per node per statement if the node is meant to denote or to indicate the subject/object of that specific statement. This is a small tweak but it concerns a central part in the process of encoding meaning in machine processabe ways.

Named Graphs

tl;dr
Named Graphs in RDF could be given sound semantics by providing properties and classes that allow to express if a graph name merely labels a graph or actually refers to it.
That would pave the ground for very fine grained graph structures that could cater for lots of aspects like e.g. provenance, security, versioning, viewpoints, plausibility, structural properties etc.

The question of what a resource actually refers to – itself or something it describes - resurfaced in an unexpected but dogged way when the RDF 1.1 WG tried to standardize Named Graphs. It turned out that existing systems were based on incompatible semantic assumptions about what the graph name refers to: the graph itself (naming semantics) or some other nodes that somehow describe or attribute the graph (labeling semantics). Both semantics have merits and are widely adopted but are incompatible to each other.
The WG pondered the idea of adding a class Graph to RDF and asserting denotational graph naming semantics through statements of the following form:

```
:someUriNamingTheGraph rdf:type rdf2:Graph .
```

I couldn't find any mention of how to express labeling semantics in the RDF 1.1 WG archives but I'd suggest that graph labeling is equivalent to implicitly adding a blank node that names the graph and has the graph label as a property :

```
[] rdf:type rdf2:Graph ;
  rdfs:label :someUriLabelingTheGraph .
```

The RDF 1.1 WG in the end didn't standardize any such vocabulary to express and tell apart the intended semantics of graph naming. However disambiguation by "social meaning" is not a viable option when designing formal semantics and so Named Graphs where standardized as a purely syntactic vehicles without model theoretic semantics, following the infamous route taken by RDF standard reification. Like with reification RDF once again stopped just short of specifying a semantically sound meta modelling mechanism. This time it would even have been syntactically pleasing.
However some essential semantics are in place: while graph names only identify graphs they can indeed denote statements in graphs. It's not possible to add attributes to a graph without semantically sound naming but it is possible to add attributes to a statement contained in a graph in a semantically sound way. This would work with both RDF standard reification quads and with functionally derived identifiers as proposed above.
A further extension of RDF vocabulary could be defined to express default graph naming semantics per dataset.

Semantically sound graph denotation is only the base on which very expressive representations can be build. I will not go into further detail about possible graph attributions, contextualizations, inheritance mechanisms etc. but want to hint at the possibility that very fine grained and elaborate graph attribution structures might be required to cater for all the different aspects under which graphs might be attributed. This does not mean that a "context logic" would be required to handle such diversity but it may involve a lot of filtering in queries.

Blank Nodes

tl;dr
Blank nodes in RDF are overloaded with functionality. Concise attributions would help to disambiguate and clarify their intended semantics and function by e.g. marking a blank node as head of some specific structure like a nested list or adding a hint on how to best retrieve the complete subgraph they belong to, maybe through a Concise Bound Description algorithm.

The graph labeling formalization above is just one example of all the things blank nodes can be used for and that is probably the main problem with them: blank nodes serve too many purposes and it's often hard to see what exactly they represent. That leads to unreasonable or plain wrong expectations and consequently frustrating and hard to debug unintended behaviours. Concise statement reification could help to attribute blank nodes with further hints about their intended semantics, recommended retrieval strategies, structural characteristics etc - thereby taking some cognitive load from this very overloaded instrument.
The underlying problem is of course that the structural backbone of RDF, graphs, carry very little intrinsic meaning. This makes them so flexible and well suited for the core task of the Semantic Web: connecting disparate resources in unforseen ways. OTOH this malleability also makes everyday use cases that rely on the inherent semantics of nested lists or tables tedious and verbose. Blank nodes do fill all the syntactic holes but fail at communicating the intended semantics.

Modeling

tl;dr
Concise reification would elegantly support Property Graph style modeling and many common n-ary relation scenarios.
Semantically sound graph naming would enable interoperable graph attribution and entailments over graph properties.
Being able to succinctly encode structural information in statement and graph attributes would help to make up for the lack of structure (and structural syntactic sugar like lists and tables) in RDF.

The almost exclusive reliance on pure graphs as modeling vehicle hasn't served the Semantic Web too well. Other communities like Property Graphs and WikiData have developed a modeling style that attributes primary relations with further information that is secondary to the main relation. It seems that a lot of people find this modeling style very intuitive. Many n-ary relations - a notoriously difficult topic with RDF - also fit into this scheme quite well as the participants in a n-ary relation are often not equally important but are attributes to one primary relations.
Concise and semantically sound statement reification is essential to support such modeling styles. Being able to refer to a statement through an iD would eliminate the need to define the infamous RDF reification quad.
Named graphs can greatly facilitate the definition of graph boundaries where algorithms like CBD and other graph bounding techniques are too cumbersome or inflexible. This is a prerequisite for all kinds of graph attribution and contextualization.

How To Proceed

tl;dr
- Define a semantic extension to RDF that covers semantically sound and concise statement reification and graph denotation.
- Define a baseline triple implementation and outline different ways of interoperable optimizations as quads or quints that focus on statement reification or statement grouping (named or labeled or both) or some combination thereof.
- Show in a few examples how this facilitates common idioms like Property Graph style modeling, n-ary relations, and easier structuring and subgraph bounding.
- Publish that as a W3C Note "and they will come".

I've tried to show how some of the long standing and nagging issues of the Semantic Web are related and how a few small additions in between can make a big difference alltogether.

None of the proposals above require changes to RDF 1.x or prior standardization as some garnd new RDF 2.0. A central part - sound semantics for RDF standard reification - doesn't even need new vocabulary. Further additions like succinct statement identifiers per hashing function as well as some vocabulary to define graph naming semantics could be specified as what RDF calls a semantic extension.
Ideally that would provide a triple based implementation of the proposal as a baseline for compatible implementations that focus on further optimizations like quad stores with multiple naming semantics or quint stores with graph naming and statement identification columns. The work on Named Graphs in the RDF 1.1 WG has shown that use cases vary quite a bit. Although it is possible through some syntactic trickery to bring graph naming, graph labeling and statement identification under one semantically sound hood in a shared 4th column it might be more fruitful for now to let different optimizations co-exist and just ensure interoperability. From there further standardization efforts towards e.g. RDF 2.0 can evolve.

Will such a proposal gain traction? Do we need to go through another painful round of RDF standardization first? As n-ary relations show, a W3C Note can go a long way: many people will gladly accept guidance from a W3C stamped recipe - and if its

relations show, a W3C Note can go a long way. Many people will gladly accept guidance from a W3C stamped recipe - and if its only for "You never get fired for hiring IBM" reasons. And there's a huge need for such guidance as people are hindered and frustrated by the lack of semantically sound and well established meta modeling techniques on the Semantic Web.