Well Behaved RDF:
A Straw-Man Proposal for Taming Blank Nodes

David Booth, Ph.D.
david@dbooth.org
Draft 2012-12-13 - Comments invited

Latest version of this document:
http://dbooth.org/2013/well-behaved-rdf/Booth-well-behaved-rdf.pdf (PDF)

Abstract. The RDF language (Resource Description Framework) allows nodes in an RDF graph to be unlabeled – "blank nodes". While blank nodes and certain other features are convenient for RDF authors, their unrestricted use causes complications to RDF consumers, such as when attempting to compare RDF graphs, which in the general case is as difficult as the graph isomorphism problem. This paper proposes a straw-man profile of the RDF language – *Well Behaved RDF* – that constrains the use of certain features (notably blank nodes) to facilitate simpler RDF processing.

Keywords: RDF, blank nodes, bnodes, canonicalization, graph isomorphism

1 Introduction

Resource Description Framework (RDF)[1] is a web-friendly, syntax independent, knowledge representation language that is the foundation for Linked Data[2] and the Semantic Web. In RDF, information is represented as a graph of labeled arcs and nodes. Nodes may either be URIs[3] (or more recently IRIs[4]), literals or *blank nodes* (a/k/a *bnodes*) – unlabeled nodes that represent existential variables.

The RDF standard is quite lax in the use of some of its features, and this can cause difficulties to applications that process RDF, even if those features are convenient for some uses. Most notable: blank nodes.

While blank nodes are convenient for RDF authors, they can be troublesome for RDF consumers.[5][6] Below are two of the main problems. There are others as well, such as entailment issues,[7] that will not be discussed here.

1.1 Blank node labels are unstable across serializations.

When an RDF graph is serialized, syntactic elements known as *blank node labels*[8] or *blank node identifiers*[9] are generated to identify the blank nodes within

that particular serialization of the graph. However, there is no guarantee that the same label will be generated for a given blank node each time the graph is serialized. For example, a blank node may be assigned the label `_:b24` in one Turtle[10] serialization but the exact same blank node may be assigned the label `_:x97` in a different Turtle serialization. There is no standard, reliable way to reference a blank node across graph serializations.

One practical difficulty this causes is that it is impossible to (directly) make RDF statements about a blank node from outside of the graph in which that blank node appears. For example, although a particular blank node may have been identified by the label `_:b24` when a Turtle-serialized RDF graph was loaded into a SPARQL[11] server, one cannot later merge another Turtle-serialized RDF graph that attempts to add additional statements about that blank node and expect that `_:b24` will refer to that same blank node.

Another common problem occurs when RDF data is queried, and the results of one query are to be used in forming a follow-up query, such as in a drill-down scenario. If the first query returns a blank node, its label cannot reliably be used to refer to the same blank node in a subsequent query. One may be tempted to criticize the SPARQL specification for this problem, but the problem traces back to RDF itself, in the fact that blank nodes do not have stable names.

It is worth pointing out that these difficulties were foreseen (at least in principle) by the authors of the W3C Architecture of the World Wide Web (AWWW),[12] as the use of blank nodes clearly violates the web architectural good practice that anything of importance should be given a URI. As the AWWW states: “A resource should have an associated URI if another party might reasonably want to . . . make or refute assertions about it . . . .”

1.2 Difficulty canonicalizing a graph.

A second, related problem is that unrestricted RDF graphs cannot always be readily serialized into a canonical form, because of blank nodes. This was discussed by Carroll[13] in the context of generating digital signatures for RDF graphs. He showed that “RDF canonicalization is graph isomorphism (GI) complete”, which in practical terms means that there is no known algorithm that can efficiently canonicalize an arbitrary RDF graph.

The most important practical consequence of this problem is that RDF graphs cannot be easily compared, either to determine equality or to compute their differences. This is of course a problem in the context of creating and verifying digital signatures for RDF graphs, as Carroll has discussed. But it is also a major problem when attempting to do regression testing using RDF data. Every serious software development effort uses regression testing to ensure that new errors have not been introduced when software components are upgraded. Other media types can be readily compared byte-for-byte using simple, standard tools such as *cmp*.[14] But

when we use RDF to represent our data, regression testing suddenly becomes far more difficult.

Many of us in the Semantic Web community have been touting the merits of RDF over more conventional information representation languages such as XML, yet the seemingly simple task of comparing two documents is suddenly much harder than it was for other languages. Must we live with these problems when using RDF?

2 Blank node debates

Blank nodes have been discussed and debated a lot in the RDF community. For example, Miličić[15] notes: "Due to the absence of a name (URI), manipulating data containing blank nodes is much harder – they make otherwise trivial operations far more complex." Hawke [16] points out: "In general, blank nodes are a convenience for the content provider and a burden on the content consumer." Cyganiak[17] states: "The higher the percentage of blank nodes in a dataset, the less useful it is." Heath and Bizer[18] even recommend avoiding blank nodes altogether when publishing Linked Data: ". . . all resources in a data set should be named using URI references."

Mallea, Arenas, Hogan, and Polleres[19] provide an excellent examination of blank nodes, the problems they create and the ways they are used in practice. They note that the way people use blank nodes often differs from their formal semantics (as existential variables), and outline some options for either retaining or changing the semantics of blank nodes. They also mention in passing that "One other possibility we considered was not allowing blank nodes to ever be explicitly labeled", and it is this option that this paper pursues.

3 The Convenience of Blank Nodes

Despite these well-recognized problems, blank nodes can also be an undeniable convenience; hence the hesitation to give them up. Who would want to mint a URI instead of using a blank node in the following common OWL[20] idiom, from an example in [21]?

```
:Person
    a       owl:Class ;
    rdfs:subClassOf
            [ a       owl:Restriction ;
              owl:allValuesFrom :Diagnosis_Relation ;
              owl:onProperty :has_diagnosis
            ] .
```

And when writing a list (a/k/a an RDF *Collection*), who would want to mint URIs for all of the blank nodes that are implicitly created – one for each list item – such as in this Turtle [22] example?

```
@prefix : <http://example.org/stuff/1.0/> .
:a :b ( "apple" "banana" )
```

It would be nice to retain the convenience of blank nodes – especially for things like lists and n-ary relations[23] – without causing problems with RDF graph comparison or unstable labels.

4 Skolemization

One option for eliminating blank nodes from an RDF graph might be to convert them into URIs via skolemization.[24][25] Blank node skolemization does (subtly) change the formal semantics of an RDF graph, though the change may be acceptable to the graph's author. But more importantly, although it may help in some cases, it would not necessarily solve the problem.

If the skolemization is done before the RDF author publishes the graph, then it is roughly equivalent to the author minting explicit URIs instead of using blank nodes, except that if the author subsequently publishes another version of the graph, and the skokemization process generates different URIs for the blank nodes, then the result is unstable URIs instead of unstable blank node labels. On the other hand, if skolemization is performed by other parties after the graph is published, then there is no guarantee that those other parties will generate the same skolem URIs for the blank nodes, thus leading to a similar problem.

Finally, if a deterministic skolemization algorithm were standardized, such that different parties skolemizing the same graph could be guaranteed to generate the same skolem URIs for the blank nodes, then the skolemization algorithm itself would essentially be solving the RDF graph canonicalization problem. In essence, this would be shifting the potentially difficult canonicalization burden to the skolemization process instead of avoiding it.

5 Taming blank nodes

Carroll[26] pointed out that some RDF graphs with blank nodes are easy to canonicalize, while others are hard. In the general case canonicalization is as difficult as solving the graph isomorphism problem. But graphs whose blank nodes form a tree structure – i.e. graphs containing no blank node cycles – can be canonicalized in polynomial time.

Mallea, Arenas, Hogan, and Polleres[27] evaluated nearly 4 million RDF graphs from the web and showed that over 98% of the graphs had no blank node cycles, i.e.,

the blank nodes formed tree structures. Thus, canonicalization should be easy in the vast majority of cases!

However, the dilemma is that although very few RDF graphs use blank nodes in a way that causes canonicalization to be difficult, an RDF processor that wishes to be robust must account for the *possibility* that it might receive such a graph, and thus it must be programmed to deal with it. In other words, *the many applications that process RDF pay for the sins of the few RDF graphs that use blank nodes in problematic ways.*

The W3C RDF working group[28] is well aware of these problems with blank nodes, but it is doubtful that the group's charter[29] would permit a change to the RDF standard to restrict any of the currently permitted uses of blank nodes. Furthermore, some RDF authors may strongly prefer the unrestricted use of blank nodes.

For these reasons, this paper proposes a voluntary profile of RDF – Well Behaved RDF – that limits the use of blank nodes, and potentially other features, in order to simplify RDF processing. This would allow software developers who only wish to support Well Behaved RDF to create their software more easily, while enabling them to easily tell their users the input limitations and expectations of their software. At the same time, those who wish to support the full range of RDF would be free to do so as well.

6 A Straw-Man Definition of Well Behaved RDF

The purpose of Well Behaved RDF is to enable simpler tools and applications to process RDF.

*Definition: A **Well Behaved RDF graph** is an RDF graph that conforms to the following restrictions.*

1. *It can be serialized as Turtle without the use of explicit blank node identifiers. I.e., only blank nodes that are implicitly created by the bracket "[. . .]" or list "(. . .)" notation are permitted.*
2. *It uses no deprecated features of RDF.*

The first restriction guarantees that the blank nodes in a Well Behaved RDF graph will not contain cycles, because it is not possible to create a blank node cycle using only implicit blank nodes. At the same time, this restriction can eliminate the problem of unstable blank node labels in RDF serialization, because the graph can be serialized to a syntax such as Turtle without using blank node labels. No labels, no unstable labels. It would also reduce the need for skolemization.

Note that Turtle is mentioned only as a convenient means of specifying this restriction on the use of blank nodes. A Well Behaved RDF graph is not required to be serialized as Turtle. This blank node restriction could be phrased differently (in

terms of avoiding blank node cycles), but it would probably be more complicated to explain that way.

The second restriction is included in case the RDF working group decides to deprecate certain features.

7 Best practices and other questions

This straw-man proposal could be augmented with a number of other restrictions that would not hamper the vast majority of RDF applications, and could aid their developers. In some sense it becomes a question of where to draw the line between defining an RDF profile, and listing RDF best practices. Here are some additional questions that could be considered; undoubtedly there are many more.

- Should the use of rdf:first and rdf:rest be limited to well-formed, rdf:nil-terminated lists, i.e., those that can be serialized as Turtle lists "(. . .)"?
- Should restrictions be placed on the use of RDF Containers?
- What, if anything, should be said about non-lean[30] RDF graphs?
- Should anything be said about RDF reification?
- Should typed literals be required to be well formed per their type definitions?
- Should literals be required to be in canonical form, such as "1"^^xsd:integer instead of "+0001"^^xsd:integer?
- Should the use of XSD[31] numeric datatypes be limited to xsd:integer and xsd:decimal (avoiding other derived types)?

There is plenty of room for ideas and debate on the details.

8 Conclusions

This paper has presented a straw-man proposal for an RDF profile that would restrict the use of certain RDF features (notably blank nodes) to non-problematic RDF idioms. This would permit RDF processing chains to be significantly simplified for the many applications that could function perfectly well with those restrictions.

Acknowledgments. Thanks to Any Seaborne for some of the datatype suggestions.

References

1. Klyne, G., Carroll, J. (editors): Resource Description Framework (RDF): Concepts and Abstract Syntax. (2004) W3C. http://www.w3.org/TR/rdf-concepts/ Retrieved 2012-12-12.

2. Berners-Lee, T.: Linked Data. (2006 & 2009) W3C. http://www.w3.org/DesignIssues/LinkedData.html Retrieved 2012-12-12.

3. Berners-Lee, T., Fielding, R., Masinter, L.: Uniform Resource Identifier (URI): Generic Syntax. RFC 3986. (2005) IETF. http://www.ietf.org/rfc/rfc3986.txt Retrieved 2012-12-12.

4. Duerst, M., Suignard, M.: Internationalized Resource Identifiers (IRIs). RFC 1987. (2005) IETF. http://www.ietf.org/rfc/rfc3987.txt Retrieved 2012-12-12.

5. Hawke, S.: A blank node issue. (2011) W3C. http://lists.w3.org/Archives/Public/semantic-web/2011Mar/0068.html Retrieved 2012-12-12.

6. Cyganiak, R.: Blank nodes considered harmful. (2011) cygri's notes on web data. http://richard.cyganiak.de/blog/2011/03/blank-nodes-considered-harmful/ Retrieved 2012-12-12.

7. Mallea, A., Marcelo, A., Hogan, A., Polleres, A.: On Blank Nodes. (2011) ISWC, Part I, LNCS 7031, pp. 421-437. Springer-Verlag. http://web.ing.puc.cl/~marenas/publications/iswc11.pdf Retrieved 2012-12-12.

8. Beckett, D., Berners-Lee, T., Prud-hommeaux, E., Carothers, G.: Turtle, Terse RDF Triple Language. W3C Working Draft 10 July 2012. http://www.w3.org/TR/turtle/#BNodes Retrieved 2012-12-12.

9. Beckett, D. (editor): RDF/XML Syntax Specification (Revised). W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-syntax-grammar/#section-Syntax-blank-nodes Retrieved 2012-12-12.

10. Beckett, D., Berners-Lee, T., Prud-hommeaux, E., Carothers, G.: Turtle, Terse RDF Triple Language. W3C Working Draft 10 July 2012. http://www.w3.org/TR/turtle/#BNodes Retrieved 2012-12-12.

11. Harris, S., Seaborne, A. (editors): SPARQL 1.1 Query Language. W3C Proposed Recommendation 08 November 2012. http://www.w3.org/TR/sparql11-query/ Retrieved 2012-12-12.

12. Jacobs, I., Walsh, N. (editors): Architecture of the World Wide Web, Volume One. W3C Recommendation 15 December 2004. http://www.w3.org/TR/webarch/#uri-benefits Retrieved 2012-12-12.

13. Carroll, J.: Signing RDF Graphs. (2003) HP Laboratories Bristol, HPL-2003-142. http://www.hpl.hp.com/techreports/2003/HPL-2003-142.pdf Retrieved 2012-12-12.

14. Anonymous: cmp (Unix). (2012) Wikipedia.org. http://en.wikipedia.org/wiki/Cmp_%28Unix%29 Retrieved 2012-12-12.

15. Miličić, V.: Problems of the RDF model: Blank Nodes. (2011) Bew Citnames. http://milicicvuk.com/blog/2011/07/14/problems-of-the-rdf-model-blank-nodes/ Retrieved

2012-12-12.

16. Hawke, S.: A blank node issue. (2011) W3C. http://lists.w3.org/Archives/Public/semantic-web/2011Mar/0068.html Retrieved 2012-12-12.

17. Cyganiak, R.: Blank nodes considered harmful. (2011) cygri's notes on web data. http://richard.cyganiak.de/blog/2011/03/blank-nodes-considered-harmful/ Retrieved 2012-12-12.

18. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. (2011) Morgan & Claypool. http://linkeddatabook.com/editions/1.0/ Retrieved 2012-12-12.

19. Mallea, A., Marcelo, A., Hogan, A., Polleres, A.: On Blank Nodes. (2011) ISWC, Part I, LNCS 7031, pp. 421-437. Springer-Verlag. http://web.ing.puc.cl/~marenas/publications/iswc11.pdf Retrieved 2012-12-12.

20. McGuinness, D., van Harmelen, F. (editors): OWL Web Ontology Language, Overview. W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-features/ Retrieved 2012-12-12.

21. Noy, N., Rector, A. (editors): Defining N-ary Relations on the Semantic Web. W3C Working Group Note 12 April 2006. http://www.w3.org/TR/swbp-n-aryRelations/ Retrieved 2012-12-12.

22. Beckett, D., Berners-Lee, T., Prud-hommeaux, E., Carothers, G.: Turtle, Terse RDF Triple Language. W3C Working Draft 10 July 2012. http://www.w3.org/TR/turtle/#sec-examples Retrieved 2012-12-12.

23. Noy, N., Rector, A. (editors): Defining N-ary Relations on the Semantic Web. W3C Working Group Note 12 April 2006. http://www.w3.org/TR/swbp-n-aryRelations/ Retrieved 2012-12-12.

24. Anonymous: Skolem normal form. (2012) Wikipedia.org. http://en.wikipedia.org/wiki/Skolem_normal_form Retrieved 2012-12-12.

25. Anonymous: Skolemization. (2012) W3C RDF Working Group Wiki. http://www.w3.org/2011/rdf-wg/wiki/Skolemization Retrieved 2012-12-12.

26. Carroll, J.: Signing RDF Graphs. (2003) HP Laboratories Bristol, HPL-2003-142. http://www.hpl.hp.com/techreports/2003/HPL-2003-142.pdf Retrieved 2012-12-12.

27. Mallea, A., Marcelo, A., Hogan, A., Polleres, A.: On Blank Nodes. (2011) ISWC, Part I, LNCS 7031, pp. 421-437. Springer-Verlag. http://web.ing.puc.cl/~marenas/publications/iswc11.pdf Retrieved 2012-12-12.

28. Anonymous: RDF Working Group. (2012) W3C web site. http://www.w3.org/2011/rdf-wg/wiki/Main_Page Retrieved 2012-12-12.

29. Hawke, S., Herman, I., Wood, D., Schreiber, G.: RDF Working Group Charter. (2011) W3C. http://www.w3.org/2010/09/rdf-wg-charter Retrieved 2012-12-12.

30. Hayes, P. (editor): RDF Semantics. W3C (2004). http://www.w3.org/TR/rdf-mt/#deflean Retrieved 2012-12-12.

31. Biron, P., Malhotra, A.: XML Schema Part 2: Datatypes Second Edition. W3C Recommendation 28 October 2004. http://www.w3.org/TR/xmlschema-2/ Retrieved 2012-12-12.