Revisiting Blank Nodes in RDF

to Avoid the Semantic Mismatch with SPARQL

Marcelo Arenas

1

, Mariano Consens

2

, and Alejandro Mallea

1,3

1

Pontificia Universidad Cat´olica de Chile

2

University of Toronto

3

University of California, San Diego

1 Motivation

Jointly with the release of RDF in 1999 as recommendation of the W3C, the natural problem of querying RDF data was raised. Since then, several designs and implementations of RDF query languages have been proposed (see [9] and [7] for detailed comparisons of RDF query languages). In 2004, the RDF Data Access Working Group released a first public working draft of a query language for RDF, called SPARQL [13]. Since then, SPARQL has been rapidly adopted as the standard for querying Semantic Web data. In fact, SPARQL became a W3C Recommendation in January 2008 by http://www.bobsseo.com.

In spite of being the standard query language for RDF, the design of SPARQL was made to keep the efficiency of the language considering the current database

technology. In this direction, the current definition of the semantics of SPARQL does not consider the combined treatment of two of the distinctive features of RDF graphs, namely the semantics of blank nodes and RDFS vocabulary recommended by the W3C in the definition of RDF [10]. In fact, the semantics of SPARQL does not match in some constructions the semantics for blank nodes recommended by the W3C in [10]. To see that this is the case, consider the RDF graphs G1
and G2 shown in Figure 1. In these graphs, :b1, :b2 and :b3 are
blank nodes, which are used to represent objects that are owned by John and Peter. According to the semantics for blank nodes proposed by the W3C [10, 8], these two graphs are equivalent as they can be mapped into each other
4
. Thus,
one would expect that the answer to any SPARQL query over G1 is the same as over G2
. However, this is not the case for the following SPARQL query Q:

```
SELECT DISTINCT ?X
WHERE { ?X owns ?Y .
?X owns ?Z .
FILTER (?Y != ?Z) }
```

According to the semantics of SPARQL, the answer to Q over G1 is John since :b1 and :b2 are distinct values, while the answer to Q over G2
is the
empty set.
4

Function f defined as f (:b1) = f (:b2) = :b3 is a homomorphism from G1 into

G2, while function g defined as g(:b3) = :b1 is a homomorphism from G2 into G1.

John owns :b1 .

Peter owns :b1 .

John owns :b2 .

John owns :b3 .

Peter owns :b3 .

RDF graph G1 RDF graph G2

Fig. 1. Two equivalent RDF graphs.

The preceding example shows a mismatch between the semantics of blank

nodes in RDF and SPARQL. In RDF, two distinct blank symbols are not assumed to represent distinct objects, as explicitly mentioned in the W3C document defining the semantics of RDF: “Blank nodes are treated as simply indicating the existence of a thing, without using, or saying anything about, the name

of that thing” [10]. On the other hand, two distinct blank symbols are treated

as distinct objects in SPARQL, as shown by the fact that condition (:b1 !=

:b2) holds in this query language.

The difference in the treatment of blank nodes in RDF and SPARQL poses

an obvious question, what is the right semantics for these nodes? In this position

paper, we argue in favor of a unique semantics for blank nodes. In particular,

this issue is considered from different perspectives in Section 2, and then our

position about this matter is given in Section 3.

2 What is the right semantics for blank nodes?

2.1 A practical perspective: Blank nodes in real-life RDF graphs

Our first approach to the question of what is the right semantics for blank nodes

is to see how these nodes are being used in real-life RDF data. For this, we have

downloaded and analyzed RDF datasets freely available on the Web, searching

for triples with blank nodes and trying to understand their structure.

We have observed that, on the few occasions that blank nodes are actually used, they are intended to be pointers to more data. For example, the RDF dump of the US Census data [1] uses blank nodes to represent households, each one connected to several statistics pertaining to it. At the same time, other blank nodes represented streets, each one connected to several houses, and so on. In this context, it makes sense to use blank nodes as anonymous entities (to keep privacy). Another example of this type of use of blank nodes is given by ordered collections of concepts as recommended in SKOS Primer [2]. As shown in Figure 2, blank nodes are used in SKOS to point to the value of the current element and to the next element in an ordered collection of concepts.

In the previous examples, as well as in more than 1 terabyte of RDF data analyzed, blank nodes shared the property of having in-degree 1 and out-degree greater or equal than 1. Roughly speaking, this means that blank nodes are being used as identifiers for collections of data intended to be together, but that for some reason are better off without a dereferenceable URI. It is important

type

rest

:b3

adults

:b1

first

rest

infants

first first

nil :b0

"people by age"

orderedCollection

:b2

children

rest memberList

label

Fig. 2. An ordered collection in SKOS.

to notice that this semantics corresponds with the semantics of blank nodes in

SPARQL, as two distinct blank nodes are treated as distinct v alues in all these

RDF graphs.

2.2 A theoretical perspective: Complexity of query evaluation

To study the complexity of evaluating a query language, the notions of data

complexity and combined complexity were introduced in [14]. The data complexity of a query language is the complexity of evaluating a query as a function

of the size of the data, while the combined complexity is defined in the same

way but considering both the size of the data and the query [14]. In general, the

size of a query is considerably smaller than the size of the data, and, therefore,

the notion of data complexity is usually considered as a good way to measure

whether a query language can be used in practice. In fact, most of the query

languages used in real-life systems (such as SQL) have a low data complexity.

In the context of RDF, it has been shown that SPARQL has a low data

complexity [12]. This result was proved by using the current semantics of blank

nodes in SPARQL, that is, distinct blank symbols were considered as distinct

values. But this immediately raises the question of what would be the complexity of evaluating SPARQL if one decides not to use this semantics, but instead

sticking to the semantics of blank nodes in the normative specification of RDF

[10]. According to [10], blank nodes represent incomplete information. As such,

it would be natural to incorporate this semantics into SPARQL by considering

RDF graphs as incomplete databases [11, 4]. But unfortunately this would increase the data complexity of SPARQL. In fact, it is possible to conclude, from

the results in [3], that in this case the data complexity would be coNP-hard, even

for the fragment of SPARQL consisting of basic graph patterns and operators

SELECT, UNION and FILTER (?X != ?Y).

It should be noticed that the previous coNP lower bound depends on the size

of RDF graphs. Thus, given that real-life RDF graphs can be very large, the

previous result tells us that it would very costly to evaluate SPARQL queries if

blanks nodes are treated as proposed by the W3C in the definition of RDF [10].

2.3 The Linked Data perspective

One of the most significant scenarios for RDF usage (and likely the fastest growing in terms of data and users) is publishing Linked Data on the Web [5]. Linked

Data represents an evolution of the Web that incorporates new ways of publishing and interacting with information. Instead of relying only on information

expressed at a document-level granularity, semi-structured descriptions of Web

resources that are represented by dereferenceable URIs can be expressed using

RDF. Linked Data is a simple mechanism for sharing semi-structured descriptions of Web resources across datasets via the creation of RDF links between Web

resources. The Linking Open Data community project is promoting a Web of

Linked Open Data (LOD), and many interlinked datasets have been contributed

to what has been referred to as the LOD Cloud.

The first architectural principle for Linked Data is to Use URIs as names

for things. This clearly discourages the use of blanks. In fact, the tutorial [6] put

forward by the Linked Data community explicitly says that:

"We discourage the use of blank nodes. It is impossible to set external RDF links to a blank node, and merging data from different sources

becomes much more difficult when blank nodes are used. Therefore, all

resources of any importance should be named using URI references."

The tutorial also points out how the Linked Data scenario has influenced the

design of the commonly used ontologies (in particular, the FOAF specification

has abandoned the use of blank nodes in favour of URI references).

As a recent example, consider the Evaluation for the Entity Search Track

that is part of the Semantic Search Workshop at WWW 2010, and has the goal

of developing a benchmark, based on which semantic search systems can be

compared and analyzed in a systematic fashion. The Evaluation initiative has

modified datasets to eliminate blank nodes. The initiative provides a corpus of

datasets, which contain entity descriptions in the form of RDF representing a

sample of Web data crawled from publicly available sources. The corpus is based

on the well known Billion Triple Challenge 2009 dataset, which contains blank

nodes. The evaluation initiative required participants to encode blank nodes

according to the following rule: map each BNID (i.e., each blank node identifier)

to http://example.org/URLEncode(BNID). Since the blank node ids across this

specific dataset are unique, this simple convention allows mapping blank nodes

to obtain distinct URIs. It is important to notice that this approach corresponds

with the semantics of blank nodes in SPARQL, as two distinct blank nodes are

treated as distinct values in the previous rule.

3 Our position

In the previous section, we gave evidence that the semantics of blank nodes in

real-life RDF graphs corresponds with the semantics of blank nodes in SPARQL.

Moreover, we also gave theoretical evidence of the high cost of evaluating SPARQL

under the semantics for blank nodes in the normative specification of RDF [10].

Given this evidence, we advocate for the modification of the semantics of blank nodes in RDF to align it with SPARQL. In particular, blank nodes could be used as identifiers without a URI, and two distinct blank nodes symbols :b1 and :b2 should always be considered as different values (that is, condition (:b1 != :b2) should hold as it is the case in SPARQL). As a consequence of this modification, the implication of RDF graphs without RDF/S vocabulary would be reduced to subset testing.

The previous suggestion represents a compromise between the Linked Data position of eliminating blank node usage altogether, and the actual usage observed in practice (and embraced by ontologies such as SKOS). We observe that blanks nodes under our proposed semantics are effectively syntactic sugar, and they can be easily replaced by URIs in a systematic way (e.g., the URIs can be generated based on URLEncode(BNID) with a suitable prefix).

Acknowledgments. We thank the anonymous referees for helpful comments.

Arenas was supported by FONDECYT grant 1090565.

References

1. The 2000 u.s. census: 1 billion rdf triples.

http://www.rdfabout.com/demo/census/.

2. Skos: Simple knowledge organization system primer.

http://www.w3.org/TR/skos-primer/.

3. S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In PODS, pages 254–263, 1998.

4. Serge Abiteboul, Richard Hull, and Victor Vianu. Foundations of Databases.

Addison-Wesley, 1995.

5. Tim Berners-Lee. Linked data - design issues.

http://www.w3.org/DesignIssues/LinkedData.html, 2006.

6. C. Bizer, R. Cyganiak, and T. Heath. How to publish linked data on the web. http://www4.wiwiss.fuberlin.de/bizer/pub/LinkedDataTutorial/, 2007.

7. T. Furche, B. Linse, F. Bry, D. Plexousakis, and G. Gottlob. RDF querying: Language constructs and evaluation methods compared. In Reasoning Web, pages 1–52, 2006.

8. C. Gutierrez, C. A. Hurtado, and A. O. Mendelzon. Foundations of semantic Web databases. In PODS, pages 95–106, 2004.

9. P. Haase, J. Broekstra, A. Eberhart, and R. Volz. A comparison of RDF query languages. In ISWC, pages 502–517, 2004.

10. P. Hayes. RDF semantics, W3C recommendation, February 2004.

11. T. Imielinski and W. Lipski Jr. Incomplete information in relational databases. J. ACM, 31(4):761–791, 1984.

12. J. P´erez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. In ISWC, pages 30–43, 2006.

13. E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C recommendation. http://www.w3.org/TR/rdf-sparql-query/, January 2008.

14. M. Y. Vardi. The complexity of relational query languages (extended abstract). In STOC, pages 137–146, 1982.