On Blank Nodes⋆

Alejandro Mallea[1], Marcelo Arenas[1], Aidan Hogan[2], and Axel Polleres[2,3]

[1] Department of Computer Science, Pontificia Universidad Católica de Chile, Chile
{aemallea,marenas}@ing.puc.cl
[2] Digital Enterprise Research Institute, National University of Ireland Galway, Ireland
{aidan.hogan,axel.polleres}@deri.org
[3] Siemens AG Österreich, Siemensstrasse 90, 1210 Vienna, Austria

Abstract. Blank nodes are defined in RDF as 'existential variables' in the same way that has been used before in mathematical logic. However, evidence suggests that actual usage of RDF does not follow this definition. In this paper we thoroughly cover the issue of blank nodes, from incomplete information in database theory, over different treatments of blank nodes across the W3C stack of RDF-related standards, to empirical analysis of RDF data publicly available on the Web. We then summarize alternative approaches to the problem, weighing up advantages and disadvantages, also discussing proposals for Skolemization.

1 Introduction

The Resource Description Framework (RDF) is a W3C standard for representing information on the Web using a common data model [18]. Although adoption of RDF is growing (quite) fast [4, § 3], one of its core features—blank nodes—has been sometimes misunderstood, sometimes misinterpreted, and sometimes ignored by implementers, other standards, and the general Semantic Web community. This lack of consistency between the standard and its actual uses calls for attention.

The standard semantics for blank nodes interprets them as existential variables, denoting the existence of some unnamed resource. These semantics make even *simple entailment checking* intractable. RDF and RDFS entailment are based on simple entailment, and are also intractable due to blank nodes [14].

However, in the documentation for the RDF standard (*e.g.*, RDF/XML [3], RDF Primer [19]), the existentiality of blank nodes is not directly treated; ambiguous phrasing such as "blank node identifiers" is used, and examples for blank nodes focus on representing resources which do not have a natural URI. Furthermore, the standards built upon RDF sometimes have different treatment and requirements for blank nodes. As we will see, standards and tools are often, to varying degrees, ambivalent to the existential semantics of blank nodes, where, *e.g.*, the standard query language SPARQL can return different results for two graphs considered equivalent by the RDF semantics [1].

Being part of the RDF specification, blank nodes are a core aspect of Semantic Web technology: they are featured in several W3C standards, a wide range of tools, and
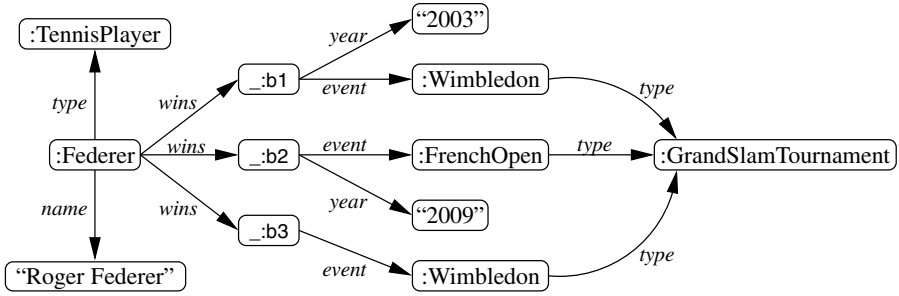
Fig. 1. An RDF graph for our running example. In this graph, URIs are preceded by ':' blank nodes by '_:' and literals are enclosed in quotation marks.

hundreds of datasets across the Web, but not always with the same meaning. Dealing with the issue of blank nodes is thus not only important, but also inherently complex and potentially costly: before weighing up alternatives for blank nodes, their interpretation and adoption—across legacy standards, tools, and published data—must be considered.

In this paper, we first look at blank nodes from a background theoretical perspective, additionally introducing *Skolemization* (§ 3). We then look at how blank nodes are used in the Semantic Web standards: we discuss how they are supported, what features rely on them, issues surrounding blank nodes, and remarks on adoption (§ 4). Next, we look at blank nodes in publishing, their prevalence for Linked Data, and what blank node graph-structures exist in the wild (§ 5). Finally, in light of the needs of the various stakeholders already introduced, we discuss proposals for handling blank nodes (§ 6).

Throughout this document, we use the the RDF graph given in Figure 1 to illustrate our discussion. The graph states that the tennis player :Federer won an event at the :FrenchOpen in 2009; it also states twice that he won :Wimbledon, once in 2003.

2   Preliminaries

We follow the abstract representation [13,20] of the formal RDF model [14,18]. This means we will consider standard notation for the sets of all URIs (**U**), all literals (**L**) and all blank nodes (**B**), all being pairwise disjoint. For convenience of notation, we write **UBL** for the union of **U**, **B** and **L**, and likewise for other combinations. In general, we write (s, p, o) for an RDF triple, and assume that $(s, p, o) \in \mathbf{UB} \times \mathbf{U} \times \mathbf{UBL}$.

For the purposes of our study, we define an *interpretation* of a graph as per [20], but without considering literals or classes since they are irrelevant for the study of blank nodes. Also, we do not consider the use of vocabularies with predefined semantics (*e.g.*, RDFS or OWL). Graphs that do not use such vocabularies are called *simple*. More precisely, define a *vocabulary* as a subset of **UL**. Given an RDF graph G, denote by terms(G) the set of elements of **UBL** that appear in G, and denote by voc(G) the set terms(G) ∩ **UL**. Then an *interpretation* $\mathcal{I}$ over a vocabulary V is a tuple $\mathcal{I} = (Res, Prop, Ext, Int)$ such that: (1) *Res* is a non-empty set of *resources*, called the *domain* or *universe* of $\mathcal{I}$; (2) *Prop* is a set of property names (not necessarily disjoint

from or a subset of *Res*); (3) $Ext : Prop \rightarrow 2^{Res \times Res}$, a mapping that assigns an *extension* to each property name; and (4) $Int : V \rightarrow Res \cup Prop$, the *interpretation mapping*, a mapping that assigns a resource or a property name to each element of V, and such that *Int* is the identity for literals. Given an interpretation mapping *Int* and a function $A : \mathbf{B} \rightarrow Res$, we define the *extension* function $Int_A : V \cup \mathbf{B} \rightarrow Res \cup Prop$ as the extension of *Int* by A, that is, $Int_A(x) = Int(x)$ if $x \in V$, and $Int_A(x) = A(x)$ if $x \in \mathbf{B}$. Based on interpretations, we have the fundamental notion of *model*:

Definition 1. *An interpretation* $\mathcal{I} = (Res, Prop, Ext, Int)$ *is a* model *of G if $\mathcal{I}$ is an interpretation over* $\mathsf{voc}(G)$ *and there exists a function $A : \mathbf{B} \rightarrow Res$ such that for each* $(s, p, o) \in G$, *it holds that* $Int(p) \in Prop$ *and* $(Int_A(s), Int_A(o)) \in Ext(Int(p))$.

The existentiality of blank nodes is given by the extension A above. We say that a graph is *satisfiable* if it has a model. Simple RDF graphs are trivially satisfiable thanks to Herbrand interpretations [14], in which URIs and literals are interpreted as their (unique) syntactic forms instead of "real world" resources. This will be important in distinguishing Skolemization in first-order logic from Skolemization in RDF.

Recall that in the context of RDF a subgraph is a subset of a graph. Then a graph G is *lean* if there is no map $h : \mathbf{UBL} \rightarrow \mathbf{UBL}$ that preserves URIs and literals ($h(u) = u$ for all $u \in \mathbf{UL}$) such that the RDF graph obtained from G by replacing every element u mentioned in G by $h(u)$, denoted by $h(G)$, is a proper subgraph of G; otherwise, the graph is *non-lean* and (formally speaking) contains redundancy. For instance, the graph G in Figure 1 is *non-lean* as the triple (_:b3, *event*, :Wimbledon) is redundant: if h maps _:b3 to _:b1 and is the identity elsewhere, then $h(G)$ is a proper subgraph of G.

3 Theoretic Perspective

In this section we look at theoretic aspects of blank nodes, focusing on background theory with respect to existentials (§ 3.1) and Skolemization (§ 3.2) in first-order logic.

3.1 Existential Variables

As per Section 2, the existentiality of blank nodes is given by the extension function A for an interpretation mapping *Int*. We now show that this interpretation of blank nodes can be precisely characterized in terms of existential variables in first-order logic.

Let G be an RDF graph. We define $Th(G)$ to be a first-order sentence with a ternary predicate triple as follows. Let $\mathbf{V}$ be an infinite set of variables disjoint with $\mathbf{U}$, $\mathbf{L}$ and $\mathbf{B}$, and assume that $\rho : \mathbf{ULB} \rightarrow \mathbf{ULV}$ is a one-to-one function that is the identity on $\mathbf{UL}$. Now, for every triple $t = (s, p, o)$ in G, define $\rho(t)$ as the fact triple($\rho(s), \rho(p), \rho(o)$), and define $Th(G)$ as $\exists x_1 \cdots \exists x_n (\bigwedge_{t \in G} \rho(t))$, where $x_1, \ldots, x_n$ are the variables from $\mathbf{V}$ mentioned in $\bigwedge_{t \in G} \rho(t)$. Then we have the following equivalence between the notion of entailment for RDF graphs and the notion of logical consequence for first-order logic.

Theorem 1 ([9]). *Given RDF graphs G and H, it holds that $G \models H$ if and only if* $Th(G) \models Th(H)$. $\square$

This theorem reveals that the implication problem for RDF can be reduced to implication for existential first-order formulae without negation and disjunction.

RDF implication in the presence of existentials is NP-Complete [14,26]. However, Pichler *et al.* [22] note that for common blank node morphologies, entailment becomes tractable. Let G be a simple RDF graph, and consider the (non-RDF) graph $\mathsf{blank}(G) = (V, E)$ as follows: $V = \mathbf{B} \cap \mathsf{terms}(G)$ and $E = \{(b, c) \in V \times V \mid b \neq c$ and there exists $P \in \mathsf{terms}(G)$ such that $(b, P, c) \in G$ or $(c, P, b) \in G\}$. Thus, $\mathsf{blank}(G)$ gives an undirected graph connecting blank nodes appearing in the same triple in G. Let G and H denote two RDF graphs with m and n triples respectively. Now, performing the entailment check $G \models H$ has the upper bound $O(n^2 + mn^{2k})$, where $k = \mathsf{tw}(\mathsf{blank}(H)) + 1$ for $\mathsf{tw}(\mathsf{blank}(H))$ the *treewidth* of $\mathsf{blank}(H)$ [22]. Note that we will survey the treewidth of blank node structures in published data in Section 5.1.2, which gives insights into the expense of RDF entailment checks in practice.

3.2 Skolemization

In first-order logic (FO), Skolemization is a way of removing existential quantifiers from a formula in prenex normal form (a chain of quantifiers followed by a quantifier-free formula). The process was originally defined and used by Thoralf Skolem to generalize a theorem by Jacques Herbrand about models of universal theories [5].

The central idea of Skolemization is to replace existentially quantified variables for "fresh" constants that are not used in the original formula. For example, $\exists x \forall y \, R(x, y)$ can be replaced by $\forall y \, R(c, y)$ where c is a fresh constant, as this new formula also states that there exists a value for the variable x (in fact, $x = c$) such that $R(x, y)$ holds for every possible value of variable y. Similarly, $\forall x \exists y \, (P(x) \rightarrow Q(y))$ can be replaced by $\forall x (P(x) \rightarrow Q(f(x)))$, where the unary function f does not belong to the underlying vocabulary, as in this case we know that for every possible value of variable x, there exists a value of variable y that depends on x and such that $P(x) \rightarrow Q(y)$ holds. When the original formula does not have universal quantifiers, only constants are needed in the Skolemization process; since only existential quantifiers are found in simple RDF graphs (see Definition 1), we need only talk about Skolem constants. However, if Skolemization was used to study the satisfiability of logical formulae in more expressive languages (*e.g.*, OWL), Skolem functions would be needed.

The most important property of Skolemization in first-order logic is that it preserves satisfiability of the formula being Skolemized. In other words, if ψ is a Skolemization of a formula φ, then φ and ψ are equisatisfiable, meaning that φ is satisfiable (in the original vocabulary) if and only if ψ is satisfiable (in the extended vocabulary, with the new Skolem functions and constants). Nevertheless, this property is of little value when Skolemizing RDF graphs since we recall that all simple RDF graphs are satisfiable.

4   Blank Nodes in the Standards

We now look at the treatment of blank nodes in the RDF-related standards, *viz.* RDF syntaxes, RDFS, OWL, RIF and SPARQL; we also cover RDB2RDF and SPARQL 1.1.

4.1   RDF Syntaxes

We first give general discussion on the role of blank nodes in RDF syntaxes.

Support for blank nodes. All RDF syntaxes allow blank nodes to be explicitly labeled; in N-Triples, explicit labeling is necessary. Explicit labels allow blank nodes to be referenced outside of nested elements and thus to be used in arbitrary graph-based data even though the underlying syntaxes (*e.g.*, XML) are inherently tree-based. Note that we will study cyclic blank node structures in published data later in Section 5.1.2.

Features requiring blank nodes. Blank nodes play two major (and related) roles in all but the N-Triples syntax. First, aside from N-Triples, all syntaxes passively default to representing resources as blank nodes when optional URIs are omitted. Second, blank nodes are also used in shortcuts for n-ary predicates and RDF lists (aka. containers) [14, § 3.3.1] in Turtle and RDF/XML, as well as containers [14, § 3.3.2] and reification [14, § 3.3.1] in RDF/XML. For example, the Turtle shortcut:

    :GrandSlam :order (:AustralianOpen :FrenchOpen :Wimbledon :USOpen) .

represents an RDF list. This would be equivalently representable in Turtle's square-bracket syntax (*left*), and full verbose forms (*right*) as follows:

```
:GrandSlam :order                          :GrandSlam :order _:b1 .
  [ rdf:first :AustralianOpen ; rdf:rest    _:b1 rdf:first :AustralianOpen .  _:b1 rdf:rest _:b2 .
  [ rdf:first :FrenchOpen ; rdf:rest        _:b2 rdf:first :FrenchOpen .      _:b2 rdf:rest _:b3 .
  [ rdf:first :Wimbledon ; rdf:rest         _:b3 rdf:first :Wimbledon .       _:b3 rdf:rest _:b4 .
  [ rdf:first :USOpen ; rdf:rest rdf:nil ]]]]   _:b4 rdf:first :USOpen .      _:b4 rdf:rest rdf:nil .
```

In the two shortcut notations, the labels of the "structural" blank nodes are left implicit. Similar shortcuts using unlabeled blank nodes hold for n-ary predicates, reification and containers in RDF/XML. Note that such shortcuts can only induce "trees" of blank nodes; *e.g.*, _:b1 :p _:b2 . _:b2 :p _:b1 . cannot be expressed without explicit labels.

Issues with blank nodes. Given a fixed, serialized RDF graph (*i.e.*, a document), labeling of blank nodes can vary across parsers and across time. Checking if two representations originate from the same data thus often requires an isomorphism check, for which in general, no polynomial algorithms are known (*cf. e.g.* [6] in the RDF context). Further, consider a use-case tracking the changes of a document over time; given that parsers can assign arbitrary labels to blank nodes, a simple syntactic change to the document may cause a dramatic change in blank node labels, making precise change detection difficult (other than on a purely syntactic level). We note that isomorphism checking is polynomial for "blank node trees" (*e.g.*, as generated for documents without explicit blank node labels) [17].

In practice. Parsers typically feature a systematic means of labeling blank nodes based on the explicit blank node labels and the order of appearance of implicit blank nodes. The popular Jena Framework[1] offers methods for checking the RDF-equivalence of two graphs. We will further discuss blank nodes in publishing in Section 5.

4.2 RDF Schema (RDFS)

RDF Schema (RDFS) is a lightweight language for describing RDF vocabularies, which allows for defining sub-class, sub-property, domain and range relations between class

---

[1] `http://jena.sourceforge.net/`

and property terms (as appropriate). The RDFS vocabulary—incl., *e.g.*, rdfs:domain, rdfs:range, rdfs:subClassOf and rdfs:subPropertyOf—is well-defined by means of a (normative) model-theoretic semantics, accompanied by a (non-normative) set of entailment rules to support inferencing [14,20].

Support for blank nodes. RDFS entailment is built on top of simple entailment, and thus supports an existential semantics for blank nodes.

Features requiring blank nodes. Blank nodes are used as "surrogates" for literals through entailment rules LG/GL in [14]. This is necessary for completeness of the entailment rules w.r.t. the formal semantics such that literal terms can "travel" to the subject position of triples by means of their surrogates [14]; for instance, the triples

  :Federer atp:name "Roger Federer" . atp:name rdfs:range atp:PlayerName .

entail the triple _:b$_{Federer}$ rdf:type atp:PlayerName by the RDFS rules LG and RDFS2 [14]. Here, the "surrogate" blank node _:b$_{Federer}$ represents the actual literal, whereas the direct application of rule RDFS2 (without LG) would result in a non-valid RDF triple having a literal in the subject position; *viz.* "Roger Federer" rdf:type atp:PlayerName .

Issues with blank nodes. Existential semantics for blank nodes makes RDFS entailment NP-Complete [14,13,26,20]. Further, ter Horst [26] showed that the RDFS entailment lemma in the non-normative section of the RDF Semantics is incorrect: blank node surrogates are still not enough for the completeness of rules in [14, § 7], where blank nodes would further need to be allowed in the predicate position. For example, consider the three triples (1) :Federer :wins _:b1, (2) :wins rdfs:subPropertyOf _:p, and (3) _:p rdfs:domain :Competitor, we still cannot infer the triple :Federer rdf:type :Competitor, since the required intermediate triple :Federer _:p _:b1 is not valid in RDF.

In practice. To overcome the NP-completeness of simple entailment, RDFS rule-based reasoners often apply Herbrand interpretations over blank nodes such that they denote their own syntactic form: this "ground RDFS entailment" is equisatisfiable and tractable [26]. Avoiding the need for (or supplementing) literal surrogates, reasoners often allow various forms of "generalized triples" in intermediate inferencing steps, with relaxed restrictions on where blank nodes and literals can appear [26,11].

4.3   Web Ontology Language (OWL)

The Web Ontology Language (OWL) is a more expressive language than RDFS (but which partially re-uses RDFS vocabulary). With the advent of OWL 2, there are now eight standard profiles of OWL. OWL Full and OWL 2 Full are given an RDF-Based Semantics [24] and so are compatible with arbitrary RDF graphs, but are undecidable [11]. OWL Lite, OWL DL, OWL 2 EL, OWL 2 QL and OWL 2 RL are given a Direct Semantics based on Description Logics (DL) formalisms and are decidable [11], but are not compatible with arbitrary RDF graphs. Further, OWL 2 RL features the OWL 2 RL/RDF entailment ruleset: a partial axiomatization of the RDF-Based Semantics.

Support for blank nodes. The RDF-Based Semantics is built on top of simple entailment [24], and considers blank nodes as existentials. The OWL Direct Semantics does not treat blank nodes in assertions, where special DL-based existentials are instead

supported; *e.g.*, the implicit assertion that :Federer won "something" can be expressed in the DL axiom Federer $\in \exists$wins.$\top$ (*i.e.*, on a level above blank nodes).

Features requiring blank nodes. For the DL-based (sub)languages of OWL, blank nodes are used to map between DL-based structural syntaxes and RDF representations: the structural syntaxes feature special n-ary predicates represented with blank nodes in RDF. For example, the DL concept $\exists$wins.$\top$ is expressed structurally as ObjectSomeValuesFrom(OPE(:wins) CE(owl:Thing)), which maps to the following three RDF triples:

    _:x a owl:Restriction . _:x owl:someValuesFrom owl:Thing . _:x owl:onProperty :wins .

Blank nodes are also required to represent RDF lists used in the mapping, *e.g.*, of OWL union classes, intersection classes, enumerations, property chains, complex keys, *etc.* An important aspect here is the locality of blank nodes: if the above RDF representation is valid in a given graph, it is still valid in an Open World since, *e.g.*, an external document cannot add another value for owl:onProperty to _:x.

Issues with blank nodes. Once RDF representations are parsed, DL-based tools are agnostic to blank nodes; existentials are handled on a higher level. RDF-based tools encounter similar issues as for RDFS; *e.g.*, OWL 2 RL/RDF supports generalized triples [11] (but otherwise gives no special treatment to blank nodes).

In practice. Rule-based reasoners—supporting various partial-axiomatizations of the RDF-Based Semantics such as DLP [12], pD* [26] or OWL 2 RL/RDF [11]—again often apply Herbrand interpretations over blank nodes. ter Horst proposed pD*sv [26] which contains an entailment rule with an existential blank node in the head to support owl:someValuesFrom, but we know of no system supporting this rule.

4.4   SPARQL Protocol and RDF Query Language (SPARQL)

SPARQL [23] is the standard query language for RDF, and includes an expressive set of query features. An important aspect of SPARQL is the notion of *Named Graphs*: SPARQL querying is defined over a *dataset* given as $\{G_0, (u_1, G_1), \ldots, (u_n, G_n)\}$ such that $u_1, \ldots, u_n \in \mathbf{U}$, and $G_0, \ldots, G_n$ are RDF graphs; each pair (u_i, G_i) is called a *named graph* and G_0 is called the *default graph*.

Support for blank nodes. With respect to querying over *blank nodes in the dataset*, SPARQL considers blank nodes as constants scoped to the graph they appear in [23, § 12.3.2]. Thus, for example, the query:

    SELECT DISTINCT ?X WHERE { :Federer :wins ?X . ?X :event :Wimbledon . }

issued over the graph depicted in Figure 1 would return $\big\{\{(?\mathsf{X}, \_\mathsf{:b1})\}, \{(?\mathsf{X}, \_\mathsf{:b3})\}\big\}$ as distinct solution mappings, here effectively considering blank nodes as constants. Interestingly, SPARQL 1.1—currently a Working Draft—introduces a COUNT aggregate, which, with an analogue of the above query, would answer that :Federer won an event at :Wimbledon *twice*. Posing the same COUNT query over a lean (and thus RDF equivalent [14]) version of Figure 1 would return *once*.

Features requiring blank nodes. SPARQL uses *blank nodes in the query* to represent *non-distinguished variables*, *i.e.*, variables which can be arbitrarily bound, but which cannot be returned in a solution mapping. For example:

SELECT ?p WHERE { ?p :wins _:t . _:t :event :Wimbledon . _:t :year _:y . }

requests players who have won *some* Wimbledon event in *some* year (*viz.*, :Federer).[2]
We note that such queries can be expressed by replacing blank nodes with fresh query
variables. A special case holds for CONSTRUCT templates which generate RDF from
solution mappings: a blank node appearing in a query's CONSTRUCT clause is replaced
by a fresh blank node for each solution mapping in the resulting RDF.

Issues with blank nodes. A practical problem posed by blank nodes refers to "round-
tripping", where a blank node returned in a solution mapping cannot be referenced in a
further query. Consider receiving the result binding (?X, _:b1) for the previous DISTINCT
query; one cannot ask a subsequent query for what year the tournament _:b1 took place
since the _:b1 in the solution mapping no longer has any relation to that in the originating
graph; thus, the labels need not correspond to the original data. Further, SPARQL's
handling of blank nodes can cause different behavior for RDF-equivalent graphs, where,
e.g., leaning a graph will affect results for COUNT.

In practice. Where round-tripping is important, SPARQL engines often offer a special
syntax for effectively Skolemizing blank nodes. For example, ARQ[3] is a commonly
(re)used SPARQL processor which supports a non-standard <_:b1> style syntax for
terms in queries, indicating that the term can only be bound by a blank node labeled
"b1" in the data. Virtuoso[4] [8] supports the <nodeID://b1> syntax with similar purpose,
but where blank nodes are only externalized in this syntax and (interestingly) where
isBlank(<nodeID://b1>) evaluates as true.

4.5   RDB2RDF

In the last 10 years, we have witnessed an increasing interest in publishing relational
data as RDF. This has resulted in the creation of the RDB2RDF W3C Working Group,
whose goal is to standardize a language for mapping relational data into RDF [7,2].
Next we show the current proposal of the Working Group about the use of blank nodes
in the mapping language.

Support for blank nodes. The input of the mapping language being developed by
the RDB2RDF Working Group is a relational database, including the schema of the
relations being translated and the set of keys and foreign keys defined over them. The
output of this language is an RDF graph that may contain blank nodes.

Features requiring blank nodes. The RDF graph generated in the translation process
identifies each tuple in the source relational database by means of a URI. If the tuple
contains a primary key, then this URI is based on the value of the primary key. If the
tuple does not contain such a constraint, then a blank node is used to identify it in the
generated RDF graph [2].

Issues with blank nodes. In the mapping process, blank nodes are used as identifiers
of tuples without primary keys [2], and as such, two of these blank nodes should not be

---

[2] Further note that blank nodes are scoped to Basic Graph Patterns (BGPs) of queries.

[3] http://jena.sourceforge.net/ARQ/

[4] http://virtuoso.openlinksw.com/

considered as having the same value. Thus, the existential semantics of blank nodes in RDF is not appropriate for this use.

5   Blank Nodes in Publishing

In this section, we survey the use of blank nodes in RDF data published on the Web. The recent growth in RDF Web data is thanks largely to the pragmatic influence of the Linked Data community [4,15], whose guidelines are unequivocal on the subject of blank node usage; in a recent book, Bizer *et al.* [15] only mention blank nodes in the section entitled "RDF Features Best Avoided in the Linked Data Context", as follows:

> *"The scope of blank nodes is limited to the document in which they appear,* [...] *reducing the potential for interlinking between different Linked Data sources.* [...] *it becomes much more difficult to merge data from different sources when blank nodes are used,* [...] *Therefore, all resources in a data set should be named using URI references."*   —**[15, § 2.4.1]**

With this (recent) guideline discouraging blank nodes in mind, we now provide an empirical study of the prevalence of blank nodes in published data (§ 5.1.1), and of the morphology of blank nodes in such data (§ 5.1.2). Finally, we briefly discuss the results of a poll conducted on public Semantic Web mailing lists (§ 5.2).

5.1   Empirical Survey of Blank Nodes in Linked Data

With the previous guidelines in mind, we now present an empirical survey of the prevalence and nature of blank nodes in Linked Data published on the Web. Our survey is conducted over a corpus of 1.118 g quadruples (965 m unique triples) extracted from 3.985 m RDF/XML documents through an open-domain crawl conducted in May 2010. The corpus consists of data from 783 different pay-level domains, which are direct subdomains of either top-level domains (such as dbpedia.org), or country code second-level domains (such as bbc.co.uk). We performed a domain-balanced crawl: we assign a queue to each domain, and in each round, poll a URI to crawl from each queue in a round-robin fashion. This strategy led to 53.2% of our raw data coming from the hi5.com FOAF exporter, which publishes documents with an average of 2,327 triples per document: an order of magnitude greater than the 140 triple/doc average from all other domains [16]. Note that this corpus represents a domain-agnostic *sample* of RDF published on the Web. The bias of sampling given by the dominance of hi5.com is important to note; thus, along with measures from the monolithic dataset, we also present per-domain statistics. Details of the crawl and the corpus are available (in significant depth) in [16, § 4].

5.1.1   Prevalence of Blank Nodes in Published Data

We looked at terms in the data-level position of triples in our corpus: *i.e.*, positions other than the predicate or object of rdf:type triples which are occupied by property and class terms respectively. We found 286.3 m unique terms in such positions, of which

Table 1. Top publishers of blank nodes in our corpus

# domain	bnodes	%bnodes	LOD?
1 hi5.com	148,409,536	87.5	✗
2 livejournal.com	8,892,569	58.0	✗
3 ontologycentral.com	2,882,803	86.0	✗
4 opiumfield.com	1,979,915	17.4	✗
5 freebase.com	1,109,485	15.6	✓
6 vox.com	843,503	58.0	✗
7 rdfabout.com	464,797	41.7	✓
8 opencalais.com	160,441	44.9	✓
9 soton.ac.uk	117,390	19.1	✓
10 bbc.co.uk	101,899	7.4	✓

165.4 m (57.8%) were blank nodes, 92.1 m (32.2%) were URIs, and 28.9 m (10%) were literals. Each blank node had on average 5.233 data-level occurrences (the analogous figure for URIs was 9.41 data-level occurrences: 1.8× that for blank nodes). Each blank node occurred, on average, 0.995 times in the object position of a non-rdf:type triple, with 3.1 m blank nodes (1.9% of all blank nodes) not occurring in the object position; conversely, each blank node occurred on average 4.239 times in the subject position of a triple, with 69 k (0.04%) not occurring in the subject position.[5] Thus, we surmise that (i) blank nodes are prevalent on the Web; (ii) most blank nodes appear in both the subject and object position, but occur most prevalently in the former, possibly due to the tree-based RDF/XML syntax.

Again, much of our corpus consists of data crawled from high-volume exporters of FOAF profiles—however, such datasets are often not considered as Linked Data, where, *e.g.*, they are omitted from the Linked Open Data (LOD) cloud diagram due to lack of links to external domains.[6] Table 1 lists the top ten domains in terms of publishing unique blank nodes found in our corpus; "**%bnodes**" refers to the percentage of all unique data-level terms appearing in the domain's corpus which are blank nodes; "**LOD?**" indicates whether the domain features in the LOD cloud diagram. Of the 783 domains contributing to our corpus, 345 (44.1%) did not publish any blank nodes. The average percentage of unique terms which were blank nodes for each domain—*i.e.*, the average of **%bnodes** for all domains—was 7.5%, indicating that although a small number of high-volume domains publish many blank nodes, many other domains publish blank nodes more infrequently. The analogous figure including only those domains appearing in the LOD cloud diagram was 6.1%.

5.1.2   Structure of Blank Nodes in Published Data

As per Section 3.1, checking $G \models H$ has the upper bound $\mathcal{O}(n^2 + mn^{2k})$, where k is one plus the *treewidth* of the blank node structure $\mathsf{blank}(H)$ [22]. Intuitively, treewidth

---

[5] We note that in RDF/XML syntax—essentially a tree-based syntax—blank nodes can only ever occur once in the object position of a triple unless rdf:nodeID is used, but can occur multiple times in the subject position.

[6] See http://lod-cloud.net/

provides a measure of how close a given graph is to being a tree; for example, the treewidth of trees and forests is 1, the treewidth of cycles is 2 (a cycle is "almost" a tree: take out an edge from a cycle and you have a tree) and the treewidth of a complete graph on n vertexes is $n - 1$. A detailed treatment is out of scope, but we refer the interested reader to [22]. However, we note that all graphs whose treewidth is greater than 1 are cyclic. From this, it follows that simple entailment checking only becomes difficult when blank nodes form cycles:

> "[...] *in practice, an RDF graph contains rarely blank nodes, and even less blank triples.*[7] *Hence, most of the real RDF graphs are acyclic or have low treewidth such as 2, and the entailment can be tested efficiently* [...]."   —[22, § 4]

To cross-check this claim, we ran the following analysis over all documents (RDF graphs) in our corpus. For each document G, we extracted blank(G) and separated out the connected components thereof using a UNION-FIND algorithm [25]. We found 918 k documents (23% of all documents) containing blank nodes, with 376 k (9% of all documents) containing "non-reflexive" blank triples. We found a total of 527 k non-singleton connected components, an average of 1.4 components per document with some blank triple. We then checked the treewidth of all 527 k components using the QUICKBB algorithm [10], where the distribution of values is given in Table 2. Notably, 98.4% of the components are acyclical, but a significant number are cyclical (treewidth greater than 1). One document[8] contained a single component C with 451 blank nodes and 887 (undirected) edges and a treewidth of 7. Figure 2 renders this graph, where vertexes are blank nodes and edges are based on blank triples; for clarity, we collapse groups of n disconnected vertexes with the same neighbors into single nodes labeled n (note that these are not n-cliques).

We conclude that the majority of documents surveyed contain tree-based blank node structures. However, a small fraction contain complex blank node structures for which entailment is potentially very expensive to compute.

Table 2. tw distribution

treewidth	# components
1	518,831
2	8,134
3	208
4	99
5	23
6	–
7	1
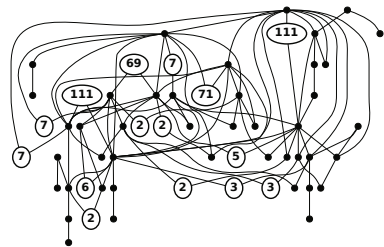
Fig. 2. C where tw$(C) = 7$

---

[7] In the terminology of [22], a *blank triple* is an element of $\mathbf{B} \times \mathbf{U} \times \mathbf{B}$.

[8] http://www.rdfabout.com/rdf/usgov/congress/people/B000084

5.2   Survey of Publishing

To further understand how blank nodes are used, we made a simple poll[9] asking users what is their intended meaning when they publish triples with blank nodes. Herein, we briefly discuss the results, and we refer the interested reader to the web page for more details. We sent the poll to two W3C's public mailing lists: Semantic Web and Linked Open Data, and got 88 responses. In order to identify active publishers, we asked participants to indicate which datasets appearing in the LOD cloud (if any) they have contributed to; 10 publishers claimed contributions to a current LOD dataset.

At the top of the web page, before the questions, we explicitly stated that *"... the poll is trying to determine what you intend when you publish blank nodes. It is not a quiz on RDF Semantics. There is no correct answer"*.

In the first question, we asked participants in which scenarios they would publish a graph containing (only) the following triple: :John :telephone _:b. We chose the :telephone predicate as an abstract example which could be read as having a literal or URI value. Participants were told to select multiple options which would cover their reason(s) for publishing such a triple. The options were: (*1.a*) *John has a tel. number whose value is unknown*; (*1.b*) *John has a tel. number but its value is hidden,* e.g., *for privacy*; (*1.c*) *John has no tel. number*; (*1.d*) *John may or may not have a tel. number*; (*1.e*) *John's number should not be externally referenced*; (*1.f*) *I do not want to mint a URI for the tel. number*; and (*1.g*) *I would not publish such a triple*. The results were as follows:

	1.a	*1.b*	*1.c*	*1.d*	*1.e*	*1.f*	*1.g*
all (88)	46.6%	23.9%	0%	2.3%	18.2%	37.5%	41.0%
lod (10)	20%	0%	0%	0%	0%	30%	70%

In the second, we asked participants to select zero or more scenarios in which they would publish a graph containing (only) the following triples: :John :telephone _:b1, _:b2. The options were (*2.a*) *John does not have a tel. number*; (*2.b*) *John may not have a tel. number*; (*2.c*) *John has at least one tel. number*; (*2.d*) *John has two different tel. numbers*; (*2.e*) *John has at least two different tel. numbers*; and (*2.f*) *I would not publish such triples*. The results were as follows:

	2.a	*2.b*	*2.c*	*2.d*	*2.e*	*2.f*
all (88)	0%	0%	23.9%	23.9%	35.2%	50.0%
lod (10)	0%	0%	0%	10%	40%	70%

The poll had an optional section for comments; a number of criticisms ($\sim$12) were raised about the :telephone example used and the restriction of having only one or two triples in the graph. This leaves ambiguity as to whether the participant would publish blank nodes at all (*intended*) or would not publish that specific example (*unintended*). Thus, we note that answers *1.g* and *2.f* might be over-represented. Also, one concern was raised about the "right" semantics of blank nodes in RDF (namely, that John has a telephone number, without saying anything about our knowledge of the number) not being an alternative; this was a deliberate choice.

---

[9] http://db.ing.puc.cl/amallea/blank-nodes-poll

Table 2. Implications of existential semantics, local constants semantics and total absence of blank nodes in theoretical aspects, standards and publishing

	standard	issue	Existential		Local constants		No Blank Nodes	
THEORY	**RDF**	*entailment*	NP-Complete	X	NP-Complete	X	PTime	✓
		equivalence	NP-Complete	X	NP-Complete	X	PTime	✓
	RDFS	*entailment*	NP-Complete	X	NP-Complete	X	PTime	✓
STANDARDS	**Syntaxes**	*shortcuts*	no change	✓	no change	✓	Sk. scheme	~
	OWL	*RDF mapping*	no change	✓	no change	✓	needs attention	X
	RIF	—	—	—	—	—	—	—
	SPARQL	*semantics*	mismatch	X	aligns	✓	aligns	✓
		query syntax	no change	✓	may need attention	~	may need attention	~
		round tripping	Sk. scheme	~	Sk. scheme	~	no change	✓
	RDB2RDF	*no primary key*	mismatch	X	aligns	✓	Sk. scheme	~
PUBL.	**RDF**	*unknown values*	no change	✓	no change	✓	Sk. scheme	~
		legacy data	ambiguous	~	unambiguous	✓	unambiguous	✓

Despite the limitations of the poll, we can see that blank nodes are not typically published with the intent of a non-existent/non-applicable semantics (*1c,1d,2b*). Obvious as it might be, the most conclusive result is that blank nodes are a divisive issue.

6 Alternative Treatments of Blank Nodes

Having covered the various desiderata for blank nodes across the several stakeholders, we now consider some high-level alternatives in light of the discussion thus far. Table 2 summarizes the implications of three conceptual paradigms for blank nodes in different aspects of theory and practice. In the column "Existential", we consider blank nodes with the current semantics of RDF (existential variables). In the column "Local constants" we consider blank nodes as constants with local scope. In the last column, we assume that blank nodes are eliminated from the standard.

The approaches in the second and third columns often require an agreed Skolemization scheme or at least the definition of frame conditions/best practices that guarantee non-conflicting Skolemization across the Web (when considering to eliminate blank nodes in published datasets) or within implementations (when merging datasets with "local constants"). Wherever we state "Sk. scheme", we refer to issues which may be solved by such agreed mechanisms used to generate globally unique URIs from syntax with implicit or locally scoped labels. We discuss the practicalities of (and proposals for) such schemes in Section 6.1. The core principle here is that locally-scoped artifacts can not "interfere" with each other across documents. For example, to enable syntactic shortcuts in the absence of blank nodes (*i.e.*, when the RDF universe consists of **UL**), the URIs generated must be globally unique to ensure that legacy resources are not unintentionally referenced and redefined when labels are not given.

In terms of the SPARQL *query syntax*, the use of blank nodes is purely syntactic and is decoupled from how RDF handles them. Even if blank nodes were discontinued,

the blank node syntax in queries could still be supported (though a fresh syntax may be preferable). In terms of SPARQL *round tripping*, we refer to enabling blank nodes (either local or existential) to persist across scopes, where one possible solution is, again, Skolemization schemes. Note that SPARQL 1.1 will allow to "mint" custom URIs in results as an alternative to blank nodes.

For RDB2RDF mappings where *no primary key* is available, when removing blank nodes completely, the problem is again essentially one of implicit labeling, where a (possibly specialized) Skolemization scheme would be needed. We note that, unlike for existentials, a constant semantics for blank nodes would align better with the underlying semantics of the database.

For RDF *legacy data*, our intention is to note that for a consumer, how ad-hoc data should be interpreted remains ambiguous in the presence of non-leanness. Again, taking our original example and assuming it has been found on the Web, with the existential blank node semantics it is not clear whether leaning the data would honor the intent of the original publisher. Deciding whether to lean or not may then, *e.g.*, affect SPARQL query answers. For the other alternatives, leanness is not possible and so the ambiguity disappears. Conversely, we note that in a practical sense, publishers do not lose the ability to state *unknown values* in the absence of existential variables; such values can be expressed as *unique* constants which have no further information attached. Finally, we note that Table 2 does not quite cover all concrete alternatives. One other possibility we considered was not allowing blank nodes to ever be explicitly labeled, such that they form trees in the syntaxes, essentially enforcing all graphs to have a blank-treewidth of 1. Note that some syntaxes (like Turtle and RDF/XML without nodeID) guarantee a tree-structure for blank nodes. As discussed in Sections 3.1 & 4.1, this would make the isomorphism-checks and simple and RDF(S) entailment-checks tractable, although still with an implementational cost.

6.1   Skolemization Schemes

The following ideas have been proposed as recommended treatment of blank nodes in RDF. They do not necessarily require changes in the standards—or at least not to the semantics of RDF—but are intended as guidelines for publishers (*i.e.*, "best practices"). We will consider a set $\mathbf{S}$ of Skolem constants, such that every time Skolemization occurs, blank nodes are replaced with elements of $\mathbf{S}$. Different alternatives will consider different behaviors and nature for this set.

6.1.1 $\mathbf{S} \subseteq \mathbf{U}$, Centralized

The idea of this alternative is to offer a centralized service that "gives out" fresh URIs upon request, ensuring uniqueness of the generated constants on a global scale. Formally, there would be a distinguished subset of the URIs, $\mathbf{S} \subseteq \mathbf{U}$, such that all Skolem constants belong to $\mathbf{S}$. Every time the service gets a request, it returns an element $s \in \mathbf{S}$ such that s has not been used before. This is very similar to what URL shorteners do[10]. Since the returned constants are also URIs, they can be used in published documents.

It is not clear who should pay and take responsibility for such a service (the obvious candidate being the W3C). The costs of bandwidth and maintenance can be too high.

---

[10] See, for example, `http://bit.ly/` or `http://is.gd/`.

For example, the system should be able to cope with huge individual requests (see table 1). Also, the very idea of centralizing a service like this seems to go against the spirit of the Semantic Web community, though this might not be relevant for everyone. Further, the mere existence of such a system will not guarantee that users will only use this option when Skolemizing. Publishers will still have the freedom of using other methods to replace blank nodes with constants of their own choice.

6.1.2   S $\subseteq$ U, Decentralized

Similar to the previous case, but with no central service, so each publisher will generate their constants locally. This has been discussed already by the RDF Working Group[11] and by the general community through the public mailing list of the Semantic Web Interest Group[12]. In both cases, the proposal is to establish a standard (but voluntary) process for generating globally unique URIs to replace blank nodes for querying, publishing and performing other operations with RDF graphs. A requirement of this process is that it contains a best practice that avoids naming conflicts between documents. Naming conflicts on the Web are typically avoided by pay level domains, since they guarantee a certain level of "authority"[16]. Along these lines, at the time of writing, the proposal of the RDF Working Group is to add a small section on how to replace blank nodes with URIs to the document "RDF Concepts" [18, §6.6]. The idea would be that blank nodes are replaced with well-known URIs [21] with a registered name that is to be decided (probably "genid" or "bnode") and a locally-unique identifier, which would make a globally-unique URI, since publishers would only be supposed to use their own domains. For example, the authority responsible for the domain example.com could mint the following URI for a blank node:

http://example.com/.well-known/bnode/zpHvSwfgDjU7kXTsrc0R

This URI can be recognized as the product of Skolemization. If desired, a user could replace it with a blank node. Moreover, publishers can encode information about the original blank node in the identifier, such as the name of the source graph, the label of the blank node in that graph, and the date-time of Skolemization.

It should be noted though that, although no central control authority is needed for such decentralized Skolemization, this proposal would allow third parties malicious, non-authoritative use of bnode-URIs which are not in their control (*i.e.* outside their domain): the often overlooked "feature" of blank nodes as local constants not modifiable/redefinable outside the graph in which they are published would be lost.

7   Conclusions

In this paper, we have provided detailed discussion on the controversial and divisive issue of blank nodes. Starting with formal considerations, we covered treatment of blank nodes in the W3C standards, how they are supported, and what they are needed for. The main use-case for blank nodes is as locally-scoped artifacts which need not be explicitly

---

[11] See http://www.w3.org/2011/rdf-wg/wiki/Skolemisation.
[12] See http://www.w3.org/wiki/BnodeSkolemization.

labeled. We also looked at how blank nodes are being published in Linked Data, where they are perhaps more prevalent than the best-practices would suggest; we also note that although rare, complex structures of blank nodes are present on the Web.

Informed by our earlier discussion, we proposed and compared three conceptual paradigms for viewing blank nodes. The first one is rather radical and involves eliminating blank nodes from the standard, with the additional consequences of having to deal with legacy data and with possible changes in other standards that rely on blank nodes in one way or another. The second one consists in standardizing the already widespread interpretation of blank nodes as local constants; most standards (like OWL and SPARQL) would not need to change at all. The third one is twofold: keeping the existential nature of blank nodes in RDF, and ensuring this is the meaning that other standards follow, for example, by means of a best-practices document; in this case, even if SPARQL were to follow the notions of leanness and entailment, query answering would not be expensive in most cases due to a good structure for blank nodes in currently published data. In all these alternatives, a Skolemization scheme would be handy as an aid for publishers to update their would-be obsolete data. Finally, we note that no alternative stands out as "the one solution to all issues with blank nodes". Discussion is still open and proposals are welcome, but as the amount of published data grows rapidly, a consensus is very much needed. However, in the absence of an undisputed solution, the community may need to take an alternative which might not be the most beneficial, but the least damaging for current and future users.

References

1. Arenas, M., Consens, M., Mallea, A.: Revisiting Blank Nodes in RDF to Avoid the Semantic Mismatch with SPARQL. In: RDF Next Steps Workshop (June 2010)
2. Arenas, M., Prud'hommeaux, E., Sequeda, J.: Direct mapping of relational data to RDF. W3C Working Draft (March 24, 2011),
   `http://www.w3.org/TR/rdb-direct-mapping/`
3. Beckett, D., McBride, B.: RDF/XML Syntax Specification (Revised). W3C Recommendation (February 2004), `http://www.w3.org/TR/rdf-syntax-grammar/`
4. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – The Story So Far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
5. Buss, S.R.: On Herbrand's Theorem. In: Leivant, D. (ed.) LCC 1994. LNCS, vol. 960, pp. 195–209. Springer, Heidelberg (1995)
6. Carroll, J.J.: Signing RDF graphs. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 369–384. Springer, Heidelberg (2003)
7. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language. W3C Working Draft (March 24, 2011), `http://www.w3.org/TR/r2rml/`
8. Erling, O., Mikhailov, I.: RDF Support in the Virtuoso DBMS. In: Pellegrini, T., Auer, S., Tochtermann, K., Schaffert, S. (eds.) Networked Knowledge - Networked Media. SCI, vol. 221, pp. 7–24. Springer, Heidelberg (2009)
9. Franconi, E., de Bruijn, J., Tessaris, S.: Logical reconstruction of normative RDF. In: OWLED (2005)
10. Gogate, V., Dechter, R.: A Complete Anytime Algorithm for Treewidth. In: UAI, pp. 201–208 (2004)

11. Grau, B.C., Motik, B., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language: Profiles. W3C Recommendation (October 2009), `http://www.w3.org/TR/owl2-profiles/`
12. Grosof, B., Horrocks, I., Volz, R., Decker, S.: Description Logic Programs: Combining Logic Programs with Description Logic. In: WWW (2004)
13. Gutierrez, C., Hurtado, C., Mendelzon, A.O.: Foundations of Semantic Web Databases. In: 23rd ACM SIGACT-SIGMOD-SIGART (June 2004)
14. Hayes, P.: RDF Semantics. W3C Recommendation (February 2004)
15. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space, vol. 1. Morgan & Claypool (2011)
16. Hogan, A.: Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora. PhD thesis, DERI Galway (2011)
17. Kelly, P.J.: A congruence theorem for trees. Pacific Journal of Mathematics 7(1) (1957)
18. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation (February 2004), `http://www.w3.org/TR/rdf-concepts/`
19. Manola, F., Miller, E., McBride, B.: RDF Primer. W3C Recommendation (February 2004)
20. Muñoz, S., Pérez, J., Gutierrez, C.: Simple and Efficient Minimal RDFS. J. Web Sem. 7(3), 220–234 (2009)
21. Nottingham, M., Hammer-Lahav, E.: Defining Well-Known Uniform Resource Identifiers (URIs). RFC 5785 (April 2010), `http://www.ietf.org/rfc/rfc5785.txt`
22. Pichler, R., Polleres, A., Wei, F., Woltran, S.: dRDF: Entailment for Domain-Restricted RDF. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 200–214. Springer, Heidelberg (2008)
23. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation (January 2008), `http://www.w3.org/TR/rdf-sparql-query/`
24. Schneider, M.: OWL 2 Web Ontology Language RDF-Based Semantics. W3C Recommendation (October 2009), `http://www.w3.org/TR/owl2-rdf-based-semantics/`
25. Tarjan, R.E., van Leeuwen, J.: Worst-case analysis of set union algorithms. J. ACM 31(2), 245–281 (1984)
26. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. J. of Web Sem. 3, 79–115 (2005)