Blank Nodes in RDF

Lei Chen
College of Computer and Information, Hohai University, Nanjing, Jiangsu Province, P.R.China 210098;
Department of Computer and Information Engineering, Huainan Normal University,
Huainan, Anhui Province, P.R.China 232038
Email: chenleihnnu@gmail.com

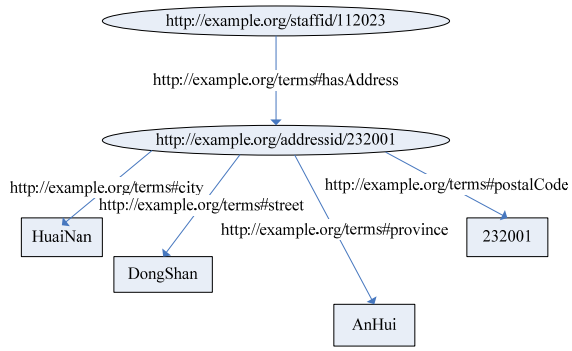
Haifei Zhang, Ying Chen and Wenping Guo
College of Computer and Information, Hohai University, Nanjing, Jiangsu Province, P.R.China 210098
Email: { haifeizhang , yingchen_2009, gwp,}@hhu.edu.cn

***Abstract*—Semantic Web plays an important role in the Web of future. The RDF data is the key component which establishes the basis of the Semantic Web. In this paper, we conlude the usages and the possible problems of blank nodes of RDF with detailed analyses of the applications and semantics of the blank nodes in RDF graphs. Give special attentions to the inconsistency between RDF semantics and SPARQL semantics of blank nodes. Employ the concept of "lean graph" in the pre-process of the RDF data operation, propose a method of using the entailment relations between RDF graphs and the transformation from blank nodes to the URI references to eliminate the blank nodes in RDF graphs, and give the theoretic background to support the method. Lastly, some referenced methods of transforming the blank nodes to URI references are provided.**

***Index Terms*—blank nodes, RDF, SPARQL, lean graph, entailment**

I. INTRODUCTION

With the emergence of the Semantic Web, Resource Description Framework (RDF) has been a prevalent data model which is used to describe knowledge in some domains [1]. RDF is a graph-like data structure with respect to expresses the resources and the relations between resources using the nodes and the directed edges between the nodes in the graph (as can be seen in Fig. 1(a)). According to the features of the descriptions in knowledge, we can also express an RD F graph in a group of RDF triples (see Fig. 1(b)).



(a) an RDF graph

```
exstaff:112023  exterms:has Address  exaddressid:232001.
exaddressid:232001  exterms:province  "AnHui".
exaddressid:232001  exterms:city  "HuaiNan".
exaddressid:232001  exterms:street  "DongShan".
exaddressid:232001  exterms:postalCode  "232001".
```

(b) a group of RDF triples

Figure 1. An RDF graph and its corresponding RDF triples

Blank node is a special kind of node in RDF graphs, there is nothing to identify them. In RDF, a blank node is a node representing some resource for which a URI reference or literal is not given. The resource represented by a blank node is also called an anonymous resource. By RDF standard, a blank node can only be used as subject or object when we describe konowledge and information in RDF triples. Some blank nodes may have their node IDs, but not all blank nodes have to be labeled in all RDF serializations. Blank node just expresses the semantic of "something exists" as a tag. When we use the RDF triples to express an RDF graph, and mostly, we often use the form "_:xxx" to show this is a blank node in the RDF triple sets. For instance, the node "http://example.org/addressid/232001" was replaced by a blank node, and thus the RDF graph and the according RDF triples are shown in the Fig. 2(a) and 2(b).

Blank nodes are used to represent these unknown resources, and blank nodes are also used when the relationship between a subject node and an object node is n-ary (as is the case with collections).

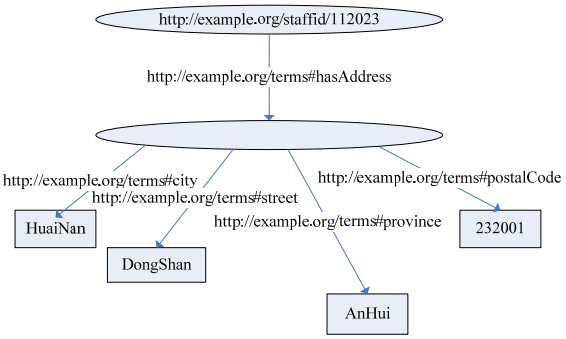
This paper gives a detailed analysis about the blank node from two respects, one is the usabilities of the blank node and the other is the negative factors. In section 2, we conclude the usages of the blank node we analyze the possible of the problem and the theoretic background in section 3, and give the reference methods in section 4; lastly, we conclude the paper in section 5.

Corresponding author: Lei Chen (chenleihnnu@gmail.com)

doi:10.4304/jsw.7.9.1993-1999

II. THE FUNCTIONS OF BLANK NODES

In Linked data publication, it is not proposed to describe information using blank nodes [2], but the common mechanisms of publishing linked data can not be avoided to use blank nodes, which gives the developers some conviences. The main reason is that the blank node has the capabilities to express some special semantics in knowledge description. We attribute the



(a) an RDF graph with one blank node

```
exstaff:112023  exterms:has          Address _:a1.
_:a1            exterms:province     "AnHui".
_:a1            exterms:city         "HuaiNan".
_:a1            exterms:street       "DongShan".
_:a1            exterms:postalCode   "232001".
```

(b) a group of RDF triples with one blank node

Figure 2. An RDF graph with a blank node and its corresponding RDF triples

usages to 5 aspects:

A. *Blank Nodes Have the Capability to Describe the Structural Information to Encapsulate the N-ary Association and the Closed Sets.*

As shown in Fig.2, when the information to be described is of multi-component structure, and we need not or can not identify the whole information, we use the blank nodes to express the existence of the information.

Also in RDF, we often need to describe groups of things, for example, several authors of a book and some students of a team. The container is a structural concept in RDF data model. There are three types of container in RDF model, which are Bag, Sequence and Alternative. The Bag (rdf:Bag) represents a group of resources or literals, where there is no significance in the order of the members; the Sequence (rdf:Seq) represents a group of resources or literals, possibly including duplicate members, where the order of the member is significant; and, the Alternative (rdf:Alt) represents a group of resources or literals that are alternatives. And we can describe these three types containers data structure with blank nodes. Fig. 3 gives such a collection in RDF triples with blank nodes:

```
exbook:001   exterms:author   _:auther.
_:auther     rdf:type         rdf:Bag.
_:auther     rdf:_1           exauthor:010.
_:auther     rdf:_2           exauthor:015.
_:auther     rdf:_3           exauthor:023.
```

Figure 3. An rdf:Bag container in RDF

Another case is that we can describe a collection with a blank node in RDF graphs. it is needed to use the blank node to express the anonymous nodes so that we can organize the members of the collections (see Fig.4).

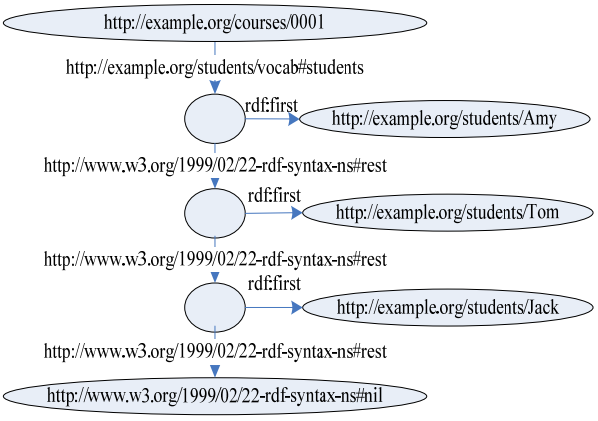


Figure 4. A collection in RDF

B. *Blank nodes Have the Capability to Describe the Refication.*

Refication is used to describe other RDF statements using the RDF format, for instance, to record information about when statements were made, who made them, or other similar information (this is sometimes referred to as "provenance" information). The RDF reification vocabulary consists of the type rdf:Statement, and the properties rdf:subject, rdf:predicate, and rdf:object, with these vocabularies, we can transform any RDF triple to its refication, for example, we have an RDF triple as:

exproduts: No001 exterms:weight "2.4"^^xsd:decimal.

Then we get the refication of the triple:

_b rdf:type rdf:Statement .
_b rdf:subject exproducts: No001 .
_b rdf:predicate exterms:weight .
_b rdf:object "2.4"^^xsd:decimal

Here, the blank node _b is used to express the refication. With the refication, we can use another triple to describe the information about the original triple:

exstore:No010 exterms:publish _b.

C. *Blank Nodes Can Hide the Unexposible Information*

In many senses, the publishers may not want to expose their data completely, so the blank nodes can help him or her to shield some sensitive information. For example, the shop want to publish some shopping information, we can replace the real customer's identity with the blank node:

_:a1 exterms:buy exproduct:No001.
_:a1 exterms:buyTime "2011-6-11"^^xsd:date.

Thus, the browsers outside just only can get the information about the shopping, but can not know any other information about the identity of the customer. The blank nodes protected the inner information in a good manner.

D. *Blank Nodes Can Express the Multi-relationship*

RDF can only describes the binary relationship directly, like $p(s,o)$, which p is the binary predicate, and *s, o* are

the two parameters of the predicate *p*. As to the multi-relationship $p(s_1, s_2, ..., s_n)$, the RDF data structure must expresses this using an indirected form with the help of blank nodes. We can choose one participant s_1 as the subject of the relationship p, and a blank node as the object, then, we create a group of relationships $p_2, p_3, ..., p_n$ to express the relationships between the blank node and the participants $s_2, s_3, ..., s_n$, respectively, we can express this in a rule below:

$$p(s_1, s_2, ..., s_n) :- p(s_1, b), p2(b, s_2), p3(b, s_3), ..., pn(b, s_n) \quad (1)$$

Here, b is the blank node and $p_2, p_3, ..., p_n$ is the relationship between the blank node and the $s_2, s_3, ..., s_n$ respectively. Fig.2 also describes the multi-relationship between a person and his address. As depicted there, the blank node is used to connect all of the participants like a bridge.

E. Blank Node Can Describe the Entities Which Are Hard to Identify

Blank node has no URI reference, but it can describe the entity hard to identify. For example, sometimes, data publishers like to identify a person with the email address, but the problem is when the person and his or her email co-occur at the same time, we can not distinguish them, just see the triple below:

<mailto:leichen_2009@163.com>
exterms:email <mailto:leichen_2009@163.com>.

So, if there is no a recognized way to identify some resource, we can use the blank node to describe the information about the resource temporarily, just like the triples below:

_:b exterms:email <mailto:leichen_2009@163.com>.
_:b rdf:type exterms:Person.
_:b exterms:name "Chenney"

III. THE PROBLEMS OF THE BLANK NODE AND THE ANALYSIS OF THE REASONS

In RDF graph, blank node has no name and identity, it just entails something exists, so, in the RDF semantics, blank nodes can be understood as the existential variables. Blank node can bring some conveniences to express the information of the resources and the relationships between them, but at the same time, it also brings some problems and negative impacts in using them, we concluded the problems into three aspects:

A. The Problem of SPARQL Queries.

SPARQL is the RDF query language recommended by the W3C and it employees the "graph pattern matching" in evaluating the result of a query [3]. The core component of SPARQL queries is a set of triple patterns in the form of $< s, p, o >$. Here, s, p, o corresponds to the subject, predicate, and object of an RDF triple, but they can be variables as well as RDF terms. Within a SPARQL query, the user specifies the known RDF terms of triples and leaves the unknown ones as variables in triple patterns. A triple pattern matches a subset of the RDF data; here the RDF terms in triple in the triple pattern correspond to the ones in the RDF data. A triple pattern applied to an RDF graph generates an unordered bag of solutions. A solution is a set of bindings, each of which consists of a pair of a variable and its bound value, that is, corresponding RDF terms in the matched subset of the RDF data. The result of a group of triple patterns is the join of the results of individual triple patterns.

In SPARQL 1.1, basic graph pattern matching is defined as " A basic graph pattern is matched against the active graph for that part of the query. Basic graph patterns can be instantiated by replacing both variables and blank nodes by terms, giving two notions of instance: Blank nodes are replaced using an RDF instance mapping, from blank nodes to RDF terms; variables are replaced by a solution mapping from query variables to RDF terms. [10]" This shows a point that the blank nodes in RDF graphs should be reduced or eliminated.

Based on the definition of the query matching in SPARQL [4], we also know that the blank nodes in RDF are treated as the RDF terms when SPARQL queries are evaluating over RDF graphs, which means that the labels of blank nodes, just like the URI references and the literals, are all regarded as the identities of the resources. This semantic of blank node in SPARQL is different from which in RDF semantic obviously, as can be seen in the example below:

Suppose there are two RDF graphs here:

G	G'
ex:a ex:b _:b1.	ex:a ex:b _:b3.
ex:p ex:b _:b1.	ex:a ex:b _:b3.
ex:a ex:b _:b2	

Figure 5. two equivalent RDF graphs

We know that the graph G is equivalent to the graph G' in terms of RDF semantics. Now we pose a SPARQL query over them respectively:

```
q:
select DISTINCT ?x
where { ?x ex:b ?y.
        ?x ex:b ?z.
        FILTER (?y!=?z) }
```

Then, we get the result {ex:a} from the graph G and NULL from the graph G'. This means that we pose the same query over two logically equivalent graphs respectively and get different results, which shows the difference between the semantics of RDF and SPARQL. In RDF all the blank nodes are explained as the existential variables and the blank nodes just tell the existence of some resources and nothing else to describe; but in SPARQL, different blank nodes represent different values, which is the reason why we can get the result {ex:a} from the graph G.

We can also see that the graph G is not lean, on the other hand, the graph G' is a lean graph [5], so, according to the RDF semantics, the (ex:a ex:b _:b2) is a redundant triple, the meaning it expresses is involved in the semantics of the triple "ex:a ex:b _:b1". So, as can be seen in this example, when there are some redundant triples in an RDF graph, some queries may be evaluated in a wrong way.

In other words, since RDF blank nodes allow infinitely many redundant solutions for many patterns, there can be infinitely many pattern solutions (for example obtained by replacing URI references with blank nodes or blank nodes by different blank nodes). It is necessary, therefore, to somehow delimit the solutions for a basic graph pattern. If we delete the redundant blank nodes in the original graph and restrict some entailments evaluation about blank nodes, the inconsistency of SPARQL query would be reduced.

B. The Problem of Merging RDF Graphs

The main purpose of the Semantic Web is to express the data on the web in RDF graphs, the scope is Web-wide-spread, so the manage and the merge the RDF data distributed on the Web is the important operators of the Web of data. It is easy to merge the graphs without the blank nodes, the ground graphs; the result of the merge of these RDF graphs is the union of these graphs. But when we merge some graphs with blank nodes, it should pay attention to whether the graphs have the same blank node labels or not. The blank nodes with the same labels in two RDF graphs could not be regarded as same because of the local feature of the blank nodes. In other words, the scope of the validity of a blank node is the RDF graph in which the blank node is involved. Ref.[5] proposed a method which renames the blank node with the unique label which has not been presented in the two original graphs, but as analyzed before, if the triples with the blank node represent the redundant information, after renamed the blank nodes and merge the two graphs, the redundant information are also merged into the result graph, so, we advise that we should 'leanize' the RDF graphs to be merged, and rename the blank node labels not eliminated yet before the merge operation, and thus we get a result graph with less redundant information.

C. The Problem of Publishing Linked Data

As discussed in Linked Data community, two points must be solved in the process of accessing the data on the Semantic Web. The first was a remark that when using http-type URI references, there are expectations that something actually exists at that web URL. The second was the open question about how Semantic Web clients are supposed to find RDF data on the web. The notion of Linked Data is to bring the concept and benefits of hyperlinking between HTML documents on the World Wide Web to RDF documents on the Semantic Web. The core principle is that http-type URI references should be used for RDF resources, so that RDF documents can exist at those locations describing the resources. When those documents mention other resources, if they have http-type URI references then Semantic Web clients can jump from document to document finding more information as it goes.

Based on the acquirement of the data access, one principle of publishing linked data is to avoid from using blank nodes for blank nodes make your data untouched with the information outside [2]. Since blank nodes have no URI references, so the linkages will be broken at the points of the blank nodes, which will brings many troubles in the data query and mining. And what's more, some ontology specification like FOAF has also dropped blank nodes in favor of URI references. So eliminate the blank nodes in your RDF graph is also an important pre-process before you publish your data on the Web.

IV. Try To Eliminate The Blank Nodes In Your RDF Graphs

As investigated before, we recognized that the blank node brings both the facilities and troubles to our data management and process, especially on the inconsistency of query on the RDF datasets. The root course of the inconsistency between the RDF semantics and the SPARQL semantics on blank node is, in RDF semantics, blank nodes are treated as the variables, and in SPARQL blank nodes are treated as constant symbols. When a SPARQL query evaluates over an RDF graph, it treats the blank nodes as local constant terms. It is perhaps impossible to clear all the blank nodes in an RDF graph, but the blank nodes in the triples which represent the redundant information and the blank nodes which can be mapped into some URI references may be eliminated to make the RDF graph leaner and cleaner.

We mainly propose three methods to reduce the blank nodes in RDF graphs, first, we employ the entailment relations between RDF subgraphs to make an RDF graph lean; and second, we utilize the "owl:InverseFunctionalProperty" to map some blank nodes to existing URI references in the RDF graph; and lastly, we give a heuristic method to assign URI references to the blank nodes.

Definition 1 (Instance)

Suppose U , L , B are three mutually disjoint sets of URI references, Literals and blank nodes, m is a mapping from B to ULB ($U \cup L \cup B$), an RDF graph G' generated by m on the RDF graph G is an instance of G .

According to the definition, every RDF graph is the instance of itself and has itself as the instance.

Definition 2 (Proper instance)

Suppose U , L , B are three disjoint sets of URI references, Literals and blank nodes respectively, m is a mapping from B to UL ($U \cup L$), an RDF graph G' generated by m on the RDF graph G is an proper instance of G .

According to the definition, the proper instances have less blank nodes than the original RDF graph.

Definition 3 (Lean graph)

An RDF graph is lean if it has no instance which is a proper subgraph of the graph.

As we can see, Non-lean graphs have internal redundancy and express the same content as their lean subgraphs [5]. The procedure of the leanization is the procedure of reducing the blank nodes. According to the definition of the lean graph, there are two ways to leanize an RDF graph, one is to use the entailments between the subgraphs to reduce the triples with blank nodes, and the other is to map some blank nodes to the URI references existing in the graph already.

A. *Use Entailments between Subgraphs to Reduce the Blank Nodes*

Definition 4 (Simple Interpretations)

Let V be a set of terms. An interpretation of V is a 5-tuple $< I_R, I_P, l_{ext}, l_s, l_l >$ where I_R is a set of resources containing $L_P(V)$, I_P is a set of properties, $l_{ext} : I_P -> 2^{I_R \times I_R}$ maps each property to a set of pairs of resources (the extension of the property), $l_s : U(V) -> I_R \cup I_P$ maps each URI reference to a resource or a property, and $l_l : L_T(V) -> I_R$ maps each typed literal to a resource.

Definition 5 (Models)

Let G be an RDF tripleset, and V be a set of terms that contains the set of terms of G, i.e. such that $(U(V(G)) \cup L(V(G))) \subseteq V$. An interpretation $< I_R, I_P, l_{ext}, l_s, l_l >$ of V is a model of G iff there exists a mapping $\tau : V(G) -> I_R \cup I_P$ such that:

G1:
ex:a rdf:type foaf:Person.
ex:a foaf:name "Alice".
ex:a foaf:knows ex:b

G2:
_:b rdf:type foaf:Person.
_:b foaf:name "Alice".
_:b foaf:knows ex:b

G3:
_:b rdf:type foaf:Person.
_:b foaf:name "Alice".
_:b foaf:knows ex:b
_:b foaf:mbox <mailto:alice@work.example>

G4:
_:b rdf:type foaf:Person.
_:b foaf:name "J. Alice".
_:b foaf:knows ex:b
_:b foaf:mbox <mailto:alice@work.example>

Figure 6. A group of RDF subgraphs

1. for each plain literal $l \in L_P(V(G)), \tau(l) = l$;
2. for each typed literal $l \in L_T(V(G)), \tau(l) = \tau_l(l)$;
3. for each uriref $u \in U(V(G)).\tau(u) = \tau_s(u)$
4. for each blank $b \in B(V(G)), \tau(b) \in I_R$;
5. for each triple $< s, p, o > \in G, < \tau(s,)\tau(o) > \in l_{ext}(\tau(p))$.

Definition 6 (Satisfiability, Entailment)

Let G and H be two RDF triple-sets. We say that G is satisfiable if there exists an interpretation that is a model of G. we say that H is a semantic consequence of G (we also say that G entails H, and note $G \models H$) if every model of G is also a model of H.

Entailment is the key idea which connects model-theoretic semantics to real-world applications [5][10][13]. The RDF semantics specification defines four increasingly expressive normative entailment relations between RDF graphs, namely simple, RDF, RDFS, and D entailment, where the latter extends RDFS entailment with support for data-type (e.g., strings and integers). Support A and B are two RDF subgraphs, if A entails B, then any interpretation that makes A true also makes B true, so that an assertion of A already contains the same "meaning" as an assertion of B; one could say that the meaning of B is somehow contained in, or subsumed by, that of A. Furthermore, we also can define a possible extension of RDFS entailment that is more in line with description logic, namely OWL DL and OWL 2 DL. The difference between these specifications is the complexity of the entailment evaluation and the reasoning. As concluded in Ref.[7], simple, RDF, and RDFS entailment are NP-complete in the combined size of the graphs. This high complexity is due to the presence of blank nodes (essentially existentially quantified variables): if the entailed graph is known to be ground, the respective problems turn out to be decidable in polynomial time. So we can recognize that blank nodes increase the complexity of the data management and process.

Here we just only consider the simple entailment defined as a foundation of the entailment in RDF semantics.

Blank node is regarded as existential variable in RDF semantics, it means "there exists something", so, when the triples with blank nodes occur with some common triples (the triples without blank nodes component) in a RDF graph, the meaning they have asserted perhaps has already been involved in some other triples. In other words, the subgraph composed of some common triples entails the subgraph composed of some triples with some blank nodes.

However, since an infinity of interpretations must be evaluated according to the definition of the entailment in RDF semantics, We give a theorem below to help to determinate if a entailment relation exists between two RDF subgraphs or not.

Theorem 1(Interpolation Lemma)

Let G and H be two RDF subgraphs. Then $G \models H$ iff there exists an instance H' of H such that $H' \subseteq G$.

Suppose G1, G2, G3 and G4 are four RDF subgraphs below:

If G1 and G2 co-occur in an RDF graph, according to the definition of entailment in RDF semantics, the meaning of G2 is contained in the meaning of G1, so we consider that the information expressed by G2 is redundant, and to avoid to the inconsistency of SPARQL query and other operations, subgraph G2 should be eliminated.

Compared G1 with G3, as we can not tell whether the value of foaf:mbox of the resource ex:a is <mailto:alice@work.example> or not, so we can not decide whether G1 entails the G3 or not; Based on the definition, we know G1 does not entail G4 obviously.

B. *Use Owl:Inverse Functional Property to Map Blank Nodes to URI References*

If a property is declared to be inverse-functional, the object of a property statement uniquely determines the subject (some individual). More formally, if we state that p is an owl:InverseFunctionalProperty, then this asserts that a value y can only be the value of p for a single instance x, i.e. there cannot be two distinct instances x1

and x2 such that both pairs (x1, y) and (x2, y) are instances of p.

owl:InverseFunctionalProperty is just like the concept of primary key in the relational schema of database. To an entity, the inverse functional property value is unique [6], so we can use the character to determine if the entity described by a blank node and the entity described by a URI reference are the same entity or not, which means that, if the value of the owl:InverseFunctionalProperty (for example, the e-mail address) of the entity described by a blank node is equivalent to the corresponding value of the entity described a URI reference, then we can believe the two entities are same.

G:
ex:a rdf:type foaf:Person.
ex:a foaf:mbox <mailto:alice@work.example>
_:b rdf:type foaf:Person.
_:b foaf:name "Alice".
_:b foaf:knows ex:b.
_:b foaf:mbox <mailto:alice@work.example>

G':
ex:a rdf:type foaf:Person.
ex:a foaf:name " Alice".
ex:a foaf:mbox <mailto:alice@work.example>
ex:a foaf:knows ex:b.

Figure 7. map a blank node to a URI reference existing

In Fig.7, although G is a lean graph according to the RDF semantics, as we can see, the values of owl:InverseFunctionalProperty of the blank node _:b and the URI reference ex:a are equivalent, so we can say the entity described by ex:a and the entity described by _:b is the same entity, and we can map '_:b' to 'ex:a' and eliminate some triples in G. The result after the operators is G'. As we can see, the graph G' is 'leaner' than G'.

Specially in OWL 2 recommended by W3C, the property InverseFunctionalProperty is replaced with HasKey, which states that each instance of some class is uniquely identified by one object property expression—that is, no two distinct instances can coincide on the values of all object property of the HasKey and all data property expressions. The HasKey property is similar to the property InverseFunctionalObjectProperty, the main differences being that the former property is applicable only to individuals that are explicitly named in an ontology, while the letter property is also applicable to individuals whose existence is implied by existential variable. We refer the interested readers to Ref. [8].

C. Treat Blank Nodes as Constant Symbols and the Skomemization

Perhaps the blank nodes are stubborn yet after we use the entailment and the simple inference to leanize a RDF graph. How to process the blank nodes in the RDF graphs is an important but easy to overlook to applications and developers. Most of the current platforms just regard the blank nodes as constant symbols simply. It is noteworthy that, these constants should be local to the RDF graph in which the blank nodes embedded.

The mechanism of treating blank nodes as local constant symbols is the main reason for the inconsistency between the applications like SPARQL query engines and the RDF semantics.

Sometimes, the skolemization mechanism from logic perhaps help to give a constant name to the blanknodes.

Blank nodes do not have an intrinsic name in the RDF abstract syntax. In situations where such a name is required, implementations may systematically replace blank nodes in an RDF graph with some constant symbols. Systems wishing to do this should use a Skolem function employed in the field of logics to generate a constant symbol for each blank node in an RDF graph. We call this the skolemization constant.

The use of Skolemization in the definition of some entailment regime (e.g. simple entailment in RDF semantics) makes blank nodes explicit described. For example, the subgraph G bellows can be transformed to G' with the skomemization mechanism:

G:	G':
ex:a ex:b _:c.	ex:a ex:b skol:c.
_:d ex:e ex:f	skol:d ex:e ex:f

Figure 8. Skolemization of blank nodes

Since we skolemize the blank nodes after leanizing the RDF graph, the result graph involved the skolemized subgraph like G' does not have inner redundancy. And the skolemization constants can be regarded as local identifies of some resources. The meaning of the skolemization is to transform the variables (the blank nodes) to the constants respectively which helps the reasoning on the RDF graphs.

D. Transform Blank Nodes into URI References

The most suitable method may be to give each blank node a URI reference. One of the rules of publishing linked data is to identify your resource with a URI reference, which makes others link to the resources you have published, and thus the Web of data does. Systems may wish to mint some Skolem URI references in such a way that they can recognize the URI references as having been introduced solely to replace a blank node, and map back to the source blank node where possible. Also, systems which want the Skolem URI references to be recognizable outside of the system boundaries should use a well-known URI/IRI with a registered name. The common manner is to find out the primary key information of the entity which can identify it uniquely, like the email address, student's id and the post code etc., and then give a URI reference which has the primary key information as its component. For example, if you identify a student entity, you may use the "http://example/students/id/2009070503" as the URI reference; if you identify an address, you may use the "http://example/city/zip-code/210098" as another URI reference. Here the components "2009070503" is the student id and the "210098" zip code respectively. The principles about how to choose the URI references is that the URI references you use can 1) identify the entity uniquely; 2) express the identity information of the entity; and, 3) exist formally and permanently.

V. CONCLUSION

In this paper, we investigated the meaning and the drawback of blank nodes in RDF, specially, we compared the semantics of blank nodes in RDF semantics and SPARQL semantics, and showed a mismatch between the semantics of blank nodes in RDF and SPARQL. We also concluded the conveniences and the inconveniences of blank nodes when using them and gave a detailed analysis. We proposed that a leanization pre-process should be excuted when we pose some operations like SPARQL queries over the RDF graphs, and we got a conclusion that the triples with blank nodes as their components which represent the redundant information is the main course of the inconsistency when we evaluate SPARQL queries over some RDF graphs. Although we believed that blank nodes in RDF will still play an important role, we still discourage the use of blank nodes because of the impossible access to some RDF data through RDF links; Merging data from different sources also becomes difficult when blank nodes are involved in the graphs to be merged, so we must harmonize the usages and the shortcomings of the blank nodes, in other words, some acknowledged methods to reduce the blank nodes and transform the blank nodes to some URI references are needed.

REFERENCES

[1] Graham Klyne, Jeremy J. Carroll and Brian. McBridge, "Resource Description Framework (RDF): concepts and abstract syntax", http://www.w3.org/TR/rdf-concepts/, February 2004.

[2] Chris Bizer, Richard Cyganiak and Tom Heath, "How to publish linked data on the Web", http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/, July 2007.

[3] Eric Prud'hommeaux and Andy Seaborne, "SPARQL query language for RDF", http://www.w3.org/TR/rdf-sparql-query/, January, 2008.

[4] Enrico Franconi and Sergio Tessaris "The Semantics of SPARQL" http://www.inf.unibz.it/krdb/w3c/sparql/, 2005.

[5] Patrick Hayes and Brian McBride. RDF Semantics. http://www.w3.org/TR/rdf-mt/, February 2004.

[6] M Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler and Ian Horrocks et.al, "OWL Web Ontology Language Reference" http://www.w3.org/TR/owl-ref/, February 2004.

[7] Jos de Bruijn and Stijn Heymans, "Logic foundations of RDFS(S) with datatypes", Journal of Artificial Intelligence Research, vol.38, Issue 1, pp. 535-568, May 2010.

[8] Pascal Hitzler, Markus Krotzsch, Bijan Parsia, Peter F. Patel-Schneider and Sebastian Rudolph, "OWL 2 Web Ontology Language Primer", http://www.w3.org/TR/owl2-primer/, October 2009.

[9] Frank Manola, Eric Miller and Brian McBride, "RDF Primer", http://www.w3.org/TR/rdf-syntax/, February 2004.

[10] Steve Harris, Andy Seaborne, Eric Prud'hommeaux, "SPARQL query language", http://www.w3.org/TR/sparql11-query/, May 2011.

[11] Birte Glimm, Chimezie Ogbuji, Sandro Hawke, Ivan Herman, Bijan Parsia and Axel Polleres, et.al, "SPARQL 1.1 entailment regimes", http://www.w3.org/TR/2011/WD-sparql11-entailment-20110512/, May 2011.

[12] Jean-Francois Baget, "RDF entailment as graph homomorphism", Lecture Notes in Computer Science, Volume 3729, 2005, pp.82-96.

[13] Herman J. and ter Horst, "Extending the RDFS entailment lemma", Lecture Notes in Computer Science, Volume 3298, 2004, pp.77-91.

[14] Reinhard Pichler, Axel Polleres, Sebastian Skritek and Stefan Woltran, "dRDF: entailment for domain-restricted RDF", Lecture Notes in Computer Science, Volume 5021, 2008, pp.200-214.

[15] Reinhard Pichler, Axel Polleres, Sebastian Skritek and Stefan Woltran, "Redundancy elimination on RDF graphs in the presence of rules, constraints, and queries", Lecture Notes in Computer Science, Volume 6333, 2010, pp.133-148.

[16] Ian Hrrocks, Peter F., Patel-Schneider, "Reducing OWL entailment to description logic satisfiability", Lecture Notes in Computer Science, Volume 2870, 2003, pp.17-29.

[17] Jos de Bruijn and Stijn Heymans, "Logical Foundations of (e)RDF(S): complexity and reasoning", Lecture Notes in Computer Science, Volume 4825, 2007, pp.86-99.

[18] Herman Horst, "Semantic Web Ontologies and entailment_ complexity aspect", Intelligent Algorithms in Ambient and Biomedical Computing, 2006, Part III, pp.215-242.

[19] Giovambattsta Lanni, Thomas Krennwallner, Alessandra Martello and Axel Pollers, "Dynamic querying of mass-storage RDF data with rule-based entailment regimes", Lecture Notes in Computer Science, Volume 5823, 2009, pp. 310-327.

[20] Georgios Meditskos and Nick Bassiliages, "Combining a DL reasoner and a rule engine for improving entailment-based OWL reasoning", Lecture Notes in Computer Science, Volume 5318, 2008, pp. 277-292.

[21] Dean Allemang and James Hendler, "Semantic Web for the Working Ontologist", Elsevier (Singapore) Pte Ltd, 2009.

[22] T. Berners-Lee, W. Hall, J. Hendler, K. O'Hara, N. Shadbolt and D. J. Weitzner. "A Framework for Web Science", Foundations and Trends in Web Science, Volume 1, No 1, 2006.

[23] A. Borgida, "On the Relationship between Description Logic and Predicate Logic Queries", Proceedings if the Third International Conference on Information and Knowledge Management, 1994

Lei Chen, born in 1980, Ph.D. candidate in college of computer and information, Hohai University. He is also a lecture in the Dept. of computer and information engineering of HuaiNan normal university. His research interests focus on Semantic Web, databases, and ontology engineering.

Haifei Zhang, born in 1980, Ph.D. candidate in college of computer and information, Hohai University. His research interests focus on logics, GIS, and databases.

Ying Chen, born in 1980, PhD. Candidate in college of computer and information, Hohai University. His research interests focus on data mining and databases.

Wenping Guo, born in 1978, Ph. D. candidate in college of computer and information, Hohai University. His research interests focus on ontology annotation, network security, and Semantic Web.