Bipartite Graphs as Intermediate Model for RDF

Jonathan Hayes[1,2] and Claudio Gutierrez[1]

[1] Dept. of Computer Science, Universidad de Chile
[2] Dept. of Computer Science, Technische Universität Darmstadt, Germany
{jhayes,cgutierr}@dcc.uchile.cl

Abstract. RDF Graphs are sets of assertions in the form of subject-predicate-object triples of information resources. Although for simple examples they can be understood intuitively as directed labeled graphs, this representation does not scale well for more complex cases, particularly regarding the central notion of connectivity of resources.
We argue in this paper that there is need for an intermediate representation of RDF to enable the application of well-established methods from Graph Theory. We introduce the concept of Bipartite Statement-Value Graph and show its advantages as intermediate model between the abstract triple syntax and data structures used by applications. In the light of this model we explore issues like transformation costs, data/schema-structure, the notion of connectivity, and database mappings.

Keywords: RDF Model, RDF Graph, RDF Databases, Bipartite Graph

1 Introduction

The World Wide Web was originally built for human consumption, and although everything on it is machine-readable, the data is not machine-understandable [LS99]. The Resource Description Framework, RDF [MSB04], is a language to express metadata about information resources on the Web proposed by the WWW Consortium (W3C). It is intended that this information is suitable for processing by applications and thus is the foundation of the Semantic Web [BL98]. RDF statements are triples consisting of a subject, a predicate and an object. The subject is the resource being described, the predicate is some kind of property and the object is a property value. A set of RDF triples is called a *RDF Graph*, a term formally introduced by the RDF documentation [KC04] and motivated by the underlying "graph data model".

The graph-like nature of RDF is indeed intuitively appealing, but a naive formalization of this notion presents problems. Currently, the RDF specification documents do not distinguish clearly among the term "RDF Graph", the mathematical concept of graph, and the graph-like visualization of RDF data. The definition provided in the *RDF Concepts and Abstract Syntax* document [KC04] can be understood as a representation scheme of RDF Graphs by means of directed labeled graphs (see an example in figure 1). This notion is used extensively

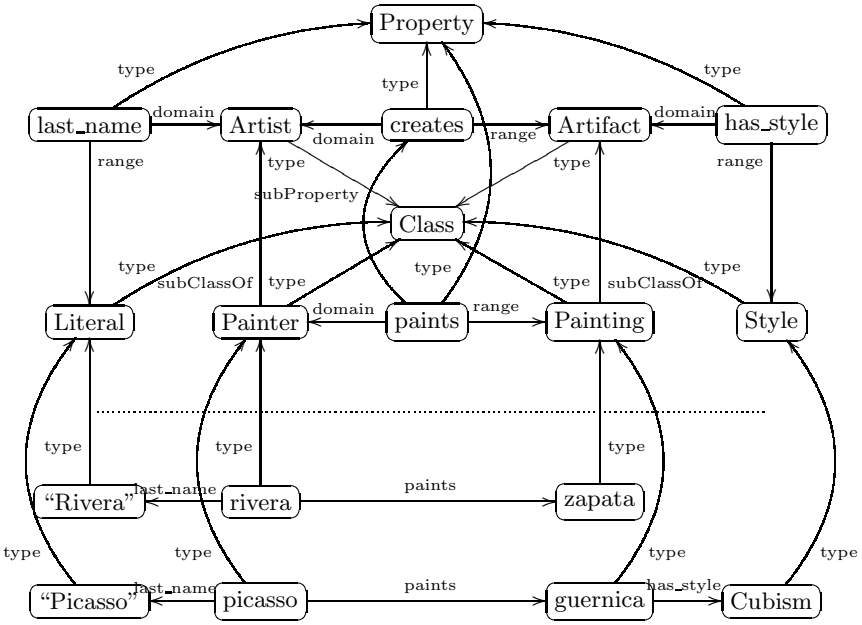

Fig. 1: The museum example. A non-standard graph where edge labels and nodes can represent the same object. For example, *paints* occurs as a node as well as arbitrarily often in the role of edge labels

throughout the RDF documentation, especially for the visualization of simple examples. This representation, based on the idea of representing a triple (a, b, c) by $a \xrightarrow{b} c$, produces labeled graphs where resources possibly can occur as edge labels as well as node labels. This is inconvenient from several points of view. Allowing such multiple occurrences of resources jeopardizes one of the most important aspects of graph visualization, which is the implicit assumption that the complete information regarding a node in a graph is obtained by its place in the drawing and its incident edges. Nevertheless, the essential drawback of the representation mentioned above is the fact that it is not a standard mathematical model to which we can apply well-established techniques. In fact, when reasoning formally over RDF data, e.g. as described in the RDF Semantics document [Hay04], one has to operate with sets of triples. Although well-defined formally, a set of triples is a model that due to multiple occurrences of the same resource in the data structure leads to undesirable redundancies and does not capture the graph-like nature of RDF data, particularly regarding connectivity of resources.

We propose to model RDF Graphs as bipartite graphs. RDF Graphs are naturally hypergraphs, and hypergraphs are bipartite graphs. (Bipartite) graphs are well known mathematical objects which, as formal representation, have several advantages over the triple representation or the directed labeled graph represen-

tation discussed above. Among these advantages are: algorithms for the visualization of data for humans [dBETT94,Mäk90], a formal framework to prove properties and specify algorithms, libraries with generic implementations of graph algorithms, and of course, techniques and results of graph theory. Representing RDF data by standard graphs could have several other advantages by reducing application demands to well-studied problems from graph theory. A few examples at hand: Difference between RDF Graphs: When are two RDF Graphs the same? [BL01,Car01] Entailment: Determining entailment between RDF Graphs can be reduced to graph mappings: Is graph A isomorphic to a subgraph of graph B? [Hay04]. Minimization: Finding a minimal representation of a RDF Graph is important for compact storage and update in databases [GHM04]. Semantic relation between information resources: metrics and algorithms for semantic distance in graphs [AMHAS03,RE03]. Clustering [CFLZ03,ZHD+01] and graph pattern mining algorithms [VGS02] to reveal regularities in RDF data.

Contributions. In this paper we provide a formal graph-based intermediate model of RDF, which intends to be more concrete than the abstract RDF model to allow the exploit of results from graph theory, but still general enough to allow specific implementations. The contributions are the following: 1. We present a class of bipartite graphs representing an intermediate model for RDF. 2. We study properties of this class of graph and the transformation of the mapping of RDF data into them and vice versa. 3. We explore formalizations of the intuitive notion of "semantic relation" between resources in RDF specifications and study the structure of a RDF specification in terms of its schema and its raw data. 4. We discuss how these notions can be applied by looking at current storage and retrieval systems for RDF.

Related Work. There is little work on formalization of the RDF Graph model besides the guidelines given in the official documents of the W3C, particularly *RDF Concepts and Abstract Syntax* [KC04] and *RDF Semantics* [Hay04]. There are works about algorithms on different problems on RDF Graphs, among them T. Berners-Lee's discussion of the Diff problem [BL01] and J. Carroll's study of the RDF Graph Matching Problem [Car01]. Although not directly related to graph issues, there is work on the formalization of the RDF model itself that touches our topic: a logical approach that gives identities to statements and so incorporates them to the universe [YK02], a study oriented to querying that gives a formal typing to the model [KAC+02] and results on normalization of RDF Graphs [GHM04]. Recently, in the field of RDF storage and querying the graph nature of RDF has gained interest. We survey this area in section 5.

2 Preliminaries

RDF. The atomic structure of the RDF language is the statement. It is a triple, consisting of a subject, a predicate and an object. These elements of a triple can be *URIs* (Uniform resource Identifiers), representing information resources; *literals*, used to represent values of some datatype; and *blank nodes*, which represent anonymous resources. There are restrictions on the subject and predicate

of a triple: the subject cannot be a literal, and the predicate cannot be a blank node. Resources, blanks and literals are sometimes referred to as *values*.

A *RDF Graph* is a set of RDF triples. Let T be a RDF Graph. Then univ(T), the set of all values occurring in all triples of T, is called the *universe* of T; and vocab(T), the *vocabulary* of T, is the set of all values of the universe that are not blank nodes. The *size* of T is the number of statements it contains and is denoted by $|T|$. With subj(T) (resp. pred(T), obj(T)) we designate all values which occur as subject (resp. predicate, object) of T.

Let V be a set of URIs and literal values. We define RDFG(V) := $\{T \mid T$ is RDF Graph and vocab(T) $\subseteq V\}$, i.e. the set of all RDF Graphs with a vocabulary included in V. There is a distinguished vocabulary, *RDF Schema* [BG04] that may be used to describe properties like attributes of resources (traditional attribute-value pairs), and to represent relationships between resources. It is expressive enough to defines classes and properties that may be used for describing groups of related resources and relationships between resources.

Example RDF Graph 1 The `prefix:suffix` notation abbreviates URIs. The *wos* prefix identifies a "Web of Scientists" vocabulary (*rdfs* is RDF Schema)

```
1: <wos:Ullman> <wos:coauthor> <wos:Aho>
2: <wos:Greibach> <wos:coauthor> <wos:Hopcroft>
3: <wos:coauthor> <rdfs:subPropertyOf> <wos:collaborates>
4: <wos:Greibach> <wos:researches> <wos:topics/formalLanguages>
5: <wos:Valiant> <wos:researches> <wos:topics/formalLanguages>
6: <wos:Erdös> <wos:researches> <wos:topics/graphTheory>
7: <wos:Aho> <wos:collaborates> <wos:Kernighan>
8: <wos:Hopcroft> <wos:coauthor> <wos:Ullman>
```

Graphs. A *graph* is a pair $G = (N, E)$, where N is a set whose elements are called *nodes*, and E is a set of unordered pairs $\{u, v\}$, the *edges* of the graph. Two edges are said to be *incident* if they share a node. Observe that the definition implies that the sets N and E are disjoint. A graph G is a *multigraph* if E is a multiset, thus permitting multiple edges between two nodes. A graph $G = (N, E)$ is said to be *bipartite* if $N = U \cup V, U \cap V = \emptyset$ and for all $\{u, v\} \in E$ it holds that $u \in U$ and $v \in V$. A *directed graph* is a graph where the elements of E are ordered, i.e. $E \subseteq N \times N$.

In order to express more information, a graph can be *labeled*. A graph (N, E), together with a set of labels L_e and an edge labeling function $l_e : E \to L_e$ is an *edge-labeled* graph. A graph is said to be *node-labeled* when there is a node label set and a node labeling function, as above. We will write (N, E, l_n, l_e).

The notions of path and connectivity will be important in what follows. A *path* is a sequence of edges $e_1, \ldots, e_n$ with each edge e_i is incident to e_{i-1}, for $i \in [2, n]$. The *label* of the path is $l_e(e_1) \cdots l_e(e_n)$. Two nodes x, y are *connected* if there exists a path $e_1, \ldots, e_n$ with $x \in e_1$ and $y \in e_n$. The *length* of a path is the number of edges it consists of.

RDF as Directed Labeled Graphs. Now we can formalize the definition of *directed labeled graph* corresponding to an RDF Graph T, as described in [KC04],

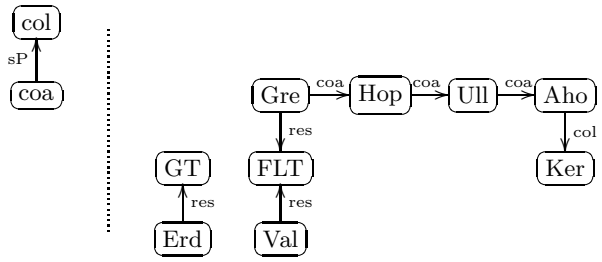

Fig. 2: RDF Graph 1 in page 4 represented by a directed labeled graph. Labels have been abbreviated to their first letters.

as the multigraph (N, E, l_n, l_e), where $N = \{v_x : x \in \text{subj}(T) \cup \text{obj}(T)\}$, and $l_n(v_x) = x$, and $E = \{(s, o) : (s, p, o) \in T\}$, and $l_e(s, o) = p$. Figure 2 presents an example of such a graph. Observe that the set of edge labels and node labels might not be distinct. In the introduction we mentioned the problems that could arise out of this.

$\mathcal{E}$ = { { coauthor, subPropertyOf, collaborates },{ Ullman, coauthor, Aho },{ Greibach, coauthor, Hopcroft } }

V= { collaborates, coauthor, subPropertyOf, Aho, Greibach, Hopcroft, Ullman }

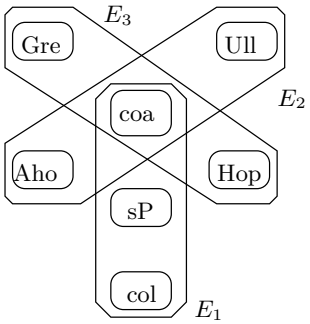

Fig. 3: Example of a simple 3-uniform hypergraph. This hypergraph represents the first three statements of the example on page 4

Hypergraphs. Informally, hypergraphs are systems of sets which extend the notion of graphs allowing edges to connect any number of nodes. For background see [Duc95]. Formally, let $V = \{v_1, \ldots, v_n\}$ be a finite set, the *nodes*. A *hypergraph on* V is a pair $\mathcal{H} = (V, \mathcal{E})$, where $\mathcal{E}$ is a family $\{E_i\}_{i \in I}$ of subsets of V. The members of $\mathcal{E}$ are called *edges*. A hypergraph is *simple* if all edges are distinct. A hypergraph is said to be *r-uniform* if all edges have the cardinality r. A r-uniform hypergraph is said to be *ordered* if the occurrence of nodes in every edge are numbered from 1 to r.

Hypergraphs can be described by binary edge-node *incidence matrices* (as any graph). In this matrix rows correspond to edges, columns to nodes: entry $m_{i,j}$ equals 1 or 0, depending on whether E_i contains node n_j or not. To the incidence matrix of a hypergraph $\mathcal{H} = (V, \mathcal{E})$ corresponds a *bipartite incidence graph* $B = (N_V \cup N_\mathcal{E}, E)$, which is defined as follows. Let N_V be the set of node names of $\mathcal{H}$ which labeled the columns of the matrix, and $N_\mathcal{E}$ the set of edge

names labeling its rows. Then E contains an edge $\{v_i, e_j\}$ for each $v_i \in N_V, e_j \in N_\mathcal{E}$ where the matrix entry $m_{i,j}$ is 1. The obtained graph B can be read to have an edge $\{v, e\}$ exactly when the hypergraph node represented by v is member of the hypergraph edge represented by e. It is evident that B is bipartite. Figure 4 shows the incidence matrix of a hypergraph and the bipartite incidence graph derived from it.

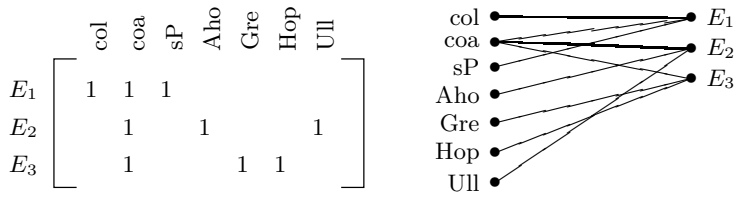

Fig. 4: Incidence matrix representing the hypergraph of Example 3 and the corresponding incidence graph. In the case of an ordered hypergraph, matrix entries will indicate the position of the occurence of the node in the edge

3 Bipartite Statement-Value Graphs

Deriving Bipartite Graphs from Hypergraphs. One of the major problems encountered in trying to model RDF Graphs as classical graphs is the fact that an edge or labeled edge cannot represent the ternary relation given by a RDF triple. Therefore it is natural to turn the attention to graphs with 3-node-connecting edges instead of classical 2-node edges, that is, hypergraphs.

Proposition 1. *Any RDF Graph can be represented by a simple ordered 3-uniform hypergraph. Every RDF triple corresponds to a hypergraph edge, the nodes being the subject, predicate and object in this order. The node set of the hypergraph is the union of all the edges.*

The converse of the proposition also holds when imposing constraints on the occurrences of blank nodes and literals: blank nodes may not be predicates and literals may not serve as subjects or predicates.

As stated in the preliminaries section, hypergraphs can be represented by incidence matrices where membership of a node in a edge is marked with a '1'. In the case of the hypergraph representing a RDF Graph, the nodes of an edge are ordered and we label them by S, P or O to represent the role (Subject, Predicate, or Object) of the information resource. Hence, when deriving the bipartite incidence graph of this incidence matrix, an edge will be added for every S, P, O entry of the matrix, and this edge will be labeled with the corresponding character. The only difference between the graph derived from the incidence matrix of any hypergraph and a RDF Graph hypergraph is the fact that each edge has one of three labels.

Mapping RDF to Bipartite Statement-Value Graphs. This section presents a direct transformation mapping RDF Graphs to bipartite graphs. Let $\mathcal{B}$ be the

set of bipartite labeled graphs $G = (V \cup St, E, l_n, l_e)$, $V \cap St = \emptyset$, where each edge in E connects a node in V with a node in St, and $l_e : E \to L_e$ and $l_n : V \to L_n$ are labeling functions. The elements of V are called the *value nodes* and those of St the *statement nodes*.

Definition 1 (BSVG). *Let T be a RDF Graph. Then we define a map β : RDFG $\to \mathcal{B}$ as follows: $\beta(T) = (V \cup St, E, l_n, l_e) \in \mathcal{B}$, is the* Bipartite Statement-Value Graph (BSVG) representing T, *with $V = \{v_x : x \in \text{univ}(T)\}$; $St = \{st_t : t \in T\}$; and the set of edges E is built as follows: for each triple $t = (x, y, z) \in T$ add the edges $\{st_t, v_x\}$ with label S, $\{st_t, v_y\}$ with label P, and $\{st_t, v_z\}$ with label O. The labeling of the nodes is given by:*

$$l_n(v_x) := \begin{cases} (x, d_x) & \text{if } x \text{ is literal } (d_x \text{ is the datatype identifier of } x) \\ x & \text{else} \end{cases}$$

Note that $\beta(T)$ is a *3-regular* bipartite graph, because the degree of each node in St is 3. This representation incorporates explicitly the statements as nodes in the graph. BSVGs are well-defined and one can go back and forth between T and $\beta(T)$:

Proposition 2. *For each RDF Graph T there is a unique Bipartite Statement-Value Graph $\beta(T)$ representing it, and vice versa. Moreover, there exists a function $\beta^{-1} : \beta(\text{RDFG}) \to \text{RDFG}$ satisfying $\beta^{-1}(\beta(T)) = T$.*

The graph created from T has reasonable size, and can be obtained efficiently:

Proposition 3. *Let T be a RDF Graph and $\beta(T) = (V \cup St, E, l_n, l_e)$. Then:*

1. *$\beta(T)$ can be computed in time $O(|T| \lg |T|)$.*
2. *The graph $\beta(T)$ is bounded as follows: $|St| = |T|$, $|V| = |\text{univ}(T)|$ and $|E| = 3|T|$.*

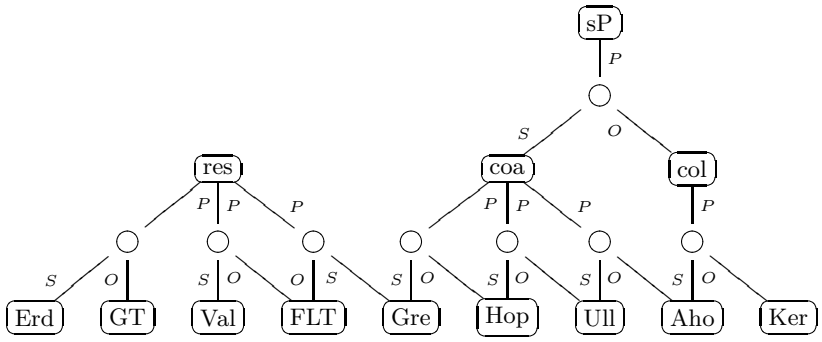

Fig. 5: The Bipartite Statement-Value Graph of the RDF Graph on page 4. Statement nodes are represented by circles and edge labels S, P, O indicate their subject, predicate and object

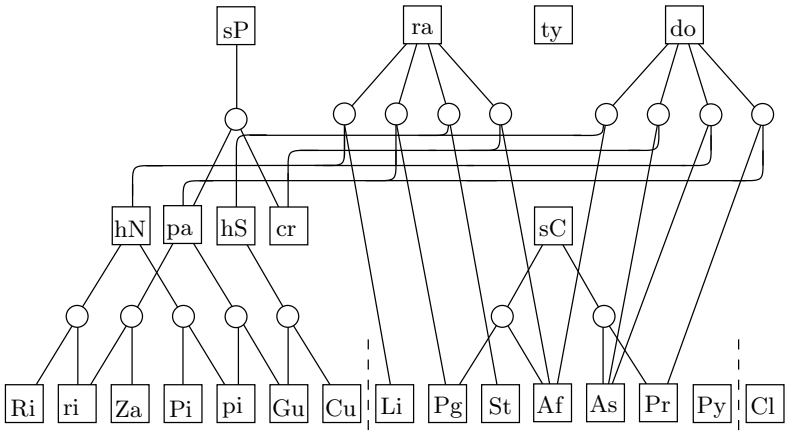

Fig. 6: BSVG of the museum example. Edge labels have been omitted for clarity. Drawing levels indicate the order of the values nodes. At the bottom level are values which never occur as predicates: class instances like (bold letters indicate the abbreviations) the literal "**Ri**vera", **ri**vera (resource), **Za**pata, classes such as **P**ainting, **A**rti**f**act, **A**rti**s**t, **P**ainte**r**, **P**roperty, and the meta-class **Cl**ass. Simple properties include **h**as_**N**ame and **pa**ints, and properties of properties are **s**ub**P**roperty, **do**main, **ra**nge and **ty**pe. Declarations with property "type" are shown in figure 7

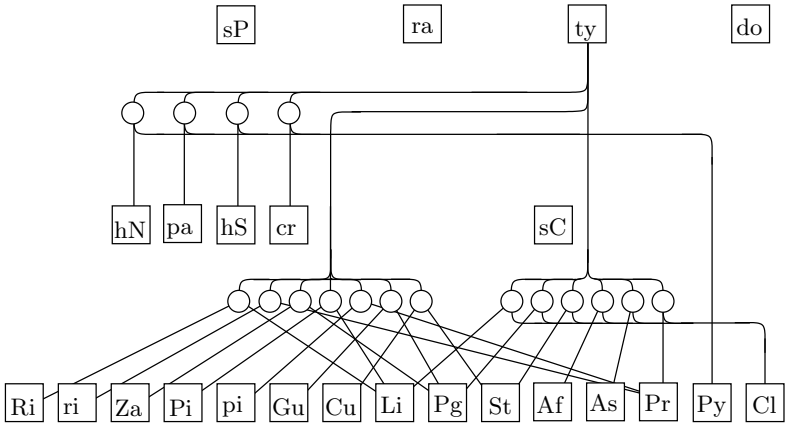

Fig. 7: Type declarations of the BSVG in figure 6

Structure of RDF Graphs. RDF Graphs consist of values and statements. A first coarse-grained division of the statements is the following: those that define a schema, and those that are data structured under this schema (see figure 1 – the dotted line represents this division). Although RDF does not distinguish between schema-defining and data statements, this distinction is natural when considering

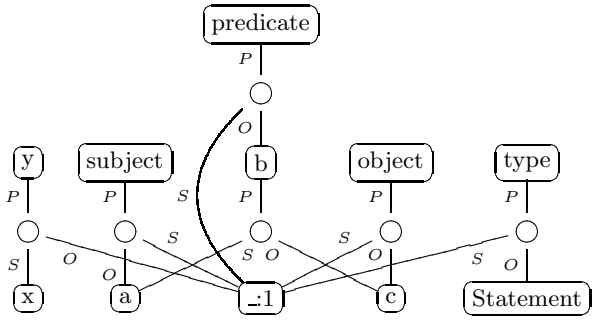

Fig. 8: Stratified drawing of the reification of a statement. A resource x makes a proposition y about the statement (a, b, c)

storage [MAYU03] and querying [KAC+02] in databases. Unfortunately, a plain discrimination between the data and schema parts of a RDF specification is not always possible. Moreover, features like extensibility of specifications and reification make this divide difficult to grasp formally. In the following we present two approaches to this issue.

Definition 2. *A* data subgraph *of a RDF Graph T is a maximal subgraph T' satisfying* $(\mathrm{subj}(T') \cup \mathrm{obj}(T')) \cap \mathrm{pred}(T') = \emptyset$. *The* schema subgraph *associated to T' is $T \setminus T'$.*

Notice that a RDF Graph T does not have a unique data-subgraph, e.g. consider $\{(a, b, c), (b, d, e)\}$ where each statement alone is a data subgraph. Moreover, a RDF Graph could have exponentially many different ones.

An alternative approach is to decompose the RDF Graph T in logical levels.

Definition 3. *Let T be a RDF Graph. Define $V_i(T)$ and $St_j(T)$ by mutual recursion as follows: $V_0(T)$ is the set of values of T that are not predicates, $V_n(T)$ is the set of values of T that are predicates of statements in $St_n(T)$, and $St_{n+1}(T)$ is the set of statements of T whose predicate and object are elements of $\bigcup_{j<n} V_j(T)$.*

The order *of T is the maximum n such that V_{n-1} is not empty.*

T is stratified *if* $\mathrm{univ}(T) = \bigcup_{j \geq 0} V_j(T)$.

Example (Stratification).

- $T = \{(a, a, b)\}$ cannot cannot be stratified. (For example the axiomatic triple `(rdf:type, rdf:type, rdf:Property)`). $V_0(T) = \{b\}$ and $V_j(T) = \emptyset$ for $j > 0$.
- Reification of a triple (a, b, c) is stratified (see figure 8).
- The museum example (figure 1) can be logically partitioned in three levels (see figure 6).

Proposition 4. *T is not stratified iff in $\beta(T)$ there is a cycle from a value node with label $[(S \cup O)P(SO \cup OS)^*]^*$.*

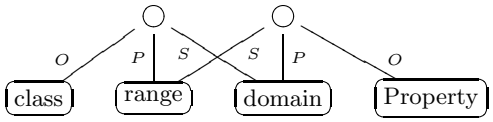

Fig. 9: Example of a non-stratified RDF Graph.

Although the complete axiomatic specification of RDF is not stratified (see figure 9), most RDF data currently found in practice is stratified – if RDF axiomatic triples are not considered – and has small order (no bigger than 3). Also, if T is stratified, the graph obtained by reifying each of its triples is stratified. However, the union of two stratified RDF Graphs is not necessarily stratified.

4 Connectivity of RDF Data

A RDF Graph expresses more information than the sum of the meanings of the individual statements it contains. This section introduces the notion of *connectivity*, which is essential to capture *semantic relations* between resources. The importance of this notion for processing RDF data has been argued in several contexts [GLMB98,KCP,HBEV04,MAYU03].

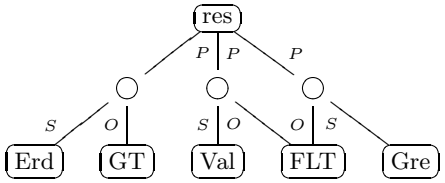

Fig. 10: Representing statements 4-6 of the RDF Graph on page 4

The intuitive notion of "connected resources" in an RDF Graph can be defined as follows: Two resources x and y of a RDF Graph T are *connected* if there exists a sequence of RDF triples $(t_1, t_2, \ldots, t_n)$, $t_k = (s_k, p_k, o_k) \in T$, for which it holds that $x \in \{s_1, p_1, o_1\}$, $y \in \{s_n, p_n, o_n\}$ and for all $i < n$, $\{s_i, p_i, o_i\} \cap \{s_{i+1}, p_{i+1}, o_{i+1}\} \neq \emptyset$. This notion corresponds precisely to the well established notion of connectivity in graphs:

Proposition 5. *Let T be a RDF Graph. The resources x, y are connected in T iff there exists a path in $\beta(T)$ between the corresponding nodes v_x and v_y.*

Example. Consider figure 10: Greibach and Valiant are connected through a path showing their common study of formal language theory (`FLT`). Nevertheless, there are also irrelevant relations resulting out of the existence of paths: Erdös and Greibach are connected by a path via the node `res`.

As we see, the above notion is of limited use: Not all paths between two resources are "meaningful" The next definition attempts to capture such paths.

Definition 4. *Let T be a RDF Graph. Given two resources u, v, there is an* horizontal path *between them if there is a path in $\beta(T)$ from u to v whose label does not contain P. The path is* oriented *if its label is in $(SO)^*$. The path is* cycle-free *if it does not contain cycles.*

If we compare to relational databases, the notion of horizontal path corresponds roughly to values of tuples linked via joins. Paths which contain predicate labels, i.e. "vertical" paths, are paths passing through the schema of the database, giving usually less relevant relationships unless they have special properties, like transitivity. For example, the two RDF Schema transitive predicates, `subPropertyOf` and `subClassOf`, give rise to "vertical" paths.

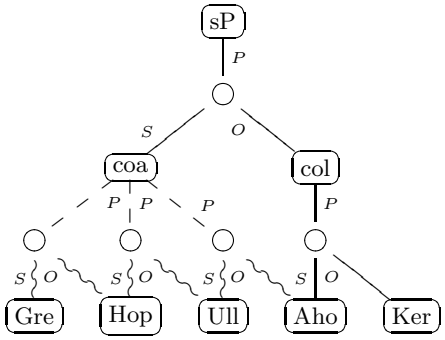

Fig. 11: Example for the transitive closure computation for coauthorship

Example. Sometimes it is desirable to query for paths of arbitrary length – examples include the "citation tree" to measure the influence of a publication or people related by coauthorship to each other (e.g. Erdös number computation). When statements (s, p, o) formulate that (s, o) are instances of the relation p, this amounts to computing the transitive closure of p. Figure 11 depicts a simple example showing a computation of the transitive closure of coauthorship. Consider the node *coauthor*, find its P-neighbors (dashed lines), and look for $(SO)^*$ paths among the children (curly lines).

5 RDF Graphs and Databases

RDF database technology is very recent from almost every point of view. The main challenges it faces are the features not present in traditional databases. Examples of this are a variable schema, the type discipline, restrictions on class inheritance and type membership, etc. Nevertheless, the main issue is probably the graph-like flavor of data and queries: connectivity, paths, distance, aggregate information like degree (of a node). Additional important issues to consider are the support for data (graph) mining and processing.

Today there are several systems for storing and querying RDF (see [PG04] and [HBEV04]). Also there are systems that have several database features, e.g. transactional systems like Kowari [3]. The majority of them –for which documen-

[3] *kowari:metastore*, `http://www.kowari.org/`

tation is available– use mappings to the relational model plus implementation decisions to improve the performance. To illustrate the point, we will briefly describe two of the most popular and best documented, Jena and Sesame.

Jena and Sesame. Jena's original design (Jena-1) used two alternatives approaches to store an RDF Graph: (1) Three tables: one for statements, one for literals and one for resources; The main problem was the heavy use of joins to answer queries. (2) One statement table, with indexes by subject, by predicate and by object. Experience with Jena-1 lead to a more simple schema. Essentially, in Jena-2 [WSKR03] triples are stored into a statement table. Space consumption is optimized using compression of common URI prefixes and pointer to long strings. Also, there is a special treatment of *common statement patterns* (CSP). Such patterns are the result of high-level RDF constructs such as bags, sequences or reifications, and patterns induced from regularities in the user data.

One weakness of the previous system is that it does not automatically incorporates the semantics of RDFS vocabulary. Sesame [BKvH02] concentrates on RDFS-awareness. It uses the SeRQL query language, which incorporates transparently the RDFS vocabulary. The concrete data storage is implemented differently according to the underlying DBMS. For example, for PostgreSQL, which support class hierarchies, the RDF data is stored by property and by subject/object-value into a hierarchy of tables. For MySQL, RDFS information is stored in separate tables. All RDF data is stored into a RDF Statement table.

Storing RDF as Explicit Graphs. Analyzing the mapping of Jena and Sesame in the light of our model, on can conclude that these implementations are essentially direct storing of the BSVG of the corresponding RDF data. Version (1) of Jena stores in one table V and in other St. In version (2), the indexes correspond to group all edges labeled S, all edges labeled P, and all edges labeled O.

We claim that one could refine these models by taking advantage of the structure of the BSVG graph. Moreover, due to the graph-like nature of most queries, the optimization process of queries should be done using the well established techniques from graph theory for paths and connectivity. On the same lines, other graph theoretic problems, like graph pattern mining are important for storage techniques, e.g. CSP in Jena [WSKD03]. Here bipartite Clustering such as yahoo [CFLZ03] could be of much use. Similarly, efficient database indexing through pattern mining [GW97] and improving database storage through pattern mining [DFS99].

We argued that a RDF Graph's semantic relies very much on the connectivity of the resources described, which are not taken into account in a mere triple storage. One of the main features of a query language for RDF was expressed in the classic position paper [GLMB98] by saying that it should be based on the simple mechanism of subgraph matching, being the "simple graph traversal" one of its key features. In the database literature has been studied the problem of storing graphs in databases to be able to answer efficiently typical path-like and graph-like queries [Güt94], as well as graph traversal query language extensions [MS90]. With the emergence of unstructured data and XML, their

tree-like nature of data gave new impetus to this line of research, for example, query languages with path expressions [MW89].

6 Conclusions

We introduced a representation of RDF Graphs in terms of classical bipartite graphs which highlights the graph-nature of RDF, and permits the direct application of graph libraries. We presented preliminary results about the structure and lines of development of the model. We argued the advantages of the model compared to the triple representation and to the directed labeled graph representation used currently by default.

We are using the model to approach diverse algorithmic problems of RDF databases, particularly graph-like notions in querying and storage. Future work includes refinement of the BSVG model and aspects of visualization.

References

[AMHAS03] Boanerges Aleman-Meza, Chris Halaschek, Ismailcem Budak Arpinar, and Amit P. Sheth. Context-Aware Semantic Association Ranking. In I. F. Cruz, V. Kashyap, S. Decker, and R. Eckstein, editors, *Proceedings of SWDB'03*, 2003.

[BG04] Dan Brickley and R.V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation.* World Wide Web, `http://www.w3.org/TR/2003/PR-rdf-schema-20040210/`, 10 February 2004.

[BKvH02] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In I. Horrocks and J. A. Hendler, editors, *The Semantic Web - ISWC 2002, Sardinia, Italy, Proceedings*, volume 2342 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2002.

[BL98] Tim Berners-Lee. *Semantic Web Road map.* World Wide Web, `http://www.w3.org/DesignIssues/Semantic.html`, September 1998.

[BL01] Tim Berners-Lee. *The RDF-DIFF Problem.* World Wide Web, `http://www.w3.org/DesignIssues/Diff`, 2001.

[Car01] Jeremy J. Carroll. *Matching RDF Graphs.* World Wide Web, `http://www.hpl.hp.com/techreports/2001/HPL-2001-293.html`, 2001.

[CFLZ03] John Joseph M. Carrasco, Dan Fain, Kevin Lang, and Leonid Zhukov. *Clustering of Bipartite Advertiser-Keyword Graph.* Overture Research Technical Report, `http://labs.yahoo.com/publications/17.pdf`, 2003.

[dBETT94] Giuseppe di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Algorithms for Drawing Graphs: An Annotated Bibliography. *Comput. Geom. Theory Appl.*, 4:235–282, 1994. Zu finden auch unter `http://www.cs.brown.edu/people/rt/gd-biblio.html`.

[DFS99] Alin Deutsch, Mary F. Fernandez, and Dan Suciu. Storing Semistructured Data with STORED. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 431–442. ACM Press, 1999.

[Duc95] Pierre Duchet. Hypergraphs. In R.L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, pages 381–432. Elsevier Science B.V., Amsterdam, 1995.

[GHM04] C. Gutierrez, C. Hurtado, and A. O. Mendelzon. Foundations of Semantic Web Databases. In *ACM Symposium on Principles of Database Systems (PODS)*, 2004.

[GLMB98] R. V. Guha, Ora Lassila, Eric Miller, and Dan Brickley. *Enabling Inferencing*. World Wide Web, `http://www.w3.org/TandS/QL/QL98/pp/enabling.html`, 1998.

[Güt94] Ralf Hartmut Güting. GraphDB: Modeling and Querying Graphs in Databases. In J.B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of VLDB'94*, pages 297–308. Morgan Kaufmann, 1994.

[GW97] Roy Goldman and Jennifer Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, editors, *VLDB'97, Proceedings*, pages 436–445. Morgan Kaufmann, 1997.

[Hay04] Patrick Hayes. *RDF Semantics. W3C Recommendation*. World Wide Web, `http://www.w3.org/TR/2003/PR-rdf-mt-20040210/`, 10 February 2004.

[HBEV04] Peter Haase, Jeen Broekstra, Andreas Eberhart, and Raphael Volz. *A Comparison of RDF Query Languages*. World Wide Web, `http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/rdfquery.pdf`, 2004.

[KAC+02] Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. RQL: A Declarative Query Language for RDF. In *Proceedings of 2002 WWW conference*. ACM Press, 2002.

[KC04] Graham Klyne and Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation*. World Wide Web, `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`, 10 February 2004.

[KCP] Greg Karvounarakis, Vassilis Christophides, and Dimitris Plexousakis. Querying Semistructured (Meta)Data and Schemas on the Web: The case of RDF and RDFS.

[LS99] Ora Lassila and Ralph R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation*. World Wide Web, `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222`, 1999.

[MAYU03] Akiyoshi Matono, Toshiyuki Amagasa, Masatoshi Yoshikawa, and Shunsuke Uemura. An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays. In I. F. Cruz, V. Kashyap, S. Decker, and R. Eckstein, editors, *Proceedings of SWDB'03*, pages 151–168, 2003.

[MS90] Michael V. Mannino and Leonard D. Shapiro. Extensions to Query Languages for Graph Traversal Problems. *IEEE Trans. Knowl. Data Eng.*, 2(3):353–363, 1990.

[MSB04] Eric Miller, Ralph Swick, and Dan Brickley. *Resource Description Framework (RDF) / W3C Semantic Web Activity*. World Wide Web, `http://www.w3.org/RDF/`, 2004.

[MW89] A. O. Mendelzon and P. T. Wood. Finding Regular Simple Paths in Graph Databases. In *Proceedings of the 15th Conference on Very Large Databases, Morgan Kaufman pubs. (Los Altos CA), Amsterdam*, 1989.

[Mäk90] Erkki Mäkinen. How to Draw a Hypergraph. *Intern. J. Computer Math.*, 34:177–185, 1990.

[PG04] E. Prud'hommeaux and B. Grosof. *RDF Query and Rules: A Framework and Survey.* World Wide Web, `http://www.w3.org/2001/11/13-RDF-Query-Rules/`, 2004.

[RE03] A. Rodríguez and M. Egenhofer. Determining Semantic Similarity Among Entity Classes from Different Ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):442–456, 2003.

[VGS02] N. Vanetik, Ehud Gudes, and Solomon Eyal Shimony. Computing Frequent Graph Patterns from Semistructured Data. In *Proceedings of ICDM 2002, Maebashi City, Japan*. IEEE Computer Society, 2002.

[WSKD03] Kevin Wilkinson, Craig Sayers, Harumi Kuno, and Luping Ding. *Application-Specific Schema Design for Storing Large RDF Datasets.* World Wide Web, `http://www.hpl.hp.com/techreports/2003/HPL-2003-170.html`, 2003.

[WSKR03] Kevin Wilkinson, Craig Sayers, Harumi Kuno, and Dave Reynolds. Efficient RDF Storage and Retrieval in Jena2. In I. F. Cruz, V. Kashyap, S. Decker, and R. Eckstein, editors, *Proceedings of SWDB'03*, 2003.

[YK02] G. Yang and M. Kifer. On the Semantics of Anonymous Identity and Reification. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE, Irvine, USA, Proceedings*, volume 2519 of *Lecture Notes in Computer Science*. Springer, 2002.

[ZHD+01] Hongyuan Zha, Xiaofeng He, Chris H. Q. Ding, Ming Gu, and Horst D. Simon. Bipartite Graph Partitioning and Data Clustering. In *Proceedings of the 2001 ACM CIKM, Atlanta, USA*, pages 25–32. ACM, 2001.