Grammar-Based Geodesics in Semantic Networks

Marko A. Rodriguez[*]
Digital Library Research and Prototyping Team
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Jennifer H. Watkins[†]
International and Applied Technology
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

(Dated: June 26, 2007)

In graph theory, a geodesic is the shortest path between two vertices in a connected network. The geodesic is the kernel of various network metrics including radius, diameter, eccentricity, closeness, and betweenness. These metrics are the foundation of much social network research and thus have been studied extensively in the domain of single-relational, or unlabeled, networks (both in their directed and undirected forms). However, geodesics for unlabeled networks do not translate directly to multi-relational, or semantic networks, where vertices are connected to one another by any number of edge labels. Here, a more sophisticated method for calculating a geodesic is necessary. This article presents a grammar-based technique for calculating geodesics in semantic networks. A grammar is a user-defined abstract representation of a semantic path that respects the ontological classes of a particular semantic network. A discrete "walker" utilizes the grammar to determine which paths to include in its geodesic calculation. The grammar-based model forms a general framework for studying geodesic metrics in semantic networks.

I. INTRODUCTION

The study of networks (i.e. graph theory) is the study of the relationship between vertices (i.e. nodes) as defined by the edges (i.e. arcs) connecting them. In path analysis algorithms, a path metric function maps an ordered vertex pair into a real number, where that real number is the length of the path connecting to the two vertices. Metrics that utilize the shortest path between two vertices in their calculation are called geodesic metrics. The geodesic metrics that will be reviewed in this article are shortest path, eccentricity [15], radius, diameter, betweenness centrality [14], and closeness centrality [3].

If G^1 is a single-relational network, then $G^1 = (V, E)$, where $V = \{i, \ldots, j\}$, is the set of vertices and $E \subseteq V \times V$ is a subset of the product of V. In a single-relational network, the edges have a single meaning, or semantic. While a single-relational network supports the representation of a homogeneous set of relationships, a semantic network supports the representation of a heterogeneous set of relationships. For instance, in a single-relational network it is possible to represent humans connected to one another by friendship edges; in a semantic network, it is possible to represent humans connected to one another by friendship, kinship, collaboration, communication, etc. relationships.

A semantic network denoted G^n can be defined as a set of single-relational networks such that $G^n = (V, \mathbb{E})$, where $\mathbb{E} = \{E_0, E_1, \ldots, E_n\}$ and for any $E_n \in \mathbb{E}$, $E_n \subseteq V \times V$ [8]. The meaning, or semantic, of a relationship in G^n is determined by its set $E_n \in \mathbb{E}$. Perhaps a more convenient semantic network representation and the one to be used throughout the remainder of this article is that of the triple list where $G^n \subseteq (V \times \Omega \times V)$ and Ω is a set of edge labels. A single edge in this representation is denoted by a triple $\tau = \langle i, \omega, j \rangle$, where vertex i is connected to vertex j by the semantic ω.

In some cases, it is possible to isolate sub-networks of a semantic network and represent the isolated network in an unlabeled form. Unlabeled geodesic metrics can be used to compute on the isolated component. However, in many cases, the complexity of the path description does not support an unlabeled representation. These scenarios require "semantically aware" geodesic metrics that respect a semantic network's ontology (i.e. the vertex classes and edge types) [5, 23].

The semantic network is not simply a directed labeled network; it is a high-level representation of complex objects and their relationship to one another according to defined ontological constraints. Thus, to determine a semantic path between vertex i and j such that it passes through some k for which k is a male student that has worked with an old colleague of i, no such algorithm currently exists. While there exist various algorithms to study semantically typed paths in a network [1, 2, 18, 26, 28], each algorithm assumes only a path between two vertices and does not investigate other features of the intervening vertices. The benefit of the grammar-based geodesic model presented in this article is that complex paths can be represented that include

[*]Electronic address: marko@lanl.gov
[†]Electronic address: jhw@lanl.gov

path "bookkeeping" which investigates intervening vertices even though they may not be included in the final path solution. For example, it may be important to determine a set of "friendship" paths between two human vertices, but in doing so, every intervening human must work for some particular organization and furthermore, must have a particular position in that organization. While a set of friendship paths is the result of the function, the path detours to determine employer and position are not. The technique for doing this is the primary contribution of this article.

A secondary contribution is the unification of the grammar-based model proposed here with the grammar-based model proposed in [22] for calculating stationary probability distributions in a subset of the full semantic network (e.g. eigenvector centrality [6] and PageRank [10]). With the grammar-based model, a single framework exists that ports many of the popular unlabeled network analysis algorithms into the semantic network domain.

The third contribution of this article is the presentation of these ideas within the context of the large-scale semantic network data model called the Resource Description Framework (RDF). There is an increasing need to represent multi-relational data and furthermore, to analyze it. By presenting the concepts of this article from this technological standpoint, it is hoped that social network researchers will get a fundamental understanding of the benefit of this modeling domain that can support on the order of 10^9 edge semantic networks.

The first half of this article will define a popular set of geodesic metrics for unlabeled single-relational networks. It will become apparent from these definitions, that the more advanced metrics rely on the shortest path metric. The second half of the article will present the grammar-based model for calculating a semantically meaningful shortest path in a semantic network. The other geodesics follow from this definition.

II. GEODESICS IN SINGLE RELATIONAL NETWORKS

This section will review a collection of popular geodesic metrics used to characterize a vertex-to-vertex relationship, a vertex, and a network. The following list enumerates these metrics and identifies whether they are path, vertex, or network metrics:

- in- and out-degree: vertex metric
- shortest path: path metric
- eccentricity: vertex metric
- radius: network metric
- diameter: network metric
- closeness: vertex metric
- betweenness: vertex metric.

It is worth noting that besides in- and out-degree, all the metrics mentioned utilize a path function $\rho : V \times V \rightarrow Q$ to determine the set of paths between any two vertices in V. The premise of this article is that once a path function is defined for a semantic network, then all of the other metrics are directly derived from it. In the semantic network path function, $\rho : V \times V \times \Psi \rightarrow Q$ returns the number of paths between two vertices according to a grammar Ψ.

Before discussing the grammar-based geodesic model for semantic networks, this section will review the geodesic metrics in the domain of single-relational networks.

A. In- and Out-Degree

The simplest structural metric for a vertex is the vertex's degree. While this is not a geodesic metric *per se*, it is presented as the concept will become necessary in the later section regarding semantic networks.

For directed networks, any vertex $i \in V$ has both an in-degree and an out-degree. The set of edges in E that have i as either its in- or out-edge is denoted $\Gamma^- : V \rightarrow E$ and $\Gamma^+ : V \rightarrow E$, respectively. If

$$\Gamma^-(i) = \{(x, y) \mid (x, y) \in E \ \wedge \ y = i\}$$

and

$$\Gamma^+(i) = \{(x, y) \mid (x, y) \in E \ \wedge \ x = i\}$$

then, $\Gamma^-(i)$ is the subset of edges in E incoming to i and $\Gamma^+(i)$ is the subset of edges outgoing from i. The cardinality of the sets is the in- and out-degree of the vertex, denoted $|\Gamma^-(i)|$ and $|\Gamma^+(i)|$, respectively.

B. Shortest Path

The shortest path metric is the foundation for all other geodesic metrics. This metric is defined for any two vertices $i, j \in V$ such that the sink vertex j is reachable from the source vertex i in G^1 [13]. If j is unreachable from i, the shortest path between i and j is undefined. The shortest path between any two vertices i and j in an unweighted network is the smallest of the set of all paths between i and j. If $\rho : V \times V \rightarrow Q$ is a function that takes two vertices and returns a set of paths Q where for any $q \in Q$, $q = (i, \ldots, j)$, then the shortest path between i and j is the $min(\bigcup_{q \in Q} |q| - 1)$, where min returns the smallest value of its domain. The shortest path function is denoted $s : V \times V \rightarrow \mathbb{N}$ with the function rule

$$s(i, j) = min\left(\bigcup_{q \in \rho(i,j)} |q| - 1\right).$$

It is important to subtract 1 from the path length since a path is defined as the set of edges traversed, not the set of vertices traversed. Thus, for the path $q = (a, b, c, d)$, the $|q|$ is 4, but the path length is 3.

Note that p returns the set of all paths between i and j. Of course, with the potential for loops, this function could potentially return a $|Q| = \infty$. Therefore, in many cases, it is important to not consider all paths, but just those paths that have the same cardinality as the shortest path currently found and thus are shortest paths themselves. It is noted that all the remaining geodesic metrics require only the shortest path between i and j.

C. Eccentricity, Radius, and Diameter

The radius and diameter of a network require the determination of the eccentricity of every vertex in V. The eccentricity metric requires the calculation of $|V| - 1$ shortest path calculations of a particular vertex [15]. The eccentricity of a vertex i is the largest shortest path between i and all other vertices in V such that the eccentricity function $e : V \rightarrow \mathbb{N}$ has the rule

$$e(i) = max\left(\bigcup_{j \in V} s(i,j)\right),$$

where max returns the largest value of its domain.

The radius of the network is the minimum eccentricity of all vertices in V [27]. The function $r : G \rightarrow \mathbb{N}$ has the rule

$$r(G^1) = min\left(\bigcup_{i \in V} e(i)\right).$$

Finally, the diameter of a network is the maximum eccentricity of the vertices in V [27]. The function $d : G \rightarrow \mathbb{N}$ has the rule

$$d(G^1) = max\left(\bigcup_{i \in V} e(i)\right).$$

D. Closeness and Betweenness Centrality

Closeness and betweenness centrality are popular network metrics for determining the “centralness” of a vertex. Closeness centrality is defined as the mean shortest path between some vertex i and all the other vertices in V [3, 16, 25]. The function $c : V \rightarrow \mathbb{R}$ denotes the closeness function and has the rule

$$c(i) = \frac{\sum_{j \in V} s(i,j)}{|V|}.$$

Betweenness centrality is defined for a vertex in V [7, 14, 20]. The betweenness of $i \in V$ is the number of shortest paths that exist between all vertices $j \in V$ and $k \in V$ that have i in their path divided by the total number of shortest paths between j and k, where $i \neq j \neq k$. If $\sigma : V \times V \rightarrow Q$ is a function that returns the set of shortest paths between any two vertices j and k such that

$$\sigma(j,k) = \bigcup_{q \in p(j,k)} q : |q| - 1 = s(j,k)$$

and $\hat{\sigma} : V \times V \times V \rightarrow Q$ is the set of shortest paths between two vertices j and k that have i in the path, where

$$\hat{\sigma}(j,k,i) = \bigcup_{q \in p(j,k)} q : (|q| - 1 = s(j,k) \ \wedge \ i \in q),$$

then the betweenness function $b : V \rightarrow \mathbb{R}$ has the rule

$$b(i) = \sum_{i \neq j \neq k \in V} \frac{\hat{\sigma}(j,k,i)}{\sigma(j,k)}$$

It is worth noting that in [20], the author articulates the point that the shortest paths between two vertices is not necessarily the only mechanism of interaction between two vertices. Thus, the author develops a variation of the betweenness metric that favors shortest paths, but does not utilize only shortest paths in its betweenness calculation.

III. SEMANTIC NETWORK GRAMMARS

A semantic network is a directed labeled graph. However, a semantic network is perhaps best interpreted in an object-oriented fashion where complex objects (i.e. multi-vertex elements) are connected to one another according to various relationship types. While a particular human is represented by a vertex, metadata associated with that individual is represented in the vertices adjacent to the human vertex (e.g. the human’s name, address, age, etc.). In many instances, particular metadata vertices are sinks (i.e. no outgoing edges). In other cases, the metadata of an individual is another complex object such as the friend of that human or the human’s employer.

The topological features of a semantic network are represented by a data type abstraction called an ontology (i.e. a semantic network schema). A popular semantic network representation is the Resource Description Framework (RDF) [19]. RDF Schema (RDFS) is a schema language for developing RDF ontologies in RDF [9]. This article will present all of its concepts from the perspective of RDF and RDFS primarily due to the fact that these are standard data models with a large application-base. However, these ideas can be generalized to any semantic network representation. The first subsection will briefly introduce the concept of RDF and RDFS before describing an ontology for designing geodesic grammars.

A. Introduction to RDF/RDFS

The RDF data model represents a semantic network as a triple list where the vertices and edges (both called resources) are Uniform Resource Identifiers (URI) [11], blank nodes, or literals. If the set of all URIs is denoted U, the set of all blank nodes is denoted B, and the set of all literals is denoted L, then an RDF network is the triple list G^n such that

$$G^n \subseteq ((U \cup B) \times U \times (U \cup B \cup L)).$$

The first resource of a triple is called the subject, the second is called the predicate, and the third is called the object. A single triple $\tau \in G^n$ is denoted as $\tau = \langle s, p, o \rangle$.

All URIs are namespaced such that the URI `http://www.lanl.gov#marko` has a namespace of `http://www.lanl.gov#` and a fragment of `marko`. In many cases, for document and diagram clarity, a namespace is prefixed in such a way that the previous URI is represented as `lanl:marko`. In this article, the namespaces for RDF and RDFS will be prefixed as `rdf` and `rdfs`, respectively.

Blank nodes are "anonymous" vertices and are not discussed in this article as they will not pertain to any of the concepts presented. Literals are any resource that denotes a string, integer, floating point, date, etc. The full taxonomy of literal types is presented in [4].

In RDFS, every vertex is tied to some platonic category representing its `rdfs:Class` using the `rdf:type` property. Moreover, every edge label has domain/range restrictions that determine the vertex types that the edge labels can be used in conjunction with. Because the instance of an ontology obeys the defined constraints of the ontology, the modeler has an abstract representation of the topological features of the semantic network instance in terms of classes (vertices) and properties (edge labels). For example,

$$\langle \texttt{lanl:hasFriend}, \texttt{rdfs:domain}, \texttt{lanl:Human} \rangle$$
$$\langle \texttt{lanl:hasFriend}, \texttt{rdfs:range}, \texttt{lanl:Human} \rangle$$

states that any resource of type `lanl:Human` can have a friend that is only of type `lanl:Human`. Therefore, the following three triples are legal according to the simple ontology above:

$$\langle \texttt{lanl:marko}, \texttt{rdf:type}, \texttt{lanl:Human} \rangle$$
$$\langle \texttt{lanl:jen}, \texttt{rdf:type}, \texttt{lanl:Human} \rangle$$
$$\langle \texttt{lanl:marko}, \texttt{lanl:hasFriend}, \texttt{lanl:jen} \rangle.$$

However, the three statements

$$\langle \texttt{lanl:marko}, \texttt{rdf:type}, \texttt{lanl:Human} \rangle$$
$$\langle \texttt{lanl:fluffy}, \texttt{rdf:type}, \texttt{lanl:Dog} \rangle$$
$$\langle \texttt{lanl:marko}, \texttt{lanl:hasFriend}, \texttt{lanl:fluffy} \rangle$$

are not legal according to the ontology because `lanl:fluffy` is a `lanl:Dog` and a `lanl:Human` cannot befriend anything that is not a `lanl:Human`.

The ontology and legal instance of the previous example are diagrammed in Figure 1. However, for the sake of brevity and clarity of the diagram, the domain and range properties of a class can be abbreviated as in Figure 2. The abbreviated ontological diagram will be used throughout the remainder of this article. It is important to note that both the RDFS ontology and RDF instance network are represented in RDF and thus both instances and ontology are contained within a single semantic network.

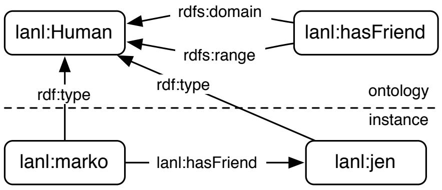

FIG. 1: The full representation of all triples in the ontology and instance layers of the semantic network example.

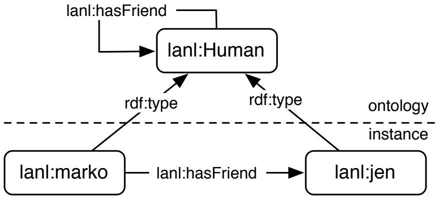

FIG. 2: The abbreviated representation of the ontology and instance layers of the semantic network example.

Finally, an important concept in RDFS is `rdfs:Class` and `rdf:Property` subsumption as denoted by the `rdfs:subClassOf` and `rdfs:subPropertyOf` predicates, respectively. With the `rdfs:subClassOf` and `rdfs:subPropertyOf` predicates, it is possible to generate concept hierarchies. For the purposes of this article, it is only necessary to understand that subsumption is transitive such that if

$$\langle \texttt{lanl:fluffy}, \texttt{rdf:type}, \texttt{lanl:Dog} \rangle$$
$$\langle \texttt{lanl:Dog}, \texttt{rdfs:subClassOf}, \texttt{lanl:Mammal} \rangle$$
$$\langle \texttt{lanl:Mammal}, \texttt{rdfs:subClassOf}, \texttt{lanl:Animal} \rangle,$$

then it can be inferred that because `lanl:fluffy` is a `lanl:Dog`, `lanl:fluffy` is also both a `lanl:Mammal` and a `lanl:Animal`. Transitivity exists for the `rdfs:subPropertyOf` predicate as well.

B. Defining a Grammar

This subsection will define the RDFS ontology for creating a grammar. Any user-defined grammar must obey this ontology. The grammar constructed from this ontology determines the meaning of the value returned by a "semantically aware" geodesic function. Any grammar instance is denoted $\Psi \subseteq ((U \times B) \times U \times (U \times B \times L))$.

The instance of a grammar is represented in RDF and the ontology of the grammar is represented in RDFS. Figure 3 diagrams the ontology of the geodesic grammar, where edges represent properties whose tail is the domain of the property and whose head is the range of the property. Furthermore, the dashed edges denote the RDFS property `rdfs:subClassOf`.

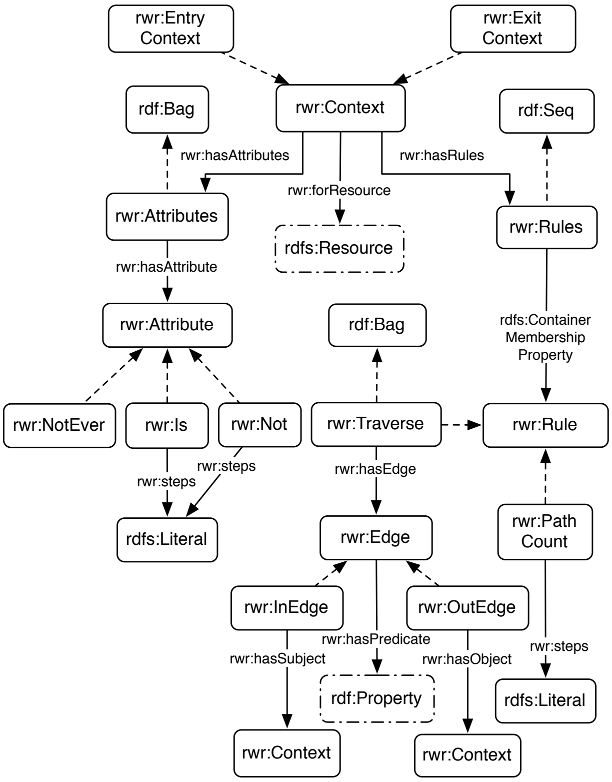

FIG. 3: The ontology for a geodesic path grammar.

The remainder of this section will present an informal review of the major components of the grammar ontology. The next section will formalize all aspects of the resources diagrammed in Figure 3.

Grammar-based geodesics rely on a discrete walker. The walker utilizes a Ψ grammar to constrain its path through G^n. The combination of a walker and a Ψ is a breadth-first search through a particular sub-network of G^n. That sub-network is abstractly represented by Ψ, but not fully realized until after the execution of Ψ on G^n.

Any Ψ is a collection of `rwr:Context` resources connected to one another by `rwr:Traverse` resources. Each `rwr:Context` is an abstract representation of a legal step along a path that a walker can traverse on its way from source vertex i to sink vertex j. An `rwr:Context` has an associated `rwr:forResource` property. The object of that property determines the set of legal vertices that that the `rwr:Context` can resolve to. Only when a walker utilizes a grammar do the `rwr:Contexts` have a resolution to a particular vertex in G^n. `rwr:Context` resolution is further constrained by the `rwr:Rules` and `rwr:Attributes` of the `rwr:Context` in Ψ.

Two important data structures that are used in a grammar are the `rdf:Bag` and `rdf:Seq`. An `rdf:Bag` is an unordered set of elements where each element of the `rdf:Bag` is the object of a triple with predicate `rdf:li`. An `rdf:Seq` is an ordered set of elements where each element of the `rdf:Seq` is the object of a triple with a predicate that is an `rdfs:subPropertyOf` `rdfs:ContainerMembershipProperty` (i.e. `rdf:_1`, `rdf:_2`, `rdf:_3`, etc.).

There exist two `rwr:Rules` (an `rdfs:subClassOf` `rdf:Seq`): `rwr:PathCount` and `rwr:Traverse`. The `rwr:PathCount` rule instructs the walker to record the vertex, edge, and directionality in the ordered path set that is ultimately returned by the grammar-based geodesic algorithm. The `rwr:Traverse` rule instructs the walker to select some outgoing or incoming edge of its current vertex as defined by the set of `rwr:Edges` associated with the `rwr:Traverse` rule. If more than one choice should exist for the walker, the walker chooses both by cloning itself and having each clone take a unique branch of the path.

There exist three `rwr:Attributes` (an `rdfs:subClassOf` `rdf:Bag`): `rwr:NotEver`, `rwr:Is`, and `rwr:Not`. In some instances, when traversing to a new vertex, the walker must respect the fact that it has already seen a particular vertex. The `rwr:NotEver` attribute ensures that the resolution of the `rwr:Context` is not a previously seen vertex, thus preventing infinite loops. The `rwr:Is` attribute allows the walker to explore an area around a particular vertex (i.e. other paths not directly associated with the return path) while still ensuring that the walker returns to the original vertex. Finally, the `rwr:Not` attribute ensures that the walker does not return to a *particular* previously seen vertex.

If vertex i is the head of the path (i.e. source), then it is defined in an `rwr:EntryContext`. If vertex j is the tail of the path (i.e. sink), then it is defined in an `rwr:ExitContext`. The purpose of the walker is to move from source to sink in G^n by respecting the `rwr:Rules` and `rwr:Attributes` of the `rwr:Contexts` that it traverses in Ψ. Figure 4 diagrams the relationship between a walker, its grammar Ψ, and its network instance G^n. The grammar acts as a user-defined "program" that the walker executes, where the language of that program is defined by the grammar ontology.

The next section will formalize the grammar.

IV. FORMALIZING THE GRAMMAR-BASED MODEL

Once a grammar has been defined according to the constraints of the ontology diagrammed in Figure 3, the path function $\rho : V \times V \times \Psi \rightarrow Q$ can be executed. The function ρ returns the set of all paths between any two vertices $i, j \in V$. This section will define the rules by

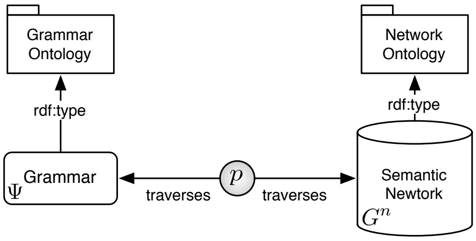

FIG. 4: A walker p walks both Ψ and G^n.

which ρ interprets its domain parameters and ultimately derives a path set.

The grammar-based model requires the walker to query G^n such that it can determine the set of legal vertices and edges that it can traverse. Moreover, the walker must be able to query Ψ in order to know which `rwr:Rules` and `rwr:Attributes` to respect. The mechanism by which the walker queries G^n and Ψ is called the symbol binding model. For example, the following query

$$\begin{aligned} X = &\{?x \mid \langle ?x, \texttt{lanl:hasFriend}, \texttt{lanl:jhw}\rangle \in G^n \\ &\wedge \langle ?x, \texttt{lanl:worksFor}, \texttt{lanl:LANL}\rangle \in G^n\} \end{aligned}$$

would fill the unordered set X with all people that have `lanl:jhw` as their friend and who work for `lanl:LANL`. A more advanced query example is

$$\begin{aligned} X = &\{?x, ?y \mid \langle ?x, \texttt{lanl:hasFriend}, ?y\rangle \in G^n \\ &\wedge \langle ?y, \texttt{lanl:worksFor}, \texttt{lanl:LANL}\rangle \in G^n \\ &\wedge \langle ?x, \texttt{lanl:worksFor}, \texttt{lanl:PNNL}\rangle \in G^n\}. \end{aligned}$$

In the above query, the set X is an unordered set of ordered pairs of friends where one of the friends works at `lanl:LANL` and the other works at `lanl:PNNL`.

A. Initializing a Walker p

The path function ρ is supplied with a start vertex i, an end vertex j, and a grammar Ψ. Upon the execution of ρ, a single walker, denoted p, is created and added to the set of walkers P, where at $n = 0$, $|P| = 1$, and $n \in \mathbb{N}$ is in discrete time. The set P may increase in size over the course of the algorithm as clone particles are created where multiple legal options exist for traversal.

Every walker has two ordered multi-sets associated with it: g^p and q^p. The multi-set g^p is an ordered set of vertices, edges, and edge directions traversed by p, where g^p_n is the vertex location of p at time step n. The element $g^p_{n'}$ denotes the predicate (i.e. edge label) used by p to traverse to g^p_n and the element $g^p_{n''}$ denotes the directionality of the predicate used in that traversal. For example, suppose $g^p = (\texttt{lanl:marko}, \texttt{lanl:hasFriend}, +, \texttt{lanl:jhw}, \texttt{lanl:hasFriend}, +, \texttt{lanl:norman})$. In the presented path, $g^p_0 = \texttt{lanl:marko}$, $g^p_{1'} = \texttt{lanl:hasFriend}$, $g^p_{1''} = +$, $g^p_1 = \texttt{lanl:jhw}$, $g^p_{2'} = \texttt{lanl:hasFriend}$, $g^p_{2''} = +$, and $g^p_2 = \texttt{lanl:norman}$. Note that $g^p_{0'} = \emptyset$ and $g^p_{0''} = \emptyset$. The example path is diagrammed in Figure 5.

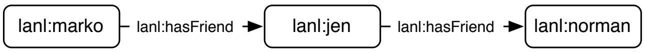

FIG. 5: An example of a g^p path.

The multi-set q^p is an ordered set of vertices, edges, and directionalities that are recorded by p along its path through G^n. The set q^p maintains the same indexing schema of $'$ and $''$ as g^p. The main distinction between g^p and q^p is that q^p is the returned path, *not* the actual path of p. If p reaches its destination `rwr:ExitContext` in Ψ and thus vertex $j \in V$, then the set q^p is one of the elements in the return set Q of the path function ρ. Thus, for the grammar-based geodesic model,

$$Q = \bigcup_{p \in P} q^p : (q^p_0 = i \ \wedge \ q^p_{\frac{|q^p|-1}{3}} = j).$$

The $\frac{|q^p|-1}{3}$ is necessary to transform the length of q^p into an index in n time (due to the $'$ and $''$ notation convention) because the set q^p includes edge labels and edge directionality as well as vertices.

B. Entering G^n and Ψ

The initial walker p starts its journey at the `rwr:EntryContext` in Ψ and the vertex i in V. Thus, $g^p_0 = i$. As in Figure 3, the `rwr:EntryContext` must be the domain of the predicate `rwr:forResource` whose range is i. An `rwr:EntryContext` must have no `rwr:Attributes` and must have the rule `rwr:PathCount` such that $q^p_0 = i$.

From $i \in V$ and the `rwr:EntryContext` in Ψ, p will move to some new $k \in V$ and some new `rwr:Context` in Ψ. Before discussing the `rwr:Traverse` rule, it is necessary to discuss the attributes that determine the set of legal edges that can be traversed by p.

C. The `rwr:NotEver` Attribute

The `rwr:NotEver` attribute is useful for ensuring that path loops do not occur and thus cause the path algorithm to run indefinitely. If p is trying to traverse to a new `rwr:Context` at $n+1$ and that `rwr:Context` has the `rwr:NotEver` attribute, then

$$\overline{X}(p)_{n+1} = \bigcup_{m \leq n} g^p_m.$$

The set $\overline{X}(p)_{n+1}$ is the set of vertices to which p can legally resolve the $n+1$ `rwr:Context`. Note that the definition of $\overline{X}(p)$ does not include edge labels or edge directionality, only vertices. This is due to the fact that the time index (n) of g^p are not superscripted with $'$ or $''$.

D. The rwr:Is Attribute

The `rwr:Is` attribute guarantees that the vertex resolved to by a particular `rwr:Context` is a vertex seen on a previous step of the walker's g^p. For instance, suppose that a walker must check that a particular individual works for the Los Alamos National Laboratory before traversing a different edge label of `lanl:jhw`. This problem is diagrammed in Figure 6.

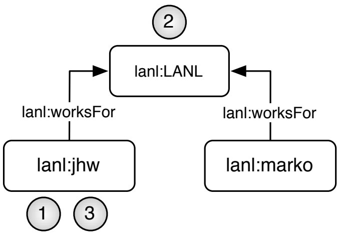

FIG. 6: `rwr:Is` can be used to ensure that a walker backtracks.

In Figure 6, the walker is at `lanl:jhw` at time step $n = 1$. At time step $n = 2$, the walker must check to see if `lanl:jhw lanl:worksFor lanl:LANL`. To do so, the walker will traverse `lanl:worksFor` edge. Upon validating the `lanl:LANL`, the walker must return back to `lanl:jhw`. Therefore, the walker will take the inverse of the `lanl:worksFor` edge (i.e. oppose the directionality of the edge). However, despite the existence of an inverse `lanl:worksFor` edge to `lanl:marko`, the walker should not clone itself. Therefore, in order to specify that the walker must return to `lanl:jhw`, it is important to use the `rwr:Is` attribute such that only a single walker p returns to `lanl:jhw` at $n = 3$ and P is unchanged.

The set of all legal vertices that an `rwr:Context` can resolve to is defined by the set O, where if ψ is the `rwr:Context` at $n+1$ that maintains an `rwr:Is` attribute, then

$$\begin{aligned} M = \{?m \mid & \langle \psi, \texttt{rwr:hasAttributes}, ?x \rangle \in \Psi \\ & \langle ?x, \texttt{rwr:hasAttribute}, ?y \rangle \in \Psi \\ & \langle ?y, \texttt{rdf:type}, \texttt{rwr:Is} \rangle \in \Psi \\ & \langle ?y, \texttt{rwr:step}, ?m \rangle \in \Psi \} \end{aligned}$$

and

$$O(p)_{n+1} = \bigcup_{m \in M} g^p_{n-m}.$$

The set $O(p) \subseteq V$ is the set of legal vertex resources that the $n + 1$ `rwr:Context` ψ can resolve to and is used in the calculation of an `rwr:Traverse` at n.

E. The rwr:Not Attribute

The `rwr:Not` attribute determines the set of vertices that the $n + 1$ `rwr:Context` cannot resolve to. This is similar to the $\overline{X}(p)$ set, except that it is for some n, not for all n in the past. For example, suppose that the walker must only consider an article co-authorship network. This problem is diagrammed in Figure 7.

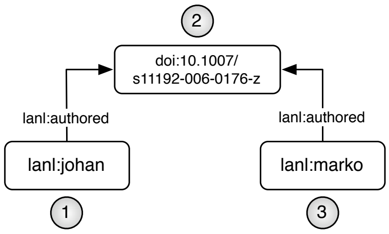

FIG. 7: `rwr:Not` can be used to ensure that a walker does not backtrack.

In Figure 7, the walker must determine if the article `doi:10.1007/s11192-006-0176-z` has at least 2 co-authors. In order to do so, the walker must not return to `lanl:jbollen` at $n = 3$. If

$$\begin{aligned} M = \{?m \mid & \langle \psi, \texttt{rwr:hasAttributes}, ?x \rangle \in \Psi \\ & \langle ?x, \texttt{rwr:hasAttribute}, ?y \rangle \in \Psi \\ & \langle ?y, \texttt{rdf:type}, \texttt{rwr:Not} \rangle \in \Psi \\ & \langle ?y, \texttt{rwr:step}, ?m \rangle \in \Psi \} \end{aligned}$$

and

$$X(p)_{n+1} = \bigcup_{m \in M} g^p_{n-m},$$

then $X(p) \subseteq V$ is the set of vertices that the $n + 1$ `rwr:Context` ψ must not resolve to and is used in the calculation of an `rwr:Traverse` at n.

F. The rwr:Traverse Rule

The `rwr:Traverse` rule is perhaps the most important aspect of the grammar. An `rwr:Traverse` rule of an `rwr:Context` determines the next `rwr:Context` that p should traverse to in Ψ as well as the next $k \in V$. It utilizes the previously defined attribute sets $\overline{X}(p)$, $O(p)$, and $X(p)$ in its calculation. An `rwr:Traverse` rule is composed of a set of `rwr:Edge`s that can be either incoming or outgoing. Thus, unlike in directed networks, the path of a p is not constrained by the directionality of the edges. The Γ functions are defined as $\Gamma : V \times P \to G$ and t is the `rwr:Traverse` rule of the current `rwr:Context` ψ. Therefore, if

$$\begin{aligned} Y_{\text{out}} = \{?y \mid & \langle t, \texttt{rwr:hasEdge}, ?y \rangle \in \Psi \\ & \langle ?y, \texttt{rdf:type}, \texttt{rwr:OutEdge} \rangle \in \Psi \}, \end{aligned}$$

$$\begin{aligned} Y_{\text{in}} = \{?y \mid & \langle t, \texttt{rwr:hasEdge}, ?y \rangle \in \Psi \\ & \langle ?y, \texttt{rdf:type}, \texttt{rwr:InEdge} \rangle \in \Psi \}, \end{aligned}$$

$$
\begin{aligned}
\Gamma^{+}(a,p) = \bigcup_{y \in Y_{\text{out}}} \{\langle a, ?\omega, ?b\rangle \mid\ & \langle a, ?\omega, ?b\rangle \in G^n \\
& \wedge\ \langle y, \texttt{rwr:hasPredicate}, ?w\rangle \in \Psi \\
& \wedge\ (\langle ?\omega, \texttt{rdfs:subPropertyOf}, ?w\rangle \in G^n \\
& \quad \vee\ ?\omega = ?w) \\
& \wedge\ \langle y, \texttt{rwr:hasObject}, ?x\rangle \in \Psi \\
& \wedge\ \langle ?x, \texttt{rwr:forResource}, ?z\rangle \in \Psi \\
& \wedge\ (\langle ?b, \texttt{rdf:type}, ?z\rangle \in G^n \vee ?b = ?z) \\
& \wedge\ (O(p)_{n+1} = \emptyset \vee ?b \in O(p)_{n+1}) \\
& \wedge\ ?b \notin X(p)_{n+1} \wedge ?b \notin \overline{X}(p)_{n+1}\},
\end{aligned}
$$

and

$$
\begin{aligned}
\Gamma^{-}(a,p) = \bigcup_{y \in Y_{\text{in}}} \{\langle ?b, ?\omega, a\rangle \mid\ & \langle ?b, ?\omega, a\rangle \in G^n \\
& \wedge\ \langle y, \texttt{rwr:hasPredicate}, ?w\rangle \in \Psi \\
& \wedge\ (\langle ?\omega, \texttt{rdfs:subPropertyOf}, ?w\rangle \in G^n \\
& \quad \vee\ ?\omega = ?w) \\
& \wedge\ \langle y, \texttt{rwr:hasSubject}, ?x\rangle \in \Psi \\
& \wedge\ \langle ?x, \texttt{rwr:forResource}, ?z\rangle \in \Psi \\
& \wedge\ (\langle ?b, \texttt{rdf:type}, ?z\rangle \in G^n \vee ?b = ?z) \\
& \wedge\ (O(p)_{n+1} = \emptyset \vee ?b \in O(p)_{n+1}) \\
& \wedge\ ?b \notin X(p)_{n+1} \wedge ?b \notin \overline{X}(p)_{n+1}\},
\end{aligned}
$$

then

$$\Gamma(a,p) = \Gamma^{+}(a,p) \cup \Gamma^{-}(a,p),$$

where $\Gamma(a,p)$ is the set of legal edges that p can traverse given its current V location of a and Ψ location ψ. Note that the set $\Gamma(a,p)$ has a unique set of elements. If $\Gamma(a,p) = \emptyset$, then p halts.

Unlike the grammar-based eigenvector model of [22], the geodesic requires the searching of all legal paths. In line with a breadth-first search, all network branches are checked. Thus, for every triple $\tau \in \Gamma(a,p)$, a clone walker is created and added to P. This idea will be made more salient in the example to follow.

G. The `rwr:PathCount` Rule

The `rwr:PathCount` rule is the mechanism by which values in g^p get appended to q^p, where q^p is the path returned by p at the end of the algorithm's execution. The rule instructs p to append a path segment in g^p to the ordered multi-set q^p. If a particular `rwr:Context` ψ has the `rwr:PathCount` rule with the `rwr:step` x such that $x \in \mathbb{N}$, then p will append $g^p_{n-x'}$, $g^p_{n-x''}$, and g^p_{n-x} to q^p such that none of the elements copied from $g^p = \emptyset$ and they are added in their respective order.

The next section will present the aforementioned rules and attributes within the framework of a particular social network ontology in order to demonstrate a practical application.

V. GEODESICS IN A SEMANTIC SOCIAL NETWORK

This section will present two examples of the previously presented ideas to the problem of calculating semantically meaningful geodesic functions within a semantic social network. Figure 8 presents an RDFS network ontology that will be used throughout the remainder of this section. Note that the domain and range of the properties are denoted by the tail and head of the edge, respectively.

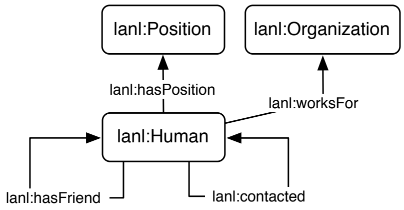

FIG. 8: An example semantic social network ontology.

Figure 9 diagrams an example instance that respects the ontological constraints diagrammed in Figure 8.

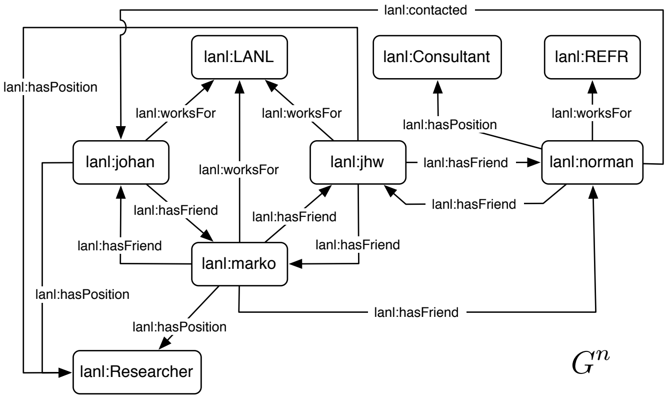

FIG. 9: An example semantic social network instance.

The first example will demonstrate how to determine all the non-recurrent paths between the vertex `lanl:johan` and `lanl:norman` such that only friendship paths are taken, but those intervening friend vertices must have a `lanl:Researcher` position. The second example will present a grammar that simulates an unlabeled network path calculation by ignoring vertex types and edge labels.

Note that the two examples presented are for locating all paths between a source and a sink vertex. This is for demonstration purposes only. If one required only the shortest path, once a path between the source and sink has been found, the algorithm can halt. In unweighted networks, using a breadth-first search algorithm, the first path discovered is always the shortest path [12].

A. A Non-Recurrent Paths Grammar

Figure 10 presents a geodesic grammar that determines the set of all non-recurrent paths between `lanl:johan` and `lanl:norman` according to `lanl:hasFriend` relationships where every friend along the walker's path must be a `lanl:Researcher`.

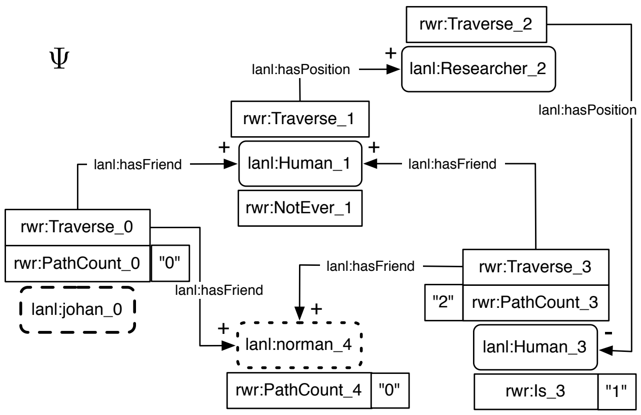

FIG. 10: A grammar to determine all non-recurrent `lanl:hasFriend` paths from `lanl:jbollen` to `lanl:norman`.

Note the diagrammatic conventions used to represent a grammar. Every `rwr:Context`, `rwr:Rule`, and `rwr:Attribute` has a _# after its type. This is to denote that each representation of the same `rwr:Context`, `rwr:Rule`, or `rwr:Attribute` is, in fact, a distinct vertex in Ψ. The label of the `rwr:Context` is the object of the `rwr:forResource` property minus the _#. Furthermore, the dashed contexts are `rwr:EntryContexts` and the dotted contexts are `rwr:ExitContexts`. Thus, `lanl:johan_0` is the source context and `lanl:norman_4` is the sink context in Ψ, and where `lanl:johan` is the source vertex and `lanl:norman` is the sink vertex in G^n.

The `rwr:Rules` of an `rwr:Context` are represented in their order of execution from bottom to top. The `rwr:Attributes` are associated, in no particular order, with their respective `rwr:Context`. If a rule or attribute requires a literal `rwr:step` specification, that literal is appended to its respective rule or attribute. The `+` or `-` symbol on the head of an edge denotes whether the `rwr:Traverse` edge is an `rwr:OutEdge` or `rwr:InEdge`, respectively.

At $n = 0$, $g_0^{p_0} = \texttt{lanl:johan}$ and $P = \{p_0\}$. The first rule to be executed is the `rwr:PathCount_0` rule in which p_0 will register $g_0^{p_0}$ in q^p such that $q_0^{p_0} = g_0^{p_0}$. After adding `lanl:johan` to q^{p_0}, the walker will execute the `rwr:Traverse_0` rule. The `rwr:Traverse_0` rule yields a $\Gamma(\texttt{lanl:johan}, p_0) = \{\langle\texttt{lanl:johan}, \texttt{lanl:hasFriend}, \texttt{lanl:marko}\rangle\}$. If `lanl:norman` was a friend of `lanl:johan`, then that edge would have been represented in $\Gamma(\texttt{lanl:johan}, p_0)$ as well. Because $\texttt{lanl:marko} \notin g^{p_0}$, the `rwr:NotEver_1` attribute of the `Human_1` context has an $\overline{X}(p_0)_1 = \emptyset$.

At $n = 1$, the current path of p_0 is $g^{p_0} = (\texttt{lanl:johan}, \texttt{lanl:hasFriend}, +, \texttt{lanl:marko})$ and the current return path $q^{p_0} = (\texttt{lanl:johan})$. There exists only one rule at `rwr:Human_1`. The `rwr:Traverse_1` rule dictates that p_0 take an outgoing edge from `lanl:marko` to a `lanl:Researcher` position. Given that there is only one edge that can be traversed, $\Gamma(\texttt{lanl:marko}, p_0) = \{\langle\texttt{lanl:marko}, \texttt{lanl:hasPosition}, \texttt{lanl:Researcher}\rangle\}$.

At $n = 2$, the current path of p_0 is $g^{p_0} = (\texttt{lanl:johan}, \texttt{lanl:hasFriend}, +, \texttt{lanl:marko}, \texttt{lanl:hasPosition}, +, \texttt{lanl:Researcher})$ and the current return path $q^{p_0} = (\texttt{lanl:johan})$. The only rule of the `lanl:Researcher_2` context is to return the human that was last encountered as specified by the `rwr:Is_3` attribute of the next `lanl:Human_3` context. Thus, $\Gamma(\texttt{lanl:Researcher}, p_0) = \{\langle\texttt{lanl:marko}, \texttt{lanl:hasPosition}, \texttt{lanl:Researcher}\rangle\}$.

At $n = 3$, the current path of p_0 is $g^{p_0} = (\texttt{lanl:johan}, \texttt{lanl:hasFriend}, +, \texttt{lanl:marko}, \texttt{lanl:hasPosition}, +, \texttt{lanl:Researcher}, \texttt{lanl:hasPosition}, -, \texttt{lanl:marko})$. Given the `rwr:PathCount_3` rule with a `rwr:step` of 2, $q^{p_0} = (\texttt{lanl:johan}, \texttt{lanl:hasFriend}, +, \texttt{lanl:marko})$. The `rwr:Traverse_3` rule provides a $\Gamma(\texttt{lanl:marko}, p_0)$ with two edges such that $\Gamma(\texttt{lanl:marko}, p_0) = (\langle\texttt{lanl:marko}, \texttt{lanl:hasFriend}, \texttt{lanl:jhw}\rangle, \langle\texttt{lanl:marko}, \texttt{lanl:hasFriend}, \texttt{lanl:norman}\rangle)$. Note that the edge $\langle\texttt{lanl:marko}, \texttt{lanl:hasFriend}, \texttt{lanl:johan}\rangle$ does not exist in $\Gamma(\texttt{lanl:marko}, p_0)$ because of the `rwr:NotEver_1` attribute at the `lanl:Human_1` context (i.e. $\overline{X}(p_0)_4 = \{\texttt{lanl:johan}, \texttt{lanl:marko}\}$). Because two edges exist in $\Gamma(\texttt{lanl:marko}, p_0)$, p_0 is cloned such that $P = \{p_0, p_1\}$, $g^{p_0} = g^{p_1}$, and $q^{p_0} = q^{p_1}$. The walker p_0 will take one edge and p_1 will take the other edge.

At $n = 4$, p_1 will be at `lanl:norman` in G^n and thus at an `rwr:ExitContext` in Ψ. However, before p_1 halts, `rwr:PathCount_4` is executed such that $Q = \{q^{p_1}\} = \{(\texttt{lanl:johan}, \texttt{lanl:hasFriend}, +, \texttt{lanl:marko}, \texttt{hasFriend}, +, \texttt{lanl:norman})\}$. At the completion of `rwr:PathCount_4` there are no other rules to execute and thus p_1 halts. The walker p_0, on the other hand, will be at `lanl:jhw` at $n = 4$. It is not until $n = 7$ that p_0 arrives at `lanl:norman`.

At $n = 7$, $q^{p_0} = (\texttt{lanl:johan}, \texttt{lanl:hasFriend}, +, \texttt{lanl:marko}, \texttt{lanl:hasFriend}, +, \texttt{lanl:jwh}, \texttt{lanl:hasFriend}, +, \texttt{lanl:norman})$. At $n = 7$, the grammar is complete and $|Q| = 2$.

The shortest path of Q is defined as the function $s : V \times V \times \Psi \rightarrow \mathbb{N}$, where

$$s(i, j, \Psi) = min\left(\bigcup_{q\in\rho(i,j,\Psi)} \frac{|q|-1}{3}\right).$$

The 1 must be subtracted from $|q|$ in order to not include source vertex i as a step and then must be divided by 3 so as to avoid the inclusion of the edge label and directionality of the edge in the path length calculation. In the example presented, the shortest "researcher-constrained friendship" path is 2. From s, it is possible to generate all other geodesic functions as defined in Section II.

B. A Grammar to Simulate Unlabeled Geodesics

This section presents another example of the grammar-based geodesic algorithm. In this example, the grammar presented is equivalent to removing the edge labels and directionality from the semantic network and calculating a traditional geodesic metric on it. Figure 11 presents the grammar where, in RDFS, rdfs:Resource is the base type of all resources (vertices and edge labels). Thus, all rwr:Contexts and rwr:Edges can legally resolve to any vertex and edge label, respectively.

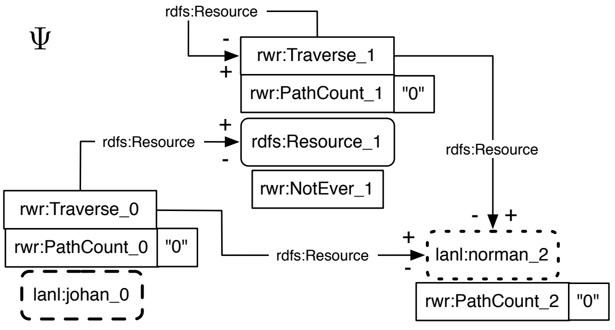

FIG. 11: An unconstrained grammar to determine all non-recurrent paths from lanl:jbollen to lanl:norman.

The grammar in Figure 11 will determine the set of all non-recurrent paths between lanl:johan and lanl:norman such that any edge type can be traversed to any vertex type. The central rwr:Context is the rdfs:Resource_1 context. A walker will loop over rwr:Resource_1 until it can find an edge to make the final traversal to lanl:norman. Note the use of both rwr:OutEdges (+) and rwr:InEdges (-). With both edges accessible, the walker can walk in any direction on the network. Thus, this grammar is equivalent to executing a geodesic on an undirected and unlabeled version of the semantic network. Finally, the grammar will produce no recurrent paths because of the rwr:NotEver_1 rule.

Given this Ψ and the original social network instance G^n diagrammed in Figure 9, the shortest path between lanl:johan and lanl:norman is (lanl:johan, lanl:contacted, $-$, lanl:norman) with a path length of 1. To contrast, in the first example when the walker's path was constrained to researcher friendship relationships, the shortest path between lanl:johan and lanl:norman was 2.

VI. ANALYSIS

The semantic network is an unweighted network. Thus, determining the shortest path between any two vertices is best solved by a breadth-first algorithm. The grammar-based walker, through cloning, is analogous to a breadth-first search through the network. However, not all edges are considered by the walker and thus, the running time of the algorithm is less than or equal to $O(|V| + |G^n|)$. The determination of the running time of the algorithm is grammar dependent. In order to calculate the running time of a particular grammar, it is important to calculate the number of vertices and edges of the grammar-specified types in G^n. In the worst case situation, the walker population P will have traversed all vertices and edges from the source to ultimately locate the sink. However, because the network is unweighted, once the sink has been found by a single $p \in P$, the shortest path has been determined so the algorithm is complete.

VII. CONCLUSION

The digital footprint left by individuals as they go about their lives interacting with each other and worldly and conceptual artifacts has created a prevalence of large-scale multi-relational data sets. This article has presented an introduction to the standards used to represent such data sets in the computer, library, and biological sciences (e.g. RDF) [5, 21, 24], as well as a technique to port some of the most fundamental network analysis algorithms into this large-scale and multi-relational realm.

There currently exist many technologies to support large-scale semantic network models represented according to the RDF specification. High-end modern-day triple stores support on the order of 10^9 triples [17]. While many centrality algorithms are costly on large networks, by restricting the search to semantically meaningful subsets of the full semantic network, as defined by a grammar, geodesic metrics can be reasonably executed on even the most immense and complex of data sets.

With the grammar-based technique presented in this article, the complete range of geodesic metrics are made available for semantic network analysis. It is the hope that with RDF technology, the technique presented in this article, and the technique for calculating the primary eigenvector of a network as presented in [22], social network researchers will be able to confidently move into the compute intensive arena of billion edge semantic networks.

Acknowledgments

Marko A. Rodriguez is funded by the MESUR project (http://www.mesur.org) which is supported by a grant from the Andrew W. Mellon Foundation.

[1] Aleman-Meza, B., C. Halaschek-Wiener, I. B. Arpinar, C. Ramakrishnan, and A. P. Sheth, 2005, IEEE Internet Computing **9**(3), 37, ISSN 1089-7801.

[2] Anyanwu, K., and A. Sheth, 2003, in *Proceedings of the Twelfth International World-Wide Web Conference* (Budapest, Hungary).

[3] Bavelas, A., 1950, The Journal of the Acoustical Society of America **22**, 271.

[4] Biron, P. V., and A. Malhotra, 2004, *XML Schema Part 2: Datatypes Second Edition*, Technical Report, World Wide Web Consortium.

[5] Bollen, J., M. A. Rodriguez, H. Van de Sompel, L. L. Balakireva, and A. Hagberg, 2007, in *ACM World Wide Web Conference* (ACM Press, Banff, Canada).

[6] Bonacich, P., 1987, American Journal of Sociology **92**(5), 1170.

[7] Brandes, U., 2001, Journal of Mathematical Sociology **25**(2), 163.

[8] Brandes, U., and T. Erlebach (eds.), 2005, *Network Analysis: Methodolgical Foundations* (Springer, Berling, DE).

[9] Brickley, D., and R. Guha, 2004, *RDF Vocabulary Description Language 1.0: RDF schema*, Technical Report, World Wide Web Consortium, URL `http://www.w3.org/TR/rdf-schema/`.

[10] Brin, S., and L. Page, 1998, Computer Networks and ISDN Systems **30**(1–7), 107.

[11] Burners-Lee, T., , R. Fielding, D. Software, L. Masinter, and A. Systems, 2005, Uniform Resource Identifier (URI): Generic Syntax.

[12] Cormen, T. H., C. E. Leiserson, and R. L. Rivest, 1999, *Introduction to Algorithms* (MIT Press).

[13] Dijkstra, E. W., 1959, Numerische Mathematik **1**, 269.

[14] Freeman, L. C., 1977, Sociometry **40**(35–41).

[15] Harary, F., and P. Hage, 1995, Social Networks **17**, 57.

[16] Leavitt, H. J., 46, Journal of Abnornal and Social Psychology , 38.

[17] Lee, R., 2004, *Scalability Report on Triple Store Applications*, Technical Report, Massachusetts Institute of Technology.

[18] Lin, S., 2004, in *Sixteenth Conference on Innovative Applications of Artificial Intelligence*, edited by D. L. McGuinness and G. Ferguson (MIT Press), pp. 991–992.

[19] Manola, F., and E. Miller, 2004, RDF primer: W3C recommendation, URL `http://www.w3.org/TR/rdf-primer/`.

[20] Newman, M. E., 2005, Social Networks **27**(1), 39, URL `http://arxiv.org/abs/cond-mat/0309045`.

[21] Portwin, K., and P. Parvatikar, 2006, in *XTech: Building Web 2.0* (Amsterdam, Netherlands).

[22] Rodriguez, M. A., 2007, *Grammar-based Random Walkers in Semantic Networks*, Technical Report, Los Alamos National Laboratory, URL `http://www.soe.ucsc.edu/~okram/papers/random-grammar.pdf`.

[23] Rodriguez, M. A., 2007, in *38th Annual Hawaii International Conference on Systems Science (HICSS'07)* (Waikoloa, Hawaii).

[24] Ruttenberg, A., T. Clark, W. Bug, M. Samwald, O. Bodenreider, H. Chen, D. Doherty, K. Forsberg, Y. Gao, V. Kashyap, J. Kinoshita, J. Luciano, *et al.*, 2007, BMC Bioinformatics **8**(3), S2, ISSN 1471-2105, URL `http://www.biomedcentral.com/1471-2105/8/S3/S2`.

[25] Sabaidussi, G., 1966, Psychometrika **31**, 581.

[26] Sheth, A. P., I. B. Arpinar, C. Halaschek, C. Ramakrishnan, C. Bertram, Y. Warke, D. Avant, F. S. Arpinar, K. Anyanwu, and K. Kochut, 2005, Journal of Database Management **16**(1), 33.

[27] Wasserman, S., and K. Faust, 1994, *Social Network Analysis: Methods and Applications* (Cambridge University Press, Cambridge, UK).

[28] Zhuge, H., and L. Zheng, 2003, in *Proceedings of the Twelfth International World Wide Web Conference (WWW03)* (Budapest, Hungary).