Graphitti: An Annotation Management System for Heterogeneous Objects

Sandeep Gupta, Christopher Condit, Amarnath Gupta

San Diego Supercomputer Center, University of California San Diego
9500 Gilman Dr., La Jolla, CA 92093, USA

{sandeep, condit, gupta}@sdsc.edu

Abstract— **Annotation is the process of supplementing data with additional information that was not part of the actual observation, but reflects post-facto comments and associations made by a user who analyzes the data. While annotation management systems are emerging in the field of relational data, such systems for scientific applications, where there is a wide heterogeneity in the types of annotable data, are almost nonexistent. In this demonstration paper, we describe *Graphitti*, a tool that (i) allows a user to annotate a wide variety of scientific data, and (ii) allows a user to query data and their annotations in a seamless manner.**

I. Introduction

Informally, annotation has been described as the process of superimposing information on an existing database [7]. Often, this additional information was not originally intended to be a part of the collected data, and hence no data or schema structure was designed to hold it. Annotating data is a very common practice in science, where scientists would literally "mark" experimental observation with comments, and often use annotations to share their opinions in a collaborative study. As larger scale experiments are conducted and larger collaborations are formed, management of the annotated data becomes a serious challenge.

In recent times, the emerging importance of annotation in scientific data management has been recognized by the Information Management community, leading to a variety of research in annotation management such as [5], [2], [6], [3], [8]. However, these efforts mostly center around more traditional forms of data like relational tables and XML documents.

***Graphitti* Annotation Model.** In *Graphitti* we have focused on a different aspect of annotation management for scientific data, that presents its own challenges. We consider the problem of creating an annotation platform where a scientist can create and search through annotations on the widely heterogeneous types of data that they produce. Thus, DNA sequences, molecular interaction graphs, 3D models of proteins, images showing expressions of a protein, would all get annotated with a comment, or a reference to a known piece of knowledge; sometimes an annotation will depict a newly discovered correlation between two different pieces of data. In all cases, the annotation will construct a "marked" portion of data object, and attach a set of marked object portions to an annotation. From this perspective, we consider an annotation as a "linker object" that connects the *annotation content* (i.e., the comment itself) to one or more *annotation referents* (i.e., the object

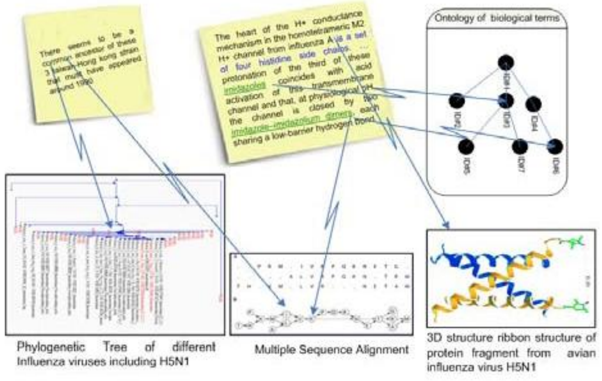

There seems to be a common ancestor of these 3 Taiwan-Hong kong strain that must have appeared around 1990
The heart of the H+ conductance mechanism in the homotetrameric M2 H+ channel from influenza A is a set of four histidine side chains. ... protonation of the third of these imidazoles coincides with acid activation of this transmembrane channel and that, at physiological pH the channel is closed by two imidazole-imidazolium dimers, each sharing a low-barrier hydrogen bond.
Ontology of biological terms
Phylogenetic Tree of different Influenza viruses including H5N1
Multiple Sequence Alignment
3D structure ribbon structure of protein fragment from avian influenza virus H5N1

Fig. 1. An annotation scenario from an interdisciplinary study of the Influenza virus

fragments that are marked for annotation). A collection of annotation contents and referents would induce a graph, where there are two types of nodes, the contents and the referents, and a directed edge connects a content to a referent. If the same referent is connected to two different annotations, possibly by two different scientists, the two annotations become indirectly related. We call this the *a-graph* (see figure 1); it is the "connection structure" that associates the substructures of all other types of data. *Graphitti* addresses several challenges of annotation specified in [1] namely, annotation of objects at different granularities, and performing annotation queries in the general context of heterogeneous data types. More importantly, it allows a user to formulate queries over data and annotations, like: *Find annotations that contain the term "protein:TP53" and have paths to all mouse brain images having at least 2 regions annotated with ontology term "Deep Cerebellar nuclei"*. The demonstration presents a prototype to annotate and query graph-connected substructures of heterogeneous types of data.

II. System Design

The *Graphitti* system includes the following components. The **annotation content** produced by *Graphitti* is an XML document whose elements consist of Dublin core attributes and other user-defined tags. The collection of all annotations constitutes a database of XML documents. The collection-searching operations is performed using standard XQuery.

The **data objects** and their metadata are modeled as type-specific relations stored in a relational database – thus DNA sequences, protein sequences, images etc. all have their metadata stored in separate tables. The raw actual data is also stored in the same tables in their native formats.

The **annotated substructures** of the primary data are stored in a collection of interval trees for 1D data (e.g. sequences) and a collection of R-tree for 2D and 3D data (e.g., image regions). Simple techniques are used to keep the number of the index structures small. A single interval tree is created per chromomosome instead of per annotated DNA sequence; regions all brain images of the same resolution are referenced with respect to the same brain coordinate system, and placed in a single R-tree, and so forth. Some of the operations will apply on all substructures (called SUB_X below) in our purview:

$ifOverlap : SUB\_X \times SUB\_X \longmapsto \{0, 1\}$, which returns true if the two substructures overlap. Some other operators will only operate on substructures that have specific properties.

$next : SUB\_X \longmapsto SUB\_X$ is applicable on data types for which there is a strict ordering on the domain; it returns the sub-structure encountered next in the ordering input substructure. The semantics of "next" depends upon the data types.

$intersect : SUB\_X \times SUB\_X \longmapsto SUB\_X$, which returns the intersection of two SUB_X. This operation is valid for convex data types such as sequences and rectangles.

Quite often, scientists use a **common vocabulary** or **ontology** to ensure their term references are compatible. In *Graphitti* we use OntoQuest [4] where ontologies are modeled as graphs whose nodes correspond to terms and edges are domain-specific quantified binary relationships between term pairs. An annotation only points to ontology nodes. Common operations on ontologies include:

$CI : C \longmapsto I^+$, returns the set of all instances of a concept.

$CRI : C \times R \longmapsto I^+$ returns the set of all instance of a concept by relation R.

$CmRI : C \times R^+ \longmapsto I^+$ returns the set of all instances of a concept $c \in C$ restricted to a set of relation types.

$mCmRI : C^+ * R^+ \longmapsto I+$ returns the all the instances reachable via any concepts from a set using only edges from R^+.

$SubTree(X, R1)$: Returns the subtree under X restricted to edge relation R1.

$SubTree(X, R1) - SubTree(Y, R1)$: If Y is a descendent of X, then this operation returns the subtree under X minus the subtree under Y restricted to relation R1.

The **a-graph structure** that connects nodes of the XML annotation trees to (i) nodes of the interval trees and R-trees and (ii) ontology nodes. It is implemented in a directed labeled multigraph data structure we have developed, and serves as a general-purpose "labeled join index". The two primitive operations on the a-graph are:

$path(node1, node2)$: that returns a path between the given nodes

$connect(node1, node2, \ldots)$: that returns a connection subgraph intervening the given nodes.

Query Processing. Queries in *Graphitti* are essentially graph queries that resemble SPARQL expressions extended to handle (i) XQuery-like path expressions on *a-graphs*, (ii) type-specific predicates on interval trees, (iii) XQuery fragments to retrieve fragments of annotation. The result of a query can be (a) a collection of heterogeneous substructures (b) fragments of XML documents and (c) connection subgraphs. The query processor operates by separating subqueries that belong to the different types of data elements, finding a feasible order among these subqueries, and collating partial results from these subqueries into a set of type-extended connection subgraphs. At this point no further optimizations are performed. The user queries are posed through the GUI shown in Figure 3, and translate directly to a query expression.

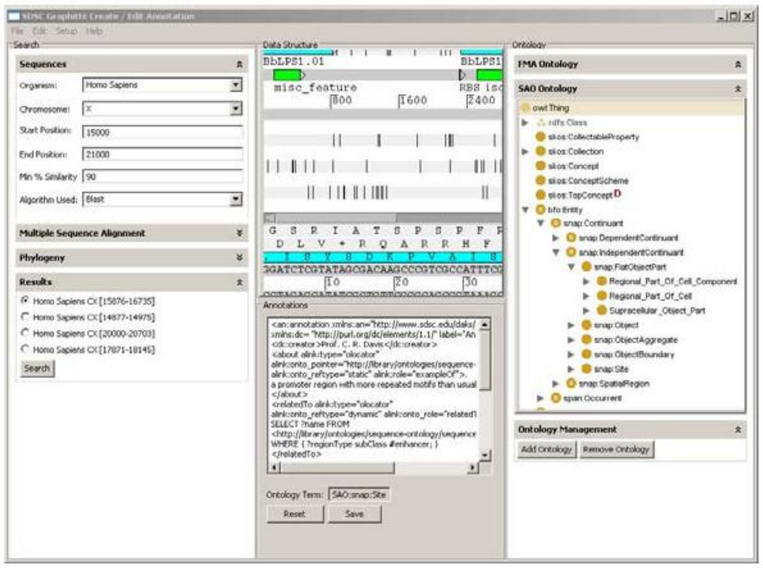
III. The Demonstration

The *Graphitti* system is a Java-based application that displays three tabbed panels for creating annotations, querying annotations and system administration. We describe the first two here with a virology and a neuroscience application. **Annotation Tab.** The annotation tab, shown in Fugure 2 consists of three panels.

The top part of the left panel presents a search window containing a menu button for each kind of data registered to the system for the Avian Influenza study. In the demonstration we will show DNA sequeces, RNA sequences, multiple sequence alignment structures, phylogenetic trees, interaction graphs and relational records – a representative subset of the types of data used in the study. The search window displays a form to query the specific data type. The search window in Figure 2 shows the query form for DNA sequences. The bottom part of the right panel shows a result window that displays search results. The user would browse through these results and locate a biological object she wants to annotate. Once the object is selected for annotation, the sequence can be dragged to the upper part of the central panel to perform the annotation.

The central panel has a number of menus for marking the substructures of different structures. For a sequence, the user chooses a linear interval marker and marks the start and end points of all subintervals that would be referred to by a single annotation. Similarly, there are block set markers for relational records. The lower part of the central panel shows the annotation structure. As the use searches for and drags each referent of the new to-be-committed annotation into the upper part of the central panel, the annotation structure at the bottom gets filled in. The XML text of the annotation content is written directly into the annotation structure at the bottom of the center panel.

The right panel of the annotation tab holds one or more ontologies (Figure 2 shows two). The user browses the ontology class structure with OntoQuest, selects a node, and then chooses "insert" from a menu to add an ontology reference. After the annotation gets created, the user may view it as an XML-structured object (and edit it if needed) before it is committed to the annotation storage. **Query Tab.** The query tab also has three panels – the leftmost panel is the *query formulation panel*, the central panel is the *result viewer panel* and the right panel is called the *correlated data viewer panel*. The typical query mode follows a "search, browse and explore" paradigm where the user first issues a query to find a number of data objects, and then browses through them to



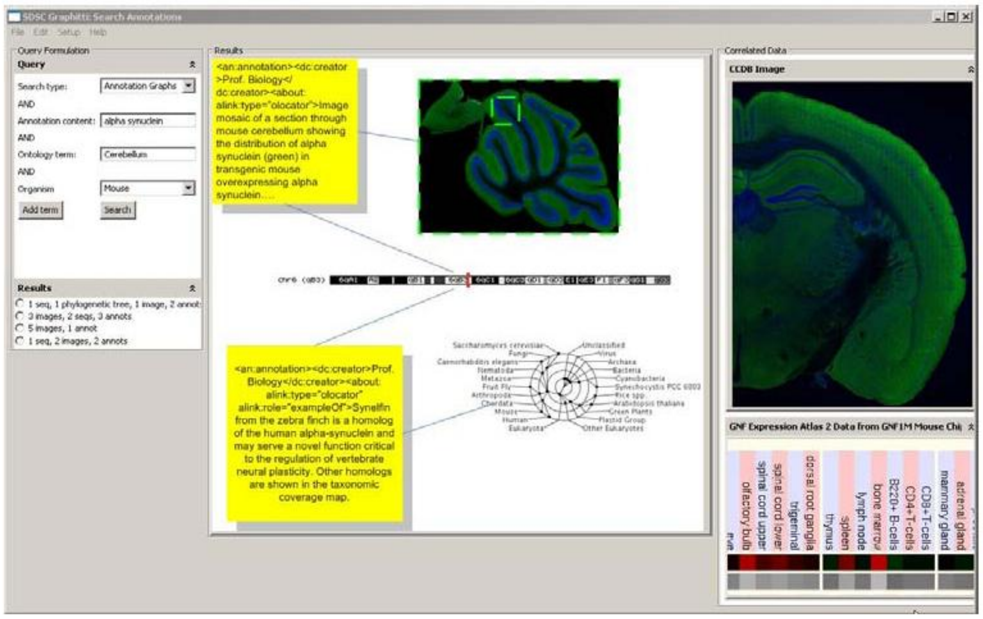
Organism: Homo Sapiens
Chromosome: X
Start Position: 15000
End Position: 21000
Min % Similarity 90
Algorithm Used: Blast

Homo Sapiens CX [15876-16735]
Homo Sapiens CX [14877-14975]
Homo Sapiens CX [20000-20703]
Homo Sapiens CX [17071-18145]

```
<an:annotation xmlns:an="http://www.sdsc.edu/daks/
xmlns:dc= "http://purl.org/dc/elements/1.1/" label="An
<dc:creator>Prof. C. R. Davis</dc:creator>
<about alink:type="olocator"
alink:onto_pointer="http://library/ontologies/sequence-
alink:onto_reftype="static" alink:role="exampleOf">.
a promoter region with more repeated motifs than usual
</about>
<relatedTo alink:type="olocator"
alink:onto_reftype="dynamic" alink:onto_role="related1
SELECT ?name FROM
<http://library/ontologies/sequence-ontology/sequence
WHERE { ?regionType subClass #enhancer; }
</relatedTo>
```

Ontology Term: SAO:snap:Site

Fig. 2. The annotation tab of *Graphitti* . The search form on the left is for sequences, and two ontologies used for annotation.



Search type: Annotation Graphs
AND
Annotation content: alpha synuclein
AND
Ontology term: Cerebellum
AND
Organism Mouse

1 seq, 1 phylogenetic tree, 1 image, 2 annot
3 images, 2 seqs, 3 annots
5 images, 1 annot
1 seq, 2 images, 2 annots

<an:annotation><dc:creator>Prof. Biology</dc:creator><about alink:type="olocator">Image mosaic of a section through mouse cerebellum showing the distribution of alpha synuclein (green) in transgenic mouse overexpressing alpha synuclein....

<an:annotation><dc:creator>Prof. Biology</dc:creator><about alink:type="olocator" alink:role="exampleOf">Synelfin from the zebra finch is a homolog of the human alpha-synuclein and may serve a novel function critical to the regulation of vertebrate neural plasticity. Other homologs are shown in the taxonomic coverage map.

Fig. 3. The query tab of *Graphitti* . The query produces an annotation graph consisting of a sequence, an image and a phylogenetic tree related to the protein α-synuclein. The correlated data shows another image and a μ-array result.

retrieve detailed metadata information stored in the relational system, and then explores associated objects (e.g., what other annotations have been made on *this* sequence).

The query formulation panel, designed for biologist users, allows one to specify whether the results of the query should be annotation contents, annotation referents, or annotation graphs. Figure 2 shows how the user may specify the query *find annotated sequences of all proteins belonging to an ontological class, where 4 consecutive non-overlapping intervals in the sequence has annotations having the keyword "protease" in each of them.* The user constructs an example annotation graph and places conditions on the nodes, node groups, and graphs. In the example shown the query graph consists of 5 sequence node, (one main sequence and 4 subsequences) and 4 annotation nodes. The edges from the subsequence nodes to the sequence node are labeled by the possible inter-sequence relationships known to the system. The condition on the annotation content ("protease" should be a substring) is specified on the left. The condition that the subsequences should be consecutive and disjoint are specified as graph constraints as shown on the left panel.

The results of the query are placed in the center panel which is organized in pages. In cases where subgraphs of the *a-graph* are returned as a result, each connected subgraph forms a result page. For each result object, the user may wish to see the entire witness structure (the annotations and the subsequences they annotate) – this is an example of what we call *correlated data viewing*, results of which appear on the right panel.

Once an *a-graph* fragment is returned as part of a query result, the user may wish to find related information about them. A typical request is to search for the ontology terms mapped to the objects in the result. The ontology appears in the right panel. One can then use the data object in the right panel to browse through further related results.

References

[1] W. G. Aref, M. Y. Eltabakh, and M. Ouzzani. bdbms - a database management system for biological data. In *CIDR*, pages 196–206, 2007.

[2] D. Bhagwat, L. Chiticariu, W. C. Tan, and G. Vijayvargiya. An annotation management system for relational databases. In *Proc. of 13th Int. Conf. VLDB*, pages 900–911. Morgan Kaufmann, 2004.

[3] P. Buneman, S. Khanna, and W.-C. Tan. On propagation of deletions and annotations through views. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 150–158, June 2002.

[4] L. Chen, M. E. Martone, A. Gupta, L. Fong, and M. Wong-Barnum. Ontoquest: Exploring ontological data made easy. In *Proc. 31st Int. Conf. on Very Large Databases (VLDB)*, pages 1183–1186, 2006.

[5] F. Geerts, A. Kementsiesidis, and D. Milano. MONDRIAN: Annotating and querying databases through colors and blocks. In *Proc. Int. Conf. Data Eng. (ICDE)*, page 82, April 2006.

[6] M. Gertz and K.-U. Sattler. Integrating scientific data through external, concept-based annotations. In *Proc. of the EEXTT*, volume 2590 of *Lecture Notes in Computer Science*, pages 220–240. Springer, 2002.

[7] S. Murthy, D. Maier, and L. M. L. Delcambre. Querying bi-level information. In *Proc. of the 7th Int. Workshop on the Web and Databases (WebDB)*, pages 7–12, 2004.

[8] D. Srivastava and Y. Velegrakis. Intensional associations between data and metadata. In *Proc. of the 2007 ACM SIGMOD Int. Conf. on Management of Data*, pages 401–412, New York, NY, USA, 2007. ACM Press.