SEMANTA: AN ONTOLOGY DRIVEN SEMANTIC LINK ANALYSIS FRAMEWORK

by

MULLAI T. SHANMUHAN

(Under the Direction of I.BUDAK ARPINAR)

ABSTRACT

In today's Web, there is an overwhelming amount of information, but it is still very hard for users to locate useful information such as *semantic links* between different entities. Semantic links are transitive relations between entities and concepts scattered among different knowledge and information sources. In Semanta we provide two kinds of querying capabilities: *entity based queries* – to find semantic links between any two entities, and *relation based queries* – to find entities that are related to a given entity through a specific relationship, which may be user-defined. Resembling human-thinking, Semanta uses background information captured as instance and abstract knowledge (i.e., an ontology) in RDF and RDFS, respectively, to further unearth hidden relationships in dynamic information resources consisting of XML documents. The Semanta API and a prototype, built over it are discussed, along with algorithms for gathering *hints* in the ontology layers and using them to look for semantic links.

INDEX WORDS: Semantic Web, Information Retrieval, Link Analysis

SEMANTA: AN ONTOLOGY DRIVEN SEMANTIC LINK ANALYSIS FRAMEWORK

by

MULLAI T. SHANMUHAN

B.E. Anna University, India, 1998

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTERS OF SCIENCE

ATHENS, GEORGIA

2003

SEMANTA: AN ONTOLOGY DRIVEN SEMANTIC LINK ANALYSIS FRAMEWORK

by

MULLAI T. SHANMUHAN

Major Professor:	I. Budak Arpinar
Committee:	John Miller Khaled Rasheed

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
May 2003

DEDICATION

To the Light of my life, Subash Kalbarga.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. I. Budak Arpinar, for his invaluable guidance and support. I would also like to thank Dr. John Miller and Dr. Khaled Rasheed for agreeing to be a part of my committee. Thanks to Boanarges Aleman-Meza for reviewing the document. Last but not the least I would like to thank Subash and Thamarai for their moral support throughout the course of this degree.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

1.1 Motivation

Today a massive amount of data is available on the Internet, as well as in private organizational databases. The volume of this data is increasing continuously. However, despite the abundance of information, most of it cannot be used effectively for decision-making purposes causing knowledge starvation. The focus of contemporary data and information retrieval systems has been to provide efficient support for the querying and retrieval of data. Significant academic and industrial research has now transitioned to mainstream search engines, such as AltaVista, Google, and Teoma. There has also been noteworthy progress in metadata extraction, which involves recognition of entities such as names of persons, locations, and in some cases, domain specific attributes of entities.

Due to the increasing move from data to knowledge, and the increasing popularity of the vision of the Semantic Web [LHL01], there is significant interest and ongoing work, in automatically extracting and representing the metadata as semantic annotations to documents and services on the Web ([SFJCM02], [HSK02], [Dill03]). Several communities such as the Gene Ontology Consortium, Federal Aviation Administration (Aviation Ontology), Molecular Biology Ontology Working Group, Stanford University's Knowledge Systems Lab (Enterprise Ontology), are also coming together, to effectively conceptualize the domain knowledge, and enable standards for exchanging, managing and integrating data more efficiently. Research in the Semantic Web has also spawned several commercially viable products through companies such as Semagix [Sem] and Ontoprise [Ont] to name a few.

Given these developments, the stage is now set for the next generation of technologies in information retrieval, which will facilitate getting actionable knowledge and information from massive data sources thereby assisting in information analysis. Many users try to analyze information by either browsing the information space, or using a search engine. Search engine based systems only locate documents based on keywords or key phrases. These approaches are not very representative of what the user actually wants. Therefore, most of the retrieved documents are either irrelevant, or contain the information buried deep within other data. The onus is then on the user, who must decide, which of the retrieved documents are relevant, and then use their mental model, of the information they are looking for, in order to obtain the relevant information.

One of the main goals of this work, is to ease the process of analyzing across different sources of structured or semi-structured data and to provide the user with dependable links, that can be used in the decision making process. Specifically, we wish to go beyond searching for documents given a list of keywords, or analyzing documents to identify entities. The work described in this thesis deals primarily with the semantic link discovery aspect of information analysis. It will enable the user to semantically process, the many layers of knowledge and information and uncover previously unknown and potentially interesting links ([W75], [SAK03]). These links or complex relationships lend meaning to information, making it understandable and actionable, and provide new and possibly unexpected insights.

The work discussed in this thesis, Semanta, proposes to support querying, discovering, and retrieving *semantic links* between the entities. Semantic link discovery involves unearthing the following types of knowledge: *semantic links* – meaningful relationships between two objects, and, *related entities* – discovering objects linked to a given object through a specific relationship. An example of finding semantic links is discovering possible ways in which two persons might be connected. Finding all co-writers of a given author is an example for finding related entities.

Semanta is a framework for discovering implicit and explicit links between entities by exploiting the underlying semantics using ontologies. An explicit link between the entities is a direct relationship between them and it is already present within an ontology or resource. An implicit link is one that can be found, only by gathering and extrapolating information from multiple ontologies or resources. Links thus discovered are treated as knowledge and can be incorporated into the framework.

1.2 Challenges

Developing a system that discovers semantic links, rather than finding documents or data is challenging for several reasons. The goal of looking for links will require new forms of processing data and relevant knowledge, and associated techniques of creating and maintaining a variety of relationships. Each document may describe (and hence be annotated with) many entities. The number of links i.e., paths connecting entities directly or through a knowledge base, however, is vastly larger. Emphasis should also be on finding relevant relationships efficiently and in a scalable manner. In the absence of explicit links between entities, it is essential to look for indirect relationships that maybe hidden and will need to be found using existing links.

To achieve the goal of finding semantic links, it is essential to model an information-system that closely reflects the real world. This model has to be able to compactly capture the rudimentary concepts of a given domain and should also enable insertions and updates at later stages. While these domain concepts form the building blocks in representing the knowledge of a domain, it will be prohibitively expensive to rely on them, in entirety, to build a sound information system. Semi-structured data when included in the system will greatly enhance the richness of the system and help in effectively modeling a domain's information system. While perusing the domain concepts and the semi-structured data for semantic links it is imperative to make use of the knowledge from domain concepts in traversing the semi-structured data.

The semantic links obtained will have to be presented to the user in order to aid him/her in the final decision making process. In the quest for finding links, it is also possible to find too many of them between the entities. Therefore, it is also important to locate interesting and meaningful relations and to rank them before presenting to the user. The parameters involved in ranking the discovered links could be correctness of the results, trustworthiness of resources used, and level of pertinence to the user. However, this issue is not addressed in detail in this thesis and is mentioned here for completeness.

1.3 Example Queries

In this sub-section we discuss a few example queries that are addressed in Semanta. These queries are dealt with in more detail in the rest of the document.

Consider a fictitious example, where a journalist tries to find possible relationships between a political organization, called 'Party X' and the 'Energy Sector'. By perusing a multitude of profiles of people belonging to the party and looking at companies' sponsors, it can be seen that there exists at least a few people in key roles in the Party who have ties with companies in the energy sector. For example, an important fundraiser in the Party X may have ties to some energy companies. And some members might have served as board members in leading energy companies in the past.

Consider another query for finding people well known to Saddam Hussein. This would classify as an example for finding related entities (people) of a given entity (Saddam Hussein) through a relationship, which can be named 'positive associate'. A positive associate here can be seen as a collection of predefined relationships such as 'mother-of', 'sibling-of', 'works-for', 'funded-by', etc. Finding related entities can thus be seen as classifying a set of predefined relationships into a more complex relationship, and looking for entities related to the given entity

through such a relationship. Several other examples of link discovery applications can be found in intelligence analysis (e.g., finding links between immigrants, front businesses, and currency transactions to locate money laundering operations), genetics (e.g., discovering semantic links between proteins and genes), or pharmaceutical research (e.g., finding counter effects between different drugs).

1.4 Contributions

The information system of Semanta also known as the knowledge store is built to closely reflect the real world, by utilizing the advancements in the field of ontologies. An ontology consists of definitional components (schema level) and assertional components (instance level) which includes explicit knowledge or facts. The relevant work involves development of domain independent (e.g., Wordnet) and domain specific (e.g., GO [AL02], UMLS [NLM03]) nomenclatures, taxonomies and ontologies. This is complemented by technologies to create and maintain topic or enterprise specific ontologies [G02].

While ontologies are the building blocks in knowledge representation systems, XML documents are used complementarily to represent more up-to-date and dynamic information and are incorporated in Semanta to completely model a domain's information system. Significant advances in automatic and semi-automatic data extraction using wrappers [KT02] can be exploited, to add semi-structured data to enrich the knowledge store. Although we do not address information extraction techniques from ordinary Web documents, Semanta can be used with those systems able to generate meta-data in XML. Most importantly we traverse links in dynamic resources and ontology layers in a correlated way which differentiates our work from other approaches where the knowledge base is built once and does not use emerging fresh information.

A query submitted by the user sets Semanta looking for semantic links in the ontology. If no links exist, hints are gathered and are used to generate queries for XML documents. Using the

generated queries and by perusing the XML documents, direct and indirect links are found and presented to the user.

The main contributions made by Semanta can be summarized as follows:

- The design of the knowledge store of Semanta as a three layer architecture is a novel approach which ensures that it keeps pace with the rapidly evolving data. It also reduces the expense of maintaining all known information in the Ontology layers as knowledge by categorizing information as background knowledge and dynamic data. The background or domain knowledge once created by consulting with the experts in the domain is compact and will need little future updates. The Information Source layer on the other hand, can include new XML documents and more frequent updates.
- Semanta classifies user queries for analyzing semantic links as *entity based queries* and *relation based queries*. By providing this classification, Semanta enriches the quality of queries and addresses the issues involved in the next generation of information retrieval tools. Semanta also introduces the concept of using 'relationship ontology' to define complex relations for relation based queries. By treating relationships as first class objects in the queries, richer and interesting information can be discovered.
- The process of looking for semantic links for the queries obtained from the user involves using the information gleaned from the Ontology layers while searching for links in the Information Source layer domain knowledge. Also, to curtail the process of searching for links from exploding, heuristics such as directed breadth-first search and interactive deepening are presented to filter paths and present links which will be more useful to the user.

The remainder of the thesis is organized as follows. Chapter 2 discusses other research work related to Semanta. In Chapter 3, the semantic network, the knowledge store of Semanta is discussed. Chapter 4 discusses the different kinds of queries addressed by Semanta. The system architecture and implementation details are presented in Chapter 5. In Chapter 6, the process of

finding semantic links in Semanta is explained in more detail with sample scenarios. And finally, conclusions and future work are presented in Chapter 7.

CHAPTER 2

RELATED WORK

The work done in Semanta encompasses semantic web, knowledge discovery, and information retrieval research areas. The related work to Semanta are thus based on these areas. The related projects are detailed with the similarities and differences with respect to Semanta under these categories.

Although Semanta involves information retrieval, it differs from traditional keyword based search engines. First of all keyword-based search engines do not attempt to detect (even explicit) links of the keywords across documents. They only try to detect presence of all keywords within the same document. Semanta on the other hand tries to detect not only explicit but also implicit links between entities across documents. Also, Semanta attempts to look at entities both in isolation and in their entirety, which results in finding richer and more meaningful information.

InfoQuilt [STP01] is a framework for human assisted knowledge discovery. It extends support for semantics by supporting computations involving user-defined relationships, for instance causal relationships. It also aims to support human-assisted knowledge discovery by allowing users to pose questions that involve complex and hypothetical relationships amongst concepts both within and across domains. In InfoQuilt the onus of defining relationships is placed on the user. However in Semanta, only existing transitive links in ontology and information source layers (i.e. XML documents) are found.

MREF (Metadata REFerence Links) [SS98] allow logical relationships between Web artifacts and are specified as RDF statements. They can represent information requests involving keyword-based, attribute-based and content-based specifications involving various types of

metadata and are treated as virtual objects in the InfoQuilt system. In this way, HREF links in Web documents can be annotated with some meta-data.

SHOE [HH00] uses XML-like tags and advanced artificial intelligence technology to depart from traditional keyword-based search engines. The process of searching using SHOE first involves selecting a suitable ontology and providing some of the values for the properties. Theses values are then used to trigger a search in the knowledge-base. SHOE provides for a more meaningful search than contemporary keyword-based search engines. The user is limited to search within a given ontology and its subclasses whereas Semanta tries to find the documents that relate to a given ontology and find the possible links across ontologies.

OntoBroker [DSMR98] uses an ontology to extract reason and generate metadata in the Web. It has a broker architecture with the following three core elements: a query interface for formulating queries, an inference engine to derive answers and a web-crawler used to collect the required knowledge from the Web. The OntoBroker relies on AI techniques for creating the ontology as well as for inferring.

OntoSeek [GMV99] is a system designed for content-based information retrieval from online yellow pages and product catalogs. OntoSeek combines an ontology-driven content-matching mechanism with moderately expressive representation formalism. The system relies on a large linguistic ontology called “Sensus” to perform the match between queries and data. It assumes that the information encoding and retrieval processes will involve a degree of interactivity with a human user.

Structured Argumentations for Analysis (SEAS) [LHR00] is a system to aid intelligence analysts in seeking and interpreting evidence pertaining to analytic tasks. It is based on structured argumentation, a methodology where analysts record their reasoning in structured arguments, relative to argument templates that pose a set of hierarchically related multiple choice questions that are designed to address a specific analytic task. Through its graphical visualizations of

arguments and templates, they can be understood at both summary and detailed levels and compared and contrasted with other arguments and templates.

Table 2.1 Comparison of Works Related to Semanta

Tool	Techniques	Data Format	Semantic Queries
InfoQuilt	Ontology	Data extracted using wrappers and extractors	Explores a hypothetical relationship by enabling the user to break it down into multiple IScapes. e.g. 'Does nuclear tests cause earthquakes?'
SHOE	Ontology, AI	SHOE annotated Web pages	On selecting the ontologies of interest, user fills the fields of interest through a GUI, which is converted to a Parka query. e.g. 'Find articles on SHOE by Heflin'
OntoBroker	Ontology, AI	Documents that are annotated by existing ontologies	User inputs Object, Class, Attribute and Value fields of a selected ontology through a GUI e.g. 'Find out about the research subjects of a researcher named Smith or Feather'
SEAS	Structured argumentation	Data is stored as argument templates, arguments, and situation descriptors	Analyst authors an argument template made up of a hierarchy of multiple choice questions. Based on the answers to theses questions, SEAS indicates the result which might range from green(OK) to red(alert) e.g. 'Assessing the outlook for project success based on the current situation'
RHO	Ontology, Graph Theory	RDF	User should provide the URIs of the entities in a query for which paths are found. e.g. 'Retrieve all passengers associated with a terrorist organization'

The LINDI (Linking Information for Novel Discoveries and Insight) project [RHF02] aims to develop a text data mining system, for linking information and enabling discoveries. The main goal is to help automated discovery of new information from large text collections. As a

step towards the goal of text mining, they are developing empirical algorithms for semantic analysis of natural language text.

The Rho operator [AS03] is concurrent work at the LSDIS lab, which enables querying for semantic associations. It defines a set of associations that can be identified and uses RDF query languages and graph algorithms to find associations. However, it does not address searching through dynamic information resources and relation based queries vis-à-vis Semanta. It does address maintaining large knowledge bases and scalability issues.

The related work just discussed is summarized in Table 2.1 based on the techniques used in processing the query, format of the information store on which the queries are performed and the nature of the queries processed by the system.

CHAPTER 3

SEMANTIC NETWORK

A semantic network is the underlying knowledge and information store of Semanta, and it encapsulates domain knowledge, dynamic and structured data. In other words, it represents the framework from which links will be discovered and forms the core of Semanta. This chapter discusses in detail the requirements and issues associated with the semantic network. The implementation details are also discussed for each layer.

The semantic network consists of three layers and links connecting the nodes across the layers. The three layers are: Class Base (CB) Layer, Object Base (OB) Layer, and Information Source (IS) Layer. The first two layers constitute the domain knowledge, and are also referred to as the Ontology layers. The last layer consists of documents that are characterized by structure and contains dynamic information. The three layers of the semantic network are modeled after real-life decision-making processes. When trying to reach a decision, we normally process the information that is already known to us (background information) and then accrue more current details from other resources and process them subsequently. In Semanta, the Ontology Layers capture the background information and the Information Source layer represents more up-to-date information.

In the semantic network, ontologies involve both high-level concepts (i.e., classes) and their instances (i.e., objects) and their inter-relationships. The design choice to maintain the knowledge base as a two-layer classification was made for the following reasons – First, by segregating the instances that validate the domain knowledge, there is a higher degree of adaptability induced in the knowledge base, i.e. instances can be validated and accrued without having to disturb the core knowledge base, in order to reflect the evolving world view.

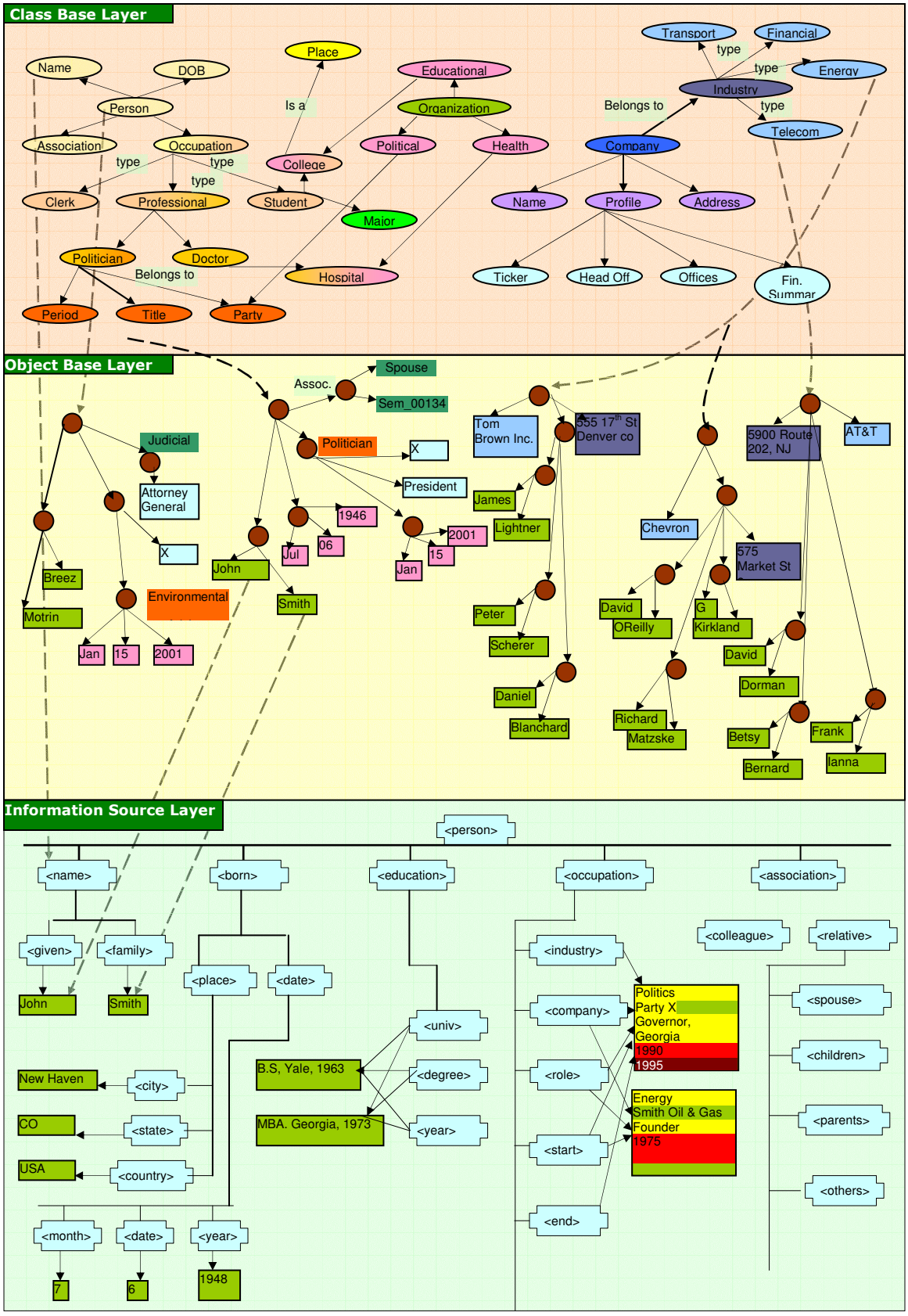


Figure 3.1 Semantic Network – Knowledge Store of Semanta

Second, while finding links between entities, the Object Base and Information Source layers will both be gleaned independently for possible links. Figure 3.1 depicts the three layers that form the core of Semanta which are discussed in detail in the rest of this chapter.

3.1 Class Base Layer

This layer encapsulates information about classes and relationships between the entities in all the ontologies known to Semanta. Links exist between this layer and the Object Base and Information Source Layers. Nodes in this layer represent classes and links between nodes represent relationships. A class could be a very basic concept in a domain – e.g., a person's name. A class need not necessarily be an indivisible unit – 'Name' could comprise of other classes such as 'Given', 'Family', and 'Nickname'.

Each node has information about the name of the class, the attributes that belong to this class, and a set of pointers. The pointers link to: the location where the class is defined, other classes, special relationship nodes, and nodes in other layers.

Relationships between classes are represented by links with arrows between nodes. Relationships can either be predefined or user-defined. Predefined relationships between the nodes include parent-child, container relationships etc., and are common across domains. User-defined relationships may vary from domain to domain and include relations such as mother-of, works-for, etc. Each relationship will have its name, pointers to classes that it relates to and instances that validate the relationship.

Class Base Layer is implemented as a collection of Resource Description Framework Schema (RDFS) files [RDF]. RDFS enables communities to formally define properties in terms of the classes of resource so that different RDF files using a particular RDFS can share the same vocabulary. RDF models the real-word with a set of statements [RDF]. A statement is a triplet constituted by subject, property, and object. Every statement is a model of how a subject is related to an object by a specific property. RDF treats all subjects, properties and objects as

resources, thus enabling a richer expression of knowledge. RDF uses a property-based approach in defining ontologies, thereby enabling easy extension of existing vocabularies to newer ontologies. Figure 3.2 shows an RDF statement represented as a graph, and as a triple. Note that the figure has favored relative URIs over absolute URIs for brevity in representing resources. The Class Base layer nodes and arcs, in the figure, are shown with dotted lines, in the graph model, to demonstrate their relationship with the Object Base layer.

Each schema file in the Class Base layer encapsulates the knowledge of a particular domain. *Person.rdfs* contains classes and relationships that are closely linked to a person – information such as name, occupation, roles and relations. The schema named *industry.rdfs* consists of classes and relationships with respect to the industry – sectors, company profile, financial profile, etc. The vocabulary used across the files in the Class Base layer is assumed to be uniform, for e.g., the 'occupation' class will have the same semantics across different schema files.

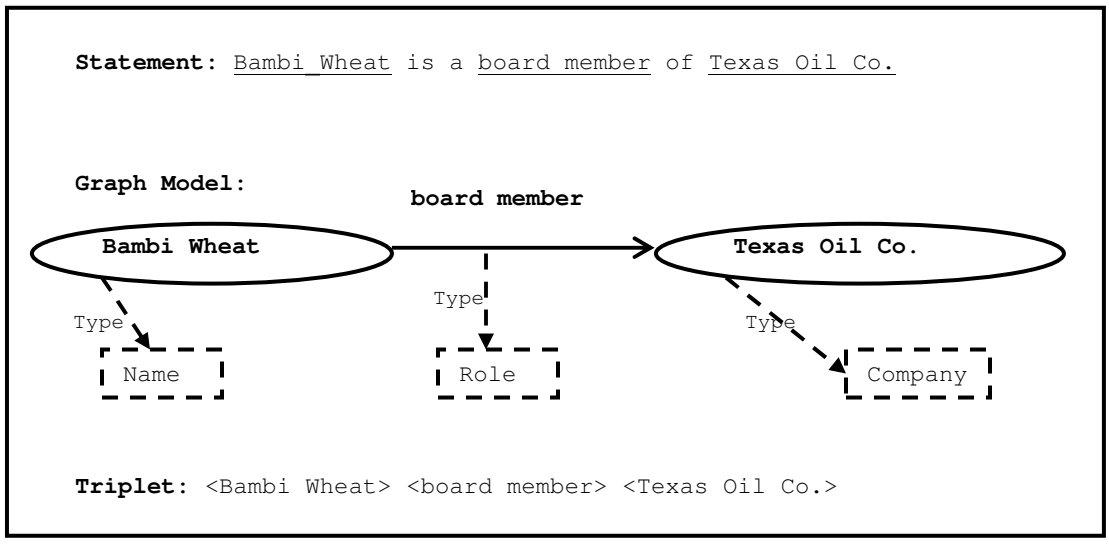


Figure 3.2 RDF Statement Represented as a Graph and as a Triplet

3.2 Object Base Layer

This layer contains all instances of nodes in the Class Base Layer known to Semanta. This layer, together with the information in the Class Base Layer, constitutes the ontology of Semanta. Each node in this layer contains an id to uniquely identify the node, instance values of classes and optional attribute values of classes. Apart from this, there are also pointers to the Class Base and Information Source Layers. In Figure 3.1, the Object Base layer nodes are represented in rectangular boxes. The blank round nodes that are not labeled represent an instance of a class node in the Class Base layer. The blank node contains pointers to instance values of the attributes of a class. Pointers exist from the Class Base layer to the blank nodes, shown by dashed links in the Figure 3.1, and they identify the class to which the instance belongs.

The nodes of Object Base Layer represent the instances of ontology and are defined using Resource Description Framework (RDF). The Object Base layer consists of a collection of RDF files, with such statements, modeling the real world.

3.3 Information Source Layer

The Information Source layer complements the semantic network, by providing an encapsulation for data that can be categorized as structured and fresh. The Information Source layer is gleaned to provide richer or more current information, when the user queries Semanta. This is due to the impracticality of extracting entities and relations from all information resources in a very frequent and efficient way. The nodes in this layer consist of tags and their corresponding instance values. Every instance value or literal present in the Information Source layer will be linked to the corresponding value node in this layer. Similarly, nodes in the Class Base layer will also be linked to the corresponding tag nodes in this layer. In Figure 3.1 boxes such as ‘person’, ‘name’, ‘education’, denote the tag nodes and rectangular boxes such as ‘John’, ‘Smith’, and ‘USA’ denote the value nodes.

Information Source Layer is implemented as a collection of XML files that share the same vocabulary with the Class Base and Object Base layers. The inherent structure and markups of XML files are leveraged, in order to achieve semantic dimension. The Information Source layer is implicitly linked to the other layers by the markups and the structure of the documents. For instance, the information presented in the RDF statement in Figure 3.2 can be seen to also be incorporated in Table 3.1 which represents an XML document in the Information Source layer. It can be seen that the Information Source layer is linked to the ontology layers by sharing the same vocabulary, for instance, *industry* in the Information Source layer and Class Base layer will mean the same.

Table 3.1 XML Document in the Information Source Layer

```
<person>
   <name>
      <first> Bambi </first>
      <last>  Wheat </last>
   </name>

   <birth>
      <date>… </date>
      <place>… </place>
   </birth>

   <occupation>
      <industry> Energy </industry>
      <company> Texas Oil Co. </company>
      <role>   Member, Board of Directors </role>
      <duration>……</duration>
   </occupation>
   ……
   ……
   <occupation>
      <industry> Finance </industry>
      <company> Charles Financial Corporation </company>
        <role> Member, Board of Directors     </role>
        <duration>…… </duration>
   </occupation>
</person>
```

3.4 Inter-Layer Links

Class-Object links connect classes to objects of the ontology. Instances belonging to classes of user-defined relationships are also connected using these links. In Figure 3.1, class-object links are represented by dashed links between the Class Base and Object Base layers – 'Person' node

from Class Base layer is linked to a blank node denoting the instance of 'Breeze Motrin' in the Object Base layer. Class-IS links **c**onnect the tag nodes of documents in IS layer to the Class Base layer. The dashed links between the Class Base and Information Source layers represent these links in Figure 3.1 – 'Name' node in Class Base layer is linked to the 'name' tag in the Information Source layer. Object-IS links connect every value or literal in the Object Base layer to value nodes in the Information Source layer. The dashed links in Figure 3.1 denote Object-IS links – a blank node denoting the name 'John Smith' in the Object Base layer can be seen linked to the 'John Smith' value nodes in the Information Source layer. Note that only a few of the inter-layer links are shown in the figure in order to reduce visual complexity.

CHAPTER 4

SEMANTIC LINK QUERIES IN SEMANTA

The search and discovery of semantic links deeply buried in the massive amount of data is a very complex problem. A simple case of link could be a direct link that is present in the knowledge store of Semanta. On the other hand links might have to be extrapolated based on the existing links and/or by leveraging the knowledge of domain concepts while perusing the XML documents. This chapter defines 'semantic links' in the realm of Semanta and classifies the queries that are addressed by Semanta and discusses them in detail in the realm of Semanta.

At the outset we would like to make the following distinctions: finding a semantic link/path or finding relations will be used interchangeably and refers to finding links between entities. Finding related entities refers to finding entities that are related to a given entity through a specific kind of relation. And, finally querying will be used more broadly to refer to finding links and/or finding related entities.

A semantic link can be formally defined as a set of relations connecting a set of nodes across the three layers of the knowledge store. It is the end result of a query in Semanta. In Figure 4.1 different types of Semantic links obtained as a result of queries are illustrated. Nodes with dotted lines represent the entities of interest to the user. Links with edges represent the semantic links found by Semanta between the entities. The three layers are denoted in the figure by lines between them. The elliptical nodes and links in the first layer denote the Class Base (CB) Layer. Rectangular nodes and links in the second layer denote the Object Base (OB) Layer. Rectangular nodes in the third layer represent the Information Source (IS) Layer.

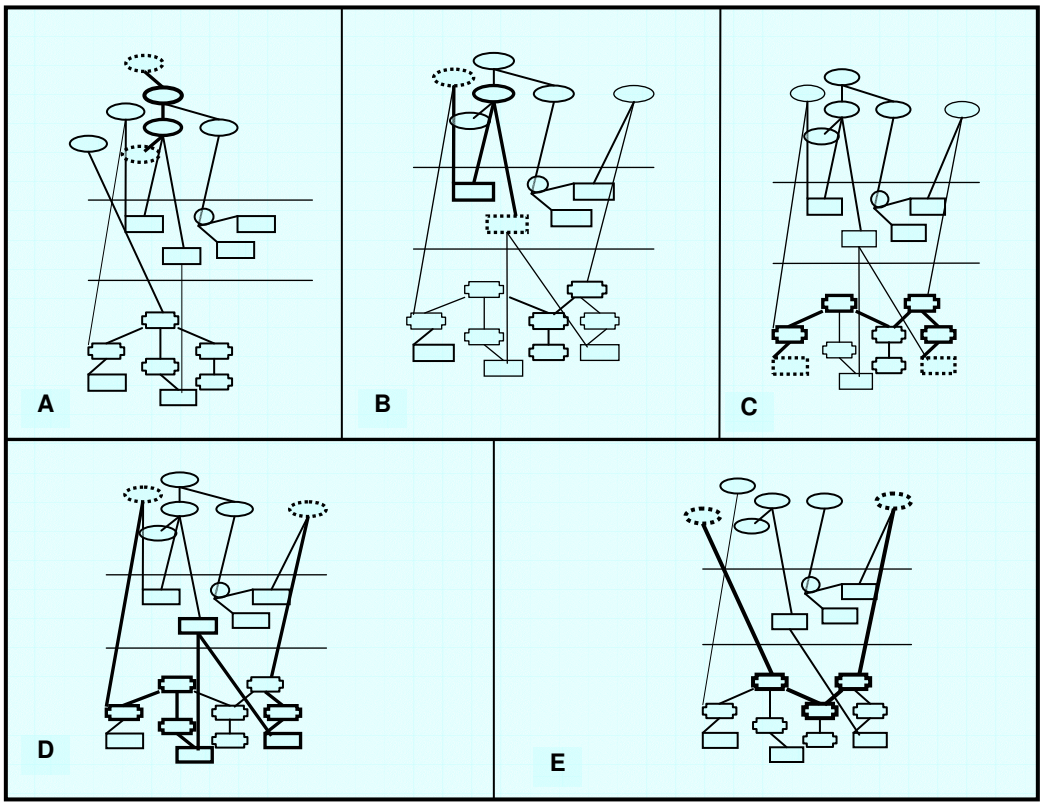


Figure 4.1 Types of Semantic Links

The semantic links in Figure 4.1.A represent semantic links that exist between entities in the Class Base Layer alone. Figure 4.1.B illustrates semantic links that spans the Class Base and Object Base Layers. Figure 4.1.C denotes semantic links that exist within the Information Source Layer. Figure 4.1.D and Figure 4.1.E illustrates links that span across all the Ontology layers and Information Source layer. Figure 4.1.D denotes the fact that each layer might be visited repeatedly to obtain the end semantic link – the links initially start from the Ontology layers, visits IS layer, returns to the Ontology layers and then again visits Information Source layer before finally ending the path in the Class Base layer. The prototype currently handles cases A, B,C and E.

The example shown in Figure 4.2 demonstrates a semantic link between 'Energy' and 'Party X' that spans across the three layers. In this example the link exists because of the presence of a person with occupations in both the Energy sector and the Party X. The collection of paths in

the 'semantic link' is identified by dashed lines and the nodes belonging to the semantic link between the input entities are colored gray. Elliptical nodes belong to the Class Base Layer. Rounded rectangles belong to the Object Base Layer and the rest of the nodes belong to the Information Source layer.

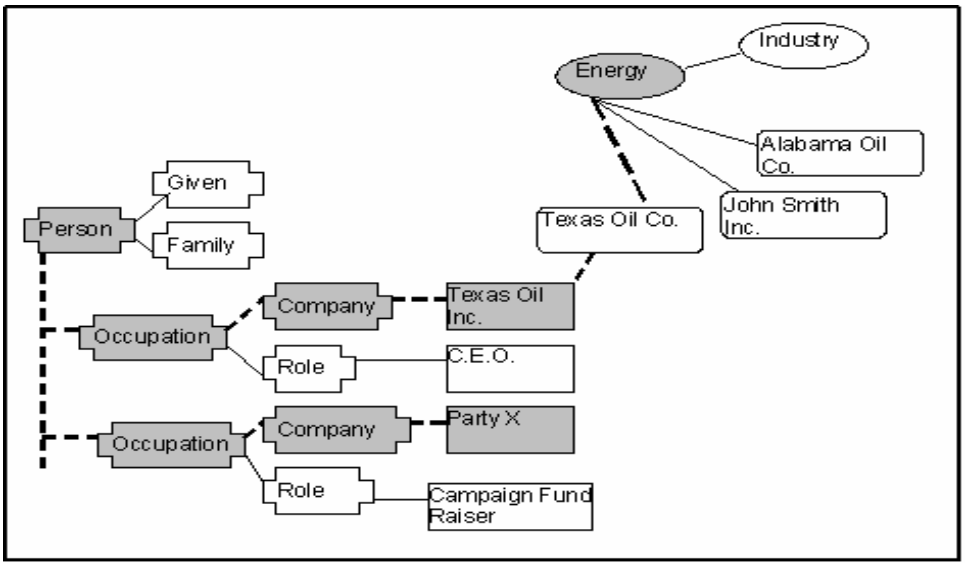


Figure 4.2 Example Demonstrating a Semantic Link across the three Layers

4.1 Entity Based Queries

Entity based queries refers to cases where the user is interested in finding links between any two entities. Entities in this context may refer to a class, property/attribute, attribute-value, tag or literal in the semantic network. This chapter discusses such queries in detail with examples. Let e_1 and e_2 refer to the entities provided by the user, for which we need to find links in Semanta. Based on the category of the inputs, they can be classified into the following three types:

i) Type 1: class/property, class/property,

ii) Type 2: class/property, literal and

iii) Type 3: literal, literal.

Type 1 category, consists of cases where both entities belong to class, or property category or when one of them belongs to the class and the other to the property category. The presence of a path between the entities in this case could be the result of a relationship between them that can be classified as:

i) Class-attribute relationship,

ii) Parent-child relationship,

iii) Linked classes relationship,

iv) Co-classes relationship.

A node refers to a class node in the Class Base layer or an instance node in the Object Base layer or an element in the Information Source layer. The entities e_1, e_2 are said to be associated with nodes n_1, n_2 respectively based on the node to which they belong in the semantic network. And, the task of finding a link between e_1 and e_2 can be extended to find a path between n_1 and $n2$.

A '*class-attribute*' relationship exists between nodes n_1 and n_2 when n_2 is either an attribute or a class that is contained in n_1. n_1 could be several number of hops (nHops) deeply contained in n2. Consider the 'address' class node whose 'city' attribute has 'Los Angeles' as value. Then, 'address' and 'Los Angeles' are said to share a class-attribute relationship that is one hop away. A '*parent-child*' relationship involves a node n1 that is a grandparent (grandchild) of a node n_2, removed by nHops generations (or levels). A hierarchical relationship such as 'Industry—Telecom—Wireless Communications Services' shares a parent-child relationship between 'Industry' and 'Wireless Communications Services' that is two hops away.

'*Linked classes*' relationship refers to nodes that are linked by relationships, which are predefined in the Class Base layer. For instance, a node 'Person' might be linked to an 'Organization' through a property 'member-of', that has 'Person' as its domain, 'Organization' as its range and can be represented by the triple, "Person <member-of> Organization". A linked class relationship might be restricted by specifying the maximum number of links allowed - a

linked class relationship such as “Person <member-of> Organization <funded-by> Organization” is related by 2 hops, if there exists instances validating such a relationship.

‘*Co-classes*’ relationships provide a broader and less specific relationship between nodes. They refer to classes that are related in a similar manner. For instance, consider the triples, “A<prop-p1>B”, “D<prop-p1>B” and that there exists no relationship between node A and node D. Based on the above triplets, node A and node D are said to be related by the co-classes relationship as they share the same relationship (prop-p1) with the same node (node B). In a real world example, it can be seen that two people working for the same organization are in a way related even though they might not have any direct links, because of their common property of being employed by the same employer.

While looking for semantic links between any two entities, Semanta recursively tries to ascertain nodes that are linked through any of these categories of relationships. By looking for these relationships the directionality of the links are no longer important as Semanta looks for all nodes linked to it irrespective of the direction of the link. Also, a path found between two entities might have several segments, where each segment might belong to a different category of relationship.

A path found, based on the aforementioned categories might exist between nodes in the Class Base layer alone without involving nodes in the Object Base layer. Such a path need not be very interesting as it might be intuitive and already known to the user. Therefore, a path thus found, is further enriched by validating with the instances in the Object Base layer, if any exist. For instance, a user is interested in finding possible information about availability of minorities in the judicial department. At the Class Base layer, there exists information, which denotes at a broad level, that people work for the judicial department. Semanta however tries to find instances that validate such a claim and then presents it to the user.

A *Type 2* category of input consists of a class or property input and a literal input. Finding a link for this category will have to involve both the Class Base and Object Base layers.

Consider an example for this category – a query to find relationships between the 'Energy Sector' and the 'Party X', if any such relationship exists, in the knowledge store. The process of finding semantic links for this query is detailed in Chapter 6.

When both inputs are literals they fall under the *Type 3* category. The literal values might either be property values or class values in the Object Base layer or text elements in the Information Source layer. For literals that belong to the Object Base layer, the corresponding class and property nodes in the Class Base layer are ascertained and it now suffices to find paths between these nodes. A query to find links between 'Liming' and 'Robert', where both are literal values will come under this category, and finding the links between these entities is detailed in Chapter 6.

4.2 Relation Based Queries

Relations based queries in Semanta involve queries in which, the user is interested in finding entities that are related to a given entity through a specified relationship. If a user specifies a relationship in the query it is important to find other relevant relations for the given relation. We propose a novel idea of using a *relationship ontology* which involves relations as first class objects and relations among them. Here we give initial results of this idea but full-fledged specification of a relationship ontology is out of the scope of this work.

The relationship specified by the user is a complex relationship defined as part of the relationship ontology and is not directly present in the Object Base and/or the Information Source layers of Semanta. This section defines and discusses the use of complex relationship in finding related entities.

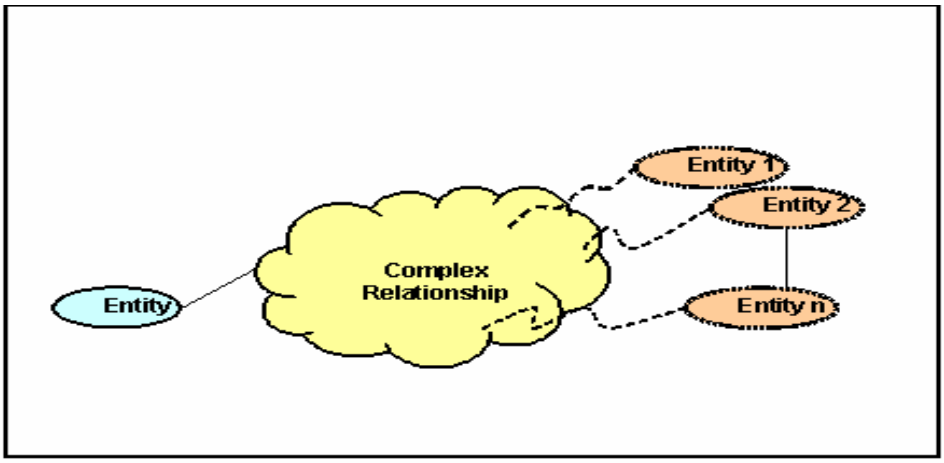


Figure 4.3 Entities Related through a Complex Relationship

Figure 4.3 depicts a relation based query, where the inputs will be 'Entity' and 'Complex Relationship'. Semanta finds paths with entities (Entity1, Entity2, … Entity n) that validate such a query. The paths and entities to be found are shown by dotted lines in the figure. 'Entity' can involve a class or instance from the Class Base or Object Base Layer. A complex relation is defined as a specialized collection of member relations with some structural restrictions. These member relations, in turn, can either be complex relations or relations defined in the Class Base Layer.

A 'complex' relation is broadly categorized into the following categories: OR complex relation, AND complex relation and Template complex relation. These categories are detailed in the following sub-sections.

4.2.1 OR Complex Relation

An OR complex relation is a group of relations with no significance in the order of the member relations. It also does not require that all members be present in the resulting sub-graph. This relation can be seen as an implementation of the OR relation among the member relations and is defined using the RDF container *rdf:Bag.* An example of such a complex relation is defined and illustrated in Figure 4.4.

This example defines a complex relation ‘positive-associate’ that is made up of the relations, ‘mother-of’, ‘father-of’, ‘brother-of’, etc. When Semanta is queried with the input {‘Tom Drake’, ‘positive-associate’}, it returns sub-graphs from the Class Base and Object Base layers that contain instances validating the relations for ‘Tom Drake’. The resulting sub-graphs might have paths that link nodes through one or more of the member relations.

```
<rdf:Description rdf:about="&Semanta;complex-relation">
      <Semanta:positive-associate >
         <rdf:Bag>
            <rdf:li rdf:resource="&Semanta;mother-of”/>
            <rdf:li rdf:resource="&Semanta;father-of”/>
            <rdf:li rdf:resource="&Semanta;brother-of”/>
            <rdf:li rdf:resource="&Semanta;spouse-of”/>
            <rdf:li rdf:resource="&Semanta;works-for”/>
            <rdf:li rdf:resource="&Semanta;funded-by”/>
         </rdf:Bag>
      </Semanta:positive-associate>
   </rdf:Description>
```

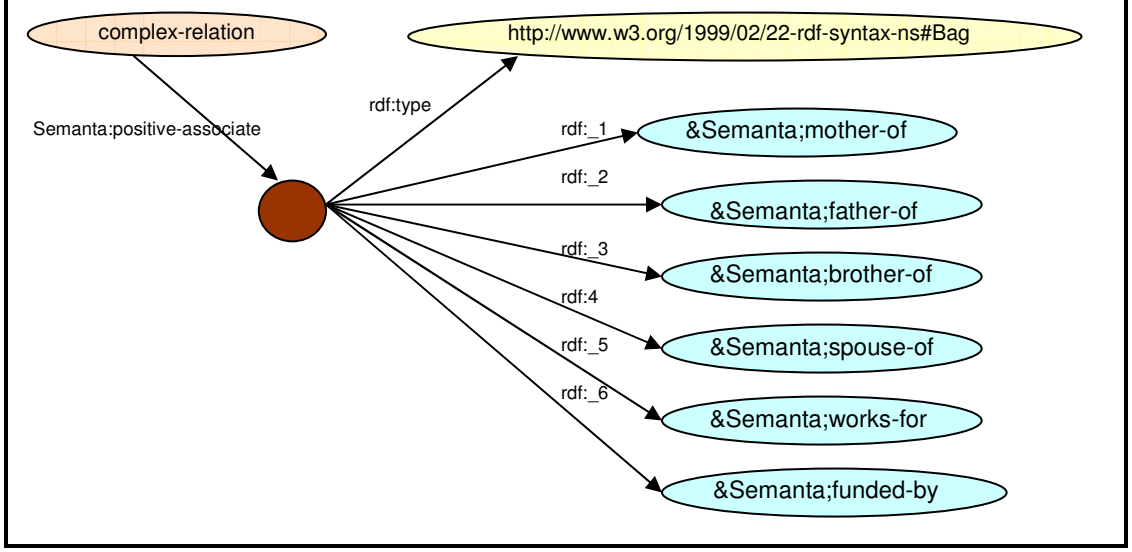


Figure 4.4 Definition of OR Complex Relation

4.2.2 AND Complex Relation

AND complex relation is a group of relations where all the member relations will have to be present in the resulting paths found. They can be used to summarize concepts represented using

RDF triplets. By requiring the presence of all member relations in the sub-graph, an AND relation is being enforced among the member relations. It is defined using the RDF collection and an example is shown in Figure 4.5.

This example illustrates a complex relation 'has-immigrated' that requires the resulting sub-graph to validate the presence of relations 'born-in', 'lives-in' and 'requires-work-permit'.

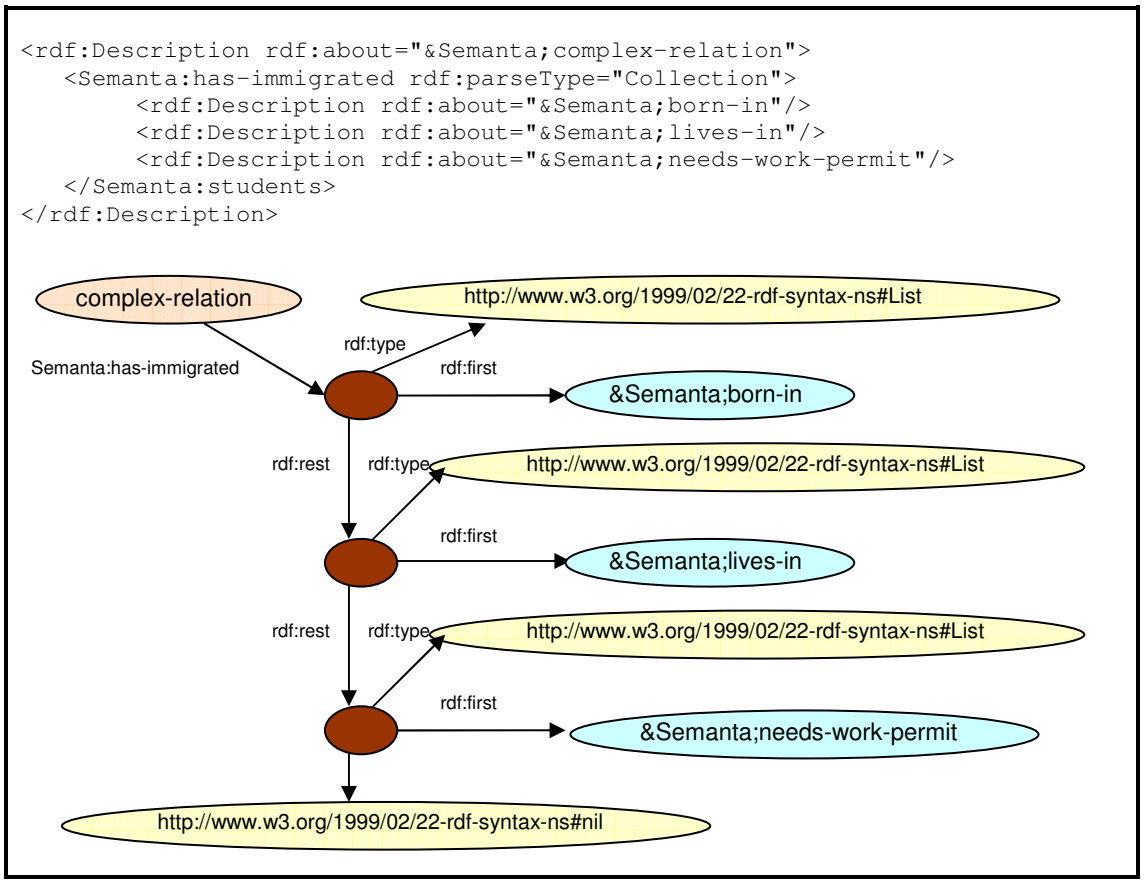


Figure 4.5 Definition of AND Complex Relation

4.2.3 Template Complex Relation

'Template complex' relations are used to capture snapshots in the Class Base and Object Base layers. A template is defined by a set of RDF triplets. A template consists of classes represented by nodes and properties represented by arcs. The properties in the template can also be complex relations. A property in a template can also define attributes such as multiplicity, transitivity,

equivalence and inverse. A property with multiplicity defined requires the presence of multiple objects in the Object Base layer, connected through the relation to the subject, i.e., a 'works-for' relation with multiplicity defined, requires that there be more than one 'employee' linked through 'works-for' to an organization in the Object Base layer.

Transitivity, equivalence and inverse can be used to broaden the scope of finding sub-graphs that match the template. Equivalence attribute indicates a list of relations that can be treated as the specified relation while finding the sub-graph. Inverse attribute defines the set of relations that are an inverse of the specified relation. An inverse relation implies a link between the subject and predicate in the opposite direction. For example, defining 'works-for' and 'pays-salary' to be in inverse, will indicate that the triplet "organization<pays-salary >person" is equivalent to the triplet "person<works-for >organization".

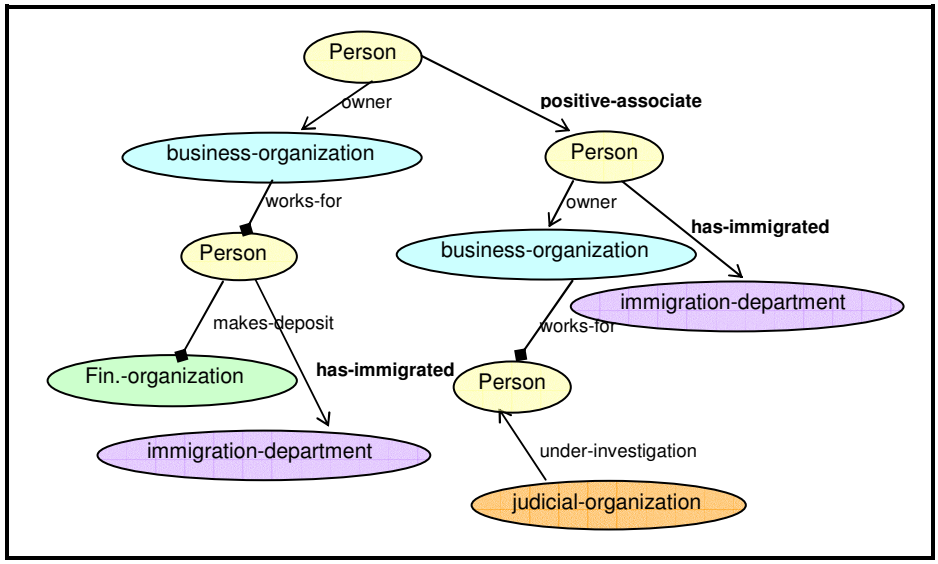


Figure 4.6 Template Capturing Money-laundering Scenario

A template capturing the money-laundering scenario is shown in Figure 4.6. Sub-graphs validating the template would contain instances of an immigrant making multiple deposits in a financial organization and working for a business organization that is owned by somebody well

known to the owner (who is an immigrant) of another business organization that employs people under investigation by a judicial organization such as the FBI. The links with block arrows such as ‘makes-deposit’ indicate a relation with multiplicity, i.e., a person makes multiple deposits with a financial organization. Links marked by relations in bold denote complex relations – ‘has-immigrated’ is a sequence relation and ‘positive-associate’ is a bag relation.

A template relation is introduced in this thesis. However, we leave its implementations as a future work.

CHAPTER 5

SYSTEM ARCHITECTURE AND IMPLEMENTATION

This chapter discusses the system architecture of Semanta, detailing the data and process modules of the system. Figure 5.1 depicts the architecture of Semanta. In the core of the system is the knowledge store of Semanta, also known as the semantic network. It consists of the 3 layers – Class Base layer, Object Base layer and Information Source layer as discussed in Chapter 3. The knowledge store is accessed through the Semanta API (built over the interfaces provided by Jena and JDOM).

Consider an example query looking for links between the 'Energy Sector' and the 'Party X'. These inputs will be obtained as entities through the User Interface Module. The 'Path Finder' module accesses the knowledge store using the Semanta API to check if the entities exist in the ontology layers. It also looks for any direct links that might exist between the entities. When no direct links exist, the 'Hints Generator' module generates hints associated with the input entities and passes them onto the ISS Collector.

The modules in the 'ISL (Information Source Layer) Path Finder' are 'Direct Path Finder' and 'Indirect Path Finder'. The ISL Path Finder generates XPath queries based on the hints from the ontology layers. The candidate documents are identified based on the queries and are then searched to find direct or indirect links. The links from ontology layers and from the Information Source layer are then gathered and can be ranked before sending them back to the User Interface module (future prototype item). The collected links might also be optionally used to update the knowledge store.

The modules in Figure 5.1 are discussed in more detail in the following sub-sections. The modules that are not currently implemented in Semanta – 'Information Source Extractor' and 'Gather and Rank' are shown using dotted lines in Figure 5.1.

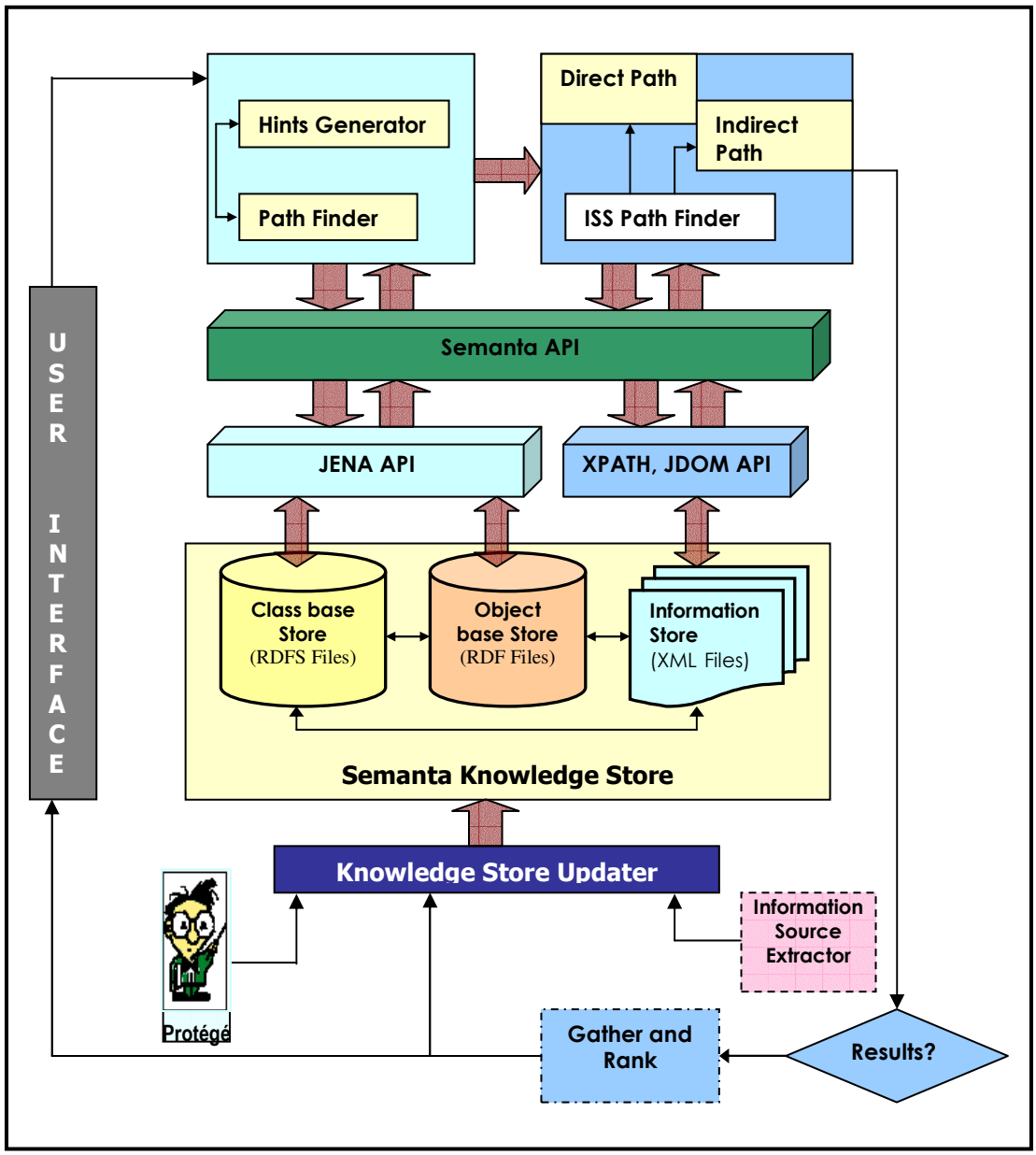


Figure 5.1 System Architecture of Semanta

5.1 Knowledge Store Updater

Knowledge Store Updater acts as an interface for updating the knowledge repository of the system. The ontology is initially created using Protégé [PRO], an ontology and knowledge base editor. Currently, Semanta does not support automatic mapping of web pages to XML documents. However, extractors and wrappers [LRST02] tools can be used periodically to map web pages from trusted resources to XML documents stored in the Information Source Layer. The results of queries can also be treated as knowledge and updated back to the system, under supervision, to further enrich the knowledge store.

5.2 User Interface

The User Interface module is responsible for getting inputs from the user and presenting the results to the user. This section details the design principles behind this module. A query language is defined and the presentation of the outputs is discussed.

5.2.1 Semantic Link Query Language

An ontology O, is defined as a set of concepts with attributes and their relationships and formally specified by

$$O = \{\{<O_1, A_1>, <O_2, A_2>\ldots<O_m, A_m>\},\{<R_1, O_i, O_j>\ldots<R_n, O_k, O_l>\}\}$$

Where,

$\mathbf{O_i}$ - a concept/class

$\mathbf{A_i}$ - a set of attributes for a given class

$\mathbf{R_i}$ - a relationship between two classes

We also use $\mathbf{o_l}$ and $\mathbf{a_l}$ to denote instance of a class and instance of an attribute respectively.

A query in Semanta can be constructed using the following three sections:

Query:

$$< [O_i| o_i| O_i.A_i| o_i.a_i| x] [R_i| x] [O_i| o_i| O_i.A_i| o_i.a_i| x]>$$

Where, the first part **[O_i| o_i| $O_i.A_i$| $o_i.a_i$| *x*],** is referred in this document as 'left entity' or 'e_1', the middle part refers to a relation and the third part is referred to as 'right entity' or 'e_2'. And '***x***' is be used to denote the part that needs to be found. A left or right entity could denote a class, attribute instance or literal. A query of the form 'e_1 **x** e_2' indicates the user is interested in finding all possible paths between the entities e_1 and e_2. A query of form 'e_1 R_i x' indicates the user is interested in finding all possible entities that are related to e_1 through the R_i relationship. Table 5.1 lists some sample queries.

Table 5.1 Example Queries

Type	Example
O_i **x** O_k	*1) 'University' x 'Music Groups'* *2) 'Mountain' x 'Casualties'*
O_i **x** o_k o_i **x** O_k	*1) 'University' x 'R.E.M.'* *2) 'Nyiragongo'(Volcano) x 'Casualty'*
o_i **x** o_k	*1) 'UGA' x 'R.E.M.'* *2) 'Smith' x 'Enron'*
$O_i.A_i$	*'Company.Ceo'*
$o_i.a_i$	*'AlQeida.Afghanistan'*
O_i R_i **x**	*'Person' 'positive-associate' x*
o_i R_i **x**	*'Halliburton Company' 'employs' x*

Ontology constraints are used to restrict the discovery of links within certain ontologies and/or information sources. This mechanism aids the user in implicitly stating his domain of interest, by stating the ontologies and information sources to be used in the process. Explicit omission of ontologies/information sources is also possible. The syntax for stating the domain of interest is:

<[ONT-URI|alias]>, <[ONT-URI|alias]>...┐<[IS-URI|alias]>..

The '¬' symbol denotes the omission of an ontology or information source. For instance, a user might be interested in finding relations between two persons excluding the academia domain, whereby links found that belong to the academia domain will not be processed.

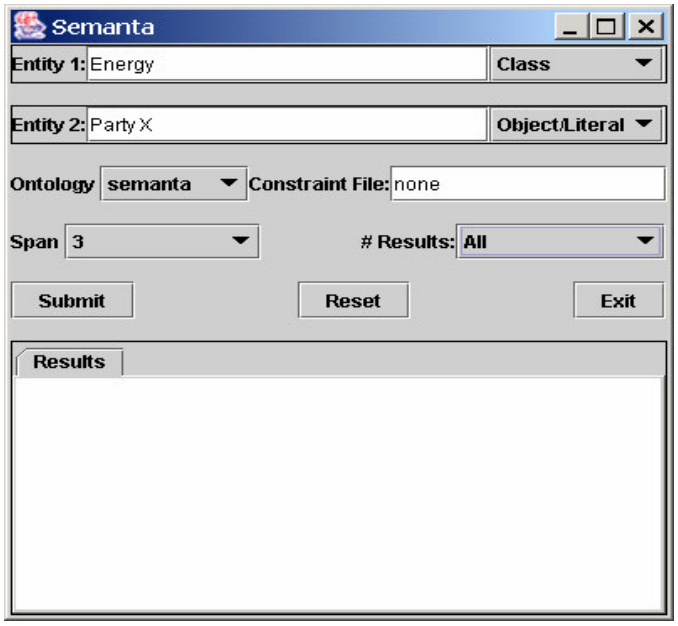


Figure 5.2 Snapshot of Input Screen

Semantic constraints are optional suggestions that can be used, while finding links. They can be used to place restrictions on the type of links that the user might be interested in as results. Also, it can be used as a controller for stating how close a link the user wants, by stating attributes of relations such as transitivity, inverse-of, etc., that can be applied while searching for links. The user can also introduce new notions of links using relation ontology, while finding related entities. A relation-ontology defines complex relationships (as discussed in Chapter 4.2) in terms of predefined relationships present in the knowledge store. This feature is currently not supported in Semanta.

In a query (snapshot in Figure 5.2), **Span** indicates the number of hops or links to traverse, in order to look for a given node, while finding links. Span can be relaxed in successive

queries to increase the probability of finding links. The user can also choose the **number of results** option to select between *single* result (which would obtain the first available link) and *all* possible results (to view all the links found).

5.2.2 Ranking the Semantic Links

The links found based on the user inputs, are presented as paths between the input nodes (entities and/or relations). The paths can be broadly classified into direct paths and indirect paths. Direct paths are instances where a direct path already exists in the knowledge store. Indirect paths are those for which a path has been found by extrapolating the information present in the knowledge store.

The presentation of the results to the user should satisfy the following design criteria:

(i) User should be able to comprehend the results easily thereby aiding him/her in the end decision making

(ii) More detail of each result should be available, where needed

(iii) Information regarding the source from which results have been inferred should be available

(iv) Summarizing results, based on parameters such as relationships, path-lengths, etc., should be present on request

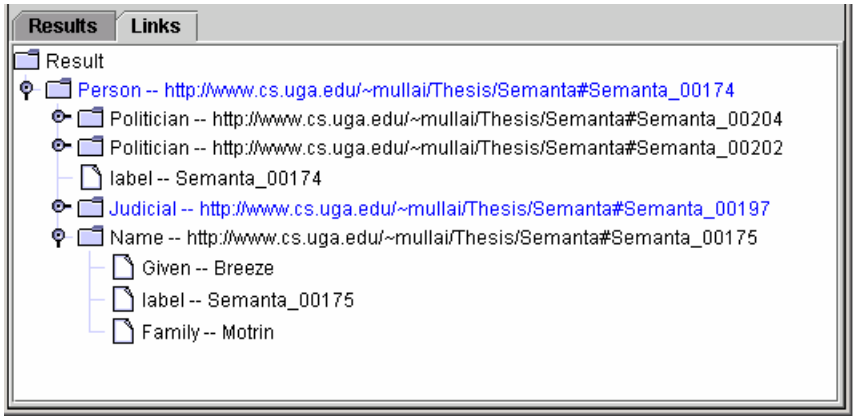


Figure 5.3 Snapshot of Results Screen

Taking the above constraints into consideration, the results have been presented in a tree view implemented using JTree, as shown in Figure 5.3. The figure shows a validation for a path between entity 'Person' and entity 'Judicial'. Input entities are identified in a different color from the rest of the nodes and are also shown within boxes. In the presence of multiple results, the user can click on specific results and expand the tree for each result and get more detailed information. The source for each segment of the path will be provided, where needed, in order to help the user in making decisions about the authenticity of the results.

5.3 Semanta API

The Semanta API module is used to access the knowledge store of Semanta. It uses Jena to access the ontology layers and employs JDOM and XPath to access the Information Source layer. The Semanta API consists of CB Layer API, OB Layer API and IS Layer API modules. Some of the methods of these modules are listed in Table 5.2. This section details the interaction of Semanta API modules with the underlying technologies.

Jena [JEN] is a Java API for manipulating RDF models. It provides statement centric and resource centric methods. Statement centric methods are used to manipulate the RDF model as a set of triplets and resource centric methods are used to manipulate them as a set of resources. Semanta builds over Jena 1.5.0 API to access the ontology layers.

5.3.1 CB Layer API

The CB (Class Base) Layer API provides information of the ontology, by accessing RDF Schema files. This is done using Jena 1.5.0. However, Jena does not have any in-built support to access RDF Schema files. It treats Schema files also as regular RDF files. In order to access the schema information, the Class Base Layer API has been implemented over the Jena API by accessing RDF Schema files as RDF files. The methods provided by the Class Base Layer API

(summarized in Table 5.2), are used to ascertain the presence of a class or property in the schema and also to obtain super and sub-classes/properties of a given class/property. The level parameter denotes the number of hops (span), between the given and the desired nodes. For instance, a level of 2 in *getChildren* for a class 'Industry' will return all their children (such as Telecom, Energy, etc.) and their children too (such as Wireless Communication Services, Petroleum Products Distribution, etc.)

5.3.2 OB Layer API

The OB (Object Base) Layer API provides an abstraction layer to facilitate access to the RDF files in the framework. Since the Object Base layer consists of instances validating (or referring to) the Class Base layer, there exists a fair amount of inter-links between these two layers. Therefore, methods are provided to find cross-references. The methods in the OB Layer API are shown in Table 5.2. The routines involve finding whether a given literal is present in this layer, identifying the class to which it belongs, collecting the attribute values of a given class and finding instances belonging to the same class.

Table 5.2 Semanta API

CB Layer API – Class Based	**CB Layer API – Property Based**

`isClassPresent(class)`
`getAttributes(class)`
`getChildren(class,level)`
`getAncestors(class,level)`
`getLinkedClasses(class,level)`
`getCoClasses(class)`
	`isPropertyPresent(property)`
	`getDomainClasses(property)`
	`getRangeClasses(property)`
	`getSubProperties(property)`
	`getSuperProperties(property)`

OB Layer API	**IS Layer API**

`isLiteralPresent(lit)`
`getClassName(lit)`
`getClassID(lit)`
`listInstances(class)`
`getAttrValues(class)`
`getContainer(class)`
`getPropName(propVal)`
	`isTagPresent(collection)`
	`isLiteralPresent(collection)`
	`isPatternPresent(collection)`
	`getCommonNodes(doc, doc)`
	`matchDocuments(doc, doc)`
	`queryAndCollect(collection, query)`

5.3.3 IS Layer API

The structured data present in the IS (Information Source) Layer consists of XML documents that contribute toward making the knowledge store of Semanta more complete and up-to-date. The XML documents known to Semanta are stored via Apache XIndice [XIN], which is a native XML database provided, by the Apache Software Foundation. A recent survey of native XML databases can be found in [HM03].

Semanta builds on top of the XPath [XPT] API (which is implemented by XIndice), for accessing parts of the documents. For generating queries across multiple documents and finding indirect paths between the documents, JDOM [JDO] is used. JDOM is a tree-based API for accessing and manipulating XML documents.

The methods provided by the IS Layer API are shown in Table 5.2. They can be used to check for the presence of a tag or literal, performing XPath queries on a collection of documents, checking for the presence of a pattern in documents, and finding matching patterns between documents.

5.4 Searching the Ontology Layers

The inputs obtained from the User Interface module are first processed with respect to the Ontology layers (Class Base Layer and Object Base Layer), by the 'Path Finder' and 'Hints Generator' modules. The goal of the Path Finder module is to find paths between the input entities using the CB Layer API and OB Layer API. Hints Generator module enriches the inputs with information that can be obtained from the Ontology layers and passes them onto the Information Source Layer processing. These modules are discussed in detail in the following sub-sections.

5.4.1 Path Finder

In order to find semantic links between the input entities, within the Class Base and Object Base layers, this module tries to ascertain if any semantic relationship exists between them. A relationship at this level is broadly classified as class/property relation, property/property relation and class/class relation.

A class-property relation encompasses the following cases:

i) The property is an attribute of the class,

ii) The property links the class to another class,

iii) The property is a transitive link to another class that is present n hops away from the given class.

The class-property relation section of Figure 5.4 illustrates these kinds of relationships. Dotted ellipse denotes the class node and dotted line denotes the property, which manifests this relationship.

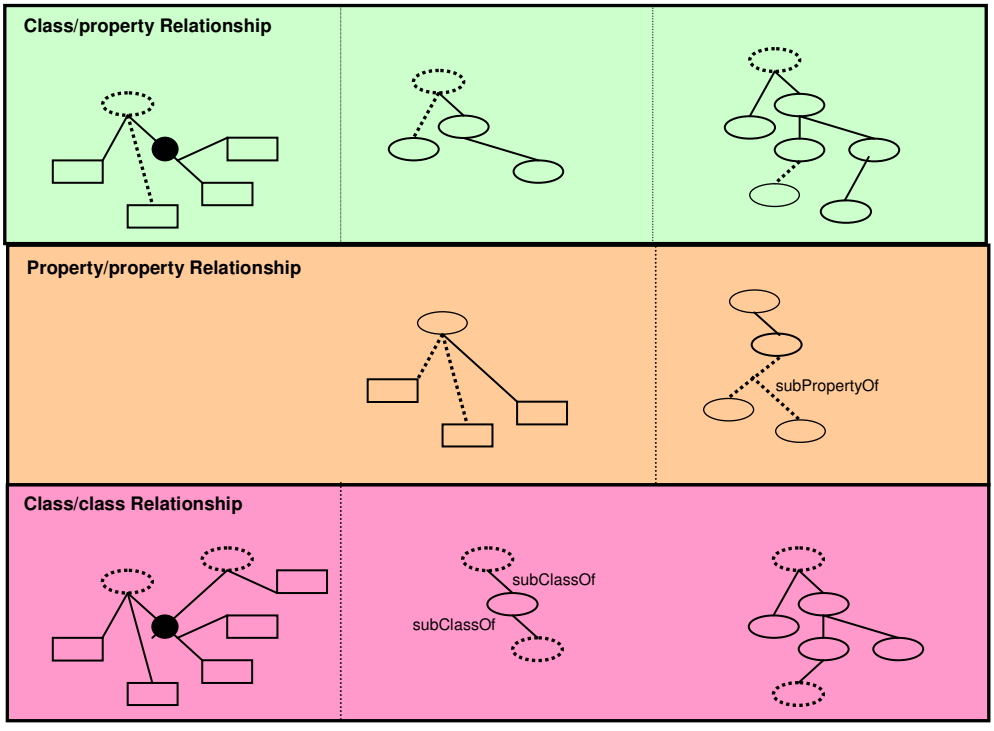


Figure 5.4 Relationships in the Ontology Layers

A property-property relation encompasses the following cases:

i) Both properties belong either to the same class or the same instance of the class

ii) There exists an hierarchical (sub Property or super Property) relationship between the properties

The dotted lines in the 'Property-property Relationship' section of Figure 5.4 denote the property links that share this relation.

A class-class relation encompasses the following cases:

i) The instances of the classes have property values that match

ii) There exists an hierarchical (subclass or super Class) relationship between the classes

iii) The classes are linked either directly or indirectly (n hops) through property links

The dotted ellipses in the 'Class-class Relationship' section of Figure 5.4 denote class nodes that share a class-class relation.

These relationships are discovered by using the methods provided by the CB Layer API and OB Layer API modules. The process of finding a path between entities (e_1 and e_2), in the Ontology layers starts with detecting the node associated with e_1 entity. The nodes related to this node, that are one hop away are gathered as a set without any duplicates. While gathering the nodes the relations linking the nodes are also processed to check if the other entity e_2 exists. The nodes related to the gathered set are gathered repeatedly until the desired *span* value is reached. The gathered nodes are then searched for the other input entity e_2 in order to establish a path. The time to taken to look for paths (T_{path}) in the Ontology layers can be as:

$$T_{path} = T_{gather} + T_{process}$$

Where, T_{gather}, refers to the time taken to gather the nodes and $T_{process}$ is the time taken to process the gathered nodes. Assuming that a given node is connected to *n* other nodes, including class nodes in Class Base Layer and instance nodes in Object Base Layer,

$$T_{path} = [n^1 + n^2 + \ldots + n^{span-1} + n^{span}\ ] + [n^1 + n^2 + \ldots + n^{span-1} + n^{span}\ ] = O(n^{span})$$

The process of looking for paths within the Ontology layers is akin to breadth-first search where the search starts from e1 and from there on each neighbor is visited and checked for the presence of e2. The complexity of this algorithm as discussed above indicates that it will not scale very well as the average number of nodes connected to each node increases. The span parameter can be adjusted to indicate the user's willingness to compromise the speed of the search for getting useful results. Apart from using span, we present two alterations of this search technique – *Directed BFS (Breadth-First Search)* and *Interactive Deepening* to address scalability issues. These techniques have not been implemented in Semanta yet.

We propose a directed BFS strategy in which, the number of neighbors that are going to be accessed by a given node is reduced by using heuristics. Directed BFS has been explored for locating data efficiently in peer-to-peer networks [YG02]. The heuristic used in looking for paths in the Ontology layers is based on the domain to which the neighboring nodes belong. Users can define an ordered set of domains/regions of interest (disinterest) to them. By specifying the domains the user indicates his/her preference of the nodes through which paths have to pass.

A domain/region translates to a set of Class Base nodes and links in Semanta's lingua. Domains of interest take values ranging between 0-1.0, where a lower value indicates lesser preference. Domains that the user is explicitly not interested in get negative values. Neutral domains will have values assigned. All the nodes in a given domain share the same value assigned to the domain to indicate the fact that they will be treated as the same.

The directed BFS heuristic discussed is defined by the policy, P= [*window_size*, *span*]. The algorithm looks for the neighbors of nodes in a particular level. There exist three cases of interest at this point:

i) All neighboring nodes belong to the same domain

ii) None of the neighboring nodes belong to any specified domain

iii) Neighboring nodes belong to multiple domains

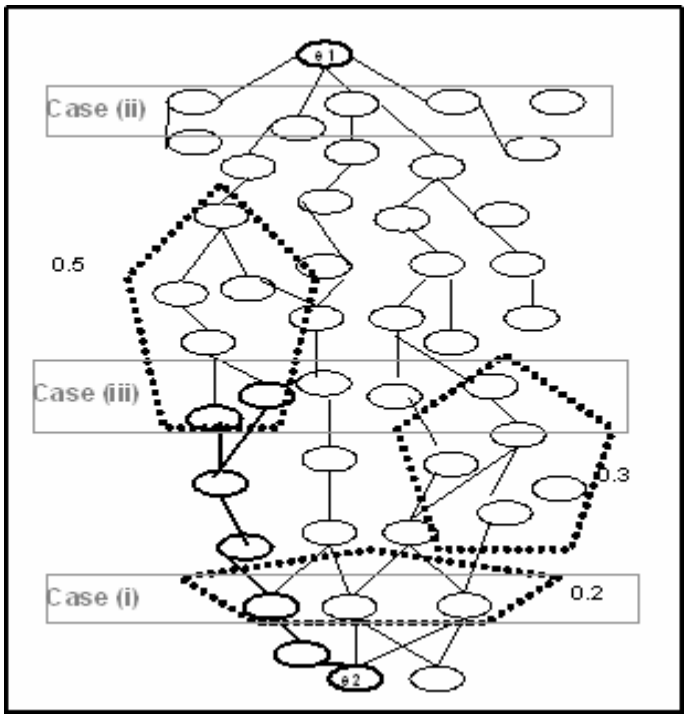


Figure 5.5 Directed Breadth-First Search

A portion of the ontology on which directed breadth-first search is performed is depicted in Figure 5.5. The nodes between which we are trying to find paths are identified as **e1** and **e2**. The dotted pentagon boxes refer to nodes that belong to domains of interest provided by the user. The numbers along the pentagon boxes indicate the values associated with the domains. The rectangular boxes refer to nodes that are being processed at a specific level and identify the three cases listed above.

In the first case, where all neighboring nodes belong to the same domain, all the nodes will be processed because within a domain all nodes are treated equally. When none of the nodes belong to any domain (case ii) all the nodes will be processed, because they are neutral nodes and no information is known about them. In the case were neighboring nodes belong to multiple domains (case iii), the nodes take the value of the domain to which they belong and *window_size* number of nodes with highest values will be processed. These steps will be repeated for every level until *span* is reached. The algorithm discussed here is outlined in Table 5.3.

Consider a policy, P= [2, 20]. In Figure 5.5, consider the instance where processing is at the level indicated by case (iii). At this level 5 nodes exist with varying values associated with them. Since the window size is 2, only of two of the nodes within the domain with value 0.5 are selected for further processing. The path from these selected nodes after further processing is indicated by dark edges in the diagram.

Table 5.3 Algorithm for Directed BFS

```
1. Identify e1, e2, LSet, RSet. l = 1
2. Until l == span
  {
     2.1.  For each node n1, in Lset
          {
           For each neighbor of  n1
          {
               /*Assign values for the nodes*/
              If nodes belong to given domains
                   assign the domains value
             Else
                assign 0
          }
        }
   2.2.  /* Gather nodes for processing */
   If no nodes belong to selected domains
       Gather all nodes as NSet
  If all nodes belong to the same domain
      Gather all nodes as NSet
  Else
      Gather the 'window_size' nodes with the highest values as NSet

  2.3. Process NSet
      If e2 belongs to NSet, indicate presence of a path
  2.4.  l++
  2.5.  LSet = NSet
}
```

The ideal case for this algorithm occurs when the user specifies many domains, so that the heuristic can be used in effectively in favoring a selected few neighbors. The algorithm deteriorates to breadth-first search algorithm's performance when *window_size* is large or when no or few domains are specified.

The *interactive deepening* is a variation of breadth-first search, which gets input from the user at specific intervals to curtail the number of nodes visited. The user visually selects the nodes of interest that will be pursued in the future thereby limiting the number of nodes and assisting in the process of path discovery.

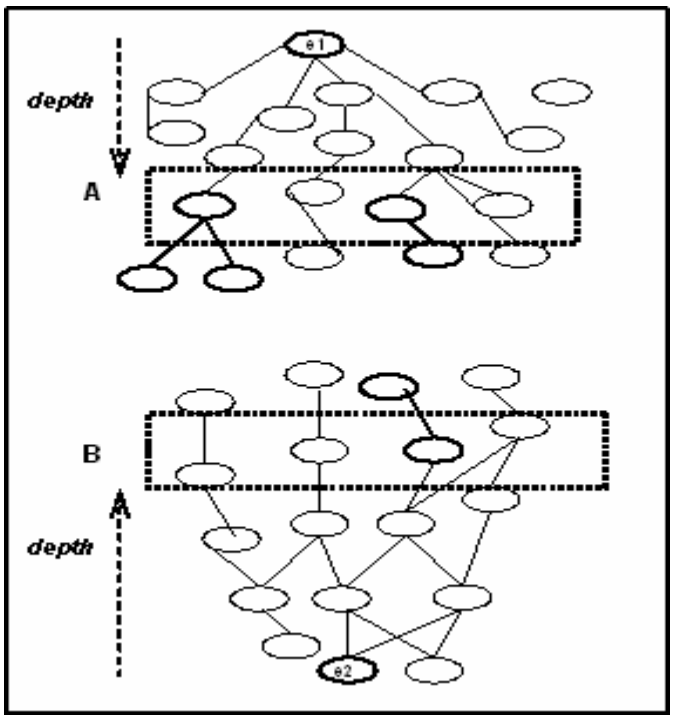


Figure 5.6 Interactive Deepening

The policy is specified by, P= [*depth, span*]. The process of looking for paths follows the regular breath-first search until nodes at level=*depth,* are reached. Once this level is reached, the existing paths are presented to the user. The user selects the nodes at this level which s/he wishes to pursue. Thereafter, only the selected nodes are considered for finding subsequent paths. This process is repeated each time current level mod *depth*= 0 until *span* is reached.

In order to better assist the user in choosing the nodes, the following alternation is used: At first, path is traversed starting from e1 until *depth* is reached, at which point the user makes selections. Then the path is traversed starting from e2 and nodes are presented to user for

selecting on reaching *depth*. Thus the paths are traversed both from e1 and e2, by alternating between them. This gives the user a more wholesome picture of the existing paths and better assists the user in selecting useful nodes as compared to pursuing the path from one side alone.

The algorithm is depicted in Figure 5.6 where e1 and e2 are the nodes between which we are interested in finding paths. The dotted rectangle refers to the stage when the user will be provided with the current sub-graph to select the nodes within the rectangle for future processing. Initially the rectangle is reached starting from e1 (A), and once the user selects the nodes, the process starts from e2's end (B).

Table 5.4 Algorithm for Interactive Deepening

```
1. Identify e1, e2, LSet, RSet. l = 1. direction=left
2. Until l == span
  {
      Until l mod depth== 0
       {
         If (direction == left)
             CSet = LSet
         Else
              CSet = RSet
         For each node n1, in CSet
            For each neighbor node of n1
            {
                Gather all directly connected nodes in NSet
                Process NSet
                l++
            }
       }
       if (direction == left)
       {
          Present all paths from e1 to NSet to user
          Gather nodes of interest in level l as LSet
          direction = right
          If e2 belongs to LSet indicate presence of a path
       }
       else
       {
          Present all paths from e2 to NSet to user
          Gather nodes of interest in level l as RSet
          direction = left
          If e1 belongs to RSet indicate presence of a path
       }
       l++
  }
```

The algorithm discussed is detailed in Table 5.4. The interactive deepening algorithm will behave like a breadth-first search algorithm when *l1=l2* or when the user is interested in all the nodes identified at multiples of *l2* levels.

Consider a policy P= [3, 25] for this approach. Then on reaching the window at '**A'** in Figure 5.6, where the current level is 3, the user is presented the nodes to select. The selected nodes are shown with dark edges and only paths from these nodes will be processed in the future. Once the user selects these nodes at A, processing continues from the other end.

5.4.2 Hints Generator

The 'Hints Generator' module takes inputs provided from the user and further enriches it by parsing the Ontology layers. As a result of this, it generates hints for the next level of processing at the Information Source layer. A hint is formally defined as a collection of class nodes, instance nodes and properties in the vicinity of an entity.

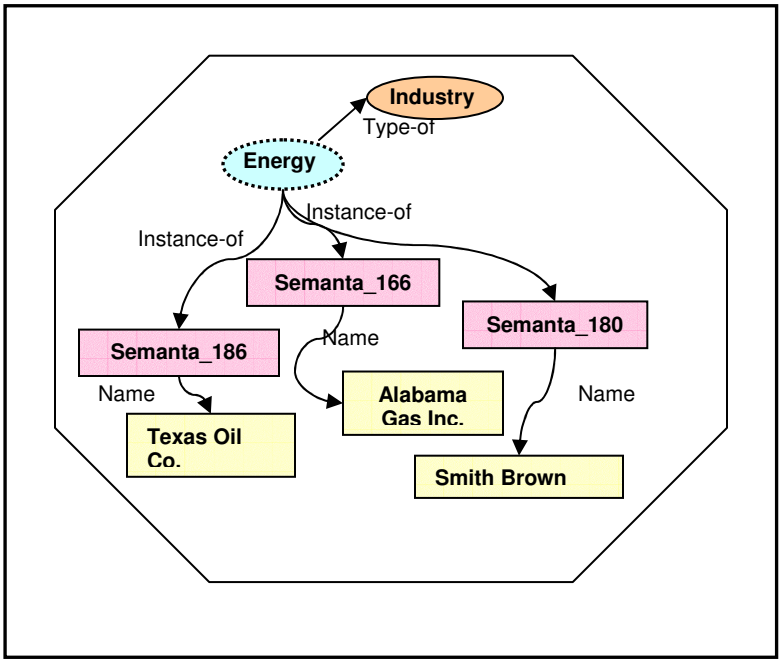


Figure 5.7 Hints for 'Energy Sector'

For example the hints gathered for ‘Energy Sector’ input is shown in Figure 5.7. It can be seen as a snapshot of the nodes surrounding the ‘Energy’ node. The class nodes are denoted by ellipse, properties by arcs and instance nodes by rectangles.

The hints are gathered for entities by parsing RDFS and RDF files using the Semanta API. The hints thus gathered are processed by the path finder modules of the Information Source Layer. Hints are further discussed with respect to finding paths in the Information Source layer in Chapter 6.2

5.5 Searching the Information Source Layer

The ‘Direct Path Finder’ and ‘Indirect Path Finder’ are the modules used in finding semantic links in the Information Source Layer. Finding paths in the Information Source Layer is initiated under the following circumstances: (i) The Ontology layers do not contain the input entities and (ii) The Ontology layers do not contain links between the entities, in which case they pass on the hints. The hints are used to generate XPath queries, which are used in selecting the documents for further processing to ensure the presence of links.

A direct path is one that is based on the ‘parent-child’ or ‘sibling’ relationship in the XML document. An indirect path is either based on links amongst the hints or is based on finding matching patterns between documents. A pattern is a string that captures the structure of the elements in the document. Here, patterns can either pertain to tag elements or to text elements or both. The hints from the ontology layer will be translated to patterns to query the Information Source layer. The process of finding paths in the Information Source Layer is discussed with examples in Chapter 6.3.

CHAPTER 6

TRAVERSING THE SEMANTIC LINKS

In this section we discuss the process of finding semantic links between entities in detail with examples. Figure 6.1 illustrates the process of finding links between entities 'A' and 'F'. Each relation shown between entities within the octagon boxes denotes relations that are either directly present in the knowledge store or are inferred. Relations can be inferred in the Ontology layers based on the categories discussed in Chapter 5.4.1 or can be detected as paths in the Information Source Layer based on the presence of parent-child or sibling relationship in the XML Documents. The relations are shown in different octagons to indicate that they might be deduced/present within different layers or information sources. Based on the known/inferred relations, the links can be established between entity 'A' and entity 'B' as shown within the rectangle box.

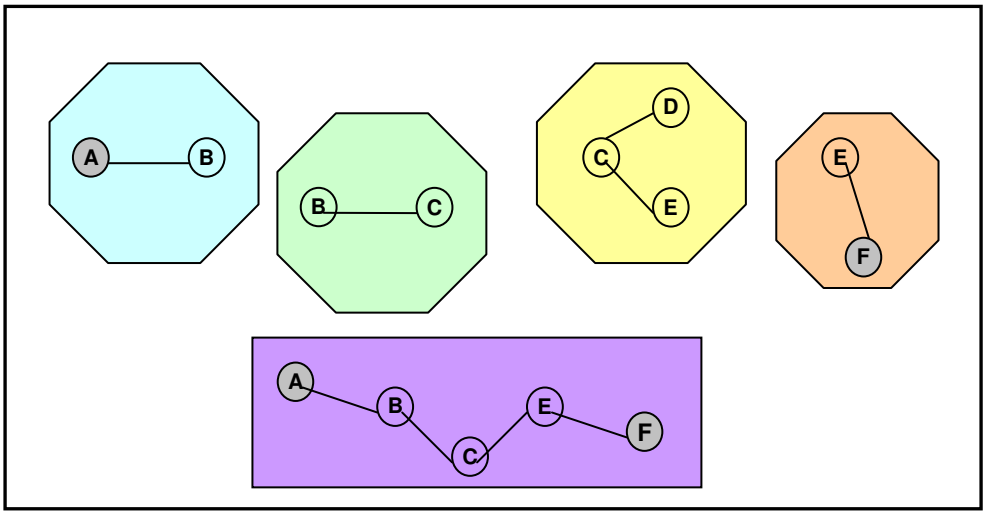


Figure 6.1 Connecting the Semantic Links

The process of detecting semantic links between the given entities can thus be seen as identifying and detecting links within the layers of the knowledge store and using them to find

links across the 3 layers and eventually processing all the relations to obtain a path that connects nodes across the layers.

The algorithm for finding links in the knowledge store of Semanta, is given in Table 6.1 and illustrated in Figure 6.2.

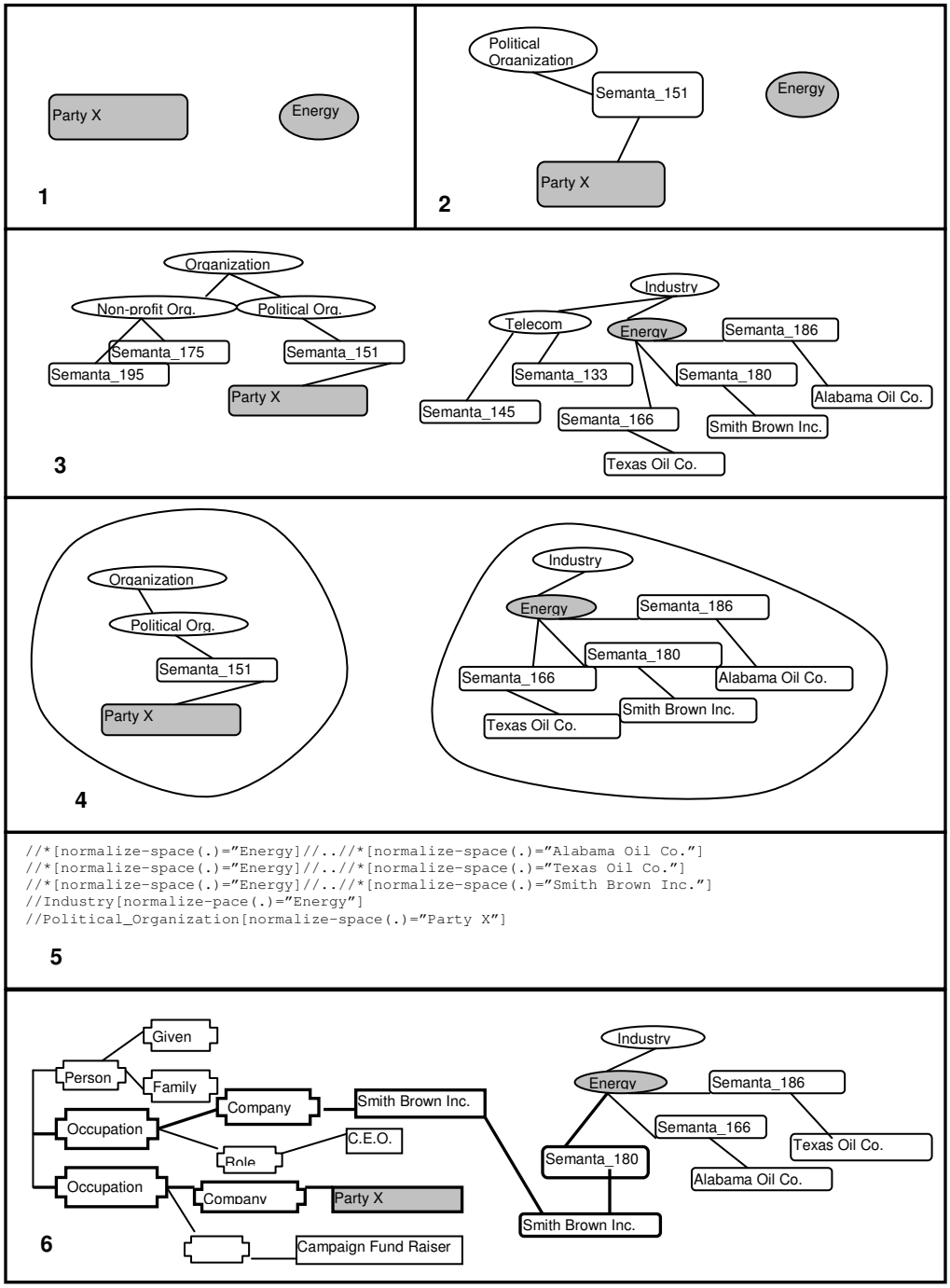


Figure 6.2 Process of Finding Semantic Links

The first step involves identifying the inputs and their related nodes, which is illustrated in Figure 6.2.1 and 6.2.2 and is detailed in Chapter 6.1. Once the nodes are identified, Semanta checks for paths that might exist within the Ontology layers based on the categories of relations discussed in Chapter 4.1 – this is illustrated in figure 6.2.3 and discussed in Chapter 6.2. If no paths exist within the Ontology layers, hints are generated which is shown in Figure 6.2.4. They are also discussed further with an example in Chapter 6.2. The hints are used to generate XPath queries (Figure 6.2.5) also elaborated in Chapter 6.3. The results of queries are used in finding links within the Information Source Layer. Finding links within the Information Source layer is detailed in the algorithm presented in Table 6.3. Finally the results obtained from the Information Source layer and the hints from the Ontology layers are used in presenting the semantic links spanning the 3 layers of the knowledge store as shown in Figure 6.2.6.

6.1 Identify Candidates for Left and Right Nodes

At first, based on the input from the user, such as entity e_1='Energy Sector' and entity e_2='Party X', nodes of interest are identified as left and right. In general, e_1/ e_2 can be a property, a class or a literal in the knowledge store as mentioned earlier. The inputs are validated and their respective categories are found, based on the categories discussed in Chapter 4: Type 1 (e_1 is either a class or property, e_2 is either a class or property), Type 2 (e_1 is either a class or property, e_2 is a literal) and Type 3 (e_1 is a literal, e_2 is a literal).

Once the inputs are categorized, then the Left-Right (LR) node-sets are initialized. The LR nodes are the nodes associated with e_1 and e_2 respectively. There can be multiple nodes for left and right, i.e., it is possible that there can be more than one node matching the input across the Class Base and Object Base layers. For property/literal inputs, the nodes containing the attribute/value have to be identified first. Then, the corresponding class nodes in the Class layer are identified. For instance, if e_1 is a literal 'University of Georgia', then all class instances that

have e_1 as their attribute will be gathered as L set. After identifying the set of LR nodes, the objective of finding path between e_1 and e_2 reduces to finding path(s) between the LR nodes.

Table 6.1 Algorithm for Finding the Semantic Links

Input:

{ e_1, e_2} , constraints
e – Entity (class/property /literal)
e.g. – {'Energy', 'Party X'}

Output:

Set of ordered semantic links (paths) that span across the 3 layers of Semanta

Algorithm:

1) ***Validate inputs***
   a. *Check if e_1, e_2 exist as class/property/literal in the ontology*
   b. *Identify the category to which e_1, e_2 belong*

2) ***Check if a path exists between e_1, e_2 in the CB-OB Layers based on the following categories***
   a. *Class-Attribute*
   b. *Children*
   c. *Ancestors*
   d. *Linked classes*
   e. *Co-classes*

3) ***Gather Hints from the CB-OB layers***
   a. *For e_1*
      Collect classes, properties, instances that are immediately known to e_1 (limit using span)
   b. *For e_2*
      Collect classes, properties, instances that are immediately known to e_2 (limit using span)

4) ***Check if paths exist in the IS Layer (see Algorithm 2)***
   a. *Direct paths*
   b. *Indirect paths*

5) ***Collect paths in CB-OB layers and IS layer and present it to the user***

6.2 Paths in Ontology Layers

After identifying the LR nodes, Semanta tries to find paths, across the Class Base and Object Base layers, connecting these nodes. If no such paths exist, hints are gathered from Class Base and Object Base layers and are used to find paths in the Information Source layer. The process of finding links, involves working at each layer independently, as well as using the information

gathered at each layer to further enrich the information across the layers. It involves going between layers when the information currently obtained is insufficient to find the path.

A path in Class Base and Object Base layers is found based on the class-attribute relationship, parent-child relationship, linked-classes relationship, or co-classes relationship discussed earlier. After each relationship is found, it is validated by instances in the Object Base layer. When existing information is not sufficient to find complete paths that link the entities, hints are gathered from the ontology layers and, the Information Source layer is further perused to find complete paths.

Consider as an example, a query to find relationships between the 'Energy Sector' and the 'Party X'. At the outset, Semanta finds the nodes associated with these inputs. The Class Base layer contains the information that 'Energy' is a class node, which is of type 'Industry', and the Object Base layer contains three instances belonging to 'Energy' namely 'Texas Oil Co.', 'Alabama Oil Co.', and 'Smith Brown Inc.'. As for the 'Party X', Semanta gathers the information, that it is the value of the name-property that belongs to an instance of 'Political Organization'. Based on the Class Base and Object Base layers, there do not exist any of the types of aforesaid links between the entities.

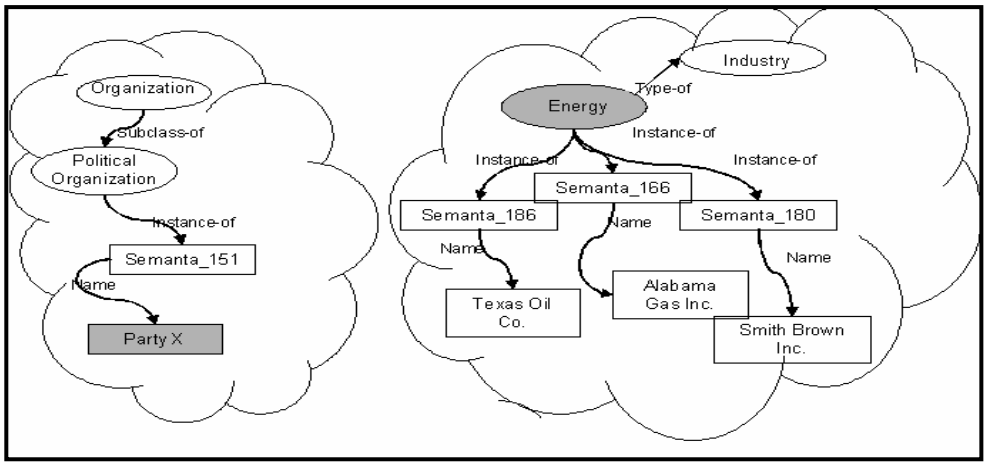


Figure 6.3 Hints from the Ontology Layers

The information gathered by Semanta, in the Ontology layers, is encapsulated as hints and is passed onto the Information Source layer. The hints are depicted in Figure 6.3. The hints-set associated with each input is gathered under class, property and instance categories. The hints associated with 'Energy Sector' can be seen from the Figure 6.3: Class-Hints: 'Industry', Property-Hints: 'Name', Instance-Hints: 'Semanta_186', 'Semanta_180', 'Semanta_166'. The hints associated with 'Party X' are: Class-Hints: 'Political Organization', Property-Hints: 'Name', Instance-Hints: 'Semanta-151'.

6.3 Paths in Information Source Layer

If the information found in the first two layers is not sufficient to find the path, i.e., when there exists no path in the Class Base and Object Base layers or when the inputs do not belong to the Class Base and Object Base layers, then the Information Source layer is accessed. The tag names in this layer, can be mapped to the class names and attribute names in the Class Base layer, and the text elements in the XML documents can be mapped to the Object Base layer values, if present, thereby providing implicit links across the layers. Also, the hints found so far in the ontology layers can be used to get more information in the Information Source layer.

The algorithm for finding links in the Information Source layer is presented in Table 6.3. Accessing the information at the Information Source layer, involves composing queries automatically based on the half-paths. The path expressions and quantified expressions of XPath are used to generate queries for the Information Source layer. The results of these queries are used to locate the documents of interest in the Information Source layer, and these documents are further processed, to find paths between the entities. Consider the hint-sets presented in Figure 6.3, generated in order to find links between 'The Party X' and 'the energy department'. Based on the hint-sets, the XPath queries, generated are shown in Table 6.2. The documents that contain segments for the queries are then collected.

Table 6.2 XPath Queries based on the Hints

```
//*[normalize-space(.)="Energy]//..//*[normalize-space(.)="Texas Oil Co."]
//*[normalize-space(.)="Energy]//..//*[normalize-space(.)="Alabama Oil Co"]
//*[normalize-space(.)="Energy]//..//*[normalize-space(.)="Smith Brown
Inc."]
//Industry[normalize-pace(.)="Energy"]
//Political_Organization[normalize-space(.)="Party X"]
```

Once the documents are collected, we need to find some kind of connection between elements in the hints set. The paths found, can be either direct or indirect paths. A path based on sibling or parent-child relationship between 'Energy' and 'Party X', would classify as a direct relationship. For instance, the presence of the following XML segment qualifies as a direct path between 'Energy' and 'Party X', as they share a sibling relationship because of a common parent (<occupation>).

```
<occupation>
              <industry type="Energy">
                     <company> Texas Oil Co. </company>
                     <role> CEO </role>
                     <start_date> 0/0/1995 </start_date>
                     <end_date> </end_date>
              </industry>

              <industry type="Politics" >
                     <company> Party X </company>
                     <role> Secretary of Defense </role>
                     <start_date> 0/0/1989 </start_date>
                     <end_date> 0/0/1993 </end_date>
              </industry>
</occupation>
```

Indirect paths are obtained by further processing the hints sets. They are the paths that might exists between the elements of the hint-sets, or, might be found by relaxing the notion of link in the Information Source layer. As an example of the first category, trying to find a path between an instance of Energy ('Alabama Oil Co') and the given entity ('Party X') might yield an indirect path in the presence of the following XML segment.

```
<election year="2000" >
       <party="Party X">
              <contributor>
                     <name> Alabama Oil Co.</name>
                     <amount> 113,800</amount>
              </contributor>
```

```
        </party>
</election>
```

Table 6.3 Algorithm for Finding Path in the Information Source Layer

Input:

e_1, e_2
e_1-hints, e_2-hints

Output:

Path-list, P

Algorithm:

1) ***Identify the set of XML documents that might be of interest***
   a) Based on e_1, e_2, e_1-hints and e_2-hints
      Generate strings for XPath queries
      Collect documents that contain results for the queries - docset
   b) Based on e_1 and e_1 hints
      Generate string patterns
      Collect documents that contain the patterns - docset1
   Based on e_2 and e_2 hints
      Generate string patterns
      Collect documents that contain the patterns – docset2

2) ***Find direct links***
   Siblings or parent-child relationships exist within a document between e_1 and e_2.
   Also e_1, e_2 can be either tag or text element.

3) ***Find indirect links***
   a) Common parent at nHops away
      e_1 is related to any of the elements in the e_2-hints by a common parent that is nHops level from the nodes
      Else, E2 is related to any of the elements in the e_1-hints similarly

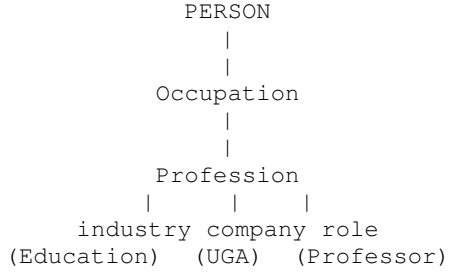
   b) Segment matches
      There exist matches between documents in docset1 and docset2 at sub-tree Level – should have identical nodes, values along with the structure

Also, indirect links that come under the second category are based on finding matching segments between XML documents. Here, a matching segment implies that the two segments of the document should have identical nodes and values along with the structure in the XML document.

Table 6.4 Example Denoting Indirect Path in the Information Source Layer

Robinson.xml	Cai.xml
<person> <name> <given> Robert </given> <family> Robinson </family> </name> **<occupation>** <profession> <industry> Education </industry> <company>Univ of Michigan<company> </profession> <**profession**> **<industry> Education </industry>** **<company>Univ of Georgia</company>** **<role> Professor </role>** <start_date>0/0/1984 </start_date> </profession> </occupation> </person>	**<person>** <name> <given> Liming </given> <family> Cai </family> </name> <education> <Univ>Texas AM University</Univ> <Degree> Ph.D. </Degree> <Year> 1994 </Year> </education> **<occupation>** **<profession>** **<industry> Education </industry>** **<company>Univ of Georgia</company>** **<role> Professor </role>** <start_date> 0/0/2002</start_date> </profession> </occupation> </person>

Matching Pattern



This category is particularly useful in finding paths when there are no parent or sibling relationships between entities. For instance, consider a Type-3 inputs category of inputs (e_1 is a literal, e_2 is a literal), 'liming' and 'robert', which are both literal values. Based on the XPath queries, Semanta gathers documents robinson.xml and cai.xml from the Information Source layer that has related information. Segments of the documents are presented in Table 6.4. As can be seen there exists no direct paths between the inputs in any of these documents. However, exist matching segments between the documents (highlighted in bold). Also, the matching segment is

shown as a sub-tree in the table. Although there exists no direct links between the two inputs, it can be gathered from the information presented, that there exists a relationship based on their profession. This relationship is further strengthened by the fact that they were teaching for the same institution.

On being unable to find direct or indirect paths in the Information Source layer, the span can be relaxed for the next level and hints are again collected, based on which, the Information Source layer is checked for paths.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this thesis we have discussed the motivations for a tool that enhances information analysis. The semantic network, a 3-tier knowledge store consisting of the Class Base Layer, Object Base Layer and Information Source layer, was discussed in detail. Semanta leverages the evolving technologies of Semantic Web, such as RDF and RDFS to define the Class Base and Object Base layers. Information Source layer is made of XML documents. The queries that help users go beyond keyword searches were categorized as entities based queries and relation based queries and were discussed with respect to Semanta. The design and implementation details of the Semanta API, for accessing the semantic network, have been discussed. Finally, the algorithms involved in finding links were discussed with a few sample scenarios.

The remaining part of this section discusses issues that can further enhance the capabilities of Semanta. Template complex relations explained in Chapter 4 can be supported, building on the existing framework provided by Semanta, thereby enabling richer information analysis.

The queries supported by Semanta involve searching for paths in multiple ontologies and in a multitude of documents in the Information Source layer. Visualization tools that can present to the user, the process of finding paths and sub-graphs across the 3 layers of Semanta will greatly enhance its usability. Such tools can also be used to graphically select (or omit) sections of ontologies or documents, in order to restrict the search to a user-defined region.

The binding between the Information Source layer and the Ontology layers can be more strictly enforced, by having the XML documents conform to XML Schema, which in turn can be based on the ontology layers.

As the Information Source layer begins to increase rapidly in order to reflect the dynamic nature of World-view, the need to be able to refer to entities in parts of other documents will become essential. This can be accomplished by using XLink and XPointer technologies and incorporating support for these technologies in Semanta.

REFERENCES

[AL02] M. Ashburner and S.Lewis, "*On ontologies for biologists: the Gene Ontology - uncoupling the Web*", Silico Biology, Novartis Symposium 247: 66-83, 2002.

[AS03] K. Anyanwu, A. Sheth, "*The rho operator. Enabling Querying for Semantic Associations on the Semantic Web",* The Twelfth International World Wide Web Conference 20-24 May 2003, Budapest, Hungary

[Dill03] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. Tomlin, J. Zien, "*SemTag and Seeker: Bootstrapping the semantic Web via automated semantic annotation"*, The 12th International World Wide Web Conference, Budapest, Hungary, May 2003.

[DSMR98] D. Fensel, S. Decker, M. Erdmann, and R. Studer, "*Ontobroker: How to make the WWW Intelligent"*. In Proceedings of KAW'98, the 11th Knowledge Acquisition Workshop, Alberta, Canada, April 1998

[ERCIM] Special Issue on Semantic Web, ERCIM (the European Research Consortium for Informatics and Mathematics) News No. 51, October 2002.

[G02] F. Gandon, "*Ontology Engineering: a survey and a return on experience*", Research Report of INRIA, RR4396, France - March 2002.

[GMV99] N. Guarino, C. Masolo and G. Vetere, *"Ontoseek: Content-based Access to the Web"*, IEEE Intelligent Systems, pages 70-80, 1999

[GR01] Y.Gil, V.Ratnakar, "*TRELLIS: An Interactive Tool for Capturing Information Analysis and Decision Making",* Internal Report, Yolanda Gil, Varun Ratnakar, August, 2001

[HH00] J. Heflin and J. Hendler, "*Searching the Web with SHOE"* In Artificial Intelligence for Web Search. Papers from the AAAI Workshop. WS-00-01. AAAI Press, Menlo Park, CA, 2000. pp. 35-40.

[HM03] C. Halaschek and J. Miller, "*Native XML Databases Today*", XML-Journal, Volume 04, Issue 01(January - February 2003) pp.22-27.

[HSK02] B. Hammond, A. Sheth, and K. Kochut, "*Semantic Enhancement Engine: A Modular Document Enhancement Platform for Semantic Applications over Heterogeneous Content, in Real World Semantic Web Applications*", V. Kashyap and L. Shklar, Eds., IOS Press, pp. 29-49, December 2002.

[ICI] International Consortium of Investigative Journalists – www.ici.org

[JDO] JDOM, www.jdom.org

[Jen] Jena, http://www.hpl.hp.com/semweb/jena.htm

[KT02] S. Kuhlins and R. Tredwell ,"*Toolkits for Generating Wrappers – A Survey of Software for Automated Data Extraction from Websites*", 2002.

[LHL01] T.Berners-Lee, J.Hendler, and O.Lassila, "*The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*", Scientific American, May 2001

[LHR00] J. Lowrance, and I. Harrison and A. Rodriguez, "*Structured Argumentation for Analysis"*, in Proceedings of the 12th International Conference on Systems Research, Informatics, and Cybernetics: Focus Symposia on Advances in Computer-Based and Web-Based Collaborative Systems, Baden-Baden, Germany, pp. 47-57, Aug 2000

[LRST02] A. Laender, B. Ribeiro-Neto, A. Silva, and J. Teixeira, "*A Brief Survey of Web Data Extraction Tools"*, in: SIGMOD Record, Volume 31, Number 2, June 2002

[NLM03] NLM (National Library of Medicine), 2003 UMLS Knowledge Sources, 14th Edition

[Ont] Ontoprise® GmbH, http://www.ontoprise.com

[PG] A. Pretschner, S.Gauch, *"Ontology Based Personalized Search"*, In Proceedings of, 11th IEEE International Conference on Tools with Artificial Intelligence, pp.391-398, Chicago, November

[Prot] Protégé, http://protege.stanford.edu/index.html

[RDF] Resource Description Framework Technical Report, http://www.w3.org/RDF/

[RHF02] B. Rosario, M. Hearst, and C. Fillmore, "*The Descent of Hierarchy, and Selection in Relational Semantics*", in Association for Computational Linguistics-02, July, 2002.

[SAK03] A. Sheth, I. B. Arpinar, and V. Kashyap, "*Relationships at the Heart of Semantic Web: Modeling, Discovering, and Exploiting Complex Semantic Relationships*", Enhanceing the Power of the Internet Studies in Fuzziness and Soft Computing, M. Nikravesh, B. Azvin, R. Yager and L. Zadeh, Springer-Verlag, 2003 (in print).

[Sem] Semagix Ltd, http://www.semagix.com/home/

[SFJCM02] U. Shah, T. Finin, A. Joshi, R. S. Cost, and J. Mayfield, "*Information Retrieval on the Semantic Web*", 10th International Conference on Information and Knowledge Management, November 2002.

[SS98] K. Shah and A. Sheth, "*Logical Information Modeling of Web-accessible Heterogeneous Digital Asset",* In Proceedings of the Forum on Research and Technology Advances in Digital Libraries (ADL), Santa Barbara, CA, April, 1998.

[STP01] A. Sheth, S. Thacker, and S. Patel, *"Complex Relationships and Knowledge Discovery Support in the InfoQuilt System,"* LSDIS Lab Technical report, September 2001

[W75] W.A. Woods, "*What's in a link: foundations for semantic networks"*, In D.G. Bobrow and A.M. Collins, (Eds.), Representation and Understanding: Studies in Cognitive Science, New York: Academic Press. p. 35-82, 1975.

[Xind] Apache Xindice, http://xml.apache.org/xindice/

[XML] eXtensible Markup Language Technical Report, http://www.w3.org/XML/

[Xpat] XPath Technical Report, http://www.w3.org/TR/xpath

[YG02] B. Yang and H. Garcia-Molina, "*Efficient Search in Peer-to-Peer Networks*", In Proceedings' of the International Conference on Distributed Computing Systems (ICDCS), July 2002.