SEMANTIC INTEROPERATION FOR HETEROGENEOUS INFORMATION SOURCES: FIRST STEPS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Jan Frederic Jannink
February 2000

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Gio Wiederhold (Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Mark Musen

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Hector Garcia-Molina

Approved for the University Committee on Graduate Studies:

Abstract

Ever increasing amounts of information are available in digital form for use in existing and emerging applications. Data sources, formats and descriptions are accessible in a diversity unimaginable a few years ago. This information seldom comes with a complete specification or schema, even though much of it contains some regular structure. Existing specifications or ontologies, developed separately from the data, are of no direct benefit in organizing such volumes of data. Tools are needed to assist domain experts linking information from diverse and changing sources.

This thesis presents an algebraic framework for managing semantic heterogeneity between information sources, with particular focus on one large scale repository developed using the algebra. This repository is a graph of dictionary terms related by their definitions as extracted from an on-line Oxford English Dictionary resource. The application for the repository is a set of tools that extract semantic relationships from the structure of the graph to support the expert in linking information from other heterogeneous sources.

The dictionary repository is a directed labeled graph, four times the size of two other lexical repositories, WordNet from Princeton U. and MindNet from Microsoft Research, but required orders of magnitude less development and maintenance effort. The operators used to build the repository apply equally well to thesauri, encyclopedias, and other dictionaries. Two algorithms over the repository provide assistance to experts in domain interoperation. ArcRank computes the most relevant arcs between terms, building on an extension of PageRank. All Pairs Similarity uses ArcRank values to compute which terms have the most similar link structure.

Acknowledgments

More than most endeavors of this type, this work is the result of many people's efforts and patience. First and foremost, I thank my wife Dorothy, without whom none of it would have been possible. There is no doubt in my mind that her love was the fuel that rekindled my desire to complete this work. Even more importantly, she opened my eyes again to the outside world, when I had lost sight of all the possibilities it has to offer. I owe a debt to everyone who believed in me when I wasn't sure I should believe in myself. Carolyn Tajnai and Suzanne Bentley at the Stanford Computer Forum helped me early on. Nice experiences, like the design of the T-shirt commemorating the 32^{nd} anniversary of the Computer Science Department, are a result of those contacts. Laura Haas at IBM Almaden Research lab gave me new confidence in my ability. The Garlic team provided a great environment to learn to enjoy coding again.

I thank Gio Wiederhold, my advisor, who gave me a great problem to work on, and remarkable freedom to pursue various approaches to attack it. Professors Erich Neuhold and Rudi Studer, on sabbatical from Germany, contributed to the creative process. My fellow research group members Prasenjit Mitra, Vasan Pichai, Danladi Verheijen forced me to stop taking shortcuts when explaining ideas. Conversations with Mark Musen, Harold Boley and Martin Kersten sharpened the technical arguments. Interaction with Stefan Decker helped close gaps in my work. True friends, such as Narayanan Shivakumar, Luca de Alfaro, Sudarshan Chawathe, have helped me in both good and bad times.

It is only fitting to remember the example set by my managers and coworkers at the Toshiba R & D center in Kawasaki, Japan, in particular, T. Kodama, T. Kamitake, and Y. Shobatake. It was their thorough knowledge of the networking field that first inspired me to consider graduate studies. I recognize my brother Jean-Luc, who just completed his own PhD, and from whom I learned the virtues of healthy competition. Last but not least, I thank my parents, who have always shown me love, and still provide moral support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Optical illusions and mirages tell us that our senses are not always capable of providing us with an accurate account of reality. Inspired by the work of Magritte, the preceding pages contain a lexical component which implicitly encourages us to question where reality ends and representation begins. Words are objects in their own right, but their principal purpose is to represent communication about other objects. Semantic heterogeneity arises out of the ambiguity inherent in the separation between words and what they represent. In what follows we examine the relevance of semantic heterogeneity to data management, and a systematic approach to dealing with it.

The emergence of the World Wide Web as a universal medium of information exchange has radically transformed our ability to access and exploit heterogeneous information sources. This shift has exposed issues that have seldom met with deep inquiry in the context of database research. The assumptions about regularity and quality of information that go virtually unchallenged in traditional database research are demonstrably not valid in this environment. Recently the author has had a first hand experience with business errors caused by semantic heterogeneity on the Web. He received a targeted mass-mailing offering "the Visa platinum card exclusively for scuba divers," despite the fact that his only affiliation to diving is the hobby of *springboard* diving. The author's home page is the only public source of this information. Such *semantic mismatch* is by no means unusual. Most mass mailings also blindly apply the feminine gender to the first name Jan.

In the field of artificial intelligence, the methodology for representing a domain is named *ontology*, a term borrowed from philosophy. Ontology concerns itself with the representation of the objects in the universe and the web of their various connections. The traditional task

of ontologists has been to extract from this tangle a single ordered structure, in the form of a tree or lattice [Sow00]. This structure consists of the terms that represent the objects, and the relationships that represent connections between objects.

We propose to defer the task of globally classifying terms and relationships, and to focus instead on composing them for use as we need them. We distinguish *context* from concept, the unit of ontological abstraction, and *composition* from subsumption, or containment, the relationships which commonly provide structure to ontologies. Context expresses the conditions under which statements about concepts are true. Composition recognizes that statements from separate sources about the same objects are not a priori true in the same context. Whenever we compose information from separate contexts, we must create a new context to define the relationships between terms in the original contexts. As the composition of contexts requires the creation of new contexts we view this composition as an algebraic operation. Given the tools of context and composition, it now becomes possible to consider potential savings in the cost of constructing and maintaining ontologies from components, over the costs of monolithic ontologies.

The following sections outline the material and contributions of the thesis along with related work. In Section 1.1 we identify the specific problems addressed in the thesis, in other words, the hypotheses. Section 1.2 defines the terms and their usage in this thesis, thus the context of the thesis. We describe the solutions we have identified and implemented in Section 1.3, and present the related work in Section 1.4.

1.1 Semantic Interoperation

This section presents a definition of the problem we address in this work. We seek methods to achieve *efficient* identification of *relevant* information in heterogeneous repositories, *verification* of its appropriateness, its deployment and *maintenance* for *application specific* needs.

The unstated assumptions of this problem statement are the following:

- relevancy is defined with respect to application requirements
- no a priori consistency between separate repositories
- data in individual sources may contain errors and irregularities

- sources may change autonomously at any time

More important than the assumptions are the consequences of the problem statement. In particular, efficiency drives us to avoid attempting to integrate the entire content of information sources. Instead, we focus on achieving interoperation of the portions of sources relevant to a specific application. Likewise, the requirement to maintain knowledge from independently changing sources means it is unproductive to convert more than the essential information from any source.

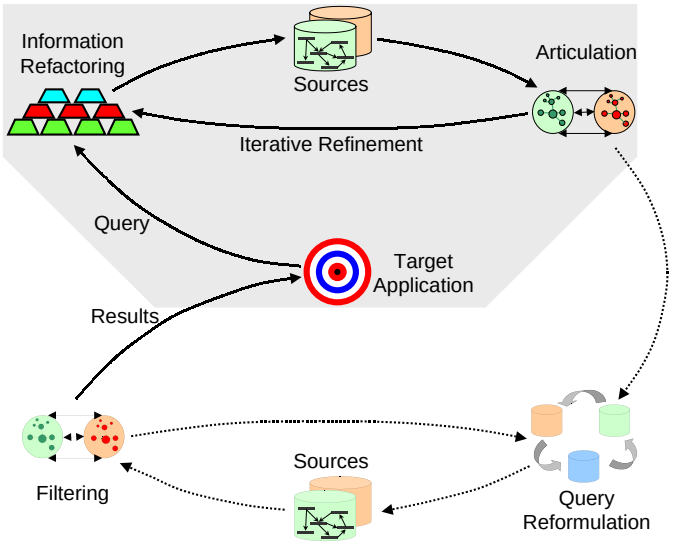

Figure 1.1: Process View of Querying in Heterogeneous Environments

We continue with a process level view of our problem space, and then a detailed illustration of the technical aspects of this research. Figure 1.1 shows a sequence of steps that are involved in obtaining a query result for an application. We assume that the application requires information from one or more sources designed for a different original purpose. The target application is at the center of the figure. Information refactoring is the process by which the relevant source information is cast into a form useful to the application. This iterative process builds a representation of the source information. Query reformulation is a process which optimizes the development of query expressions over this new representation, given the source capabilities. Both processes are necessary to fulfill generic query requirements of an application. The magnitude of scope of querying semantically heterogeneous information sources is large enough that this dissertation discusses only information

refactoring, as indicated by the shading of the upper half of Figure 1.1.

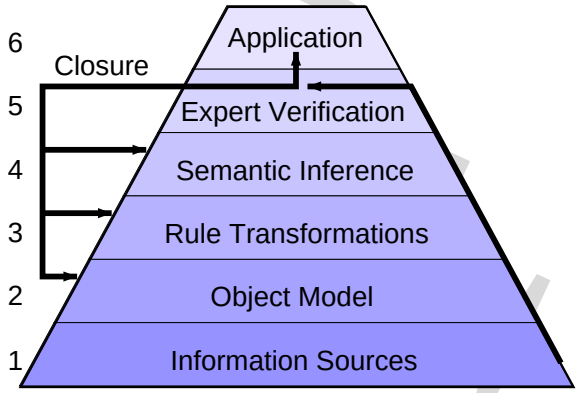

Figure 1.2: Information Refactoring for Novel Applications

We recast the information refactoring process in the trapeze of Figure 1.2. Each layer represents an aspect of the presentation of the source information to the target application. At the interface of each layer there are issues we must resolve in order to arrive at a correct specification of our research.

1. data sources storing information in their individual original formats
2. object structures representing relevant information in each source
3. rules transforming objects and relating them between sources
4. operators governing the presentation of these object structures
5. experts adapting the operators to articulate the sources correctly
6. customers and applications using the articulated source information

Figure 1.2 shows the layering listed above along with the flow of information from sources to the application. Typically, the information flow will bypass some or all of the intermediate stages between the information source and the application. In particular, when a data source is designed specifically for a given application the intermediate transformations do not need to take place. However, the increasingly common case of heterogeneous information sources and applications requires at least some transformations. Additionally, it is frequently the case that initial transformations are insufficient to cover all of the

requirements of an application. In this case, we must resort to multiple iterations of the transformation loop indicated in the figure. This iterated closure is a critical aspect of the problem space.

In Section 1.3 we show how our work addresses the problem statement without making simplifying assumptions. Also, we show our contributions at each of the layers of the problem space. Finally, we show how our motivating example itself has applications within our framework.

1.2 Terminology

The research in this thesis finds itself at the crossroads of database systems and artificial intelligence. Since this work considers the problem of composing information from multiple heterogeneous sources, it is fitting that the first task of importance is to reconcile the vocabularies of these fields in the context of this work. Indeed, the terms relating to the problems covered in this thesis are not the same in the two fields, nor are the definitions of these terms entirely consistent within the same field. The list below correlates relevant terms from both fields that have overlap in their meanings. To the left in the list below we have the term or terms from the database field which relates to the term from the AI field to the right. Wherever a term is in boldface in a particular row it is a preferred term in this thesis.

- Tuple/**Object** $\leftrightarrow$ Instance/Frame
- **Term**/Reference $\leftrightarrow$ Slot/Attribute
- Class/Relation $\leftrightarrow$ Frame model/**Domain**
- **Relationship** $\leftrightarrow$ Facet
- View $\leftrightarrow$ **Context**/Microtheory
- **Wrapper** $\leftrightarrow$ Extractor
- Mediator $\leftrightarrow$ Facilitator/**Articulation**
- **Repository**/Source $\leftrightarrow$ Knowledge Base
- **Schema**/Metadata $\leftrightarrow$ **Ontology**

We will generally use the terminology listed above in boldface. In Chapter 2 we provide formal definitions for the most significant terms above, as they apply to the thesis. For now, to clarify the discussion that follows, we provide informal definitions for these terms below:

Object

an abstraction of a physical entity, or a concept

Term

a named reference to an object

Domain

an information scope where terms are used consistently

Relationship

a mathematical relation between objects of a domain

Context

an object that specifies semantic consistency constraints on sources

Wrapper

explicit specification of information in an underlying source

Articulation

explicit linkages between sources and a target application

Repository

an information source associated with a domain

Schema

a structural specification for a collection of objects

Ontology

a set of terms and the relationships between them

We present an example centered around the term 'dog'. The dictionary does not associate a gender with the generic definition of dog. Dog in this context is associated with both genders of the animal. However, in the specialized domain of dog clubs, the object 'dog' is only an abstraction of the male of the species. An ontology which relies on both repositories for its definition of dog, must reconcile the differing views with an articulation.

The only cases in which the above terms will be used with differing definitions, is to describe related work, when that work's definitions deviate from the above. We now define some terms which are derived from the results of our research, but have not generally been used in either of the database or the AI communities.

Interoperation

use of autonomous sources without affecting their autonomy

Consistency

a relation specifying that term meanings be everywhere equivalent

Congruity

a relation of relevancy of source data to application requirements

Similarity

a relation of semantic kinship between information in distinct sources

Closure

operations iterated over a source to obtain a steady state output

1.3 Framework and Contributions

This research introduces a novel approach to describing the process of extracting information from disparate sources and enabling it to interoperate. Figure 1.3 shows how the results of the work are applied in a practical setting such as supply chain management. Our contribution, illustrated to the left of the dashed line, is a repository that supports the development of an articulation development tool. This tool in turn simplifies the job of constructing articulations between diverse information sources, such as supplier and facility databases, as shown in the figure.

The following are the three contributions of this dissertation:

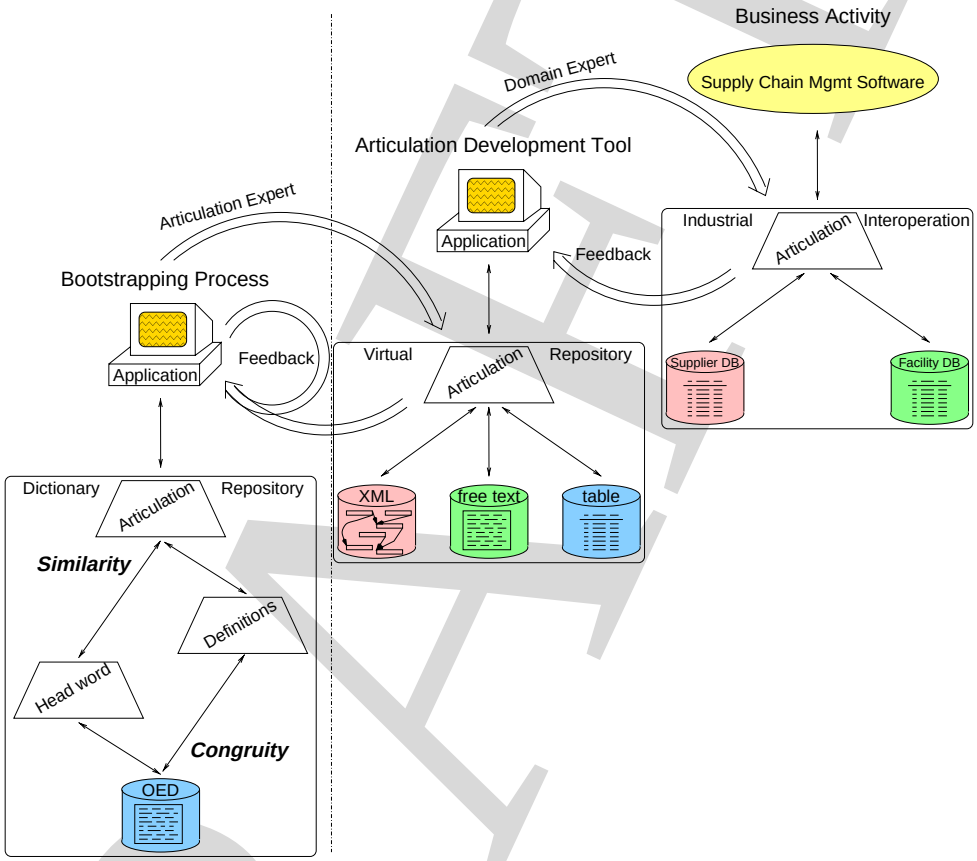

Figure 1.3: Interoperation in Supply Chain Environment

- framework for rapid development, and maintenance of articulations
- dictionary repository of semantic relationships between terms
- algorithms for extracting structural relationships from graphs

These contributions are made possible by the development of the simple framework of an object model and rule language. These two form the basis of an ontology algebra, which allows an algebraic expression of information extraction and composition. The algebra, important in its own right, is incompletely specified in this thesis, and is presented simply as infrastructure to the main body of work.

Our first work is a simple object model, AMO (Atoms Modeled in Objects), in the spirit of OEM [GCCM96] and YAT [CDSS98]. AMO represents objects, their contents and the objects' relationships to other objects. AMO is designed to maximize simplicity of

conversion from HTML, XML, database tables, and also text files. The model captures both the reification of uninterpreted data into objects, as well as the transformation of objects to relate them to others. We use AMO as our bridge between the first and second layers of the problem space as depicted in Figure 1.2.

We construct, and maintain repositories with the rule language AMORL. This rule language consists of nine functions on object patterns. These functions are classified into the categories of constructors, connectors, editors and converters. An object structure can be transformed into any other using these rules, and the common portions of any structures can be mapped into a single structure. We will show that there is a serial ordering of rules that, when interpreted sequentially, results in the same object structures. AMORL bridges the representation gaps between object structures extracted from different sources.

In order to motivate our work through an example, we use an on-line dictionary, namely the Oxford English Dictionary, second edition. We begin by extracting a glossary of terms from the head-words of the dictionary, and separately, a set of definitions that match the glossary. Then we generate a graph from the glossary, such that each use of a word in a definition results in an arc between the defined and defining word. The extraction requires knowledge of the dictionary domain, as well as the language, The combination of the glossary and definitions into a graph requires knowledge of English morphology, and the idiosyncrasies of usage in the dictionary. Manipulating this data set is not a toy problem. Its 510,000 defined terms and eight million definition words defy any attempts at systematic manual treatment.

Having generated the dictionary graph we demonstrate our application of the graph structure. We apply a ranking algorithm to the nodes of the graph, then to its arcs, in order to express for each given term which are the most important terms in its definition and which are the most important terms that use the given term in their own definition. With this ranking, we are able to generate classes of terms that have strong kinship to each other, and determine which terms subsume others, and which terms they specialize. This is a demonstration of a semantic decomposition of a large graph structure into smaller hierarchies. The benefit of this decomposition is that these classes of related terms may be used to bootstrap the process of relating other information sources that use varying nomenclature.

The semantic algebra is a higher level language built on top of the rule language, in the

same way relational operators depend on selection conditions, join parameters and projection attributes. The algebra is a set of composable operators that transform repositories into repositories. The novel aspect of the algebra is that a form of closure is applied to the operators, such that each successive application of the operator to the initial source represents a change to the exported object graph. The iterative application of the operator terminates when a fixpoint is reached (and the repository no longer changes), or alternatively when a termination criterion has been met.

Our presentation of the semantics of the algebra shows that it adequately represents the creation, refinement and maintenance of articulations over information sources. We cover the role of the termination predicate, a unique feature which enables the incorporation of closure into the algebra. We discuss the role of the expert and how the expert's reasoning capabilities affect the power of the algebra. We also examine how an inference engine supports the expert in the refinement of the representation of information sources. It is at this level of abstraction that the inference engine provides the greatest level of support to the expert.

There has been a significant amount of work touching on various aspects of this research, but semantic heterogeneity has also been a subject of intense study in philosophy, literature and art. Section 1.4 devotes space to both of these types of investigations.

1.4 Related Work

The starting point for the work in this thesis comes from a proposal for an algebra for ontology composition [Wie94]. This proposal foresaw problems in maintaining knowledge combined from autonomous knowledge sources. In this section we examine some of the other work that is relevant to the thesis.

1.4.1 Research in Related Areas

Integration of data, knowledge and systems is an area that has been gaining attention in recent years. The relevance of interoperation as opposed to integration has become manifest with the growth of the Internet. The literature on object models and rule languages appears in a wide range of domains, so we restrict ourselves here to citations in the database field. Citations on formal contexts and articulations appear primarily in the area of artificial intelligence.

Semantic integration is the focus of the research in [BJBB+97], and it presents an agent architecture, in which ontology agents mediate in cases of semantic heterogeneity. The derivation of the domain models used by the agents is not considered. The system in [GKD97] establishes a reference schema for the information it integrates. In practice each application requires its own schema, so there is a double translation necessary for any information the client applications receive from data sources. As the reference schema must serve many applications, it must also be extremely general, and simultaneously highly detailed. Such a schema is sensitive to any change in the sources it integrates. The difficulties in creating a reference schema independent of client applications are not considered in their work.

The Tsimmis project [PG95] introduced the object exchange model (OEM) [GCCM96], which predates the extensible mark-up language XML, and is quite similar. The IFO model [AH87] considers semantics, but does not focus on issues relating to heterogeneity. YAT [CDSS98] is a model that has the property that it is self describing, and can represent heterogeneous sources, but lacks facilities to restructure them for consistency.

Sequence Datalog [MB95] is a language that can manipulate ordered sequences. The ability to handle sequences of objects is a vital part of restructuring languages. EDITOR [AM97] is a computationally complete language for restructuring text documents, but does not have an object model. This language has a well-defined class of programs that are polynomial restructurings, and therefore efficient.

Description Logics (DL) are a focus of attention in [Hul97] and are used to represent the information content of sources. There is also discussion of efforts to integrate DL with query languages. The research published in [CG97] introduces a set of primitive operators on objects in semistructured data graph, in order to express change in the graph as an edit script. This work does not consider semantic inconsistency, as the graphs are assumed to be snapshots in time of the same structure.

The notion of articulation between *microtheories* is presented in [Guh91]. Microtheories are contexts, that are statically linked through their articulations into a larger complete ontology, such as the CYC system. Transformation of microtheories, and the repercussions on the articulations are not considered in this work. This system ignores the need for maintenance when knowledge about a domain changes. The Open Directory [Net99b] is a novel approach to the construction of a large information repository, which relies on thousands of editors to maintain the structure as it grows. The editors serve in effect as

articulations for their areas of expertise.

Some work on the use of context to handle database object semantics is found in [KS96]. This work uses a semantic proximity relationship, but does not discuss relevance of source data to a client application. The system in [HM93] uses gradations of a consistency relationship in relating information sources. In considering semantic reconciliation between data source and data receiver, the work in [SM91] covers some issues relating to congruity.

Tsimmis is geared towards query answering, and wrapper specification is considered orthogonally to query answering with a separate mediator specification language (MSL). A more recent version of this language, TSL, or tree specification language, with more efficiently computable capabilities is described in [VP99].

Tsimmis builds wrappers and mediators that are capable of handling some semantic heterogeneity, but not systematically nor explicitly. For example, a new relation may be created by manually defining the join of two relations in underlying sources with semantically differing names. Wrappers may resolve conflicts and inconsistency, but do so silently.

Query languages for semistructured data such as Lorel [AGM+97], UnQL [BDHS96], XML-QL [DFF+98] skirt the issue of heterogeneity entirely. Wrappers that provide data for these query languages are generated independently, specified with a separate language, and must be maintained separately from the query system.

Recent theoretical results from [NAM98] define the problem of finding common schema in semistructured data, which roughly corresponds to our `Intersection` operator. MDM, the data model of the Rufus system [RS91] describes *Aggregate* and *Partition* operators, which we subsume in the `Summarize` operator we first present in Chapter 2.

The Onion system [MKW00], building on the work presented here, investigates the problems of applying the `Intersect` operator to semistructured data. A taxonomy of information extraction techniques [CJN+00] provides a foundation for the choice of the `Extract` and `Filter` operators.

In the past few years the problems associated with ontology merging and alignment have become active areas of research. The medical informatics field has played a pioneering role in investigating this integration problem.

The medical informatics community has had a long involvement in efforts to integrate and reuse separately designed and maintained knowledge sources. The UMLS system [COS98] is the largest of these systems. Recent efforts to improve access to these integrated knowledge sources include [Pra97] and [OSSM99]. Semi-automatic integration of

medical thesauri is described in [DHR+98]. Other work on integrating thesauri for automatic translation, and other applications is presented in [NTR98]. Governmental organizations are also involved in efforts to enable mediation between autonomously developed systems, as evidenced by work on INEEL [PHW+99].

A rule-based approach to semantic integration is presented in [BCV99]. It uses a description logic to generate a shared ontology for the source information, and rules to map terms to the common format. However, the notion of maintenance and considerations of scalability are lacking from the discussion. Also, there is no description of the integration process, as an application of an algebra over the source domains.

The fundamental role of semantic relationships in an algebra is unique to our work, but the relationships themselves appear in other work as well as ours.

In considering semantic reconciliation between data source and data receiver [SM91] covers some issues relating to congruity. It does not discuss maintenance issues nor the actual algorithms used to achieve semantic reconciliation.

WHIRL [Coh98] uses textual similarity to find co-referent terms in distinct sources with high accuracy. The textual similarity measure it defines is one example of the multitude of initial similarity measures that approximate the similarity relationship between sources.

Techniques for the transformation of ontologies and program specifications are discussed in the context of category theory. While this work describes transformations where the domains are complete or exception free, it lacks flexibility to deal with real world data and erroneous specifications.

In [UHW+98] the problem of reuse of ontologies is considered, and a handcrafted adaptation of an ontology is presented. This work applies a one-time transformation of the ontology to fit the requirements of the target application. The notion of maintenance is not considered in this work, as changes to the specific source ontology are infrequent.

The work on specification morphisms [Smi93] develops a system to iteratively refine a program specification into a working application. Our work follows similar principles for deriving contexts from sources, and refining them for use in a target application. It extends the notion of specification to allow for inaccuracy, inconsistency and incompleteness found in real world information sources.

Dictionaries have been the subject of computer investigations for approximately forty years. Large scale lexical corpora have been constructed in the the past ten years, and

automated construction of these thesauri has occurred in the last five years. In particular, the computational linguistics community is interested in so-called semantic parsing of dictionary entries.

Some early work on constructing taxonomies[Ams80] and extracting *semantic primitives* [Dai86] used a graph generated from the dictionary definitions. MindNet [RDV98] is an example of a *lexical knowledge base* that relates terms according to some two dozen relationships. MindNet is generated by phrase parsing in the dictionary. attempts to distinguish semantic relationships between terms by the sentence structure and keywords in the definitions. A freely available system based on the Webster's dictionary data is found in [CL 98].

The WordNet system [MBF+90] is a corpus of nouns, verbs and adjectives that lists lexical relationships between entries in the system. It is publicly available and has been widely used in other linguistic and computational research. It is specific about the relationships between entries, but is therefore limited to a small set of possible relationships. Also, it separates the different parts of speech into separate categories, and is complete only in limited domains.

A number of new algorithms exploit the structure of large graphs to extract semantic features from them. The main focus of these algorithms is the World Wide Web, but they apply to directed labeled graphs in general.

The PageRank algorithm for ranking the importance of web pages is due to [PB98]. PageRank is a flow algorithm over the graph structure of the World Wide Web that models the links followed during a random browse through the Web. It is the starting point for the algorithm we use to determine the strength of the relationship between dictionary terms.

Latent semantic indexing [DDL+90] and hypertext *hubs and authorities* [Kle98] exploit properties of eigenvectors to answer queries over a corpus of text documents or web pages. The eigenvectors are computed from the adjacency matrix of a graph representing the structure of the corpus. These methods reject a priori stop words such as 'the' or 'and', and any words that appear overly frequently in the document corpus. LSI does not measure either the relative importance of relationships between terms. However, the underlying mathematics of these systems are close to our dictionary application.

In the domain of the World Wide Web, there is a great interest in finding related pages. Most Internet portals, such as Netscape [Net99a], and news services now provide access to similar pages or related articles in their collections. Dean and Henzinger have done research

on finding structurally similar pages in [DH99]. An efficient all pairs proximity algorithm is presented in [GSVG98].

The starting point for the relationship extraction algorithm is the DIPRE algorithm in [Bri98]. Other work that uses the notion of patterns and sample sets to extract relations can be found in [GW99]. In [CF99], on the other hand, there is a focus on universal techniques for extracting relations from web pages. The notion of related web pages in [DH99], and that of clustering search results in [ZE99] are closely tied to the kinship relationship we extract from the dictionary data. In a similar vein to our work the CLEVER group [pro99a] uses the hubs and authorities approach, as defined by Kleinberg, to cluster web pages.

1.4.2 Semantic Heterogeneity in Literature, and Art

Finally, we dedicate a section to the peripheral work that illustrates the issues of semantic heterogeneity more succinctly and clearly than any examples or toy problems can. An insightful analysis of Magritte's painting, written by the philosopher Foucault, appears in [Fou73]. The worlds displayed in Magritte's work, called heterotopias by Foucault, are powerful examples of semantic mismatch. In particular, Magritte's use of lexical components within the art is disconcerting, because we are irresistibly drawn to interpret it. Doing so immediately changes the meaning of the entire piece within which the words appear.

Figure 1.4: Upside Down

Another strongly visual example of the semantic issues resulting from the inclusion of text into art is a design by Scott Kim in Inversions [Kim81]. The design, shown in Figure 1.4, when taken as text reads 'upside down', no matter which orientation you give it. Other works that contain multiple heterogeneous meanings within single lexical domains are listed

with brief explanations below.

Lewis Carroll's Jabberwocky [Car90]

> 'Twas brillig, and the slithy toves
> Did gyre and gimble in the wabe;
> All mimsy were the borogoves,
> And the mome raths outgrabe.

The verses of Jabberwocky are infinitely interpretable, since most of the terms are invented. The invented terms are, however, so strongly reminiscent of actual English words that we do attempt to make sense of them. It is remarkable that the verb 'to chortle' entered into the English language as a result of appearing in this poem.

Howard Chace's Ladle Rat Rotten Hut [Cha56]

> Wants pawn term, dare worsted ladle gull hoe lift wetter murder
> inner ladle cordage, honor itch offer lodge, dock, florist.
> Disk ladle gull orphan worry putty ladle rat cluck wetter ladle rat hut,
> an fur disk raisin pimple colder Ladle Rat Rotten Hut.

This fairy tale, reinterpreted, makes no sense at all when read literally. However, when read out loud, or even better, having the text read out loud by someone else, makes the words sound like the original tale:

> Once upon a time, there was a little girl who lived with her mother...

C. van Rooten's Mots d'Heures, Gousses, Rames [van67]

Raseuse arrête, valet de Tsar bat loups	Roses are red, violets are blue
Joues gare et suite, un sot voyou	Sugar is sweet and so are you

The above verse, when read aloud with proper French pronunciation, sounds quite like the well known Mother Goose nursery rhyme shown to its right.

Smullyan's logic puzzles [Smu87]

> Two cards have *successive, positive integers* written on them. Two logicians each take one of them, without seeing the other's card. The ensuing dialog, in which each logician speaks in turn, is a sequence of n statements,
>
> A: I don't know your number
>
> B: I don't know your number
>
> ...
>
> A or B: I know your number
>
> What are the numbers on the cards?

This logical puzzle is an example of how additional information collapses two distinct world views into a coherent state. Simply stating, back and forth, their ignorance of the cards values, the logicians eventually rule out one of the possible values of their colleague's card. If A makes the n^{th} statement, the cards show the values $[n, n+1]$, if B makes the last statement the cards have the values $[n, n-1]$.

1.5 Thesis Outline

The remainder of the thesis proceeds as follows. Chapter 2 presents the framework for the following chapters. This work initially appeared in [JPVW98]. Chapter 3 fleshes out the claims of the thesis which are then demonstrated in the subsequent chapters. Chapter 4 describes the Webster's dictionary source and the iterative process of defining a wrapper over the source. This work was presented in [JW99]. Chapter 4 also shows how the work on the Webster's source was extended to the much larger Oxford English Dictionary, with little additional effort. Chapter 5 presents the ranking algorithms which enable restructuring of newly incorporated sources. [Jan99a] describes these algorithms. Chapter 5 also describes iterative techniques for extracting subsuming, specializing, and kinship relationships from ranked sources. These results are reported in [Jan00]. To provide further infrastructural details, Chapter 6 investigates properties of an algebra defined over wrapped sources. This is work that appeared in [JMN+99]. Conclusions and directions for future work are given in Chapter 7. Appendix A describes the relationship between our object model, and semi-structured data, XML, Frames, as well as UML. The appendix also shows how to convert between these models and ours. Finally, Appendix B presents a script which transforms the raw source data from the Oxford English Dictionary into two relations, the first associating the head words to a numerical key, the second all definition words to the key of the head

word they define.

Chapter 2

General Algebraic Approach and Contributions

Chapter Outline

In this chapter we provide an overview of the background for the thesis work. We begin by defining ontologies, domains and articulations. We proceed with a simple object model, AMO. We present a definition of context that we use to encapsulate object graphs. Contexts are the unit of semantic consistency in our framework, and are the operands of the algebra. We continue with a presentation of AMORL, a language of primitive rule operations that transform objects. We begin by listing the algebraic operators, to motivate the sections that follow. We then define and motivate our model of semantic consistency, considering two inherently semantic relationships: congruity and similarity. While the two relationships are similar, distinguishing the two is important for scalability.

2.1 Architecture

The amount of potentially useful data available on the Internet is growing at a tremendous rate. There are thousands of sites which contain overlapping or complementary information. It would be most convenient to access related information in a uniform fashion through a single interface. For example, there is enough freely available data on airline schedules and fares, car rentals and hotel reservations to plan any trip. Without a unified access interface, however, it remains much easier to book travel arrangements through an actual travel agent.

In many domains such as the sciences, freight shipping, construction, pharmaceuticals, such professional intermediaries do not exist at all. While some programmatic interfaces do exist, they are handcrafted and require specialized tools to maintain.

To remedy this state of affairs we consider two fundamental obstacles to the development of such interfaces. First, we examine the task of identifying and keeping track of exactly the relevant portions of each data source, a task which is complicated by the *congruity problem*. The congruity problem, discussed in more detail in Section 6.1.5, occurs because sources contain not only superfluous material, but also incomplete or partially incorrect data. Note that we use superfluity, incompleteness and incorrectness not as absolute terms, but only relative to a new application for which the sources were not originally designed. Second, we describe the problem of identifying the parts of different sources that overlap, because they are identical or have related content. We call the recognition of this overlap the *similarity problem*, discussed in Section 6.1.6, and it arises out of the autonomy and semantic heterogeneity of distinct information sources. The congruity and similarity relationships are inherently semantic and are in general not automatically reducible to a program or mathematical expression. We approach this limitation by considering how to compute approximations of these relationships. Therefore we introduce the notion of congruity and similarity *measures*. These are created, refined and maintained interactively with a human specialist, using an algebra to expose individual pieces of the relationships.

The algebra operates on semistructured data. Inputs to the operators, as well as their outputs are directed labelled graphs. These graphs are assumed to be connected components. We define these connected directed labelled graphs below as our unit of semantic consistency. The nodes and arcs of directed graphs model terms and relationships of these data. We require connectedness because we consider disjoint graphs to have no a priori semantic relationship, and therefore to be independent information sources, or *contexts*. However, in our system disjoint graphs may *articulate*, that is, join together, according to a similarity measure defined by application requirements. Each algebraic operator listed in Section 2.1.5 takes as input graphs of semistructured data and transforms them according to a congruity measure or a similarity measure. Having directed graphs both as inputs and outputs to the operators guarantees that the algebra is composable. The `Summarize` operator provides the foundation for defining the congruity measure between source data and its new target application. The `Match` operator serves to define the similarity measure

between information sources. The remaining operators of the algebra generalize characteristic applications of semistructured data that are currently under investigation. Unary operators all take a congruity measure, while binary operators require a similarity measure. Operators we have used for one particular source are presented in detail in Chapter 5.

The novelty of the algebra is threefold. First, we have decoupled the selection of congruent parts of the source data from the determination of the articulation points between complementary information sources. Second, the congruity and similarity measures are created, refined and maintained in an iterative process using the algebra rather than a separate language. Third, we selected operators of the algebra to mirror classes of applications of semistructured data, rather than low level abstract primitives which are difficult to compose into meaningful operations. However, the algebra still retains the ability to compose and reorder its operations according to formal rules.

2.1.1 Repository Instance

Typical Web sources are semistructured, that is, they contain well defined structural elements, but do not have a fully regular schema. There is a growing literature on generating wrappers for individual semistructured data sources [MMK98], and on the subsequent access of the data through a query interface [LRO96]. Also, there is some research in the area of view maintenance of these wrapped sources. However, the wrappers seen today are handcrafted using languages separate from the query language, and the primary assumption for view maintenance is that the source structure and formatting is stable. In practice, handcrafting wrappers is only feasible as long as the total number of data sources is small, and as long as the sources themselves are stable. When the data source is large as well as irregular handcrafting the wrapper in itself becomes an onerous task.

Our initial experiments were with an on-line version of the 1913 Webster's dictionary that is available through the Gutenberg Project [PRO99b]. The original dictionary is a corpus of over 50 MB containing some 112,000 terms, and over 2,000,000 words in the definitions alone. The source data of the dictionary was originally scanned and converted to text via character recognition software, and therefore contains thousands of errors and inconsistencies. Dealing with errors helps our approach to achieve the robustness needed for real-world settings. Our target application for the dictionary data is the construction of a graph of the definitions from which we can determine related terms, and automatically generate thesaurus entries. Misspellings and incompleteness in the terms and definitions,

as well as errors in the labelling of the data resulted in over five percent of the data being incorrectly interpreted using a naive wrapper. Later we developed an entirely new repository based on the Oxford English Dictionary, Second Edition [Tom99], a corpus of 570 MB. The resulting structure is over four times the size of the Webster's repository, and demonstrates the scalability of our approach. In Section 6.1.7 we show how we iteratively refine a simple wrapper using the `Summarize` operator from our algebra to reduce the exception rate below one percent.

Bringing similar information together from multiple data sources introduces the problem of semantic heterogeneity. The data in an individual source serves a purpose originally defined by the source's maintainer. This purpose is partially spelled out in the schema and structure of the data set. The applications which use the data usually express implicit assumptions about the data, which are not contained in the data itself. When gathering information from multiple sources to apply to a new use, it is necessary to explicitly resolve differences in the semantics of the data. It is formally sufficient to resolve only the differences that are germane to the new context to which the data is to be applied. Unfortunately, there are no computational methods for recovering implicit semantics from multiple data sets. Instead, we developed semi-automatic techniques for exposing incorrect, incomplete and inconsistent data within a source. Such semi-automatic techniques reduce the burden of manually deriving the source semantics that are relevant to the new applications of the source data.

We find an example of substantial source heterogeneity from the NATO [NAT99] Web pages starting at: `http://www.nato.int/`. This site contains information about the NATO alliance, as well as the Partnership for Peace alliance, and contains links to various pages from their member nations. In particular, two pages provide links to web pages maintained by the governments of the members of these two organizations. On the surface, these pages would appear to represent similar information, but they revealed substantial heterogeneity when we used them in combination. Beyond the differences in language of expression, it was necessary to consider the categories of differences below in order to work with the data we collected. Using the algebra's `Match` operator allowed us to iteratively define conditions for combining NATO member government data from their Web sites.

Since we assume sources are autonomous, they may change at any time. We are able to separate changes of a source that affect their relevancy to our application from those changes that affect their similarity to other sources with which we combine them. These definitions

form the underpinning for the formal definitions of the congruity and similarity measures. We begin by pinning down a definition of ontology, domain context and articulation.

2.1.2 Ontologies

The term ontology is used to mean different things by almost every researcher who works with ontologies. We have developed a hierarchy of definitions that cover the most common usages of the term [DJ00]. In this work, however, we will stick to a very simple mathematical definition of ontology.

$$O = (S, r_1, ..., r_n, m^*) \tag{2.1}$$

An ontology O is a set S, whose elements form the domain of one or more relations r_i, and are also the domain of a mapping m^* whose range are objects and concepts in the external world. This mapping may be implicitly defined in the labels applied to the items of the set, or may be deducible from the relations between object items. The mapping is not directly representable, since its range contains external world objects, but it is critical to the definition. We may write a proxy function m to stand in for the mapping m^* when we agree on the terms that represent the external objects. From this standpoint, we see that an ontology represents an agreement to define the relationships between external world objects in a certain form, and the mapping m^* is the vocabulary that represents that agreement. The proxy function m mentioned above is equivalent to an interpretation function as defined for logic [GN88].

Given this definition, we now clarify the notion of ontological domain. Note that we distinguish ontological domain, from mathematical domain as used above. In the text that follows we will use domain to mean ontological domain. A domain is simply a vocabulary, that represents a set of external objects consistently. We may use the domain vocabulary as our mapping in an ontology, because each term in the vocabulary is used to represent one object from the external world. We will also consider n-tuples of terms from a domain to be part of the domain.

The above definition of ontology illuminates the need for articulations. When we want to use an ontology for a new application, or combine terms from more than one source ontology for use in an application, we must adapt the semantic mapping, or more properly the proxy functions associated with the ontologies. Articulations enable this adaptation and allow us to access a controlled portion of one or two source ontologies, using the vocabulary

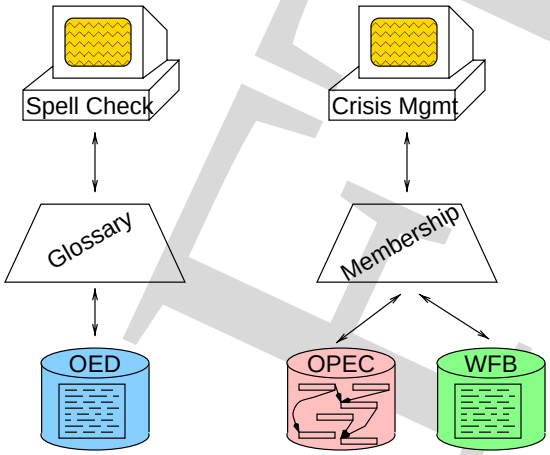

Figure 2.1: Unary and Binary Articulations

of a target application. Note that we consider both unary and binary articulations where unary and binary refer to the number of source ontologies. Articulation is needed not only to adapt sources to operate together, but also to adapt sources to the requirements of a new application for which the sources were not originally designed. Formally, an articulation is one or two functions from a source domain to a target domain, associated with composition rules over mathematical sets. We use the functions to describe how a vocabulary is transformed into a distinct target vocabulary. Each term transformed by the function is associated with composition rules that describe how the items in the ontology are viewed through the articulation.

As examples, Figure 2.1 illustrates how a spell checking application can benefit from a unary articulation that glossarizes the defined terms in the Oxford English Dictionary. Likewise, a crisis management application, can use a binary articulation that combines current OPEC membership information from the OPEC web pages, with geographical and political information about OPEC members taken from the World Fact Book. The following sections cover the definitions of the objects which are members of an ontology, and the composition rules, which allow us to construct articulations.

2.1.3 Object Model

Rather than invent yet another model to represent the objects of our ontologies, we simply allow any model that satisfies the semantic abstractions listed below. We call our instance of this model *Atoms Modeled in Objects* or AMO for short. These object semantics are

general enough to subsume existing models, and are powerful enough to simulate others. Any adequate object model must provide for the following abstractions:

Reference

object identity

Atom

object references, strings, numbers, sets

Attribute set

labeled set of atoms

Value

sequence of atoms and referenced object values

An object has a value which may contain unique named attributes, as well as uninterpreted strings and numbers, or references to other objects. A reference is an object identifier. References from an object to others represent local identifiers of the referred object. References without a source object are OIDs which are considered global with respect to the domain of the information source. Reachability of objects through repeated traversal of references in the AMO model is the sole criterion of object existence. The set of objects reachable from a given object constitute its universe. The set of objects reachable from OID (globally) referenced objects, constitute the universe of a domain. Figure 2.2 shows each of the above abstractions' graphical representation. Below we present primitive operations over AMO which relate the different parts of the model.

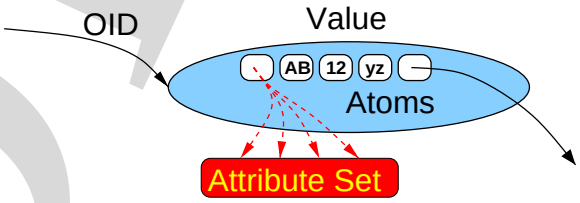

Figure 2.2: AMO Object Model

Note that only objects have an identity to which others can refer. All other components of the model are atomic values, which can not be shared, and have no existence independent of the object that contains them. Atoms may however be copied, edited and reified. Objects

have a value, which is defined as the sequence of values that compose it. This model corresponds to XML supplemented with object identity, where a bracketed `<obj>...</obj>` XML object corresponds to an object in our model, and its URI is its identifier. This style of object access corresponds very closely to applying a document object model (DOM) to XML. The object model allows us to represent, without modification, HTML and XML documents, data in Stanford's OEM [GCCM96] format, plain text and database relations. Furthermore, it is rich enough to model more complex relationships such as inheritance, and typing. This expressiveness allows it to simulate UML [Fow98] and frame [Kar92] models.

2.1.4 Rule Language

While the algebra transforms contexts and the object graphs they represent, it is also convenient to be able to express these transformations specifically in terms of operations on the objects that constitute the graphs. These primitive operations form a rule language out of which we construct the similarity and congruity expressions defined below. These expressions are constraints to the algebraic operators, as selection and join conditions are in the relational algebra. Note that the conversion operators allow transformations from values to objects, which enables their use in wrapping sources which are not already within the object model. These conversion operators also allow uninterpreted substructure of an object to be transformed into attributes of the object. Such refinement of structure remedies the differing granularities of objects which would otherwise be similar. The operators listed below form expressions that operate on an idealized data stream from the information source to the application. Expressions concatenated together are executed in sequential order, and require multiple passes of the data stream, while nested expressions represent a single pass of the data stream.

- constructor

  `create`

  generate new objects to *proxy*, or stand in for and refer to existing objects. Takes an object pattern, and an OID set as parameters.

- connectors

  `match`

generate new objects to proxy for and equate distinct objects from separate sources. Takes two object patterns and an OID set as parameters.

`inherit`

generate new proxy objects in a subsumption relationship between separate sources. Takes an existing OID, an object pattern and an OID set as parameters.

- editors

`insert`

insert an atom into object at a specified position. Takes an object pattern, an atom set and a position set as parameters.

`edit`

edit an atom within object at a specified position. Takes a source pattern, a replacement value, an atom set and a position set as parameters.

`move`

move an atom within object to a specified position. Takes a source pattern, a destination value, an atom set and a position set as parameters.

`delete`

delete an atom from object at a specified position. Takes an object pattern, an atom set and a position set as parameters.

- converters

`reify`

replace an atom in an object with a reference to an object containing that value. Takes an object pattern and a position set as parameters.

`fuse`

replace the reference(s) in the referring object with the values of the referred object in the appropriate position. Takes an object pattern and a position set as parameters.

Strictly speaking the set of operators above is not minimal. It is simple to simulate the `inherit` object, `edit` value, and `move` value rules using the other rules. All of the rules take one or more patterns as parameters. A pattern identifies one or more objects in an object graph by value or by reference. The types of pattern predicates are:

- path expressions over references
- set membership expressions over attributes
- regular expressions over strings
- relational expressions over numbers

Note that the last three pattern expression types refer to values contained within an object, whereas the first expression type refers to any path that reaches an object. Given a classification of vehicles by the medium in which they operate, and their purpose of use, `vehicle.land.recreational.*` is a path expression that will reach objects such as bicycle, skates, skis, and `vehicle.*.cargo` reaches freight train, tanker ship, and cargo plane. Membership expressions appear in path expressions, as well as on their own and are combined disjunctively or conjunctively. Assuming that vehicles have attributes `wheel`, `license` and `motor`, then the *disjunctive* `vehicle.land.recreation.*.[wheel|motor]` can return in-line skate or snowmobile, which have either attribute. In contrast, the *conjunctive* expression `vehicle.air.recreational.*.[wheel&license]` returns piper cub, but not hang glider (since hang gliders don't have wheels).

Note that a context is itself an object and that the rule language specifies how the context is populated with values from sources. Constructors create new objects, not represented directly in sources. Connectors generate proxy objects that stand in for one or more objects from sources, which may then be modified using editors and converters. In the following section we list the algebraic operators without going deeper into their definitions.

2.1.5 Algebraic Operators

The algebra consists of a composable set of operators that transform contexts into contexts. These contexts, defined in more detail in the Section 6.0.1, encapsulate ontologies with a guarantee of semantic consistency. The operators are listed below with a brief description of the operation they perform. The abbreviated form of operator names given here is also used in the thesis.

- Unary operators

  S `Summarize`

  term classification

G `Glossarize`

listing of terms

F `Filter`

object instance reduction

E `Extract`

schema simplification

- Binary operators

M `Match`

term corroboration and refactoring

D `Difference`

schema distance measure

I `Intersect`

schema discovery

B `Blend`

schema extension

Unary operators reformulate source information with respect to the requirements of the target application. `Summarize` is the canonical unary operator. It is used to establish and refine a context within which the source knowledge meets the requirements of the application. For example, S groups a set of objects by an attribute's value, by the attribute's presence or absence, or by objects' current path expression. Binary operators express the linkages between sources that are germane to the needs of an application. Playing the same role for binary operators that S plays for unary operators, `Match` or M is the canonical binary operator, and is used to develop and refine articulations between sources. M groups objects in two sources together based on attribute values and path structure. We will define the operators in greater detail in Chapter 6, for those wanting a more complete overview of the infrastructure supporting the claims of the thesis. These claims are the subject of Section 2.2, which lead us to the hypotheses defined in Chapter 3.

2.2 Thesis Contributions

This section outlines the major contributions of the thesis. Although database systems typically allow a very rich query capability over the data they maintain, there is a deep assumption that the data conforms strictly to its schema. This assumption is so pervasive in the database field that it is very difficult to present work under different conditions. In this thesis our first assumption is that it is non-trivial to bring information from multiple sources into a single consistent format. Even stronger, it is non-trivial to refactor a single large information source for a new purpose, regardless of its level of consistency for its original application. The scope of this thesis does not permit us to build a complete query answering system, but instead we focus on this lower level problem of restructuring information.

2.2.1 Framework for Rapid Development and Maintenance of Articulations

We show how the object model makes it easy to establish new structures given a model of the source information. We define how the rule language supports the restructuring of the data as inconsistencies and other irregularities become evident in the data. Taking an on-line Webster's dictionary as our example we create a 97,000 node directed graph from the dictionary definitions. We see the adaptability of the rules creating this graph when the underlying data source is updated. We reuse the rules in two other related data sources: first, a Roget's thesaurus, with a few thousand nodes in the resulting graph, demonstrates resilience across data sources; second, the Oxford English Dictionary, with one third of a million nodes, shows the scalability of the rule system.

2.2.2 Repository of Relationships between Dictionary Terms

With this new graph structure it is possible to consider a large number of relationships between dictionary terms, and the possibility of computing them from the structure itself, rather than manually extracting them. First off, is it possible to determine the most important definitional terms in the dictionary? More conservatively, can we find for a given term which are the words in its definition, that contribute most to its meaning? Also, how are similar terms related in the graph structure? As it turns out, new algorithms adapted from graph theory point in the right direction.

2.2.3 Extracting Structural Relationships from Graphs

Considering the entire graph as a flow network, we look for steady states of flow across the arcs between the nodes. This flow represents the importance, with respect to usage, of each term in the dictionary. Once the notion of network flow over the graph exists, we define further relationships. For example, similar terms in the dictionary should have similar definitions. Using the flow beteen nodes it becomes possible to quantify what similar means. With these relationships the cost of computing similar terms across other repositories is reduced to a linear time operation.

In the next chapter we present the question the dissertation answers, the hypotheses that form the basis of the answer, and develop the approach to our solution.

Chapter 3

Research Hypotheses

Chapter Outline

As suggested in Chapter 1 this thesis focuses on a narrow portion of the information refactorization problem. A single sentence characterization of the question it addresses is:

How do we begin to develop a systematic approach to the interoperation of heterogeneous sources?

The answer to this question is in three parts, and combines not only information resources but also the framework to create them and the algorithms that use them. In this chapter we present our hypotheses and our approaches to their solution.

3.1 Thesis Assumptions

In this section we present the hypotheses we demonstrate in the dissertation.

Hx. 1 *a framework to compose autonomous information sources lowers the cost of articulation development, maintenance and reuse*

Hx. 2 *a repository of relationships between dictionary terms developed using the framework demonstrates its scalability*

Hx. 3 *algorithms making use of the repository support domain experts in articulating other information sources*

Individually, these hypotheses each have merit, as we'll see in the following chapters, but together their significance is that they indicate an answer to the question posed at the

beginning of the chapter. In the next section we look at the issues brought up by the first two hypotheses.

3.2 Articulation Development, Maintenance and Reuse

Recall that in this dissertation we consider varied information sources, from plain text to relational tables. We assume that, in a first step, source information is represented by proxy objects generated using AMORL, the rule language for the AMO object model defined in Section 2.1.4. The important point that we will see is that the proxy representations and the rules that generate them are iteratively developed in the course of applying operators to the source. The notion of approximate solutions that are iteratively improved is central to the all of the operators presented in the following subsections. The iterative process is also the same when considering an articulation over two preexisting ontologies for a new target application. This similarity suggests that we can not begin to consider the problem of articulation on well defined source ontologies, without first examining the development of ontologies, via the articulation mechanism, from arbitrary data sources such as plain text or XML tagged data.

3.2.1 Development

The work presented in Chapter 4 illustrates the development of an articulation over an on-line Webster's dictionary source. The target application of the articulation is the computation of semantic relationships between the dictionary's defined terms, based on the words in their definitions. We have generalized the definition of articulation in Section 2.1.2 so that it covers refactoring of a single source with respect to a target application. Thus, an articulation is a sequence of operators, such as those listed below. Each operator applies an AMORL ruleset to objects of one or two sources, and returns the result of its operation to the target application. Articulations are unary or binary, based on the number of sources they operate on. Articulations initially represent a rough model of the source or sources, and are refined iteratively to meet the requirements of the target application. The sequence of operations listed below follows the order in which the raw dictionary data is transformed into a directed graph with head words as terms that label the nodes of the graph.

Glossarize

to construct a repository from the dictionary it is necessary to enumerate all of the items that must be considered separate, and then generate proxies for them

Extract

to associate definition terms with each entry we must find those terms within the dictionary entry, and enter them in a list associated with a proxy in the sequence from above

Match

to link dictionary entries using definition terms we need to match each extracted term to a proxy in the glossary, and generate a link to this proxy from the proxy it associated with in the previous step

These three operations illustrate both unary and binary operations from the algebra. The unary operations demonstrate the preparation of the relevant portions of the dictionary source data. The use of a binary operation on information from a single source shows that single sources can contain inconsistencies when refactored for a new application.

3.2.2 Maintenance

The resulting repository described in Chapter 5 contains just under 97,000 objects, referred to by two million arcs, and requires just under two hours to generate from the ftp source. The maintenance problem for this repository is the following: the source maintainer's goal is to add updates and revisions to the dictionary, while also correcting errors discovered in the original data. At irregular intervals, approximately semi-annually, these changes are published to the ftp site. The cost of maintenance is the effort it requires to adapt the articulation and regenerate the repository to the level of consistency it had prior to the update. We measure the cost of adaptation in two ways. First, the number of rules added and deleted in the articulation's operations. Second, the amount of time spent between the time the new source data is available, and the time a revised repository, based on the new data, becomes available.

3.2.3 Reuse and Scalability

The ability to reuse the articulation on a different source for the same application demonstrates that articulations can also save development time. We constructed a small scale

repository based on Roget's thesaurus, with two thousand nodes and 15,000 arcs between them. To test its accuracy we computed a structural similarity algorithm over the entire structure.

Verifying the scalability of the Webster's articulation required modifying it to accept the Oxford English Dictionary as a source. At around ten times the size of its original source, the 570 MB raw data file converts into a 325,000 object, eight million arc graph. When assessing the effectiveness of articulation scalability, we consider the time required to modify the articulation, the time to compute the graph, and test the data on a real problem, together with the original Webster's repository.

3.3 Dictionary Repository Algorithms Support Expert

The dictionary repository explicitly models two relationships present in the source data. First, term x appears in term y's definition, and second, term x uses term y in its definition. What is less clear is that the structure of the repository allows for the computation of other semantic relationship between words. Algorithms over the graph defined in Chapter 5 expose these implicit relationships for use in other applications. First, we compute structurally significant definitional terms. These are the terms that are frequently used in definitions. As expected, articles and common prepositions dominate this ranking. The insight is that few words contribute significantly to their rankings. The valuable measure is the one which ranks words with respect to the terms that provide a significant contribution to the words' usage. We can go on to use these rankings to find similar words in separate structures, based on the similarity of their usage rankings in the dictionary repository. For example, in Figure 2.1, the binary articulation between the OPEC web pages, and the world factbook needs an initial assessment of the most likely common vocabulary between the two sources. The dictionary repository, and a word similarity computation provides this functionality. To summarize, we use semantic relationships between the graphs terms to support the development of other articulations, and we compute these relationships from the repository's structure:

1. significant definitional terms
2. relevant arcs between terms
3. similar objects

3.4 Review

In this chapter we have asked what are the first steps in enabling interoperation of heterogeneous information systems. Our response implies that we need a combination of an operational framework to perform necessary transformations, and a vocabulary bootstrapped from the framework to assist in subsequent transformations. The following chapters piece together the research to demonstrate these claims.

Chapter 4

Articulation Development and Maintenance

Chapter Outline

This chapter investigates the refactoring of existing knowledge bases for uses that were unanticipated by their original creators. These knowledge sources are assumed to be autonomous and are not changed by the refactoring process, but may change as they are updated for their original application domain. The techniques developed for refactoring knowledge bases are therefore also applied to the maintenance of such refactored information when the underlying knowledge sources change. The autonomy of diverse knowledge sources is an obstacle to integrating all pertinent knowledge within a single knowledge base. The cost of maintaining integrated knowledge within a single knowledge base grows both with the volatility and the number of the sources from which the information originates. Establishing and maintaining application specific portions of knowledge sources are therefore major challenges to ontology management.

Rather than materializing all of the information from the sources into a single knowledge base, we present algebraic operations that enable the construction of virtual knowledge bases geared towards a specific application. Operators express the relevant parts of a source and the conditions for combining sources using AMORL defined in Chapter 2. Rules which expose the relevant parts of a source determine what we call a congruity measure between the source and its target application. The rules which articulate knowledge from diverse sources establish a similarity measure between them.

The example we present in this chapter is an on-line dictionary that is autonomously maintained [MIC96]. We use the source data from the dictionary definitions to develop a repository that can be used for such purposes as the generation of thesaurus style entries [Jan99b]. Due to its size, irregular structure and the autonomous nature of its roughly semi-annual updates (intended by the sources maintainer to bring it up to date), this dictionary has provided us with an ideal test bed to examine the issues of creation and maintenance of knowledge contexts using an algebraic methodology.

4.0.1 Preliminaries

This chapter presents a case study of repository construction from an autonomous source, for use in a real world application. In the course of developing the application, the underlying data has been updated three times, requiring repeated execution of a source refactoring operation to bring the repository up to date. We show how a principled approach based on algebraic operators has made it possible to create and maintain access to this information with a low overhead cost.

Proprietary dictionaries [Mir99] and encyclopedias [Enc99] are present on the World Wide Web, through an interface that provides access to one term at a time in the typical case. Most freely available dictionaries are at best partial, or limited to a specific domain [Ger96]. Among the most extensive freely available corpora we find WordNet [MBF+90], which suffers from being hand-crafted, and does not claim to be a complete language reference. However, there does exist an on-line version of the 1913 Webster's dictionary that is available through the Gutenberg Project [PRO99b]. The raw dictionary data is a corpus of over 50 MB containing some 125,000 terms, and over 2,000,000 words in the definitions alone.

The source data of the dictionary was originally scanned and converted to text via character recognition software, and therefore contains thousands of errors and inconsistencies. The abundance of errors in the dictionary data makes it an ideal test bed for our research. Dealing with incorrectness in sources provides our approach with the robustness needed for real-world settings. Our target application for the dictionary data is the construction of a graph of the definitions from which we can determine related terms, and automatically generate thesaurus-style entries. Accuracy in the data is important for meaningful results, since we run flow algorithms on the graph structure. Misspellings and incompleteness in the terms and definitions, as well as errors in the labelling of the data resulted in over five

percent of the data being incorrectly interpreted using a naive wrapper. We iteratively refined the naive wrapper using an operator from our algebra to reduce the exception rate below one percent.

The Webster's dictionary is an ideal data set for our purposes, in that it contains inconsistencies, and is autonomously updated. Our use for the information contained in the dictionary is typical of a demanding application, as it provides a legitimate service and it has strict tolerances for how much erroneous information it allows. We discuss, within an algebraic framework, how we establish and maintain a context that takes the source data, and makes it available to the application, meeting the application's requirements. We present the initial derivation of the context for our application, and show in the following section how the refinement process is repeated, albeit with less overhead, when the source undergoes change. In the final section we describe our application, and discuss our future directions of research.

In the next section, we describe the dictionary repository, as well as its creation and refinement. We focus on four operators used in this process, the `Glossarize`, `Extract`, `Match` and `Summarize` (G, E, M, S, respectively) operators. The G operator generates the objects in the repository, while E associates a sequence of terms with each object, and M generates references between objects based on the sequences created by E. The S operator is the primary tool for refining and maintaining a context between a knowledge source and an application.

4.1 Webster's Dictionary

We have been using data from the Webster's dictionary to research the automatic extraction of thesaurus style entries, such as for the term **Egoism** in Figure 4.1. The relationships between terms are expressed using the implicit structure contained in the dictionary, rather than explicitly marked ones such as synonyms. The purpose of the application is to serve as a tool in reducing the occurrence of lexical mismatch when merging diverse ontologies.

The edition of Webster's dictionary was originally published in 1913, and was recently (1996) converted to text format from scanned images. The resulting text is tagged using SGML to mark the parts of the definitions. Definitions from the dictionary data for **Egoism** are shown in Figure 4.2.

The dictionary is organized as a set of head words, associated with a pronunciation,

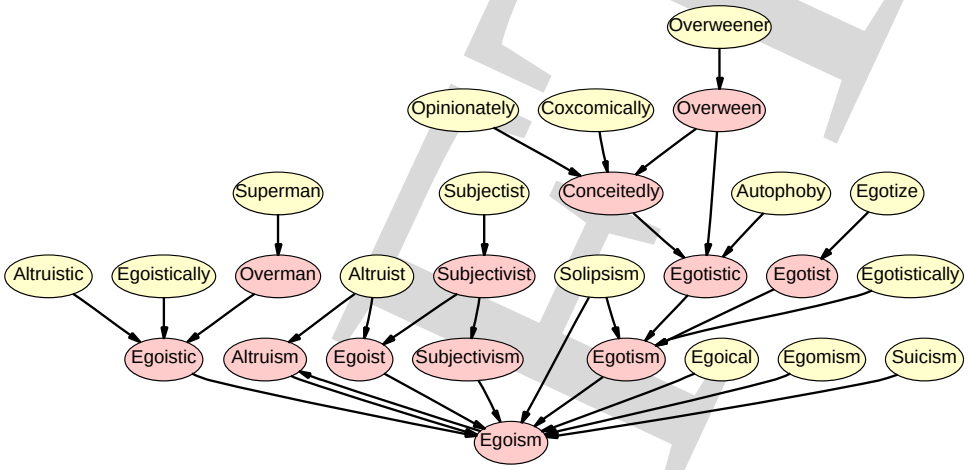

Figure 4.1: Automatic Thesaurus Extraction from Dictionary

etymology, possibly multiple definition entries, quotes and other tagged items. For the purpose of our application we were interested in the head words `<hw>...</hw>` and definitions `<def>...</def>`. Head words and definitions are in a many to many relationship, as each definition refers to different senses of a term, and each head word has variant spellings. We constructed a directed graph from the definitions as follows:

1. each head word and definition grouping is a node
2. each word in a definition node is an arc to the node having that head word

Substantial manipulation is required to bring the dictionary data into a format ready for generating a graph [JW99]. Other problems in the transformation process are listed below.

- syllable and accent markers in head words
- mis-tagged fields
- misspelled head words
- stemming and irregular verbs (Hopelessness)
- common abbreviations in definitions (etc.)
- undefined words with common prefixes (un-)

```
<p><hw>E"go*ism</hw> <pr>(?)</pr>,
<pos>n.</pos> <ety>[F.
<ets>\'82go\'8bsme</ets>,
fr. L. <ets>-ego</ets>
I.  See <er>I</er>, and
cf. <er>Egotism</er>.]</ety>
<sn>1.</sn> <fld>(Philos.)</fld>
<def>The doctrine of certain extreme
adherents or disciples of Descartes and
Johann Gottlieb Fichte, which finds all
the elements of knowledge in the
<xex>ego</xex> and the relations which
it implies or provides for.</def></p>

<p><sn>2.</sn>
<def>Excessive love and thought of self;
the habit of regarding one's self as the
center of every interest; selfishness; --
opposed to <xex>altruism</xex>.</def></p>
```

Figure 4.2: Webster Definition of Egoism

- multi-word head words (Water Buffalo)
- undefined hyphenated and compound words

Markers such as in `E"go*ism` are removed first. We discover fields that are missing an end tag, when head words are inordinately long, or definitions are missing, or contain other definitions inside them. Misspelled head words are recognized when the definitions consistently use a variant form of the term. Definitions use many forms of a word that do not appear as a head word. In particular, word forms that contain multiple suffixes rarely are head words, so we need procedures to convert definition words to head words. Likewise, abbreviations, proper nouns, and words with very common prefixes are infrequently defined. Finally, hyphenated words and compound words are very inconsistently defined and used. Thousands of cases occur where words are defined with a hyphen and used without, and vice-versa.

When a conjugated verb form appears as a head word we use it for generating graph arcs. Otherwise we stem definition words until we find a head word that matches. Also, whenever we find instances of a multi-word head word in the definitions, we prefer it over the individual words for generating a graph arc. Since words often appear multiple times in a single definition we allow multiple arcs between graph nodes. Dealing with undefined terms and spelling errors is the most complex issue in the graph generation, and accounts for the quasi-totality of the structural errors in the graph.

Even after accounting for accented characters, a naive script is unable to properly assign over five percent of the words, because of the above mentioned differences between the actual data in the dictionary and its assumed structure. Any errors in the computation of the graph would affect any subsequent computation of related terms for the thesaurus application. Therefore, we set a goal of 99% accuracy in the conversion of the dictionary data to a graph structure.

4.2 Repository Construction

In this section we show how we use the algebraic framework to rapidly generate a large object repository from a text source. The product of the algebraic operators are articulations, the rulesets that generate the repository. We show that without complete knowledge of both sources and target application, iterative construction of articulations is necessary.

4.2.1 Extraction from Source Data

In this section we cover the application of G, E and M to the dictionary source.

The `Glossarize` operator G takes source data and returns a list containing all of the objects in the source data. The effect of G is to flatten any complex object structure in a source. In the case of the Webster's dictionary, there are no objects in the source initially. We use AMORL to reify objects from the raw data. The method of generating proxy objects to represent the raw source is as follows:

1. segment data into chunks containing a minimum of:
   - one head word (marked by `<hw>`, `</hw>` tags)
   - one definition (marked by `<def>`, `</def>` tags)
2. merge chunks if head words match a pre-existing chunk
3. enumerate chunks as objects
4. label objects using head word value

The sequence is implemented in Figure 4.1. The `Extract` operator E takes source data and generates objects whose contents are a list of primitive values. In the case of the dictionary, only the tagged definition of the source is used to generate a list of string values associated with the glossarized objects of the repository. The method of extracting the definition portion of the proxy objects defined above is as follows:

1. from each object chunk, select the `<def>`, `</def>` tagged data
2. generate a list using the white space separated terms in the definitions
3. replace the current object chunk's value with the newly created list

Figure 4.1 also shows the AMORL operations to compute E over the unstructured object chunks created by G. Finally, `Match` takes the value of each object (a list computed above by E), and determines for each element of the list which object best matches it.

Table 4.1: Congruity Expression for G and E Operations

```
                        // assign contents to object
    dictionary = create(".*") {
                        // generate dictionary entries
      entry = reify("<hw>\(.*\)<hw>") {
                        // insert head word values
(1)     head_word = insert("<hw>\([^<]+\)</hw>") {
                        // remove syllable and accent markers
          self = edit("[\"‘*]", "")
        }
                        // insert definition object
(2)       definition = reify("<def>\(.*\)</def>") {
                        // insert definition word attributes
          word[] = insert("\([^ ]+\)[ \n]")
        }
      }
    }
```

4.2.2 Constructing the Congruity Expression

The dictionary repository is constructed from the raw corpus data starting with a congruity expression. The simplest form of the script that generates such an expression consists of the following AMORL operations. Table 4.1 shows the first cut of the expression that (1) generates the objects for G and (2) the sequences of definition terms for E given the Webster's dictionary source data. Regular expressions are simplified for conciseness of presentation:

This script creates an object that represents the entire source, which is then subdivided into chunks containing at least one head word and one definition, which are then extracted into separate sets within the chunk. Each script fragment such as the one above represents operations in the course of a single pass through the source data, the output of which may be passed to another script. This initial script very approximately expresses the congruity relationship between the dictionary and the thesaurus application. The final script, containing over 400 conversion operations, that perform over 300,000 transformations, is presented here, in a separate appendix. It is of note that the operation the script performs bears a resemblance to the alignment operation described in [CHR97]. What is different is that the script remains associated with the output, and is part of the definition of the context which

represents that output. In the following section, we show how the S operator allows us to capture some of the classes of conversions which enhanced the above initial script.

In the previous sections we presented the algebraic operations we use to structure the dictionary repository. Here we present a definition for the `Summarize` operator and show its use in refining the repository.

4.2.3 Summarization of Exceptions

This section shows how we take the results of one iteration of the algebraic operation, determine those results which do not match application requirements, and aggregate them to find classes of exceptions.

The S operator is a unary operator that transforms source data according to a predicate which corresponds to a congruity expression. The predicate therefore consists of a ruleset from the rule language. S creates a new object, in effect a context, that encapsulates the information of the source, and populates the object with results of an aggregation operation over the source information. The application that motivates the existence of the S operator is data classification. The aggregation over the source data effectively groups the source into equivalence classes. Given contexts c_1, c_2, a ruleset e that defines the congruity expression, the syntax of the operator is as follows:

$$c_2 = \mathsf{S}_e(c_1)$$

Formally, the matching predicate e partitions the objects of the initial context c_1 into n equivalence classes. The constructed result context c_2, is an object consisting of $n+1$ values: the first is e, and the following n values are sets $s_1 \ldots s_n$ of references to each of the objects of c_1. One of the equivalence classes of the result context is an exception class, for objects that can not match e. Since it is difficult to follow the capabilities of the S operator from a description alone, we present an extended example from our research. Following the example we describe the algorithm for establishing a congruity measure between a source data set and its new application.

4.3 Repository Refinement, Maintenance and Reuse

In this section we see how we use the algebra to iteratively refine the dictionary context, by discovering anomalies in the source data, and incorporating resulting fixes into the

consistency guarantees of the context. We show how this process is equivalent to the maintenance process when the underlying source data changes.

4.3.1 Iterative Context Refinement

This section shows how the exception classes of one iteration of an operator are expressed in new rules that extend the articulation for another iteration of the operator.

The S operator provides a simple method for assessing the contents of a context. For example, $\mathsf{S}_{\texttt{len(hw)div20}}(\texttt{dictionary})$ returns the entries of the dictionary, grouped by length of the head words. Applying this operation on the actual data revealed terms with missing end tags in the data (implying long head word length). Once the errors were identified, the rules to convert terms with missing end tags are added to the definition of the set "`hw`" above. Using S we were also able to determine that other tags were equally valuable as head words, that we needed to remove accentuation from foreign words, and discover spelling errors in the head words by analyzing the frequency of words found in definitions, but not as head words. The context refinement algorithm is as follows:

1. For i in $\{1, \cdots, n\}$
2. $\quad c_{i'} = \mathsf{S}_{e_i}(c_i)$
3. $\quad$ Generate rule(s) r_i to handle exception objects
4. $\quad$ Insert r_i into ruleset for c_i creating c_{i+1}
5. $\quad$ Generate e_{i+1}

4.3.2 Maintaining the Ontology

This section shows how updates to the information sources requires adjustments to the articulation built on top of them. These adjustments are shown to be equivalent to articulation improvements defined above.

In the course of developing the wrapper to the Webster's dictionary the maintainer of the source data performed a major revision of the source, affecting 10%-25% of all of the entries. These changes are part of an ongoing effort to correct and extend the dictionary, and they included corrections in the tagging of the entries, spelling corrections, reformatting of the text, addition of notes and comments, etc. By maintaining statistics with the S operator on the process of extracting the relevant parts of the dictionary, we were able to note which rules were no longer needed because the exception they handled had been updated. A comparison of the terms that we could not classify in the old and updated sources revealed

new errors that had been introduced in the data. As it turns out there was relatively little within the wrapper that required correction when the source changed. Having the congruity measure within the algebraic framework significantly simplified the process of identifying and handling the changes.

4.3.3 Revisions for Reuse

This section shows that the operator code developed for one dictionary source is portable and scalable. The code update required three weeks and 1000 lines of code, while the refactoring code execution time scaled linearly with the data size. The resulting repository is four times the size of the original. The articulation adaptation is also be represented as an iterative improvement. Table 4.2 shows the ratio of sizes of the two repositories. The two entries in bold are the most significant in the scalability of the algorithms. The first is relevant in the initial construction of the repository, the second in the algorithms computed over the repository structure.

Table 4.2: Repository Size Comparison

Repository Component	OED Size	Webster Size	Ratio
raw data	570 MB	50 MB	11.4
terms	510,648	112,897	**4.52**
nodes	326,703	96,800	3.38
arcs	8,117,455	2,021,549	**4.02**
unique arcs	5,270,129	1,438,532	3.66

The major insight gained by this effort is again that whenever refactoring bodies of information for a new purpose, it is crucial to evaluate that the requirements of the new application are met by the semantics of the source. The initial "naive" expression to adjust the refactoring code for the Oxford English Dictionary was developed in a few days. However, this transformation uncovered literally thousands (7%) of the definitions empty or containing single word connectors such as 'Hence', 'So' and 'Also', and thousands more containing the words 'see prev.' or 'see next.' An additional two weeks of coding was necessary to account for these factors. It is important to note that the source semantics are not incorrect but, in their existing form, were unsuited to the new application which we were targeting.

Table 4.3 gives the ratio in execution time for the construction and ranking of the two repositories. The first two phases have a computational cost that is dominated by the

Table 4.3: Repository Construction Times

Construction Phase	OED Time	Webster Time	Ratio
segmentation/glossarization	72 min.	14 min.	5.14
graph matching	134 min.	21 min.	6.38
ranking functions	309 min.	77 min.	4.01
overall times	9.5 hr.	1.85 hr.	4.60

terms of the repositories. Effectively, the first phase finds all of the repository terms, and the second phase matches up definition words to those terms. The second order cost in the first two phases is the size of the raw data, since both phases must pass through the entire data set to find the terms and definition words. The ranking phase is dominated in computational cost by the number of arcs in the repository graph, since a function is repeated computed for each arc. Overall, we observe a linear growth in computation time with respect to the size of the source data. In the next section we discuss the first tools developed to visualize the repository structure.

4.4 Graph Manipulation Toolkit

This section presents tools for graph exploration, and visualization that expose the overall structure of labeled directed graphs. The dictionary repository contains a central *kernel*, which is strongly connected, surrounded by smaller clusters, which link to the kernel, but are not linked to by any nodes. After processing by the graph manipulation tools the kernel and satellite clusters emerge from the tangle of repository links.

4.4.1 Peeler

The graph peeler is a tool that iteratively cuts nodes from a directed graph, when they have no incoming links. As an example, a singly linked list loses its first element to the peeler at every iteration, until it is an empty list. In contrast, a cycle of links never loses an element. Simple garbage collectors use this reference counting mechanism to reclaim memory space no longer needed by an application. This algorithm is linear in the number of arcs.

4.4.2 Kernel Finder

The *kernel* finder takes a graph as input, and returns the largest strongly connected component of the graph. A directed graph is strongly connected when there exists a round trip through the graph between any two nodes of the graph. The graph peeler serves as a preprocessing step to improve the efficiency of the process. Once we have computed the kernel of the graph, we examine the clusters of nodes that are not in the kernel. These so-called *satellite* clusters are not required to be strongly connected. A cluster is defined as follows: each cluster has a seed that only has arcs to the kernel. Add nodes to the cluster if they have an arc to the seed, or to another node in the cluster. The seed is always a single node.

If the seed is a pair of nodes that refer to each other, and to nodes in the kernel, we show by contradiction that this pair must also belong to the kernel (we assumed this is not the case).

4.4.3 Arc Filter

Now that we have defined the notion of cluster, it is interesting to visualize them. To do this, we first remove extraneous arcs. These are arcs from the cluster to the kernel. By removing them, we are left with the the arcs that define the cluster. The arc filter takes two graphs as input, and returns a subgraph of the first, with no arcs to nodes in the second.

4.4.4 Cluster Joiner

We may join clusters together when they share nodes. This situation occurs when a node has arcs that refer to terms in otherwise distinct clusters. By joining clusters, we have the ability to consider possible meaning associated with the aggregated set of terms. As an example, one joined cluster contained sub-clusters relating to marine dinosaurs, dinosaurs of the Triassic age, flying dinosaurs, as well as a broad category of dinosaurs. This super-cluster has single nodes that join the sub-clusters, each to the broad category of dinosaurs. The value of joining the clusters is to be able to visualize the larger structure and recognize that it has its own unity of meaning as well. Figure 4.3 shows that the clusters, as well as the joined clusters follow a zipfian distribution according to their size. The satellite clusters are sorted in decreasing order of size, before plotting the graph. We predict the likelihood of a cluster of any given size in the dictionary by constructing a probability distribution based

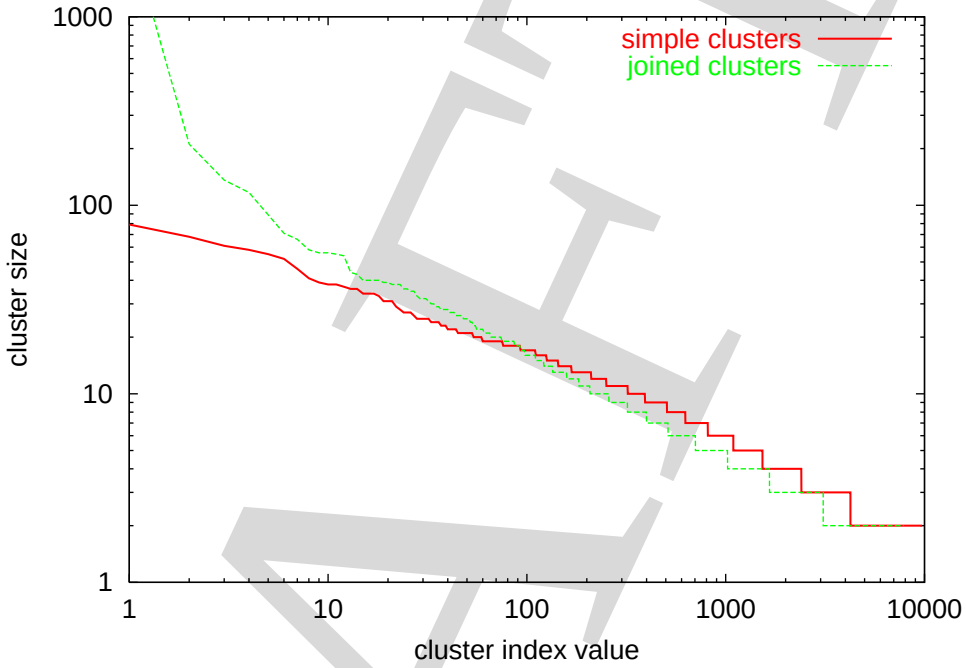

Figure 4.3: Cluster Size Chart

on an exponential. The exponent is given by the slope of the line that approximates the plots in the figure. Table 4.4 gives more statistics on cluster sizes and their number. Note the small median size of satellite clusters. This statistic is an indication of how preponderant the importance of the kernel is.

Table 4.4: Cluster Count and Size Information

Cluster Type	Number
total cluster count	50,485
singleton satellites	40,742
multi-node satellites	9,743
kernel size	27,005
median satellite size	2
average satellite size	3.69

4.4.5 Visualization Front End

This section describes two graph visualization techniques, one for small sparse multi-level clusters, the other for richly connected objects. The first centers longer object chains, the

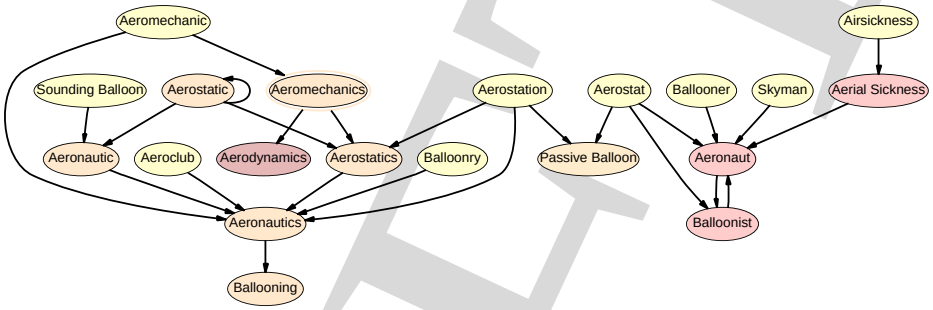

Figure 4.4: Sparse Cluster Visualization

other ranks centers objects around important incoming and outgoing arcs. Figure 4.4 illustrates how satellite clusters from the graph can be visualized. After applying the graph manipulation tools to the dictionary repository, the clusters smaller than 20 nodes in size are easily rendered using graph visualization tools such as dot from AT&T Research [ATT99]. Figure 4.1 is another representative example of the first style of visualization, while Figure 5.10 represents the latter style. Note that Figure 5.10 has many more arcs than are displayed. The actual number of outgoing and incoming arcs is marked in the central node. In order to select the best nodes to display we need a mechanism to select the most significant arcs for any given node. This is the topic of Chapter 5.

4.5 Comparison to Other Systems

This section compares the dictionary repository to two others, the first manually constructed, the second generated by phrase parsing a dictionary. We discuss where the dictionary repository approach expands on the capabilities of existing lexical repositories. We show the broadness of the repository as a complement to the preciseness of the relationships in other repositories.

WordNet has been in development since 1990, and its design has been elaborated since 1986. Its current revision, **WordNet 1.6** was released in 1998, and includes four principal data files, and a number of programs to aid in searching and displaying the data. Of the existing electronic lexical tools, WordNet is the one that most closely resembles the Webster's repository.

MindNet is not publicly available, but its scale is 159,000 head words and 713,000 relationships between head words. Its development began in 1992, and it supports 24

Name	Source	Nodes	Senses	Arcs	Time	Precision
Webster's	Stanford	96800	112897	1438532	4 mo.	99%+
OED	Stanford	326k	510k	5389k	3 wk.	98%
WordNet	Princeton	99642	173941	726445	8 yr.	98%+
MindNet	Microsoft Research		159000	713000	4 yr.	15%-95%

different relationships between terms. It appears that it suffers from problems, both in terms of accuracy and completeness of extraction.

The relationships WordNet defines between terms are more precise, as they were manually entered, however there are necessarily fewer of them, and they are far from exhaustive. Also, since the design of WordNet long preceded its implementation, artificial concepts, such as non-existent words, and artificial categorizations, such as non-conforming adjectives, were introduced when the repository was built. These constructs are a valid ad hoc approach to make the terms conform to the design, but they do not arise out of the usage of the language. WordNet carefully distinguishes between senses of a term, and separates a term into multiple entries when it may be used as different parts of speech, i.e., to *run* vs. a computer *run* vs. a *run* salmon (a run salmon is a salmon that has completed the return to its spawning grounds). The Webster's repository distinguishes senses of a term based only on usage, not on grammar. Another significant difference between the two structures is that the data in WordNet is separated by lexical categories, whereas the Webster's repository allows any relationship between terms to exist. Table 4.5 makes some simple numerical comparisons between the two systems.

Having compared the repositories numerically, it is necessary to illustrate with an example what the Webster's repository provides. Specifically, it relates terms without defining

the type of relationship, just the importance of the relationship. In Section 5.2.3 we examine some subgraphs that emerge from the repository data, based on terms relating to transportation.

4.6 Review

In this chapter we have examined the creation of a large scale repository, explored its structure and discussed the procedure used in its construction. We have shown that the techniques used in the construction are usable in different settings, and demonstrated the scalability of the operators used in the dictionary repository construction. The OED repository's construction time scaled linearly from the construction time for the Webster's repository.

Chapter 5

Dictionary Repository Algorithms

Chapter Outline

This chapter describes the ArcRank model of relationships between nodes in a directed labeled graph, such as hypertext. We present a ranking algorithm for directed arcs, and the algorithm for extraction of hierarchical relationships between words in a dictionary. Using ArcRank we create a thesaurus style tool to aid in the integration of texts and databases whose content is similar but whose terms are different. These algorithms produce repositories that complement handcrafted thesauri, by determining more complete relationships between words, although they are less specific. Exploiting hierarchies of relationships between words paves the way for broadening and related term queries in web-based repositories. In this chapter we also show how the ArcRank model of arc importance allows us to define similar paths, and compute all pairs of similar nodes. More precisely, when nodes are reachable through similar arc paths, or reach similar node sets, then they themselves are similar. Using ArcRank, we apply a simple generalization of the pattern/relationship extraction algorithm to generate sets of nodes having a *kinship* or near synonymy relationship. Extending this result, we use the algorithm to also produce the subsuming and specializing relationships that generate the kinship sets.

5.0.1 Preliminaries

The principal obstacle in integrating information from multiple sources is their semantic heterogeneity. The most easily recognized form of heterogeneity is when different terms are used to mean the same thing: lexical heterogeneity. Even more frequently terms have some

incomplete overlap of meaning, and we can recognize the cases of overlap. Even so, there is no algorithmic procedure to resolve problems of lexical heterogeneity authoritatively. However, we still desire assistance in determining semantically related terms.

The starting point for this work is the hypothesis that structural relationships between terms are relevant to their meaning. These relationships become interesting when all items in the domain of interest contain them, and are organized according to them. Dictionary definitions form a closed domain in the sense that the words used in definitions, taken as a set, should be defined elsewhere in the dictionary. This property leads to a directed labeled graph representation of the dictionary. Nodes of the graph model definitions, head words are labels for the nodes, and a word in a definition represents an arc to the node having that word as a label. Notable collections which are not closed include encyclopedias, which cover a set of terms equivalent to the dictionary nouns, and search engines, which return documents for all but stop words.

At first glance, the PageRank model of Web structure does not lend itself to direct application in non-hypertextual domains, since it relies on the link structure to compute a ranking over items. However, we have found that a related model, which we call ArcRank, is useful for extracting relationships between words in a dictionary. This model expresses the importance of a word when used in the definition of another. The attraction of using the dictionary as a structuring tool is precisely that head words are distinguished terms for the definition text. This extra information allows types of analysis that are not currently performed in traditional data mining and IR, where no term is assigned as 'head word' of a document.

Using the extraction technique discussed in Chapter 4 we generate such a graph structure and then use it to create thesaurus entries for all words defined in the structure including *stop words* such as 'the', 'a', 'of' that most systems single out to ignore. The thesaurus engine, based on our relationship ranking technique, constructs more complete repositories than manually constructed thesauri, since every term has at least one relationship defined through its dictionary definition. These relationships are less specific than those in the manually constructed thesauri. The thesaurus graph is a potentially important tool for systems integration experts. More interestingly, we also find that the algorithms that generate the thesaurus may be applied to document classification and the ranking of results of mining queries.

Given a directed graph whose nodes and arcs have been ranked according to the ArcRank

Table 5.1: PageRank

input: directed graph, *output*: scored node list

1. Make adjacency list representation of directed graph
2. Make rank array of size $|n|$ for graph nodes
3. Set (round 0) rank $p_{0s} = 1/n$ for all nodes s
4. While `rankchange` > `threshold` (round i)
5. For nodes s in $\{1 \cdots |n|\}$ (ranking step)
6. For arcs $a_{s,t}$ in s's adjacency list a_s
7. Transfer rank $p_{is}/|a_s|$ from source s to target t
8. For nodes s in $\{1 \cdots |n|\}$ (adjustment step)
9. Normalize, if needed, rank p_{is} wrt to total rank
10. Compute `rankchange` from previous iteration
11. Return final values from rank array

model, it becomes possible to consider the extraction of hierarchical relationships between terms. Indeed, the structure made explicit in the data, as in Figure 5.8, indicates that it is rich enough to find terms similar to **locomotive**, based on the terms which relate to **locomotive**. We show how the all pairs similarity algorithm allows us to group terms by kinship, according to their link structure. We show how these clusters, along with the accompanying link structures also define ancestor candidates for the clusters. The number of ancestor candidates vary according to the selection and scope of the kinship clusters.

5.1 Term Importance from Graph Structure

This section motivates the iterative measurement of an object's rank, based on its structural relationships in the object graph to which it belongs.

In this section we present the basis of our dictionary structuring techniques. Before presenting the ArcRank measure, we present the PageRank algorithm, and the variants we have used in our experiments.

5.1.1 PageRank

The PageRank algorithm forms the basis of the ranking technique described in this chapter, and is important to define before discussing the ranking of arcs. Table 5.1 shows a

pseudocode description of the algorithm.

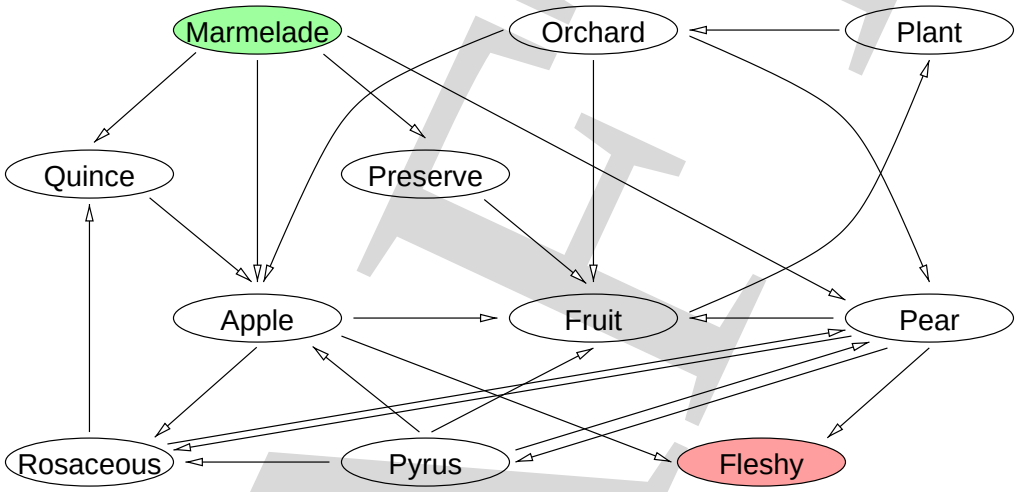

Figure 5.1: Source and Sink Nodes in Dictionary Subgraph

This algorithm is a flow algorithm over the graph's arcs. It assumes no capacity constraints limiting the amount of flow possible between two nodes. All nodes begin with an initial ranking, in our case a constant $1/|n|$, where $|n|$ is the number of nodes in the graph. At each iteration, nodes distribute their rank to their neighbors on outgoing arcs, and receive rank from neighbors on incoming arcs. The total outgoing flow from a node is never greater than its rank, $\sum_t a_{s,t} \leq p_s$, nor is any individual $a_{s,t}$ ever less than zero. The intuition behind the flow is that more richly connected areas of the graph carry larger capacity, and therefore nodes in these areas maintain a higher rank. The rank flow of nodes in strongly connected aperiodic graphs is shown to converge to a steady state [MP95]. Steady state flow is desirable, because it allows us to assert stable relationships between nodes in the graph. In practice, we accept variability in the flow between nodes, as long as the total variability over the entire graph lies below a threshold.

In general graphs, nodes and clusters of nodes with only outgoing arcs act as sources which lose all of their rank. Likewise, nodes with incoming arcs only act as sinks for the rank of their neighbors. The dictionary graph contains both source and sink nodes: source nodes represent words which are never used in other words' definitions, sink nodes are words whose definitions are not found in the dictionary. Figure 5.1 shows a representative subgraph of the dictionary repository centered around the node **Fruit**. In this subgraph, **Marmalade** is a source node, while **Fleshy** is a sink node. In the repository taken as a whole, sinks

consist of misspellings, proper nouns such as geographical and Latin species names, and scientific formulae, which we do not consider. In PageRank, the rank of sources, sinks and weakly connected clusters do not reflect their local structural differences well. The intuitive idea is that sources and weakly connected clusters lose all of their rank to the overall graph, while sinks continuously accumulate rank, giving them an increasingly disproportionate importance. In our algorithm, the final rank of a node should be defined in such a way that, when any two nodes have a distinct pattern of connections, then their rank will differ. We adapt the algorithm from Table 5.1 in one of the following three ways so that sources and weakly connected clusters preserve some rank at each iteration.

1. redistribute $b\%(b/100)$ of total graph rank before each iteration
2. limit rank transfer to a fraction $1/c$ of a node's rank
3. add a self-arc $a_{t,t}$ (node t is both source and target) to nodes
4. add a *gateway* node g and arcs $a_{g,n}$ and $a_{n,g}$ for all nodes n

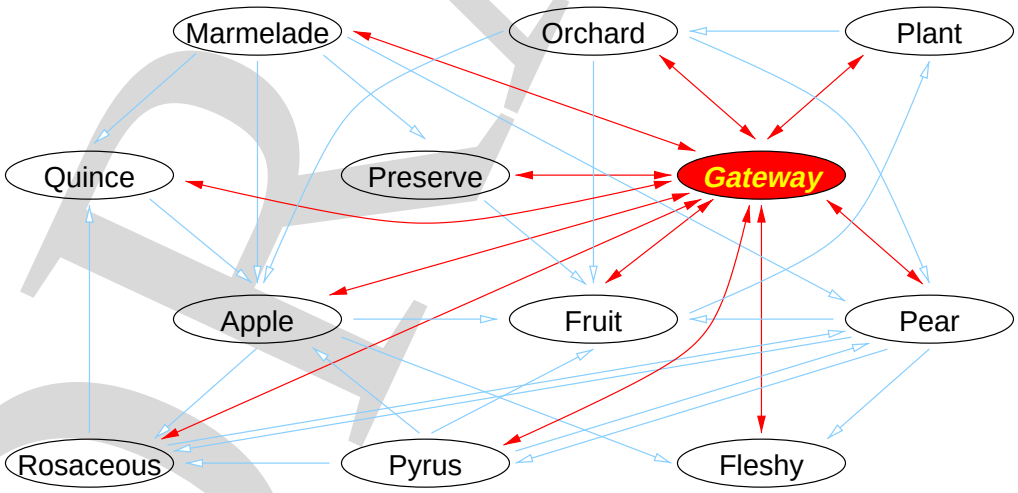

Figure 5.2: Addition of Gateway Node to Dictionary Subgraph

By selecting a non zero threshold for termination of PageRank, and one of the above adaptations, we ensure that all graph nodes preserve a non zero rank. Figure 5.2 shows a small subgraph of the repository centered around **Fruit**, with the addition of a gateway node. The arrows from the gateway node are bidirectional to indicate they consist of an

arc in each direction, as defined above. We show here that, given a node t, at iteration i with rank $p_{i\,t}$, the following holds:

Theorem 1 $\boxed{\forall t \in G, p_{i\,t} > 0}$

Proceeding by induction, we have: by definition, at the initial iteration, $p_{0\,t} = 1/n > 0$. *Assuming the property holds at iteration i, the following holds:*

$$p_{i+1\,t} = \{b/100, 1/c, p_{i\,t}/(|a_t| + 1)\} + \sum_{v \neq t} a_{v,t}$$

Since, by definition all quantities on the right hand side are positive and greater than zero, $p_{i+1\,t}$ *is greater than zero. As indicated by the equation, this property holds for each PageRank variant enumerated above.*

We see that PageRank for dictionary terms represents the transitive contribution of each term to the definitions of all of the dictionary terms. We capitalize on this property to compute the relative importance of terms with respect to each other. This measure is a feature of the arcs between nodes, or equivalently in the dictionary, the usage of terms in the definitions of others.

5.1.2 Relative Arc Importance

In the dictionary application, PageRank suffers from some inherent limitations. First of all, PageRank is inherently a node-oriented algorithm. The top ranked nodes are the common conjunctions and prepositions, which convey little conceptual meaning, and are commonly considered stop words by other applications. It is clear that on its own, PageRank is insufficient to conceptually organize the dictionary structure. We may consider an extension to PageRank which assigns to each arc the amount of rank that flows across it at each iteration. As an absolute measure, this extension is also unsatisfactory, because it favors flows between the most highly ranked terms, that is, between stop words. Besides this obvious extension, there appears to be no self-evident technique to extract an absolute arc-based measure from PageRank.

However, our original goal is to identify the most important arcs for a given individual node. By casting our ranking problem in terms of our original goal we see that rather than an absolute measure, a relative measure between nodes is preferable. For any term in the dictionary, the words that signify the most in their definition should correspond to the arcs

in the graph which are most significant in a ranking of arcs. Hence we arrive at the relative measure of arc relevance. Given an edge e, having source node s with rank p_s, target node t with rank p_t, and given $|a_s|$ outgoing arcs from s, the arc relevance r for e is defined as:

$$r_e = \frac{p_s/|a_s|}{p_t}$$

When s and t share several (m) edges $e_1 \ldots e_m$, we sum the arc ranks to compute the importance of t in the definition of s:

$$r_{s,t} = \sum_{e=1}^{m} \frac{p_s/|a_s|}{p_t}$$

$r_{s,t}$ measures the relative contribution of the rank of s to the rank of t which we show has desirable properties, such as:

Theorem 2 $\boxed{0 < r_{s,t} \leq 1}$

This follows directly from Theorem 1 and the definition of p_t, since both numerator and denominator must be positive and $p_t = \sum_v p_v/|a_v| = p_s/|a_s| + \sum_{v \neq s} p_v/|a_v| \Rightarrow p_t \geq p_s/|a_s|$.

Note that the arc importance measure is an indicator valid only in the immediate local vicinity of the end points of the arc. There is no reason to expect it to be globally commensurate. Having established an arc importance measure we are ready to present the ArcRank algorithm, and walk through a hierarchical set of relationships the algorithm uncovers.

PageRank extensions

gateway node preserves structure and guarantees convergence

This section demonstrates a technique for graph transformation that affects each object in a minimal and isomorphic fashion, and creates a single strongly connected component.

5.2 Arc Importance from PageRank

support for arc rank measure

This section shows that arc importance is a local measure of flow in the stationary distribution of a directed graph. Arc importance provides a value, that while local to the arcs' source and target nodes, respects a flow measure that is germane to the entire graph structure. Removal of low ranking arcs minimizes change to stationary distribution.

In the previous section, we have computed a relative measure of arc importance. Here we show how to rank it with respect to both the source and target nodes, to promote

arcs which are important to both endpoints. We discuss the repository we construct using ArcRank, and compare it to other systems.

5.2.1 ArcRank Algorithm overview

The ranking of an arc according to the arc importance metric defined above is typically different at the source and the target node. Indeed, it is possible for the highest arc importance value of arcs from a source node to be the lowest value for arcs coming into the target node. ArcRank, defined in Table 5.2, computes a mean of the ranked importance of arcs, so as to promote arcs which are important both to the source nodes and to the target nodes.

Table 5.2: ArcRank

input: triples (source s, target t, arc importance $v_{s,t}$)

1. given source s and target t nodes
2. at s, sort v_{s,t_j} and rank arcs $r_s(v_{s,t_j})$
3. at t, sort $v_{s_i,t}$ and rank arcs $r_t(v_{s_i,t})$
4. compute ArcRank: mean$(r_s(v_{s,t}), r_t(v_{s,t}))$
5. Rank Arcs *input*: sorted arc importance
   - sample values $\{0.9, 0.75, 0.75, 0.75, 0.6, 0.5, \ldots, 0.1\}$
   - equal values take same rank $\{1, \mathbf{2}, \mathbf{2}, \mathbf{2}, \ldots\}$
   - number ranks consecutively $\{1, 2, 2, 2, \mathbf{3}, \ldots\}$

Other rank numbering techniques resulted in skewed output. Competition style ranking, which counts equal values equally, but orders subsequent values differently, disadvantages arcs to nodes with many in-arcs. Given the same sample values from the above, the boldface value in the list here shows where this ranking differs: $\{1, 2, 2, 2, \mathbf{5}, 6, \ldots\}$. Also, computing rank as a fraction of the total number of ranks: $\{1/n, 2/n, \ldots, \mathbf{n}/\mathbf{n}\}$ favors arcs to nodes with a larger number of distinct ranks.

The ArcRank algorithm is more space intensive than PageRank, because it is arc oriented, but is fast and easily made into a disk based version. It essentially requires two passes through the data, and storage for twice the number of arcs. In the course of developing ArcRank, we derived a further extension to PageRank. The idea is to vary according to the arc importance ratio the amount of a source node's rank transfered to the targets. Tuning

this optimization properly strengthens strong relationships, weakens less important ones. The additional cost is minimal, and requires ranking arcs and summing ranks per node, before pushing value across arcs.

5.2.2 The Webster's Repository

The repository we have built [Jan99b] has a very general structure, and it is defined by usage alone. There are no preimposed limitations, based on grammatical models, as to how terms relate. There is no attempt to perform a complete parse of the sentences in the definitions. This decision is based on two concerns: the first is that dictionary definitions are often terse, non-grammatical sentences in which the parts of speech can be hard to identify. The second is that no parser exists that always correctly identifies the scope of every part of speech in a phrase. Since one goal of the structure is to use every definition word possible, without exceptions, it is not acceptable to use a parsing technique to preassign meanings to the graph arcs. Similarly, the scope of negation is not considered. As a result we often find opposites clustered together in the graph. Such clustering is appropriate, since opposites are used very similarly.

This repository is the only one that does not exclude stop words, and as a result we are able to find that stop words most strongly relate to each other. On the down side, the type of relationship expressed in the repository is not always self evident, especially since many definitions and terms are now obsolete. Also, the accuracy of the ArcRank measure increases with the amount of data, and much of the dictionary contains very sparse definitions. Due to this sparseness we often find that ArcRank will promote arcs to lower ranked targets. Also, misleadingly, the sparseness of data makes a simple metric of ranking sources by the paucity of arcs work well, when it would otherwise fail.

5.2.3 Browsing the Dictionary Repository

Because of the size of the dictionary repositories, there is an inherent tradeoff when considering visualization techniques. The current trend in information visualization is to pursue sophisticated algorithms for displaying as large a portion of the graph structure as possible. This decision is often to the detriment of the legibility of the information in the nodes. Given our assertion that ArcRank is a high quality ranking of each node's incoming and outgoing arcs, we have chosen a different approach. Instead of displaying a

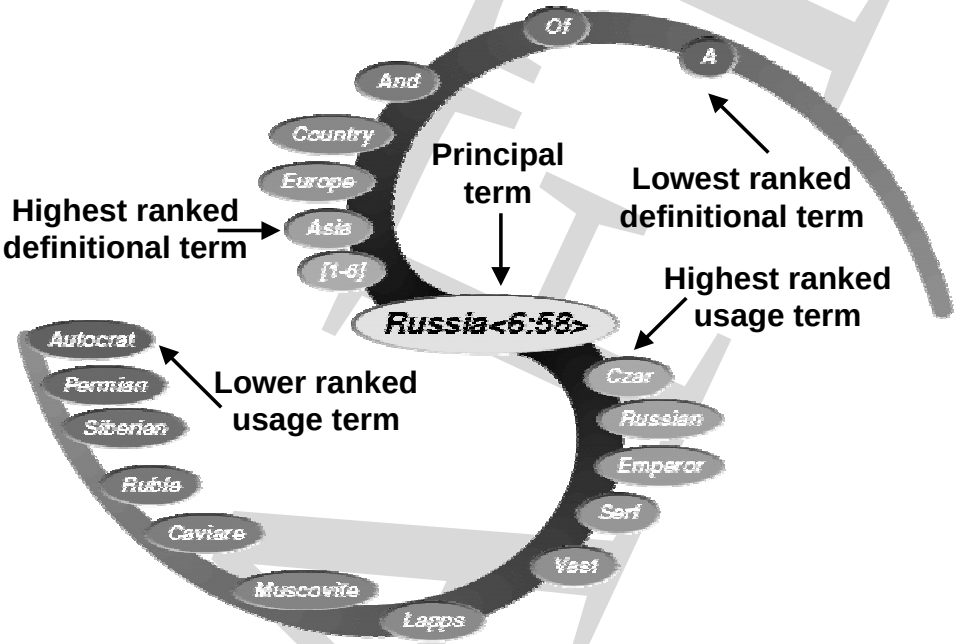

Figure 5.3: Sample Node from Web Repository Interface

hard to understand tangle of nodes and arcs, we have chosen for our browser as simple a display as possible, that still effectively communicates the notion of ranked relationships between nodes. The result of this effort is shown in Figure 5.3, and can be tested on-line at: `http://skeptic.stanford.edu/data/`

We do not show all arcs for every node, but instead display the nodes associated with the highest ranked arcs. These are arrayed over a spiral curve, such that the nodes closest to the center are most closely related to the center node, based on the ArcRank measure. In Figure 5.3, with a little effort, we can deduce that the dictionary entry for **Russia** reads 'A country of Europe and Asia'. Note that ArcRank ensures that the stop words **And**, **Of** and **A** have lower ranking than the other terms in the definition. In the set of terms that use **Russia** in their definition, the quality of the ordering is less immediately clear, but there is no question of the relatedness of these terms to the central node. In the discussion that follows, we use an interface that allows us to ascertain when a node is both in an other's definition, while using the other in its own definition (a situation that occurs more frequently than anticipated).

It is instructive to browse through the repository to get an idea of how it organizes the

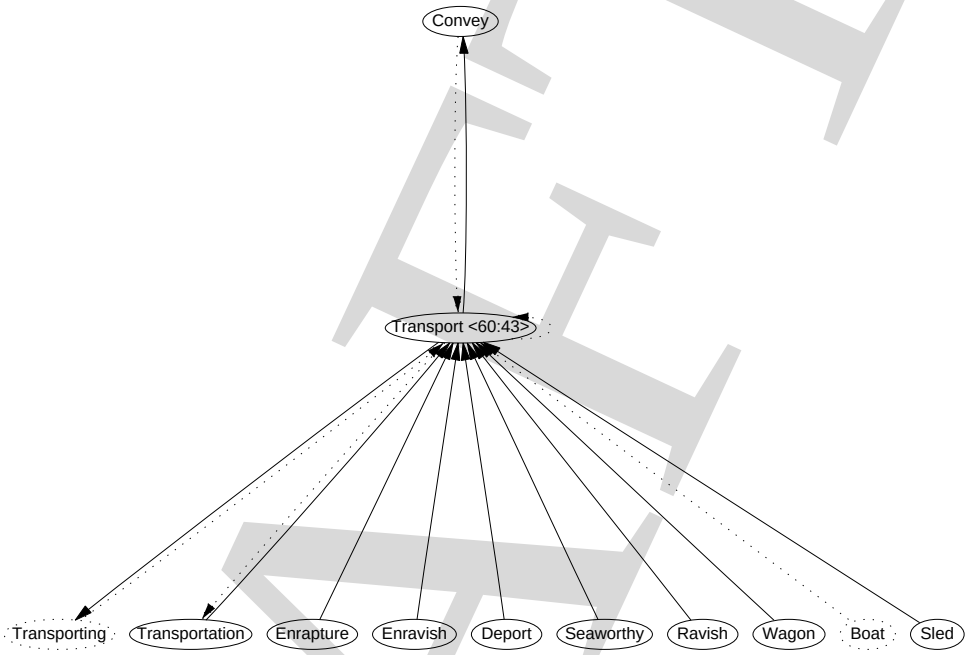

Figure 5.4: Terms Relating to Transport

dictionary terms. The example below is prompted by an interest in developing a transportation ontology to support logistics applications. We start at the term **Transport** as shown below in Figure 5.4. The general form of charts generated using the repository, such as Figure 5.8, frame a term above by the terms used in its definition and below by terms that use it in their definition. These terms are placed from left to right in order of their ArcRank measure. The ArcRank values are not displayed to maximize the clarity of the charts. No more than the two dozen most significant associated terms are displayed: the label for the central term contains a count of incoming and outgoing arcs of the form <*outgoing, incoming*>. In addition to the ArcRank measure on arcs, each term has an associated PageRank value (not displayed). Arcs and Term borders are dotted when the arc's source node has a lower PageRank than its target node.

In Figure 5.4, which has been pruned for clarity, we see that the term **Convey** is used in transport's definition. **Transport** has multiple senses, including one relating to emotional transport, which is emphasized in the set of terms that use **Transport** in their definition. When we next examine the term chart for **Convey**, Figure 5.5, we find **Transport**, along with transported, and cargo which are also significant for the logistics ontology. Other terms in the set illustrate the more general nature of **Convey** as compared to **Transport**. Further

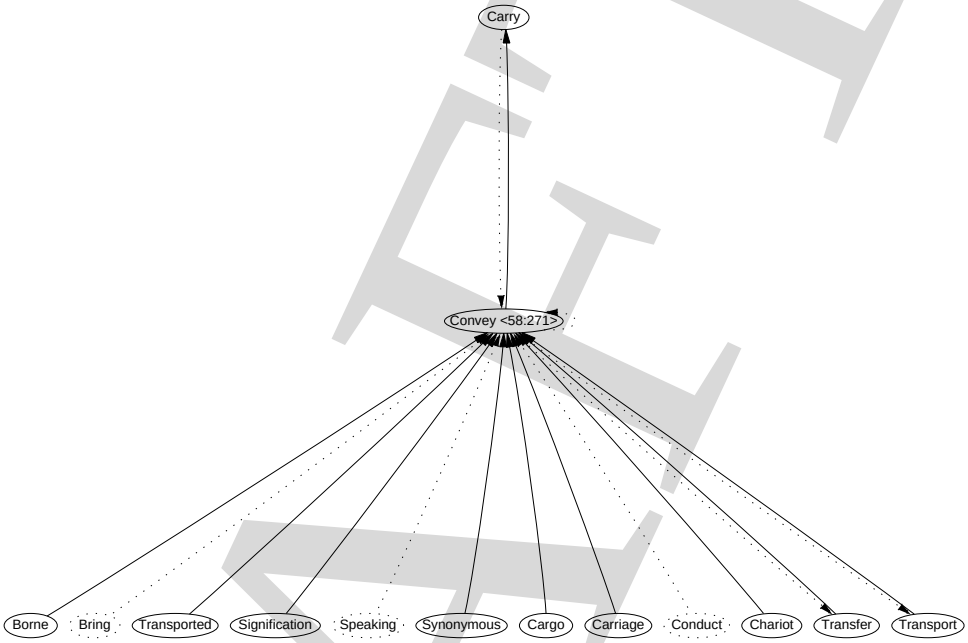

Figure 5.5: Convey Generalizes Transport

browsing upwards in the repository takes us to the chart for **Carry** in Figure 5.6. Note how **Carry** subsumes **Convey** in the sense of transport, and that the term transported is also in its set of terms. We expect too that **Hold**, given its position as a parent of **Carry** in this chart, expresses a more general notion relating to **Carry**.

Starting from **Transport** in the downwards direction, we select **Wagon** and arrive at Figure 5.7. **Wagon** is not a specialization of transport, although transport does subsume it: a wagon is one of a number of forms of transport. We see that terms such as **Car** and **Vehicle** also shown in Figure 5.7 represent a generalization relationship for **Wagon**. Also, terms such as **Charioteer**, **Caravan** and **Wheelwright** relate to wagon without being specializations. **Locomotive** is however a specialization, and we continue the browse with the chart in Figure 5.8.

The chart for **Locomotive** illustrates a spectrum of relationships between terms, some of which are altogether unexpected, such as locomotive's relationship to the term **Appendix**. A glance at the definition of locomotive in Figure 5.9 reveals that a reference to an illustration in the appendix of the dictionary appears inappropriately within the definition field of the term. This unfortunate noise in the source data appears throughout the repository, but is typically outweighed by the amount of useful relationships that emerge

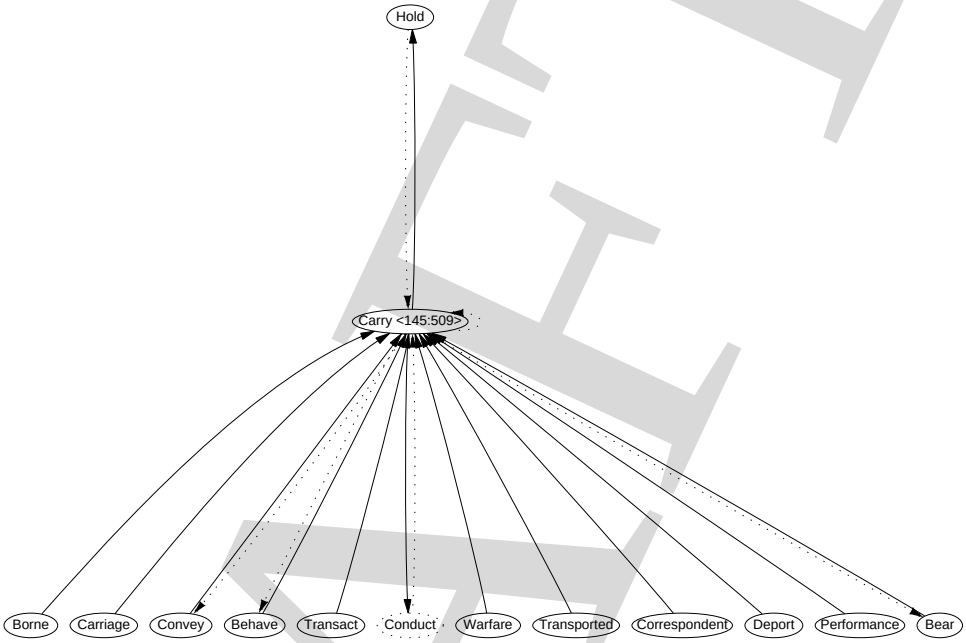

Figure 5.6: Carry Subsumes Convey

from the repository structure. In this case, for instance, the other associated terms all respect some subsuming or entailment relationship to locomotive.

A more careful examination of the definition in Figure 5.9 shows that ArcRank is able to eliminate all stopwords from achieving high ranking, as well as words such as 'especially', 'one', 'communicate', 'motion', 'bear', 'convey', 'draw', which are either not directly related to **Locomotive**, or too general. Indeed, we've seen that 'bear' and 'convey' appeared at a higher level of abstraction in Figure 5.6.

5.2.4 Further Examples

In the examples above we have seen verbs and nouns, and their organization as facilitated by the ArcRank algorithm. In this section we see that the ranking ability of the algorithm extends to adjectives, adverbs, pronouns, prepositions, and even somewhat to stop words. Recall that the unique aspect of ArcRank is that it excludes no terms in its ranking mechanism, so it is able to provide information about stopwords. No other ranking methods handle stopwords as gracefully. We also observe the algorithm handles with no degradation the addition of nodes that represent proper nouns, such as country names.

Figure 5.10 provides a strong example of the clustering of concepts that ArcRank

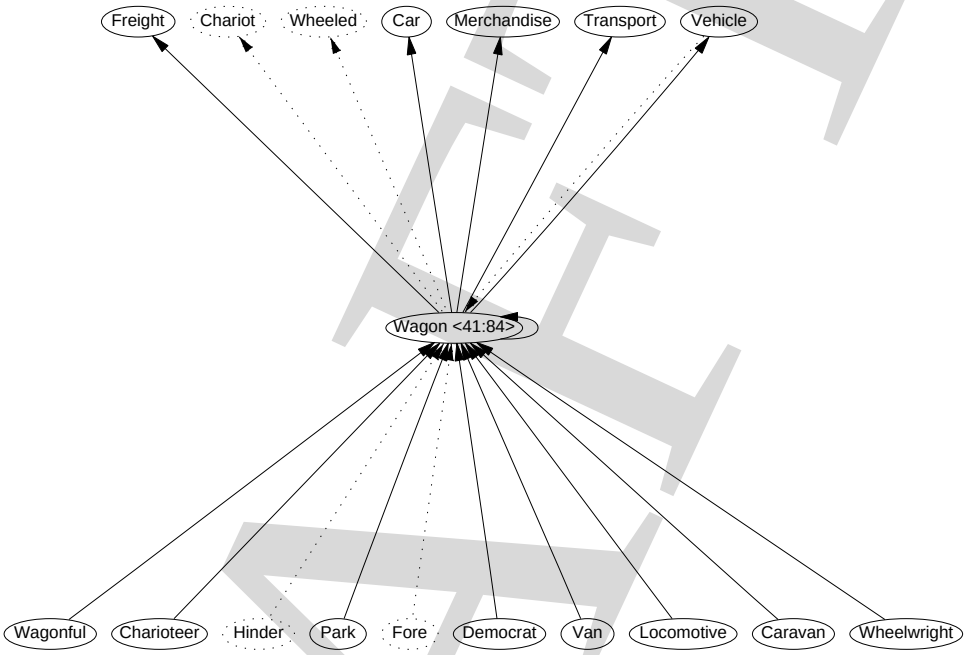

Figure 5.7: Wagon as a Means of Transport

achieves. Light as well as a multitude of terms relating to darkness figure prominently in this chart. Note that although adjectives are predominant, nouns and verbs are also present among the important relationships. Such relationships between parts of speech are typical of those that are not captured in WordNet and MindNet.

Figure 5.11 shows how time oriented adverbs cluster around the term 'ever'. Note again that as in the previous example the antonym for the term, *never*, is also present. The largest category of terms in the chart are adverbs, and two thirds of the terms are time oriented.

Figure 5.12 shows a common adverb, and presents us its two common senses, 'also' (as in me too) and 'excess' (as in too much). A class of terms prefixed with *Over-* emerges clearly. This category contains verbs, adjectives, nouns and adverbs, and could not appear in WordNet, since WordNet separates the parts of speech. The problem of disambiguating the senses of terms is left for the next chapter.

Figure 5.13 shows a borderline stopword, since the term 'it' appears in just about 5% of the dictionary terms. Note how well the ranking algorithm performs. The term *pronoun* is selected as having a top ranking arc out of 4866 possible candidates. Two classes of strongly relating terms are those that characterize 'it' as a pronoun as well as a set of other stopwords.

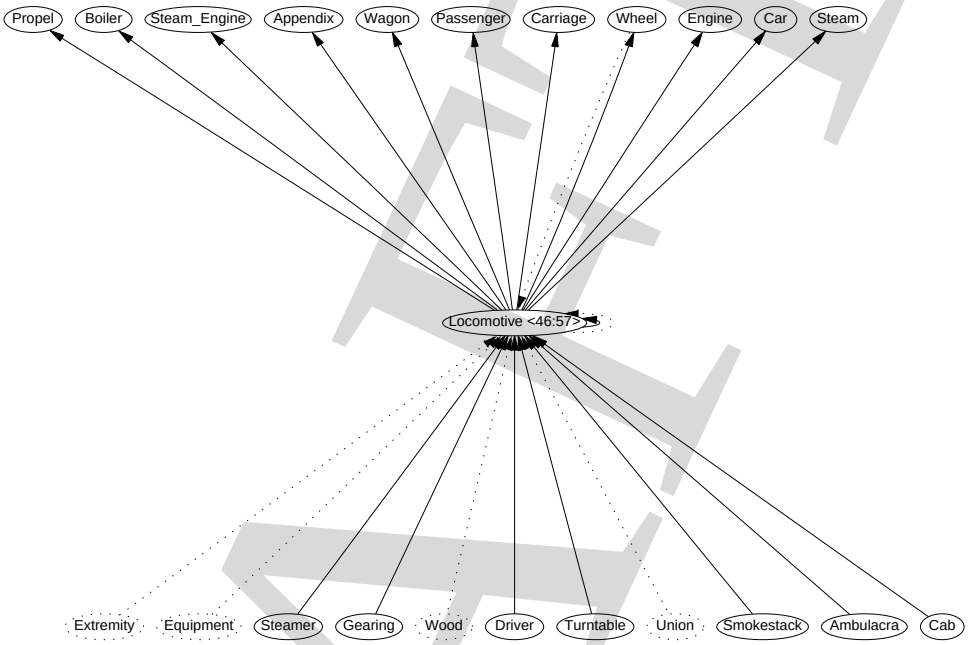

Figure 5.8: Locomotive Specializes Wagon

The stopword *to* in Figure 5.14 still manages to achieve a most interesting top ranking of nodes. The incoming arcs are all from common verbs, and from the other stopword 'The'. The notion of infinitive, and arrival are the top two outgoing arcs for the term. ArcRank produces useful results where other algorithms are unable to perform, and in categories that Wordnet does not cover.

Perhaps the most interesting of all of the categories we examine in this section are the set of special nodes created to represent words that are not defined in the dictionary. Proper nouns, for the most part, are not defined in the Webster's dictionary. However, many of these nouns have a related adjective which *is* defined. For example, Scotland $\rightarrow$ Scottish. In these cases, we add a node to our original definition chart, labeled by the proper noun, and having as a single word definition the related adjective. We use ArcRank to order the terms that use the proper noun to glean insight into the meaning of the proper noun. As we see in the case of Scotland, the terms listed provide information about Scotland. Even *logarithm* is relevant, as John Napier who developed the natural logarithms was from Scotland.

The ability to add terms to the structure suggests a number of possible extensions to the repository to handle misspellings in the definition terms. For example, when confronted with a word that does not appear as a dictionary head word, such as 'bamana', we currently

```
<hw>Lo"co*mo‘tive</hw> <pr>(?)</pr>, <pos>n.</pos>
<def>A locomotive engine; a self-propelling wheel carriage,
especially one which bears a steam boiler and one or more
steam engines which communicate motion to the wheels and thus
propel the carriage, -- used to convey goods or passengers,
or to draw wagons, railroad cars, etc.
See <xex>Illustration</xex> in Appendix.</def></p>
...
```

Figure 5.9: Partial Definition of Locomotive

ignore the term. Instead, we may take the lexically most similar word 'banana', using a cost model developed for spelling errors, to be the single word definition for 'bamana', and just add 'bamana' to the repository.

In the next section we suggest applications that build on the ArcRank algorithm. In particular we are interested in using ArcRank to guide the extraction of hierarchical relationships between terms.

5.3 ArcRank Applications

Having traveled through a very small sample of the structure of the repository, it becomes clear that the ordering provided by ArcRank is in itself not sufficient to automatically extract the significant terms relating to a given term. An algorithm to achieve this extraction is the basis for the application we are building on top of the repository, and discussed later. As it turns out the rankings provided by PageRank and ArcRank enable an efficient extraction procedure to make explicit the hierarchical structures embedded in the relationships between terms.

The ArcRank algorithm identifies the relative importance of nodes to each other. The applications that suggest themselves immediately therefore are finding similar nodes, finding nodes that subsume classes of similar nodes, and finding classes of related nodes that specialize a node. Much ongoing research is investigating the idea that we may find that nodes are related when they share similar sets of arcs. This research assumes that all arcs are equivalently important, and therefore is limited in its ability to recognize on which arcs

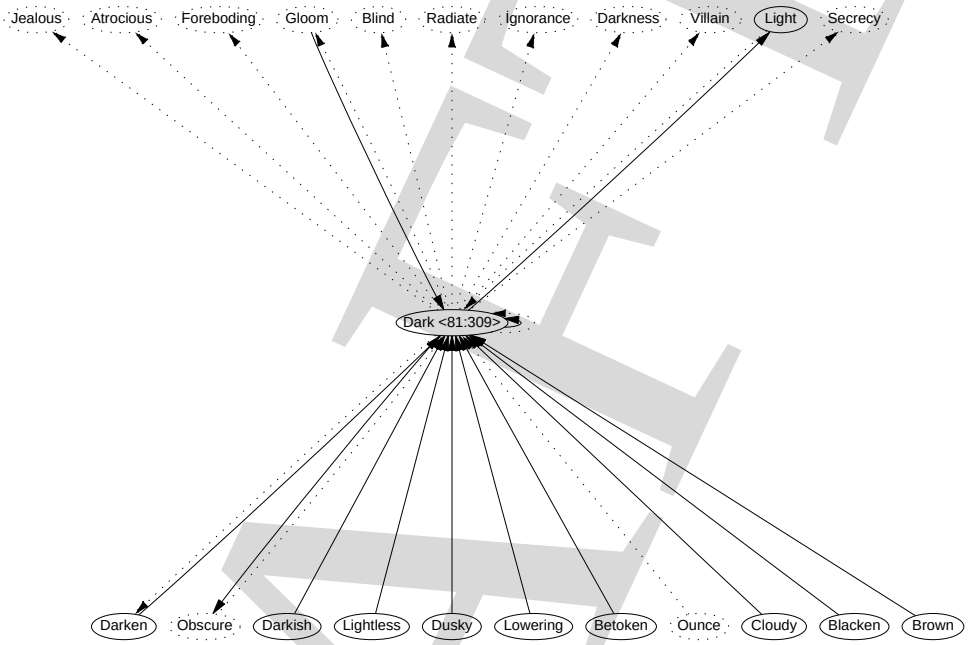

Figure 5.10: Adjective 'Dark'

to focus. ArcRank provides precisely the information we need to make the distinction between important arcs and less important arcs, to improve the quality of the extraction of related nodes.

5.3.1 Comparison to Other Measures

In this section we present a qualitative experiment comparing several different ranking methods over the dictionary repository structure. We begin with human assessments of the data quality.

We have collected survey data from volunteers who assessed twenty terms taken from the Webster's repository. For each of these principal terms (dictionary head words) the volunteers scored twenty words as to whether they were pertinent to the meaning of the principal term. The survey was conducted in such a way to minimize any chance of inadvertant bias entering into the results. Forms at `http://skeptic.stanford.edu/cgi-bin/survey.cgi` are generated on the fly, randomizing both the order in which principal terms appear as well as the order of the terms they are scored against. We use the scores as a benchmark to compare different methods of ranking the importance of words in dictionary definitions. We compare ArcRank against five different rankings measuring the importance of a word

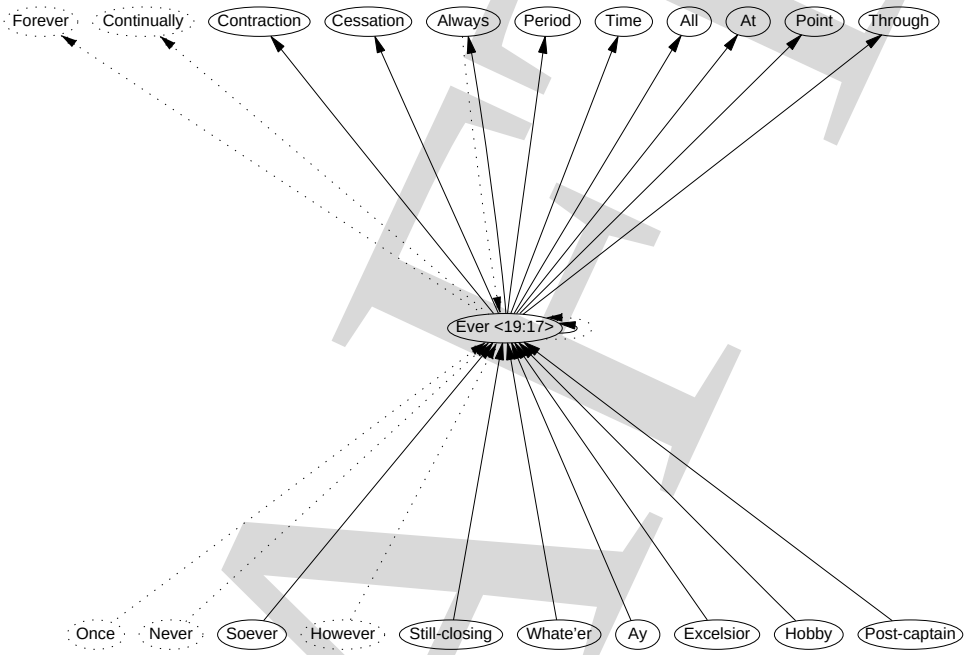

Figure 5.11: Adverb 'Ever'

relative to the principal term:

1. **PageRank** term importance based on the PageRank of the word
2. **CLEVER** term importance based on the authorities ranking of the word in the dictionary graph
3. **word order** term importance based on the order in which the word appears in the principal term's definition.
4. **frequency** term importance based on the frequency of its appearance in the definition multiplied by the inverse frequency of its appearance in the repository
5. **random** term importance based on a random permutation of the words in the definition.

In the next section we examine the use of the repository for an real multi-source interoperation experiment.

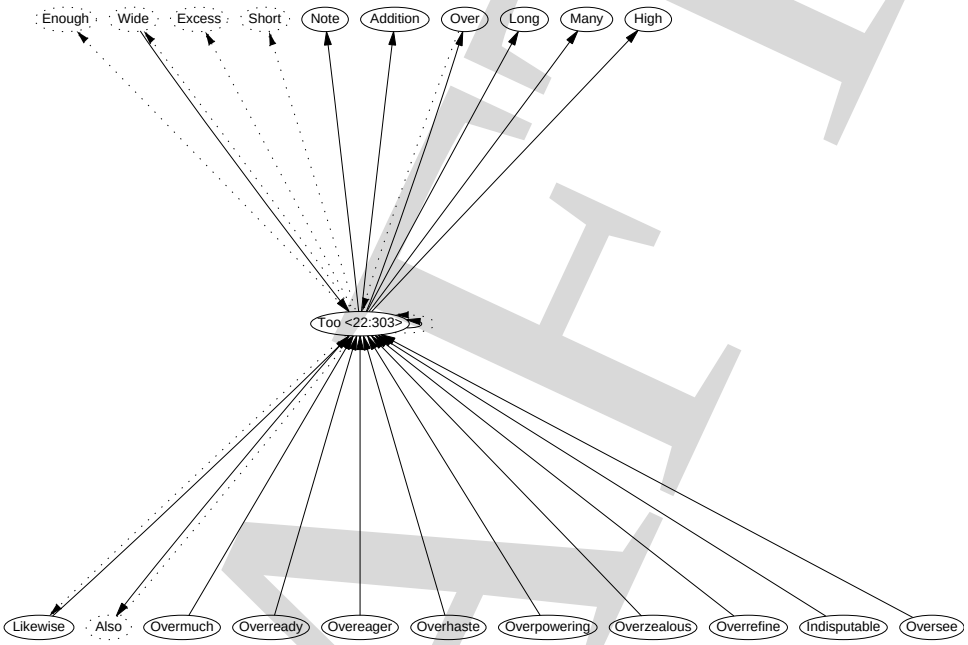

Figure 5.12: Adverb 'Too'

5.3.2 Multi Source Articulation Support

In Chapter 6 we will develop an example which shows how the repository is used to generate an initial estimate of similarity between terms in distinct information sources. We use the ArcRank values to choose the most likely candidates for similarity between terms in the two sources. The basic approach for this method is to consider the terms from each source as nodes in a bipartite graph. We immediately remove from the graph the nodes which are equivalent on both sides of the graph. The remaining nodes from each side rank the others based on similarity computed according to the algorithm in Section 5.4. We then can use a variant of the stable marriage algorithm to select the best matches between nodes from both sources.

5.4 Term Similarity from Arc Importance

In this section we present the basis for a relationship of kinship or *coherence* between dictionary terms. We use this notion to define a kinship extraction algorithm, and the all pairs similarity algorithm. We use these in Chapter 6 to support the generation of new articulations between heterogeneous sources.

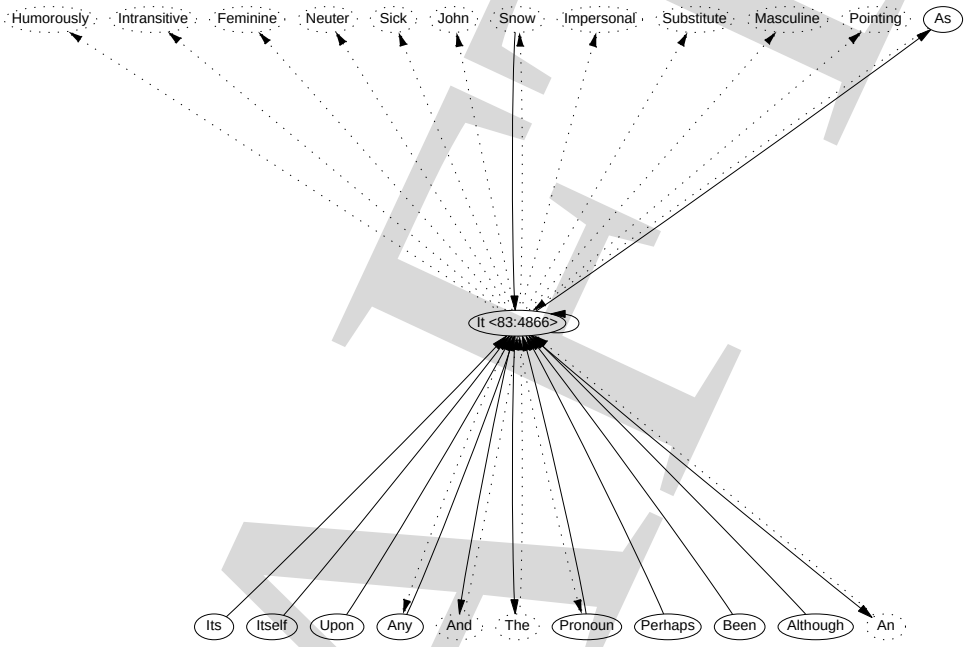

Figure 5.13: Pronoun 'It'

5.4.1 Dictionary Definition Pattern

So far, in this chapter we have capitalized on a simple relation between dictionary terms to compute a second, ArcRank, which we posit has a relevancy semantics. Given terms x and y, this amounts to the following:

- y is present in the definition of x
- y is *relevant* to the definition of x

In the next section we use this relevancy assumption to compute kinship between dictionary terms. Our guiding principle is that terms that are coherent to each other, must share relevant terms in their definitions, and must be relevant to some of the same terms.

5.4.2 Kinship Relationship Extraction

The kinship relationship extraction algorithm is the basic technique described in this chapter. This is a new algorithm based on the principles of the Pattern/Relation extraction algorithm [Bri98], which does not consider hierarchical relationships. It is novel in that it is inherently self-limiting, through the thresholding. Also, that work only considers lexical

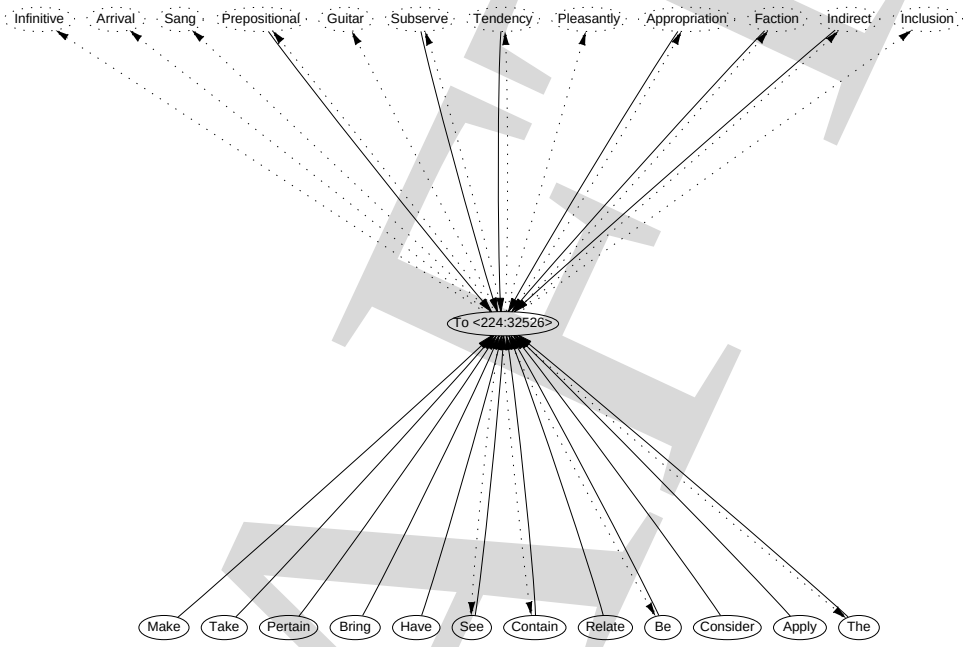

Figure 5.14: Stopword 'To'

patterns in the the formulation of the problem, whereas these results show that it is possible to use the most general structural relationships between terms as input patterns of the algorithm.

Having a repository with rank relationships between terms, it becomes possible to extract groups of related terms based on the strengths of their relationships. In particular, we are interested in extracting three relationships: *subsuming*, *specializing* and *kinship*. The kinship relationship is a similarity relationship broader than synonymy. We are able to achieve this extraction using a new iterative algorithm, based on the Pattern/Relation extraction algorithm [Bri98], as follows in Table 5.3.

Table 5.3: Extract Relation

input graph with ArcRank computed, & seed arc set, *output* local hierarchy based on seed arc set

1. Compute set of nodes that contain arcs comparable to seed arc set
2. Threshold them according to ArcRank value
3. Extend seed arc set, when nodes contain further commonality
4. If node set increased in size repeat from 1.

The output of the algorithm computes a set of terms that are related by the strength

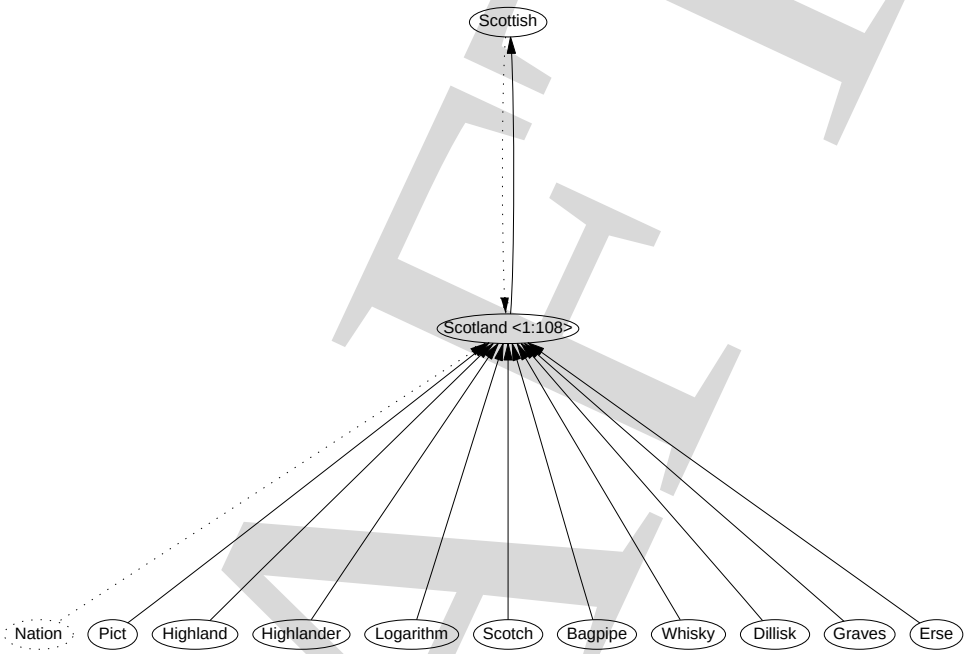

Figure 5.15: Artificial Node 'Scotland'

of the associations in the arcs that they contain. These associations correspond to local hierarchies of subsuming and specializing relationships, and the set of terms are related by a kinship relationship. The algorithm is naturally self-limiting via the thresholds.

This approach allows us to distinguish senses of terms when they engender different structures according to the algorithm. Indeed, the senses of a word such as *hard*, are distinguished by the choice of association with *tough* and *severe*. Also, ranking the different senses of a term by the strength of its associations with other terms allows us to uncover the principal senses of a term.

5.4.3 All Pairs Similarity

This section presents a new algorithm for iteratively computing the similarity of objects based on the importance ranking relationship developed in the previous sections. We show the use of similarity data to support the development of new articulations for interoperation.

We revisit in Figure 5.16 the fruit subgraph seen earlier with the view of comparing the similarity between **Apple** and **Pear**. By inspection, we see that the two terms share, in this small subgraph alone, incoming arcs from **Marmalade** and **Orchard**, as well as outgoing

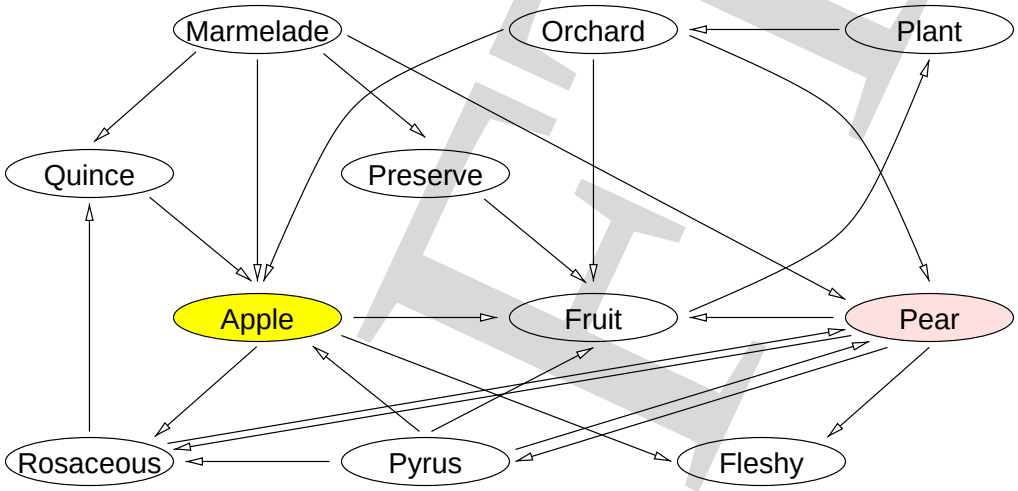

Figure 5.16: Similarity between Apple and Pear

arcs to **Fruit**, **Rosaceous** and **Fleshy**. More interestingly, since **Rosaceous** and **Pyrus** have similar arcs they have a computed similarity value. Then the paths **Apple** $\rightarrow$ **Rosaceous** $\rightarrow$ **Pear** and **Pear** $\rightarrow$ **Pyrus** $\rightarrow$ **Apple** have a similarity value associated with them, therefore contributing to the similarity of **Apple** and **Pear**. In general, every two paths of equal length, with equal end-points, where for every i, the ith items in the paths have a non-null similarity value, contribute to the similarity value of the endpoints of the paths.

5.5 Review

In this chapter we have presented the ranking algorithms which form the basis of all of our computations over the dictionary repository structure. We compute the flow over each arc in the repository, as it approaches a steady state, and compute the arcs' relative importance to be the proportion of the destination's node's PageRank that is contributed by the arc. The arc importance values are ranked for each node, and the rankings are used to compute an ArcRank value for each arc of the repository. ArcRank is the basis for the all pairs similarity algorithm between nodes of the repository, and also for the initial similarity evaluation of nodes from differing sources for which an articulation is being constructed. The all pairs similarity algorithm can be computed in $m \log n$ time where m is the number of arcs and n the number of nodes. The other algorithms are linear in the size of the number of arcs of their inputs.

Chapter 6

Algebraic Infrastructure

Chapter Outline

In this chapter we provide a more detailed description of the algebraic infrastructure for the thesis. Recall that in Chapter 2 we only presented a brief description of the object model AMO, and it associated rule language AMORL, before discussing the main contributions of the thesis. It is the case, however, that the thesis work would not be possible without the underpinnings described in this chapter. The algebraic operators described here, three of which were extensively used for the thesis work, do not have as complete a coverage as the topics of the previous chapters, but are relevant to the better comprehension of the matieral therein.

We begin with a discussion of semantic context, followed by a listing of algebraic operators considered in the context of the thesis work. We continue with a more in depth presentation of these operators,

6.0.1 Semantic Context

We motivate the need for context by observing that there is no global notion of consistency of information. Models of knowledge that are appropriate for one application may be useless for another. Identical terms in separate sources will invariably have differing semantics, while distinct terms, even within the same source, may have equivalent semantics. What we desire, is the ability to specify that the semantics of the objects relevant to an application are locally consistent, and free of mismatch.

We define, following [Guh91], contexts to be objects that encapsulate other objects.

Contexts assert the validity of statements about the objects they encapsulate. In other words, given an appropriate set of statements about its objects, a context provides guarantees about their consistency. Since we use contexts to model knowledge obtained from diverse sources for application specific uses, we are concerned with two specific relationships: *congruity* and *similarity*. The former expresses the relevance of source information to the target application, the latter identifies equivalent and mergeable objects between different sources. While the two relationships resemble each other, distinguishing the two is important for maintenance and scalability. This distinction is motivated for example in our earlier work on Ontology composition [JPVW98]. Because we assume sources are autonomous, they may change at any time. In particular, as the number of sources grows the likelihood of change at any time increases dramatically. By distinguishing congruity and similarity we are able to separate changes of a source that affect their relevancy to our application from those changes that affect their similarity to other sources with which we combine them.

In our system, contexts are an object whose value consists of a ruleset and a sequence of objects represented by the ruleset. As implied by the previous statement object values are a sequence of values, both of primitive and object types. The ruleset itself is an object, whose interpretation defines other objects. The ruleset transforms source knowledge into an object set that meets the consistency requirements of the target application. The consistency guarantee, as embodied by a congruity expression is written in the AMORL rule language defined in Chapter 2. In the next section we present the operators of the ontology algebra in more detail.

6.1 Operators

In this section we review the operators that compose the algebra. We also detail the application that motivates the inclusion of each of the operators in the algebra. We begin with unary operators, which all transform a source according to a congruity measure. The binary operators, in contrast, take a similarity measure to combine information from two sources.

6.1.1 Unary Operators

Of the unary operators, the first two maintain source information, and wrap it in a new object or set, while the two others reduce the source according to an additional predicate. These operators take a directed graph as input and return one as output. To preserve composability of the algebra, these operators take a directed graph as input and return one as output.

Summarize

The canonical unary operator of our algebra, `Summarize` or S creates an object that encapsulates the information of the source, and populates the object with results of an aggregation operation over the source information. The application that motivates the existence of the S operator is data classification. The aggregation over the source data effectively groups the source into equivalence classes.

Glossarize

The `Glossarize` or G operator creates an object that contains the set of objects from the source, without any of the substructure. G effectively flattens the source data into members of a single set. G is motivated by the need to list all terms that are subordinate to an object.

Filter

The `Filter` or F operator reduces the instance objects from the source data according to a selection predicate. At its most restrictive, this operator returns only the schema structure of the source. It is the complement to the G operator. F is important in reducing the size of a source for verification and validation purposes.

Extract

The `Extract` or E operator reduces the schema objects, as well as corresponding instances, from the source data according to a selection predicate. Wrappers that build on existing wrappers use E to present a reduced view of the original wrapper.

6.1.2 Binary Operators

As with the unary operators there are two operators that maintain the source data as they combine them, and two which reduce the sources.

Match

The prototypical binary operator, `Match` or M returns the information from both sources, along with a new object that contains a sequence of pairs of references to matching objects in both sources. Where matching objects differ in name or granularity, new objects are created as necessary to mark the transformation.

Blend

The `Blend` or B operator extends matching objects from both sources. The transformation adds the attributes to each object in a source that are present in the other. The effect of this operation is to map a copy of all the objects reachable from the matching objects in one source onto the other source. The need to extend the schema of objects with the attributes they contain in other sources motivates the addition of B to the algebra. The resulting objects represent blended subclasses of the original source objects.

Intersect

The `Intersect` operator or I returns only the portions of the sources that match. Both copies of the matching objects are returned, since the objects that mark name, granularity or reference heterogeneity in the sources, differ between the two sources. The common application for I in practice is the identification of common schema between sources. I returns objects that conform to the schema defined in both sources.

Difference

The `Difference` operator or D returns only the portions of each source which are unique to it. In effect, this corresponds to a symmetric difference between the two sources. The substructure supporting the differing objects in each source (those objects and attributes that keep the graph of the source information connected) is also preserved. In practice, D is used to determine the semantic distance between sources. The cost of transforming the differing objects is in fact the measure of this distance.

6.1.3 Summary

The eight operators above take semi-structured object graphs as parameters and return them as results. The unary operators are augmented by a congruity expression, that defines constraints on the objects transformed by the operator. Binary operators use a similarity expression to specify constraints on the objects that match between sources.

6.1.4 Semantic Consistency

Before defining the relationships below, it is necessary to define the unit of semantic consistency we use throughout. As we have seen above, there is no meaningful notion of global semantic consistency. Recall that the object model we defined above is structurally congruent to HTML. In a graph as large and heterogeneous as the World Wide Web, we must use heuristics to define semantically consistent subgraphs. For example, content based heuristics such as: pages located at the same server, having the same URL prefix, pages using relative addressing, or sharing the same style sheet, are candidates for being considered semantically consistent. With XML, document type definitions (DTDs), determine a set of consistent documents, as long as the DTDs are properly used. The types of heterogeneity within documents that we need to address are as follows:

- language differences

  term differences head of state is president vs. chancellor

  term definitions president is ceremonial vs. powerful

  term polymorphism officials are simultaneously ministers & parliament members

- structural differences

  term multiplicity government is Commons & Lords vs. Parliament

  term relation ministers are in parliament vs. cabinet

  term repetition council of state is separately government & prime minister's office

In practice, even single page documents contain inconsistencies and incorrect data. Therefore, we do not define an a priori measure of semantic consistency at any granularity. Instead, we define a congruity measure, which expresses a data source's relevance to an intended application. Only data sources which enforce the same congruity measure are considered semantically consistent with respect to an application.

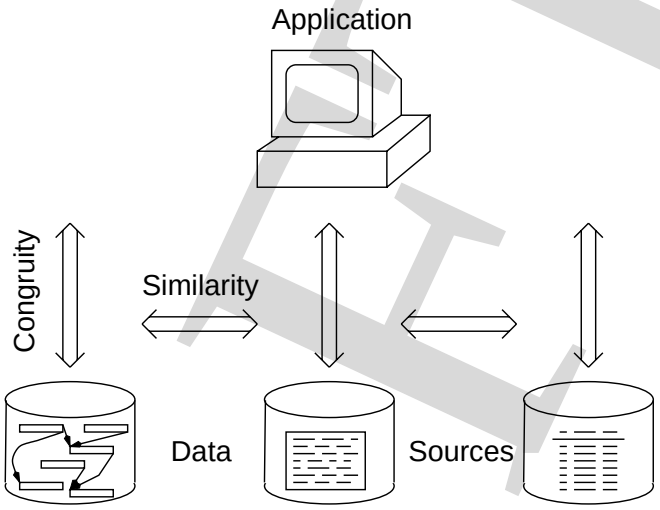

Figure 6.1: Relationships between Sources and Target Application

6.1.5 Congruity Measure

The congruity measure is a binary relationship between a data source and a target application of the data. It states explicitly the portion of the source that is relevant to the target application. Data which is only partially useful due to error or incompleteness is also expressed in the measure. In Figure 6.1 the congruity measure is represented by the arrow in the vertical dimension.

In our system we write the congruity measure as a script of the primitive operators defined in Section 2.1.4 above. The empty script, as implied above, represents no relationship between a source and the target application. A script creating a single object that encapsulates the entire source, is equivalent to accepting the entire source contents for the target. A script that establishes a congruity measure is also called *congruity expression*. In Section 6.1.7 we give a detailed example of an iterative use of the algebra to establish a congruity expression between an on-line Webster's dictionary and an application that graphs its structure.

When multiple information sources require different congruity measures to be brought together within a single application, we do not assume that they are mutually consistent. Any blending of information from differing sources requires additionally the application of a similarity metric.

6.1.6 Similarity Measure

The similarity measure is a binary relationship between two data sources. It identifies in two sources, the objects which can be considered equivalent, as well as those objects which may be merged for the purposes of the target application without being identical. The horizontal arrow in Figure 6.1 corresponds to the direction of the similarity relationship in our system.

The similarity measure is also represented by a script, or *similarity expression* of the primitive operators in our system. When there are no matches between sources the measure is the empty script. Otherwise, scripts for a similarity measure consist of an object containing a set of references for each matching pair of objects in the sources. We show the use of the `Match` operator to create and refine a similarity measure between two government web sites.

6.1.7 Wrapper Semantics

The `Summarize` operator is a unary operator that transforms source data based on a predicate which corresponds to a congruity measure. As described in Section 6.1.5 the congruity measure is not an ideal congruity relationship, but rather an approximation of the ideal. We need a congruity relationship to apply the `Summarize` operator, but we can not establish such a relationship prior to applying the operator. This paradox is what compels us to consider approximations of the relationshipi. Indeed, we use approximations to bootstrap the construction of better approximations. The algorithm establishing a congruity measure must therefore be iterative and begin with an initial congruity measure. This initial congruity measure represents a first order approximation of the congruity relationship. At each iteration, an analysis of the outcome of the previous `Summarize` operation allows a refinement of the congruity measure to account for newly recognized exceptions and misclassified data. The iterative refinement continues, until the congruity measure approximates the ideal, within the tolerance level of the application that requires the data. Each application of the operator keeps statistics on the performance of the congruity measure. If an application of the operator produces an inferior result, an exception occurs. If the result of a prior application of `Summarize`, or `S` for short, is acceptable it will be used. Otherwise, a follow-up round of refinement begins. In this way, changes to sources which affect the performance of `S` are detected and signaled. Note that the procedure for maintaining the

congruity measure is identical to the initial creation and refinement of the measure.

6.1.8 Applying the `Summarize` (S) Operator

The S operator applied to a data source with a congruity measure generates an object which reports the result of an aggregation operation over the source objects as transformed by the congruity measure. For example, $\mathsf{S}_{max(10,len(hw))}$`(dictionary)` returns the ten longest head words of the dictionary. Applying this operation on the actual data revealed terms with missing end tags in the data. Once the errors were identified, the rules to convert terms with missing end tags are added to the definition of the set "hw" above. Using S we were also able to determine that other tags were equally valuable as head words, that we needed to remove accentuation from foreign words, and discover spelling errors in the head words by analyzing the frequency of words found in definitions, but not as head words.

6.1.9 Maintaining the Wrapper

In the course of developing the wrapper to the Webster's dictionary, a major revision of the source data occurred, affecting 10%-25% of all of the entries. These changes are part of an ongoing effort to correct and extend the dictionary, and they included corrections in the tagging of the entries, spelling corrections, reformatting of the text, addition of notes and comments, etc. By maintaining statistics with the S operator on the process of extracting the relevant parts of the dictionary, we were able to note which operations were no longer needed because the exception they handled had been updated. A comparison of the terms that we could not classify in the old and updated sources revealed new errors that had been introduced in the data. As it turns out there was relatively little within the wrapper that required correction when the source changed. Having the congruity measure within the algebraic framework significantly simplified the process of identifying and handling the changes.

6.1.10 Wrapper Mediation

Systems such as Strudel [FFLS98] provide the ability to restructure consistent pages within a web site. Ultimately, we would also like to structure pages with similar content across web sites, in order to more easily browse the pages of interest. As an example of sites we would like to browse in a similar fashion, we chose the government web sites of NATO members

and their partners. Figure 6.2 below represents a partial set of pages and links from the Finnish government's web site. The specially shaped nodes serve as visual aids in relating this graph with Figure 6.3. Similarly shaped nodes in the two figures are considered similar by the NATO website.

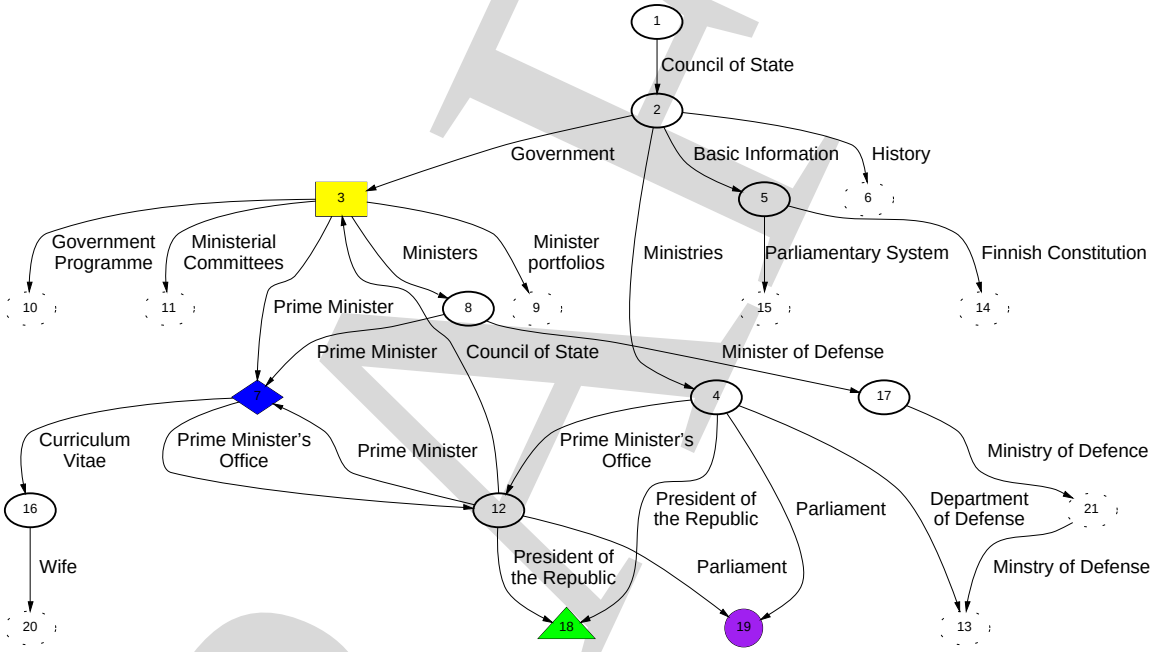

Figure 6.2: Partial Graph of Finnish Government Web Site

6.1.11 The Match (M) Operator

The Match operator takes two graphs and finds objects that correspond to each other based on a similarity measure that indicates how the correspondence is to be determined. We will highlight the matching using two object graphs obtained from the websites of NATO countries using the S operator defined in Section 6.1.7. Websites can be thought of as structured as a graph with a root page which has links to other related pages. The labelled graph structure of each website is constructed where each page is a node and all the links found on the page are modeled as outgoing arcs. Each arc is labelled with the text found along with the link, which describes the contents of the pages that the anchor point to. Each node in the graph corresponds to a Web page and is assigned a term, based upon the tags on its incoming arcs. The matching of the nodes is based on the similarity of labels,

where similarity is determined based upon the similarity measure. Typical mismatches that exist in such object graphs are as follows:

Structural Mismatches these types of mismatches occur when the same term in one source matches multiple terms in another and causes one node in a graph to match with many in the other. For example, the we see that the Prime Minister of the U.K. in Figure 6.3 is simultaneously Lord of the Treasury. The Finnish Prime Minister is not Treasury Minister, which is the closest counterpart to Lord of the Treasury. The node for the British Prime Minister should thus match two nodes in the Finnish graph.

Instance Mismatches these mismatches occur because in one source an instance of a class is not an instance of the same class in the second source. The closest case of such mismatch in our figures is that Parliament in Figure 6.3 is a parent to both the House of Commons and the House of Lords, while Parliament refers to a single body in Figure 6.2.

6.1.12 Rule Based Semantic Mismatch Resolution

In the next subsections we describe heuristics which we apply to the establishment of a similarity measure between the government Web page graphs of Finland and the U.K.

Context Identifier Tagging

Our application may require that we differentiate Parliament in the U.K. and Finland governments, and associate the Finnish Parliament with the British House of Commons. We must distinguish the semantics of the same term used in two different graphs. The basic technique allows a preprocessing of terms so as to distinguish them later when performing a lexical comparison of terms. For instance, we may edit the Finnish 'Parliament' to *Parliamentary Body*. We can also consider a more general-purpose heuristic which indicates "if TermX is a descendant of Government and TermY is not, then do not match TermX to TermY". Such a heuristic generalizes a structural observation from our graphs.

Context Identifier Removal

Matching is performed based on a similarity criterion. In our running example, we want to match the government nodes of our two nation's graphs, e.g., we want to match the node

labelled **Her Majesty's Government** in Figure 6.3 with that labelled **Government** in Figure 6.2. Therefore, a set of edit operations can be supplied that strip the labels such as prefixes such as **Her Majesty's** and thereby enable the matching.

Term Mismatch Resolution

These operations simply express that two terms are semantically related and should match. Examples from our two figures include rules such as: `(Match Monarchy President)` and `(Match Defence Defense)`. The first one indicates that we intend to match the head of states though they might be named differently.

These can be more complex than such simple lookup rules if we have a theorem prover at our service, e.g., the two labels 'The UK Parliamentary System' and 'Finnish Parliamentary System' can be matched using a more complex rule like:

```
(Instance-Of Country UK)
(Instance-Of Country Finland)
(<=3D (Match ?Country1 ?Country2)
        (and (Instance-Of Country ?Country1)
                (Instance-Of Country ?Country2)))
```

Our system should generate the tuple `(Match UK Finland)`. The fact that U.K. and Finland are two countries can either be explicitly specified or can be obtained from standard knowledge-libraries. The fact that we are interested in matching countries is indicated by the last statement.

The second type of operations are needed for differences in spelling in British versus American English, for example. The second type of mismatch will in certain cases be resolved in the preprocessing stage wherein we indicate root-words of words, e.g., `(Root-Word Parliamentary Parliament)`. The preprocessor substitutes the word for its root-word before proceeding with the matching. In the absence of any of such case specific rules, we may proceed with algorithms like Porter's stemming algorithm. As with any automatic process, stemming may result in some spurious matches or cause some matches to fail, whereas the rules let one dictate exactly what we want to match and what not to match.

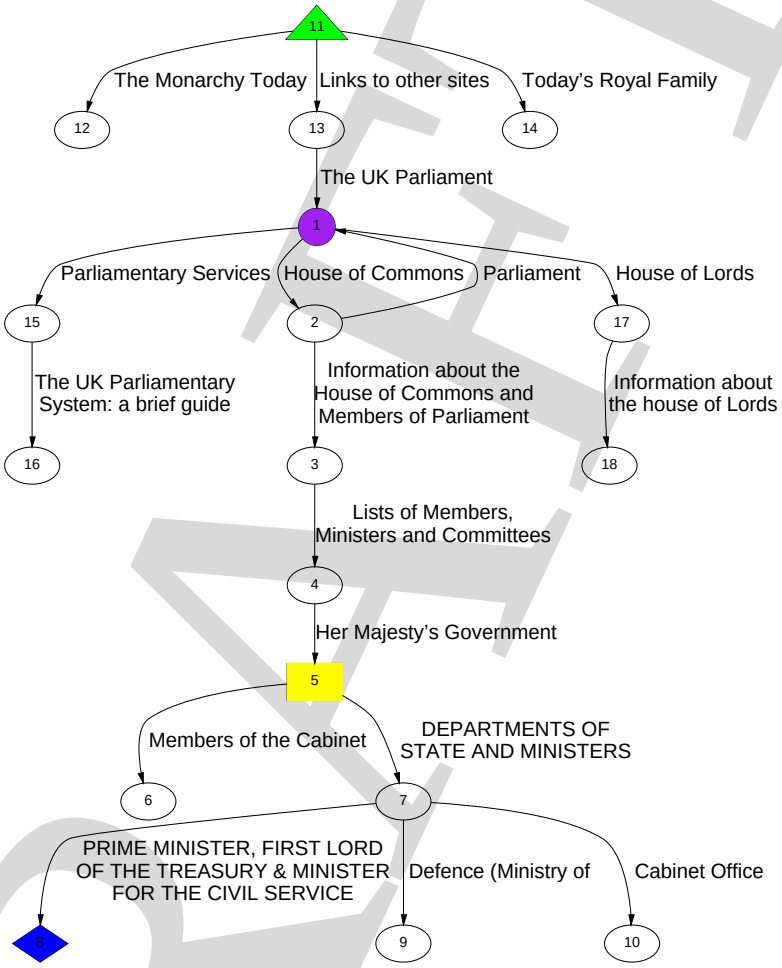

Figure 6.3: Partial Graph of U.K. Government Web Site

Irrelevant Match Resolution

A preprocessing stage removes stop-words like 'of' and 'the', which can cause spurious matches, either by looking up a table which the user can supply or by using an IR metric which assigns weights to words based on their occurrences across nodes and in a particular nodes. Words that are very frequently used across nodes are assigned very low weighting. The matching process then computes a weight which indicates the degree of match and these low weighting words do not contribute to the match and the process is as good as having deleted them. An alternate source of spurious matches is the use of stemming algorithms. Words such as minister and ministry might have be stemmed to minister and therefore result in a match, despite our wanting to preserve the difference between the two. Sanity

checking heuristics, which state explicit mismatches `(Mismatch Minister Ministry)` or a more general `(Instance-of Person X) and (Instance-of Office Y) => (Mismatch X Y)` help us in preserving the semantics we desire.

Chapter 7

Conclusions and Future Work

7.1 Novel Algebraic Architecture

This thesis defines a set of operators for semantic interoperation of heterogeneous information sources and shows how they are applied to develop articulations rapidly. These articulations serve as the mediating glue between existing information and the new applications that need the information. The articulation algebra builds on a simple object model which is general enough to easily represent Web pages, as well as database relations, and plain text. It also is sufficiently flexible to model more complex object models with strong typing and inheritance. A rule language enables the the use of objects as proxies for objects and values taken from individual sources as well as the matching of objects from disparate sources. The unique feature of the articulation algebra is that it incorporates a form of closure to model realistically the process of mediator creation, refinement and maintenance. By representing this process as a sequence of iterative steps associated with a qualitative measure, it is possible to make specific claims about the value of the information obtained using the algebra. Also, it represents mediator maintenance within the same framework. Any changes of information in the sources, expert qualifications or application requirements are fed back into the algebraic operators as a further iterative step. This iteration may induce further rounds of refinement before an amended mediator is complete. In the following paragraphs we review how the results of each thesis chapter fold into the overall conclusions of the thesis.

7.1.1 Application of Algebraic Framework

Chapter 2 introduced the object model and rule language that form the two fundamental layers underlying the articulation algebra.

In Chapter 3, we have presented the operators of an algebra for semantic interoperation of semistructured data. These operators account for the requirements of target applications through a congruity measure, and handle semantic heterogeneity of information sources using a similarity measure. These measures approximate the semantic relationships between source and target application on the one hand, and between differing sources on the other.

Our examples show the use of the algebra on real world problems. We describe the close relationship between the algebra and the process of creating, refining and maintaining wrappers and mediators for our applications. Our experience shows that building wrappers within the framework of the algebra substantially simplifies their creation, as well as improving their maintainability. The development of the algebra remains a work in progress. It is a first step in a systematic treatment of semantic heterogeneity within the framework of an algebra over semistructured data.

7.1.2 Application of Dictionary Repository

In Chapter 4 we have presented a case study demonstrating advantages of the algebraic approach to ontology management. An on-line Webster's dictionary represents an ideal test bed for the use of our ontology algebra on real world problems. We used a `Summarize` operation to define and refine a context that prepares the dictionary data for use by a thesaurus service. We showed how the consistency guarantees established for the context were for the most part preserved in the face of substantial changes to the source data. We have looked at the future directions of research on the dictionary data and its relevance as a tool to support the ontology algebra.

In Chapter 5 we have shown an important application over an information source that relies heavily on the articulation algebra. We showed how we derive a ranking algorithm that assigns an importance to the arcs of a large directed graph, based on a prior ranking of the nodes of the graph. We proceeded to determine subsuming and kinship relationships between nodes, using an algorithm which capitalizes on these ranks. These techniques represent an important step in extracting hierarchical structures from graph structured information in a way that respects the internal structure of the data.

7.2 Relevant and Future Work

This thesis work has charted a clearer representation of the process of developing semantic mediation services for the interoperation of disparate information sources. It has opened a number of avenues of further research in the following areas:

1. representing mediation results together with a confidence interval
2. casting the expert and information sources in the context of game theory
3. mining market baskets having a label in the domain of the basket items

The subsections below consider these areas in turn.

7.2.1 Anytime Algorithms

As seen in Chapter 3 we use a termination criterion to define a minimal quantitative standard for the quality of an articulation. We take a random sample from the instances accessible from the sources through the articulation to determine whether the articulation meets the minimal standard. Therefore, the termination criterion also serves as a confidence measure in the quality of the data.

If we terminate the operator closure at any time, we may collect a few random samples to define a confidence interval, which tells us how reliable our articulation is likely to be over the entire accessible data set. When our applications handle uncertainty in information, we have a new way of using the articulation algebra. Indeed, the operators take on the characteristics of an anytime algorithm, since no matter when we stop the iteration, we are able to associate a confidence interval with the resulting data. Anytime algorithms are a class of algorithms that are increasingly being considered for real-time and on-line applications.

7.2.2 Game Theory

The iterative process of mediator development is strongly related to a two player game. The information source represents an adversary to the expert, who must efficiently extract the relevant information from the source. The information source passively resists the expert's efforts. Each iteration in the operator closure represents a turn in the game. The game ends in victory for the expert when the termination criterion is reached. If a fixpoint is reached

before the termination criterion, the game ends in victory for the information source. The connection between game theory and the iterative closure employed by the operators of the algebra is one that could lead to a better understanding of the convergence and completion of the operators' execution.

7.2.3 Meta-Data Mining

In traditional market basket analysis, the basket is an aggregation that is separate from the domain of the items in the basket. It could be of interest to label each basket with the name of the most important item in the basket. Of course, we do not have access to the information which tells us what the important item in the basket is, so we must hypothesize what it could be. On the contrary, in the dictionary domain, each definition represents a basket, and the basket is labeled with the term represented by the definition. This characterization gives us two new data mining problems of interest. What new types of data mining algorithms emerge from the labeled basket problem? How does one label an unlabeled market basket problem? We have completed some preliminary work on implication rules and clustering rules that investigate the first question. We have an algorithm that extends ArcRank by generating links between the basket domain and the item domain to explore the second question.

7.2.4 Final Thoughts

The process of opening up new ground in a field invariably opens up many more problems to attack than were originally answered. We feel that this is the best sign of fruitful results in our work. We finish with a saying: you really don't know what you want until you have it.

Appendix A

Converting Other Object Representations to Our Model

Chapter Outline

Outline

A.1 Introduction

A.1.1 OEM

A.1.2 XML

A.1.3 Frames

A.1.4 UML

Appendix B

Webster's Dictionary Congruity Expression

Chapter Outline

Outline

B.1 Introduction

Script ...

Bibliography

[AGM+97] S. Abiteboul, R. Goldman, J. McHugh, V. Vassalos, and Y. Zhuge. Views for semistructured data. In *Proceedings of the Workshop on Management of Semi-Structured Data*, pages 83–90. NSF, 1997.

[AH87] S. Abiteboul and R. Hull. IFO: A formal semantic database model. *ACM Transactions on Database Systems*, 12(4):525–565, December 1987.

[AM97] P. Atzeni and G. Mecca. Cut and paste. In *Proceedings, 17th ACM Symposium on Principles of Database Systems*, pages 144–153, June 1997.

[Ams80] R. Amsler. *The Structure of the Mirriam Webster Pocket Dictionary.* PhD thesis, University of Texas, Austin, 1980.

[ATT99] ATT Research. Graphviz. http://www.research.att.com/sw/tools/graphviz/, 1999.

[BCV99] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.

[BDHS96] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *ACM SIGMOD '96*, pages 505–516. ACM, 1996.

[BJBB+97] R. J. Bayardo Jr., W. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. Infosleuth: Agent-Based semantic integration of information in open and dynamic environments. In *ACM SIGMOD '97*, pages 195–206. ACM, 1997.

[Bri98] S. Brin. Extracting patterns and relations from the world wide web. In *Proceedings of International Workshop on the Web and Databases WEBDB*, 1998. http://www-db.stanford.edu/~sergey/booklist.html.

[Car90] L. Carroll. *Alice's adventures in wonderland / More annotated Alice*. Random House, New York, NY, 1990.

[CDSS98] S. Cluet, C. Delobel, J. Simeon, and K. Smaga. Your mediators need data conversion. In *ACM SIGMOD '98*, pages 177–188. ACM, 1998.

[CF99] W. Cohen and W. Fan. Learning page independent heuristics for extracting data from web pages. In *Proceedings, 8^{th} International World Wide Web Conference WWW8*, May 1999.

[CG97] S. Chawathe and H. Garcia-Molina. Meaningful change detection in structured data. In *ACM SIGMOD '97*, pages 26–37. ACM, 1997.

[Cha56] H. L. Chace. *Anguish Languish*. Prentice Hall, Englewood Cliffs, NJ, 1956.

[CHR97] H. Chalupsky, E. Hovy, and T. Russ. NCITS.TC.T2 ANSI ad hoc group on ontology. Talk on Ontology Alignment, November 1997.

[CJN+00] A. Crespo, J. Jannink, E. Neuhold, M. Rys, and R. Studer. A survey of semi-automatic extraction and transformation. Technical report, Stanford University, 2000.

[CL 98] CL Research. Dictionary parsing project. http://www.clres.com/dpp.html, 1998.

[Coh98] W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *ACM SIGMOD '98*, pages 201–212. ACM, 1998.

[COS98] K. E. Campbell, D. E. Oliver, and E. H. Shortliffe. The unified medical language system: toward a collaborative approach for solving terminologic problems. *Journal of the American Medical Informatics Association*, 5(1):12–16, 1998.

[Dai86] D. P. Dailey. On the search for semantic primitives. *Computational Linguistics*, 12(4):306–307, 1986.

[DDL+90] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407, 1990.

[DFF+98] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A query language for XML. http://www.research.att.com/~mff/xml/w3c-note.html, August 1998. Working Draft.

[DH99] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. In *Proceedings, 8^{th} International World Wide Web Conference WWW8*, May 1999.

[DHR+98] R. H. Dolin, S. M. Huff, R. A. Rocha, K. A. Spackman, and K. E. Campbell. Evaluation of a "lexically assign, logically refine" strategy for semi-automated integration of overlapping terminologies. *Journal of the American Medical Informatics Association*, 5(2):203–213, 1998.

[DJ00] S. Decker and J. Jannink. Toward formal ontology algebras: First steps. Technical report, Stanford University, 2000.

[Enc99] Encyclopedia Britannica. Encyclopedia britannica online. http://www.eb.com/, 1999.

[FFLS98] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. Reasoning about web-site structure. In *AAAI Workshop on AI and Information Integration*, pages 505–516. AAAI, July 1998.

[Fou73] M. Foucault. *Ceci n'est pas une pipe*. Editions fata morgana, Montpellier, France, 1973. trans: This is not a Pipe, Harkness, J., U. C. Press, 1983.

[Fow98] M. Fowler. *UML Distilled*. Addison-Wesley, Reading, Massachusetts, 1998.

[GCCM96] R. Goldman, S. Chawathe, A. Crespo, and J. McHugh. A standard textual interchange format for the object exchange model. http://www-db.stanford.edu/pub/papers/oemsyntax.ps, 1996.

[Ger96] A. Geraci. The computer dictionary project: An update. *Computer*, 29(7):95, July 1996.

[GKD97] M. R. Genesereth, A. M. Keller, and O. M. Duschka. Infomaster: An information integration system. In *ACM SIGMOD '97*, pages 539–542. ACM, 1997.

[GN88] M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Palo Alto CA, 1988.

[GSVG98] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. Proximity searching in databases. In *Proceedings of the 24th VLDB Conference*, pages 26–37. VLDB, 1998.

[Guh91] R. V. Guha. *Contexts: A Formalization and Some Applications*. PhD thesis, Stanford University, 1991.

[GW99] T. Guan and K.-F. Wong. KPS: a web information mining algorithm. In *Proceedings, 8^{th} International World Wide Web Conference WWW8*, May 1999.

[HM93] J. Hammer and D. McLeod. An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems. *International Journal of Intelligent and Cooperative Information Systems*, 2(1):51–83, March 1993.

[Hul97] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. http://www-db.research.bel-labs.com/user/hull/pods97-tutorial.html, 1997. tutorial slides & references.

[Jan99a] J. Jannink. Thesaurus entry extraction from an on-line dictionary. In *Proceedings, Second International Conference on Information Fusion*. ISIF, 1999. http://www-db.stanford.edu/SKC/papers/thesau.ps.

[Jan99b] J. Jannink. Webster's dictionary repository. http://skeptic.stanford.edu/data/, 1999.

[Jan00] J. Jannink. Extracting semantic relationships from graph structured repositories. Technical report, Stanford University, 2000.

[JMN+99] J. Jannink, P. Mitra, E. Neuhold, S. Pichai, R. Studer, and G. Wiederhold. An algebra for semantic interoperation of semistructured data (extended version). In *IEEE Knowledge and Data Engineering Exchange Workshop, KDEX '99*. IEEE, November 1999.

[JPVW98] J. Jannink, S. Pichai, D. Verheijen, and G. Wiederhold. Encapsulation and composition of ontologies. In *AAAI Workshop on AI and Information Integration*. AAAI, 1998. http://www-db.stanford.edu/SKC/publications/jpvw_encaps.ps.

[JW99] J. Jannink and G. Wiederhold. Ontology maintenance with an algebraic methodology: a case study. In *Proceedings, AAAI Workshop on Ontology Management*. AAAI, 1999. http://www-db.stanford.edu/SKC/papers/summar.ps.

[Kar92] P. D. Karp. The design space of frame knowledge representation systems. Technical report, SRI International Artificial Intelligence Center, 1992.

[Kim81] S. Kim. *Inversions*. BYTE Books, Peterborough, NH, 1981.

[Kle98] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998. http://simon.cs.cornell.edu/home/kleinber/auth.ps.

[KS96] V. Kashyap and A. Sheth. Semantic and schematic similarities between database objects: a context-based approach. *VLDB Journal*, 5(4):276–304, October 1996.

[LRO96] A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd VLDB Conference*, pages 251–262. VLDB, 1996.

[MB95] G. Mecca and A. J. Bonner. Sequences, datalog and transducers. In *Proceedings, 15th ACM Symposium on Principles of Database Systems*, pages 23–35, May 1995.

[MBF+90] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Five papers on WordNet. Technical Report 43, Cognitive Science Laboratory, Princeton University, 1990.

[MIC96] MICRA inc. Webster's dictionary, 1913. ftp://ftp.uga.edu/pub/misc/webster/, 1996. also available from Project Gutenberg.

[Mir99] Mirriam-Webster. WWWebster dictionary. http://www.m-w.com/, 1999.

[MKW00] P. Mitra, M. Kersten, and G. Wiederhold. A graph-oriented model for articulation of ontology interdependencies. In *Proceedings, 7th International Conference on Extending Database Technology*, pages xx–xx, March 2000.

[MMK98] I. Muslea, S. Minton, and C. Knoblock. Wrapper induction for semistructured, web-based information sources. In *Proceedings of the Conference on Automatic Learning and Discovery, CONALD-98*, 1998.

[MP95] R. Motwani and Raghavan P. *Randomized algorithms*. Cambridge University Press, New York NY, 1995.

[NAM98] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. In *ACM SIGMOD '98*. ACM, June 1998.

[NAT99] NATO. Partnership for Peace. http://www.nato.int/pfp/partners.htm, 1999.

[Net99a] Netscape Inc. Netscape netcenter. http://www.netscape.com/, 1999.

[Net99b] Netscape Inc., Open Directory Team. Open Directory Project. http://www.dmoz.org/, 1999.

[NTR98] R. Nikolai, A. Traupe, and Kramer R. Thesaurus federations: A framework for the flexible integration of heterogeneous, autonomous thesauri. In *Proc. Conference on Research and Technology Advances in Digital Libraries (ADL'98)*, pages 46–55, 1998.

[OSSM99] D. E. Oliver, Y. Shahar, E. H. Shortliffe, and M. Musen. Representation of change in controlled medical terminologies. *Artificial Intelligence in Medicine*, 15(1):53–76, January 1999.

[PB98] L. Page and S. Brin. The anatomy of a large-scale hypertextual web search engine. *Proceedings of the 7th Annual World Wide Web Conference*, 1998.

[PG95] Y. Papakonstantinou and H. Garcia-Molina. Object fusion in mediator systems (extended version). Technical report, Stanford University, 1995.

[PHW+99] P. Panchapagesan, J. Hui, G. Wiederhold, S. Erickson, L. Dean, and A. Hempstead. The INEEL data integration mediation system. In *Proceedings, International ICSC Symposium on Advances in Intelligent Data Analysis, AIDA '99*, 1999.

[Pra97] W. Pratt. Dynamic organization of search results using the UMLS. In *Proceedings, Amia Annual Fall Symposium*, pages 480–484, 1997.

[pro99a] CLEVER project. CLEVER searching. http://www.almaden.ibm.com/cs/k53/clever.html, August 1999. HITS, Hubs and Authorities research.

[PRO99b] PROMO.NET. Project Gutenberg. http://www.gutenberg.net/, 1999.

[RDV98] S. Richardson, W. Dolan, and L. Vanderwende. MindNet: acquiring and structuring semantic information from text. In *Proceedings of COLING '98*, 1998. ftp://ftp.research.microsoft.com/pub/tr/tr-98-23.doc.

[RS91] J. Richardson and P. Schwarz. MDM: An object-oriented data model. In *Proceedings, Third International Workshop on Database Programming Languages*, pages 86–95, 1991.

[SM91] M. Siegel and S. E. Madnick. A metadata approach to resolving semantic conflicts. In *Proceedings of the 17th International Conference on Very Large Data Bases*, pages 133–145, 1991.

[Smi93] D. R. Smith. Constructing specification morphisms. *Journal of Symbolic Computation*, 15:571–606, 1993.

[Smu87] R. M. Smullyan. *Forever undecided : a puzzle guide to Gödel.* Knopf, New York, NY, 1987.

[Sow00] J. F. Sowa. *Knowledge Representation Logical, Philosophical and Computational Foundations.* Brooks/Cole, Pacific Grove, CA, 2000.

[Tom99] F. Tompa. Centre for the New OED and Text Research. http://db.uwaterloo.ca/OED/, 1999.

[UHW+98] M. Uschold, M. Healy, K. Williamson, P. Clark, and S. Woods. Ontology reuse and application. In *Formal Ontology in Information Systems, FOIS'98*, 1998.

[van67] C. H. K. van Rooten. *Mots d'heures, gousses, rames.* Penguin Books, New York, NY, 1967.

[VP99] V. Vasalos and Y. Papakonstantinou. Query rewriting for semistructured data. In *ACM SIGMOD '99.* ACM, 1999.

[Wie94] G. Wiederhold. An algebra for ontology composition. In *Proceedings of 1994 Monterey Workshop on Formal Methods*, pages 56–61. U.S. Naval Postgraduate School, September 1994.

[ZE99] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to web search results. In *Proceedings, 8^{th} International World Wide Web Conference WWW8*, May 1999.