An Algebra for Semantic Interoperation of Semistructured Data*

Jan Jannink□ Prasenjit Mitra□ Erich Neuhold△
Srinivasan Pichai□ Rudi Studer◊ Gio Wiederhold□

□ Stanford University
Stanford, CA 94305, USA
{*jan, prasen9, vasan, gio*}@
db.stanford.edu
http://www-db.stanford.edu/

△ Darmstadt U. of Technology
GMD-IPSI
D-64293 Darmstadt, Germany
neuhold@darmstadt.gmd.de
http://www.darmstadt.gmd.de/

◊ University of Karlsruhe
Institute AIFB
D-76128 Karlsruhe, Germany
studer@aifb.uni-karlsruhe.de
http://www.aifb.uni-karlsruhe.de/

Abstract

The diversity and availability of information sources on the World Wide Web has set the stage for integration and reuse at an unparalleled scale. There remain significant hurdles to exploiting the extent of the Web's resources in a consistent, scalable and maintainable fashion. The autonomy and volatility of Web sources complicates maintaining wrappers consistent with the requirements of the data's target application. Also, the sources' semantic heterogeneity requires practical methods to mediate their contents.

This paper describes our efforts in developing an algebra on semistructured data. This algebra is the tool we use to develop and maintain wrappers, and mediate their semantic content. We describe wrapper creation, refinement and maintenance as the process of developing a congruity measure between source data sets and their target application. The *congruity measure* expresses explicitly the context within which the source data is relevant for its target use. Enabling mediation between wrappers corresponds to establishing an articulation between data sources through a similarity measure. The *similarity measure* encapsulates conditions under which distinct sources may be used together.

Examples using the algebra show how the `Summarize` operator enables creation and maintenance of a target data set from a source. We define the `Match` operator and demonstrate its use in mediating the content of similar sources. We have applied these two operators respectively to an on-line dictionary with over 125,000 different terms, and to the web pages of NATO governments and their allies. We give a description of the rule language that we use to wrap and mediate sources, and conclude with the applications that motivate the other operations in the algebra.

1 Introduction

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.

* This work was supported by a grant from the Air Force Office of Scientific Research (AFOSR).

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 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 complementary. We call this the *similarity problem*, 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 problem by introducing the notion of congruity and similarity measures which approximate the relationships. 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 and outputs to the operators are directed labelled graphs with no disjoint 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 this 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 in the list below 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. These operators are listed below and discussed at greater length in Section 5.

- unary operators

  `Summarize` data classification

  `Glossarize` listing of terms

  `Filter` data instance reduction

  `Extract` structural simplification

- binary operators

  `Match` data corroboration and reformulation

  `Difference` distance measure

  `Intersect` schema discovery

  `Blend` schema extension

The novelty of the algebra is threefold. We have decoupled the selection of congruent parts of the source data from the determination of the articulation points between complementary information sources. The congruity and similarity measures are created, refined and maintained in an iterative process using the algebra rather than a separate language. 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. This paper begins with motivating examples and a review of related work. Background definitions appear in Section 2. A definition of the `Summarize` operator and an application using it follows in Section 3. Likewise, Section 4, defines an operator and its application, `Match`. Finally, the set of operators listed above is presented in Section 5.

1.1 Motivating Examples

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.

We have been experimenting with an on-line version of the 1913 Webster's dictionary that is available through the Gutenberg Project [PRO99]. The original dictionary 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. 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. In Section 3 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 only resolve 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 significant source heterogeneity from the pages of the NATO [NAT99] Web site 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.

1.2 Related Work

Inspiration for this work comes from a proposal for an algebra for ontology composition [Wie94]. The notion of articulation between ontologies [Guh91] is presented as a starting point for the development of an intersection operation. The articulation between ontologies represents the matching parts of two ontologies that are relevant to an application that requires the ontologies. Articulations blend the congruity and similarity relationships we introduce in Section 2.3

[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. We use a similar set of primitives in Section 2.2 to construct congruity and similarity measures, which approximate the corresponding semantic relationships.

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 [SM91] covers some issues relating to congruity.

Semantic integration is the focus of [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. [GKD97] is a system that 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 therefore similarly impoverished in detail. The difficulties in creating a reference schema independent of client applications are not considered in their work.

The Tsimmis project [PG95] is geared towards query answering, and wrapper specification is considered orthogonally with an independent mediator specification language (MSL). Tsimmis builds wrappers and mediators that are capable of handling some semantic heterogeneity, but not systematically nor explicitly. Wrappers may resolve conflicts and inconsistency, but do so silently. Tsimmis introduced the object exchange model (OEM) [GCCM96]. 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, and must be maintained separately from the query system.

In [NAM98] the notion of finding common schema in semistructured data roughly corresponds to our `Intersection` operator. The MDM [RS91] describes `Aggregate` and `Partition` operators, which we subsume in the `Summarize` operator presented in Section 3. We make use of the operator's ability to reduce the volume of data contained in a source so as to create and define a congruity measure between the source data and our application. In [Hul97] Description Logics (DL) are a focus of attention to represent the information content of sources. There is also discussion of efforts to integrate DL with query languages. 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 true similarity relationship between sources.

2 Background

In this section we provide an overview of the background for the algebra. We define the object model, and the primitive operations that transform objects. We proceed by defining and motivating 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. 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.

2.1 Object Model

We consider an object model for semistructured data that provides for the following abstractions:

reference object identity

value object value is a sequence of values

attribute value sets

component object references, strings, numbers, sets as primitive values

This model corresponds to XML with document identity, where an XML document corresponds to an object in our model. This model allows us to represent, without modification, HTML and XML documents, data in Stanford's OEM format, plain text and database relations. Furthermore, it is rich enough to model more complex relationships such as inheritance, and typing. Below we present primitive operations over this model which relate the different parts of the model.

2.2 Primitive Operations

While the algebra transforms object graphs, it is also convenient to be able to express these transformations 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 agebra. The set of primitives below is complete with respect to transformations of the object model defined above. The completeness proof appears in the full technical report [JMN+99]. 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 allow uninterpreted components of an object to become attributes of the object. Refinement of structure remedies the differing granularities of objects which would otherwise be similar. The operators listed below form expressions that operate on a data stream. 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.

- constructors

  create object constructor taking a sequence of values as parameters

  create set constructor taking a set of values as parameters

- connectors

  match object copy operator based on matching a value predicate

  match set repeatable copy operator based on matching a value predicate

- editors

  insert value insert a value into object at a specified position

  edit value edit a value within object at a specified position

  move value move a value within object to a specified position

  delete value delete a value from object at a specified position

- converters

  object/value string, number, set $\longleftrightarrow$ object

  object indirection object $\longleftrightarrow$ object - reference - object

  reference indirection reference $\longleftrightarrow$ reference - object - reference

2.3 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 descriptors (DTD), determine a set of consistent documents, as long as they 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..

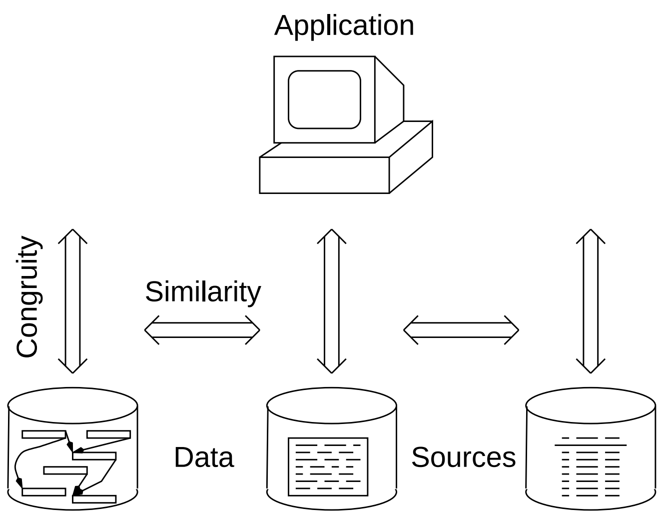

Figure 1: Relationships between Sources and Target Application

2.3.1 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 which 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 1 the congruity measure is represented by the arrow in the vertical dimension.

In our system we write the congruity relationship as a script of the primitive operators defined above. The empty script, as implied above, represents no relationship between a source and the target. A script creating a single object encapsulating the entire source, represents accepting the entire source data for the target. Such a script is also called *conguity expression*. In Section 3 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 they are mutually consistent. Any blending of information from differing sources requires additionally the application of a similarity metric.

2.3.2 Similarity Measure

The similarity measure is a binary relationship between two data sources. It identifies equivalent objects between data sources, and objects which may be merged without being identical. The horizontal arrow in Figure 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.

With definitions for the prerequisites to the algebra in hand, we are in a position to present the operators. We begin with the `Summarize` operator, and show how we use it to establish a congruity measure for a dictionary application.

3 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 2.3.1 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. 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.

Since it is difficult to grasp 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.

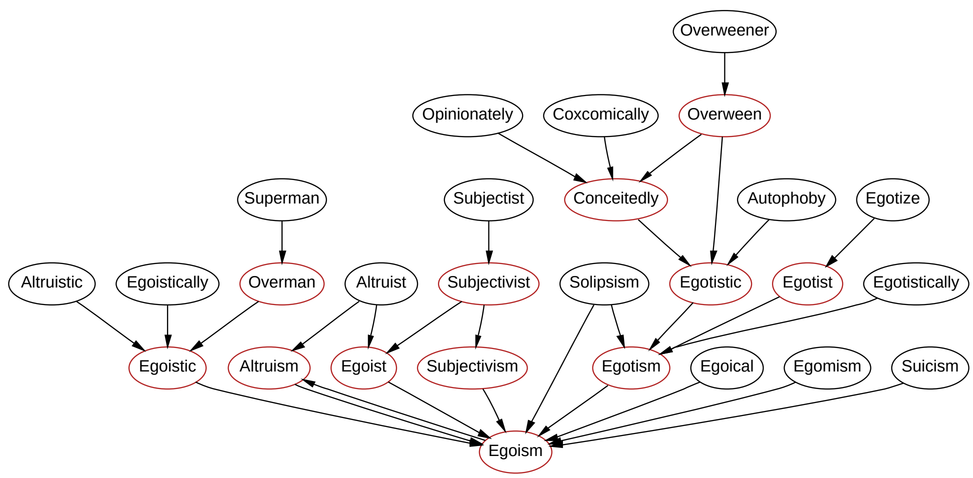

Figure 2: Automatic Thesaurus Extraction from Dictionary

3.1 Webster's Dictionary

We have been using an on-line version of the Webster's Dictionary to research the automatic extraction of thesaurus entries, such as for the term **Egoism** in Figure 2, using an implicit structure contained in the dictionary. This application supports the handling of lexical heterogeneity in data sources. This 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 to mark the parts of the definitions. Sample definitions from the dictionary data relating to Egoism are shown below:

```
<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>
```

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 alternate 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

We assumed that such a graph would be very simple to construct, but immediately ran into problems:

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

Even after accounting for accented characters, a naive script mis-assigns over five percent of the words, because of these 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.

3.2 Constructing the Congruity Expression

The initial script for the congruity measure for the thesaurus graph application over the dictionary consists roughly of the following operations (regular expressions are partially elided for conciseness of presentation):

```
match object dictionary(.*) {            // matches entire dictionary
  match set group(<hw>\(...\)*<hw>) {    // matches definition groupings
    match set hw(<hw>[^>]*</hw>) {       // matches single head words
      convert ('["`*]', '')              // removes syllable & accent markers
    }
    match set def(<def>\(...\)*</def>) { // matches single definitions
    }
  }
}
```

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 data, the output of which may be passed to another script. As mentioned in Section 3.1 this script very approximately expresses the congruity relationship between the dictionary and the thesaurus application. For lack of space, the final script, containing over 400 conversion operations, that perform over 300,000 transformations, is not presented here. Instead, we show how the S operator allows us to capture some of the classes of conversions which enhanced the above initial script.

3.3 Applying the `Summarize` 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.

3.4 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.

4 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 3 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 4. Similarly shaped nodes in the two figures are considered similar by the NATO website.

4.1 The Match or 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 3. 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 they point to. Each node in the graph corresponds to a Web page and is assigned a term, or label–we use these terms interchangeably– 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:

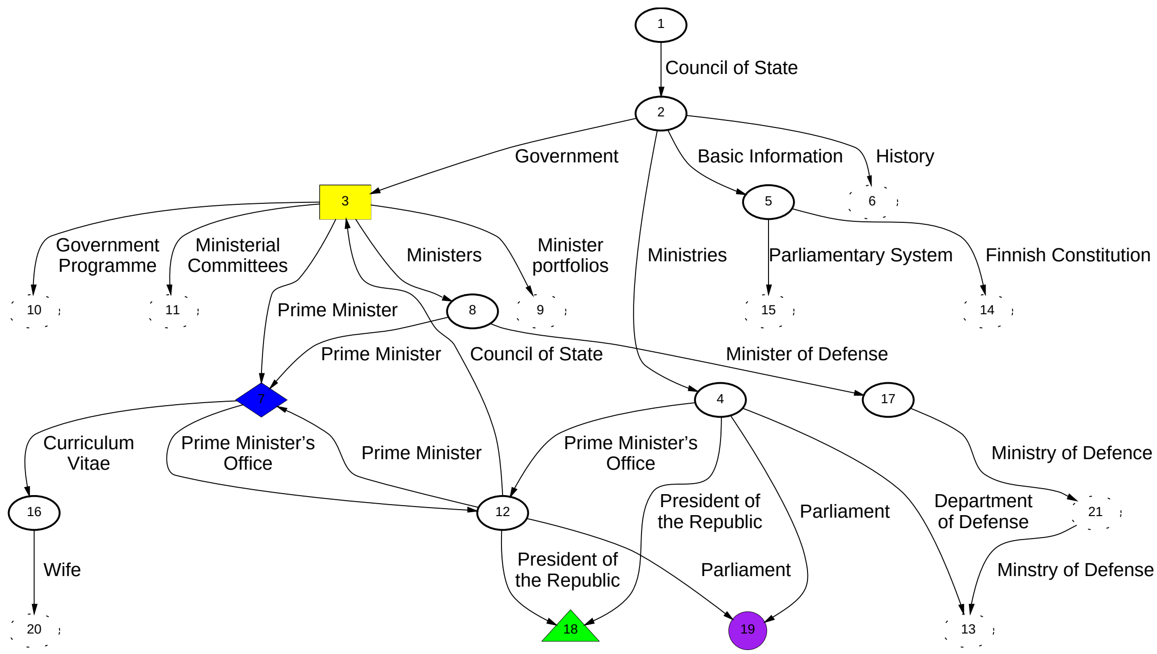

Figure 3: Partial Graph of Finnish Government Web Site

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 4 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 4 is a parent to both the House of Commons and the House of Lords, while Parliament refers to a single body in Figure 3.

4.2 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.

4.2.1 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.

4.2.2 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 4 with that labelled **Government** in Figure 3. 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.

4.2.3 Term Mismatch Resolution

These operations simply express that two terms are semantically related and should match. Examples from our two figures include rules like: `(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.

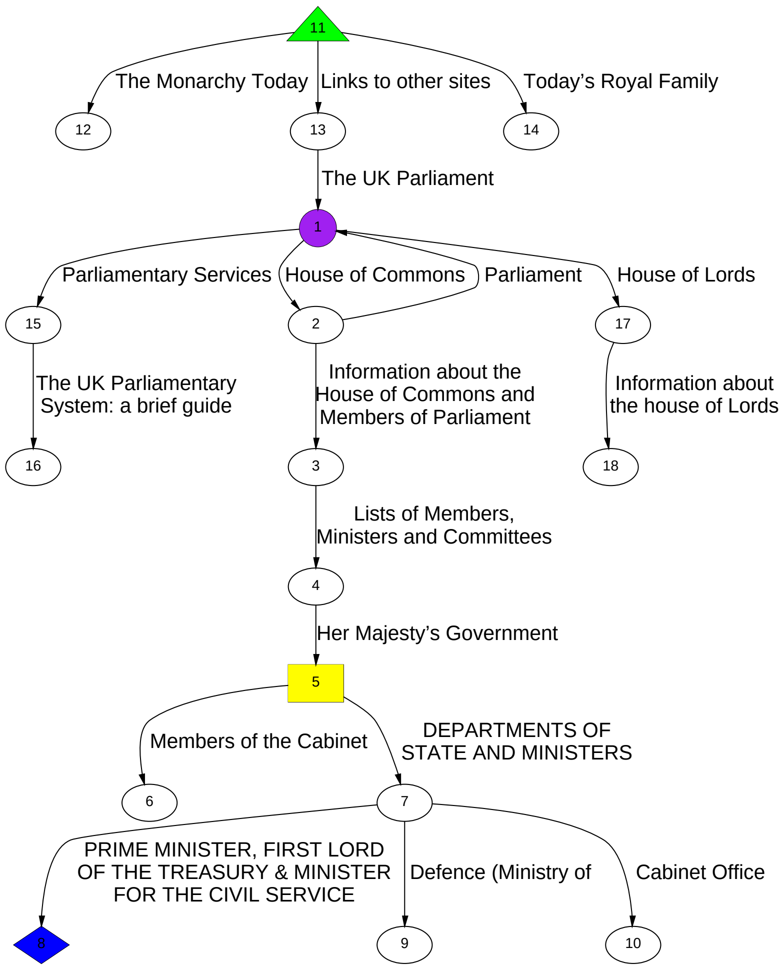

Figure 4: Partial Graph of U.K. Government Web Site

4.2.4 Irrelevant Match Resolution

A preprocessing stage removes stop-words like 'of', '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.

5 Algebra Operators

In this section we review the operators which 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. We refer readers to the technical report [JMN+99] for a complete presentation of the operators.

5.1 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.

5.1.1 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.

5.1.2 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 easily list all terms that are subordinate to an object.

5.1.3 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.

5.1.4 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.

5.2 Binary Operators

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

5.2.1 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.

5.2.2 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 application motivating the addition of B to the algebra is to extend the schema of matching objects, thereby creating blended subclasses of objects.

5.2.3 Intersect

The `Intersect` operator or I returns only the portions of the sources which 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.

5.2.4 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 Conclusion

In this paper 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 is a work in progress, and we must begin to explore performance issues, as well as scaling to larger sources. It is a first step in a systematic treatment of semantic heterogeneity within the framework of an algebra over semistructured data.

References

[AGM+97] Serge Abiteboul, Roy Goldman, Jason McHugh, Vasilis Vassalos, and Yue Zhuge. Views for semistructured data. In *Proceedings of the Workshop on Management of Semi-Structured Data*, pages 83–90. NSF, 1997.

[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.

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

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

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

[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.

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

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

[HM93] Joachim Hammer and Dennis 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.

[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). Technical report, Stanford University, 1999.

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

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

[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.

[NAM98] Svetlozar Nestorov, Serge Abiteboul, and Rajeev 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.

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

[PRO99] PROMO.NET. Project Gutenberg. http://www.gutenberg.net/, 1999.

[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.

[Wie94] Gio 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.