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.

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.

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 an application of the data, for which a source was 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.

Figure 1 illustrates a typical usage scenario for the results of our work. The upper right hand portion of the figure represents a supply chain interoperation example, with heterogeneous supplier and facility databases interfaced through an articulation. The articulation, representing a minimal, sufficient set of linkages between items in different information sources, is managed by a domain expert who compiles knowledge from relevant sources. These sources vary from free text to XML and also relational tables, with a set of articulation development tools. This knowledge supports the expert in generating the supply chain articulation. The expert also depends on feedback from the articulation to improve the precision of the suggestions made by the articulation development tools.

Articulation development tools don't appear from a void.

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

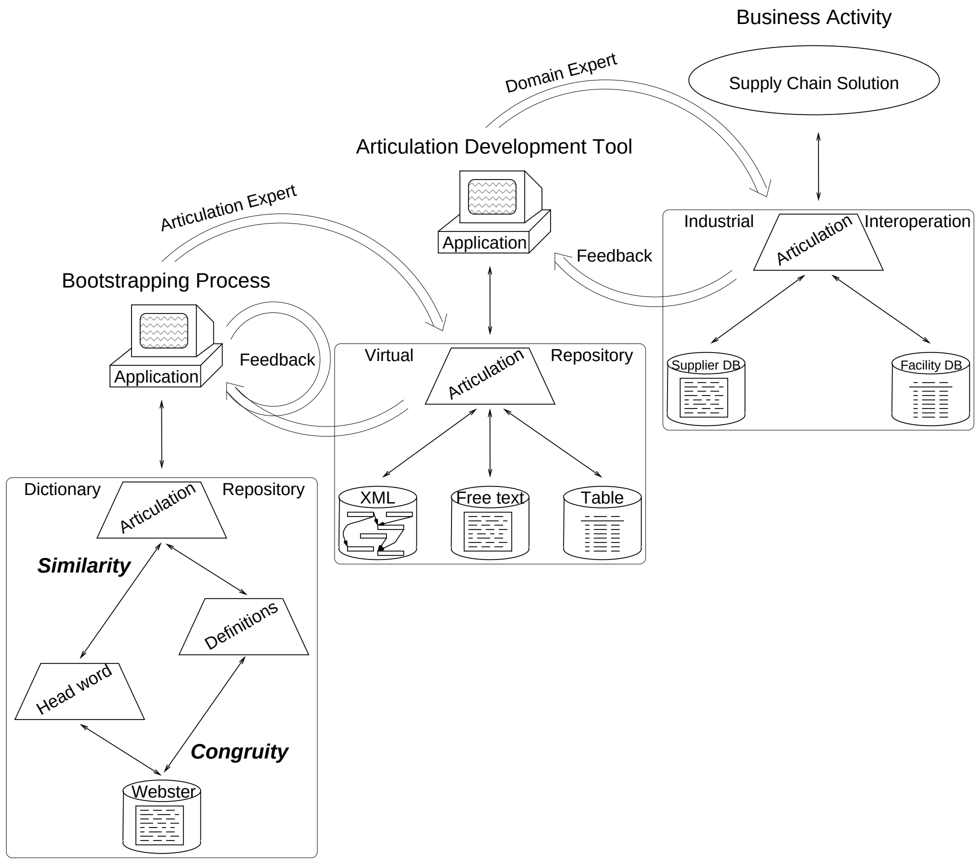

Figure 1. Typical Algebra Usage Scenario

Building tools in an ad hoc fashion, to support a principled algebraic approach is also clearly unacceptable. The lower left hand of Figure 1 shows how we develop knowledge for our tools using our algebraic approach. We have taken on-line references such as the Webster's dictionary, and extracted the set of head words or defined terms, along with their definitions, using articulations based on the congruity relationship. Next, with an articulation relying on the similarity relationship, we created a virtual repository of dictionary terms, a directed graph where terms link to others whenever they appear in their definitions. We used the growing dictionary repository to iteratively refine and improve its own structure by identifying correctable errors, such as misspellings, and discovering classes of terms, such as proper nouns that were not defined, but still integratable into the structure. This feedback loop allowed us to bootstrap the process of defining the algebra and the tools that use it, within the algebraic framework. The repository serves as both the first instance of an application for the algebra, as well as supporting the algebra itself for defining new articulations.

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. We begin with a review of related work. Background definitions appear in Section 2, and the set of operators listed above is presented in Section 3.

1.1. Motivating Example

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 [12], and on the subsequent access of the data through a query interface [11]. 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 present some results from our work using an on-line version of the 1913 Webster's dictionary that is available through the Gutenberg Project [14]. 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 4 we show how we iteratively refine a simple wrapper using the Glossarize, Filter, Summarize and Match operators from our algebra to reduce the exception rate to well below one percent. We also show how the initial work on the Webster's repository carried over to Roget's Thesaurus and the Oxford English Dictionary (OED).

1.2. Related Work

Inspiration for this work comes from a proposal for an algebra for ontology composition [16]. The notion of articulation between ontologies 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

Some work on the use of context to handle database object semantics is found in [10]. This work uses a semantic proximity relationship, but does not discuss relevance of source data to a client application. The system in [7] uses gradations of a consistency relationship in relating information sources. In considering semantic reconciliation between data source and data receiver [15] covers some issues relating to congruity. WHIRL [4] uses textual similarity to find co-referent terms in distinct sources with high accuracy. The textual similarity measure it defines is an example a similarity measure that approximate the similarity relationship between sources.

Semantic integration is the focus of [1], 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. Aurora [17] allows for the manual construction of interfaces to allow heterogeneous relational databases to interoperate using integrated views. [6] 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. Description Logics (DL) [8] are a focus of attention to represent the information content of sources. There is also discussion of efforts to integrate DL with query languages.

The Tsimmis project [13] 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. Query languages for semistructured data such as UnQL [2], XML-QL [5] 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. [3] introduces a set of primitive operators on objects in a 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.

2. Underpinnings

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 possibly nested sequence of values

attributes set of labeled references
atom attributes, strings, numbers 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 algebra. 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 [9]. 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 (strings) of an object to become attributes of the object. 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.

- constructor
  create object constructor taking a sequence of values as parameters
- connectors
  match object $a \Leftrightarrow b \longleftrightarrow a$ equates to b
  imply object $a \Rightarrow b \longleftrightarrow b$ subsumes a
- 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, attribute} $\longleftrightarrow$ object
  object indirection object $\longleftrightarrow$ object - reference - object
  reference indirection reference $\longleftrightarrow$ reference - object - reference

2.3. Semantic Consistency

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. 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.

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 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 congruity expression.

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 measure.

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 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. With definitions for the prerequisites to the algebra in hand, we are in a position to present the operators.

3. 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. 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. We refer readers to the technical report [9] for a complete presentation of the operators.

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 domains. 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 lists 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.

3.1. 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.

Summarize The canonical unary operator of our algebra, 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 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.

Filter The 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 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.

3.2. 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.

Match The prototypical binary operator, 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 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.

Intersect The I operator 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.

Difference The D operator 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.

4. 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.

4.1. 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.

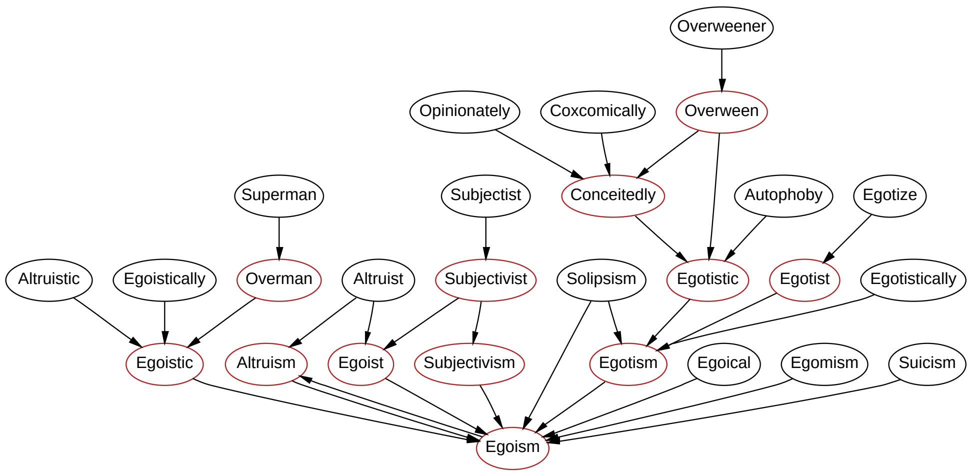

Figure 2. Automatic Thesaurus Extraction from Dictionary

4.2. 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¿[ˆ¿]*¡/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 4.1 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.

4.3. 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, $S_{max(10,len(hw))}$(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.

4.4. 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.

5. 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 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

[1] 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.

[2] 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.

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

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

[5] 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, Aug. 1998. Working Draft.

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

[7] 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, Mar. 1993.

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

[9] 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.

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

[11] 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.

[12] 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.

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

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

[15] 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.

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

[17] L. L. Yan, M. T. Özsu, and L. Liu. Accessing heterogeneous data through homogenization and integration mediators. In Proceedings, Second IFCIS International Conference on Cooperative Information Systems, pages 130–139, June 1997.