References

[1] S. B. Navathe C. Batini, M. Lenzerini. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys, Vol 18, No. 4*, pages 323–364, December 86.

[2] Won Kim and Jungyun Seo. Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer*, pages 12–18, December 1991.

[3] S. A. Mamrak, J. Barnes, and C. S. O'Connell. The benefits of using the integrated chameleon architecture for data translation. *IEEE Software*, January 1992. to appear.

[4] S. A. Mamrak, M. J. Kaelbling, C. K. Nicholas, and M. Share. Chameleon: A system for solving the data translation problem. *IEEE Transactions on Software Engineering*, 15(9):1090–1108, September 1989.

[5] Sandra A. Mamrak and Julie Barnes. Considerations for the preparation of SGML Document Type Definitions. *Electronic Publishing: Origination, Dissemination and Design*, 4(1):27–42, March 1991.

[6] *Information Processing—Text and Office Systems—Standard Generalized Markup Language (SGML)*, October 1986. ISO 8879-1986 (E).

to translate an instance of the source schema to an instance of the unified schema to an instance of the target schema.

5 Summary and Future Work

We have described a Universal Framework for Data Translation. The goal of this framework is to provide a better understanding of the generalities of the data translation problem. We intend to investigate the validity of this framework by building tools to implement various layers of it.

We have already achieved success with a tool for automatically generating the code to translate record-oriented data from arbitrary specific encodings to our generalized representation. This tool will be of considerable use for applying our framework to database applications.

Our work will concentrate on the general model portion of the framework. We are currently developing the Nagiya data model, an object-oriented model that is general enough to support translations between various data models. We are also developing a tool for specifying relationships between schema in this model so that translation code can be automatically generated.

Acknowledgment. We wish to thank the other members of the Nagiya group for their efforts and input into this work. These include: Ashutosh Agharkar, Krishna Balasubramanian, John Boyd, Greg Nichols, Dr. C.S. O'Connell, Kalyanaraman Viswanathan, Peter Ware, and Andrew Zwerin

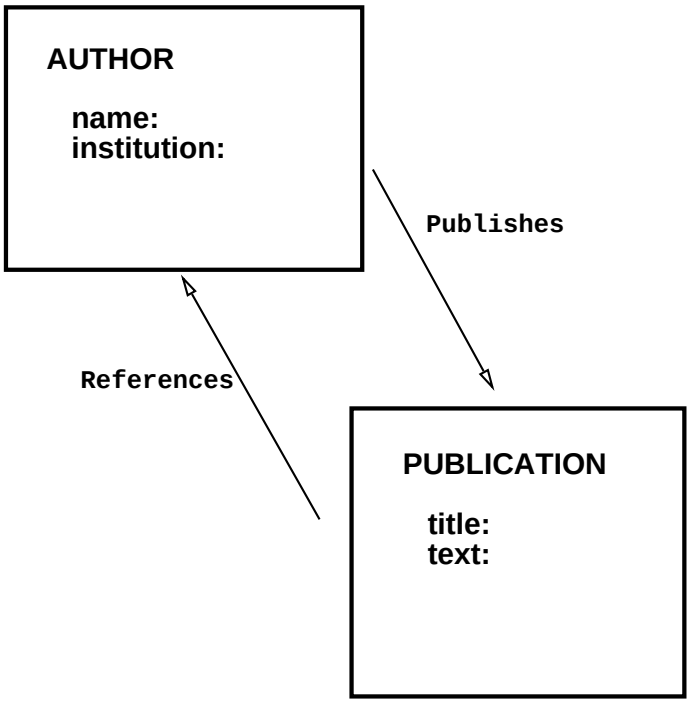

Figure 10: Uniform Schema/General Model

This notation denotes that an instance of every DOC object in the source data is related to an instance of AUTHOR and an instance of PUBLICATION in the unified schema where the AUTHOR and PUBLICATION are connected by a Publishes link. Also the appropriate relationships between the object attributes are specified.

Similarly the same relationship can be defined for integrating the unified schema with the target schema in the general model. This relationship, define from the unified schema to the target schema may look like:

```
Link(Publishes)            --> Object(AUTHOR_INFO)
                               Object(DOC_TEXT)
Attr(AUTHOR.name)          --> Attr(AUTHOR_INFO.author)
Attr(AUTHOR.institution)   --> Attr(AUTHOR_INFO.institution)
Attr(PUBLICATION.title)    --> Attr(AUTHOR_INFO.title)
Attr(PUBLICATION.title)    --> Attr(DOC_TEXT.title)
Attr(PUBLICATION.text)     --> Attr(DOC_TEXT.text)
```

Given this mapping this, it possible to automatically generate the code

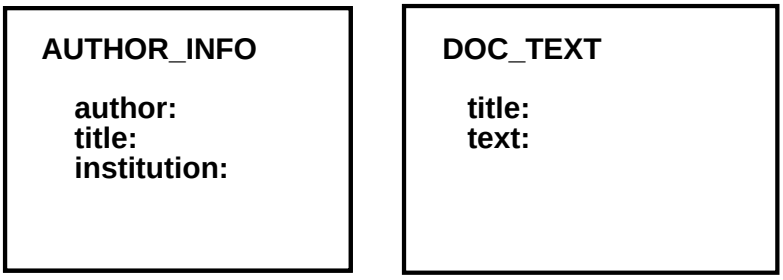

Figure 9: General Model/Specific Schema of Target Data

schema also allows the framework to enjoy the benefits of the intermediate form approach to data translation.

The uniform schema in this example is a schema in the general model that is independent of any individual applications view of information. The uniform schema we have chosen for this example is shown in Figure 10.

This schema consists of two object classes, author and publication. The relationships denote that authors "publish" publications, while publications "reference" authors.

In order to integrate the source schema with the uniform schema it is necessary to to recognize conflicts or relationships between the objects in each of the schema. We can define this relationship as a mapping from the source schema to the unified schema as follows:

```
Object(DOC)             --> Object(AUTHOR)
                            Object(PUBLICATION)
                            Link(Publishes,AUTHOR,PUBLICATION)
Attr(DOC.title)         --> Attr(PUBLICATION.title)
Attr(DOC.author)        --> Attr(AUTHOR.name)
Attr(DOC.affiliation) --> Attr(AUTHOR.institution)
Attr(DOC.body)          --> Attr(PUBLICATION.text)
```

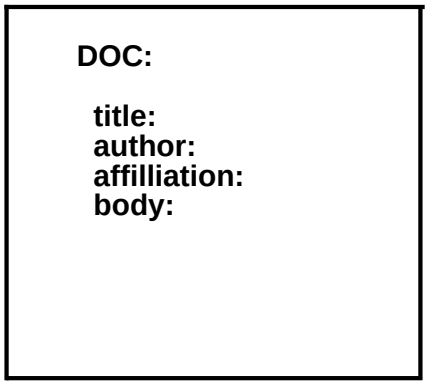

Figure 8: General Model/Specific Schema of Source Data

making all non-terminals with productions that are not a lone terminal an object class and each non-terminal with a production of a lone terminal an attribute of the appropriate class. Any non-terminal within a non-terminal will be denoted by the appropriate relationship. The example schema of the source document will represent a trivial transformation to a schema in the general model and is shown in Figure 8.

Likewise the target data schema can be transformed to a schema in the general model. In the case of relational data, one mapping to the general model is to define a class for each table type, each field in the table becomes an attribute of the class. Such a mapping of the target relational schema produces the general model schema shown in Figure 9.

4.3 Integration of General Model Schema

The final layer in our framework involves classifying and resolving the conflicts between the source and target schema within the general model by way of a uniform schema. Although the translation between source and target schema can be accomplished by defining the conflicts between the two specific schema in the general model, the use of a uniform schema in the framework is important. The uniform schema allows the translation between source and target to be conceived within a common set of objects. The uniform

```
<db>
<table><table_name>author_info</table_name>
<record>
<field><name>author</name>
<value>S. A. Mamrak</value>
</field>
<field><name>title</name>
<value>Toward a Universal Framework For
        Data Translation
</value>
</field>
<field><name>institution</name>
<value>Ohio State University</value>
</field>
</record>
<table><table_name>doc_text</table_name>
<record>
<field><name>title</name>
<value> ....
   o
   o
   o
</db>
```

Figure 7: General Encoding/Specific Model of Target Data

```
<doc>
  <title> Toward a Universal Framework for Data
          Translation
  </title>
  <author>S. A. Mamrak</author>
  <affiliation>Ohio State UNiversity</affiliation>
  <body>Several methods currently exist for ...
  </body>
</doc>
```

Figure 6: General Encoding/Specific Model of Source Data

By definition, relational schema consist of a set of tables, all tables consist of a set of tuples and all tuples consist of attributes. It is acceptable to use the similar terminology of tables, records and fields. We can encode all relational data as a context-free grammar of the form:

```
DB         --> TABLE+
TABLE      --> TBL_NAME, RECORD+
TBL_NAME   --> data
RECORD     --> FIELD+
FIELD      --> NAME, VALUE
NAME       --> data
VALUE      --> data
```

The target data represented in this general encoding is shown in Figure 7.

This first phase has the effect of hiding all of the encoding peculiarities of the individual applications from the translation specification process.

4.2 Specific vs. General Data Model

The second layer in our framework calls for transforming the instances of the individual specific data models into one common more general model.

Assume a model that consists of object classes, attributes of objects in those classes and relationships between instances of the classes. The source data which is described by a context-free grammar, can be transformed by

```
@table:author_info
@record
@field:author
@value:S. A. Mamrak
@field:title
@value:Toward a Universal Framework
   For Data Translation
@field:institution
@value:Ohio State University
@table:doc_text
@record
@field:title
@value:Toward a Universal Framework
   For Data Translatio
@field:text
@value:Several methods currently ...
```

Figure 5: Specific Encoding/Specific Model of Target Data

model. As an example consider the source document. Documents can be characterized as hierarchical objects. These hierarchies can be modeled as context-free grammars. We can declare the data model of the document to be the set of context-free grammars. The specific context-free grammar shown above would be the schema within the data model and the document shown in Figure 3 is a specific encoding of an instance of that schema.

A similar classification is true for the target representation. The data model is the relational model, the schema within the data model is that of the two tables shown in Figure 4. The specific encoding of an instance of that relational schema for the target application is shown in Figure 5.

The result of applying the first level of the framework to our source and target data would be the generalized representations of the specific schema instances within their respective data models.

In the case of the source data we use a general encoding for representing instances of context-free grammars where the lone terminal symbol represents some string of text. This particular encoding can be defined using SGML [6, 5]. The general encoding of the source data is shown in Figure 6.

In the case of the target data it is also necessary to determine a general encoding for the target data model. In this example the target data is represented in a schema within the relational data model. We can represent all instances of relational schema with a general encoding as well.

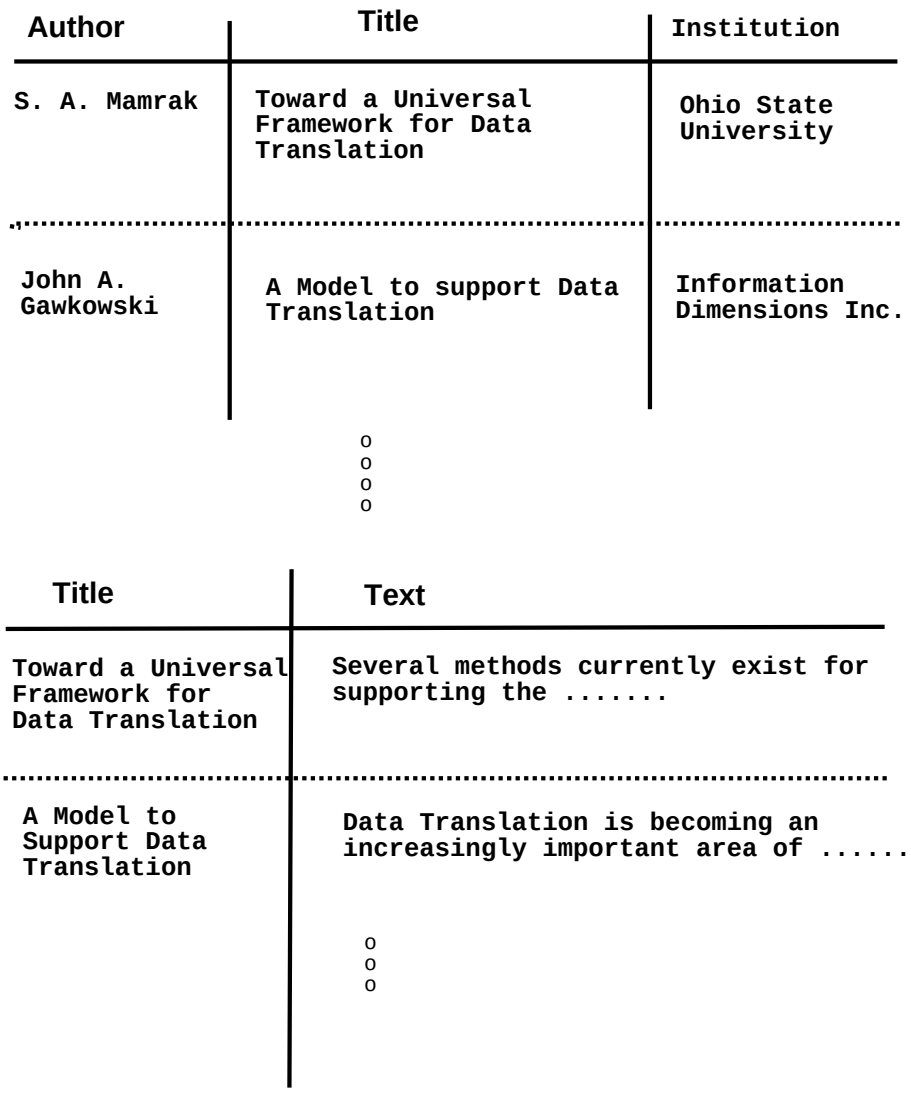

Figure 4: Relational Database Schema of Target

```
o
o
\begin{document}
\hyphenation{}
{\bf Toward a Universal Framework for Data
Translation}
\vspace{.1in}
{\small
S. A. Mamrak\\
Ohio State University\\
}
\vspace{.1in}
\end{center}
\section{Introduction}
 Several methods currently exist for supporting
 the ......
 o
 o
 o
```

Figure 3: Specific Encoding/Specific Model of Source Data

```
AFFILIATION  --> text
BODY          --> text
```

The goal of the translation is to load documents conforming to the above grammar into a relational database application that uses the schema shown in Figure 4.

This schema consists of 2 relations, one representing author information and one representing document titles and text.

4.1 Specific vs. General Encoding

In this translation scenario the source and target data are each intended to be manipulated by fundamentally different applications. The source, by the LaTeX text formatting system and the target by the BASISplus relational database system. In addition each of these applications relies on a different data model. According to the framework, in order to eventually resolve all of the differences between these applications, we must first abstract away the specific encoding schemes of the applications. This process will allow us to concentrate on the data objects being translated rather than the peculiar encodings used to mark those objects.

The first process in the framework is the transition of data from a specific encoding in a specific data model to a general encoding within the same data

is a different schema in the model. Some grammars will require different occurrence of symbols. Some will require different orderings. Thus, a schema integration is required to complete the translation cycle.

The second example is that of translation between data encoded using the same database management package, e.g., Oracle. No encoding translation is required because the data have identical lexical encodings. No model translation is required because the data share a common, general data model, i.e., one of tables, records, fields, and field values. Schema integration is required because conflicts among the individual schemas will likely arise. What is a table in one data set may be a field in another, and so on. Thus, only the schema integration task must be executed to achieve the desired translation between the data objects.

The third example is that of translation between a manuscript and a database. Encodings will be different, i.e., a tag/data stream from the text formatter and an ASCII export file from the database. Models will also vary, i.e., a grammar for the manuscript and the table/record/field model for the database. Assuming a model translation to an object-oriented model, schema conflicts will likely result. Thus, for this example, all three translation steps of encoding translation, model translation and schema integration are required to complete the translation process.

4 A Sample Translation

In this section we describe a translation between documents prepared for the text formatting system LaTeX and a relational database. The requirements of this task are to create a translator which will accept documents in LaTeX format and prepare them for loading into the relational database system BASISplus. A sample document is shown in Figure 3.

The source documents for this particular translation are encoded using the LaTeX formatting language. These types of documents are hierarchical by nature and can be described by the following context-free grammar:

```
DOC          --> TITLE, AUTHOR, AFFILIATION, BODY
TITLE        --> text
AUTHOR       --> text
```

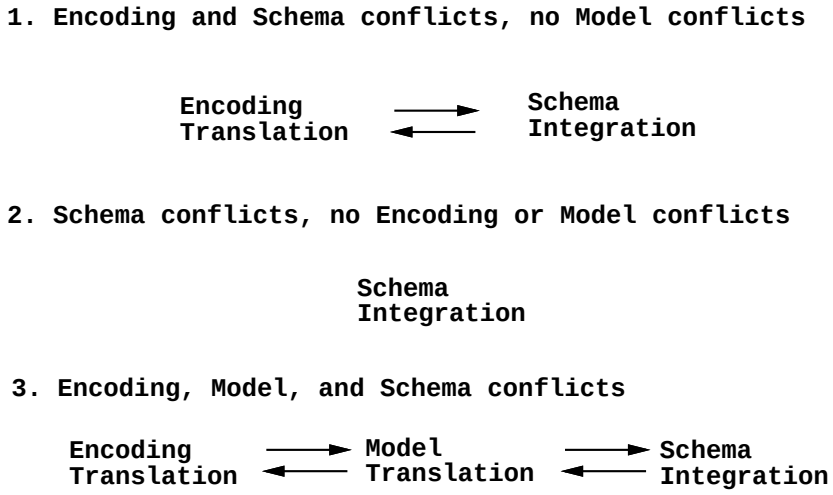

Figure 2: Some Paths through the Framework

schema, vary, as do the required translation steps.

The first example is that of translation among data encoded using text-formatting encoding schemes. This is the class of translation that is supported in the ICA. An encoding translation is required because text formatters typically use widely varying encoding schemes. Translation of these specific schemes to a general encoding scheme in which all tags in the scheme are unique and explicit greatly eases the task of data translation.

Model translation is not required in this example. The data objects that are encoded using text formatters can typically be modeled using a grammar. That is, most documents can be described as a hierarchy of objects, where order and occurrence of objects is important. For example, an article may be defined as consisting of optional frontmatter, one required body, and optional rearmatter. Frontmatter may be further decomposed into exactly one title, one or more authors, an address, and an optional date. A body may be decomposed into sections, subsections, and paragraphs, and so on. Because this is a common model to all data objects in this translation scenario, no model translation is required.

Schema integration is required. The class of context-free grammars is the general data model, but each grammar to which a document conforms

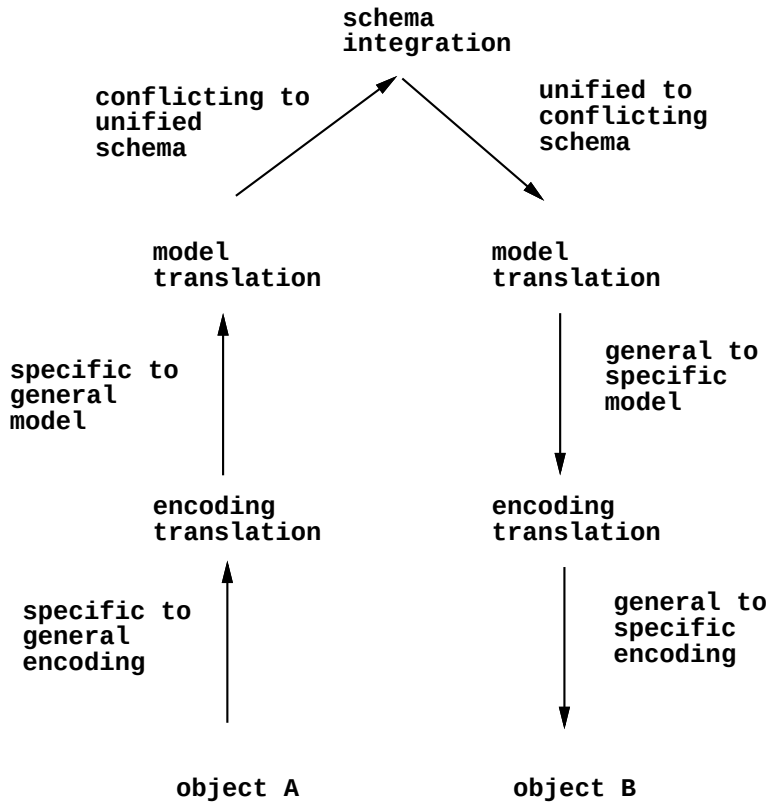

Figure 1: A Framework for Universal Data Translation

full generality. The model is shown in Figure 1.

The framework relies on representing data in increasingly higher-lever abstractions until it becomes possible to express mappings between common objects. In the general case, we wish to achieve a translation between two different data objects, say from object A to object B. Within the framework, this would require for object A a translator from specific to general encoding, specific to general model, and conflicting to unified schema. Then, required for B, is a translator from unified to conflicting schema, general to specific model, and general to specific encoding to complete the translation bridge.

3.1 Examples of Translations Within the Framework

To further elaborate the framework, we present three examples of how it may be used to describe translation among different sets of data objects. Figure 2 depicts the path through the framework that is required for each example translation. In the examples the kinds of conflicts, i.e., encoding, model or

the model, i.e., they may have different schemas. Eliminating these organizational conflicts allows focus to shift to the true commonalities among the data objects. For example, in an object-oriented model, classes in one schema may be modeled as relationships in another schema. Or one object in one schema may be decomposed into two objects in another schema. Or, objects may be named differently in the two schemas. This translation step would re-organize the individual object-oriented models using a unified, integrated schema.

The "true" translation can be considered only at this level of abstraction. That is, at some point in any translation it is necessary to conceptualize the source and target objects within some common space. The unified schema in the general model provides this space. The translation can then be defined by resolving conflicts between specific schema and the unified schema within the general model. Kim [2] has shown that it is possible to classify conflicts between schema in the relational model. Our efforts will concentrate on the classification of conflicts between schema in a more general object-oriented data model. The output of this set of conflict resolutions will be the code to automatically translate data from one schema to another in the general model.

Database schema integration has been studied in a great amount of detail in the past [1], but these efforts have typically been for supporting heterogeneous database systems via a uniform query interface. Our intention is to use the concept of schema integration within a common general model as a means to specify the automatic generation of translators to and from an intermediate representation in order to support the translation of data between multiple applications that may or may not rely on fundamentally different data models. Therefore our work is distinctly different from the area of heterogeneous databases.

3 A Proposed Framework to Support Universal Data Sharing

In support of this view of the data translation problem, we propose a universal framework for electronic data translation. This framework emphasizes the three distinct processes that are inherent to the data translation task in its

The first task in developing translations to or from some arbitrary representation of data involves abstracting away any specific encoding issues. This abstraction isolates the well understood problems of lexical analysis and parsing and allows focus to shift to the underlying instance of the particular data model being considered. We call this step a shift from a specific encoding in a specific data model to a general encoding in a specific data model.

For example, there exist several relational database products such as Oracle, BASISplus and DBase. Each of these products deal with the same type of objects, (i.e. tables, records, fields) but each uses a different ASCII encoding scheme to dump and load data. When dealing with data in this domain the first stage of translation would involve encoding all relational database representations into some generalized representations of tables, records and fields.

2.2 Model Translation

The second component in the data translation process involves translating a representation from a specific data model to a general data model. This step is required because data to be translated may be organized using different specific data models. Eliminating issues relating to specific logical organization of data allows focus to shift away from semantics of varying data models to a model with common organizational semantics.

This stage represents a transition from a specific data model with a specific schema to a general data model with a specific schema.

For example, some manuscripts may be modeled as a set of hierarchical relationships among objects, i.e., a grammar. Alternatively, some databases are modeled as tables, records, and fields. This translation step would remodel manuscripts and databases using a general model in which sections, chapters, tables, and so on, are represented using a common scheme, independent of the semantics of grammar or table-based models.

2.3 General Model Schema Integration

The third component involves translating from conflicting schemas to a unified schema within the general data model. This step is required because data represented using the same model may be organized differently within

of well understood problems such as lexical analysis and parsing in order to concentrate on the fundamental problem of the task at hand, that is the mappings between objects in the source and target representations.

The goal of the Nagiya research project is to build upon the foundation provided by the Chameleon project in order to support the automatic generation of translators between different types of applications and fundamentally different data models. It is our intention to provide generalizations that apply to a much larger class of data translation problems.

In an effort to meet and understand these more general data-sharing challenges, we have proposed a universal framework for electronic data translations. This framework emphasizes three distinct processes that are required for the translation task in its full generality. Such generalizations shall contribute to solving the problem of allowing data to be easily transported in an environment with heterogeneous applications.

In addition we are developing a set of tools that implement the various levels of the framework. The goal of this implementation is twofold, 1) to validate the ideas proposed in our framework; 2) to create a toolset that will allow translation code to be automatically generated for a large class of translation problems.

2 Distinct Tasks in Data Translation

In order to better understand the problem of data translation we have categorized the problem of defining translations into three independent tasks or problems. The three underlying assumptions of our work is that any data that is electronically represented will will 1) have a specific encoding; 2) conform to a specific set of underlying logical constructs or a data model; 3) represent a specific schema within the data model. In this section we describe these three assumptions and their relationship to one another within the general data translation problem.

2.1 Encoding Translations

Any data electronically represented, can be viewed as a specific encoding of some instance of a schema within some underlying logical representation or data model.

1 Introduction

The Chameleon Research Project at the Ohio State University has successfully demonstrated that there exists certain generalizations involving translation of data from one electronic form to another [4]. The result of this work has been an integrated architecture for developing translators within this domain. The Integrated Chameleon Architecture (ICA) provides a high-level interface for defining translations. The benefits of using such an architecture for building translators include [3]:

- elimination of coding errors through the use of a high level specification interface
- complete and correct translations by enforcement of a methodology
- increase in productivity
- ease of integration of additional translators by use of an intermediate form model

The current Chameleon Architecture has proven to be successful by limiting its focus to one particular class of data objects having similar syntax and semantics. The problem of data sharing in general involves more than issues of syntax and semantics. Often this problem introduces mismatch in the underlying logical structure of differing data representations. Consider the case of oncology researchers who need to share output from a flow cytometry machine with a statistical analysis package, or electronic publishers who need to include tabular information from a relational database as part of an electronic manuscript.

These problems represent a situation where information needs to be physically translated from one encoding scheme representing one data model to an encoding of an entirely different data model.

The current state-of-the-art for these types of situations involves hand-coding translators whenever such a problem arises. This task is typically labor-intensive and often results in much of the same work being done over and over again. We believe that much of the translation problem can be generalized in such a way as to support the development of tools that will allow the creator of a translation to abstract away the nitty-gritty details

Contents

1 Introduction **2**

2 Distinct Tasks in Data Translation **3**
2.1 Encoding Translations 3
2.2 Model Translation 4
2.3 General Model Schema Integration 4

3 A Proposed Framework to Support Universal Data Sharing **5**
3.1 Examples of Translations Within the Framework 6

4 A Sample Translation **8**
4.1 Specific vs. General Encoding 9
4.2 Specific vs. General Data Model 12
4.3 Integration of General Model Schema 14

5 Summary and Future Work **17**

ABSTRACT

Electronic representations of data objects vary widely. The same information can typically be represented in numerous ways. These differences can be realized at various levels of abstraction.

Such variations can be the result of 1) multiple views of data; 2) differences between application data models; 3) differing proprietary formats of applications. This plethora of underlying representations of electronic data makes it extremely difficult for information to be transported between applications.

The most general response to the data translation problem is to provide a mechanism whereby, given the description of two data representations, code is automatically generated to translate between them.

In an effort to approach this ultimate goal, we propose a universal framework for data translation. This proposed framework captures certain generalities that are present in all translation tasks. In addition, we are developing an architecture that is modeled after this framework. The goal of the architecture is to validate the ideas proposed in the framework, showing that a large part of the data translation problem can be generalized

Keywords: code generation, data translation, intermediate form, object oriented data model

Toward a Universal Framework for Data Translation

John A. Gawkowski
Information Dimensions Inc.
5080 Tuttle Crossing Blvd.
Dublin, Oh. 43017
email: gawkowski@idicl1.idi.battelle.org
phone: (614) 761-7436

S. A. Mamrak
Department of Computer and Information Science
The Ohio State University

November 23, 1992