

# Combining the Best of Global-as-View and Local-as-View for Data Integration\*

Li Xu

Department of Computer Science  
University of Arizona South  
lxu@email.arizona.edu

David W. Embley

Department of Computer Science  
Brigham Young University  
embley@cs.byu.edu

**Abstract:** Currently, there are two main basic approaches to data integration: Global-as-View (GAV) and Local-as-View (LAV). However, both GAV and LAV have their limitations. In a GAV approach, changes in information sources or adding a new information source requires revisions of a global schema and mappings between the global schema and source schemas. In a LAV approach, automating query reformulation has exponential time complexity with respect to query and source schema definitions. To resolve these problems, we offer TIQS as an alternative point of view that is neither GAV nor LAV. The approach uses source-to-target mappings based on a predefined conceptual target schema, which is specified ontologically and independently of any of the sources. The proposed data integration system is easier to maintain than both GAV and LAV, and query reformulation reduces to rule unfolding. Compared with other data integration approaches, our approach combines the advantages of GAV and LAV, mitigates the disadvantages, and provides an alternative for flexible and scalable data integration.

## 1 Introduction

Data integration refers the problem of combining data residing at autonomous and heterogeneous sources, and providing users with a unified global schema [Hal01]. Two main concepts constitute the architecture of a data integration system [Ull97]: wrappers and mediators. A *wrapper* wraps an information source and models the source using a *source schema*. A *mediator* maintains a *global schema* and *mappings* between the global and source schemas. As is usual, we focus here on data integration systems that do not materialize data in the global schema. Whenever a user poses a query in terms of relations in the global schema, the mediator uses a *query-reformulation* procedure to translate the query into sub-queries that can be executed in sources such that the mediator can collect returned answers from the sources and combine them as the answer to the query.

Currently, there are two main initiatives to integrate data and answer queries without materializing a global schema: Global-as-view (GAV) [CGMH<sup>+</sup>94] and Local-as-View(LAV) [LRO96, GKD97]. [CLL01] surveys the most important query processing algorithms proposed in the literature for LAV, and describes the principle GAV data integration systems

---

\*This material is based upon work supported by the National Science Foundation under grant IIS-0083127.

and the form of query processing they adopt. In a GAV approach, query reformulation reduces to simple rule unfolding (standard execution of views in ordinary databases). However, changes in information sources or adding a new information source requires a database administrator (DBA) to revise the global schema and the mappings between the global schema and source schemas. Thus, GAV is not scalable for large applications. LAV scales better, and is easier to maintain than GAV because DBAs create a global schema independently of source schemas. Then, for a new (or changed) source schema, the DBA only has to give (adjust) a *source description* that describes source relations as views of the global schema. Automating query reformulation in LAV, however, has exponential time complexity with respect to query and source schema definitions. Thus, LAV has low query performance when users frequently pose complex queries.

As data explodes on the Web, E-business applications such as comparison shopping and knowledge-gathering applications such as vacation planning raise the following issues for approaches to data integration. (1) The number of sources to access and integrate is large. (2) The sources are heterogeneous, autonomous, and possibly change frequently. (3) New sources continually become available and become part of the system. (4) Users frequently pose queries over the system to retrieve data. (5) As applications evolve, DBAs may wish to change the global schema to include some new items of interest. To address these issues and the problems of GAV and LAV, we present an alternative point of view, a Target-based Integration Query System (TIQS), that is neither GAV nor LAV. It aims at combining the best of the two basic approaches: GAV's simple query reformulation and LAV's scalability.

The following characteristics describe our solution.

1. Each relation in a target schema, which is our global schema, is predefined and independent of any source schema. Moreover, we wrap sources in isolation, without reference to the global schema.<sup>1</sup> In contrast, in a GAV approach, DBAs revise the global schema to include all items in sources, and in a LAV approach, DBAs adjust the source schemas such that they contain only source relations that can be described by views over the global schema.
2. A set of *mapping elements* in a source-to-target mapping maps a source schema to a target schema. Because we wrap sources independently, source and target schemas use different structures and vocabularies. Automated *schema matching* techniques have been proven to be successful in extracting mapping elements between two schemas. [RB01] surveys these techniques. Clio [MHH00] has an extensive tool set to aid users semi-automatically generate mappings. [XE03a, XE03b, Xu03] provide many mappings automatically, with accuracies ranging from 92%-100%; these mappings are not just 1-1 mappings, but include many indirect mappings discussed later in this paper. Thus, TIQS is capable of specifying views over source schemas that match with elements in the target schema semi-automatically.
3. When a new information source becomes available (changes), a source-to-target mapping must be created (adjusted). With the assistance of semi-automatic mapping tools, the maintenance requires less manual work than either GAV or LAV.

---

<sup>1</sup> Often these sources are structured, and we simply take the local schema without change [ETL02].

4. Whenever a users poses queries in terms of target relations, query reformulation is rule unfolding as in GAV by simply applying the generated source-to-target mappings.
5. If the target schema evolves, the mapping tool semi-automatically generates (or adjusts) mapping elements between the new target schema and the source schemas. This involves less DBA effort than for either GAV or LAV.

TIQS operates in two phases: design and query processing. In the design phase, the system synergistically automates the generation of source-to-target mappings. Mapping elements in source-to-target mappings are expressions over source schema elements that produce *virtual target-view elements*. This leads automatically to a rewriting of every target element as a union of corresponding virtual target-view elements. In the query processing phase, a user poses queries in terms of target relations. Query reformulation thus reduces to rule unfolding by applying the view definition expressions for the target relations in the same way database systems apply view definitions.

TIQS's contributions are (1) a unique approach to data integration using source-to-target mappings based on a predefined target schema that combines the advantages and mitigates the limitations of GAV and LAV, and (2) an extended relational algebra to describe source-to-target mappings, whose implementation is readily available based on schema matching techniques described in [XE03a, XE03b]. We organize the contributions in this paper as follows. Section 2 presents the components of TIQS. Section 3 describes an extended relational algebra for source-to-target mappings. Section 4 discusses the solution to query reformulation and gives theorems to prove that TIQS gives *sound answers* to a query using a *maximally contained reformulation*. Section 5 reviews the other alternatives to GAV and LAV. In Section 6 we summarize and make concluding remarks.

## 2 The Data Integration System

**Definition 1.** A *data integration system*  $I$  is a triple  $(T, \{S_i\}, \{M_i\})$ , where  $T$  is a target schema,  $\{S_i\}$  is a set of  $n$  source schemas, and  $\{M_i\}$  is a set of  $n$  source-to-target mappings, such that for each source schema  $S_i$  there is a mapping  $M_i$  from  $S_i$  to  $T$ ,  $1 \leq i \leq n$ .

We use rooted hypergraphs to represent both target and source schemas in  $I$ . A hypergraph includes a set of object sets  $O$  and a set of relationship sets  $R$ . Therefore, a schema element is either an object set or a relationship set. An object set either has associated data values or has associated object identifiers (OIDs), which we respectively call *lexical* and *non-lexical* object sets. The root node is a designated non-lexical object set of primary interest. Figure 1, for example, shows two schema hypergraphs (whose roots are *house* and *House*). In the hypergraphs, lexical object sets are dotted boxes, non-lexical object sets are solid boxes, functional relationship sets are lines with an arrow from domain object set to range object set, and nonfunctional relationship sets are lines without arrowheads. For a schema  $H$ , which is either a source schema or a target schema, we let  $\Sigma_H$  denote the union of  $O$  and  $R$ . For source views, we let  $V_H$  denote the extension of  $\Sigma_H$  with derived

object and relationship sets over a source  $H$ .

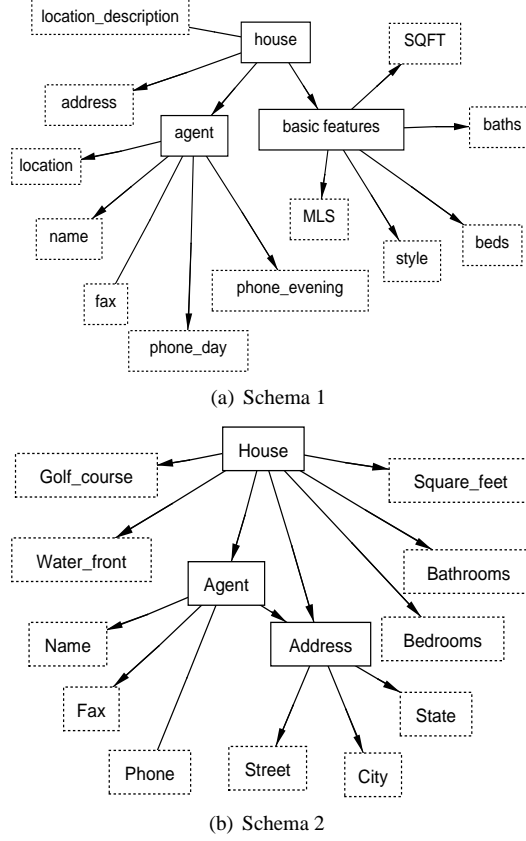


Figure 1: Source Graphs for Schema 1 and Schema 2

A source-to-target mapping  $M_i$  for a source schema  $S_i$  with respect to a target schema  $T$  is a function  $f_i(V_{S_i}) \rightarrow \Sigma_T$ . Intuitively, a source-to-target mapping  $M_i$  represents inter-schema correspondences between a source schema  $S_i$  and a target schema  $T$ . If we let Schema 1 in Figure 1(a) be the target and let Schema 2 in Figure 1(b) be the source, for example, a source-to-target mapping between the two schemas includes a semantic correspondence, which declares that the lexical object set *Bedrooms* in the source semantically corresponds to the lexical object set *beds* in the target. If we let Schema 1 be the source and Schema 2 be the target, a source-to-target mapping declares that the union of the two sets of values in *phone\_day* and *phone\_evening* in the source corresponds to the values for *Phone* in the target.

We represent semantic correspondences between a source schema  $S$  and a target schema  $T$  as a set of mapping elements. A mapping element is either a *direct match* which binds a schema element in  $\Sigma_S$  to a schema element in  $\Sigma_T$ , or an *indirect match* which binds a

virtual schema element in  $V_S$  to a target schema element in  $\Sigma_T$  through an appropriate *mapping expression* over  $\Sigma_S$ . A mapping expression specifies how to derive a virtual schema element through manipulation operations over a source schema. We denote a mapping element as  $(t \sim s \Leftarrow \theta_s(\Sigma_S))$ , where  $\theta_s(\Sigma_S)$  is a mapping expression that derives a source element  $s$  in  $V_S$ ,<sup>2</sup> and  $t$  is a target schema element in  $\Sigma_T$ .

As part of the mapping declarations, TIQS derives a set of *inclusion dependencies* for each target element based on the collected source-to-target mappings. Each mapping element  $\omega$ ,  $(t \sim s \Leftarrow \theta_s(\Sigma_S))$ , implies an inclusion dependency, which we denote as  $S.s \subseteq t$ . This declares that the facts for schema element  $s \in V_S$ , can be loaded into the target as the facts for schema element  $t$ . As is typical for integration systems with non-materialized global schemas, we make an “open world assumption.” Thus, the facts for the source element  $s$  in the mapping element  $\omega$  are only a subset of facts for the target element  $t$ ; and if there exists a source element  $s' \in V_{S'}$  and another mapping element  $\omega'$ ,  $(t \sim s' \Leftarrow \theta_{s'}(\Sigma_{S'}))$ , the facts for both  $s$  and  $s'$  can be facts for  $t$ . In general, for each target schema element  $t \in \Sigma_T$  in the data integration system  $I$ , we denote the set of inclusion dependencies for  $t$  as  $\{S_i.s_j \subseteq t | (t \sim s_j \Leftarrow \theta_{s_j}(\Sigma_{S_i})) \in M_i, s_j \in V_{S_i}, S_i \in I, M_i \in I, T \in I\}$ .

### 3 Algebra for Source-to-Target Mappings

Each object and relationship set (including virtual object and relationship sets) in the target and source schemas are single-attribute or multiple-attribute relations. Thus, relational algebra directly applies to the object and relationship sets in a source or target schema. The standard operations, however, are not enough to capture the operations required to express all the needed source-to-target mappings. Thus, we extend the relational algebra.

To motivate our use of standard and extended operators, we list the following problems we must face in creating virtual object and relationship sets over source schemas.

- *Union and Selection.* The object sets, *phone\_day* and *phone\_evening* in Schema 1 of Figure 1(a) are both subsets of *Phone* values in Schema 2 of Figure 1(b), and the relationship sets *agent – phone\_day* and *agent – phone\_evening* in Schema 1 are both specializations of *Agent – Phone* values in Schema 2. Thus, if Schema 2 is the target, we need the union of the values in *phone\_day* and *phone\_evening* and the union of the relationships in *agent – phone\_day* and *agent – phone\_evening* in Schema 1; and if Schema 1 is the target, we should find a way to separate the day phones from the evening phones and separate the relationships between agents and day phones from those between agents and evening phones.
- *Merged and Split Values.* The object sets, *Street*, *City*, and *State* are separate in Schema 2 and merged as *address of house* or *location of agent* in Schema 1. Thus, we need to split the values if Schema 2 is the target and merge the values if Schema 1 is the target.
- *Object-Set Name as Value.* In Schema 2 the features *Water\_front* and *Golf\_course* are object-set names rather than values. The Boolean values “Yes” and “No” associated with them are not the values but indicate whether the values *Water\_front* and *Golf\_course* should

<sup>2</sup>Note that the mapping expression may be degenerate so that  $(t \sim s)$  is possible.

be included as description values for *location.description* of *house* in Schema 1. Thus, we need to distribute the object-set names as values for *location.description* if Schema 1 is the target and make Boolean values for *Water\_front* and *Golf\_course* based on the values for *location.description* if Schema 2 is the target.

- *Path as Relationship Set.* The path *house – basic features – beds* in Schema 1 semantically corresponds to the relationship set *House – Bedrooms* in Schema 2. Thus, we need to join and project on the path if Schema 2 is the target and make a virtual object set for *basic features* and virtual relationship sets for *house – basic features* and *basic features – beds* over Schema 2 if Schema 1 is the target.

We use the following operations over source relations to resolve these problems<sup>3</sup>.

- *Standard Operators.* Selection  $\sigma$ , Union  $\cup$ , Natural Join  $\bowtie$ , Projection  $\pi$ , and Rename  $\rho$ .
- *Composition  $\lambda$ .* The  $\lambda$  operator has the form  $\lambda_{(A_1, \dots, A_n), A} r$  where each  $A_i$ ,  $1 \leq i \leq n$ , is either an attribute of  $r$  or a string, and  $A$  is a new attribute. Applying this operation forms a new relation  $r'$ , where  $\text{attr}(r') = \text{attr}(r) \cup \{A\}$  and  $|r'| = |r|$ . The value of  $A$  for tuple  $t$  on row  $l$  in  $r'$  is the concatenation, in the order specified, of the strings among the  $A_i$ 's and the string values for attributes among the  $A_i$ 's for tuple  $t'$  on row  $l$  in  $r$ .
- *Decomposition  $\gamma$ .* The  $\gamma$  operator has the form  $\gamma_{A, A'}^R r$  where  $A$  is an attribute of  $r$ , and  $A'$  is a new attribute whose values are obtained from  $A$  values by applying a routine  $R$ . Applying this operation forms a new relation  $r'$ , where  $\text{attr}(r') = \text{attr}(r) \cup \{A'\}$  and  $|r'| = |r|$ . The value of  $A'$  for tuple  $t$  on row  $l$  in  $r'$  is obtained by applying the routine  $R$  on the value of  $A$  for tuple  $t'$  on row  $l$  in  $r$ .
- *Boolean  $\beta$ .* The  $\beta$  operator has the form  $\beta_{A, A'}^{Y, N} r$ , where  $Y$  and  $N$  are two constants representing *Yes* and *No* values in  $r$ ,  $A$  is an attribute of  $r$  that has only  $Y$  or  $N$  values, and  $A'$  is a new attribute. Applying this operation forms a new relation  $r'$ , where  $\text{attr}(r') = (\text{attr}(r) - \{A\}) \cup \{A'\}$  and  $|r'| = |\sigma_{A=Y} r|$ . The value of  $A'$  for tuple  $t$  in  $r'$  is the literal string  $A$  if and only if there exists a tuple  $t'$  in  $r$  such that  $t'[\text{attr}(r) \cap \text{attr}(r')] = t[\text{attr}(r) \cap \text{attr}(r')]$  and  $t'[A]$  is a  $Y$  value.
- *DeBoolean  $\mathfrak{B}$ .* The  $\mathfrak{B}$  operator has the form  $\mathfrak{B}_{A, A'}^{Y, N} r$ , where  $Y$  and  $N$  are two constants representing *Yes* and *No* values,  $A$  is an attribute of  $r$ , and  $A'$  is a new attribute. Applying this operation forms a new relation  $r'$ , where  $\text{attr}(r') = (\text{attr}(r) - \{A\}) \cup \{A'\}$  and  $|r'| = |\pi_{\text{attr}(r) \cap \text{attr}(r')} r|$ . The value of  $A'$  for tuple  $t$  in  $r'$  is  $Y$  if and only if there exists a tuple  $t'$  in  $r$  such that  $t'[\text{attr}(r) \cap \text{attr}(r')] = t[\text{attr}(r) \cap \text{attr}(r')]$  and  $t'[A]$  is the literal string  $A'$ , or is  $N$  if and only if there does not exist any tuple  $t'$  in  $r$  such that  $t'[\text{attr}(r) \cap \text{attr}(r')] = t[\text{attr}(r) \cap \text{attr}(r')]$  and  $t'[A]$  is the literal string  $A'$ .
- *Skolemization  $\varphi$ .* The  $\varphi$  operator has the form  $\varphi_{f_A}(r)$ , where  $f_A$  is a skolem function, and  $A$  is a new attribute. Applying this operation forms a new relation  $r'$ , where  $\text{attr}(r') = \text{attr}(r) \cup \{A\}$  and  $|r'| = |r|$ . The value of  $A$  for tuple  $t$  on line  $l$  in  $r'$  is a functional term that computes a value by applying the skolem function  $f_A$  over tuple  $t'$  on line  $l$  in  $r$ .<sup>4</sup>

As an example, let Schema 1 in Figure 1 be a target schema  $T$ , and let Schema 2 be a source schema  $S$ . Figure 2 shows the derivation over the source schema and the source elements in the source-to-target mapping. The shaded boxes denote virtual object sets, and

<sup>3</sup>In the notation, a relation  $r$  has a set of attributes, which corresponds to the names of lexical or non-lexical object sets;  $\text{attr}(r)$  denotes the set of attributes in  $r$ ; and  $|r|$  denotes the number of tuples in  $r$ .

<sup>4</sup>When applying *Skolemization* operations, we introduce functional terms based only on tuple values that do not contain functional terms. This leads to a finite evaluation.

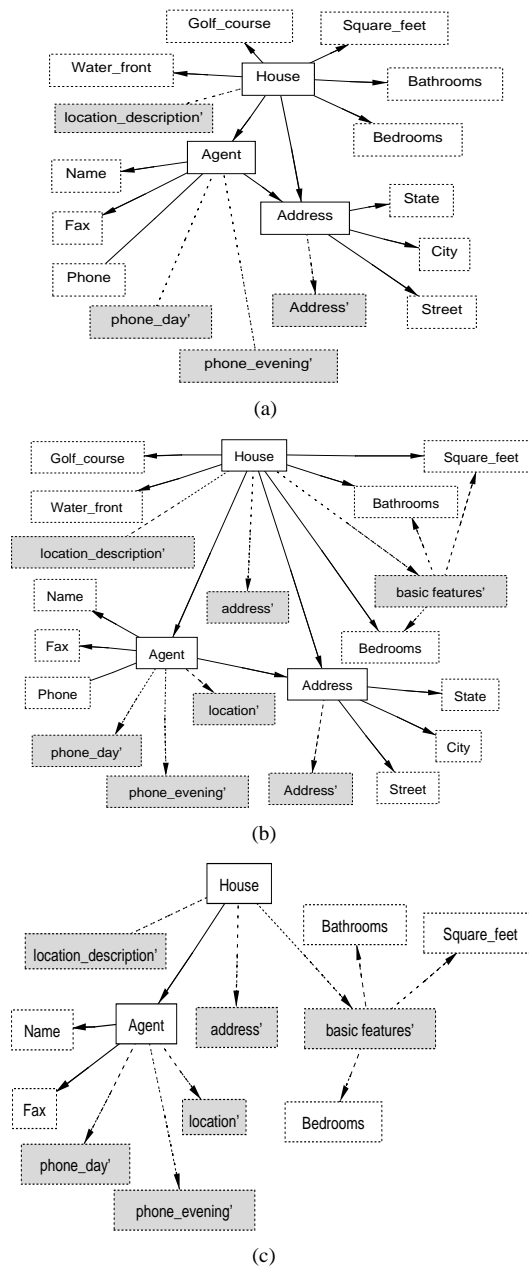


Figure 2: Derivation of Virtual Object and Relationship Sets from Schema 2 for Schema 1

the dashed lines denote virtual relationship sets. There are two main steps in the derivation (see [XE03a, XE03b, Xu03] for details).

**Step 1:** Use instance-level information to derive virtual object and relationship sets. The implemented matching system applies expected-data-value techniques [XE03b] to derive virtual object and relationship sets. Figure 2(a) shows the virtual object and relationship sets derived after applying the following instance-level transformations.

- Derivation of  $location\_description'$  and  $House - location\_description'$ .

$$\begin{aligned} House - location\_description' &\Leftarrow \\ &\rho_{Golf\_course' \leftarrow location\_description'} \beta_{Golf\_course, Golf\_course'}^{Yes, No} (House - Golf\_course) \\ &\cup \rho_{Water\_front' \leftarrow location\_description'} \beta_{Water\_front, Water\_front'}^{Yes, No} (House - Water\_front) \\ location\_description' &\Leftarrow \pi_{location\_description'} (House - location\_description') \end{aligned}$$

- Derivation of  $Address'$  and  $Address - Address'$ .

$$\begin{aligned} Address - Address' &\Leftarrow \pi_{Address, Address'} \lambda(Street, ", ", City, ", ", State), Address' ( \\ &\quad Address - Street \bowtie Address - City \bowtie Address - State) \\ Address' &\Leftarrow \pi_{Address'} (Address - Address') \end{aligned}$$

- Derivation of  $phone\_day'$ ,  $Agent - phone\_day'$ ,  $phone\_evening'$ , and  $Agent - phone\_evening'$ .<sup>5</sup>

$$\begin{aligned} Agent - phone\_day' &\Leftarrow \rho_{Phone \leftarrow phone\_day'} \sigma_{KEYWORD(day)} (Agent - Phone) \\ phone\_day' &\Leftarrow \pi_{phone\_day'} (Agent - phone\_day') \\ Agent - phone\_evening' &\Leftarrow \rho_{Phone \leftarrow phone\_evening'} \sigma_{KEYWORD(evening)} (Agent - Phone) \\ phone\_evening' &\Leftarrow \pi_{phone\_evening'} (Agent - phone\_evening') \end{aligned}$$

**Step 2:** Use schema-level information to derive virtual object and relationship sets. The matching techniques apply source and target schema structural characteristics to derive virtual object and relationship sets. Figure 2(b) shows the object and relationship sets in  $V_S$  after applying the following schema-level transformations.

- Derivation of  $Agent - location'$ ,  $location'$ ,  $House - address'$ , and  $address'$ .

$$\begin{aligned} House - address' &\Leftarrow \rho_{Address' \leftarrow address'} \pi_{House, Address'} (House - Address \\ &\quad \bowtie Address - Address') \\ Agent - location' &\Leftarrow \rho_{Address' \leftarrow location'} \pi_{Agent, Address'} (Agent - Address \\ &\quad \bowtie Address - Address') \\ address' &\Leftarrow \pi_{address'} (House - address') \\ location' &\Leftarrow \pi_{location'} (Agent - location') \end{aligned}$$

- Derivation of  $basic\_features'$ ,  $House - basic\_features'$ ,  $basic\_features' - Square\_feet$ ,  $basic\_features' - Bedrooms$ , and  $basic\_features' - Bathrooms$ .<sup>6</sup>

<sup>5</sup>We may be able to recognize keywords such as *day-time*, *day*, *work phone*, *evening*, or *home* associated with each listed phone in the source. If so, we can apply the selection operator to sort out which phones belong in which set (if not, a human expert may not be able to sort these out either). We implement the *KEYWORD* predicate by applying data-extraction techniques described in [ECJ<sup>+</sup>99].

<sup>6</sup>When applying the Skolemization operator to derive the virtual object set *basic\\_features'*, the system makes *basic\\_features'* functionally dependent on *House* to match the functional dependency between *basic\\_features* and *house* in the target schema.



$$\begin{aligned}
House - basic\_features' &\Leftarrow \varphi_{f_{basic\_features'}}(House) \\
basic\_features' - Bathrooms &\Leftarrow \pi_{basic\_features', Bathrooms}(House - basic\_features' \\
&\quad \bowtie House - Bathrooms) \\
basic\_features' - Bedrooms &\Leftarrow \pi_{basic\_features', Bedrooms}(House - basic\_features' \\
&\quad \bowtie House - Bedrooms) \\
basic\_features' - Square\_feet &\Leftarrow \pi_{basic\_features', Square\_feet}(House - basic\_features' \\
&\quad \bowtie House - Square\_feet)
\end{aligned}$$

- Specializations of  $Agent - phone\_day'$  and  $Agent - phone\_evening'$ .<sup>7</sup>

$$\begin{aligned}
Agent - phone\_day' &\Leftarrow \sigma_{COMPATIBLE(agent-phone\_day)}(Agent - phone\_day') \\
Agent - phone\_evening' &\Leftarrow \sigma_{COMPATIBLE(agent-phone\_evening)}(Agent - phone\_evening')
\end{aligned}$$

At this point, the object and relationship sets in Figure 2(c) correspond exactly to the source elements in the mapping elements between  $T$  and  $S$ . For example,  $(house \sim House)$ ,  $(address \sim address')$ ,  $(house - address \sim House - address')$ , and so forth.

## 4 Query Reformulation

The data integration system  $I$  collects the information in the design phase. In the query-processing phase, the system reformulates user queries in polynomial time.

To specify the semantics of  $I$ , we start with a valid interpretation  $D_{S_i}$  of a source schema  $S_i \in I$ ,  $1 \leq i \leq n$ . For an interpretation of a schema  $H$  to be *valid*, each tuple in  $D_H$  must satisfy the constraints specified for  $H$ . In our running example, assume we have a valid interpretation for Schema 2 in Figure 1. A target interpretation  $D_{S_i T}$  with respect to  $D_{S_i}$  in  $I$  (1) is a valid interpretation of  $T$ , and (2) satisfies the mapping  $M_i$  between  $S_i$  and  $T$  with respect to  $D_{S_i}$ . Assume that the mapping function for  $M_i$  is  $f_i$ . If  $f_i$  matches  $s_k$  with  $t_j$ ,  $c$  is a tuple for  $t_j$  in  $D_{S_i T}$  if and only if  $c$  is a tuple for  $s_k$  derived through applying the mapping expression  $\theta_{s_k}(\Sigma_{S_i})$  over  $D_{S_i}$ . The semantics of  $I$ , denoted as  $sem(I)$ , are defined as follows:  $sem(I) = \{D_{S_i T} \mid D_{S_i T} \text{ is a target interpretation with respect to } D_{S_i}, S_i \in I\}$ . We are able to prove that if a source has a valid interpretation, then we can load data from the source into the target such that the part of the target populated from the source will necessarily have a valid interpretation [BE03].<sup>8</sup>

Assume that a query language used to express user queries is relational algebra. Here, the queries are Select-Project-Join queries over elements in  $\Sigma_T$ . Let  $q$  be a user query and  $q_I$  denote the result of evaluating  $q$  on  $sem(I)$ .

When evaluating query answers  $q_I$  for  $q$ , the data integration system transparently reformulates  $q$  as  $q^{Ext}$ , a query over the source schemas in  $I$ . Let a query  $q$  be  $\pi_{(\bar{X})} \sigma_P(r_1 \bowtie$

<sup>7</sup>The system specializes the relationship sets in the source so that they are compatible with the functional dependencies in the corresponding relationship sets in the target. The predicate *COMPATIBLE* defaults to the first one or allows a user to decide how the selection should work. See [BE03] for a full explanation about source-target constraint incompatibilities.

<sup>8</sup>The theorem in [BE03] is for individual sources. When sources share objects, both the object-identification problem and the data-merge problem need a resolution. (Note that neither this paper nor other papers that focus on GAV/LAV resolve these problems. The focus of GAV/LAV is on mediation, mappings, and query reformulation.)

$r_2 \bowtie \dots \bowtie r_N$ ), where for  $1 \leq i \leq N$ ,  $\text{attr}(q) = \overline{X}$ ,  $\text{attr}(r_i) = \overline{X_i} \cup \overline{Y_i}$ ,  $\overline{X_i} \subseteq \overline{X}$ ,  $\overline{Y_i} \cap \overline{X} = \emptyset$ ,  $Y = \bigcup_{i=1}^N (\overline{Y_i})$ , and  $P$  is a predicate over  $\overline{X} \cup \overline{Y}$ . The data integration system reformulates  $q$  on  $I$  to obtain  $q^{Ext}$  based on inclusion dependencies collected for each target element in the design phase. Since  $q$  is in terms of elements in  $\Sigma_T$ , each target relation  $r_i$  in  $q$  corresponds to a set of inclusion dependencies  $ID_i$ ,  $1 \leq i \leq N$ , collected in the design phase. Each member in  $ID_i$  has the form  $S_j.e_S \subseteq r_i$ , where  $e_S \in V_{S_j}$ ,  $1 \leq j \leq n$ , and  $n$  is the number of sources. Then, to obtain  $q^{Ext}$ , we substitute each  $r_i$  in  $q$  by  $\bigcup_{(S_j.e_S \subseteq r_i) \in ID_i} (S_j.e_S)$ . Note that a source element  $e_S$  may be virtual, derived by applying the mapping expression  $\theta_{e_S}(\Sigma_{S_j})$ .<sup>9</sup> Thus, when sending a sub-query decomposed from  $q^{Ext}$  to the information source  $S_j$ , the system also sends the mapping expression  $\theta_{e_S}(\Sigma_{S_j})$  such that the source  $S_j$  correctly derives source facts for  $r_i$  in the target.

With query reformulation in place, we can now prove that query answers are *sound*—every answer to a user query is a fact according to the source(s)—and that query answers contain all the facts the sources have to offer—*maximal* for the query reformulation.

**Theorem 1.** *Let  $I = (T, \{S_i\}, \{M_i\})$  be a data integration system. Let  $D = \{D_{S_i} | S_i \in I\}$  be the set of valid interpretations of source schemas in  $I$  and let  $q_D^{Ext}$  be the query answers obtained by evaluating  $q^{Ext}$  over  $D$ . Given a user query  $q$  in terms of target relations, a tuple  $\langle a_1, a_2, \dots, a_M \rangle$  is a sound answer in  $q_I$  for  $q$ .*

*Proof.* (See Appendix A.)

**Theorem 2.** *Let  $I = (T, \{S_i\}, \{M_i\})$  be a data integration system. If  $q^{Ext}$  is a reformulated query in  $I$  for a query  $q$  in terms of target relations,  $q^{Ext}$  is a maximally contained reformulation for  $q$  with respect to  $I$ .*

*Proof.* (See Appendix B.)

## 5 Related Work—Other Alternatives to GAV and LAV

[FLM99] proposed a *Global-Local-as-View* (GLAV) approach, which combines expressive powers of both LAV and GAV. In a GLAV approach, the independence of a global schema, the maintenance to accommodate new sources, and the complexity to reformulate queries are the same as in LAV. However, instead using a restricted form of first-order logical sentences as in LAV and GAV to define view definitions, GLAV uses flexible first-order sentences such that it allows a view over source relations to be a view over global relations in source descriptions. Thus, GLAV can derive data using views over source relations, which is beyond the expressive ability of LAV, and it allows conjunctions of global relations, which is beyond the expressive ability of GAV. Our solution, TIQS, also has the ability to derive views over source schemas. The sets of view-creation operators, however, are incompatible—in TIQS we do not have a recursive operator, and GLAV has nothing comparable to merge/split or Boolean operators. Moreover, GLAV claims no ability to semi-automate the specification of source descriptions.

<sup>9</sup>We keep non-lexical objects in different sources separate by consistently introducing new OIDs for target objects.

[CCGL02] proposed a translation algorithm to turn LAV into GAV such that it can keep LAV's scalability and obtain GAV's simple query reformulation. The translation results in a logic program that can be used to answer queries using rule unfolding. However, even though the translation to obtain the logic program is in polynomial time, the evaluation of the logic program could produce an exponential number of facts because of recomputing source relations over all source data. In contrast, TIQS encapsulates views for source relations in mapping elements. Since the view definitions are immediately available, query processing in TIQS has better query performance than the translation approach. As in [FLM99], [CCGL02] claims no ability to semi-automate the specification of source descriptions.

## 6 Conclusion

This paper describes TIQS, an approach to data integration based on a predefined target schema, which combines the advantages and avoids the limitations of both GAV and LAV. This solution has polynomial-time query reformulation and is scalable for large applications. DBAs create the target schema and wrap source schemas independently, so that neither the target schema nor the source schemas are contingent respectively on the source schemas or the target schema. Moreover, we have an implementation that either creates or helps create the needed mappings. Thus, TIQS increases both scalability and usability over previously proposed approaches.

## References

- [BE03] J. Biskup and D.W. Embley. Extracting information from heterogeneous information sources using ontologically specified target views. *Information Systems*, 28(3):169–212, May 2003.
- [CCGL02] A. Cali, D. Calvanese, G. De Giacomo, and M. Lenzerini. On the expressive power of data integration systems. In *Proceedings of 21st International Conference on Conceptual Modeling (ER2002)*, pages 338–350, Tampere, Finland, October 7–11 2002.
- [CGMH<sup>+</sup>94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of the 10th Meeting of the Information Processing Society of Japan*, pages 7–18, Tokyo, Japan, October 7–18 1994.
- [CLL01] D. Calvanese, D. Lembo, and M. Lenzerini. Survey on methods for query rewriting and query answering using views. Technical report, University of Rome, Roma, Italy, April 2001.
- [ECJ<sup>+</sup>99] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record Web pages. *Data & Knowledge Engineering*, 31(3):227–251, November 1999.
- [ETL02] D.W. Embley, C. Tao, and S.W. Liddle. Automatically extracting ontologically specified data from HTML tables with unknown structure. In *Proceedings of the 21st*

- International Conference on Conceptual Modeling (ER2002)*, pages 322–337, Tampere, Finland, October 7–11 2002.
- [FLM99] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI'99)*, pages 67–73, Orlando, Florida, July 18–22 1999.
  - [GKD97] M.R. Genesereth, A.M. Keller, and O.M. Duschka. Infomaster: An information integration system. In *Proceedings of 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD'97)*, pages 539–542, Tucson, Arizona, May 13–15 1997.
  - [Hal01] A.Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, December 2001.
  - [LRO96] A.Y. Levy, A. Rajaraman, and J.J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB'96)*, pages 251–262, Mumbai (Bombay), India, September 3–6 1996.
  - [MHH00] R. Miller, L. Haas, and M.A. Hernandez. Schema mapping as query discovery. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, pages 77–88, Cairo, Egypt, September 10–14 2000.
  - [RB01] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, December 2001.
  - [Ull97] J.D. Ullman. Information integration using logical views. In F.N. Afrati and P. Kolaitis, editors, *Proceedings of the 6th International Conference on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40, Delphi, Greece, January 8–10 1997.
  - [XE03a] L. Xu and D.W. Embley. Discovering direct and indirect matches for schema elements. In *Proceedings of the 8th International Conference on Database Systems for Advanced Applications (DASFAA 2003)*, pages 39–46, Kyoto, Japan, March 26–28 2003.
  - [XE03b] L. Xu and D.W. Embley. Using domain ontologies to discover direct and indirect matches for schema elements. In *Proceedings of the semantic integration workshop collocated with the second international Semantic Web Conference (ISWC-03)*, pages 105–110, Sanibel Island, Florida, October 2003.
  - [Xu03] Li Xu. *Source Discovery and Schema Mapping for Data Integration*. PhD thesis, Department of Computer Science, Brigham Young University, Provo, Utah, August 2003.

## A Theorem 1.

Let  $I = (T, \{S_i\}, \{M_i\})$  be a data integration system. Let  $D = \{D_{S_i} | S_i \in I\}$  be the set of valid interpretations of source schemas in  $I$  and let  $q_D^{Ext}$  be the query answers obtained by evaluating  $q^{Ext}$  over  $D$ . Given a user query  $q$  in terms of target relations, a tuple  $\langle a_1, a_2, \dots, a_M \rangle$  in  $q_D^{Ext}$  is a sound answer in  $q_I$  for  $q$ .

*Proof (sketch).* Let a query  $q$  be  $\pi_{(\overline{X})} \sigma_P(r_1 \bowtie r_2 \bowtie \dots \bowtie r_N)$ , where for  $1 \leq i \leq N$ ,  $r_i$  is a target relation,  $attr(q) = \overline{X}$ ,  $attr(r_i) = \overline{X}_i \cup \overline{Y}_i$ ,  $\overline{X}_i \subseteq \overline{X}$ ,  $\overline{Y}_i \cap \overline{X} = \emptyset$ ,  $Y = \bigcup_{i=1}^N (\overline{Y}_i)$ , and  $P$  is a predicate over  $\overline{X} \cup \overline{Y}$ . Assume that a tuple  $a = \langle a_1, a_2, \dots, a_M \rangle$  is a tuple in  $q_D^{Ext}$  and  $a$  is not a tuple in  $q_I$ . In  $I$ , the query reformulation procedure translates  $q$  into  $q^{Ext}$  as  $\pi_{(\overline{X})} \sigma_P(\bigcup_{(S_j.e_S \subseteq r_1) \in ID_1} (S_j.e_S) \bowtie \bigcup_{(S_j.e_S \subseteq r_2) \in ID_2} (S_j.e_S) \bowtie \dots \bowtie \bigcup_{(S_j.e_S \subseteq r_N) \in ID_N} (S_j.e_S))$ , where for  $1 \leq i \leq N$ ,  $ID_i$  is a set of inclusion dependencies collected for  $r_i$  in the design phase of  $I$ , and for  $1 \leq j \leq n$ ,  $S_j$  is a source schema collected from one of the  $n$  sources in  $I$  and  $e_S$  in  $S_j.e_S$  is a source element in  $V_{S_j}$ . Thus, since  $a \in q_D^{Ext}$ , there must exist  $N$  source relations,  $s_1, s_2, \dots$ , and  $s_N$ , and  $N$  tuples,  $c_1, c_2, \dots$ , and  $c_N$ , such that  $c_i \in s_i$  and  $a = \pi_{(\overline{X})} \sigma_P(c_1 \bowtie \dots \bowtie c_N)$ , where  $s_i \in V_{S_j}$ ,  $S_j.s_i \subseteq r_i$ ,  $1 \leq i \leq N$ , and  $S_j \in I$ . Since  $S_j.s_i \subseteq r_i$  in  $ID_i$ , based on the derivation of an inclusion dependency, there must exist a mapping element  $(r_i \sim s_i \Leftarrow \theta_{s_i}(\Sigma_{S_j})) \in M_j$ , where  $M_j$  is a source-to-target mapping between  $S_j$  and  $T$  in  $I$ . Since  $c_i \in s_i$  and  $(r_i \sim s_i \Leftarrow \theta_{s_i}(\Sigma_{S_j})) \in M_j$ , based on the semantics of a mapping element,  $c_i \in r_i$  and the tuple  $c_i$  is derived from  $D_{S_j}$  by evaluating  $\theta_{s_i}(\Sigma_{S_j})$ . Since  $c_i \in r_i$  and  $(r_i \sim s_i \Leftarrow \theta_{s_i}(\Sigma_{S_j})) \in M_j$  and  $c_i$  is derived by evaluating  $\theta_{s_i}(\Sigma_{S_j})$  on  $D_{S_j}$ , based on the definition of a target interpretation with respect to  $D_{S_j}$ ,  $c_i \in D_{S_j}T$ . Since  $c_i \in D_{S_j}T$ , based on the definition of  $sem(I)$ ,  $c_i \in sem(I)$ . Since  $c_i \in sem(I)$ ,  $a = \pi_{(\overline{X})} \sigma_P(c_1 \bowtie \dots \bowtie c_N)$ , and  $c_i \in r_i$ ,  $1 \leq i \leq N$ , therefore  $a \in q_I$ . This is contrary to the assumption that  $a$  is not a tuple in  $q_I$ .

## B Theorem 2.

Let  $I = (T, \{S_i\}, \{M_i\})$  be a data integration system. If  $q^{Ext}$  is a reformulated query in  $I$  for a query  $q$  in terms of target relations,  $q^{Ext}$  is a maximally contained reformulation for  $q$  with respect to  $I$ .

*Proof (sketch).* Let a query  $q$  be  $\pi_{(\overline{X})}\sigma_P(r_1 \bowtie r_2 \bowtie \dots \bowtie r_N)$ , where for  $1 \leq i \leq N$ ,  $r_i$  is a target relation,  $attr(q) = \overline{X}$ ,  $attr(r_i) = \overline{X}_i \cup \overline{Y}_i$ ,  $\overline{X}_i \subseteq \overline{X}$ ,  $\overline{Y}_i \cap \overline{X} = \emptyset$ ,  $Y = \cup_{i=1}^N (\overline{Y}_i)$ , and  $P$  is a predicate over  $\overline{X} \cup \overline{Y}$ . Assume that a tuple  $a = \langle a_1, a_2, \dots, a_M \rangle$  is a tuple in  $q_I$  and  $a$  is not a tuple in  $q_D^{Ext}$ . Since  $a$  is a tuple in  $q_I$ , there must exist at least  $N$  tuples  $c_1, c_2, \dots, c_N$  in  $sem(I)$  such that  $c_i \in r_i$ ,  $1 \leq i \leq N$ , and  $a = \pi_{(\overline{X})}\sigma_P(c_1 \bowtie c_2 \bowtie \dots \bowtie c_N)$ . Therefore, since  $c_i \in sem(I)$ ,  $1 \leq i \leq N$ , based on the definition of  $sem(I)$ , a target interpretation  $D_{S_j T}$  with respect to  $D_{S_j}$  must exist such that  $c_i \in D_{S_j T}$ , where  $S_j \in I$  and  $D_{S_j}$  is a valid interpretation of  $S_j$  and  $T \in I$ . Since  $c_i \in D_{S_j T}$  and  $c_i \in r_i$ ,  $1 \leq i \leq N$ , based on the definition of  $D_{S_j T}$ , there must exist a mapping element  $(r_i \sim s_i \Leftarrow \theta_{s_i}(\Sigma_{S_j})) \in M_j$ , where  $M_j \in I$  and  $M_j$  is a source-to-target mapping between  $S_j$  and  $T$ . Since  $(r_i \sim s_i \Leftarrow \theta_{s_i}(\Sigma_{S_j}))$  and  $c_i \in r_i$ ,  $1 \leq i \leq N$ , based on the semantics of a mapping element,  $c_i \in s_i$  and  $c_i$  is derived by evaluating the mapping expression  $\theta_{s_i}(\Sigma_{S_j})$  over  $D_j$ . Moreover, since  $(r_i \sim s_i \Leftarrow \theta_{s_i}(\Sigma_{S_j}))$ ,  $1 \leq i \leq N$ , there must exist an inclusion dependency  $(S_j.s_i \subseteq r_i) \in ID_i$ , where  $ID_i$  is the set of inclusion dependencies collected for  $r_i$  in the design phase of  $I$ . Therefore, when the query reformulation procedure translates  $q$  into  $q^{Ext}$ ,  $S_j.s_i$  is a member in the union set that replaces  $r_i$  in  $q$ ,  $1 \leq i \leq N$ , and  $S_j \in I$ . Thus, the query answer to  $\pi_{(\overline{X})}\sigma_P(s_1 \bowtie \dots \bowtie s_N)$  over  $D$  is a subset of  $q_D^{Ext}$ . When evaluating  $\pi_{(\overline{X})}\sigma_P(s_1 \bowtie \dots \bowtie s_N)$ , since  $a = \pi_{(\overline{X})}\sigma_P(c_1 \bowtie \dots \bowtie c_N)$  and  $c_i \in s_i$  and  $c_i$  is derived by applying the mapping expression  $\theta_{s_i}(\Sigma_{S_j})$  over  $D_j$ , where  $S_j \in I$  and  $D_j \in D$ ,  $1 \leq i \leq N$ , therefore  $a$  is a tuple of the query answer to  $\pi_{(\overline{X})}\sigma_P(s_1 \bowtie \dots \bowtie s_N)$  over  $D$ . Since the query answer to  $\pi_{(\overline{X})}\sigma_P(s_1 \bowtie \dots \bowtie s_N)$  over  $D$  is a subset of  $q_D^{Ext}$  and  $a$  is a tuple of query answer to  $\pi_{(\overline{X})}\sigma_P(s_1 \bowtie \dots \bowtie s_N)$  over  $D$ ,  $a \in q_D^{Ext}$ . This is contrary to the assumption that  $a$  is not a tuple in  $q_D^{Ext}$ .