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

Li Xu* and David W. Embley*
Department of Computer Science
Brigham Young University
Provo, Utah 84602, U.S.A.
{lx, 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 BGLaV 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+94] and Local-as-View(LaV) [LRO96, GKD97].[1] [CLL01] surveys the most important query processing algorithms proposed in the literature for LaV, and describes the principle GaV data integration systems 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

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

[1] Although "GAV" and "LAV" are common abbreviations [CCGL02, Ull97], we prefer "GaV" and "LaV" because they better match the phrases to which they refer.

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, called BGLaV,[2] 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.[3] 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. [XE02] provides 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. (See Appendix A.) Thus, BGLaV 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.

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.

BGLaV 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*

[2] "BGLaV" is an acrynom for "BYU-Global-Local-as-View."

[3] Often these sources are structured, and we simply take the local schema without change [ETL02].

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.

BGLaV'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 [XE02]. We organize the contributions in this paper as follows. Section 2 presents the components of BGLaV. 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 BGLaV gives *certain answers* to a query using a *maximally contained reformulation*.[4] Section 5 reviews the other alternatives to GaV and LaV. In Section 6 we summarize and make concluding remarks.

2 The Data Integration System

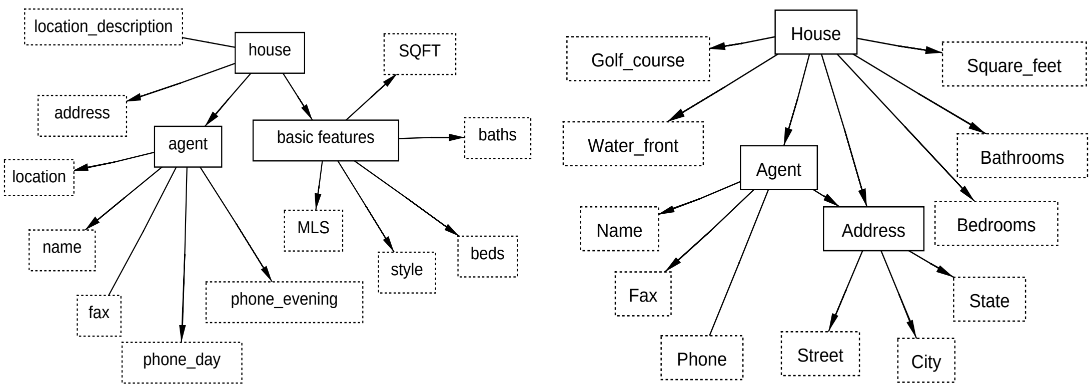
Definition 1. A *data integration system* I is a triple (T, $\{S_i\}$, $\{M_i\}$), where T is a target schema, $\{S_i\}$ is a set of n source schemas, and $\{M_i\}$ is a set of n source-to-target mappings, such that for each source schema S_i there is a mapping M_i from S_i to T, $1 \leq i \leq n$.

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 Σ_H denote the union of O and R. For source views, we let V_H denote the extension of Σ_H with derived object and relationship sets over a source H.

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 Σ_S to a schema element in Σ_T, or an *indirect match* which binds a virtual schema element in V_S to a target schema element in Σ_T through an appropriate *mapping expression* over Σ_S. A mapping expression specifies how to derive a virtual schema element through manipulation operations over

[4]The two terms are different from the terms *certain answers* and *maximally contained rewriting* in [Hal01] because [Hal01] uses the terms for query processing over materialized views, whereas we use them for non-materialized views.



(a) Schema 1

(b) Schema 2

Figure 1: Source Graphs for Schema 1 and Schema 2

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,[5] and t is a target schema element in Σ_T.

As part of the mapping declarations, BGLaV derives a set of *inclusion dependencies* for each target element based on the collected source-to-target mappings. Each mapping element ω, $(t \sim s \Leftarrow \theta_s(\Sigma_S))$, 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 ω 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 ω', $(t \sim s' \Leftarrow \theta_{s'}(\Sigma_{S'}))$, 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.

[5] Note that the mapping expression may be degenerate so that $(t \sim s)$ is possible.

- *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 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.

Currently, we use the following operations over source relations to resolve these problems[6]. (See Appendix B for examples that illustrate how the new operators work.)

- *Standard Operators. Selection* σ, *Union* $\cup$, *Natural Join* $\bowtie$, *Projection* π, and *Rename* ρ.

- *Composition* λ. The λ operator has the form $\lambda_{(A_1,\ldots,A_n),A}r$ where each A_i, $1 \le i \le n$, is either an attribute of r or a string, and A is a new attribute. Applying this operation forms a new relation r', where $attr(r') = attr(r) \cup \{A\}$ 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* γ. The γ operator has the form $\gamma^R_{A,A'}r$ 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 $attr(r') = attr(r) \cup \{A'\}$ 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* β. The β operator has the form $\beta^{Y,N}_{A,A'}r$, 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 $attr(r') = (attr(r) - \{A\}) \cup \{A'\}$ 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 $t'[attr(r) \cap attr(r')] = t[attr(r) \cap attr(r')]$ and $t'[A]$ is a Y value.

- *DeBoolean* $\reflectbox{$\beta$}$. The $\reflectbox{$\beta$}$ operator has the form $\reflectbox{$\beta$}^{Y,N}_{A,A'}r$, 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 $attr(r') = (attr(r) - \{A\}) \cup \{A'\}$ and $|r'| = |\pi_{attr(r) \cap attr(r')}r|$. The value of A' for tuple t in r' is Y if and only if there exists a tuple t' in r such that $t'[attr(r) \cap attr(r')] = t[attr(r) \cap attr(r')]$ 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 $t'[attr(r) \cap attr(r')] = t[attr(r) \cap attr(r')]$ and $t'[A]$ is the literal string A'.

- *Skolemization* φ. The φ 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 $attr(r') = attr(r) \cup \{A\}$ 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.[7]

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 the dashed lines denote virtual relationship sets. There are two main steps in the derivation (see [EJX01, XE02] for details).

[6] In the notation, a relation r has a set of attributes, which corresponds to the names of lexical or non-lexical object sets; $attr(r)$ denotes the set of attributes in r; and $|r|$ denotes the number of tuples in r.

[7] 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.

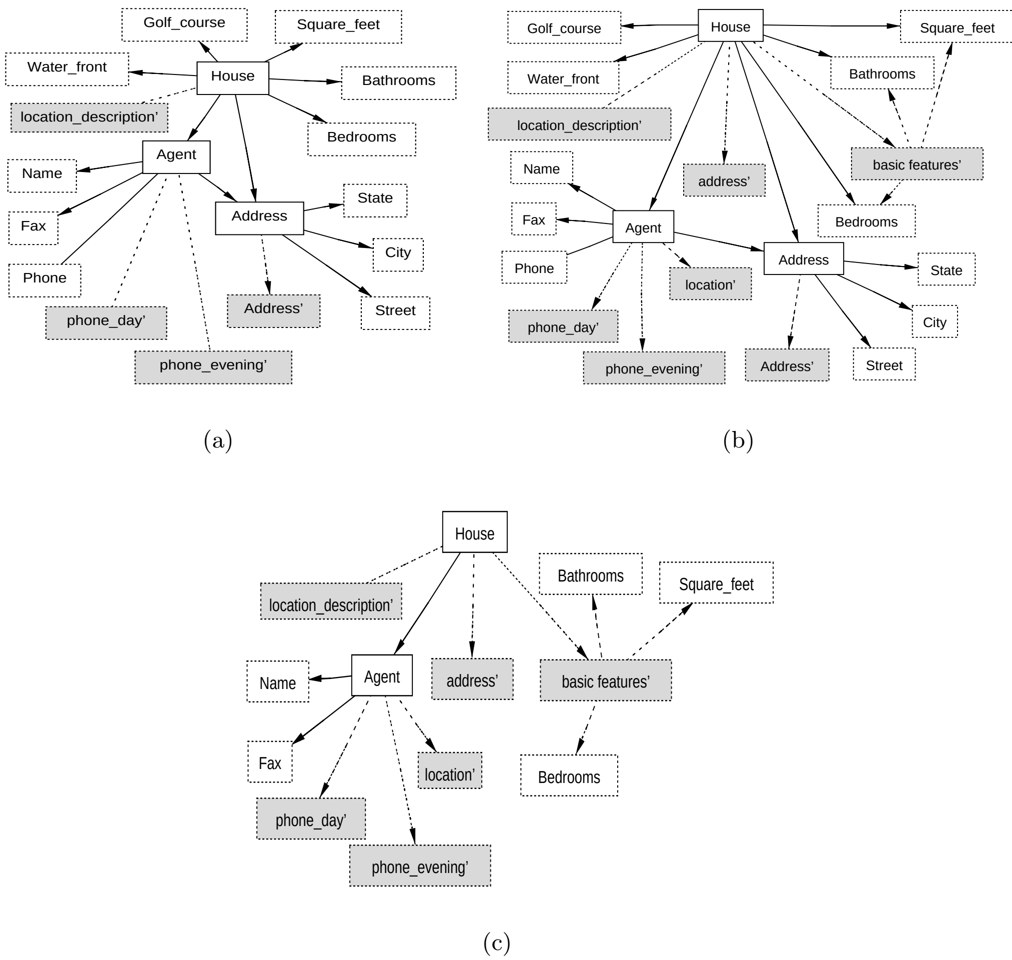


Figure 2: Derivation of Virtual Object and Relationship Sets from Schema 2 for Schema 1

Step 1: *Use instance-level information to derive virtual object and relationship sets.* The implemented matching system applies expected-data-value techniques [EJX01] 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 $House - location\_description'$.

$$House - location\_description' \Leftarrow \rho_{Golf\_course' \leftarrow location\_description'} \beta^{\text{“}Yes\text{”}, \text{“}No\text{”}}_{Golf\_course, Golf\_course'}(House - Golf\_course) \cup$$
$$\rho_{Water\_front' \leftarrow location\_description'} \beta^{\text{“}Yes\text{”}, \text{“}No\text{”}}_{Water\_front, Water\_front'}(House - Water\_front)$$
$$location\_description' \Leftarrow \pi_{location\_description'}(House - location\_description')$$

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

$$Address - Address' \Leftarrow \pi_{Address, Address'} \lambda_{(Street, \text{“}, \text{”}, City, \text{“}, \text{”}, State), Address'}(Address - Street$$
$$\bowtie Address - City \bowtie Address - State)$$
$$Address' \Leftarrow \pi_{Address'}(Address - Address')$$

- Derivation of $phone\_day'$, $Agent-phone\_day'$, $phone\_evening'$, and $Agent-phone\_evening'$.[8]

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

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 $address'$.

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

- Derivation of $basic\ features'$, $House - basic\ features'$, $basic\ features' - Square\_feet$, $basic\ features' - Bedrooms$, and $basic\ features' - Bathrooms$.[9]

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

- Specializations of $Agent - phone\_day'$ and $Agent - phone\_evening'$.[10]

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

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.

[8] 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+99].

[9] 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.

[10] 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.

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_iT} 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_iT} 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: $sem(I) = \{D_{S_iT} \mid D_{S_iT}$ 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].[11]

Assume that a query language used to express user queries is relational algebra. Here, the queries are Select-Project-Join queries over elements in Σ_T. Let q be a user query and q_I denote the result of evaluating q on $sem(I)$. We formalize q_I using the notion of *certain answers* for q.

When evaluating certain 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 $\pi_{(\overline{X})}\sigma_P(r_1 \bowtie r_2 \bowtie \ldots \bowtie r_N)$, where for $1 \leq i \leq N$, $attr(q) = \overline{X}$, $attr(r_i) = \overline{X_i} \cup \overline{Y_i}$, $\overline{X_i} \subseteq \overline{X}$, $\overline{Y_i} \cap \overline{X} = \emptyset$, $Y = \cup_{i=1}^{N}(\overline{Y_i})$, and P is a predicate over $\overline{X} \cup \overline{Y}$. 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 Σ_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 $\theta_{e_S}(\Sigma_{S_j})$.[12] 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 *certain*—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 $< a_1, a_2, \ldots, a_M >$ in q_D^{Ext} is a certain answer in q_I for q.*
Proof. (See Appendix C.)

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 D.)

[11]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.)

[12]We keep non-lexical objects in different sources separate by consistently introducing new OIDs for target objects.

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, BGLaV, also has the ability to derive views over source schemas. The sets of view-creation operators, however, are incompatible—in BGLaV 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.

[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, BGLaV encapsulates views for source relations in mapping elements. Since the view definitions are immediately available, query processing in BGLaV 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 BGLaV, 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, BGLaV 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(1), 2003. To appear, currently at http://www.deg.byu.edu/papers/int.pdf.

[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 2002.

[CGMH+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 1994.

[CLL01] D. Calvanese, D. Lembo, and M. Lenerini. Survey on methods for query rewriting and query answering using views. Technical report, University of Rome, Roma, Italy, April 2001.

[DDH01] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 509–520, Santa Barbara, California, May 2001.

[ECJ+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.

[EJX01] D.W. Embley, D. Jackman, and Li Xu. Multifaceted exploitation of metadata for attribute match discovery in information integration. In *Proceedings of the International Workshop on Information Integration on the Web (WIIW'01)*, pages 110–117, Rio de Janeiro, Brazil, April 2001.

[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 21th International Conference on Conceptual Modeling (ER2002)*, pages 322–337, Tampere, Finland, October 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, 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*, pages 539–542, Tucson, Arizona, May 1997.

[Hal01] A.Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[LRO96] A.Y. Levy, A. Rajaraman, and J.J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the Twenty-second International Conference on Very Large Data Bases (VLDB'96)*, pages 251–262, Mumbai (Bombay), India, September 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 2000.

[RB01] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 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 1997.

[XE02] L. Xu and D.W. Embley. Discovering direct and indirect matches for schema elements. 2002. Submitted for publication, currently at http://www.deg.byu.edu/papers.

A Experimental Results of Source-to-Target Mappings

We evaluated the performance of source-to-target mappings[13] based on three measures: precision, recall, and the F-measure, a standard measure for recall and precision together. Given (1) the number of direct and indirect matches N determined by a human expert, (2) the number of correct direct and indirect matches C selected by our process, and (3) the number of incorrect matches I selected by our process, we computed the recall ratio as $R = C/N$, the precision ratio as $P = C/(C+I)$, and the F-measure as $F = 2/(1/R + 1/P)$. We reported all these values as percentages.

We considered three applications: *Course Schedule*, *Faculty*, and *Real Estate* to evaluate our schema mapping approach. We used a data set downloaded from the LSD homepage [DDH01] for these three applications, and we faithfully translated the schemas from DTDs used by LSD to rooted hypergraphs. For testing these applications, we decided to let any one of the schema graphs for an application be the target and let any other schema graph for the same application be the source. Because our tests were nearly symmetrical, however, we decided not to test any target-source pair also as a source-target pair. We also decided not to test any single schema as both a target and a source. Since for each application there were five schemas, we tested each application 10 times. All together we tested 30 target-source pairs.

Application	Number of Matches	Number Correct	Number Incorrect	Recall %	Precision %	F-Measure %
Course Schedule	128	119	1	93%	99%	96%
Faculty	140	140	0	100%	100%	100%
Real Estate	245	229	22	93%	91%	92%
All Applications	513	488	23	95%	95%	95%

Table 1: Test Results

Table 1 shows a summary of the results for the data. In two of the three applications, *Course Schedule* and *Faculty*, there were no indirect matches. The *Real Estate* application exhibited several indirect matches. The problem of *Merged/Split Values* appeared twice, the problem of *Union/Selection* appeared 24 times, and the problem of *Object-Set Name as Value* appeared 5 times. Our process successfully found all the indirect matches related to the problems of *Merged/Split Values* and *Object-Set Name as Value*. For the problem of *Union/Selection*, our process correctly found 22 of the 24 indirect matches and declared two extra indirect matches.[14] Over all the indirect element mappings, the three measures (recall, precision, and F-measure) were (coincidentally) all 94%.

[13] The data presented here measures the performance of mapping elements between object sets of source and target schemas. At the time we performed these tests, relationship-set matching had not been implemented.

[14] Of these four, three of them were ambiguous, making it nearly impossible for a human to decide, let alone a machine. In two cases there were various kinds of phones for firms, agents, contacts, and phones with and without message features, and in another case there were various kinds of descriptions and comments about a house written in free-form text. The one clear incorrect match happened when our process unioned office and cell phones together and mapped them to phones for a firm instead of just mapping office phones to firm phones and discarding cell phones, which had no match in the other schema.

B Examples for New Operators in the Mapping Algebra

B.1 Composition

Let r be the following relation, where $attr(r) = \{House,\ Street,\ City,\ State\}$.

House	Street	City	State
h1	339 Wymount Terrace	Provo	Utah
h2	15 S 900 E	Provo	Utah
h3	1175 Tiger Eye	Salt Lake City	Utah

Applying the operation $\lambda_{(Street,``,\ ",City,``,\ ",Street),Address}r$ yields a new relation r', where $attr(r') = \{House,\ Street,\ City,\ State,\ Address\}$.

House	Street	City	State	Address
h1	339 Wymount Terrace	Provo	Utah	339 Wymount Terrace, Provo, Utah
h2	15 S 900 E	Provo	Utah	15 S 900 E, Provo, Utah
h3	1175 Tiger Eye	Salt Lake City	Utah	1175 Tiger Eye, Salt Lake City, Utah

B.2 Decomposition

Let r be the following relation, where $attr(r) = \{House,\ Address\}$.

House	Address
h1	Provo, Utah
h2	339 Wymount Terrace, Provo, Utah
h3	1175 Tiger Eye, Salt Lake City, Utah

Applying the operation $\gamma^{R}_{Address,Street}r$, where R is a routine that obtains values of $Street$ from values of $Address$, yields a new relation r_1, where $attr(r_1) = \{House,\ Address,\ Street\}$.

House	Address	Street
h1	Provo, Utah	
h2	339 Wymount Terrace, Provo, Utah	339 Wymount Terrace
h3	1175 Tiger Eye, Salt Lake City, Utah	1175 Tiger Eye

Similarly, applying the operation $\gamma^{R'}_{Address,City}r$, where R' is a routine that obtains values of $City$ from values of $Address$, yields a new relation r_2, where $attr(r_2) = \{House,\ Address,\ City\}$.

House	Address	City
h1	Provo, Utah	Provo
h2	339 Wymount Terrace, Provo, Utah	Provo
h3	1175 Tiger Eye, Salt Lake City, Utah	Salt Lake City

B.3 Boolean

Let r be the following relation, where $attr(r) = \{House,\ Water\ Front\}$.

House	Water Front
h1	Yes
h2	No
h3	Yes

Applying the operation $\beta^{\text{“}Yes\text{”},\text{“}No\text{”}}_{WaterFront,LotDescription} r$ yields a new relation r', where $attr(r') = \{House,\ Lot\ Description\}$.

House	Lot Description
h1	Water Front
h3	Water Front

B.4 DeBoolean

Let r be the following relation, where $attr(r) = \{House,\ Lot\ Description\}$.

House	Lot Description
h1	Water Front
h1	Golf Course
h1	Mountain View
h2	Water Front
h3	Golf Course

Applying the operation $\reflectbox{$\beta$}^{\text{“}Yes\text{”},\text{“}No\text{”}}_{Lot\ Description,Water\ Front} r$ yields a new relation r_1, where $attr(r_1) = \{House,\ Water\ Front\}$.

House	Water Front
h1	Yes
h2	Yes
h3	No

Similarly, applying the operation $\reflectbox{$\beta$}^{\text{“}x\text{”},\text{“”}}_{Lot\ Description,Golf\ Course} r$ yields a new relation r_2, where $attr(r_2) = \{House,\ Golf\ Course\}$.

House	Golf Course
h1	x
h2	
h3	x

B.5 Skolemization

Let r be the following relation, where $attr(r) = \{House\}$.

House
h1
h2
h3

Applying the operation $\varphi_{f_{Basic\ Features}} r$ yields a new relation r', where $attr(r') = \{House,\ Basic\ Features\}$.

House	Basic Features
h1	$f_{Basic\ Features}(h1)$
h2	$f_{Basic\ Features}(h2)$
h3	$f_{Basic\ Features}(h3)$

C 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 $< a_1, a_2, \ldots, a_M >$ in q_D^{Ext} is a certain answer in q_I for q.

Proof (sketch). Let a query q be $\pi_{(\overline{X})}\sigma_P(r_1 \bowtie r_2 \bowtie \ldots \bowtie r_N)$, 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 $a = < a_1, a_2, \ldots, a_M >$ 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 $\pi_{(\overline{X})}\sigma_P(\cup_{(S_j.e_S \subseteq r_1) \in ID_1}(S_j.e_S) \bowtie \cup_{(S_j.e_S \subseteq r_2) \in ID_2}(S_j.e_S) \bowtie \ldots \bowtie \cup_{(S_j.e_S \subseteq r_N) \in ID_N}(S_j.e_S))$, 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, $\ldots$, and s_N, and N tuples, c_1, c_2, $\ldots$, and c_N, such that $c_i \in s_i$ and $a = \pi_{(\overline{X})}\sigma_P(c_1 \bowtie \ldots \bowtie c_N)$, 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}, $c_i \in D_{S_jT}$. Since $c_i \in D_{S_jT}$, based on the definition of $sem(I)$, $c_i \in sem(I)$. Since $c_i \in sem(I)$, $a = \pi_{(\overline{X})}\sigma_P(c_1 \bowtie \ldots \bowtie c_N)$, 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.

D 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 $\pi_{(\overline{X})}\sigma_P(r_1 \bowtie r_2 \bowtie \ldots \bowtie r_N)$, 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 $a = < a_1, a_2, \ldots, a_M >$ 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, $\ldots$, c_N in $sem(I)$ such that $c_i \in r_i$, $1 \leq i \leq N$, and $a = \pi_{(\overline{X})}\sigma_P(c_1 \bowtie c_2 \bowtie \ldots \bowtie c_N)$. Therefore, since $c_i \in sem(I)$, $1 \leq i \leq N$, based on the definition of $sem(I)$, a target interpretation D_{S_jT} with respect to D_{S_j} must exist such that $c_i \in D_{S_jT}$, where $S_j \in I$ and D_{S_j} is a valid interpretation of S_j and $T \in I$. Since $c_i \in D_{S_jT}$ and $c_i \in r_i$, $1 \leq i \leq N$, based on the definition of D_{S_jT}, 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 $\pi_{(\overline{X})}\sigma_P(s_1 \bowtie \ldots \bowtie s_N)$ over D is a subset of q_D^{Ext}. When evaluating $\pi_{(\overline{X})}\sigma_P(s_1 \bowtie \ldots \bowtie s_N)$, since $a = \pi_{(\overline{X})}\sigma_P(c_1 \bowtie \ldots \bowtie c_N)$ 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 $\pi_{(\overline{X})}\sigma_P(s_1 \bowtie \ldots \bowtie s_N)$ over D. Since the query answer to $\pi_{(\overline{X})}\sigma_P(s_1 \bowtie \ldots \bowtie s_N)$ over D is a subset of q_D^{Ext} and a is a tuple of query answer to $\pi_{(\overline{X})}\sigma_P(s_1 \bowtie \ldots \bowtie s_N)$ over D, $a \in q_D^{Ext}$. This is contrary to the assumption that a is not a tuple in q_D^{Ext}.