2009 IEEE International Conference on Semantic Computing

An Algebraic Language for Semantic Data Integration on the Hidden Web*

Shazzad Hosain
Department of Computer Science
Wayne State University, USA
shazzad@wayne.edu

Hasan Jamil
Department of Computer Science
Wayne State University, USA
jamil@cs.wayne.edu

***Abstract*—Semantic integration in the hidden Web is an emerging area of research where traditional assumptions do not always hold. Frequent changes, conflicts and the sheer size of the hidden Web demand vastly different integration techniques that rely on autonomous detection and heterogeneity resolution, correspondence establishment, and information extraction strategies. In this paper, we present an algebraic language, called *Integra*, as a foundation for another SQL-like query language called *BioFlow*, for the integration of Life Sciences data on the hidden Web. The algebra presented here adopts the view that the web forms can be treated as user defined functions and the response they generate from the back end databases can be considered as traditional relations or tables. These assumptions allow us to extend the traditional relational algebra to include integration primitives such as schema matching, wrappers, form submission, and object identification as a family of database functions. These functions are then incorporated into the traditional relational algebra operators to extend them in the direction of semantic data integration. To support the well known concepts of horizontal and vertical integration, we also propose two new operators called *link* and *combine*. We show that these family of functions can be designed from existing literature and their implementation is completely orthogonal to our language in the same way many database technologies are (such as relational join operation). Finally, we show that for traditional relations without integration, our algebra reduces to classical relational algebra establishing it as a special case of Integra.**

I. Introduction

World Wide Web (WWW) is now a major cyberinfrastructure for information modeling, content delivery and communication. While most of its contents are designed as static HTML or XML pages and stored in user directories called the *shallow web*, increasingly a vast amount of such pages are dynamically generated using server side scripts from databases contents in response to form based queries, or stored queries presented to user as links in HTML pages. The beauty of these dynamically generated pages is that the information contents are changed every single time based on user inputs, and thus are very convenient. In this approach, more information can be delivered in a more concise way but much faster through automated engines that can be adjusted as needed without much effort. This new model is known as the *deep web* or the *hidden web*.

* Research supported in part by National Science Foundation grants CNS 0521454 and IIS 0612203.

Regardless of how the information is presented to the user, it is still difficult for any user to gather information from multiple web sources and consolidate to form coherent knowledge. This is because often such needs demand non-trivial semantic reconciliation. For example, if one asks for all the publications of professor John Smith, traditional search engine such as Google will return pages where it finds the keywords *professor*, *John* and *publications*, etc. If professor Smith worked in two different universities, for example Wayne State University (wsu) and Carnegie Mellon University (cmu), and he has a list of publications on both the universities' sites, Google will return two different pages without combining the result into a single set. Now, if we would like to consolidate his publication records with professional activities such as committee memberships, shallow web search engine Google will not be able to help us even though it is able to return the pages that have those information. Ideally, we would like to combine the publications of Professor Smith in a way similar to set union, and link the personal information in committee pages to expand the set of attributes we know about him already in a way similar to relational join.

Traditional search engines for shallow web cannot index and query hidden web databases since the pages are dynamically created in response to query forms. The lack of formal foundations and query support made it difficult to develop search engines for hidden web. Except for a very few specific applications such as *www.expedia.com*, *www.cheapair.com* in comparison shopping, ad hoc semantic integration is still at its early stage. In these applications, semantic reconciliation and data integration is totally manual for a very specific collaborating sites, and thus dynamically adding a new and arbitrary site has been extremely difficult. It is our thesis that we should be able to treat information from arbitrary hidden web source as a set of relations, and query those relations using a query language such as SQL or relational algebra. Although the goal is simple, the task is not trivial. But there is hope. The idea we pursue in this paper can be summarized as follows.

Let us assume that there is an operation that can submit values from a set of tuples to a web-form. Once the values are submitted, another operation is able to extract the required information in the form of a table. Issues that remain

978-0-7695-3800-6 2009

DOI 10.1109/ICSC.2009.94

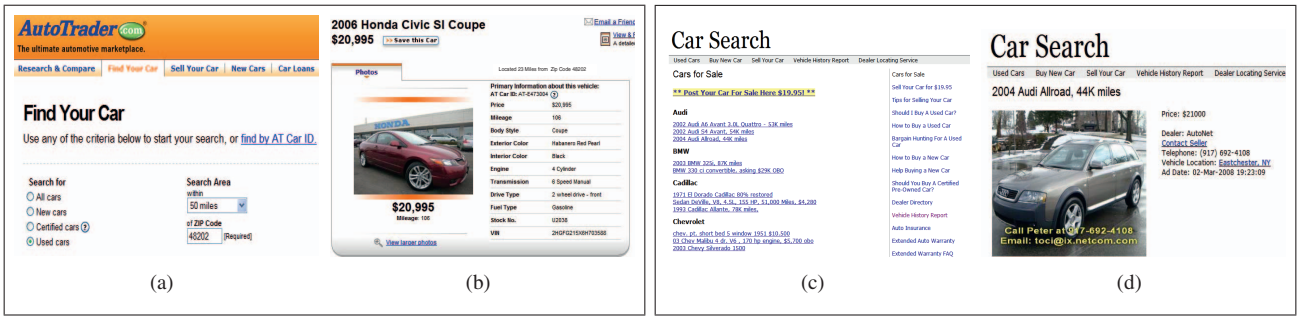
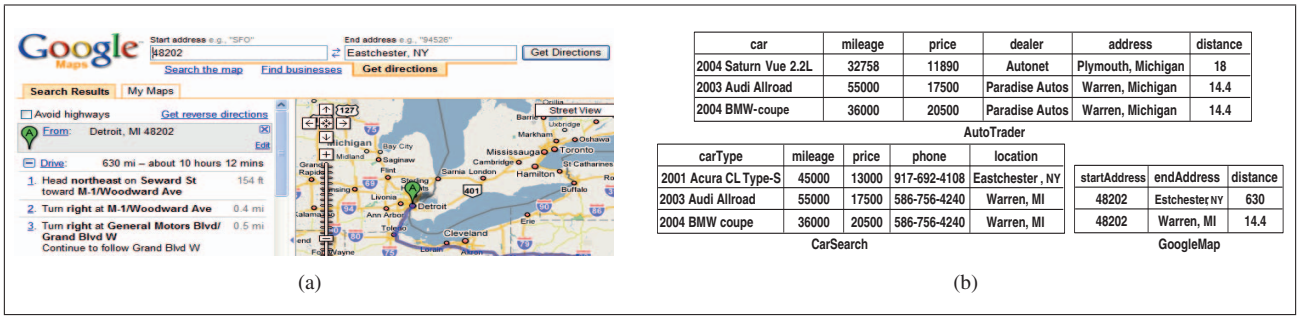


Figure 1. www.autotrader.com and www.carsearch.com



car	mileage	price	dealer	address	distance
2004 Saturn Vue 2.2L	32758	11890	Autonet	Plymouth, Michigan	18
2003 Audi Allroad	55000	17500	Paradise Autos	Warren, Michigan	14.4
2004 BMW-coupe	36000	20500	Paradise Autos	Warren, Michigan	14.4

AutoTrader

carType	mileage	price	phone	location
2001 Acura CL Type-S	45000	13000	917-692-4108	Eastchester , NY
2003 Audi Allroad	55000	17500	586-756-4240	Warren, MI
2004 BMW coupe	36000	20500	586-756-4240	Warren, MI

CarSearch

startAddress	endAddress	distance
48202	Estchester NY	630
48202	Warren, MI	14.4

GoogleMap

Figure 2. Distance from Google and extracted relations

to be addressed now is that how does the operator map the variables in the query to the web form fields and to the extracted table column names. Recall that we are accessing multiple web sites, and so the schemes are expected to be different even though we have only one query at hand. The name heterogeneity is just one dimension of the complexity present. The representation is another heterogeneity that needs to be addressed as well. That is to say, how do we know that John Smith in Wayne State University site is the same John Smith in Carnegie Mellon site? This recognition require object identification and disambiguation. Once we address these issues, we are in a much better shape to put together a query language for the hidden web, and use the following motivating example to begin our presentation of the algebraic language we would like to propose.

A. Motivating Example

Let us consider a car sales database involving *www.autotrader.com* (figure Ia), and *www.carsearch.com* (figure Ic), and the query *"list all used BMW coupe cars priced less than $25,000 with mileage no more than 50,000 and dealer location within 50 miles of zip code 48202"*.

The autotrader.com database expects the zip code and the search area as inputs (figure Ia) to generate the response shown in figure Ib. On the other hand carsearch.com database lists all cars directly as hot links as shown in figure Ic even though the information is presumably taken from a structured database, and clicking on a link produces the car information without the distance value (figure Id). If we extract the information displayed in each of these sites on these cars, we can actually come with as AutoTrader and CarSearch tables shown in figure 2b. Since carsearch.com does not show the distance of the car dealers, one can use Google map (figure 2a) to find the distance and collect the info into GoogleMap table shown in figure 2b. Now, these relations can be queried to answer the question.

A careful review of these three synthesized tables will show that although they contain similar or related information, their scheme is different in size and name, and the values are semantically similar yet syntactically distinct. This raises several questions that highlights the need for semantic integration. First, how can one identify that AutoTrader and CarSearch relations contain similar objects such as cars? Second, how can one identify that '2004 BMW-coupe' in AutoTrader relation and '2004 BMW coupe' in CarSearch relation are the same object, and if identified as the same object, what will be the combined information of that car? Third, how can CarSearch and GoogleMap be linked to get the distance of cars from zip code 48202?

B. Goals and Contributions

In our recent research, we have developed an SQL-like declarative query language for data integration and workflow design in Life Sciences, called *BioFlow* [1], [2], which is being used as the primary query language in our data management system LifeDB [3]. While BioFlow is implemented as an extension of SQL, its formal foundation is missing, and we need to understand its functionalities and capabilities in a way similar to relational algebra so that we understand BioFlow better, and possibly lay the foundation for a direct

implementation of a semantic data integration language in the near future. With this goal in mind, we propose an extension of classical relational algebra, called *Integra*, for the semantic integration of hidden Web databases. The extended operators use existing reconciliation functions such as automatic schema matching and key identification functions as user defined functions (UDF)s to resolve semantic heterogeneity. Such UDFs have played major roles in [4] to merge conflicting information into one relation. We too follow this tradition and introduce four different kinds of UDFs such as *form*, *wrapper*, *schema matching* and *key identification* functions to enrich our model and deliver the capability Integra needs to support ad-hoc integration on the hidden Web. Finally, we show that Integra is as expressive as relational algebra and the classical relational algebra is a special case of Integra.

II. Semantic Integration on the Hidden Web

The main vehicle for semantic integration is the ability to establish semantic equivalence between terms and objects (exact definition to follow). For example, 'Warren, Michigan' and 'Warren, MI' are two semantically equivalent terms given that MI is the abbreviation for Michigan. Similarly '2004 BMW coupe' is equivalent to '2004 BMW-coupe' if we ignore '-'. Also the attribute *address* in AutoTrader relation is equivalent to *location* of CarSearch relation since *address* and *location* are synonymous with each other and both have similar values. So the question is, how does the inclusion of semantic equivalence into relational algebra changes its model and the functionality. It turns out that substantial machineries need to be added to adapt, and extend relational algebra to be able to support semantic integration. In particular, it now needs to support two additional operations that are the two essential pilers of data integration – *horizontal* and *vertical* integration we have alluded to before. Roughly speaking, horizontal integration aims to increase the cardinality of a relation by including more semantic objects into the collection, while vertical integration aims to increase the degree of a relation by adding more properties to existing objects. It turns out that traditional union and join operations are not sufficient.

A. Semantic Objects and Equivalence

If we consider the relations AutoTrader and CarSearch in figure 2b, we notice several properties that make it complicated to directly use relational algebra for data integration purposes. For example, they are not union compatible (CarSearch relation has *phone* and AutoTrader relation has *distance* in figure 2b) as required by relational model, and so we cannot just take a union of these two relations (horizontal integration) without sanitizing them properly. In other words, they have vastly different schemes and data representation. Even if we somehow make the relations union compatible, we will need to be able to identify that '2004 BMW coupe' in CarSearch relation is the same object '2004 BMW-coupe' in the AutoTrader relation. It is thus reasonable to expect that in the combined information for all the cars, we will have all the attributes as shown in figure 3f, and the semantic objects as shown. Now the question is, could query these relations without sanitizing them ahead and independent of their schemes so that we do not have to worry about how and what to sanitize the candidate schemes to so that our query language will not break down just because we did not conform to one or the other input relation schemes.

For example, we would like to see that user query *"select carType, mileage, value from AutoTrader"* is computable even though the AutoTrader relation lists the attributes as *car*, *mileage* and *price*. In our view, this is a reasonable expectation since *car*, *mileage* and *price* from AutoTrader have semantically similar meaning to *carType*, *mileage* and *value* from the query. We would also like to see that statements such as *"link CarSearch, GoogleMap"* and *"combine CarSearch, AutoTrader"* achieves the same functionality as in vertical and horizontal integration in which the objects are semantically identified after the possible heterogeneity is resolved. These are the issues we plan to address in the next few sections.

III. The Integra Data Model

Like several earlier research [5], [6] we too take a relational view of the web. By doing so we attempt to transform the web content into a structured representation in order to develop a formal foundation so that a query language can be designed. Also we are motivated by the fact that the dynamic Web pages usually use tabular representation to expose their contents which closely resembles traditional relations[1]. From the existing research we know that these tables can be identified and extracted quite efficiently by wrappers, such as FastWrap [7], DEPTA [8] etc., using page scrapping. All we need to account for, how the submission and transformation operations mesh with the algebraic operators we propose. We proceed by discussing a few needed concepts to lay the foundation.

A. Semantic Equivalence of Terms

A *term* in Integra is any basic element such as an attribute name (called an *a-term*), its type (*t-term*), or a value (*v-term*). Formally, the language $\mathcal{L}$ of Integra is a triple $\langle \mathcal{A}, \mathcal{V}, \mathcal{T} \rangle$ such that $\mathcal{A}$ is the set of attribute names, $\mathcal{V}$ is a set of basic values, $\mathcal{T}$ is a set of type names with associated domains. For example, the a-term *Name* has a type *String* with domain *Name* in which *John* is a value or v-term.

[1]By making these assumptions we are in a way assuming that in the rare set of hidden web databases that support a different form of user interaction or produce information in a form other than tables, our model will not be appropriate.

Definition 1 (Term Similarity): Let ψ' be a similarity function[2] *such that for any two $t_1, t_2 \in \mathcal{A} \cup \mathcal{V} \cup \mathcal{T}$, $\psi'(t_1, t_2) \in [0, 1]$, where 0 indicates complete dissimilarity and 1 means two terms are identical. Let ψ be a function and ϵ be a threshold such that $\psi(\epsilon, \psi'(t_1, t_2)) \in \{0, 1\}$, i.e. $\psi(\epsilon, \psi'(t_1, t_2)) = 0$ if $\psi'(t_1, t_2) < \epsilon$; $\psi(\epsilon, \psi'(t_1, t_2)) = 1$ otherwise, and we say t_1 and t_2 are similar, denoted $t_1 \sim t_2$.*

From here on, we write $\psi(t_1, t_2)$ as a short hand for $\psi(\epsilon, \psi'(t_1, t_2))$ unless specified otherwise.

Definition 2 (Term Equivalence): Two a-terms $a_1, a_2 \in \mathcal{A}$ are called attribute, or a-equivalent, if $\psi(a_1, a_2) = 1$ and their associated types are identical (i.e., $\psi'(t_1, t_2) = 1$), denoted $a_1 \simeq a_2$. Two v-terms $v_1, v_2 \in \mathcal{V}$ are called value, or v-equivalent, if $\psi(v_1, v_2) = 1$, and their corresponding a-terms are equivalent (i.e., $a_1 \simeq a_2$), denoted $v_1 \simeq v_2$.

For example, $car \simeq carType$, since $car \sim carType$ and both have type string in relations AutoTrader and CarSearch (figure 2b). Similarly 'Warren, Michigan' $\simeq$ 'Warren, MI' since 'Warren, Michigan' $\sim$ 'Warren, MI' and the corresponding attributes are equivalent i.e. $address \simeq location$.

B. Database Functions

We exploit several existing integration technologies and incorporate them as user defined functions. Below, we present a brief discussion on three such functions namely (i) semantic reconciliation function μ, (ii) extraction function η, and (iii) the key discovery function κ.

1) Semantic Reconciliation Function (μ): Given two sets of terms R and S, where R and S may be relation schemas, the function μ returns a set of equivalent pair of terms. For example, $\mu(AutoTrader, CarSearch)$ returns $\{< car, carType >, < mileage, mileage >, < price, price >, < address, location >\}$. Here, μ acts as a schema matching technique such as OntoMatch [11]. However, μ also determines equivalency between two terms, where it has the functionality of ψ. We can then readily cast the term equivalence as $\simeq^{\mu}$ to denote the fact that term equivalence is with respect to the semantics of μ. We now formally define μ as schema correspondence function below:

Definition 3 (Schema Correspondence): Let R and S be two schemas, ϵ be a threshold, and δ be a distance function[3] *such that $\delta(A_i, A_j) \in [0, 1]$ for any $A_i \in R$, and $A_j \in S$. Then $\mu(R, S) = \{< A_i, A_j > | A_i \in R \wedge A_j \in S \wedge \delta(A_i, A_j) \leq \epsilon$, such that $\forall A_k \in S(\delta(A_i, A_k) > \delta(A_i, A_j))\}$.*

2) Key Discovery Function (κ): Given a relation, the key discovery function κ identifies the keys of that relation. Formally, if $r(R)$ is a relation then $\kappa(r) = \{K_1, \ldots K_n\}$, where each $K_i = \{A_i\}$ is a key of r. We can use any existing algorithms such as GORDIAN [12] as κ.

[2]Such as string edit distance [9], thesaurus at WordNet [10], etc. or a combination of such functions.

[3]A distance function δ can be defined in terms of a similarity function ψ as $\delta(A_i, A_j) = 1 - \psi(A_i, A_j)$, where $A_i \in R, A_j \in S$.

3) Extraction Function (η): Given a web page W, the function η identifies and extracts a table or relation from W. Such functions are known as wrapper induction [13] tools. In the literature we find many automatic wrapper induction tools such as FastWrap [7], DEPTA [8] etc. of which any one can be used as η.

C. Integra Operators

In this section we define the basic set of operators[4] needed to devise an algebraic query language for the hidden Web such as *selection* ($\hat{\sigma}$), *projection* ($\hat{\pi}$), *intersection* ($\hat{\cap}$), and *difference* ($\hat{-}$). We also define three new operators unique to our model called the *link* ($\hat{\lambda}$) operator for vertical integration of relations, *combine* ($\hat{\chi}$) for horizontal integration, and finally the *transform* ($\hat{\tau}$) operator to convert the hidden Web into relations.

However, it turns out that our combine operation is synonymous to the classical union operation when the pair of relations satisfy union compatibility in Codd's relational model [14]. Consequently, we do not define a separate union operator. Finally, the Cartesian product and rename operations are identical to the classical case, and hence omitted. Before we go ahead define the algebra operators, let us give the following definition.

Definition 4 (Semantic Comparison): Let μ be a reconciliation function; r_1, r_2 be two relation instances over schemes R_1, R_2; v_1, v_2 be two v-terms in tuples $t_1 \in r_1$ and $t_2 \in r_2$ corresponding to attributes A_1 and A_2; and θ be a Boolean comparator. Then, $A_1 \theta A_2$ holds for $A_1, A_2 \in R_1 \cup R_2$, denoted $A_1 \theta^{\mu}_{R_1, R_2} A_2$, if $< A_1, A_2 > \in \mu(R_1, R_2)$, and $v_1 \theta v_2$ hold. We say that $v_1 \theta v_2$ holds when (i) θ is an equality and $< v_1, v_2 > \in \mu(v_1, v_2)$, or (ii) θ is not an equality and there exists a v' such that $< v', v_2 > \in \mu(v', v_2)$ and $v' \theta v_2$ holds in classical sense[5].

To define Integra operators we will use the following relations and symbols for the rest of the section. Let r and s be two relations over the schemes $R = (A_1, A_2, \ldots, A_m)$, and $S = (B_1, B_2, \ldots, B_n)$. Also let $A = (A_1, A_2 \ldots, A_p)$, and $B = (B_1, B_2 \ldots, B_p)$, where $p \leq min(m, n)$, be the terms in R and S such that $< A_i, B_i > \in \mu(R, S)$ i.e. $A_i \simeq^{\mu} B_i, i = 1, \ldots, p$ holds.

Definition 5 (Output Schema for Binary Operators): If $r(R)$ and $s(S)$ be two relations then except for minus operation the output schema O will be $A_1, \ldots, A_m, B_{p+1}, \ldots, B_n$, meaning that, the output schema first takes attributes from the first relation and then attributes from the second relation for which there are no equivalent attributes in the first relation.

[4]From now on, we denote the classical operators with plain Greek symbols (i,e., σ), whereas the corresponding extended operators are denoted using a hat over the symbol (i,e., $\hat{\sigma}$). Furthermore, from this point on when we say operator, we mean extended operator, and the traditional operators are referred to as classical operators, unless otherwise mentioned.

[5]This definition can be slightly and easily tailored to define the cases for comparisons such as $A\theta v$, or $v_1 \theta v_2$ and left as an exercise.

car	mileage	price	dealer	address	distance
2003 Audi Allroad	55000	17500	Paradise Autos	Warren, MI	14.4
2004 BMW-coupe	36000	20500	Paradise Autos	Warren, MI	14.4

(a) selection on AutoTrader

car	price	address
2001 Acura CL Type-S	13000	Eastchester , NY
2003 Audi Allroad	17500	Warren, MI
2004 BMW coupe	20500	Warren, MI

(b) projection on CarSearch

car	mileage	price	phone	location	startAddress	distance
2001 Acura CL Type-S	45000	13000	917-692-4108	Eastchester , NY	48202	630
2003 Audi Allroad	55000	17500	586-756-4240	Warren, MI	48202	14.4
2004 BMW-coupe	36000	20500	586-756-4240	Warren, MI	48202	14.4

(c) link on CarSearch and GoogleMap

car	mileage	price	dealer	address	distance
2004 Saturn Vue 2.2L	32758	11890	Autonet	Plymouth, Michigan	14.4

(d) difference on AutoTrader and CarSearch

car	mileage	price	dealer	address	distance	phone
2003 Audi Allroad	55000	17500	Paradise Autos	Warren, Michigan	14.4	586-756-4240
2004 BMW-coupe	36000	20500	Paradise Autos	Warren, Michigan	14.4	586-756-4240

(e) intersection on AutoTrader and CarSearch

car	mileage	price	dealer	address	distance	phone
2004 Saturn Vue 2.2L	32758	11890	Autonet	Plymouth, Michigan	18	null
2003 Audi Allroad	55000	17500	Paradise Autos	Warren, Michigan	14.4	586-756-4240
2004 BMW-coupe	36000	20500	Paradise Autos	Warren, Michigan	14.4	586-756-4240
2001 Acura CL Type-S	45000	13000	null	Eastchester , NY	null	917-692-4108

(f) combine on AutoTrader and CarSearch

Figure 3. Operations on relations

1) Select Operator ($\hat{\sigma}$): The selection operation is defined in a way analogous to classical selection in the context of reconciliation function μ. If r is a relation over the schema R and T_i be either an a-term or a v-term, then

$$\hat{\sigma}^{\mu}_{A_1\theta T_1,\ldots,A_k\theta T_k}(r) = \{t | t \in r, \text{ and } \bigwedge_{i=1}^{k} A_i \theta^{\mu}_{\cup_i A_i, R} T_i \text{ hold}\}$$

Example 1: $\hat{\sigma}^{\mu}_{address='Warren,MI'}(AutoTrader)$ will return the relation shown in figure 3a. The operator selects the rows from AutoTrader relation for which the *address* value is similar to 'Warren, MI'. Since 'Warren, Michigan' $\sim$ 'Warren, MI', the operation selects two tuples and replaces 'Warren, Michigan' with 'Warren, MI' in the output relation.

2) Project Operator ($\hat{\pi}$): Projection operation is also defined along the lines of classical projection taking the reconciliation function into account. If r is a relation over the schema R, A'_i be a set of attribute names, t be any tuple in r where $t[A_i]$ denote the projection of t on attribute A_i, and $\|$ is a concatenation function, then

$$\hat{\pi}^{\mu}_{A'_1,A'_2,\ldots,A'_k}(r) = \{\|_{i=1}^{k}\, t_i | t \in r, t_i = t[A_i], \text{ and } < A_i, A'_i > \in \mu(\cup_i A'_i, R) \text{ hold for } i = 1,\ldots,k\}$$

Example 2: $\hat{\pi}^{\mu}_{car,price,address}(CarSearch)$ will return the relation shown in figure 3b. The scheme of the returned relation will replace all A_i with A'_i for which $< A_i, A'_i > \in \mu(\cup_i A'_i, R)$ holds. Let the μ function maps *car* to *carType*, *price* to *price* and *address* to *location*. The operator then projects *carType, price* and *location* from CarSearch relation and changes the attribute names to *car, price* and *address*.

3) Combine Operator ($\hat{\chi}$): Combine operation seeks to collect similar objects in one relation from multiple tables, in a way similar to classical union operation. Differently from classical union though, combine operation attempts to recognize semantic objects through key identification and semantic reconciliation in terms of schema mismatch, and hence does not enforce classical union compatibility.

Two relations are combine compatible if both relations have semantically equivalent keys.

Definition 6 (Combine Compatibility): Let r, s be two relations over the schemes R, S; u, v be tuples in r, s; $K_r = (A_i, \ldots, A_k)$ and $K_s = (B_i, \ldots, B_k)$ be two candidate keys in R and S such that $K_r \subseteq A$, $K_s \subseteq B$, and $K_r \simeq^{\mu} K_s$ i.e. $\forall (A_j \in K_r, B_j \in K_s, u[A_j] \simeq^{\mu} v[B_j], < A_j, B_j > \in \mu(R, S)$ hold for $j = 1, \ldots, k)$, then r and s are combine compatible.

Finally, let r and s be two relations over the schemes R and S, κ be a key identification function such that $\kappa(r) = K_r, \kappa(s) = K_s$, μ be a schema matching function that identifies $K_r \simeq^{\mu} K_s$, then

$$r\hat{\chi}^{\mu}_{\kappa} s = \{t | (\exists u : u \in r, \neg\exists v : (v \in s \text{ such that } K_r \simeq^{\mu} K_s), t = u \parallel (\|_{i=1}^{n-p} \perp)) \vee (\exists u : u \in s, \neg\exists v : (v \in r \text{ such that } K_s \simeq^{\mu} K_r), t = (\|_{i=1}^{m-p} \perp) \parallel u) \vee (\exists u : u \in r, \exists v : (v \in s \text{ such that } K_r \simeq^{\mu} K_s), t = u \parallel_{i=p+1}^{n} v[B_i])\}$$

Example 3: $AutoTrader \;\; \hat{\chi}^{\mu}_{\kappa} \;\; CarSearch$ will return the relation shown in figure 3f. Let $\mu(AutoTrader, CarSearch)$ returns the mapping $< car, carType >, < mileage, mileage >, < price, price >, < address, location >$. The output schema thus takes the attributes from AutoTrader relation followed by *phone* from CarSearch, for which there is no equivalent attribute in AutoTrader. Now, let the κ function identifies *car* and *carType* as the key attributes of the input relations. The key values identify distinct objects in a relation. The two distinct cars '2004 Audi Allroad' and '2004 BMW-coupe' are present in both relations. Thus in combined relation both cars have values for all attributes. Like the schema definition, values are taken first from the AutoTrader relation, then from CarSearch. If an object is present in only one relation then in the combined relation unbound attributes take *null* values. For example, '2004 Saturn Vue 2.2l' has *null* for *phone*, and '2001 Acura CL Type-S' has *null* for *dealer* and *distance*.

4) Difference ($\hat{-}$) and Intersection ($\hat{\cap}$) Operator: The difference operation (shown in figure 3d) simply removes the objects from the first relation that have semantically identical objects in the second relation, in a way similar to the classical case. The output relation takes the schema of the first relation.

The intersection operation (figure 3e) gathers semantically identical objects that appear in both relations. Since the objects are present in both relations the output relation has the combined information of the objects. Unlike the classical intersection the extended intersection operation cannot be expressed in terms of extended difference operation, i.e. $r\hat{\cap}_{\kappa}^{\mu}s \neq r\hat{-}_{\kappa}^{\mu}(r\hat{-}_{\kappa}^{\mu}s)$, because the degrees of input relations are not the same. However, if the degrees are same then $r\hat{\cap}_{\kappa}^{\mu}s = r\hat{-}_{\kappa}^{\mu}(r\hat{-}_{\kappa}^{\mu}s)$ holds.

5) Link Operator ($\hat{\lambda}$): While combine operation requires two relations having candidate keys, link operation requires one relation having a candidate key and the other having a foreign key. Thus link operation is at most one to many. Link operation is somewhat analogous to the classical natural join operation in terms of its function, but is fundamentally different semantically. In case of natural join, the two relations are joined if they have common attributes and if they have common values. But the link compatibility depends on key constraints, where the keys are usually discovered in the context of participating relations.

Now, let r and s be two relations over the schemes R and S, κ be a key identification function such that $\kappa(r) = K_r, \kappa(s) = K_s$, $K_r \subseteq A$, $K_s \subseteq S$; μ be a schema matching function that identifies $K_r \subseteq K_s$, then

$$r\hat{\lambda}_{\kappa}^{\mu}s = \{u \parallel t | (\exists u : u \in r, \exists v : (v \in s, \text{ such that } \forall A_i \in K_r \; \exists B_i \in K_s(u[A_i] \simeq^{\mu} v[B_i], < A_i, B_i > \in \mu(R,S))), \; t = \|_{i=p+1}^{n} v[B_i])\}$$

Example 4: $CarSearch\ \hat{\lambda}_{\kappa}^{\mu}\ GoogleMap$ will produce the relation shown in figure 3c. The GoogleMap relation has the key *endAddress* and CarSearch relation has the foreign key *location*, thus the link operation is one to many. The output schema of the linked relation follows the same rule of combined schema. The linked relation has the tuples that have semantically similar values for the linking attributes e.g. 'Warren, MI' links to 'Warren, MI' and so on.

6) Transform Operator ($\hat{\tau}$): The operators discussed so far perform necessary operations on traditional relations much like relational algebra operators. But for automatic integration it first needs to transform the hidden Web into relations. The *transform* operator does so by taking three different user defined functions such as matching function (μ), form function (φ), extract function (η) and the output schema (S) as parameters. The operator ($\hat{\tau}$) first submits a web form to get the result web page, then extracts the data table from the page and finally applies projection ($\hat{\pi}$) operation to get the desired relation. Algebraically the transform operator is written as the following:

$$\hat{\tau}_{\varphi,\eta,S}^{\mu}(r)$$

Let, $\hat{\tau}_{autotrader.com,FastWrap,S}^{OntoMatch}(r)$ be a transform operator to convert autotrader.com site to AutoTrader relation. Recall that a web form has a set of input controls such as text box, list, combo box etc. with associated labels. For example, figure Ia has a text box with label *'ZIP Code'* and combo box with label *'Search Area'*. For automatic form submission, we have to send http request with (control, value) pairs such as (*'Zip Code'*, 48202) and (*'Search Area'*, 50). But, let's say, the user wants to submit *zip-code* = 48202 and *radius* = 50, though *zip-code* and *radius* are not the exact match for *Zip Code* and *Search Area*. Now it is the reconciliation function, OntoMatch (μ) [11], that maps *zip-code* and *radius* to *Zip Code* and *Search Area* and helps the $\hat{\tau}$ operator to execute the form function autotrader.com to submit the control value pairs. Notice that the input relation r of $\hat{\tau}$ has the schema of [*zip-code, radius*] and a tuple of (48202, 50). After the form submission, the extraction function FastWrap extracts a relation r from the Web page. Let's say r has the schema *R* = [*carName, mileage, value, dealer, address, distance, fax*], but the user specifies the extracted schema *S* = [*car, mileage, price, delear, address, distance*]. Thus, finally the transform operator projects S from r, and as we have seen earlier the project operation uses OntoMatch [11] function again to map [*car, mileage, price, dealer, address, distance*] to [*carName, mileage, value, dealer, address, distance*].

D. Interesting Properties of Integra

Although Integra is capable of computing queries that relational algebra in general cannot, it turns out that Integra is no more expressive than relational algebra in its strict sense. This holds true only when we consider the set of functions μ, and κ as part of relational algebra for preprocessing. We can actually show that a mapping function ϑ exists such that we can translate every Integra database into a crisp relational database (i.e., $\vartheta(\mathcal{R}) = \mathcal{R}'$ where $\mathcal{R}$ are the Integra tables and $\mathcal{R}'$ are the corresponding relational tables), and responses generated on $\mathcal{Q}$ on $\mathcal{R}$ by Integra are exactly identical to the response generated by relational algebra on $\mathcal{R}'$ corresponding to $\vartheta(\mathcal{Q})$, as shown in the commutative diagram in figure 4. Based on this observation, we can propose the following two results:

Theorem 3.1: The expressive power of Integra is equivalent to relational algebra modulo the transformation function ϑ, i.e., $\mathcal{Q}(\mathcal{R}) = \vartheta(\mathcal{Q})(\vartheta(\mathcal{R}))$.

Proof Outline: We show that for every relation $r \in \mathcal{R}$, ϑ applies μ to reduce r into a canonical relational representation $r' \in \mathcal{R}'$. We can then transform the query in a similar manner by restricting the attributes and values to the canonical representation. Since the databases and queries are

semantically equivalent under μ, the responses produced by both languages must also be equivalent. The proof in the other direction is symmetric. Only special case we will have to keep in mind that the operators link ($\hat{\lambda}$) and combine ($\hat{\chi}$) are special to Integra, and their counterparts are join and union in relational algebra, and we need to account for that switch in ϑ. A weaker result, i.e., $\mathcal{Q}(\mathcal{R}) \equiv \vartheta(\mathcal{Q})(\vartheta(\mathcal{R}))$, will result if we do not choose to map to a canonical database and thereby reduce cost of mapping.

A corollary of the above result is that the relational algebra is a special case of Integra. This is because if the databases are already crisp (no schema heterogeneity exists), the special functions μ and κ has no effect on the operators and behave as identity functions. In those cases, Integra behaves exactly like relational algebra. In particular, link and combine reduces to join and union respectively.

Corollary 3.1: If $\vartheta(\mathcal{R}) = \mathcal{R}$, then $\mathcal{Q}(\mathcal{R}) = \vartheta(\mathcal{Q})(\mathcal{R})$.

The outline of the translation function can be summarized as follows:

Extended select ($\hat{\sigma}^{\mu}$) and project ($\hat{\pi}^{\mu}$) operation: Let, $\hat{\sigma}^{\mu}_{A_i \theta T'}(r)$ and $\hat{\pi}^{\mu}_{A'_i}(r)$ be two integra expression over $r(R)$. The translation algorithm ϑ takes the relation r and applies the μ function to change all *a-terms* A_i to A'_i where $A_i \simeq^{\mu} A'_i$, and all *v-terms* v to v' where $v \simeq^{\mu} v'$. Let, the changed relation is r'. Now the following relational algebraic expressions $\sigma_{A_i \theta T'}(r')$ and $\pi_{A'_i}(r')$ produces the same output as the original one.

Combine ($\hat{\chi}^{\mu}_{\kappa}$), link ($\hat{\lambda}^{\mu}_{\kappa}$), minus ($\hat{-}^{\mu}_{\kappa}$) and intersection ($\hat{\cap}^{\mu}_{\kappa}$) operation: Let r and s be two combine compatible relations. The translation algorithm applies the μ function and changes all B_is with A_is where $< A_i, B_i > \in \mu(R, S)$ and all $v' : B_i \in s$ with $v : A_i \in r$ such that $v \simeq^{\mu} v'$. After the transformation of r and s to r' and s' we can rewrite the integra expressions as the following:

$$
\begin{aligned}
r\hat{\chi}^{\mu}_{\kappa}s &\equiv (r' \bowtie s') \cup ((r - \pi_R(r' \bowtie s')) \times \langle \perp : (S - R) \rangle) \\
&\quad \cup (\langle \perp : (R - S) \times (s' - \pi_S(r' \bowtie s')) \rangle) \\
r\hat{\lambda}^{\mu}_{\kappa}s &\equiv r' \bowtie s' \\
r\hat{-}^{\mu}_{\kappa}s &\equiv r' - (r' \bowtie (\pi_{A_1 \ldots A_k}(s'))) \\
r\hat{\cap}^{\mu}_{\kappa}s &\equiv r' \bowtie s'
\end{aligned}
$$

Though, both intersection and link are expressed with join operation, the intersection is one-to-one and link is one-to-many join operation. The expression $\langle \perp : (S - R) \rangle$ represents constant relation, which has a single tuple with *null* values for all attributes present in $S - R$.

It is evident from the discussion above that all Integra operators take relations as inputs (along with appropriate functions) and transform them again into relations as required to satisfy the closure property. Once closure is satisfied, compositionality is also guaranteed because we can now nest operations to arbitrary depths as needed by an application.

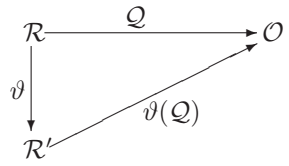


Figure 4. Commutative Diagram

IV. BioFlow

It is perhaps instructive to see how the database operators we have discussed in this paper are used in the higher level declarative statements in BioFlow, for which the algebra is being proposed. In this section we give brief overview on BioFlow, however for detailed syntax interested readers are referred to [1], [2].

Like SQL, BioFlow has two types of statements, resource description statements and data manipulation statements. Resource description statements define necessary datatables and web functions. Data manipulation statements are derived from Integra operators.

In BioFlow we treat every Web page as function, which takes inputs into its input from and returns a relation. We call it Web function and define it declaratively as the following:

```
define function AutoTrader
 extract car varchar(80), mileage integer, price
   integer, dealer varchar(50), address varchar(80),
   distance float
 using matcher OntoMatch, wrapper FastWrap
 from URL www.autotrader.com
 submit (distance varchar(20), zipcode integer);
```

In this definition of AutoTrader Web function, we treat the remote site at URL www.autotrader.com as a form function, where the system submits a *distance* and *zipcode* value, and extracts a relation named AutoTrader with attributes *car, mileage, price, dealer, address* and *distance*. The function specifies the matcher OntoMatch and key identifier GORDIAN by the using clause.

We execute the Web function using the following data manipulation statement, the call statement. During the execution, the system translates the call statement and the Web function to transform operator, where the input relation has the scheme from submit clause and the tuple from the call parameters. The system then executes the transform operator in a sequence of steps as described in section III-C6.

```
call AutoTrader("50 miles", 48202);
```

Other data manipulation statements derived from the Integra operators are listed below as $\hat{\sigma}, \hat{\pi}, \hat{\chi}, \hat{\lambda}, \hat{-}$ and $\hat{\cap}$:

```
(1) select * from AutoTrader where address =
    'Warren, MI' using matcher OntoMatch ;
(2) select car, price, address from CarSearch
    using matcher OntoMatch ;
(3) combine AutoTrader, CarSearch as AutoSearch
    using matcher OntoMatch, identifier GORDIAN;
```

```
(4) link CarSearch, GoogleMap as CarSearchGoogle
    using matcher OntoMatch, identifier GORDIAN ;
(5) select AutoTrader minus CarSearch
    using matcher OntoMatch, identifier GORDIAN ;
(6) select AutoTrader intersection CarSearch
    using matcher OntoMatch, identifier GORDIAN ;
```

V. Related Work

To the best of our knowledge, there have been little to no attempts at developing a comprehensive algebra for semantic data integration for the hidden web. However, there has been previous work on syntactic integration in the direction of schema evolution and schema integration [15], [16]. These research efforts are mostly concerned with syntactic schema manipulation and transformation, and meta data management for data integration. Some other research have focused on specific components of semantic integration. For example, the fusion operator (ϕ) [4] merges *n* input relations into one. The operator is written as $\phi_{F,CR,S}(R)$, where $F = f_1, f_2, ..., f_m$ is a list of m attributes that determines the same real world entity, $CR = cr_1, cr_2, ..., cr_k$ is a list of k conflict resolution functions and $S = s_1, s_2, ..., s_l$ is a list of l intra-group sort key. It first takes outer union of input relations and then fuses data of same real world entities identified by *F* according to the set of conflict resolution functions *CR*. Though the fusion operator consolidates semantic objects from different relations to a single relation, it has to specify the key attributes explicitly, such as F, and different conflict resolution functions for different domain specific purposes. In contrast the integra binary operators automatically identify key attributes and have general purpose reconciliation functions that are domain independent.

VI. Conclusion

Our goal in this paper was to propose a comprehensive algebraic language for semantic data integration on the hidden web. The proposed model stands out among contemporary research efforts in several significant ways. First, it is comprehensive and can serve as a complete query language. Second, the model is abstract and general enough to allow high level abstraction. Such separation allows choice of implementation strategies at the system level. We also demonstrated that Integra is compositional and closed, which not many languages can claim. As a proof of strength of Integra, we have implemented most of its features in our SQL like declarative query language BioFlow [2] for the life sciences data integration project, LifeDB [3].

References

[1] H. Jamil and A. Islam, "The power of declarative languages: A comparative exposition of scientific workflow design using bioflow and taverna," in *IEEE SCC International Workshop on Scientific Workslfows*, Los Angeles, California, July 2009.

[2] H. Jamil and B. El-Hajj-Diab, "Bioflow: A web-based declarative workflow language for life sciences," in *2nd International Workshop on Scientific Workflows*, Hawaii, 2008.

[3] A. Bhattacharjee, A. Islam, M. S. Amin, S. Hossain, S. Hosain, H. M. Jamil, and L. Lipovich, "On-the-fly integration and ad hoc querying of life sciences databases using LifeDB," in *20th International Conference on Database and Expert Systems Applications (DEXA'09)*, Linz, Austria, 2009.

[4] J. Bleiholder, "A relational operator approach to data fusion," in *31st International Conference on Very Large Data Bases (VLDB'05)*, Trondheim, Norway, 2005.

[5] E. Chu, A. Baid, T. Chen, A. Doan, and J. F. Naughton, "A relational approach to incrementally extracting and querying structure in unstructured data," in *33rd international conference on Very large data bases (VLDB'07)*, Vienna, Austria, 2007, pp. 1045–1056.

[6] S. N. Minton, C. Nanjo, C. A. Knoblock, M. Michalowski, and M. Michelson, "A heterogeneous field matching method for record linkage," in *5th IEEE International Conference on Data Mining (ICDM'05)*, 2005, pp. 314–321.

[7] M. S. Amin and H. Jamil, "FastWrap: An efficient wrapper for tabular data extraction from the web," in *IEEE International Conference on Information Reuse and Integration (IRI'09)*, Las Vegas, Nevada, August 2009.

[8] Y. Zhai and B. Liu, "Web data extraction based on partial tree alignment," in *14th international conference on World Wide Web (WWW '05)*, Chiba, Japan, 2005, pp. 76–85.

[9] E. S. Ristad and P. N. Yianilos, "Learning string edit distance," *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 20, no. 5, pp. 522–532, May 1998.

[10] *WordNet*, http://wordnet.princeton.edu/.

[11] A. Bhattacharjee and H. Jamil, "OntoMatch: A monotonically improving schema matching system for autonomous data integration," in *IEEE International Conference on Information Reuse and Integration (IRI'09)*, Las Vegas, August 2009.

[12] Y. Sismanis, P. Brown, P. J. Haas, and B. Reinwald, "GORDIAN: efficient and scalable discovery of composite keys," in *32nd International Conference on Very Large Data Bases*. VLDB Endowment, 2006, pp. 691–702.

[13] A. H. F. Laender, B. Ribeiro-Neto, and A. S. da Silva, "Debye - date extraction by example," *Data Knowl. Eng.*, vol. 40, no. 2, pp. 121–154, 2002.

[14] E. F. Codd, *The relational model for database management: version 2*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990.

[15] L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian, "Logic and algebraic languages for interoperability in multidatabase systems," *Journal of Logic Programming*, vol. 33, no. 2, pp. 101–149, 1997.

[16] C. M. Wyss and F. I. Wyss, "Extending relational query optimization to dynamic schemas for information integration in multidatabases," in *SIGMOD '07*, Beijing, China, 2007, pp. 473–484.