Data Mapping as Search*

George H.L. Fletcher and Catharine M. Wyss

Computer Science Department, School of Informatics,
Indiana University, Bloomington, USA
{gefletch, cmw}@cs.indiana.edu

Abstract. In this paper, we describe and situate the TUPELO system for data mapping in relational databases. Automating the discovery of mappings between structured data sources is a long standing and important problem in data management. Starting from user provided example instances of the source and target schemas, TUPELO approaches mapping discovery as search within the transformation space of these instances based on a set of mapping operators. TUPELO mapping expressions incorporate not only data-metadata transformations, but also simple and complex semantic transformations, resulting in significantly wider applicability than previous systems. Extensive empirical validation of TUPELO, both on synthetic and real world datasets, indicates that the approach is both viable and effective.

1 Introduction

The data mapping problem, automating the discovery of effective mappings between structured data sources, is one of the longest standing problems in data management [17, 24]. Data mappings are fundamental in data cleaning [4, 32], data integration [19], and semantic integration [8, 29]. Furthermore, they are the basic glue for constructing large-scale semantic web and peer-to-peer information systems which facilitate cooperation of autonomous data sources [15]. Consequently, the data mapping problem has a wide variety of manifestations such as schema matching [31, 34], schema mapping [17, 26], ontology alignment [10], and model matching [24, 25].

Fully automating the discovery of data mappings is an "AI-complete" problem in the sense that it is as hard as the hardest problems in Artificial Intelligence [24]. Consequently, solutions have typically focused on discovering restricted mappings such as one-to-one schema matching [31]. More robust solutions to the problem must not only discover such simple mappings, but also facilitate the discovery of the *structural transformations* [18, 39] and *complex (many-to-one) semantic mappings* [8, 14, 29, 31] which inevitably arise in coordinating heterogeneous information systems. We illustrate such mappings in the following scenario.

Example 1. *Consider the three relational databases* `Flights` *A,* B, *and* C *maintaining cost information for airline routes as shown in Fig. 1. These databases, which exhibit three different natural representations of the same information, could be managed by independent travel agencies that wish to share data.*

* The current paper is a continuation of work first explored in poster/demo presentations (IHIS05 and SIGMOD05) and a short workshop paper [11].

FlightsA

Flights:

Carrier	Fee	ATL29	ORD17
AirEast	15	100	110
JetWest	16	200	220

FlightsB

Prices:

Carrier	Route	Cost	AgentFee
AirEast	ATL29	100	15
JetWest	ATL29	200	16
AirEast	ORD17	110	15
JetWest	ORD17	220	16

FlightsC

AirEast:

Route	BaseCost	TotalCost
ATL29	100	115
ORD17	110	125

JetWest:

Route	BaseCost	TotalCost
ATL29	200	216
ORD17	220	236

Fig. 1. Three airline flight price databases, each with the same information content

Note that mapping between the databases in Fig. 1 requires (1) matching schema elements, (2) dynamic data-metadata restructuring, and (3) complex semantic mapping. For example, mapping data from `FlightsB` to `FlightsA` involves (1) matching the `Flights` and `Prices` table names and (2) promoting data values in the `Route` column to attribute names. Promoting these values will dynamically create as many new attribute names as there are `Route` values in the instance of `FlightsB`. Mapping the data in `FlightsB` to `FlightsC` requires (3) a complex semantic function mapping the sum of `Cost` and `AgentFee` to the `TotalCost` column in the relations of `FlightsC`.

1.1 Contributions and Outline

In this paper we present the TUPELO data mapping system for semi-automating the discovery of data mapping expressions between relational data sources (Section 2). TUPELO is an example driven system, generating mapping expressions for interoperation of heterogeneous information systems which involve schema matching, dynamic data-metadata restructuring (Section 2.1), and complex (many-to-one) semantic functions (Section 4). For example, TUPELO can generate the expressions for mapping between instances of the three airline databases in Fig. 1.

Data mapping in TUPELO is built on the novel perspective of mapping discovery as an example driven search problem. We discuss how TUPELO leverages Artificial Intelligence search techniques to generate mapping expressions (Sections 2 and 3). We also present experimental validation of the system on a variety of synthetic and real world scenarios (Section 5) which indicates that the TUPELO approach to data mapping is both viable and effective. We conclude the paper with a discussion of related research (Section 6) and directions for future work (Section 7).

2 Dynamic Relational Data Mapping with TUPELO

In this section we outline the architecture and implementation of the TUPELO system, illustrated in Fig. 2. TUPELO generates an effective mapping from a source relational

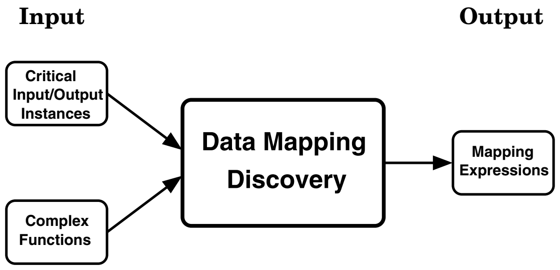


Fig. 2. Data Mapping in the TUPELO System

schema S to a target relational schema T. The system discovers this mapping using (1) example instances s of S and t of T and (2) illustrations of any complex semantic mappings between the schemas. Mapping discovery in TUPELO is a completely syntactic and structurally driven process which does not make use of a global schema or any explicit domain knowledge [2, 16].

We first introduce the mapping language $\mathcal{L}$ used in TUPELO. This language focuses on simple schema matching and structural transformations. We then discuss the Rosetta Stone principle which states that examples of the same information under two different schemas can be used to discover an effective mapping between the schemas. We close the section by describing the idea that drives data mapping in the TUPELO system, namely that data mapping is fundamentally a search problem.

2.1 Dynamic Relational Transformations

TUPELO generates expressions in the transformation language $\mathcal{L}$, a fragment of the Federated Interoperable Relational Algebra (FIRA) [39]. FIRA is a query algebra for the interoperation of federated relational databases. The operators in $\mathcal{L}$ (Table 1) extend the relational algebra with dynamic structural transformations [18, 32, 39]. These include operators for dynamically promoting data to attribute and relation names, a simple merge operator [40], and an operator for demoting metadata to data values. The operators, for example, can express the transformations in Fig. 1 such as mapping the data from `FlightsB` to `FlightsA`.

Example 2. *Consider in detail the transformation from* `FlightsB` *to* `FlightsA`. *This mapping is expressed in* $\mathcal{L}$ *as:*

$R_1 := \uparrow^{\texttt{Cost}}_{\texttt{Route}} (\texttt{FlightsB})$
Promote `Route` values to attribute names with corresponding `Cost` values.

$R_2 := \underline{\pi}_{\texttt{Route}}(\underline{\pi}_{\texttt{Cost}}(R_1))$
Drop attributes `Route` and `Cost`.

$R_3 := \mu_{\texttt{Carrier}}(R_2)$
Merge tuples on `Carrier` values.

$R_4 := \rho^{\texttt{att}}_{\texttt{AgentFee}\to\texttt{Fee}}(\rho^{\texttt{rel}}_{\texttt{Prices}\to\texttt{Flights}}(R_3))$
Rename attribute `AgentFee` to `Fee` and relation `Prices` to `Flights` (i.e., match schema elements).

The output relation R_4 *is exactly* `FlightsA`.

Table 1. Operators for dynamic relational data mapping

Operation	Effect
$\rightarrow_A^B (R)$	*Dereference Column A on B.* $\forall t \in R$, append a new column named B with value $t[t[A]]$.
$\uparrow_B^A (R)$	*Promote Column A to Metadata.* $\forall t \in R$, append a new column named $t[A]$ with value $t[B]$.
$\downarrow (R)$	*Demote Metadata.* Cartesian product of relation R with a binary table containing the metadata of R.
$\wp_A(R)$	*Partition on Column A.* $\forall v \in \pi_A(R)$, create a new relation named v, where $t \in v$ iff $t \in R$ and $t[A] = v$.
$\times(R, S)$	*Cartesian Product* of relation R and relation S.
$\rotatebox[origin=c]{180}{π}_A(R)$	*Drop* column A from relation R.
$\mu_A(R)$	*Merge* tuples in relation R based on compatible values in column A [40].
$\rho_{X \rightarrow X'}^{\texttt{att/rel}}(R)$	*Rename* attribute/relation X to X' in relation R.

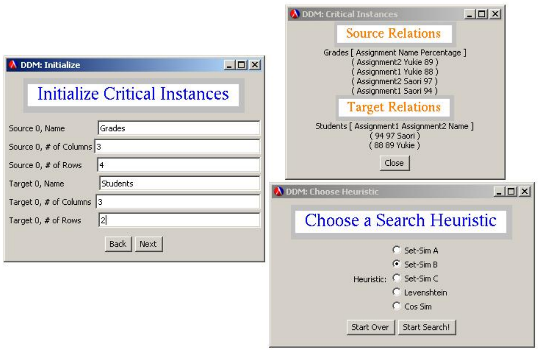
FIRA is complete for the full data-metadata mapping space for relational data sources [39]. The language $\mathcal{L}$ maintains the full data-metadata restructuring power of FIRA. The operators in $\mathcal{L}$ focus on bulk structural transformations (via the $\rightarrow$, $\uparrow$, $\downarrow$, $\wp$, $\times$, $\rotatebox[origin=c]{180}{π}$, and μ operators) and schema matching (via the rename operator ρ). We view application of selections (σ) as a post-processing step to filter mapping results according to external criteria, since it is known that generalizing selection conditions is a nontrivial problem. Hence, TUPELO does not consider applications of the relational σ operator. Note that using a language such as $\mathcal{L}$ for data mapping blurs the distinction between schema *matching* and schema *mapping* since $\mathcal{L}$ has simple schema matching (i.e., finding appropriate renamings via ρ) as a special case.

2.2 The Rosetta Stone Principle

An integral component of the TUPELO system is the notion of "critical" instances s and t which succinctly characterize the structure of the source and target schemas S and T, respectively. These instances illustrate the same information structured under both schemas. The *Rosetta Stone principle* states that such critical instances can be used to drive the search for data mappings in the space of transformations delineated by the operators in $\mathcal{L}$ on the source instance s. Guided by this principle, TUPELO takes as input critical source and target instances which illustrate all of the appropriate restructurings between the source and target schemas.

Example 3. *The instances of the three airline databases presented in Fig. 1 illustrate the same information under each of the three schemas, and are examples of succinct critical instances sufficient for data mapping discovery.*

Critical Instance Input and Encoding. Critical instances can be easily elicited from a user via a visual interface akin to the Lixto data extraction system [13] or visual interfaces developed for interactive schema mapping [1, 3, 26, 37]. In TUPELO, critical


DDM: Initialize

Initialize Critical Instances

Source 0, Name: Grades
Source 0, # of Columns: 3
Source 0, # of Rows: 4
Target 0, Name: Students
Target 0, # of Columns: 3
Target 0, # of Rows: 2

DDM: Critical Instances

Source Relations

Grades [Assignment Name Percentage]
(Assignment2 Yukie 89)
(Assignment1 Yukie 88)
(Assignment2 Saori 97)
(Assignment1 Saori 94)

Target Relations

Students [Assignment1 Assignment2 Name]
(94 97 Saori)
(88 89 Yukie)

DDM: Choose Heuristic

Choose a Search Heuristic

Heuristic: Set-Sim A [✓ Set-Sim B] Set-Sim C Levenshtein Cos Sim

Fig. 3. TUPELO GUI

instances are articulated by a user via a front-end GUI that has been developed for the system (Figure 3). Since critical instances essentially illustrate one entity under different schemas, we also envision that much of the process of generating critical instances can be semi-automated using techniques developed for entity/duplicate identification and record linkage [2, 38].

Critical instances are encoded internally in *Tuple Normal Form* (TNF). This normal form, which encodes databases in single tables of fixed schema, was introduced by Litwin et al. as a standardized data format for database interoperability [23]. TUPELO makes full use of this normal form as an internal data representation format. Given a relation R, the TNF of R is computed by first assigning each tuple in R a unique ID and then building a four column relation with attributes TID, REL, ATT, VALUE, corresponding to tuple ID, relation name, attribute name, and attribute value, respectively. The table is populated by placing each tuple in R into the new table in a piecemeal fashion. The TNF of a database is the single table consisting of the union of the TNF of each relation in the database.

Example 4. *We illustrate TNF with the encoding of database* `FlightsC`*:*

TID	REL	ATT	VALUE
t_1	`AirEast`	`Route`	`ATL29`
t_1	`AirEast`	`BaseCost`	`100`
t_1	`AirEast`	`TotalCost`	`115`
t_2	`AirEast`	`Route`	`ORD17`
t_2	`AirEast`	`BaseCost`	`110`
t_2	`AirEast`	`TotalCost`	`125`
t_3	`JetWest`	`Route`	`ATL29`
t_3	`JetWest`	`BaseCost`	`200`
t_3	`JetWest`	`TotalCost`	`216`
t_4	`JetWest`	`Route`	`ORD17`
t_4	`JetWest`	`BaseCost`	`220`
t_4	`JetWest`	`TotalCost`	`236`

The TNF of a relation can be built in SQL using the system tables. The benefits of normalizing the input instances in this manner with a fixed schema include (1) ease and uniformity of handling of the data, (2) both metadata and data can be handled directly in SQL, and (3) sets of relations are encoded as single tables, allowing natural multi-relational data mapping from databases to databases.

2.3 Data Mapping as a Search Problem

In TUPELO the data mapping problem is seen fundamentally as a search problem. Given critical instances s and t of the source and target schemas, data mapping is an exploration of the transformation space of $\mathcal{L}$ on the source instance s. Search successfully terminates when the target instance t is located in this space. Upon success, the transformation path from the source to the target is returned. This search process is illustrated in Figure 4. The branching factor of this space is proportional to $|s| + |t|$; however intelligent exploration of the search space greatly reduces the number of states visited, as we discuss next.

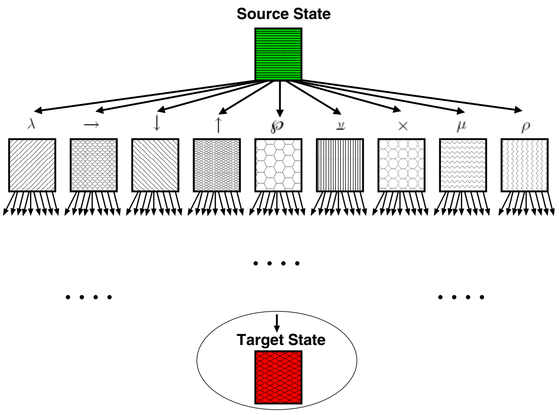


Fig. 4. Search space for data mapping discovery

Heuristic Search Algorithms. Due to their simplicity and effectiveness, we chose to implement the heuristic based *Iterative Deepening A** (IDA) and *Recursive Best-First Search* (RBFS) search algorithms from the Artificial Intelligence literature [28]. In the heuristic exploration of a state space, both of these algorithms use a heuristic function to rank states and selectively search the space based on the rankings. The evaluation function f for ranking a search state x is calculated as $f(x) = g(x) + h(x)$, where $g(x)$ is the number of transformations applied to the start state to get to state x and $h(x)$ is an educated guess of the distance of x from the target state. Search begins at the source critical instance s and continues until the current search state is a structurally identical superset of the target critical instance t (i.e., the current state contains t). The transformation path from s to t gives a basic mapping expression in $\mathcal{L}$. After this expression has been discovered, filtering operations (via relational selections σ) must be applied if necessary according to external criteria, as discussed in Section 2.1. The final output of TUPELO is an expression for mapping instances of the source schema to corresponding instances of the target schema.

The two search algorithms used in TUPELO operate as follows. IDA performs a depth-bounded depth-first search of the state space using the f-rankings of states as the depth bound, iteratively increasing this bound until the target state is reached [28]. RBFS performs a localized, recursive best-first exploration of the state space, keeping track of a locally optimal f-value and backtracking if this value is exceeded [28]. Each of these algorithms uses memory linear in the depth of search; although they both perform redundant explorations, they do not suffer from the exponential memory use of basic A* best-first search which led to the ineffectiveness of early implementations of TUPELO. Furthermore, they both achieve performance asymptotic to A*.

Simple Enhancements to Search. To further improve performance of the search algorithms, we also employed the simple rule of thumb that "obviously inapplicable" transformations should be disregarded during search. For example if the current search state has all attribute names occurring in the target state, there is no need to explore applications of the attribute renaming operator. We incorporated several such simple rules in TUPELO.

3 Search Heuristics

Heuristics are used to intelligently explore a search space, as discussed in Section 2.3. A search heuristics $h(x)$ estimates the distance, in terms of number of intermediate search states, of a given database x from a target database t. A variety of heuristics were implemented and evaluated. This section briefly describes each heuristic used in TUPELO.

Set Based Similarity Heuristics. Three simple heuristics measure the overlap of values in database states. Heuristic h_1 measures the number of relation, column, and data values in the target state which are missing in state x:

$$\begin{aligned} h_1(x) = \quad & |\pi_{\mathsf{REL}}(t) - \pi_{\mathsf{REL}}(x)| \\ & + |\pi_{\mathsf{ATT}}(t) - \pi_{\mathsf{ATT}}(x)| \\ & + |\pi_{\mathsf{VALUE}}(t) - \pi_{\mathsf{VALUE}}(x)|. \end{aligned}$$

Here, π is relational projection on the TNF of x and t, and $|x|$ is the cardinality of relation x. Heuristic h_2 measures the minimum number of data promotions ($\uparrow$) and metadata demotions ($\downarrow$) needed to transform x into the target t:

$$\begin{aligned} h_2(x) = \quad & |\pi_{\mathsf{REL}}(t) \cap \pi_{\mathsf{ATT}}(x)| \\ & + |\pi_{\mathsf{REL}}(t) \cap \pi_{\mathsf{VALUE}}(x)| \\ & + |\pi_{\mathsf{ATT}}(t) \cap \pi_{\mathsf{REL}}(x)| \\ & + |\pi_{\mathsf{ATT}}(t) \cap \pi_{\mathsf{VALUE}}(x)| \\ & + |\pi_{\mathsf{VALUE}}(t) \cap \pi_{\mathsf{REL}}(x)| \\ & + |\pi_{\mathsf{VALUE}}(t) \cap \pi_{\mathsf{ATT}}(x)|. \end{aligned}$$

Heuristic h_3 takes the maximum of h_1 and h_2 on x:

$$h_3(x) = max\{h_1(x), h_2(x)\}.$$

Databases as Strings: The Levenshtein Heuristic. Viewing a database as a *string* leads to another heuristic. Suppose d is a database in TNF with tuples

$$\langle k_1, r_1, a_1, v_1\rangle, \ldots, \langle k_n, r_n, a_n, v_n\rangle .$$

For each tuple, let $s_i = r_i \star a_i \star v_i$, where $\star$ is string concatenation. Define $\texttt{string}(d)$ to be the string $d_1 \star \cdots \star d_n$, where $d_1, \ldots, d_n$ is a lexicographic ordering of the strings s_i, potentially with repetitions. The *Levenshtein distance* between string x and string y, $L(x, y)$, is defined as the least number of single character insertions, deletions, and substitutions required to transform x into y [20]. Using this metric, we define the following *normalized Levenshtein heuristic*:

$$h_L(x) = round\left(k \frac{L(\texttt{string}(x), \texttt{string}(t))}{max\{|\texttt{string}(x)|, |\texttt{string}(t)|\}} \right)$$

where $|w|$ is the length of string w, $k \geqslant 1$ is a scaling constant (scaling the interval $[0, 1]$ to $[0, k]$), and $round(y)$ is the integer closest to y.

Databases as Term Vectors: Euclidean Distance. Another perspective on a database is to view it as a document vector over a set of terms [36]. Let $A = \{a_1, \ldots, a_n\}$ be the set of tokens occurring in the source and target critical instances (including attribute and relation names), and let

$$D = \{\langle a_1, a_1, a_1\rangle, \ldots, \langle a_n, a_n\, a_n\rangle\}$$

be the set of all n^3 triples over the tokens in A. Given a search database d in TNF with tuples $\langle k_1, r_1, a_1, v_1\rangle, \ldots, \langle k_m, r_m, a_m, v_m\rangle$, define $\bar{d}$ to be the n^3-vector $\langle d_1, \ldots, d_{n^3}\rangle$ where d_i equals the number of occurrences of the ith triple of D in the list

$$\langle r_1, a_1, v_1\rangle, \ldots, \langle r_m, a_m, v_m\rangle .$$

This term vector view on databases leads to several natural search heuristics. The standard Euclidean distance in term vector space from state x to target state t gives us a *Euclidean heuristic* measure:

$$h_E(x) = round\left(\sqrt{\sum_{i=1}^{n} (x_i - t_i)^2} \right)$$

where x_i is the ith element of the database vector $\bar{x}$.

Normalizing the vectors for state x and target t gives a *normalized Euclidean heuristic* for the distance between x and t:

$$h_{|E|}(x) = round\left(k \sqrt{\sum_{i=1}^{n} \left[\frac{x_i}{|\bar{x}|} - \frac{t_i}{|\bar{t}|} \right]^2} \right)$$

where $k \geqslant 1$ is a scaling constant and $|\bar{x}| = \sqrt{\sum_{i=1}^{n} x_i^2}$, as usual.

Databases as Term Vectors: Cosine Similarity. Viewing databases as vectors, we can also define a *cosine similarity heuristic* measure, with scaling constant $k \geqslant 1$:

$$h_{\cos}(x) = round\left(k \left[1 - \frac{\sum_{i=1}^{n} x_i t_i}{|\bar{x}||\bar{t}|} \right] \right)$$

Cosine similarity measures the cosine of the angle between two vectors in the database vector space. If x is very similar to the target t, $h_{\cos}$ returns a low estimate of the distance between them.

4 Supporting Complex Semantic Mappings

The mapping operators in the language $\mathcal{L}$ (Table 1) accommodate dynamic data-metadata structural transformations and simple one-to-one schema matchings. However, as mentioned in Section 1, many mappings involve complex semantic transformations [8, 14, 29, 31]. As examples of such mappings, consider several basic complex mappings for bridging semantic differences between two tables.

Example 5. *A semantic mapping f_1 from airline names to airline ID numbers:*

Carrier		CID
AirEast	$\xmapsto{f_1}$	123
JetWest		456

A complex function f_2 which returns the concatenation of passenger first and last names:

Last	First		Passenger
Smith	John	$\xmapsto{f_2}$	John Smith
Doe	Jane		Jane Doe

The complex function f_3 between `FlightsB` *and* `FlightsC` *which maps* `AgentFee` *and* `Cost` *to* `TotalCost`*:*

CID	Route	Cost	AgentFee		CID	Route	TotalCost
123	ATL29	100	15		123	ATL29	115
456	ATL29	200	16	$\xmapsto{f_3}$	456	ATL29	216
123	ORD17	110	15		123	ORD17	125
456	ORD17	220	16		456	ORD17	236

Other examples include functions such as date format, weight, and international financial conversions, and semantic functions such as the mapping from employee name to social security number (which can not be generalized from examples), and so on.

Support for Semantic Mapping Expressions. Any complex semantic function is unique to a particular information sharing scenario. Incorporating such functions in a non-ad hoc manner is essential for any general data mapping solution. Although there has been research on discovering specific complex semantic functions [6, 14], no general approach has been proposed which accommodates these functions in larger mapping expressions.

TUPELO supports discovery of mapping expressions with such complex semantic mappings in a straight-forward manner without introducing any specialized domain knowledge. We can cleanly accommodate these mappings in the system by extending $\mathcal{L}$ with a new operator λ which is parameterized by a complex function f and its input-output signature:

$$\lambda_{f,\bar{A}}^{B}(R).$$

Example 6. *As an illustration of the operator, the mapping expression to apply function f_3 in Example 5 to the values in the `Cost` and `AgentFee` attributes, placing the output in attribute `TotalCost`:*

$$\lambda_{f_3,\mathtt{Cost},\ \mathtt{AgentFee}}^{\mathtt{TotalCost}}(\mathtt{FlightsB}).$$

The semantics of λ is as follows: for each tuple T in relation R, apply the mapping f to the values of T on attributes $\bar{A} = \langle A_1, \ldots, A_n \rangle$ and place the output in attribute B. The operator is well defined for any tuple T of appropriate schema (and is the identity mapping on T otherwise). Note that this semantics is independent of the actual mechanics of the function f. Function symbols are assumed to come from a countably infinite set $\mathbb{F} = \{f_i\}_{i=0}^{i=\infty}$.

Discovery of Semantic Mapping Expressions. TUPELO generates data mapping expressions in $\mathcal{L}$. Extending $\mathcal{L}$ with the λ operator allows for the discovery of mapping expressions with arbitrary complex semantic mappings. Given critical input/output instances and indications of complex semantic correspondences f between attributes $\bar{A}$ in the source and attribute B in the target, the search is extended to generate appropriate mapping expressions which also include the λ operator (Figure 4).

For the purpose of searching for mapping expressions, λ expressions are treated just like any of the other operators. During search all that needs to be checked is that the applications of functions are well-typed. The system does not need any special semantic knowledge about the symbols in $\mathbb{F}$; they are treated simply as "black boxes" during search. The actual "meaning" of a function f is retrieved during the execution of the mapping expression on a particular database instance, perhaps maintained as a stored procedure. Apart from what can be captured in search heuristics, this is probably the best that can be hoped for in general semantic integration. That is, all data semantics from some external sources of domain knowledge must be either encapsulated in the functions f or somehow introduced into the search mechanism, for example via search heuristics.

This highlights a clear separation between semantic functions which interpret the symbols in the database, such as during the application of functions in $\mathbb{F}$, and syntactic, structural transformations, such as those supported by generic languages like $\mathcal{L}$. This separation also extends to a separation of labor in data mapping discovery: discovering particular complex semantic functions and generating executable data mapping expressions are treated as two separate issues in TUPELO.

Discovering complex semantic functions is a difficult research challenge. Some recent efforts have been successful in automating the discovery of restricted classes of complex functions [6, 14]. There has also been some initial research on optimization of mapping expressions which contain executable semantic functions [4].

Focusing on the discovery of data mapping expressions, TUPELO assumes that the necessary complex functions between the source and target schemas have been discovered and that these correspondences are articulated on the critical instance inputs to the system (Fig. 2). These correspondences can be easily indicated by a user via a visual interface, such as those discussed in Section 2.2. Internally, complex semantic maps are just encoded as strings in the VALUE column of the TNF relation. This string indicates the input/output type of the function, the function name, and the example function values articulated in the input critical instance.

5 Empirical Evaluation

The TUPELO system has been fully implemented in Scheme. In this section we discuss extensive experimental evaluations of the system on a variety of synthetic and real world data sets. Our aim in these experiments was to explore the interplay of the IDA and RBFS algorithms with the seven heuristics described in Section 3. We found that overall RBFS had better performance than IDA. We also found that heuristics h_1, h_3, normalized Euclidean, and Cosine Similarity were the best performers on the test data sets.

Experimental Setup. All evaluations were performed on a Pentium 4 (2.8 GHz) with 1.0 GB main memory running Gentoo Linux (kernel 2.6.11-gentoo-r9) and Chez Scheme (v6.9c). In all experiments, the performance measure is the number of states examined during search. We also included the performance of heuristic h_0 for comparison with the other heuristics. This heuristic is constant on all values $(\forall x, h_0(x) = 0)$ and hence induces brute-force blind search. Through extensive empirical evaluation of the heuristics and search algorithms on the data sets described below, we found that the following values for the heuristic scaling constants k give overall optimal performance:

	Norm. Euclidean	Cosine Sim.	Levenshtein
IDA	$k = 7$	$k = 5$	$k = 11$
RBFS	$k = 20$	$k = 24$	$k = 15$

These constant k values were used in all experiments presented below.

5.1 Experiment 1: Schema Matching on Synthetic Data

In the first experiment, we measured the performance of IDA and RBFS using all seven heuristics on a simple schema matching task.

Data Set. Pairs of schemas with $n = 2, \ldots, 32$ attributes were synthetically generated and populated with one tuple each illustrating correspondences between each schema:

$$\left\langle \frac{\texttt{A1}}{\texttt{a1}}, \frac{\texttt{B1}}{\texttt{a1}} \right\rangle \left\langle \frac{\texttt{A1 A2}}{\texttt{a1 a2}}, \frac{\texttt{B1 B2}}{\texttt{a1 a2}} \right\rangle \cdots \left\langle \frac{\texttt{A1} \cdots \texttt{A32}}{\texttt{a1} \cdots \texttt{a32}}, \frac{\texttt{B1} \cdots \texttt{B32}}{\texttt{a1} \cdots \texttt{a32}} \right\rangle$$

Each algorithm/heuristic combination was evaluated on generating the correct matchings between the schemas in each pair (i.e., A1$\leftrightarrow$B1, A2$\leftrightarrow$B2, etc.).

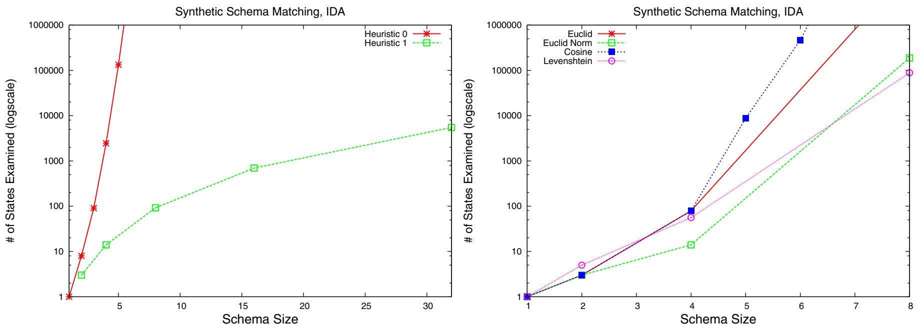


Fig. 5. Number of states examined using IDA for schema matching on synthetic schemas

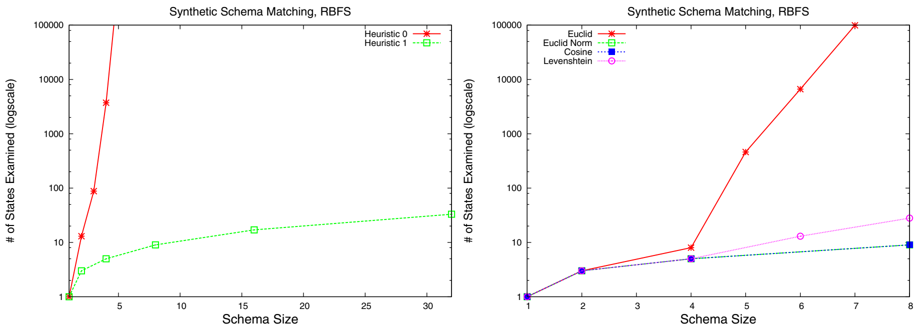


Fig. 6. Number of states examined using RBFS for schema matching on synthetic schemas. Note that the number of states examined using the normalized Euclidean and Cosine Similarity heuristics were identical.

Results. The performance of IDA on this data set is presented in Fig. 5, and the performance of RBFS is presented in Fig. 6. Heuristic h_2 performed identically to h_0, and heuristic h_3's performance was identical to h_1. Hence they are omitted in Figs 5 and 6. RBFS had performance superior to IDA on these schemas, with the h_1, Levenshtein, normalized Euclidean, and Cosine Similarity heuristics having best performance.

5.2 Experiment 2: Schema Matching on the Deep Web

In the second experiment we measured the performance of IDA and RBFS using all seven heuristics on a set of over 200 real-world query schemas extracted from deep web data sources [5].

Data Set. The Books, Automobiles, Music, and Movies (BAMM) data set from the UIUC Web Integration Repository[1] contains 55, 55, 49, and 52 schemas from deep web query interfaces in the Books, Automobiles, Music, and Movies domains, respectively. The schemes each have between 1 and 8 attributes. In this experiment, we populated

[1] http://metaquerier.cs.uiuc.edu/repository, last viewed 26 Sept 2005.

the schemas of each domain with critical instances. We then measured the average cost of mapping from a fixed schema in each domain to each of the other schemas in that domain.

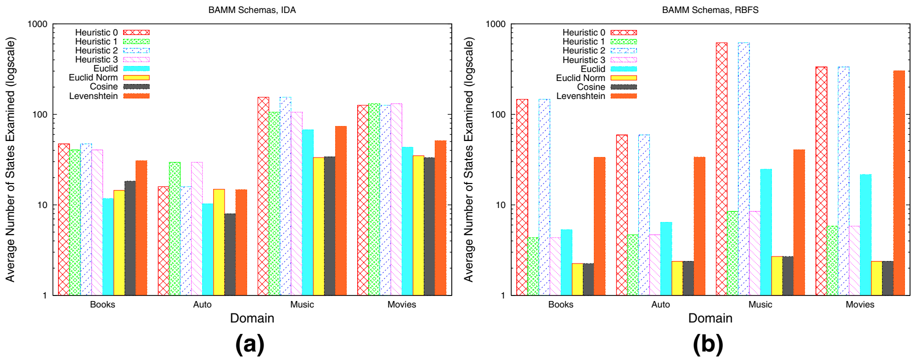


Fig. 7. Average number of states examined for mapping discovery in the four BAMM Domains using (a) IDA and (b) RBFS

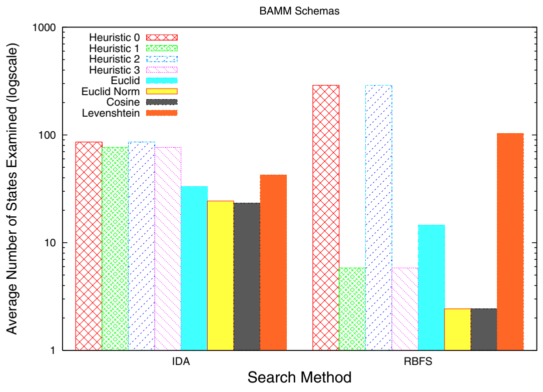


Fig. 8. Average number of states examined for mapping discovery across all BAMM domains

Results. The average performance of IDA on each of the BAMM domains is presented in Fig. 7 (a). Average RBFS performance on each of the BAMM domains is given in Fig. 7 (b). The average performance of both algorithms across all BAMM domains is given in Fig. 8. We found that RBFS typically examined fewer states on these domains than did IDA. Overall, we also found that the Cosine Similarity and normalized Euclidean heuristics had the best performance.

5.3 Experiment 3: Real World Complex Semantic Mapping

In the third experiment we evaluated the performance of TUPELO on discovering complex semantic mapping expressions for real world data sets in the real estate and business inventory domains.

Data Set. We measured performance of complex semantic mapping with the schemas for the Inventory and Real Estate II data sets from the Illinois Semantic Integration Archive.[2] In the Inventory domain there are 10 complex semantic mappings between the source and target schemas, and in the Real Estate II domain there are 12. We populated each source-target schema pair with critical instances built from the provided datasets.

Results. The performance on both domains was essentially the same, so we present the results for the Inventory schemas. The number of states examined for mapping discovery in this domain for increasing numbers of complex semantic functions is given in Fig. 9. On this data, we found that RBFS and IDA had similar performance. For the heuristics, the best performance was obtained by the h_1, h_3 and cosine similarity heuristics.

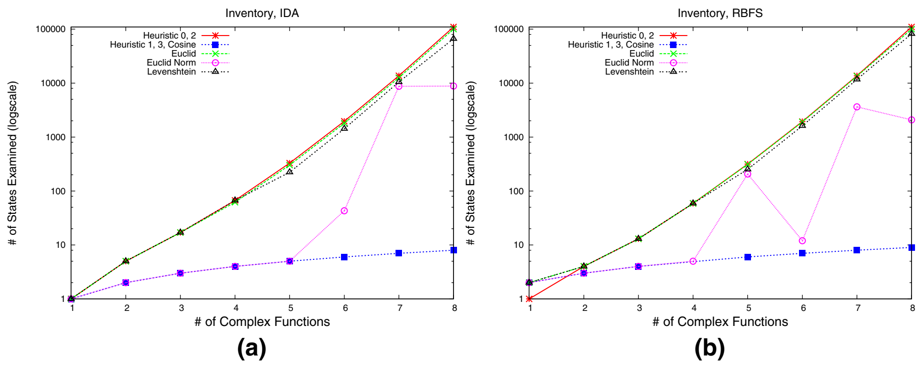


Fig. 9. Number of states for complex semantic mapping discovery in the Inventory domain using (a) IDA and (b) RBFS

5.4 Discussion of Results

The goal of the experiments discussed in this section was to measure the performance of TUPELO on a wide variety of schemas. We found that TUPELO was effective for discovering mapping expressions in each of these domains, even with the simple heuristic search algorithms IDA and RBFS. It is clear from these experiments that RBFS is in general a more effective search algorithm than IDA. Although we found that heuristic h_1 exhibited consistently good performance, it is also clear that there was no perfect all-purpose search heuristic. TUPELO has also been validated and shown effective for examples involving the data-metadata restructurings illustrated in Fig. 1 [11]. It was found in that domain that no particular heuristic had consistently superior performance. We can conclude from these observations that work still needs to be done on developing more intelligent search heuristics.

6 Related Work

The problem of overcoming structural and semantic heterogeneity has a long history in the database [8] and Artificial Intelligence [29] research communities. In Section 1

[2] http://anhai.cs.uiuc.edu/archive/, last viewed 26 Sept 2005.

we have already situated TUPELO in the general research landscape of the data mapping problem. We now briefly highlight related research not discussed elsewhere in the paper:

– *Schema Matching.* A wide variety of existing systems have leveraged Artificial Intelligence techniques for solving different aspects of schema matching and mapping. These include neural networks, Bayesian learning, and genetic programming approaches [7, 22, 27, 33]. The TUPELO view on data mapping as search complements this body of research; this view also complements the characterization of schema matching as constraint satisfaction proposed by Smiljanic et al. [35].
– *Data-Metadata Transformations.* Few data mapping systems have considered the data-metadata structural transformations used in the TUPELO mapping language $\mathcal{L}$. Systems that have considered some aspects of these transformations include [6, 9, 26].
– *Example-Driven Data Mapping.* The notion of example-based data mapping is an ancient idea, by some accounts dating back to the 4th century [30]. Recent work most closely related to the example driven approach of TUPELO include [21, 30, 33].
– *Executable Mapping Expressions.* Most schema matching systems do not address the issue of generating executable mapping expressions, which is in general considered to be an open hard problem [24]. Several notable systems that do generate such expressions include [1, 25, 26, 33].

TUPELO complements and extends this research by (1) attacking the data mapping problem as a basic search problem in a state space and by (2) addressing a broader class of mapping expressions including data-metadata transformations and complex semantics functions. We have also initiated a formal investigation of various aspects of the data mapping problem for relational data sources [12].

7 Conclusions and Future Work

In this paper we presented and illustrated the effectiveness of the TUPELO system for discovering data mapping expressions between relational data sources. Novel aspects of the system include (1) example-driven generation of mapping expressions which include data-metadata structural transformations and complex semantic mappings and (2) viewing the data mapping problem as fundamentally a sarch problem in a well defined search space. Mapping discovery is performed in TUPELO using only the syntax and structure of the input examples without recourse to any domain-specific semantic knowledge. The implementation of TUPELO was described and the viability of the approach illustrated on a variety of synthetic and real world schemas.

There are several promising avenues for future work on TUPELO. As is evident from the empirical evaluation presented in Section 5, further research remains on developing more sophisticated search heuristics. The Levenshtein, Euclidean, and Cosine Similarity based search heuristics mostly focus on the content of database states. Successful heuristics must measure both content and structure. Is there a good multi-purpose search heuristic? Also, we have only applied straightforward approaches to search with the

IDA and RBFS algorithms. Further investigation of search techniques developed in the AI literature is warranted. Finally, the perspective of data mapping as search is not limited to relational data sources. In particular, the architecture of the TUPELO system can be applied to the generation of mapping expressions in other mapping languages and for other data models. Based on the viability of the system for relational data sources, this is a very promising area for future research.

Acknowledgments. We thank the Indiana University database group, Alexander Bilke, Jan Van den Bussche, and Robert Warren for their helpful feedback and support.

References

1. Bernstein, Philip A., et al. Interactive Schema Translation with Instance-Level Mappings (System Demo). *Proc. VLDB Conf.*, pp. 1283-1286, Trondheim, Norway, 2005.
2. Bilke, Alexander and Felix Naumann. Schema Matching using Duplicates. *Proc. IEEE ICDE*, pp. 69-80, Tokyo, Japan, 2005.
3. Bossung, Sebastian, et al. Automated Data Mapping Specification via Schema Heuristics and User Interaction. *Proc. IEEE/ACM ASE*, pp. 208-217, Linz, Austria, 2004.
4. Carreira, Paulo and Helena Galhardas. Execution of Data Mappers. *Proc. ACM SIGMOD Workshop IQIS*, pp. 2-9, Paris, France, 2004.
5. Chang, K. C.-C., B. He, C. Li, M. Patel, and Z. Zhang. Structured Databases on the Web: Observations and Implications. *SIGMOD Record*, 33(3):61-70, 2004.
6. Dhamankar, Robin, et al. iMAP: Discovering Complex Semantic Matches between Database Schemas. *Proc. ACM SIGMOD*, pp. 383-394, Paris, France, 2004.
7. Doan, AnHai, Pedro Domingos, and Alon Halevy. Learning to Match the Schemas of Databases: A Multistrategy Approach. *Machine Learning* 50(3):279-301, 2003.
8. Doan, A., N. Noy, and A. Halevy (Eds). Special Section on Semantic Integration. *SIGMOD Record* 33(4), 2004.
9. Embley, D. W., L. Xu, and Y. Ding. Automatic Direct and Indirect Schema Mapping: Experiences and Lessons Learned. In [8], pp.14-19.
10. Euzenat, Jérôme et al. State of the Art on Ontology Alignment. *Tech. Report D2.2.3, IST Knowledge Web NoE*, 2004.
11. Fletcher, George H.L. and Catharine M. Wyss. Mapping Between Data Sources on the Web. *Proc. IEEE ICDE Workshop WIRI*, Tokyo, Japan, 2005.
12. Fletcher, George H.L., et al. A Calculus for Data Mapping. *Proc. COORDINATION Workshop InterDB*, Namur, Belgium, 2005.
13. Gottlob, Georg, et al. The Lixto Data Extraction Project – Back and Forth between Theory and Practice. *Proc. ACM PODS*, pp. 1-12, Paris, France, 2004.
14. He, Bin, et al. Discovering Complex Matchings Across Web Query Interfaces: A Correlation Mining Approach. *Proc. ACM KDD*, 2004.
15. Ives, Zachary G., Alon Y. Halevy, Peter Mork, and Igor Tatarinov. Piazza: Mediation and Integration Infrastructure for Semantic Web Data. *J. Web Sem.* 1(2):155-175, 2004.
16. Kang, Jaewoo and Jeffrey F. Naughton. On Schema Matching with Opaque Column Names and Data Values. *Proc. ACM SIGMOD*, pp. 205-216, San Diego, CA, 2003.
17. Kolaitis, Phokion G. Schema Mappings, Data Exchange, and Metadata Management. *Proc. ACM PODS*, pp. 61-75, Baltimore, MD, USA, 2005.
18. Krishnamurthy, Ravi, et al. Language Features for Interoperability of Databases with Schematic Discrepancies. *Proc. ACM SIGMOD*, pp. 40-49, Denver, CO, USA, 1991.

19. Lenzerini, Maurizio. Data Integration: A Theoretical Perspective. *Proc. ACM PODS*, pp. 233-246, Madison, WI, 2002.
20. Levenshtein, Vladimir I. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR* 163(4):845-848, 1965.
21. Levy, A.Y., and J.J. Ordille. An Experiment in Integrating Internet Information Sources. *Proc. AAAI Fall Symp. AI Apps. Knowl. Nav. Ret.*, pp. 92-96, Cambridge, MA, USA, 1995.
22. Li, Wen-Syan and Chris Clifton. SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Networks. *Data Knowl. Eng.* 33(1):49-84, 2000.
23. Litwin, Witold, Mohammad A. Ketabchi, and Ravi Krishnamurthy. First Order Normal Form for Relational Databases and Multidatabases. *SIGMOD Record* 20(4):74-76, 1991.
24. Melnik, Sergey. *Generic Model Management: Concepts and Algorithms, LNCS 2967*. Springer Verlag, Berlin, 2004.
25. Melnik, Sergey, et al. Supporting Executable Mappings in Model Management. *Proc. ACM SIGMOD*, Baltimore, MD, USA, 2005.
26. Miller, Renée J., Laura M. Haas, and Mauricio A. Hernández. Schema Mapping as Query Discovery, *Proc. VLDB Conf.*, pp. 77-88, Cairo, Egypt, 2000.
27. Morishima, Atsuyuki, et al. A Machine Learning Approach to Rapid Development of XML Mapping Queries. *Proc. IEEE ICDE*, pp.276-287, Boston, MA, USA, 2004.
28. Nilsson, Nils J. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann, San Francisco, 1998.
29. Noy, N.F., A. Doan, and A.Y. Halevy (Eds). Special Issue on Semantic Integration. *AI Magazine* 26(1), 2005.
30. Perkowitz, Mike and Oren Etzioni. Category Translation: Learning to Understand Information on the Internet. *Proc. IJCAI*, pp. 930-938, Montréal, Canada, 1995.
31. Rahm, Erhard and Philip A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB J.* 10(4):334-350, 2001.
32. Raman, Vijayshankar, and Joseph M. Hellerstein. Potter's Wheel: An Interactive Data Cleaning System. *Proc. VLDB Conf.*, pp. 381-390, Roma, Italy, 2001.
33. Schmid, Ute and Jens Waltermann. Automatic Synthesis of XSL-Transformations from Example Documents. *Proc. IASTED AIA*, Innsbruck, Austria, 2004.
34. Shvaiko, Pavel and Jérôme Euzenat. A Survey of Schema-Based Matching Approaches. *J. Data Semantics* IV, 2005 (to appear).
35. Smiljanic, Marko, et al. Formalizing the XML Schema Matching Problem as a Constraint Optimization Problem. *Proc. DEXA*, Copenhagen, Denmark, 2005.
36. Stephens, D. Ryan. Information Retrieval and Computational Geometry. *Dr. Dobb's Journal* 29(12):42-45, Dec. 2004.
37. Wang, G., J. Goguen, Y.-K. Nam, and K. Lin. Critical Points for Interactive Schema Matching. *Proc. APWeb, Springer LNCS 3007*, pp. 654-664, Hangzhou, China, 2004.
38. Winkler, William E. The State of Record Linkage and Current Research Problems. U.S. Bureau of the Census, Statistical Research Division, Technical Report RR99/04, 1999.
39. Wyss, Catharine M. and Edward L. Robertson. Relational Languages for Metadata Integration. *ACM TODS* 30(2):624-660, 2005.
40. Wyss, Catharine M. and Edward L. Robertson. A Formal Characterization of PIVOT / UNPIVOT. *Proc. ACM CIKM*, Bremen, Germany, 2005.