Principles of Dataspace Systems

Alon Halevy
Google Inc.
Mountain View, CA
halevy@google.com

Michael Franklin
UC Berkeley
Berkeley, CA
franklin@cs.berkeley.edu

David Maier
Portland State University
Portland, Oregon
maier@cs.pdx.edu

ABSTRACT

The most acute information management challenges today stem from organizations relying on a large number of diverse, interrelated data sources, but having no means of managing them in a convenient, integrated, or principled fashion. These challenges arise in enterprise and government data management, digital libraries, "smart" homes and personal information management. We have proposed *dataspaces* as a data management abstraction for these diverse applications and DataSpace Support Platforms (DSSPs) as systems that should be built to provide the required services over dataspaces. Unlike data integration systems, DSSPs do not require full semantic integration of the sources in order to provide useful services. This paper lays out specific technical challenges to realizing DSSPs and ties them to existing work in our field. We focus on query answering in DSSPs, the DSSP's ability to introspect on its content, and the use of human attention to enhance the semantic relationships in a dataspace.

Categories and Subject Descriptors

H.1 [**Models and principles**]; H.2.5 [**Heterogeneous Databases**]

General Terms

Algorithms, Human Factors, Languages, Management

Keywords

dataspaces, personal information management, information retrieval and databases, data integration

1. INTRODUCTION

Most data management scenarios today rarely have a situation in which all the data can be fit nicely into a conventional relational DBMS, or into any other single data model or system. Instead, users and developers are often faced with a set of loosely connected data sources and thus must individually and repeatedly address low-level data management challenges across heterogeneous collections. The first set of challenges are user-facing functions, and include locating relevant data sources, providing search and query capability, and tracing lineage and determining accuracy of the data. The second set of challenges, on the administration side, include enforcing rules, integrity constraints and naming conventions across a collection, providing availability, recovery and access control, and managing the evolution of data and metadata.

Such challenges are ubiquitous – they arise in enterprises (large or small), within and across government agencies, in large science-related collaborations, libraries (digital or otherwise), in battlefields, in "smart" homes, in search for structured content on the WWW, and even on one's PC desktop or other personal devices. In each of these scenarios, however, there is some identifiable scope and control across the data and underlying systems, and hence one can identify a space of data, which, if managed in a principled way, will offer significant benefits to the enterprise.

We recently introduced *dataspaces* as a new abstraction for data management in such scenarios, and proposed the design and development of DataSpace Support Platforms (DSSPs) as a key agenda item for the data management field [24]. In a nutshell, a DSSP offers a suite of interrelated services and guarantees that enables developers to focus on the specific challenges of their applications, rather than on the recurring challenges involved in dealing consistently and efficiently with large amounts of interrelated but disparately managed data. In particular, a DSSP helps to identify sources in a dataspace and inter-relate them, offers basic query mechanisms over them, including the ability to introspect about the contents. A DSSP also provides some mechanisms for enforcing constraints and some limited notions of consistency and recovery.

Traditionally, data integration and data exchange systems [36] have aimed to offer many of the purported services of dataspace systems. In fact, DSSPs can be viewed as the next step in the evolution of data integration architectures, but are distinct from current data integration systems in the following way. Data integration systems require *semantic integration* before any services can be provided. Hence, although there is not a single schema to which all the data conforms and the data resides in a multitude of host systems, the data integration system knows the precise relationships between the terms used in each schema. As a result, significant upfront effort is required in order to set up a data integration system.

Dataspaces are not a data integration approach; rather, they are more of a *data co-existence* approach. The goal of dataspace support is to provide base functionality over all data sources, regardless of how integrated they are. For example, a DSSP can provide keyword search over all of its data sources, similar to that provided by existing desktop search systems. When more sophisticated operations are required, such as relational-style queries, data mining, or monitoring over certain sources, then additional effort can be applied to more closely integrate those sources in an incremental,

PODS'06, June 26–28, 2006, Chicago, Illinois, USA.

"pay-as-you-go" fashion. Similarly, in terms of traditional database guarantees, initially a DSSP can only provide weaker guarantees of consistency and durability. As stronger guarantees are desired, more effort can be put into making agreements among the various owners of data sources, and opening up certain interfaces (e.g., for commit protocols). In a sense, the dataspace approach *postpones* the labor-intensive aspects of data integration until they are absolutely needed.

The following properties distinguish DSSPs from traditional databases and data integration systems:

- A DSSP must deal with data and applications in a wide variety of formats accessible through many systems with different interfaces. A DSSP is required to support *all* the data in the dataspace rather than leaving some out, as with DBMSs.
- Although a DSSP offers an integrated means of searching, querying, updating, and administering the dataspace, often the same data may also be accessible and modifiable through an interface native to the system hosting the data. Thus, unlike a DBMS, a DSSP is not in full control of its data.
- Queries to a DSSP may offer varying levels of service, and in some cases may return *best-effort* or approximate answers. For example, when individual data sources are unavailable, a DSSP may be capable of producing the best results it can, using the data accessible to it at the time of the query.
- A DSSP must offer the tools and pathways to create tighter integration of data in the space as necessary.

Much of the ongoing work in our community is already very relevant to the development of DSSPs. The goal of this paper is to describe several specific challenges to building DSSPs, put them in the context of existing recent work, and propose a set of principles to underly this body of work. This paper is slightly biased towards problems of theoretical nature, and does not attempt to be comprehensive in its coverage of the challenges, and certainly not in its coverage of existing work. The original vision for DSSPs and a description of its proposed components appear elsewhere [24].

Section 2 sets the stage by motivating dataspaces with a few examples. The subsequent sections describe the challenges. Section 3 describes challenges concerning query answering in DSSPs. Section 4 describes the need for a DSSP to introspect on its content and coverage, and Section 5 outlines how a DSSP should benefit from users' actions on dataspaces and reuse them.

2. EXAMPLES

We begin by describing several motivating applications for DSSPs that illustrate some of the main requirements from such systems.

Personal Information Management: The goal of Personal Information Management (PIM) [21, 23, 26, 47] is to offer easy access and manipulation of all of the information on a person's desktop, with possible extension to mobile devices, personal information on the Web, or even all the information accessed during a person's lifetime.

Recent desktop search tools are an important first step for PIM, but are limited to keyword queries. Our desktops typically contain some structured data (e.g., spreadsheets) and there are important associations between disparate items on the desktop. Hence, the next step for PIM is to allow the user to search the desktop in more meaningful ways. For example, "find the list of juniors who took my database course last quarter," or "compute the aggregate balance of my bank accounts." We would also like to search by association, e.g., "find the email that John sent me the day I came back from Hawaii," or "retrieve the experiment files associated with my SIGMOD paper this year." Finally, we would like to query *about* sources, e.g., "find all the papers where I acknowledged a particular grant," "find all the experiments run by a particular student," or "find all spreadsheets that have a variance column."

The principles of dataspaces in play in this example are that (1) a PIM tool must enable accessing *all* the information on the desktop, and not just an explicitly or implicitly chosen subset, and (2) while PIM often involves integrating data from multiple sources, we cannot assume users will invest the time to integrate. Instead, most of the time the system will have to provide best-effort results, and tighter integrations will be created only in cases where the benefits will clearly outweigh the investment.

Scientific data management: Consider a scientific research group working on environmental observation and forecasting, such as the CORIE System[1]. They may be monitoring a coastal ecosystem through weather stations, shore- and buoy-mounted sensors and remote imagery. In addition they could be running atmospheric and fluid-dynamics models that simulate past, current and near-future conditions. The computations may require importing data and model outputs from other groups, such as river flows and ocean circulation forecasts. The observations and simulations are the inputs to programs that generate a wide range of data products, for use within the group and by others: comparison plots between observed and simulated data, images of surface-temperature distributions, animations of salt-water intrusion into an estuary.

Such a group can easily amass millions of data products in just a few years. While it may be that for each file, someone in the group knows where it is and what it means, no one person may know the entire holdings nor what every file means. People accessing this data, particularly from outside the group, would like to search a master inventory that had basic file attributes, such as time period covered, geographic region, height or depth, physical variable (salinity, temperature, wind speed), kind of data product (graph, isoline plot, animation), forecast or hindcast, and so forth. Once data products of interest are located, understanding the lineage is paramount in being able to analyze and compare products: What code version was used? Which finite element grid? How long was the simulation time step? Which atmospheric dataset was used as input?

Soon, such groups will need to federate with other groups to create scientific dataspaces of regional or national scope. They will need to easily export their data in standard scientific formats, and at granularities (sub-file or multiple file) that don't necessarily correspond to the partitions they use to store the data. Users of the federated dataspace may want to see collections of data that cut across the groups in the federation, such as all observations and data products related to water velocity, or all data related to a certain stretch of coastline for the past two months. Such collections may require local copies or additional indices for fast search.

This scenario illustrates several dataspace requirements, including (1) a dataspace-wide catalog, (2) support for data lineage and (3) creating collections and indexes over entities that span more than one participating source.

Structured queries and content on the WWW: While the WWW is dominated by unstructured content, there are also significant and growing opportunities for posing structured queries and obtaining

[1]http://www.ccalmr.ogi.edu

structured content on the web. The deep web, the collection of content residing behind hundreds of thousands of forms, is known to contain a huge amount of very high-quality content. A more recent source of structured data has recently been introduced by GoogleBase [28]. With GoogleBase, anyone can provide content in structured form through a feed interface. The provider supplies a set of *offers* and specifies the class of the offers (e.g., cars, clinical trials, movie reviews, recipes). Each class has (at best) a set of suggested attributes, but providers are free to invent their own classes and attributes. The result is an extremely heterogeneous database about *everything*. Completely reconciling heterogeneity is not even conceivable in this context. The key challenge is to incorporate structured data and queries into the mainstream web-search experience. These web-search challenges highlight two aspects of DSSPs: (1) the need for powerful search mechanisms that accept keyword queries and select relevant structured sources that may answer them, and (2) the ability to combine answers from structured and unstructured data in a principled way.

3. QUERY ANSWERING

The first set of challenges we consider has to do with searching and querying dataspaces. As a background for our discussion, we briefly recount the logical components of dataspaces and the modes in which we expect users to interact with them.

Dataspace participants and relationships: A dataspace should contain all of the information relevant to a particular organization or entity regardless of its format and location, and model a rich collection of relationships between data repositories. Hence, we model a dataspace as a set of *participants* and *relationships*.

The participants in a dataspace are the individual data sources: they can be relational databases, XML repositories, text files, web services and software packages. They can be stored or streamed (managed locally by data stream systems), or even be sensor deployments. We use the terms participants and sources interchangeably.

Some participants may support expressive query languages, while others are opaque and offer only limited interfaces for posing queries (e.g., structured files, web services, or other software packages). Some sources will support traditional updates, while others may be append-only (for archiving purposes), and still others may be immutable.

A dataspace should be able to model any kind of relationship between two (or more) participants. On the more traditional end, we should be able to model that one participant is a view or a replica of another, or to specify a schema mapping between two participants. We would, however, like to model a much broader set of relationships such as, that source A was manually curated from sources B and C, or that sources E and F were created independently, but reflect the same physical system (e.g., mouse DNA). Relationships may be even less specific, such as that two datasets came from the same source at the same time.

Queries: We expect to see queries in a variety of languages. Most activities would probably begin with keyword queries over the dataspace, but it will also be common to see queries resulting from filling forms (which lead to queries with multiple selection predicates). When users interact more deeply with certain data sources, they may pose more complex queries, which may lead to more complex SQL or XQuery queries.

Unless explicitly specified, users typically expect the query to consider all relevant data in the dataspace, regardless of the data model in which it is stored or the schema (if any) along which it is organized. Hence, when a user poses a query in terms of some schema of a particular source (e.g., fields in a form), the expectation is that the system still try to obtain data from other sources as well. Obtaining additional answers will require transformations both on the schema and on the data model.

Answers: There are several ways in which answers to queries posed over dataspaces differ from traditional ones:

- **Ranked:** answers to queries will typically be *ranked* similar to answers from an IR engine or web-search engine. Ranking is necessary not only for keyword queries, but also for structured queries when the translations to other data sources may be approximate.
- **Heterogeneous:** answers will come from multiple sources and will be in different data models and schemas. The ranking needs to handle heterogeneity gracefully.
- **Sources as answers:** in addition to returning actual ground answers to a query, (e.g., documents or tuples), a DSSP can also return *sources*, i.e., pointers to places where additional answers can be found.
- **Iterative:** Interactions with a dataspace are not typically comprised of posing a single query and retrieving an answer. Instead, the user has an information finding task that requires posing a sequence of queries, each being a refinement or modification of the previous ones.
- **Reflection:** we expect a DSSP to reflect on the completeness of its coverage and the accuracy of its data as part of its answers. We postpone these issues to Section 4.
- **In-situ:** Our usual notion of a query answer is that it is a copy of data from a source. However, in asking exploratory queries in a database, we may want "in-situ answers", which are references to, rather than copies of, answer elements. These elements could then be viewed in their original setting, and serve as anchor points for access to surrounding information. The mark and context mechanisms of the SPARCE model [45], for example, could serve as a platform for in-situ answers.

3.1 Query answering model

As a first step in addressing the query answering challenges in dataspaces we need a new formal model of queries and answers. The model needs to account for all the aspects of answers mentioned above.

Some work has already been done on ranking answers in the context of keyword queries over relational databases and over XML documents [3, 4, 9, 30, 33], and on finding relevant information sources in large collections of formally described sources [41]. The reader is referred to [13] for a thought provoking paper on combining DB and IR technologies. However, these works need to be generalized considerably to cases where we do not have semantic mappings of sources and where the data models of the sources differ.

The general challenge we pose is the following:

CHALLENGE 1. *Develop a formal model for studying query answering in dataspaces.* □

The following specific challenge is a first step in this direction:

SUB-CHALLENGE 1.1. *Develop an intuitive semantics for answering a query that takes into consideration a sequence of earlier queries leading up to it.*

In fact, one can go further and postulate that a single query is actually not the right level of granularity to be considering. Users are typically engaged in *tasks* that involve posing several queries as well as other actions (e.g., browsing, creating content). The following challenge attempts to formalize this notion:

SUB-CHALLENGE 1.2. *Develop a formal model of information gathering tasks that include a sequence of lower-level operations on a dataspace.*

We will argue in Section 5 that if we are able to recognize tasks from individual queries, we can develop automatic methods for creating better semantic integration between sources in a dataspace.

Users will interact with a dataspace using a combination of structured and unstructured queries. The following challenges address some of the issues that arise because of that.

SUB-CHALLENGE 1.3. *Develop algorithms that given a keyword query and a large collection of data sources, will rank the data sources according to how likely they are to contain the answer.*

SUB-CHALLENGE 1.4. *Develop methods for ranking answers that are obtained from multiple heterogeneous sources (even when semantic mappings are not available).*

3.2 Obtaining answers

The most significant challenge to answering queries in dataspaces arises from data heterogeneity. It is inevitable that when data sources are authored by different individuals and organizations, they will use different terms to model the same aspects of a domain. Heterogeneity exists both at the schema level and the data level.

A data integration system relies on semantic mappings to reformulate queries from one schema to another. Significant amount of research has focused on developing efficient techniques for reformulation and for understanding how the complexity of reformulation depends on the expressive power of the language used for semantic mappings [31, 36, 39]. However, dataspaces being loosely coupled, DSSPs will typically not have semantic mappings, and even when mappings exist, they may be partial or approximate. Hence, to answer queries in a dataspace we need to shift the attention away from semantic mappings. Specifically, we pose the following challenge, followed by a set of specific proposals on how to approach it.

CHALLENGE 2. *Develop methods for answering queries from multiple sources that do not rely solely on applying a set of correct semantic mappings.* □

Recall that the goal here is *best-effort* query answering, rather that providing exact answers. The following are examples of how to pursue this challenge.

SUB-CHALLENGE 2.1. *Develop techniques for answering queries based on the following ideas, or combinations thereof:*

- *apply several approximate or uncertain mappings and compare the answers obtained by each,*
- *apply keyword search techniques to obtain some data or some constants that can be used in instantiating mappings.*
- *examine previous queries and answers obtained from data sources in the dataspace and try to infer mappings between the data sources. Whenever we have access to queries that span multiple data sources, try to infer from them how the sources are related (e.g., the join attributes should provide some hint of common domains).*

In general, one can approach Challenge 2 by observing the way people would go about searching for information in complex information spaces, and trying to generalize from these patterns and automate them. For example, the following is an illustration of the second bullet of Challenge 2.1, and could be a generalization of real interactions. Suppose we are searching for a particular person's address in a large collection of databases. We begin by posing a keyword query over the dataspace with the person's name as input. The result will point us to tuples in structured sources that include the person's name, and may show us which attributes appear in those tuples. We can then analyze these attribute names and find which are similar or related to address (using schema matching techniques). Finally, we can pose a specific structured query on the corresponding data source with the appropriate attribute names.

In parallel with developing new query-answering algorithms, we need to devise ways to measure their accuracy. Specifically, we face the following challenges:

SUB-CHALLENGE 2.2. *Develop a formal model for approximate semantic mappings and for measuring the accuracy of answers obtained with them.*

Even when there is no heterogeneity, or there is very little of it, we still need to perform mappings between data stored in differing models. For example, multiple data sets may be using essentially the same terminology, but one is stored in XML while another in a relational database or in plain text. Hence, we face the following challenge:

SUB-CHALLENGE 2.3. *Given two data sets that use the same terminology but different data models, develop automatic best-effort methods for translating a query over one data set onto the other.*

The typical example of Challenge 2.3, which has been considered in several works [1, 3, 33] is translating keyword queries onto relational databases. However, there are many other variants of this problem, such as the inverse of the above. Specifically, it is often the case that when users pose a structured query over databases, they also want the system to explore less structured information to obtain related data. Hence an interesting problem to consider is to try to extract from a SQL query a keyword query to pose to an IR-style engine.

4. DATASPACE INTROSPECTION

By its very nature, data in a dataspace will be uncertain and often inconsistent. The aforementioned best-effort query answering mechanisms will introduce additional uncertainty to query answers. Furthermore, answers may differ depending on which level of service we require in terms of latency and completeness. Hence, it is crucial that a DSSP be able to introspect on the answers it presents to its users, and specify the assumptions and lineage underlying them. This section describes several challenges concerning dataspace introspection. We begin in Section 4.1 by discussing introspection with respect to uncertainty, inconsistency and lineage, and then describe introspection more generally in Section 4.2.

While our emphasis here is on introspection for DSSPs, we argue that it is crucial for traditional database systems. In fact, some of the topics we discuss below have initially been investigated in the context of traditional databases. The difference is, however, that while in a traditional database introspection is a feature that is nice to have, in DSSPs it becomes a necessity.

4.1 Lineage, uncertainty and inconsistency

The main point we make in this section is that lineage, uncertainty and inconsistency are highly related to each other and that a DSSP should have a single mechanism that models all three. Specifically, it is often the case that inconsistencies lead to a very particular kind of uncertainty: which of a set of conflicting data values is correct. Both uncertainty and inconsistency need to be ultimately resolved, and lineage is often the only way of doing so.

Our point is not at all a completely new one. The relationship between uncertainty and lineage has recently formed the foundation for the Trio Project [7, 51], and the need to manage inconsistency along with lineage is one of the main ideas underlying the Orchestra Project [35, 49].

We briefly recall the main concepts proposed for modeling uncertainty, inconsistency and lineage. We then discuss some of the challenges that arise in modeling them as in a single formalism. In what follows we refer to the ability of a DSSP to introspect about lineage, uncertainty and inconsistency as *LUI introspection*.[2]

4.1.1 Uncertain databases

Uncertainty arises in data management applications because the exact state of the world is not known. The goal of an uncertain database is to represent a *set* of possible states of the world, typically referred to as *possible worlds*. Each possible world represents a complete state of the database. Hence, a traditional database represents a single state of the world.

Several formalisms have been proposed for uncertain databases [2, 6, 18, 29, 34, 38]). We briefly illustrate three such formalisms that will highlight some of the challenges we address later. We illustrate the formalisms with an example where structured data is extracted from unstructured text, one of the common sources of uncertain data in practice.

A very simple formalism for modeling uncertainty is *a-tuples*. An a-tuple differs from an ordinary tuple in two ways. First, instead of having a single value for an attribute, it may have several values. (For now, we will consider only a finite number of values). Second, the tuple may be a *maybe-tuple*. As an example, consider the following two tuples:

(Karina Powers, { 345-9934 | 345-9935})
(George Flowers, 674-9912) ?

The first tuple states that the phone number of Karina Powers is one of two values, whereas the second tuple is not sure whether George Flowers has a phone number (but if he does, the number is certain). Hence, these two tuples represent four possible states of the world.

A-tuples are easy to comprehend and to visualize. When uncertainty is limited to values of certain attributes, a-tuples capture it well. However, a-tuples are not closed under relational operators [18]. For example, the result of the join of two relations with a-tuples cannot necessarily be represented by a set of a-tuples.

A slightly more general formalism is *x-tuples* (recently studied [7]). An x-tuple is simply a set of ordinary tuples, meant to describe different possible states, and they too can be marked as maybe-tuples. As an example, consider the following x-tuple, where the second column is the person's work phone and the third is the work fax:

(Karina Powers, 345-9934, 345-9935) |
(Karina Powers, 345-9935, 345-9934)

[2]LUI stands for Lineage, Uncertainty and Inconsistency.

The x-tuple represents the fact that we're not sure which number is the work phone and which is the fax, and represents two possible states of the world. While x-tuples are more powerful than a-tuples, they are still not closed under relational operators.

Finally, we illustrate c-tables [2] with the following example.

Karina Powers	456-3214	$x = 1$
Karina Powers	654-1234	$x \neq 1$
Karina Powers	456-4444	$x \neq 1$

C-tables rely on assignments to variables to determine the possible worlds of the database. Intuitively, the variables correspond to different decisions about the state of the world. Any consistent assignment of the variables yields a possible world. In the example above, there are two worlds, depending on the value of x. If $x = 1$, then only the first tuple is in the database. If $x \neq 1$, then the second and third tuples are in. Hence, we can model a constraint saying that if a particular tuple is not in the database, then two other ones must be.

C-tables are closed under relational operators. In fact, c-tables can be shown to be complete. That is, given any set of possible worlds S of a schema R, there exists a database of c-tables D_S whose possible worlds are precisely S. The disadvantage of c-tables is that they are a bit harder to understand as a user. Furthermore, checking whether a set of tuples I is a possible world of D_s is known to be NP-complete.

4.1.2 Inconsistencies in databases

Inconsistent databases are meant to handle situations in which the database contains conflicting data. The most common type of inconsistency is disagreement on single-valued attributes of a tuple. For example, a database storing the salary of an employee may have two different values for the salary, each coming from different sources.

The key idea underlying inconsistent databases is to consider the different possible *repairs* of the database [5]. A repair is a minimal change that results in a consistent database. For example, if we have two tuples specifying the salary of an employee, then removing either of them will result in a consistent database. Hence, a database may have multiple repairs.

Herein lies the close relationship between uncertainty and inconsistency. Inconsistency can be viewed as being uncertain about which of the conflicting values is correct. The set of possible worlds corresponds to the different repairs of the database. In the example of disagreement on non-key attributes, the uncertainty can be expressed as a-tuples.

4.1.3 Modeling data lineage

The lineage of a tuple explains how the tuple was derived to be a member of a particular set. We distinguish *internal* lineage from *external* lineage. Internal lineage applies to tuples in answers to queries – the lineage specifies how a tuple was derived from other tuples in the database. External lineage applies to tuples that were inserted into the database – the lineage refers to the external sources or processes by which they were inserted.

For internal lineage, there is often a rather obvious definition of lineage, which can be defined in terms of the proof tree of a tuple. For example, for conjunctive queries (or even datalog queries without recursion) there is a unique proof tree for every resulting tuple under multi-set semantics. The lineage function maps a tuple to the tuples in the leaves of its proof tree.

For union queries under set semantics the representation of a proof tree is more complicated because it includes OR nodes, but it is still possible to define a natural lineage function. However,

for queries with negation or aggregation, there is no single obvious definition of lineage. (For discussions of possible lineage functions in previous literature see [8, 11, 14, 15, 40].)

4.1.4 LUI Introspection

A DSSP should provide a single unified mechanism for modeling uncertainty, inconsistency and lineage. Broadly, the challenge is the following:

CHALLENGE 3. *Develop formalisms that enable modeling uncertainty, inconsistency and lineage in a unified fashion.* □

We now outline some more specific challenges in this vein.

The relationship between uncertainty and inconsistency is fairly obvious: inconsistency often boils down to being uncertain about which state of the world is the correct one. Hence a specific challenge is the following:

SUB-CHALLENGE 3.1. *Develop formalisms that capture uncertainty about common forms of inconsistency in dataspaces.*

Ideally, existing formalisms or minor variations on them will be appropriate. Furthermore, since inconsistency leads to a very special type of uncertainty (guided by the set of repairs of the database), the uncertainty formalism should be able to leverage that special structure. A particular challenge here will be to understand the kinds of LUI that arise in highly heterogeneous environments.

We claim that lineage plays an important role in resolving uncertainty and inconsistency. The formalisms for representing uncertainty only tell us what are the possible states of the world, and will sometimes assign probabilities to each possible world. However, in many cases the only way to resolve the uncertainty is to know *where* the data came from or how it was derived.

Our goal is that a single formalism should be able to specify *all* of the following statements:

- Joe's age is either 42 or 43
- The probability that Joe is 42 is 0.6 and that he is 43 is 0.4
- According to source A, Joe is 42, and according to source B he is 43
- According to source A, whose probability of being correct is 0.6, Joe is 42, and according to source B, whose probability of being correct is 0.4, he is 43.

In fact, web-search engines already unify uncertainty and lineage in a simple way. Answers returned by the engine are ranked in some way (corresponding to uncertainty about the relevance and the quality of the answer), but they invariably provide the URL of the answers and a snippet, corresponding to the lineage information. Users take into consideration both the ranking and the URL when they decide which links to follow.

One of the main reasons we want to unify lineage and uncertainty is to reason about relationships between external sources and their effects on answers. To achieve that, we have the following challenge:

SUB-CHALLENGE 3.2. *Develop formalisms for representing and reasoning about external lineage.*

As an example, we would like to represent when different data sources are independent or not, and consider that when processing queries, to distinguish reinforcement from redundancy.

To combine uncertainty and lineage, we have the following challenge, initially addressed in [7]:

SUB-CHALLENGE 3.3. *Develop a general technique to extend any uncertainty formalism with lineage, and study the representational and computational advantages of doing so.*

Knowledge of lineage may constrain the set of possible worlds of a database, and therefore may affect the complexity of query answering (hopefully in a positive way, but that's only a conjecture).

The key aspect of achieving Challenge 3.3 is that the object to which we attribute uncertainty must match the one to which we attribute lineage. Lineage is typically associated with individual tuples in the database or entire data sources, and hence, as shown in [7], it is fairly natural to extend x-tuples with lineage information. A-tuples, however, associate uncertainty with attribute values, and therefore the associated lineage needs to be attached to attribute values. In the case of c-tables, the situation is trickier, since variables create complex dependencies among the tuples. Hence, lineage should capture values of variables and dependencies between sets of variables.

Uncertainty on views

Challenge 3.3 leads to a more general issue with modeling uncertainty. Currently, uncertainty formalisms associate uncertainty with a single schematic construct: tuples in the case of x-tuples, and attribute values in the case of a-tuples. Therefore, the choice of database schema and normalization limits the kinds of uncertainty we can express. Consider the example of extracting the tuple from unstructured data.

(Karina Powers, 123 Main St., 345-9934)

Suppose we want to associate a probability of 0.7 with this tuple, given our confidence in the extraction. However, suppose we are more confident of our name extraction, and would like to attach a probability of 0.9 to the first attribute of the tuple. There is currently no way of associating different levels of uncertainty with different parts of the tuple.

We propose to attach uncertainty with tuples in views. In our example, we would be able to associate a higher probability with the projection of the tuple on the first attribute. Note that the views need not be materialized. The challenge can be summarized as follows.

SUB-CHALLENGE 3.4. *Develop formalisms where uncertainty can be attached to tuples in views and view uncertainty can be used to derive uncertainty of other view tuples.*

An excellent first step towards this challenge is described in [17]. Beyond the basic results, we need to identify more tractable cases for such query answering. In addition, views can be used to model probabilistic integrity constraints that induce a probability distribution on the data. Reasoning with such views is an exciting challenge. Several AI formalisms (e.g., probabilistic relational models (PRMs) [27, 37]) were developed to model probabilistic constraints on collections of objects and relationships between them. We need a good understanding of the spectrum of formalisms between probabilistic views to probabilistic relational models.

4.2 Finding the right answers

The ability to introspect about data and query answers raises the next natural question: given all the dimensions along which answers can vary, what are *good* answers to a query?

Candidate answers can differ along multiple dimensions, including:

- relevance to the query,

- certainty of the answer (or whether it contradicts another)
- completeness and precision requested by the user,
- maximum latency required in answering the query.

Hence, broadly, we face the following challenge:

CHALLENGE 4. *Define metrics for comparing the quality of answers and answer sets over dataspaces, and efficient query processing techniques.* □

The concept of minimal repairs for inconsistent databases [5] is a limited version of this challenge. It is unlikely that there is a general theory that trades off all these factors in an application independent manner. In many cases, the particular context in which queries are being posed will offer more guidance on these tradeoffs. To enable specifying such preferences, we pose the following challenge:

SUB-CHALLENGE 4.1. *Develop query-language extensions and their corresponding semantics that enable specifying preferences on answer sets along the dimensions of completeness and precision, certainty and inconsistency, lineage preferences and latency.*

Note that while we expect users to express these preferences with some GUI, we still need a language for expressing these preferences on the backend.

Along with mechanisms for specifying preferences on answer sets, we need methods for *reasoning* about sets of answers so we can compare among them. Traditionally, sets of queries were compared by *query containment* [12]. A query Q_1 is said to contain Q_2 if the answer of Q_1 is always a superset of the answer to Q_2 on any given database instance. The following challenge extends query containment to the context of dataspaces:

SUB-CHALLENGE 4.2. *Define notions of query containment that take into consideration completeness and precision, uncertainty and inconsistency and lineage of answers, and efficient algorithms for computing containment.*

As a specific challenge that ties together Section 4.1 and our current discussion, consider the following:

SUB-CHALLENGE 4.3. *Develop methods for efficient processing of queries over uncertain and inconsistent data that conserve the external and internal lineage of the answers. Study whether existing query processors can be leveraged for this goal.*

This challenge is already under investigation by several projects [16, 17, 25, 35, 51]. The challenge is to extend these techniques to formalisms that model lineage, uncertainty and inconsistency, and to incorporate sophisticated ranking algorithms as part of query processing.

5. REUSING HUMAN ATTENTION

The key tenet underlying DSSPs is that they should provide some level of service on a dataspace from the very beginning. Over time the dataspace should evolve by forming more tight semantic integration between disparate sources as needed.

This section turns the attention to how the dataspace can evolve, and argues that leveraging users' attention as they interact with a dataspace is a crucial and under-utilized element to success in this endeavor. In a nutshell, every time a user performs some operation on the dataspace, she is telling us something about its semantic content or about the relationships between its disparate data sources.

Users (of varying levels) perform a multitude of actions on dataspaces, such as: asking and refining queries, browsing semantically related data items, creating electronic workspaces that aggregate related data, and even lower-level actions, such as copying values from one column of a spreadsheet to another. At a more sophisticated level, the act of creating a schema (however precise) or schema mappings is also one we can learn from. Human attention is very scarce, and hence it is crucial that DSSPs be able to leverage actions that users perform as a side-effect of their daily activities.

Broadly speaking the challenge we put forth in this section is the following:

CHALLENGE 5. *Develop methods that analyze users' activities when interacting with a dataspace and create additional meaningful relationships between sources in a dataspace or other enhancements to the dataspace.* □

Interestingly, some works argue that human attention is *not* scarce and that spare human cycles can be harnessed in interesting ways. For example, von Ahn et al. [50] create an entertaining online game, the side-result of which is massive annotation of images on the web. Despite the differences in opinion concerning human productivity, the same principles are at play in both contexts.

There have been several recent successful examples of learning from human activities to further semantic integration, focused mostly on learning semantic mappings between data sources. The LSD System [20] showed how to learn to map schemas, where the training examples are a set of mappings from data sources to a mediated schema. LSD generalized from these examples and was able to predict with high accuracy mappings of other sources into that mediated schema. Dong et al. [22], Chang and He [32], and Madhavan et al. [42] showed how to analyze large numbers of schemas or web services to learn properties that enabled their systems to automatically guess mappings between unseen sources. McCann et al. [44] took the idea further and proposed collaboration among large sets of users to create semantic integration. All of these works offer only one instance of the general principle we advocate here: they all learn from collections of schemas and mappings to produce better schema matching systems.

The following specific challenge argues that we can learn from users posing queries:

SUB-CHALLENGE 5.1. *Develop techniques that examine collections of queries over data sources and their results to build new mappings between disparate data sources.*

In Section 3 we described the notion of a task that captures several actions on the dataspace that have a common goal. A user may interact with multiple sources while fulfilling a task, and examining the specific actions is likely to reveal interesting semantic relationships between them. Hence, as a first step towards Challenge 5.1 we have the following recognition problem:

SUB-CHALLENGE 5.2. *Develop algorithms for grouping actions on a dataspace into tasks.*

Recall that semantic mappings are only one kind of relationship we may have between sources in a dataspace. Hence, our goal here is to go beyond the reuse of human attention for creating semantic mappings. Any process that creates new semantically meaningful relationships between data sources is valuable, including approximate mappings or even clustering data sources together so a human can later decide whether it is worth investing additional effort to create more precise mappings.

In addition to leveraging users' actions, the system can also take a more proactive approach to soliciting useful semantic information from users:

SUB-CHALLENGE 5.3. *Develop facilities for explicit enhancement of dataspace information that give high return on the investment of human attention.*

It won't always be necessary to leverage human attention implicitly. There will be cases where it is reasonable to expect people to augment a dataspace with explicit labels, links, annotations, classifications, and so forth. In most instances, however, humans will not make such efforts unless there is personal payoff for their activities. The goal is to give immediate and incremental leverage from any enhancements made. One approach is to support superimposing information [43] over dataspace sources, which can be used to overlay structure on subsets of dataspace information [10, 19], which can in turn support new navigation paths for query in the dataspace [46]. Another example of this approach was taken in [48], where the system used an *active learning* method to carefully select questions that helped the system reconcile different references to the same real-world object.

Finally, we have the following theoretical challenge:

SUB-CHALLENGE 5.4. *Develop a formal framework for learning from human attention in dataspaces.*

The formal framework needs to include at least the following components:

- A definition of the specific learning problem. For example, we may try to learn a semantic mapping between two specific data sources, or we may try to learn rules for mapping any pair of sources in a particular domain. (In [20] the goal was to learn rules for mapping an arbitrary source into a fixed mediated schema).
- In either of the cases above, we need a formalism for describing approximate semantic mappings and distances between semantic mappings (i.e., the distance metric for determining the quality of proposed mappings).
- We need to spell out the space of possible mappings the learning problem will consider.
- Finally, we need a way of interpreting our training examples, i.e., a mapping between training examples and constraints on the space. In the simple case, a training example is exactly a semantic mapping in the search space. However, suppose a user writes a pair of queries over two data sources, but uses the same constant in both queries. How do we turn that information into some form of a training example?

As with most theoretical analyses of learning problems that provide generous upper bounds on the complexity of learnability, we also need to understand which constraints on the domain and on the training examples making learning easier in practice.

6. CONCLUSIONS

We described a broad research agenda for building dataspace support platforms that involves a range of theoretical and practical challenges. There are, of course, several areas we did not touch on that also pose important challenges, the primary ones being: supporting relaxed notions of transactions and recovery and considering privacy and security of data.

We conclude with a few remarks on how to approach this agenda as a community. We believe research on DSSPs should be mostly bottom-up and exemplar driven, even when the contributions are of theoretical nature. This orientation has several consequences for interested researchers. First, we should start by addressing the subchallenges and addressing the bigger challenges as experience is gained. Second, we should build an infrastructure into which researchers can connect components of DSSPs as they build them. In fact, interoperability and interchangeability of multiple components is key to the success of DSSPs.

Finally, we should quickly establish one or more public "dataspace sandboxes" in which researchers can play. Each sandbox should have a significant number of distinct data sources, along with appropriate management systems or ancilliary programs for accessing them. Ideally, there would also be a corpus of search, query and task descriptions that represent realistic uses of the dataspace. While there might be an "entry-level" sandbox that a research group can download and install locally, to present dataspace problems in their full generality, a sandbox should include some massive sources and some sources that are being actively updated (and possibly have new sources joining from time to time). Such a sandbox should thus provide local hosting for DSSP components under test. Hopefully, some premier web search companies and scientific research groups can create such sample dataspaces for public consumption.

Acknowledgments

We would like to thank Serge Abiteboul, Roberto Bayardo, Phil Bernstein, Mike Carey, David DeWitt, AnHai Doan, Laura Haas, Zack Ives, Donald Kossmann, Mike Stonebraker, Dan Suciu, Jeff Ullman, Gerhard Weikum, and Stan Zdonik for useful advice and feedback on previous versions of this paper. We would like to thank the members of the Trio Project, and especially Omar Benjelloun, Anish Das Sarma and Jennifer Widom for many discussions regarding management of uncertain data and lineage. The work was funded the following grants: NSF IIS-0534762, IIS-9985114 and IIS-0415175.

7. REFERENCES

[1] Shaul Dar aand Gadi Entin, Shai Geva, and Eran Palmon. DTL's dataspot: Database exploration using plain language. In *Proc. of VLDB*, pages 645–649, 1998.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das. Dbxplorer: A system for keyword-based search over relational databases. In *Proc. of ICDE*, pages 5–16, 2002.

[4] Sihem Amer-Yahia, Nick Koudas, Amlie Marian, Divesh Srivastava, and David Toman. Structure and content scoring for xml. In *Proc. of VLDB*, pages 361–372, 2005.

[5] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent Query Answers in Inconsistent Databases. In *Proc. of ACM PODS*, 1999.

[6] D. Barbará, H. Garcia-Molina, and D. Porter. The Management of Probabilistic Data. *IEEE Trans. Knowl. Data Eng.*, 1992.

[7] O. Benjelloun, A. Das Sarma, A. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. http://dbpubs.stanford.edu/pub/2005-39, 2005.

[8] D. Bhagwat, L. Chiticariu, W. Tan, and G. Vijayvargiya. An annotation management system for relational databases. *Proc. of VLDB*, 2004.

[9] Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *Proc. of ICDE*, pages 431–440, 2002.

[10] Shawn Bowers, Lois M. L. Delcambre, and David Maier. Superimposed schematics: Introducing e-r structure for in-situ information selections. In *ER*, pages 90–104, 2002.

[11] P. Buneman, S. Khanna, and W. Tan. Why and where: A charaterization of data provenance. *Proc. of ICDT*, 2001.

[12] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[13] Surajit Chaudhuri, Raghu Ramakrishnan, and Gerhard Weikum. Integrating db and ir technologies: what is the sound of one hand clapping. In *Proc. of CIDR*, 2005.

[14] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. *VLDB Journal*, 2003.

[15] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM TODS*, 2000.

[16] N. Dalvi and D. Suciu. Efficient Query Evaluation on Probabilistic Databases. In *Proc. of VLDB*, 2004.

[17] N. Dalvi and D. Suciu. Answering Queries from Statistics and Probabilistic Views. In *Proc. of VLDB*, 2005.

[18] A. Das Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working Models for Uncertain Data. In *Proc. of ICDE*, April 2006.

[19] Lois M. L. Delcambre, David Maier, Shawn Bowers, Mathew Weaver, Longxing Deng, Paul Gorman, Joan Ash, Mary Lavelle, and Jason Lyman. Bundles in captivity: An application of superimposed information. In *Proc. of ICDE*, pages 111–120, 2001.

[20] Anhai Doan, Pedro Domingos, and Alon Halevy. Reconciling schemas of disparate data sources: a machine learning approach. In *Proc. of SIGMOD*, 2001.

[21] Xin Dong and Alon Halevy. A Platform for Personal Information Management and Integration. In *Proc. of CIDR*, 2005.

[22] Xin (Luna) Dong, Alon Y. Halevy, Jayant Madhavan, Ema Nemes, and Jun Zhang. Similarity search for web services. In *Proc. of VLDB*, 2004.

[23] S. T. Dumais, E. Cutrell, J. J. Cadiz E., G. Jancke, R. Sarin, and D. C. Robbins. Stuff i've seen: A system for personal information retrieval and re-use. In *SIGIR*, 2003.

[24] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: A new abstraction for information management. *Sigmod Record*, 34(4):27–33, 2005.

[25] Ariel Fuxman, Elham Fazli, and Renee J. Miller. Conquer: efficient management of inconsistent databases. In *Proc. of SIGMOD*, pages 155–166, New York, NY, USA, 2005. ACM Press.

[26] Jim Gemmell, Roger Lueder, and Gordon Bell. Living with a lifetime store. In *Workshop on Ubiquitous Experience Media*, 2003.

[27] Lise Getoor and John Grant. Prl: A logical approach to probabilistic relational models. *Machine Learning Journal*, 62, 2006.

[28] Google.com. Google base. base.google.com, 2005.

[29] G. Grahne. Dependency Satisfaction in Databases with Incomplete Information. In *Proc. of VLDB*, 1984.

[30] Lin Guo, Feng Shao, Chavdar Botev, and Jayavel Shanmugasundaram. XRANK: Ranked keyword search over XML documents. In *Proc. of SIGMOD*, pages 16–27, 2003.

[31] Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4), 2001.

[32] Bin He and Kevin Chen-Chuan Chang. Statistical schema integration across the deep web. In *Proc. of SIGMOD*, 2003.

[33] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. Efficient ir-style keyword search over relational databases. In *Proc. of VLDB*, pages 850–861, 2003.

[34] T. Imielinski and W. Lipski Jr. Incomplete Information in Relational Databases. *Journal of the ACM*, 1984.

[35] Z. G. Ives, N. Khandelwal, A. Kapur, and M. Cakir. Orchestra: Rapid, collaborative sharing of dynamic data. In *Proc. of CIDR*, 2005.

[36] Phokion Kolaitis. Schema mappings, data exchange, and metadata management. In *Proc. of ACM PODS*, pages 61–75, 2005.

[37] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 580–587, Madison, WI, 1998. AAAI Press.

[38] L. V. S. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian. ProbView: A Flexible Probabilistic Database System. *ACM TODS*, 1997.

[39] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS*, 2002.

[40] A. Y. Levy, R. E. Fikes, and S. Sagiv. Speeding up inferences using relevance reasoning: A formalism and algorithms. *Artificial Intelligence*, 1997.

[41] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB*, pages 251–262, Bombay, India, 1996.

[42] Jayant Madhavan, Philip A. Bernstein, AnHai Doan, and Alon Halevy. Corpus-based schema matching. In *Proc. of ICDE*, pages 57–68, 2005.

[43] David Maier and Lois M. L. Delcambre. Superimposed information for the internet. In *WebDB*, pages 1–9, 1999.

[44] R. McCann, A. Doan, A. Kramnik, and V. Varadarajan. Building data integration systems via mass collaboration. In *Proc. of the SIGMOD-03 Workshop on the Web and Databases (WebDB-03)*, 2003.

[45] Sudarshan Murthy, Lois M. L. Delcambre, David Maier, and Shawn Bowers. Putting integrated information in context: Superimposing conceptual models with sparce. In *APCCM*, pages 71–80, 2004.

[46] Sudarshan Murthy, David Maier, and Lois M. L. Delcambre. Querying bi-level information. In *WebDB*, pages 7–12, 2004.

[47] Dennis Quan, David Huynh, and David R. Karger. Haystack: a platform for authoring end user semantic web applications. In *ISWC*, 2003.

[48] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *SIGKDD*, 2002.

[49] Nicholas E. Taylor and Zachary G. Ives. Reconciling while tolerating disagreement in collaborative data sharing. In *Proc. of SIGMOD*, 2006.

[50] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of ACM CHI, Vienna, Austria*, 2004.

[51] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In *Proc. of CIDR*, 2005.