Chapter 1

UNCERTAINTY IN DATA INTEGRATION

Anish Das Sarma
Stanford University
California, USA
anish@cs.stanford.edu

Xin Dong
AT&T Labs-Research
New Jersey, USA
lunadong@research.att.com

Alon Halevy
Google Inc.
California, USA
halevy@google.com

Abstract Data integration has been an important area of research for several years. In this chapter, we argue that supporting modern data integration applications requires systems to handle uncertainty at every step of integration. We provide a formal framework for data integration systems with uncertainty. We define probabilistic schema mappings and probabilistic mediated schemas, show how they can be constructed automatically for a set of data sources, and provide techniques for query answering. The foundations laid out in this chapter enable bootstrapping a *pay-as-you-go* integration system completely automatically.

Keywords: data integration, uncertainty, pay-as-you-go, mediated schema, schema mapping

1. Introduction

Data integration and exchange systems offer a uniform interface to a multitude of data sources and the ability to share data across multiple systems. These systems have recently enjoyed significant research and commercial success [18, 19]. Current data integration systems are essentially a natural extension of traditional database systems in that queries are specified in a structured form and data are modeled in one of the traditional data models (relational, XML). In addition, the data integration system has exact knowledge of how the data in the sources map to the schema used by the data integration system.

In this chapter we argue that as the scope of data integration applications broadens, such systems need to be able to model uncertainty at their core. Uncertainty can arise for multiple reasons in data integration. First, the semantic mappings between the data sources and the mediated schema may be approximate. For example, in an application like Google Base [17] that enables anyone to upload structured data, or when mapping millions of sources on the deep web [28], we cannot imagine specifying exact mappings. In some domains (e.g., bioinformatics), we do not necessarily know what the exact mapping is. Second, data are often extracted from unstructured sources using information extraction techniques. Since these techniques are approximate, the data obtained from the sources may be uncertain. Third, if the intended users of the application are not necessarily familiar with schemata, or if the domain of the system is too broad to offer form-based query interfaces (such as web forms), we need to support keyword queries. Hence, another source of uncertainty is the transformation between keyword queries and a set of candidate structured queries. Finally, if the scope of the domain is very broad, there can even be uncertainty about the concepts in the mediated schema.

Another reason for data integration systems to model uncertainty is to support *pay-as-you-go* integration. Dataspace Support Platforms [20] envision data integration systems where sources are added with no effort and the system is constantly evolving in a pay-as-you-go fashion to improve the quality of semantic mappings and query answering. This means that as the system evolves, there will be uncertainty about the semanantic mappings to its sources, its mediated schema and even the semantics of the queries posed to it.

This chapter describes some of the formal foundations for data integration with uncertainty. We define probabilistic schema mappings and probabilistic mediated schemas, and show how to answer queries in their presence. With these foundations, we show that it is possible to completely automatically bootstrap a pay-as-you-go integration system.

This chapter is largely based on previous papers [10, 6]. The proofs of the theorems we state and the experimental results validating some of our claims can be found in there. We also place several other works on uncertainty in data integration in the context of the system we envision. In the next section, we begin by describing an architecture for data integration system that incorporates uncertainty.

2. Overview of the System

This section describes the requirements from a data integration system that supports uncertainty and the overall architecture of the system.

2.1 Uncertainty in data integration

A data integration system needs to handle uncertainty at three levels.

Uncertain mediated schema: The mediated schema is the set of schema terms in which queries are posed. They do not necessarily cover all the attributes appearing in any of the sources, but rather the aspects of the domain that the application builder wishes to expose to the users. Uncertainty in schema mappings can arise for several reasons. First, as we describe in Section 4, if the mediated schema is automatically inferred from the data sources in a pay-as-you-go integration system, there will be some uncertainty about the results. Second, when domains get broad, there will be some uncertainty about how to model the domain. For example, if we model all the topics in Computer Science there will be some uncertainty about the degree of overlap between different topics.

Uncertain schema mappings: Data integration systems rely on schema mappings for specifying the semantic relationships between the data in the sources and the terms used in the mediated schema. However, schema mappings can be inaccurate. In many applications it is impossible to create and maintain precise mappings between data sources. This can be because the users are not skilled enough to provide precise mappings, such as in personal information management [11], because people do not understand the domain well and thus do not even know what correct mappings are, such as in bioinformatics, or because the scale of the data prevents generating and maintaining precise mappings, such as in integrating data of the web scale [27]. Hence, in practice, schema mappings are often generated by semi-automatic tools and not necessarily verified by domain experts.

Uncertain data: By nature, data integration systems need to handle uncertain data. One reason for uncertainty is that data are often extracted from unstructured or semi-structured sources by automatic methods (e.g., HTML pages, emails, blogs). A second reason is that data may come from sources that are unreliable or not up to date. For example, in enterprise settings, it is common for informational data such as gender, racial, and income level to be dirty or missing, even when the transactional data is precise.

Uncertain queries: In some data integration applications, especially on the web, queries will be posed as keywords rather than as structured queries against a well defined schema. The system needs to translate these queries into some structured form so they can be reformulated with respect to the data sources. At this step, the system may generate multiple candidate structured queries and have some uncertainty about which is the real intent of the user.

2.2 System architecture

Given the previously discussed requirements, we describe the architecture of a data integration system we envision that manages uncertainty at its core. We describe the system by contrasting it to a traditional data integration system.

The first and most fundamental characteristic of this system is that it is based on a probabilistic data model. This means that we attach probabilities to:

- tuples that we process in the system,
- schema mappings,

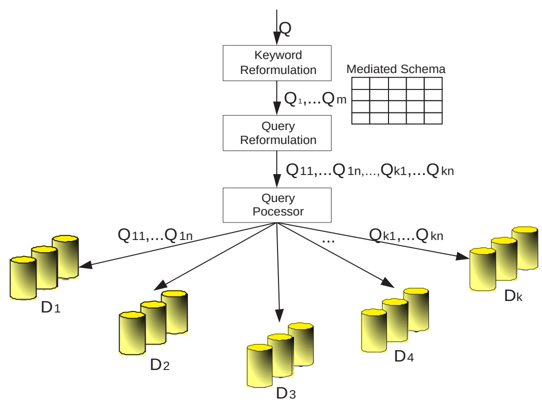

Figure 1.1. Architecture of a data-integration system that handles uncertainty.

- mediated schemas, and

- possible interpretations of keyword queries posed to the system.

In contrast, a traditional data integration system includes a single mediated schema and we assume we have as single (and correct) schema mapping between the mediated schema and each source. The data in the sources is also assumed to be correct.

Traditional data integration systems assume that the query is posed in a structured fashion (i.e., can be translated to some subset of SQL). Here, we assume that queries can be posed as keywords (to accommodate a much broader class of users and applications). Hence, whereas traditional data integration systems begin by reformulating a query onto the schemas of the data sources, a data integration system with uncertainty needs to first reformulate a keyword query into a set of candidate structured queries. We refer to this step as *keyword reformulation*. Note that keyword reformulation is different from techniques for keyword search on structured data (e.g., [22, 1]) in that (a) it does not assume access to all the data in the sources or that the sources support keyword search, and (b) it tries to distinguish different structural elements in the query in order to pose more precise queries to the sources (e.g., realizing that in the keyword query "Chicago weather", "weather" is an attribute label and "Chicago" is an instance name). That being said, keyword reformulation should benefit from techniques that support answering keyword search on structured data.

The query answering model is different. Instead of necessarily finding *all* answers to a given query, our goal is typically to find the top-k answers, and rank these answers most effectively.

The final difference from traditional data integration systems is that our query processing will need to be more adaptive than usual. Instead of generating a query answering plan and executing it, the steps we take in query processing will depend on results of previous steps. We note that adaptive query processing has been discussed quite a bit in data integration [12], where the need for adaptivity arises from the fact that data sources did not answer as quickly as expected or that we did not have accurate statistics about their contents to properly order our operations. In our work, however, the goal for adaptivity is to get the answers with high probabilities faster.

The architecture of the system is shown in Figure 1.1. The system contains a number of data sources and a mediated schema (we omit probabilistic mediated schemas from this figure). When the user poses a query Q, which can be either a structured query on the mediated schema or a keyword query, the system returns a set of answer tuples, each with a probability. If Q is a keyword query, the system first performs keyword reformulation to translate it into a set of candidate structured queries on the mediated schema. Otherwise, the candidate query is Q itself.

2.3 Source of probabilities

A critical issue in any system that manages uncertainty is whether we have a reliable source of probabilities. Whereas obtaining reliable probabilities for such a system is one of the most interesting areas for future research, there is quite a bit to build on. For keyword reformulation, it is possible to train and test reformulators on large numbers of queries such that each reformulation result is given a probability based on its performance statistics. For information extraction, current techniques are often based on statistical machine learning methods and can be extended to compute probabilities of each extraction result. Finally, in the case of schema matching, it is standard practice for schema matchers to also associate numbers with the candidates they propose (e.g., [3, 7–9, 21, 26, 34, 35]). The issue here is that the numbers are meant only as a ranking mechanism rather than true probabilities. However, as schema matching techniques start looking at a larger number of schemas, one can imagine ascribing probabilities (or estimations thereof) to their measures.

2.4 Outline of the chapter

We begin by discussing probabilistic schema mappings in Section 3. We also discuss how to answer queries in their presence and how to answer top-k queries. In Section 4 we discuss probabilistic mediated schemas. We begin by motivating them and showing that in some cases they add expressive power to the resulting system. Then we describe an algorithm for generating probabilistic mediated schemas from a collection of data sources.

	Possible Mapping	Prob
m_1 =	{(pname, name), (email-addr, email), (current-addr, mailing-addr), (permanent-addr, home-addr)}	0.5
m_2 =	{(pname, name), (email-addr, email), (permanent-addr, mailing-addr), (current-addr, home-addr)}	0.4
m_3 =	{(pname, name), (email-addr, mailing-addr), (current-addr, home-addr)}	0.1

(a)

*p*name	*e*mail-addr	*c*urrent-addr	*p*ermanent-addr
Alice	alice@	Mountain View	Sunnyvale
Bob	bob@	Sunnyvale	Sunnyvale

(b)

Tuple (mailing-addr)	Prob
('Sunnyvale')	0.9
('Mountain View')	0.5
('alice@')	0.1
('bob@')	0.1

(c)

Figure 1.2. The running example: (a) a probabilistic schema mapping between S and T; (b) a source instance D_S; (c) the answers of Q over D_S with respect to the probabilistic mapping.

3. Uncertainty in Mappings

The key to resolving heterogeneity at the schema level is to specify schema mappings between data sources. These mappings describe the relationship between the contents of the different sources and are used to reformulate a query posed over one source (or a mediated schema) into queries over the sources that are deemed relevant. However, in many applications we are not able to provide all the schema mappings upfront. In this section we introduce probabilistic schema mappings (p-mappings) to capture uncertainty on mappings between schemas.

We start by presenting a running example for this section that also motivates p-mappings (Section 3.1). Then we present a formal definition of probabilistic schema mapping and its semantics (Section 3.2). Then, Section 3.3 describes algorithms for query answering with respect to probabilistic mappings and discusses the complexity. Next, Section 3.4 shows how to leverage previous work on schema matching to automatically create probabilistic mappings. In the end, Section 3.5 briefly describes various extensions to the basic definition and Section 3.6 describes other types of approximate schema mappings that have been proposed in the literature.

3.1 Motivating probabilistic mappings

EXAMPLE 1.1 *Consider a data source S, which describes a person by her email address, current address, and permanent address, and the mediated* *schema T, which describes a person by her name, email, mailing address, home address and office address:*

```
S=(pname, email-addr, current-addr, permanent-addr)
T=(name, email, mailing-addr, home-addr, office-addr)
```

A semi-automatic schema-mapping tool may generate three possible mappings between S and T, assigning each a probability. Whereas the three mappings all map pname *to* name*, they map other attributes in the source and the target differently. Figure 1.2(a) describes the three mappings using sets of attribute correspondences. For example, mapping m_1 maps* pname *to* name*,* email-addr *to* email*,* current-addr *to* mailing-addr*, and* permanent-addr *to* home-addr*. Because of the uncertainty about which mapping is correct, we consider all of these mappings in query answering.*

Suppose the system receives a query Q composed on the mediated schema and asking for people's mailing addresses:

```
Q: SELECT mailing-addr FROM T
```

Using the possible mappings, we can reformulate Q into different queries:

```
Q1: SELECT current-addr FROM S
Q2: SELECT permanent-addr FROM S
Q3: SELECT email-addr FROM S
```

If the user requires all possible answers, the system generates a single aggregation query based on Q_1, Q_2 and Q_3 to compute the probability of each returned tuple, and sends the query to the data source. Suppose the data source contains a table D_S as shown in Figure 1.2(b), the system will retrieve four answer tuples, each with a probability, as shown in Figure 1.2(c).

If the user requires only the top-1 answer (i.e., the answer tuple with the highest probability), the system decides at runtime which reformulated queries to execute. For example, after executing Q_1 and Q_2 at the source, the system can already conclude that ('Sunnyvale') is the top-1 answer and can skip query Q_3. □

3.2 Definition and Semantics

3.2.1 Schema mappings. We begin by reviewing non-probabilistic schema mappings. The goal of a schema mapping is to specify the semantic relationships between a *source schema* and a *target schema*. We refer to the source schema as $\bar{S}$, and a relation in $\bar{S}$ as $S = \langle s_1, \ldots, s_m \rangle$. Similarly, we refer to the target schema as $\bar{T}$, and a relation in $\bar{T}$ as $T = \langle t_1, \ldots, t_n \rangle$.

We consider a limited form of schema mappings that are also referred to as *schema matching* in the literature. Specifically, a schema matching contains a set of *attribute correspondences*. An attribute correspondence is of the form

$c_{ij} = (s_i, t_j)$, where s_i is a *source attribute* in the schema S and t_j is a *target attribute* in the schema T. Intuitively, c_{ij} specifies that there is a relationship between s_i and t_j. In practice, a correspondence also involves a function that transforms the value of s_i to the value of t_j. For example, the correspondence (c-degree, temperature) can be specified as temperature=c-degree $*1.8 + 32$, describing a transformation from Celsius to Fahrenheit. These functions are irrelevant to our discussion, and therefore we omit them. This class of mappings are quite common in practice and already expose many of the novel issues involved in probabilistic mappings and In Section 3.5 we will briefly discuss extensions to a broader class of mappings.

Formally, relation mappings and schema mappings are defined as follows.

DEFINITION 1.2 (SCHEMA MAPPING) *Let $\bar{S}$ and $\bar{T}$ be relational schemas. A* relation mapping *M is a triple (S, T, m), where S is a relation in $\bar{S}$, T is a relation in $\bar{T}$, and m is a set of attribute correspondences between S and T.*

When each source and target attribute occurs in at most one correspondence in m, we call M a one-to-one relation mapping.

A schema mapping *$\overline{M}$ is a set of one-to-one relation mappings between relations in $\bar{S}$ and in $\bar{T}$, where every relation in either $\bar{S}$ or $\bar{T}$ appears at most once.* □

A pair of instances D_S and D_T *satisfies* a relation mapping m if for every source tuple $t_s \in D_S$, there exists a target tuple $t_t \in D_t$, such that for every attribute correspondence $(s, t) \in m$, the value of attribute s in t_s is the same as the value of attribute t in t_t.

EXAMPLE 1.3 *Consider the mappings in Example 1.1. The source database in Figure 1.2(b) (repeated in Figure 1.3(a)) and the target database in Figure 1.3(b) satisfy m_1.* □

3.2.2 Probabilistic schema mappings.

Intuitively, a probabilistic schema mapping describes a probability distribution of a set of *possible* schema mappings between a source schema and a target schema.

DEFINITION 1.4 (PROBABILISTIC MAPPING) *Let $\bar{S}$ and $\bar{T}$ be relational schemas. A* probabilistic mapping (p-mapping)*, pM, is a triple $(S, T, \mathbf{m})$, where $S \in \bar{S}$, $T \in \bar{T}$, and $\mathbf{m}$ is a set $\{(m_1, Pr(m_1)), \ldots, (m_l, Pr(m_l))\}$, such that*

- *for $i \in [1, l]$, m_i is a one-to-one mapping between S and T, and for every $i, j \in [1, l]$, $i \neq j \Rightarrow m_i \neq m_j$.*
- *$Pr(m_i) \in [0, 1]$ and $\sum_{i=1}^{l} Pr(m_i) = 1$.*

*p*name	*e*mail-addr	*c*urrent-addr	*p*ermanent-addr
Alice	alice@	Mountain View	Sunnyvale
Bob	bob@	Sunnyvale	Sunnyvale

(a)

*n*ame	*e*mail	*m*ailing-addr	*h*ome-addr	*o*ffice-addr
Alice	alice@	Mountain View	Sunnyvale	office
Bob	bob@	Sunnyvale	Sunnyvale	office

(b)

*n*ame	*e*mail	*m*ailing-addr	*h*ome-addr	*o*ffice-addr
Alice	alice@	Sunnyvale	Mountain View	office
Bob	email	bob@	Sunnyvale	office

(c)

Tuple (mailing-addr)	Prob
('Sunnyvale')	0.9
('Mountain View')	0.5
('alice@')	0.1
('bob@')	0.1

(d)

Tuple (mailing-addr)	Prob
('Sunnyvale')	0.94
('Mountain View')	0.5
('alice@')	0.1
('bob@')	0.1

(e)

Figure 1.3. Example 1.11: (a) a source instance D_S; (b) a target instance that is by-table consistent with D_S and m_1; (c) a target instance that is by-tuple consistent with D_S and $< m_2, m_3 >$; (d) $Q^{table}(D_S)$; (e) $Q^{tuple}(D_S)$.

A schema p-mapping, $\overline{pM}$, *is a set of p-mappings between relations in* $\bar{S}$ *and in* $\bar{T}$*, where every relation in either* $\bar{S}$ *or* $\bar{T}$ *appears in at most one p-mapping.* □

We refer to a non-probabilistic mapping as an *ordinary mapping*. A schema p-mapping may contain both p-mappings and ordinary mappings. Example 1.1 shows a p-mapping (see Figure 1.2(a)) that contains three possible mappings.

3.2.3 Semantics of probabilistic mappings. Intuitively, a probabilistic schema mapping models the uncertainty about which of the mappings in pM is the correct one. When a schema matching system produces a set of candidate matches, there are two ways to interpret the uncertainty: (1) a single mapping in pM is the correct one and it applies to all the data in S, or (2) several mappings are partially correct and each is suitable for a subset of tuples in S, though it is not known which mapping is the right one for a specific tuple. Figure 1.3(b) illustrates the first interpretation and applies mapping m_1. For the same example, the second interpretation is equally valid: some people may choose to use their current address as mailing address while others use their permanent address as mailing address; thus, for different tuples we may apply different mappings, so the correct mapping depends on the particular tuple.

We define query answering under both interpretations. The first interpretation is referred to as the *by-table* semantics and the second one is referred to as the *by-tuple* semantics of probabilistic mappings. Note that one cannot argue for one interpretation over the other; the needs of the application should dictate the appropriate semantics. Furthermore, the complexity results for query answering, which will show advantages to by-table semantics, should not be taken as an argument in the favor of by-table semantics.

We next define query answering with respect to p-mappings in detail and the definitions for schema p-mappings are the obvious extensions. Recall that given a query and an ordinary mapping, we can compute *certain answers* to the query with respect to the mapping. Query answering with respect to p-mappings is defined as a natural extension of certain answers, which we next review.

A mapping defines a relationship between instances of S and instances of T that are *consistent* with the mapping.

DEFINITION 1.5 (CONSISTENT TARGET INSTANCE) *Let* $M = (S, T, m)$ *be a relation mapping and* D_S *be an instance of* S.

An instance D_T *of* T *is said to be* consistent with D_S and M, *if for each tuple* $t_s \in D_S$, *there exists a tuple* $t_t \in D_T$, *such that for every attribute correspondence* $(a_s, a_t) \in m$, *the value of* a_s *in* t_s *is the same as the value of* a_t *in* t_t. □

For a relation mapping M and a source instance D_S, there can be an infinite number of target instances that are consistent with D_S and M. We denote by $Tar_M(D_S)$ the set of all such target instances. The set of answers to a query Q is the intersection of the answers on all instances in $Tar_M(D_S)$.

DEFINITION 1.6 (CERTAIN ANSWER) *Let* $M = (S, T, m)$ *be a relation mapping. Let* Q *be a query over* T *and let* D_S *be an instance of* S.

A tuple t *is said to be a* certain answer of Q with respect to D_S and M, *if for every instance* $D_T \in Tar_M(D_S)$, $t \in Q(D_T)$. □

By-table semantics: We now generalize these notions to the probabilistic setting, beginning with the by-table semantics. Intuitively, a p-mapping pM describes a set of possible worlds, each with a possible mapping $m \in pM$. In by-table semantics, a source table can fall in one of the possible worlds; that is, the possible mapping associated with that possible world applies to the whole source table. Following this intuition, we define target instances that are *consistent with* the source instance.

DEFINITION 1.7 (BY-TABLE CONSISTENT INSTANCE) *Let* $pM = (S, T, \mathbf{m})$ *be a p-mapping and* D_S *be an instance of* S.

An instance D_T of T is said to be by-table consistent with D_S and pM*, if there exists a mapping $m \in \mathbf{m}$ such that D_S and D_T satisfy m.* □

Given a source instance D_S and a possible mapping $m \in \mathbf{m}$, there can be an infinite number of target instances that are consistent with D_S and m. We denote by $Tar_m(D_S)$ the set of all such instances.

In the probabilistic context, we assign a probability to every answer. Intuitively, we consider the certain answers with respect to each possible mapping in isolation. The probability of an answer t is the sum of the probabilities of the mappings for which t is deemed to be a certain answer. We define by-table answers as follows:

DEFINITION 1.8 (BY-TABLE ANSWER) *Let $pM = (S, T, \mathbf{m})$ be a p-mapping. Let Q be a query over T and let D_S be an instance of S.*

Let t be a tuple. Let $\bar{m}(t)$ be the subset of $\mathbf{m}$, such that for each $m \in \bar{m}(t)$ and for each $D_T \in Tar_m(D_S)$, $t \in Q(D_T)$.

Let $p = \sum_{m \in \bar{m}(t)} Pr(m)$. If $p > 0$, then we say (t, p) is a by-table answer of Q with respect to D_S and pM. □

By-tuple semantics: If we follow the possible-world notions, in by-tuple semantics, different tuples in a source table can fall in different possible worlds; that is, different possible mappings associated with those possible worlds can apply to the different source tuples.

Formally, the key difference in the definition of by-tuple semantics from that of by-table semantics is that a consistent target instance is defined by a mapping *sequence* that assigns a (possibly different) mapping in $\mathbf{m}$ to each source tuple in D_S. (Without losing generality, in order to compare between such sequences, we assign some order to the tuples in the instance).

DEFINITION 1.9 (BY-TUPLE CONSISTENT INSTANCE) *Let $pM = (S, T, \mathbf{m})$ be a p-mapping and let D_S be an instance of S with d tuples.*

An instance D_T of T is said to be by-tuple consistent with D_S and pM*, if there is a sequence $\langle m^1, \ldots, m^d \rangle$ such that d is the number of tuples in D_S and for every $1 \leq i \leq d$,*

- *$m^i \in \mathbf{m}$, and*
- *for the i^{th} tuple of D_S, t_i, there exists a target tuple $t'_i \in D_T$ such that for each attribute correspondence $(a_s, a_t) \in m^i$, the value of a_s in t_i is the same as the value of a_t in t'_i.* □

Given a mapping sequence $seq = \langle m^1, \ldots, m^d \rangle$, we denote by $Tar_{seq}(D_S)$ the set of all target instances that are consistent with D_S and seq. Note that if D_T is by-table consistent with D_S and m, then D_T is also by-tuple consistent with D_S and a mapping sequence in which each mapping is m.

We can think of every sequence of mappings $seq = \langle m^1, \ldots, m^d \rangle$ as a separate event whose probability is $Pr(seq) = \Pi_{i=1}^{d} Pr(m^i)$. (Section 3.5 relaxes this independence assumption and introduces *conditional mappings*.) If there are l mappings in pM, then there are l^d sequences of length d, and their probabilities add up to 1. We denote by $\mathbf{seq}_d(pM)$ the set of mapping sequences of length d generated from pM.

DEFINITION 1.10 (BY-TUPLE ANSWER) *Let $pM = (S, T, \mathbf{m})$ be a p-mapping. Let Q be a query over T and D_S be an instance of S with d tuples.*

Let t be a tuple. Let $\overline{seq}(t)$ be the subset of $\mathbf{seq}_d(pM)$, such that for each $seq \in \overline{seq}(t)$ and for each $D_T \in Tar_{seq}(D_S)$, $t \in Q(D_T)$.

Let $p = \sum_{seq \in \overline{seq}(t)} Pr(seq)$. If $p > 0$, we call (t, p) a by-tuple answer of Q with respect to D_S and pM. □

The set of by-table answers for Q with respect to D_S is denoted by $Q^{table}(D_S)$ and the set of by-tuple answers for Q with respect to D_S is denoted by $Q^{tuple}(D_S)$.

EXAMPLE 1.11 *Consider the p-mapping pM, the source instance D_S, and the query Q in the motivating example.*

In by-table semantics, Figure 1.3(b) shows a target instance that is consistent with D_S (repeated in Figure 1.3(a)) and possible mapping m_1. Figure 1.3(d) shows the by-table answers of Q with respect to D_S and pM. As an example, for tuple t =('Sunnyvale'), we have $\bar{m}(t) = \{m_1, m_2\}$, so the possible tuple ('Sunnyvale', 0.9) is an answer.

In by-tuple semantics, Figure 1.3(c) shows a target instance that is by-tuple consistent with D_S and the mapping sequence $< m_2, m_3 >$. Figure 1.3(e) shows the by-tuple answers of Q with respect to D_S and pM. Note that the probability of tuple t=('Sunnyvale') in the by-table answers is different from that in the by-tuple answers. We describe how to compute the probabilities in detail in the next section. □

3.3 Query Answering

This section studies query answering in the presence of probabilistic mappings. We start with describing algorithms for returning all answer tuples with probabilities, and discussing the complexity of query answering in terms of the size of the data (*data complexity*) and the size of the p-mapping (*mapping complexity*). We then consider returning the top-k query answers, which are the k answer tuples with the top probabilities.

3.3.1 By-table query answering. In the case of by-table semantics, answering queries is conceptually simple. Given a p-mapping $pM = (S, T, \mathbf{m})$ and an SPJ query Q, we can compute the certain answers of Q under each of the mappings $m \in \mathbf{m}$. We attach the probability $Pr(m)$ to every

Tuple (mailing-addr)	Pr
('Sunnyvale')	0.94
('Mountain View')	0.5
('alice@')	0.1
('bob@')	0.1

(a)

Tuple (mailing-addr)	Pr
('Sunnyvale')	0.8
('Mountain View')	0.8

(b)

Figure 1.4. Example 1.13: (a) $Q_1^{tuple}(D)$ and (b) $Q_2^{tuple}(D)$.

certain answer under m. If a tuple is an answer to Q under multiple mappings in $\mathbf{m}$, then we add up the probabilities of the different mappings.

Algorithm ByTable takes as input an SPJ query Q that mentions the relations $T_1, \ldots, T_l$ in the `FROM` clause. Assume that we have the p-mapping pM_i associated with the table T_i. The algorithm proceeds as follows.

Step 1: We generate the possible reformulations of Q (a reformulation query computes all certain answers when executed on the source data) by considering every combination of the form $(m^1, \ldots, m^l)$, where m^i is one of the possible mappings in pM_i. Denote the set of reformulations by $Q'_1, \ldots, Q'_k$. The probability of a reformulation $Q' = (m^1, \ldots, m^l)$ is $\Pi_{i=1}^{l} Pr(m^i)$.

Step 2: For each reformulation Q', retrieve each of the unique answers from the sources. For each answer obtained by $Q'_1 \cup \ldots \cup Q'_k$, its probability is computed by summing the probabilities of the Q''s in which it is returned.

Importantly, note that it is possible to express both steps as an SQL query with grouping and aggregation. Therefore, if the underlying sources support SQL, we can leverage their optimizations to compute the answers.

With our restricted form of schema mapping, the algorithm takes time polynomial in the size of the data and the mappings. We thus have the following complexity result.

Theorem 1.12 *Let $\overline{pM}$ be a schema p-mapping and let Q be an SPJ query.*

Answering Q with respect to $\overline{pM}$ in by-table semantics is in PTIME in the size of the data and the mapping. □

3.3.2 By-tuple query answering. To extend the by-table query-answering strategy to by-tuple semantics, we would need to compute the certain answers for every *mapping sequence* generated by pM. However, the number of such mapping sequences is exponential in the size of the input data. The following example shows that for certain queries this exponential time complexity is inevitable.

Example 1.13 *Suppose that in addition to the tables in Example 1.1, we also have* U(city) *in the source and* V(hightech) *in the target. The p-mapping for V contains two possible mappings: (*{(city, hightech)}*, .8) and (∅, .2).*

Consider the following query Q, which decides if there are any people living in a high-tech city.

```
Q: SELECT 'true'
   FROM T, V
   WHERE T.mailing-addr = V.hightech
```

An incorrect way of answering the query is to first execute the following two sub-queries Q_1 and Q_2, then join the answers of Q_1 and Q_2 and summing up the probabilities.

```
Q1: SELECT mailing-addr FROM T
Q2: SELECT hightech FROM V
```

Now consider the source instance D, where D_S is shown in Figure 1.2(a), and D_U has two tuples ('Mountain View') and ('Sunnyvale'). Figure 1.4(a) and (b) show $Q_1^{tuple}(D)$ and $Q_2^{tuple}(D)$. If we join the results of Q_1 and Q_2, we obtain for the true *tuple the following probability: $0.94 * 0.8 + 0.5 * 0.8 = 1.152$. However, this is incorrect. By enumerating all consistent target tables, we in fact compute 0.864 as the probability. The reason for this error is that on some target instance that is by-tuple consistent with the source instance, the answers to both Q_1 and Q_2 contain tuple ('Sunnyvale') and tuple ('Mountain View'). Thus, generating the tuple ('Sunnyvale') as an answer for both Q_1 and Q_2 and generating the tuple ('Mountain View') for both queries are not independent events, and so simply adding up their probabilities leads to incorrect results.*

Indeed, it is not clear if there exists a better algorithm to answer Q than by enumerating all by-tuple consistent target instances and then answering Q on each of them. □

In fact, it is proved that in general, answering SPJ queries in by-tuple semantics with respect to schema p-mappings is hard.

THEOREM 1.14 *Let Q be an SPJ query and let $\overline{pM}$ be a schema p-mapping. The problem of finding the probability for a by-tuple answer to Q with respect to $\overline{pM}$ is #P-complete with respect to data complexity and is in PTIME with respect to mapping complexity.* □

Recall that #P is the complexity class of some hard counting problems (*e.g.* , counting the number of variable assignments that satisfy a Boolean formula). It is believed that a #P-complete problem cannot be solved in polynomial time, unless $P = NP$.

Although by-tuple query answering in general is hard, there are two restricted but common classes of queries for which by-tuple query answering takes polynomial time. The first class of queries are those that include only a single subgoal being the target of a p-mapping; here, we refer to an occurrence of a table in the `FROM` clause of a query as a *subgoal* of the query. Relations in the other subgoals are either involved in ordinary mappings or do not require a mapping. Hence, if we only have uncertainty with respect to one part of the domain, our queries will typically fall in this class. The second class of queries can include multiple subgoals involved in p-mappings, but return the join attributes for such subgoals. We next illustrate these two classes of queries and query answering for them using two examples.

EXAMPLE 1.15 *Consider rewriting Q in the motivating example, repeated as follows:*

```
Q: SELECT mailing-addr FROM T
```

To answer the query, we first rewrite Q into query Q' by adding the id *column:*

```
Q': SELECT id, mailing-addr FROM T
```

We then invoke BYTABLE *and generate the following SQL query to compute by-table answers for Q':*

```
Qa: SELECT id, mailing-addr, SUM(pr)
    FROM (
      SELECT DISTINCT id, current-addr
             AS mailing-addr, 0.5 AS pr
      FROM S
      UNION ALL
      SELECT DISTINCT id, permanent-addr
             AS mailing-addr, 0.4 AS pr
      FROM S
      UNION ALL
      SELECT DISTINCT id, email-addr
             AS mailing-addr, 0.1 AS pr
      FROM S)
    GROUP BY id, mailing-addr
```

Finally, we generate the results using the following query.

```
Qu:  SELECT mailing-addr, NOR(pr) AS pr
     FROM Qa
     GROUP BY mailing-addr
```

where for a set of probabilities $pr_1, \ldots, pr_n$, NOR computes $1-\Pi_{i=1}^{n}(1-pr_i)$.
□

EXAMPLE 1.16 *Consider the schema p-mapping in Example 1.13. If we revise Q slightly be returning the join attribute, shown as follows, we can answer the query in polynomial time.*

```
Q': SELECT V.hightech
    FROM T, V
    WHERE T.mailing-addr = V.hightech
```

We answer the query by dividing it into two sub-queries, Q_1 and Q_2, as shown in Example 1.13. We can compute Q_1 with query Q_u (shown in Example 1.15) and compute Q_2 similarly with a query Q'_u. We compute by-tuple answers of Q' as follows:

```
SELECT Qu'.hightech, Qu.pr*Qu'.pr
FROM Qu, Qu'
WHERE Qu.mailing-addr = Qu'.hightect
```

□

3.3.3 Top-K Query Answering. The main challenge in designing the algorithm for returning top-k query answers is to only perform the necessary reformulations at every step and halt when the top-k answers are found. We focus on top-k query answering for by-table semantics and the algorithm can be modified for by-tuple semantics.

Recall that in by-table query answering, the probability of an answer is the sum of the probabilities of the reformulated queries that generate the answer. Our goal is to reduce the number of reformulated queries we execute. The algorithm we describe next proceeds in a greedy fashion: it executes queries in descending order of probabilities. For each tuple t, it maintains the upper bound $p_{max}(t)$ and lower bound $p_{min}(t)$ of its probability. This process halts when it finds k tuples whose p_{min} values are higher than p_{max} of the rest of the tuples.

TOPKBYTABLE takes as input an SPJ query Q, a schema p-mapping $\overline{pM}$, an instance D_S of the source schema, and an integer k, and outputs the top-k answers in $Q^{table}(D_S)$. The algorithm proceeds in three steps.

Step 1: Rewrite Q according to $\overline{pM}$ into a set of queries $Q_1, \ldots, Q_n$, each with a probability assigned in a similar way as stated in Algorithm BYTABLE.

Step 2: Execute $Q_1, \ldots, Q_n$ in descending order of their probabilities. Maintain the following measures:

- The highest probability, $PMax$, for the tuples that have not been generated yet. We initialize $PMax$ to 1; after executing query Q_i and updating the list of answers (see third bullet), we decrease $PMax$ by $Pr(Q_i)$;

- The threshold th determining which answers are potentially in the top-k. We initialize th to 0; after executing Q_i and updating the answer list, we set th to the k-th largest p_{min} for tuples in the answer list;

- A list L of answers whose p_{max} is no less than th, and bounds p_{min} and p_{max} for each answer in L. After executing query Q_i, we update the list as follows: (1) for each $t \in L$ and $t \in Q_i(D_S)$, we increase $p_{min}(t)$ by $Pr(Q_i)$; (2) for each $t \in L$ but $t \notin Q_i(D_S)$, we decrease $p_{max}(t)$ by $Pr(Q_i)$; (3) if $PMax \geq th$, for each $t \notin L$ but $t \in Q_i(D_S)$, insert t to L, set p_{min} to $Pr(Q_i)$ and $p_{max}(t)$ to $PMax$.

- A list T of k tuples with top p_{min} values.

Step 3: When $th > PMax$ and for each $t \notin T$, $th > p_{max}(t)$, halt and return T.

EXAMPLE 1.17 *Consider Example 1.1 where we seek for top-1 answer. We answer the reformulated queries in order of Q_1, Q_2, Q_3. After answering Q_1, for tuple ("Sunnyvale") we have $p_{min} = .5$ and $p_{max} = 1$, and for tuple ("Mountain View") we have the same bounds. In addition, $PMax = .5$ and $th = .5$.*

In the second round, we answer Q_2. Then, for tuple ("Sunnyvale") we have $p_{min} = .9$ and $p_{max} = 1$, and for tuple ("Mountain View") we have $p_{min} = .5$ and $p_{max} = .6$. Now $PMax = .1$ and $th = .9$.

Because $th > PMax$ and th is above the p_{max} for the ("Mountain View") tuple, we can halt and return ("Sunnyvale") as the top-1 answer. □

3.4 Creating P-mappings

We now address the problem of generating a p-mapping between a source schema and a target schema. We begin by assuming we have a set of weighted correspondences between the source attributes and the target attributes. These weighted correspondences are created by a set of schema matching modules. However, as we explain shortly, there can be *multiple* p-mappings that are consistent with a given set of weighted correspondences, and the question is which of them to choose. We describe an approach to creating p-mappings that is based on choosing the mapping that maximizes the *entropy* of the probability assignment.

3.4.1 Computing weighted correspondences. A *weighted correspondence* between a pair of attributes specifies the degree of semantic similarity between them. Let $S(s_1, \ldots, s_m)$ be a source schema and $T(t_1, \ldots, t_n)$ be a target schema. We denote by $C_{i,j}, i \in [1, m], j \in [1, n]$, the weighted correspondence between s_i and t_j and by $w_{i,j}$ the weight of $C_{i,j}$. The first step is to compute a weighted correspondence between every pair of attributes, which can be done by applying existing schema matching techniques.

Although weighted correspondences tell us the degree of similarity between pairs of attributes, they do not tell us *which* target attribute a source attribute should map to. For example, a target attribute mailing-address can be both similar to the source attribute current-addr and to permanent-addr, so it makes sense to map either of them to mailing-address in a schema mapping. In fact, given a set of weighted correspondences, there could be a *set* of p-mappings that are consistent with it. We can define the one-to-many relationship between sets of weighted correspondences and p-mappings by specifying when a p-mapping is *consistent with* a set of weighted correspondences.

DEFINITION 1.18 (CONSISTENT P-MAPPING) *A p-mapping pM is* consistent with *a weighted correspondence $C_{i,j}$ between a pair of source and target attributes if the sum of the probabilities of all mappings $m \in pM$ containing correspondence (i, j) equals $w_{i,j}$; that is,*

$$w_{i,j} = \sum_{m \in pM, (i,j) \in m} \Pr(m).$$

A p-mapping is consistent with *a set of weighted correspondences* **C** *if it is consistent with each weighted correspondence $C \in \mathbf{C}$.* □

However, not every set of weighted correspondences admits a consistent p-mapping. The following theorem shows under which conditions a consistent p-mapping exists, and establishes a normalization factor for weighted correspondences that will guarantee the existence of a consistent p-mapping.

THEOREM 1.19 *Let* **C** *be a set of weighted correspondences between a source schema $S(s_1, \ldots, s_m)$ and a target schema $T(t_1, \ldots, t_n)$.*

- *There exists a consistent p-mapping with respect to* **C** *if and only if (1) for every $i \in [1, m]$, $\sum_{j=1}^{n} w_{i,j} \le 1$ and (2) for every $j \in [1, n]$, $\sum_{i=1}^{m} w_{i,j} \le 1$.*
- *Let*

$$M' = max\{max_i\{\sum_{j=1}^{n} w_{i,j}\}, max_j\{\sum_{i=1}^{m} w_{i,j}\}\}.$$

Then, for each $i \in [1, m]$, $\sum_{j=1}^{n} \frac{w_{i,j}}{M'} \le 1$ and for each $j \in [1, n]$, $\sum_{i=1}^{m} \frac{w_{i,j}}{M'} \le 1$. □

Based on Theorem 1.19, we normalize the weighted correspondences we generated as described previously by dividing them by M'; that is,

$$w'_{i,j} = \frac{w_{i,j}}{M'}.$$

3.4.2 Generating p-mappings. To motivate our approach to generating p-mappings, consider the following example. Consider a source schema (A, B) and a target schema (A', B'). Assume we have computed the following weighted correspondences between source and target attributes: $w_{A,A'} = 0.6$ and $w_{B,B'} = 0.5$ (the rest are 0).

As we explained above, there are an infinite number of p-mappings that are consistent with this set of weighted correspondences and below we list two:
pM_1:

```
m1: (A,A'), (B,B'): 0.3 m2: (A,A'): 0.3 m3:
   (B,B'): 0.2 m4: empty: 0.2
```

pM_2:

```
m1: (A,A'), (B,B'): 0.5
m2: (A,A'): 0.1
m3: empty: 0.4
```

In a sense, pM_1 seems better than pM_2 because it assumes that the similarity between A and A' is independent of the similarity between B and B'.

In the general case, among the many p-mappings that are consistent with a set of weighted correspondences **C**, we choose the one with the *maximum entropy*; that is, the p-mappings whose probability distribution obtains the maximum value of $\sum_{i=1}^{l} -p_i * logp_i$. In the above example, pM_1 obtains the maximum entropy.

The intuition behind maximum entropy is that when we need to select among multiple possible distributions on a set of exclusive events, we choose the one that does not favor any of the events over the others. Hence, we choose the distribution that does not *introduce new information* that we didn't have apriori. The principle of maximum entropy is widely used in other areas such as natural language processing.

To create the p-mapping, we proceed in two steps. First, we enumerate all possible one-to-one schema mappings between S and M that contain a subset of correspondences in **C**. Second, we assign probabilities on each of the mappings in a way that maximizes the entropy of our result p-mapping.

Enumerating all possible schema mappings given **C** is trivial: for each subset of correspondences, if it corresponds to a one-to-one mapping, we consider the mapping as a possible mapping.

Given the possible mappings $m_1, \ldots, m_l$, we assign probabilities $p_1, \ldots, p_l$ to $m_1, \ldots, m_l$ by solving the following constraint optimization problem (OPT):
`maximize` $\sum_{k=1}^{l} -p_k * \log p_k$ `subject to:`

1 $\forall k \in [1, l], 0 \leq p_k \leq 1,$

2 $\sum_{k=1}^{l} p_k = 1$, and

3 $\forall i, j : \sum_{k\in[1,l],(i,j)\in m_k} p_k = w_{i,j}$.

We can apply existing technology in solving the OPT optimization problem. Although finding maximum-entropy solutions in general is costly, the experiments described in [6] show that the execution time is reasonable for a one-time process.

3.5 Broader Classes of Mappings

In this section we describe several practical extensions to the basic mapping language. The query answering techniques and complexity results we have described carry over to these techniques.

GLAV mappings: The common formalism for schema mappings, GLAV (a.k.a. tuple-generating dependencies), is based on expressions of the form

$$m : \forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y})).$$

In the expression, φ is the body of a conjunctive query over $\bar{S}$ and ψ is the body of a conjunctive query over $\bar{T}$. A pair of instances D_S and D_T *satisfies* a GLAV mapping m if for every assignment of $\mathbf{x}$ in D_S that satisfies φ there exists an assignment of $\mathbf{y}$ in D_T that satisfies ψ.

We define *general p-mappings* to be triples of the form $pGM = (\bar{S}, \bar{T}, \mathbf{gm})$, where $\mathbf{gm}$ is a set $\{(gm_i, Pr(gm_i)) \mid i \in [1, n]\}$, such that for each $i \in [1, n]$, gm_i is a general GLAV mapping. The definition of by-table semantics for such mappings is a simple generalization of Definition 1.8 and query answering can be conducted in PTIME. Extending by-tuple semantics to arbitrary GLAV mappings is much trickier than by-table semantics and would involve considering mapping sequences whose length is the product of the number of tuples in each source table, and the results are much less intuitive.

THEOREM 1.20 *Let pGM be a general p-mapping between a source schema $\bar{S}$ and a target schema $\bar{T}$. Let D_S be an instance of $\bar{S}$. Let Q be an SPJ query with only equality conditions over $\bar{T}$. The problem of computing $Q^{table}(D_S)$ with respect to pGM is in PTIME in the size of the data and the mapping.* □

Complex mappings: Complex mappings map a set of attributes in the source to a set of attributes in the target. For example, we can map the attribute address to the concatenation of street, city, and state.

Formally, a *set correspondence* between S and T is a relationship between a subset of attributes in S and a subset of attributes in T. Here, the function associated with the relationship specifies a single value for each of the target attributes given a value for each of the source attributes. Again, the actual functions are irrelevant to our discussion. A *complex mapping* is a triple (S, T, cm), where cm is a set of set correspondences, such that each attribute in S or T is involved in at most one set correspondence. A *complex p-mapping* is of the form $pCM = \{(cm_i, Pr(cm_i)) \mid i \in [1,n]\}$, where $\sum_{i=1}^{n} Pr(cm_i) = 1$.

THEOREM 1.21 *Let $\overline{pCM}$ be a complex schema p-mapping between schemas $\bar{S}$ and $\bar{T}$. Let D_S be an instance of $\bar{S}$. Let Q be an SPJ query over $\bar{T}$. The data complexity and mapping complexity of computing $Q^{table}(D_S)$ with respect to $\overline{pCM}$ are PTIME. The data complexity of computing $Q^{tuple}(D_S)$ with respect to $\overline{pCM}$ is #P-complete. The mapping complexity of computing $Q^{tuple}(D_S)$ with respect to $\overline{pCM}$ is in PTIME.* □

Union mapping: *Union mappings* specify relationships such as both attribute home-address and attribute office-address can be mapped to address. Formally, a *union mapping* is a triple $(S, T, \bar{m})$, where $\bar{m}$ is a set of mappings between S and T. Given a source relation D_S and a target relation D_T, we say D_S and D_T are consistent with respect to the union mapping if for each source tuple t and $m \in \bar{m}$, there exists a target tuple t', such that t and t' satisfy m. A *union p-mapping* is of the form $pUM = \{(\bar{m}_i, Pr(\bar{m}_i)) \mid i \in [1,n]\}$, where $\sum_{i=1}^{n} Pr(\bar{m}_i) = 1$.

Both by-table and by-tuple semantics apply to probabilistic union mappings.

THEOREM 1.22 *Let $\overline{pUM}$ be a union schema p-mapping between a source schema $\bar{S}$ and a target schema $\bar{T}$. Let D_S be an instance of $\bar{S}$. Let Q be a conjunctive query over $\bar{T}$. The problem of computing $Q^{table}(D_S)$ with respect to $\overline{pUM}$ is in PTIME in the size of the data and the mapping; the problem of computing $Q^{tuple}(D_S)$ with respect to $\overline{pUM}$ is in PTIME in the size of the mapping and #P-complete in the size of the data.* □

Conditional mappings: In practice, our uncertainty is often conditioned. For example, we may want to state that daytime-phone maps to work-phone with probability 60% if age ≤ 65, and maps to home-phone with probability 90% if age > 65.

We define a *conditional p-mapping* as a set $cpM = \{(pM_1, C_1), \ldots, (pM_n, C_n)\}$, where $pM_1, \ldots, pM_n$ are p-mappings, and $C_1, \ldots, C_n$ are pairwise disjoint conditions. Intuitively, for each $i \in [1,n]$, pM_i describes the probability distribution of possible mappings when condition C_i holds. Conditional mappings make more sense for by-tuple semantics. The following theorem shows that the complexity results carry over to such mappings.

THEOREM 1.23 *Let $\overline{cpM}$ be a conditional schema p-mapping between $\bar{S}$ and $\bar{T}$. Let D_S be an instance of $\bar{S}$. Let Q be an SPJ query over $\bar{T}$. The problem of computing $Q^{tuple}(D_S)$ with respect to $\overline{cpM}$ is in PTIME in the size of the mapping and #P-complete in the size of the data.* □

3.6 Other Types of Approximate Schema Mappings

There have been various models proposed to capture uncertainty on mappings between attributes. [15] proposes keeping the top-K mappings between two schemas, each with a probability (between 0 and 1) of being true. [16] proposes assigning a probability for matching of every pair of source and target attributes. This notion corresponds to weighted correspondences described in Section 3.4.

Magnani and Montesi [29] have empirically shown that top-k schema mappings can be used to increase the recall of a data integration process and Gal [14] described how to generate top-k schema matchings by combining the matching results generated by various matchers. The probabilistic schema mappings we described above are different as they contain all possible schema mappings that conform to the schema matching results and assigns probabilities to these mappings to reflect the likelihood that each mapping is correct. Nottelmann and Straccia [32] proposed generating probabilistic schema matchings that capture the uncertainty on each matching step. The probabilistic schema mappings we create not only capture our uncertainty on results of the matching step, but also take into consideration various combinations of attribute correspondences and describe a *distribution* of possible schema mappings where the probabilities of all mappings sum up to 1.

There have also been work studying how to use probabilistic models to capture uncertainty on mappings of schema object classes, such as DatabasePapers and AIPapers. Query answering can take such uncertainty into consideration in computing the coverage percentage of the returned answers and in ordering information sources to maximize the likelihood of obtaining answers early. In the relational model, an object class is often represented using a relational table; thus, these probabilistic models focus on mapping between tables rather than attributes in the tables.

Specifically, consider two object classes A and B. The goal of the probabilistic models is to capture the uncertainty on whether A maps to B. One method [13] uses probability $P(B|A)$, which is the probability that an instance of A is also an instance of B. Another method [29] uses a tuple $< A, B, R, P >$, where R is a set of mutually exclusive relationships between A and B, and P is a probability distribution over R. The possible relationships considered in this model include *equivalent* $=$, *subset-subsumption* $\subset$, *superset-subsumption* $\supset$, *overlapping* $\cap$, *disjointness* $\not\cap$, and *incompatibility* $\not\sim$.

4. Uncertainty in Mediated Schema

The mediated schema is the set of schema terms (e.g., relations, attribute names) in which queries are posed. They do not necessarily cover all the attributes appearing in any of the sources, but rather the aspects of the domain that are important for the integration application. When domains are broad, and there are multiple perspectives on them (e.g., a domain in science that is constantly under evolution), then there will be uncertainty about which is the correct mediated schema and about the meaning of its terms. When the mediated schema is created automatically by inspecting the sources in a pay-as-you-go system, there will also be uncertainty about the mediated schema.

In this section we first motivate the need for probabilistic mediated schemas (p-med-schemas) with an example (Section 4.1). In Section 4.2 we formally define p-med-schemas and relate them with p-mappings in terms of expressive power and semantics of query answering. Then in Section 4.3 we describe an algorithm for creating a p-med-schema from a set of data sources. Finally, Section 4.4 gives an algorithm for consolidating a p-med-schema into a single schema that is visible to the user in a pay-as-you-go system.

4.1 P-Med-Schema Motivating Example

Let us begin with an example motivating p-med-schemas. Consider a setting in which we are trying to automatically infer a mediated schema from a set of data sources, where each of the sources is a single relational table. In this context, the mediated schema can be thought of as a "clustering" source attributes, with similar attributes being grouped into the same cluster. The quality of query answers critically depends on the quality of this clustering. Because of the heterogeneity of the data sources being integrated, one is typically unsure of the semantics of the source attributes and in turn of the clustering.

EXAMPLE 1.24 *Consider two source schemas both describing people:*

```
S1(name, hPhone, hAddr, oPhone, oAddr)
S2(name, phone, address)
```

In S2, the attribute phone *can either be a home phone number or be an office phone number. Similarly,* address *can either be a home address or be an office address.*

Suppose we cluster the attributes of S1 and S2. There are multiple ways to cluster the attributes and they correspond to different mediated schemas. Below we list a few (in the mediated schemas we abbreviate hPhone *as* hP*,* oPhone *as* oP*,* hAddr *as* hA*, and* oAddr *as* oA*):*

M1({name}, {phone, hP, oP}, {address, hA, oA})
M2({name}, {phone, hP}, {oP}, {address, oA}, {hA})

Possible Mapping	Probability
{(name, name), (hP, hPP), (oP, oP), (hA, hAA), (oA, oA)}	0.64
{(name, name), (hP, hPP), (oP, oP), (oA, hAA), (hA, oA)}	0.16
{(name, name), (oP, hPP), (hP, oP), (hA, hAA), (oA, oA)}	0.16
{(name, name), (oP, hPP), (hP, oP), (oA, hAA), (hA, oA)}	0.04

(a)

Possible Mapping	Probability
{(name, name), (oP, oPP), (hP, hP), (oA, oAA), (hA, hA)}	0.64
{(name, name), (oP, oPP), (hP, hP), (hA, oAA), (oA, hA)}	0.16
{(name, name), (hP, oPP), (oP, hP), (oA, oAA), (hA, hA)}	0.16
{(name, name), (hP, oPP), (oP, hP), (hA, oAA), (oA, hA)}	0.04

(b)

Answer	Probability
('Alice', '123-4567', '123, A Ave.')	0.34
('Alice', '765-4321', '456, B Ave.')	0.34
('Alice', '765-4321', '123, A Ave.')	0.16
('Alice', '123-4567', '456, B Ave.')	0.16

(c)

Figure 1.5. The motivating example: (a) p-mapping for S_1 and M_3, (b) p-mapping for S_1 and M_4, and (c) query answers w.r.t. $\mathbf{M}$ and $\mathbf{pM}$. Here we denote {phone, hP} by hPP, {phone, oP} by oPP, {address, hA} by hAA, and {address, oA} by oAA.

M3({name}, {phone, hP}, {oP}, {address, hA}, {oA})
M4({name}, {phone, oP}, {hP}, {address, oA}, {hA})
M5({name}, {phone}, {hP}, {oP}, {address}, {hA}, {oA})

None of the listed mediated schemas is perfect. Schema M_1 groups multiple attributes from S1. M_2 seems inconsistent because phone *is grouped with* hPhone *while* address *is grouped with* oAddress*. Schemas M_3, M_4 and M_5 are partially correct but none of them captures the fact that* phone *and* address *can be either home phone and home address, or office phone and office address.*

Even if we introduce probabilistic schema mappings, none of the listed mediated schemas will return ideal answers. For example, using M_1 prohibits returning correct answers for queries that contain both hPhone *and* oPhone *because they are taken to be the same attribute. As another example, consider a query that contains* phone *and* address*. Using M_3 or M_4 as the mediated schema will unnecessarily favor home address and phone over office address and phone or vice versa. A system with M_2 will incorrectly favor answers that return a person's home address together with office phone number. A system with M_5 will also return a person's home address together with office phone, and does not distinguish such answers from answers with correct correlations.*

A probabilistic mediated schema will avoid this problem. Consider a probabilistic mediated schema $\mathbf{M}$ that includes M_3 and M_4, each with probability 0.5. For each of them and each source schema, we generate a probabilistic mapping (Section 3). For example, the set of probabilistic mappings $\mathbf{pM}$ for S_1 is shown in Figure 1.5(a) and (b).

Now consider an instance of S_1 with a tuple

```
('Alice', '123-4567', '123, A Ave.',
          '765-4321', '456, B Ave.')
```

and a query

```
SELECT name, phone, address
FROM People
```

The answer generated by our system with respect to $\mathbf{M}$ and $\mathbf{pM}$ is shown in Figure 1.5(c). (As we describe in detail in the following sections, we allow users to compose queries using any attribute in the source.) Compared with using one of M_2 to M_5 as a mediated schema, our method generates better query results in that (1) it treats answers with home address and home phone and answers with office address and office phone equally, and (2) it favors answers with the correct correlation between address and phone number. $\square$

4.2 Probabilistic Mediated Schema

Consider a set of source schemas $\{S_1, \ldots, S_n\}$. We denote the attributes in schema $S_i, i \in [1, n]$, by $attr(S_i)$, and the set of all source attributes as $\mathcal{A}$. That is, $\mathcal{A} = attr(S_1) \cup \ldots \cup attr(S_n)$. We denote a mediated schema for the set of sources $\{S_1, \ldots, S_n\}$ by $M = \{A_1, \ldots, A_m\}$, where each of the A_i's is called a *mediated attribute*. The mediated attributes are *sets* of attributes from the sources, i.e., $A_i \subseteq \mathcal{A}$; for each $i, j \in [1, m], i \neq j \Rightarrow A_i \cap A_j = \emptyset$.

Note that whereas in a traditional mediated schema an attribute has a name, we do not deal with naming of an attribute in our mediated schema and allow users to use any source attribute in their queries. (In practice, we can use the most frequent source attribute to represent a mediated attribute when exposing the mediated schema to users.) If a query contains an attribute $a \in A_i, i \in [1, m]$, then when answering the query we replace a everywhere with A_i.

A *probabilistic mediated schema* consists of a set of mediated schemas, each with a probability indicating the likelihood that the schema correctly describes the domain of the sources. We formally define probabilistic mediated schemas as follows.

DEFINITION 1.25 (PROBABILISTIC MEDIATED SCHEMA) *Let* $\{S_1, \ldots, S_n\}$ *be a set of schemas. A* probabilistic mediated schema (p-med-schema) for $\{S_1, \ldots, S_n\}$ *is a set*

$$\mathbf{M} = \{(M_1, Pr(M_1)), \ldots, (M_l, Pr(M_l))\}$$

where

- *for each* $i \in [1, l]$*,* M_i *is a mediated schema for* $S_1, \ldots, S_n$*, and for each* $i, j \in [1, l], i \neq j$*,* M_i *and* M_j *correspond to different clusterings of the source attributes;*
- $Pr(M_i) \in (0, 1]$*, and* $\Sigma_{i=1}^{l} Pr(M_i) = 1$. □

Semantics of queries: Next we define the semantics of query answering with respect to a p-med-schema and a set of p-mappings for each mediated schema in the p-med-schema. Answering queries with respect to p-mappings returns a set of answer tuples, each with a probability indicating the likelihood that the tuple occurs as an answer. We consider by-table semantics here. Given a query Q, we compute answers by first answering Q with respect to each possible mapping, and then for each answer tuple t summing up the probabilities of the mappings with respect to which t is generated.

We now extend this notion for query answering that takes p-med-schema into consideration. Intuitively, we compute query answers by first answering the query with respect to each possible mediated schema, and then for each answer tuple taking the sum of its probabilities weighted by the probabilities of the mediated schemas.

DEFINITION 1.26 (QUERY ANSWER) *Let* S *be a source schema and* $\mathbf{M} = \{(M_1, Pr(M_1)), \ldots, (M_l, Pr(M_l))\}$ *be a p-med-schema. Let* $\mathbf{pM} = \{pM(M_1), \ldots, pM(M_l)\}$ *be a set of p-mappings where* $pM(M_i)$ *is the p-mapping between* S *and* M_i*. Let* D *be an instance of* S *and* Q *be a query.*

Let t *be a tuple. Let* $Pr(t|M_i), i \in [1, l]$*, be the probability of* t *in the answer of* Q *with respect to* M_i *and* $pM(M_i)$*. Let* $p = \Sigma_{i=1}^{l} Pr(t|M_i) * Pr(M_i)$*. If* $p > 0$*, then we say* (t, p) *is a by-table answer with respect to* $\mathbf{M}$ *and* $\mathbf{pM}$*.*

We denote all by-table answers by $Q_{\mathbf{M},\mathbf{pM}}(D)$*.* □

We say that query answers A_1 and A_2 are *equal* (denoted $A_1 = A_2$) if A_1 and A_2 contain exactly the same set of tuples with the same probability assignments.

Expressive power: A natural question to ask at this point is whether probabilistic mediated schemas provide any added expressive power compared to deterministic ones. Theorem 1.27 shows that if we consider *one-to-many* schema mappings, where one source attribute can be mapped to multiple mediated attributes, then any combination of a p-med-schema and p-mappings can be equivalently represented using a deterministic mediated schema with p-mappings, but may not be represented using a p-med-schema with deterministic schema mappings. Note that we can easily extend the definition of query answers to one-to-many mappings as one mediated attribute can correspond to no more than one source attribute.

THEOREM 1.27 (SUBSUMPTION) *The following two claims hold.*

1 *Given a source schema S, a p-med-schema $\mathbf{M}$, and a set of p-mappings $\mathbf{pM}$ between S and possible mediated schemas in $\mathbf{M}$, there exists a deterministic mediated schema T and a p-mapping pM between S and T, such that $\forall D, Q : Q_{\mathbf{M},\mathbf{pM}}(D) = Q_{T,pM}(D)$.*

2 *There exists a source schema S, a mediated schema T, a p-mapping pM between S and T, and an instance D of S, such that for any p-med-schema $\mathbf{M}$ and any set $\mathbf{m}$ of deterministic mappings between S and possible mediated schemas in $\mathbf{M}$, there exists a query Q such that $Q_{\mathbf{M},\mathbf{m}}(D) \neq Q_{T,pM}(D)$.* □

In contrast, Theorem 1.28 shows that if we restrict our attention to one-to-one mappings, then a probabilistic mediated schema *does* add expressive power.

THEOREM 1.28 *There exists a source schema S, a p-med-schema $\mathbf{M}$, a set of one-to-one p-mappings $\mathbf{pM}$ between S and possible mediated schemas in $\mathbf{M}$, and an instance D of S, such that for any deterministic mediated schema T and any one-to-one p-mapping pM between S and T, there exists a query Q such that, $Q_{\mathbf{M},\mathbf{pM}}(D) \neq Q_{T,pM}(D)$.* □

Constructing one-to-many p-mappings in practice is much harder than constructing one-to-one p-mappings. And, when we are restricted to one-to-one p-mappings, p-med-schemas grant us more expressive power while keeping the process of mapping generation feasible.

4.3 P-med-schema Creation

We now show how to create a probabilistic mediated schema $\mathbf{M}$. Given source tables $S_1, \ldots, S_n$, we first construct the multiple schemas $M_1, \ldots, M_p$ in $\mathbf{M}$, and then assign each of them a probability.

We exploit two pieces of information available in the source tables: (1) pairwise similarity of source attributes; and (2) statistical co-occurrence properties of source attributes. The former will be used for creating multiple mediated schemas, and the latter for assigning probabilities on each of the mediated schemas.

The first piece of information tells us when two attributes are likely to be similar, and is generated by a collection of schema matching modules. This information is typically given by some pairwise attribute similarity measure, say s. The similarity $s(a_i, a_j)$ between two source attributes a_i and a_j depicts how closely the two attributes represent the same real-world concept.

The second piece of information tells us when two attributes are likely to be different. Consider for example, source table schemas

```
S1: (name,address,email-address)
S2: (name,home-address)
```

Pairwise string similarity would indicate that attribute address can be similar to both email-address and home-address. However, since the first source table contains address and email-address together, they cannot refer to the same concept. Hence, the first table suggests address is different from email-address, making it more likely that address refers to home-address.

Creating Multiple Mediated Schemas: The creation of the multiple mediated schemas constituting the p-med-schema can be divided conceptually into three steps. First, we remove infrequent attributes from the set of all source attributes; that is, attribute names that do not appear in a large fraction of source tables. This step ensures that our mediated schema contains only information that is relevant and central to the domain. In the second step we construct a weighted graph whose nodes are the attributes that survived the filter of the first step. An edge in the graph is labeled with the pairwise similarity between the two nodes it connects. Finally, several possible clusterings of nodes in the resulting weighted graph give the various mediated schemas.

Algorithm 1 describes the various steps in detail. The input is the set of source schemas creating $S_1, \ldots, S_n$ and a pairwise similarity function s, and the output is the multiple mediated schemas in $\mathbf{M}$. Steps 1–3 of the algorithm find the attributes that occur frequently in the sources. Steps 4 and 5 construct the graph of these high-frequency attributes. We allow an error ϵ on the threshold τ for edge weights. We thus have two kinds of edges: *certain edges*, having weight at least $\tau + \epsilon$, and *uncertain edges*, having weight between $\tau - \epsilon$ and $\tau + \epsilon$.

Steps 6-8 describe the process of obtaining multiple mediated schemas. Specifically, a mediated schema in $\mathbf{M}$ is created for every subset of the uncertain edges. For every subset, we consider the graph resulting from omitting that subset from the graph. The mediated schema includes a mediated attribute for each connected component in the resulting graph. Since, in the worst case,

```
0: Input: Source schemas S_1, ..., S_n.
   Output: A set of possible mediated schemas.
1: Compute A = {a_1, ..., a_m}, the set of all source attributes;
2: for each (j ∈ [1, m])
       Compute frequency f(a_j) = |{i ∈ [1, n] | a_j ∈ S_i}| / n;
3: Set A = {a_j | j ∈ [1, m], f(a_j) ≥ θ}; //θ is a threshold
4: Construct a weighted graph G(V, E), where (1) V = A, and (2) for each
   a_j, a_k ∈ A, s(a_j, a_k) ≥ τ − ϵ, there is an edge (a_j, a_k) with weight
   s(a_j, a_k);
5: Mark all edges with weight less than τ + ϵ as uncertain;
6: for each (uncertain edge e = (a_1, a_2) ∈ E)
       Remove e from E if (1) a_1 and a_2 are connected by a path with only
   certain edges, or (2) there exists a_3 ∈ V, such that a_2 and a_3 are connected
   by a path with only certain edges and there is an uncertain edge (a_1, a_3);
7: for each (subset of uncertain edges)
       Omit the edges in the subset and compute a mediated schema where
   each connected component in the graph corresponds to an attribute in the
   schema;
8: return distinct mediated schemas.
```

Algorithm 1: Generate all possible mediated schemas.

the number of resulting graphs is exponential in the number of uncertain edges, the parameter ϵ needs to be chosen carefully. In addition, Step 6 removes uncertain edges that when omitted will not lead to different mediated schemas. Specifically, we remove edges that connect two nodes already connected by certain edges. Also, we consider only one among a set of uncertain edges that connect a particular node with a set of nodes that are connected by certain edges.

Probability Assignment: The next step is to compute probabilities for possible mediated schemas that we have generated. As a basis for the probability assignment, we first define when a mediated schema is *consistent with* a source schema. The probability of a mediated schema in $\mathbf{M}$ will be the proportion of the number of sources with which it is consistent.

DEFINITION 1.29 (CONSISTENCY) *Let M be a mediated schema for sources $S_1, \ldots, S_n$. We say M is* consistent with *a source schema $S_i, i \in [1, n]$, if there is no pair of attributes in S_i that appear in the same cluster in M.*

Intuitively, a mediated schema is consistent with a source only if it does not group distinct attributes in the source (and hence distinct real-world concepts)

Input: Possible mediated schemas $M_1, \ldots, M_l$ and source schemas $S_1, \ldots, S_n$.
Output: $Pr(M_1), \ldots, Pr(M_l)$.
for each ($i \in [1, l]$)
    Count the number of source schemas that are consistent with M_i, denoted as c_i;
for each ($i \in [1, l]$) Set $Pr(M_i) = \frac{c_i}{\sum_{i=1}^{l} c_i}$.

Algorithm 2: Assign probabilities to possible mediated schemas.

Input: Mediated schemas $M_1, \ldots, M_l$.
Output: A consolidated single mediated schema T.
Set T = M_1.
for ($i = 2, \ldots, l$) modify T as follows:
    for each (attribute A' in M_i)
        for each (attribute A in T)
            Divide A into $A \cap A'$ and $A - A'$;
return T.

Algorithm 3: Consolidate a p-med-schema.

into a single cluster. Algorithm 2 shows how to use the notion of consistency to assign probabilities on the p-med-schema.

4.4 Consolidation

To complete the fully automatic setup of the data integration system, we consider the problem of consolidating a probabilistic mediated schema into a single mediated schema and creating p-mappings to the consolidated schema. We require that the answers to queries over the consolidated schema be equivalent to the ones over the probabilistic mediated schema.

The main reason to consolidate the probabilistic mediated schema into a single one is that the user expects to see a single schema. In addition, consolidating to a single schema has the advantage of more efficient query answering: queries now need to be rewritten and answered based on only one mediated schema. We note that in some contexts, it may be more appropriate to show the application builder a set of mediated schemas and let her select one of them (possibly improving on it later on).

Consolidating a p-med-schema: Consider a p-med-schema $\mathbf{M} = \{(M_1, Pr(M_1)), \ldots, (M_l, Pr(M_l))\}$. We consolidate $\mathbf{M}$ into a single mediated schema T. Intuitively, our algorithm (see Algorithm 3) generates the "coarsest refinement" of the possible mediated schemas in $\mathbf{M}$ such that every cluster in any of the M_i's is equal to the union of a set of clusters in T. Hence, any two attributes a_i and a_j will be together in a cluster in T if and only if they are together in every mediated schema of $\mathbf{M}$. The algorithm initializes T to M_1 and then modifies each cluster of T based on clusters from M_2 to M_l.

EXAMPLE 1.30 *Consider a p-med-schema $M = \{M_1, M_2\}$, where M_1 contains three attributes $\{a_1, a_2, a_3\}$, $\{a_4\}$, and $\{a_5, a_6\}$, and M_2 contains two attributes $\{a_2, a_3, a_4\}$ and $\{a_1, a_5, a_6\}$. The target schema T would then contain four attributes: $\{a_1\}$, $\{a_2, a_3\}$, $\{a_4\}$, and $\{a_5, a_6\}$.* □

Note that in practice the consolidated mediated schema is the same as the mediated schema that corresponds to the weighted graph with only certain edges. Here we show the general algorithm for consolidation, which can be applied even if we do not know the specific pairwise similarities between attributes.

Consolidating p-mappings: Next, we consider consolidating p-mappings specified w.r.t. $M_1, \ldots, M_l$ to a p-mapping w.r.t. the consolidated mediated schema T. Consider a source S with p-mappings $pM_1, \ldots, pM_l$ for $M_1, \ldots, M_l$ respectively. We generate a single p-mapping pM between S and T in three steps. First, we modify each p-mapping $pM_i, i \in [1, l]$, between S and M_i to a p-mapping pM_i' between S and T. Second, we modify the probabilities in each pM_i'. Third, we consolidate all possible mappings in pM_i''s to obtain pM. The details are as follows.

1. **For each** $i \in [1, l]$**, modify p-mapping** pM_i**:** Do the following for every possible mapping m in pM_i:

   - For every correspondence $(a, A) \in m$ between source attribute a and mediated attribute A in M_i, proceed as follows. (1) Find the set of all mediated attributes B in T such that $B \subset A$. Call this set $\overline{B}$. (2) Replace (a, A) in m with the set of all (a, B)'s, where $B \in \overline{B}$.

   Call the resulting p-mapping pM_i'.
2. **For each** $i \in [1, l]$**, modify probabilities in** pM_i'**:** Multiply the probability of every schema mapping in pM_i' by $Pr(M_i)$, which is the probability of M_i in the p-med-schema. (Note that after this step the sum of probabilities of all mappings in pM_i' is not 1.)
3. **Consolidate** pM_i'**'s:** Initialize pM to be an empty p-mapping (i.e., with no mappings). For each $i \in [1, l]$, *add* pM_i' to pM as follows:

   - For each schema mapping m in pM_i' with probability p: *if m is in pM, with probability p'*, modify the probability of m in pM to $(p + p')$; *if m is not in pM*, then add m to pM with probability p.

   The resulting p-mapping, pM, is the final consolidated p-mapping. The probabilities of all mappings in pM add to 1.

Note that Step 2 can map one source attribute to multiple mediated attributes; thus, the mappings in the result pM are one-to-many mappings, and so typically different from the p-mapping generated directly on the consolidated schema. The following theorem shows that the consolidated mediated schema and the consolidated p-mapping are equivalent to the original p-med-schema and p-mappings.

THEOREM 1.31 (MERGE EQUIVALENCE) *For all queries Q, the answers obtained by posing Q over a p-med-schema $\mathbf{M} = \{M_1, \ldots, M_l\}$ with p-mappings $pM_1, \ldots, pM_l$ is equal to the answers obtained by posing Q over the consolidated mediated schema T with consolidated p-mapping pM.* □

4.5 Other approaches

He and Chang [21] considered the problem of generating a mediated schema for a set of web sources. Their approach was to create a mediated schema that is statistically maximally *consistent* with the source schemas. To do so, they assume that the source schemas are created by a *generative model* applied to some mediated schema, which can be thought of as a probabilistic mediated schema. The probabilistic mediated schema we described in this chapter has several advantages in capturing heterogeneity and uncertainty in the domain. We can express a wider class of attribute clusterings, and in particular clusterings that capture attribute correlations. Moreover, we are able to combine attribute matching and co-occurrence properties for the creation of the probabilistic mediated schema, allowing for instance two attributes from one source to have a nonzero probability of being grouped together in the mediated schema. Also, the approach for p-med-schema creation described in this chapter is independent of a specific schema-matching technique, whereas the approach in [21] is tuned for constructing generative models and hence must rely on statistical properties of source schemas.

Magnani et al. [30] proposed generating a set of alternative mediated schemas based on probabilistic relationships between *relations* (such as an Instructor relation intersects with a Teacher relation but is disjoint with a Student relation) obtained by sampling the overlapping of data instances. Here we focus on matching attributes within relations. In addition, our approach allows exploring various types of evidence to improve matching and we assign probabilities to the mediated schemas we generate.

Chiticariu et. al. [5] studied the generation of multiple mediated schemas for an existing set of data sources. They consider multi-table data sources, not considered in this chapter, but explore interactive techniques that aid humans in arriving at the mediated schemas.

There has been quite a bit of work on automatically creating mediated schemas that focused on the theoretical analysis of the semantics of merging schemas and the choices that need to be made in the process [2, 4, 23, 25, 31, 33]. The goal of these work was to make as many decisions automatically as possible, but where some ambiguity arises, refer to input from a designer.

5. Future Directions

The investigation of data integration with uncertainty is only beginning. This chapter described some of the fundamental concepts on which such systems will be built, but there is a lot more to do.

The main challenge is to build actual data integration systems that incorporate uncertainty and thereby uncover a new set of challenges, such as efficiency and understanding what are the common types of uncertainty that arise in data integration applications, so techniques can be tailored for these cases.

The work we described showed how to create p-mediated schemas and schema mappings automatically. This is only a way to bootstrap a pay-as-you-go integration system. The next challenge is to find methods to improve it over time (see [24] for a first work on doing so). We would also like to incorporate multi-table sources, rather than only single-table ones as we described so far.

Finally, when we have many data sources, the sources tend to be redundant and contain dependencies (and therefore not offer independent sources of evidence). An important line of work is to discover these dependencies and use them to provide more precise answers to queries. We are currently exploring how the formalism and techniques from this chapter can be extended to consider uncertain and interdependent data sources, and how query answering can be performed efficiently even in the presence of dependencies.

References

[1] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A system for keyword-based search over relational databases. In *Proc. of ICDE*, pages 5–16, 2002.

[2] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. In *ACM Computing Surveys*, pages 323–364, 1986.

[3] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proc. of the 14th Int. Conf. on Advanced Information Systems Eng. (CAiSE02)*, 2002.

[4] P. Buneman, S. Davidson, and A. Kosky. Theoretical aspects of schema merging. In *Proc. of EDBT*, 1992.

[5] L. Chiticariu, P. G. Kolaitis, and L. Popa. Interactive generation of integrated schemas. In *Proc. of ACM SIGMOD*, 2008.

[6] A. Das Sarma, X. Dong, and A. Halevy. Bootstrapping pay-as-you-go data integration systems. Technical report, 2008. Available at http://dbpubs.stanford.edu/pub/2008-8.

[7] R. Dhamankar, Y. Lee, A. Doan, A. Y. Halevy, and P. Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proc. of ACM SIGMOD*, 2004.

[8] H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proc. of VLDB*, 2002.

[9] A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Learning to map between ontologies on the Semantic Web. In *Proc. of the Int. WWW Conf.*, 2002.

[10] X. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. In *Proc. of VLDB*, 2007.

[11] Xin Dong and Alon Y. Halevy. A platform for personal information management and integration. In *Proc. of CIDR*, 2005.

[12] Alon Levy (Ed.). Data engineering special issue on adaptive query processing, june 2000. *IEEE Data Eng. Bull.*, 23(2), 2000.

[13] D. Florescu, D. Koller, and Alon Y. Levy. Using probabilistic information in data integration. In *Proc. of VLDB*, 1997.

[14] A. Gal. Why is schema matching tough and what can we do about it? *SIGMOD Record*, 35(4):2–5, 2007.

[15] A. Gal, G. Modica, H. Jamil, and A. Eyal. Automatic ontology matching using application semantics. *AI Magazine*, 26(1):21–31, 2005.

[16] Avigdor Gal, Ateret Anaby-Tavor, Alberto Trombetta, and Danilo Montesi. A framework for modeling and evaluating automatic semantic reconciliation. 2003.

[17] GoogleBase. http://base.google.com/, 2005.

[18] A. Y. Halevy, N. Ashish, D. Bitton, M. J. Carey, D. Draper, J. Pollock, A. Rosenthal, and V. Sikka. Enterprise information integration: successes, challenges and controversies. In *SIGMOD*, 2005.

[19] A. Y. Halevy, A. Rajaraman, and J. J. Ordille. Data integration: The teenage years. In *VLDB*, 2006.

[20] Alon Y. Halevy, Michael J. Franklin, and David Maier. Principles of dataspace systems. In *PODS*, 2006.

[21] B. He and K. C. Chang. Statistical schema matching across web query interfaces. In *Proc. of ACM SIGMOD*, 2003.

[22] V. Hristidis and Y. Papakonstantinou. DISCOVER: Keyword search in relational databases. In *Proc. of VLDB*, pages 670–681, 2002.

[23] R. Hull. Relative information capacity of simple relational database schemata. In *Proc. of ACM PODS*, 1984.

[24] S. Jeffery, M. Franklin, and A. Halevy. Pay-as-you-go user feedback for dataspace systems. In *Proc. of ACM SIGMOD*, 2008.

[25] L. A. Kalinichenko. Methods and tools for equivalent data model mapping construction. In *Proc. of EDBT*, 1990.

[26] J. Kang and J. Naughton. On schema matching with opaque column names and data values. In *Proc. of ACM SIGMOD*, 2003.

[27] J. Madhavan, S. Cohen, X. Dong, A. Halevy, S. Jeffery, D. Ko, and C. Yu. Web-scale data integration: You can afford to pay as you go. In *Proc. of CIDR*, 2007.

[28] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's deep-web crawl. In *Proc. of VLDB*, 2008.

[29] M. Magnani and D. Montesi. Uncertainty in data integration: current approaches and open problems. In *VLDB workshop on Management of Uncertain Data*, pages 18–32, 2007.

[30] M. Magnani, N. Rizopoulos, P. Brien, and D. Montesi. Schema integration based on uncertain semantic mappings. *Lecture Notes in Computer Science*, pages 31–46, 2005.

[31] R. J. Miller, Y. Ioannidis, and R. Ramakrishnan. The use of information capacity in schema integration and translation. In *Proc. of VLDB*, 1993.

[32] H. Nottelmann and U. Straccia. Information retrieval and machine learning for probabilistic schema matching. *Information Processing and Management*, 43(3):552–576, 2007.

[33] R. Pottinger and P. Bernstein. Creating a mediated schema based on initial correspondences. In *IEEE Data Eng. Bulletin*, pages 26–31, Sept 2002.

[34] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[35] J. Wang, J. Wen, F. H. Lochovsky, and W. Ma. Instance-based schema matching for Web databases by domain-specific query probing. In *Proc. of VLDB*, 2004.