Integration of Multiple Uncertain Data Sources

Wei Han

A thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

January 2016

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Wei Han**

Entitled: **Integration of Multiple Uncertain Data Sources**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

Gregory Butler _____ Chair

Gösta G. Grahne _____ Examiner

Joey Paquet _____ Examiner

Nematollaah Shiri _____ Supervisor

Approved **Sudhir P. Mudur**
Chair of Department or Graduate Program Director

January 25, 2016 **Amir Asif**

Amir Asif, Ph.D.,ing., Dean
Faculty of Engineering and Computer Science

Abstract

Integration of Multiple Uncertain Data Sources

Wei Han

Data integration is the problem of combining data from multiple autonomous data sources, and providing a unified view to the users. The problem has been studied extensively over the past two decades, and focused more on integrating traditional, exact relational data. Integration over uncertain data sources is a more recent problem and a more challenging one. The purpose of this thesis is to understand the semantics and techniques of uncertain data integration over multiple such data sources. We study existing proposals for uncertain and probabilistic data integration. As a basis of our work, we consider two integration operations, one in the possible worlds model, and the other in a compact model. We introduce the properties of the integration operations proposed for two sources, and consider these properties to develop a framework for integrating multiple sources. For this, we also extend and generalize a conversion algorithm from possible worlds model to the compact probabilistic relations. We define the integration procedure, the concept of probability consistency, and a probability adjustment method when the consistency is violated. We build a running prototype of the proposed framework to show its feasibility and to automate the probability calculation. This thesis makes a step forward to better understand the challenges and development of uncertain data integration systems.

Acknowledgments

I would like to express my sincere gratitude and respect to my supervisor, professor Nematollaah Shiri. His wisdom, knowledge and experiences help with many insightful conversations. Many ideas in this research would have not been well developed without his support and patience. I especially appreciate the philosophy he conveyed towards science, learning and life in general, which will lead me beyond the graduate study.

I am grateful to the faculty members and staff in our department for their teaching and help.

I would like to thank my friends Zahra Asadi, Soyoung Kim, Shahab Harrafi, Ali Moallemi, Iraj Hedayati, Ali Nikzad and many more, for all the discussions and support.

My gratitude goes to my family for their unconditional love and support.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

DBMS	Database Management System
epr-relation	Extended Probabilistic Relation
EVF	Event Variable Formula
IEV	Interpreted Event Variable
OLAP	Online Analytical Processing
ppr-relation	Pure Probabilistic Relation
pr-relation	Probabilistic Relation
PW	Possible World

Chapter 1

Introduction

Data integration is a challenge for applications that need to query over multiple autonomous data sources. In these sources, data is produced independently by different applications. A typical scenario is an academic institute in which different departments maintain their own databases. If student Tom is admitted as a research assistant with some grant, the Grants Office needs to record this information, and Human Resource Department also needs to have this information for payroll. The databases in this scenario do not interact with each other to further facilitate various operations in the institute. In many scenarios like this, constructing one database to support all these operations at the institute level seems to be a good solution. However, this fresh construction would be costly in time and budget. Moreover, the transition for data and users to a new solution is difficult and time-consuming. A better solution

is to build a layer of abstraction on top of existing databases, to provide a unified interface for accessing various data sources, while data sources continue serving their applications as before. Data Integration comes into picture in this sense.

Data integration has been studied extensively for relational data for more than two decades, resulted in numerous models and algorithms [6][9][12][15][18]. Integration of uncertain data, however, has attracted the attention of researchers in recent years. Kiani and Shiri proposed an integration framework for uncertain data represented in the Information Source Tracking (IST) model [14]. More recent work focused on uncertain data integration in the possible worlds semantics [4][5][19][22].

A piece of data is *uncertain* if its truth is not established definitely. In some applications, a numerical probability or confidence value is associated with the uncertain data, indicating the likelihood of its existence. Such data is called *probabilistic data*. In a real world scenario, data collected by various applications, such as web data extraction, sensor data detection, data integration and human observation, are often uncertain. These applications create an increasing need for handling uncertain data. While uncertain data modelling and processing have been popular research topics for about three decades, emerging applications require unified data access and data-driven decision making. This motivates recent research on integration of uncertain data, and the focus of this thesis.

Integration allows sharing and collaboration among data sources. In addition

to this common benefit of traditional data integration techniques, uncertain data integration can help resolve partial uncertainty and inconsistency among individual sources. The integration result provides more information than any single source and decreases uncertainty about the data sources [3]. We will elaborate on this benefit later.

The goal of this research is to study the semantics and techniques of uncertain data integration, and develop a framework for integration of multiple uncertain data sources. This extends and complements existing solutions.

1.1 Motivation

As a simple example, let us consider the information collected by police from witnesses to a robbery. The number of robbers is unknown. Witness A saw a man in a black shirt and a man in a denim jacket, one of whom must be a suspicious robber. Witness B indicates that a man in a black shirt or the man wearing a green shirt may be the robber. Even though data is recorded separately, we would like to combine and look at these observations together. First, we represent these information in possible worlds model. Possible worlds model [1] is widely recognized as the semantics of uncertain data: an uncertain relation is represented by a set of possible instance of that relation, each of which is a conventional relation. In this example, possible worlds of witnesses' statements are shown as Tables 1.1 and 1.2. For instance, in Table 1.1, D_1 and D_2

are two possible worlds for S_1.

Suspect (gender, top)
(male, black)

D_1

Suspect (gender, top)
(male, denim)

D_2

Table 1.1: Possible worlds of source S_1 (witness A)

Suspect (gender, top)
(male, black)

D_3

Suspect (gender, top)
(male, green)

D_4

Table 1.2: Possible worlds of source S_2 (witness B)

An intuitive way to combine these information is to generate the pairwise union of possible worlds of suspect relation. This gives four possible results in Table 1.3. However, according to the witnesses, the robber wears either a green shirt or a black

Suspect (gender, top)
(male, black)

D_{I1}

Suspect (gender, top)
(male, black)
(male, green)

D_{I2}

Suspect (gender, top)
(male, denim)
(male, black)

D_{I3}

Suspect (gender, top)
(male, denim)
(male, green)

D_{I4}

Table 1.3: Possible worlds of the pairwise union result

shirt. Another possibility is that the robber is in either denim or black. We need to

resolve the contradictory information from different sources. Since the combinations {green, black} and {black, denim} are invalid, then they should not be among the integrated results. The final possible results are shown in Table 1.4. Either there is

Suspect (gender, top)
(male, black)

D_1

Suspect (gender, top)
(male, green)
(male, denim)

D_2

Table 1.4: Possible worlds of the integration result

one robber who wears a black top, or there are two robbers, one in a green top and the other in a denim top.

Traditional databases cannot reflect such uncertainty in data, nor can traditional integration processes apply directly here. A desired integration framework should take uncertain data from all sources and process it with respect to inter-dependency within data.

We will review existing work and applications that motivate integration over such uncertain data. We start by introducing some applications that generate uncertain data.

Sources of Uncertain Data

Uncertain data is generated and/or processed in some application such as the follows.

- Web data extraction[11]: Web crawling extracts data from unstructured or semi-structured web sources by automatic analysis of the content. This inherent uncertain process introduces uncertain data. User A talks a lot about company B in his blog. We may conclude that A works for B with a probability of 0.8. When sources are not reliable, there can be a certain degree of confidence associated with the collected data. Media A reports that a new phone will be released in October, while Media B reports that the release date is next March. If A is mainstream and more reliable, then the probability of an October release date would be greater than the one in March.

- Sensor data detection: Due to environmental limitations, there may be a certain deviation in the data. For example, station A measures the temperature as 90F, but station B indicates a temperature of 88F. Data collected can be 90F and 88F with equal probabilities of 0.5. This preserves the raw data and retains more information than simply cleaning the data.

- Data integration process[10][16]: This type of uncertainty is generated during schema mapping. Research work in [2] contributes an automated integration framework for exact data. Given a set of data sources with different schemas, the framework automatically generates a set of possible mediated schemas, each one with a probability value attached. For every proposed mediated schema,

there is a set of mapping plans with a numerical probability value associated. Uncertainty comes from automated two-step mapping during integration.

- Human Observation: As mentioned in [23], *Christmas Bird Count (CBC)*, observations from a single spot by an individual may not be reliable. The observer is not sure about the exact category of the bird. It might be a Red-bellied Woodpecker or Golden-fronted Woodpecker.

Due to the nature of uncertain data, traditional *database management systems (DBMS)* are inadequate for uncertain data representation and management. Given such uncertain data, especially probabilistic data, traditional data integration frameworks cannot handle the uncertainty. A number of solutions have been proposed on uncertain data modelling. We will mainly discuss two important models in Chapter 2: possible worlds model and probabilistic relation model.

When the data sources are uncertain databases, in addition to the problems in traditional data integration, new challenges arise. They are described as follows.

- Semantics of integration: The meaning of the correct integration result is different from traditional integration. Instead of union data of the same relation from different sources, an uncertain integration framework needs to take data from all the sources and resolve any inconsistency and conflict, without any unreasonable loss of information.

- Probability manipulation: When the uncertain data sources are probabilistic, there is a probability distribution defined over the set of possible worlds. Probability values are taken into account during the integration process.

- Semantics of query: The purpose of a data integration system is to handle queries and provide results. How uncertain data integration affects the query of the data should be studied. Further more, query evaluation algorithms needs to be revised and developed in uncertain data integration.

Our research concentrates on the semantics and techniques of data integration. This is a major challenge for uncertain data integration. In traditional data integration, as data is exact, integrated result is usually the union of data collected from sources.

1.2 Contributions

Research in [4] investigates the problem of uncertain data integration. However the paper does not provide a concrete integration process. It defines different semantics standards for evaluation of uncertain data integration system. To integrate uncertain data sources that have interdependencies, [19] proposes a framework using *logical representation* for uncertain and probabilistic data integration. A more compact and practical approach is proposed in their follow-up work [5]. Our research is inspired by them. However, both of the works restricts the problem of probabilistic data

integration to only two sources, our work extends these results to multiple sources. Using these as the basis, the contributions of this thesis are as follows.

- We investigate properties of the proposed integration frameworks mentioned above for uncertain data and probabilistic data. We demonstrate the limitations and issues.

- We extend existing work and develop an integration framework for multiple uncertain data and probabilistic data sources that adapts the compact representation model. We generalize the algorithm to convert a set of possible worlds of an uncertain database to a compact uncertain database. We define the integration procedure, the probability consistency of the data sources, and a heuristic method to adjust probability of data sources when the consistency is violated.

- We explore the impact of integration on querying of the data sources. It in return helps us better understand the semantics of uncertain data integration.

- We build a running prototype of the proposed integration framework to show its feasibility and an integration-query system prototype to help compare query results.

Finally, we define future challenges in this work.

1.3 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 reviews the related work as background. For this, we review uncertain data models and existing uncertain data integration frameworks. In Chapter 3, we investigate idempotent, commutativity and associativity of existing integration processes. Based on these properties, we propose an extended framework to integrate multiple uncertain data sources. Chapter 4 presents detailed design and implementation of the prototypes developed. We discuss the relationship between integration and query. Finally, Chapter 5 includes concluding remarks and ideas for future work.

Chapter 2

Background and Related Work

In this chapter, we study the foundations of uncertain data integration and review the related work. In Section 2.1, we focus on the semantics of uncertain data and consider a more practical compact model. Section 2.2 reviews the proposed frameworks for integrating uncertain data sources.

2.1 Uncertain Data Modeling

Possible Worlds Semantics

As mentioned in Chapter 1, the possible worlds model is well recognized as the semantics of uncertain data: an uncertain relation represents a set of standard relations, one of which is the true state of the relation in the real world, but we do not know

exactly which one. An uncertain database is defined as a set of uncertain relations. Recall that in standard databases, the semantics of a relation is the relation itself. In addition to the notion of possible worlds, for an uncertain relation U, [19] also considers a finite set of associated tuples T(U). If the tuple set is not provided explicitly, it is generated by the union of all the tuples in the possible worlds. We use the following example from [19] to illustrate this notion.

Example 2.1. Consider two students Sara and Tom who are talking about their courses and those of a fellow student Bob. Sara says that she has taken CS100, CS101, and CS102, and Bob is in one of CS100 and CS101. Tom says that he has taken CS101 and CS102, and Bob is in one of them.

Note that here, Sara's statement (viewed as source S_1) implies that Bob has not taken CS102. The possible worlds D of relation registration(student, course) are shown in Tables 2.1 and 2.2.

registration (student, course)
(Bob, CS100)

D_1

registration(student, course)
(Bob, CS101)

D_2

Table 2.1: Possible worlds of source S_1 (Sara)

We use PW to represent the possible worlds set:

$PW(S_1) = \{D_1, D_2\}$

$PW(S_2) = \{D_3, D_4\}$

registration(student, course)
(Bob, CS101)

D_3

registration(student, course)
(Bob, CS102)

D_4

Table 2.2: Possible worlds of source S_2 (Tom)

The tuple sets T of relation registration in S_1 and S_2 are:

$T(S_1)$ = {(Bob, CS100), (Bob, CS101), (Bob, CS102)}

$T(S_2)$ = {(Bob, CS101), (Bob, CS102)}

Data integration in the standard case typically uses the *open world* assumption [9]: absence of information is interpreted as unknown, instead of being false. An unknown information can be true or false. This assumption is important, which makes data integration an extensible process for each source considered to be open to new information obtained through integration with other sources. On the other hand, based on the above definition of uncertain database in [19], a *close world* assumption is applied for each source: in each possible world, tuples from the tuple set that are not present are considered to be false. This assumption along with the tuple set increases the expressive power of the model in the sense that negative information can be captured.

Even though the possible worlds model is intuitive and simple, this representation is not practical as the number of possible worlds is exponential in the number of tuples in the database. It is thus essential to consider compact and succinct representations

for uncertain relations.

Compact Models

A probabilistic relation (pr-relation) [5] is an annotated relation, whose schema contains an additional attribute E, called the *annotation* attribute. The annotation of a tuple is a propositional logic expression over boolean *event variables*. Pr-relation follows possible worlds semantics. Truth assignment is used to covert a pr-relation to a set of possible worlds. Every possible truth assignment to set V of the event variables defines a possible world of the relation. Tuples whose associated logic expression is true under this truth assignment, are in the possible world. Every possible world can be defined by one or more truth assignments. Pr-relation is a *complete* [21] model, that is, a pr-relation represents a set of possible worlds, and any finite set of possible worlds can be represented as a pr-relation. Having this property, pr-relations can be used as an alternative compact representation of the possible worlds.

When there is no probability associated to the event variables, this relation is called a *pure probabilistic relation* (ppr-relation). The annotation attribute is used only to record dependencies among tuples. We use R_i to represent ppr-relation R in source S_i. Pr-relation is based on probabilistic databases defined in [8], in which the annotations are probability values. When there is probability distribution defined over the set of event variables, the logical expression associated with each tuple can

also be correlated to a probability value of that tuple. Later on in the chapter, we will show how this could be done. We use R_i^p to represent a pr-relation R in source S_i. Here we represent the uncertain data *registration* in the previous example as ppr-relation $registration_i$ of source S_i shown in Tables 2.3 and 2.4. Since according to Sara, Bob did not take CS102, the propositional expression of this tuple in Table 2.3 is false.

$registration_1$ (student, course, E)
(Bob, CS100, e_1)
(Bob, CS101, $\neg e_1$)
(Bob, CS102, false)

Table 2.3: Ppr-relation of source S_1 (Sara)

$registration_2$ (student, course, E)
(Bob, CS101, e_2)
(Bob, CS102, $\neg e_2$)

Table 2.4: Ppr-Relation of source S_2 (Tom)

2.2 Uncertain Data Integration Frameworks

Semantics and Assumptions

The semantics of data integration is called *superset-containment integration* [4]. This semantics applies to the following scenario: we collect data describing the real world

entities from different sources, and combine these information to get a single logical view of the "real-world" entity as accurate as possible. Even though the "real-world" view is still uncertain, it contains more information than any individual contributing source. The integrated uncertain database U should superset-contain the data sources S_i in the sense that:

$T(U) \supseteq T(S_i)$ and $PW(S_i) \supseteq \{D \cap T(U) \mid D \in PW(S_i)\}$

Intuitively, every possible world of the integrated data should be contributed by a possible world from every data source. Each possible world in the result is computed by looking at all data sources together. But as some possible world may not coexist with some possible world from other sources, not all possible worlds in each source will contribute to the integrated result. This semantics is useful when integration is used to resolve partial uncertainty to obtain more information about the real-world entities. Examples include integration of extracted structured data from unstructured content on the web and integration of weather forecasts data to do predictions.

A number of research on traditional data integration assume that data sources are independent. However, in uncertain data integration, as data sources are usually describing the same real-world scenario, dependency maybe introduced by common tuples that represent the same world entities stored in different sources. This is an assumption made in related literature.

In the following section, we review data integration approaches. The first approach uses possible worlds model as the representation of the input uncertain data. We illustrate integration of uncertain data with and without probabilities in this approach. The second integration approach uses a compact model to represent the input uncertain data. To be more precise, it uses ppr-relations for uncertain data and pr-relations for probabilistic data.

Integrating Possible Worlds with Logical Representations

As mentioned above, in order to integrate the possible worlds of uncertain data sources, dependency introduced by common tuples should be captured and maintained. The work proposed in [19] uses logical expressions to capture the relationships among the tuples. We review this proposal and illustrate the integration of uncertain data and probabilistic data sources.

Given an uncertain data source S, we assign to each tuple t_i in the tuple set T(S) a propositional variable x_i. The formula $f(D_j)$ of a possible world D_j is then defined as the conjunction of all variables x_i that are in D_j, and the conjunction of $\neg x_k$ if the corresponding tuple t_k is not in D_j. That is, $f(D_j) = \bigwedge_{t_i \in D_j} x_i \bigwedge_{t_k \notin D_j} \neg x_k$. The formula $f(S_i)$ of a data source S_i is the disjunction of formula $f(D_j)$s representing the possible worlds in the data source. That is, $f(S_i) = \bigvee_{D_j \in PW(S_i)} f(D_j)$. Given a set of uncertain sources $S = \{S_1, ..., S_n\}$ represented by such formula, the integration result S_I is the

conjunction of the logical expressions of all the sources. That is, $f(S_I) = \bigwedge_{S_i \in S} f(S_i)$. The tuple set of integration result T is the union of the tuple set of all sources, i.e. $T(S_I) = \bigcup_{S_i \in S} T(S_i)$

Uncertain Data Sources without Probability Distribution

We follows the example in [19] to illustrate the proposed integration process over uncertain data sources using logical expression.

Example 2.1. Consider the possible worlds of two data sources shown in Tables 2.5 and 2.6.

registration (student, course)
(Bob, CS100)

D_1

registration (student, course)
(Bob, CS101)

D_3

registration (student, course)
(Bob, CS100)
(Bob, CS101)

D_2

Table 2.5: Possible worlds of source S_1

T(S_1) = {(Bob, CS100), (Bob, CS101)}

T(S_2) = {(Bob, CS100), (Bob, CS201), (Bob, CS202)}

We assign the variable x_1 to (Bob, CS100), and x_2 to (Bob, CS101). Then the formula for the possible worlds in S_1 would be:

$$f(D_1) = x_1 \wedge \neg x_2,\ f(D_2) = x_1 \wedge x_2,\ f(D_3) = \neg x_1 \wedge x_2, \text{ and}$$

registration (student, course)
(Bob, CS100)

D'_1

registration (student, course)
(Bob, CS100)
(Bob, CS201)

D'_2

registration (student, course)
(Bob, CS201)

D'_3

registration (student, course)
(Bob, CS201)
(Bob, CS202)

D'_4

Table 2.6: Possible worlds of source S_2

$$f(S_1) = (x_1 \wedge \neg x_2) \vee (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_2).$$

Similarly, if we assign x_3 to (Bob, CS201) and x_4 to (Bob, CS202), we get the following formulas:

$$f(D'_1) = x_1 \wedge \neg x_3 \wedge \neg x_4, f(D'_2) = x_1 \wedge x_3 \wedge \neg x_4,$$

$$f(D'_3) = \neg x_1 \wedge x_3 \wedge \neg x_4, f(D'_4) = \neg x_1 \wedge x_3 \wedge x_4, \text{ and}$$

$$f(S_2) = (x_1 \wedge \neg x_3 \wedge \neg x_4) \vee (x_1 \wedge x_3 \wedge \neg x_4) \vee (\neg x_1 \wedge x_3 \wedge \neg x_4) \vee (\neg x_1 \wedge x_3 \wedge x_4).$$

The integration result f is the conjunction of $f(S_1)$ and $f(S_2)$, i.e., $f = f(S_1) \wedge f(S_2)$, which is simplified as follows:

$f = (x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4) \vee (x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4) \vee (x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4) \vee (x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4) \vee (\neg x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4) \vee (\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4)$

This gives six possible worlds as the integration result, shown in Table 2.7.

As can be seen, some possible worlds from different sources are not compatible, and hence cannot be integrated to produce a possible world in the result. Compatible

Registration (student, course)
(Bob, CS100)

(D_1, D'_1)

registration (student, course)
(Bob, CS100)
(Bob, CS201)

(D_1, D'_2)

registration (student, course)
(Bob, CS100)
(Bob, CS101)

(D_2, D'_1)

registration (student, course)
(Bob, CS100)
(Bob, CS101)
(Bob, CS201)

(D_2, D'_2)

registration (student, course)
(Bob, CS101)
(Bob, CS201)

(D_3, D'_3)

registration (student, course)
(Bob, CS101)
(Bob, CS201)
(Bob, CS202)

(D_3, D'_4)

Table 2.7: Possible worlds of integration result

possible worlds [20] are defined as follows:

Definition 2.2. Compatible Possible Worlds: Let S and S' be data sources containing probabilistic uncertain information $\{D_1, ..., D_m\}$ and $\{D'_1, ..., D'_{m'}\}$, respectively. Let T(S) and $T(S')$ be the tuple sets of S and S'. A pair of possible worlds (D_i, D'_j) from S and S' is said to be compatible if (1) for every tuple $t \in D_i - D'_j$, we have that $t \notin T(S')$, and (2) for every tuple $t \in D'_j - D_i$, we have that $t \notin T(S)$

Intuitively, suppose two data sources S_1 and S_2 have a common tuple t, and t is true in D_1 from S_1, while t is not in the possible world D_2 of source S_2, then D_1 and D_2 are not compatible because they are inconsistent with respect to tuple t.

This approach is more efficient than a pair-wise union computation for integration. The integration process is defined formally by resolving logical expressions, instead of manipulating enumerated possible worlds directly.

Probabilistic Data Sources

When the input data sources are probabilistic, it is represented as a set of possible worlds with a probability distribution over the possible worlds. The probability $P(D_i)$ of D_i is a value in the range [0, 1], and if PW(S) = $\{D_1, D_2, ..., D_n\}$, then $\sum_{i=1}^{n} P(D_i) = 1$ [19]. The possible worlds in integration result can be represented by the logical approach above. What is left to do is to calculate probability distribution over the possible worlds of the integration result. Previous work [19] uses a bipartite graph G to represent the integration of two data sources, and identifies *probabilistic constraints* between possible worlds of the two sources. This graph is also used to calculate probabilities.

Let S and S' be probabilistic data sources containing the possible worlds $D_1, ..., D_m$ and $D'_1, ..., D'_n$ respectively. Graph G contains m nodes for S on one side and n nodes for S' on the other side, corresponding to the possible worlds of the two sources. There is an edge connecting node N_i and N'_j if the formula of the two possible worlds $f(D_i) \wedge f(D'_j) = true$, meaning that the two possible worlds from different sources are compatible. The following theorems about the probabilistic constraints represented in

the graph are from [19]. The first theorem is about the probability constraints of the integration approach, which has to be satisfied before generating correct integrated data. The other theorem is about the properties of the integration process that help to form the graph and compute the probabilities.

Theorem 2.3. *Let G_i be a connected component of G. Let N and N' denote the nodes of G_i corresponding to the possible worlds of sources S and S', respectively. Then*

$$\sum_{D_i \in N} P(D_i) = \sum_{D'_j \in N'} P(D'_j)$$

In a connected component G_i of graph G, the sum of probabilities of the possible worlds from source S should be equal to the sum of probabilities of the possible worlds from source S'. Violation of this constraint requires a probability redistribution. We will elaborate on this topic later.

Theorem 2.4. *Let G be the compatibility graph of sources S and S'. Each connected component of G is a complete bipartite graph.*

If G contains two connected components G_1 and G_2, every node in G_1 from source S will connect to every node in G_1 from source S', but it will not connect to any node in G_2. The same rules apply for G_2. In other words, if two possible worlds in the same data source have common compatible possible worlds from the other source, their compatible possible worlds sets are the same. This theorem helps to generate

the graph more efficiently.

Theorem 2.5. *If a node N_i has no edges connected to it, then $P(D_i) = 0$;*

If a node N_i is not connected to any node from another source, it means that N_i is not compatible with any other possible worlds, and hence will not contribute to the integration result. This is used in probability calculation of the integrated data.

To calculate probability distribution of the integration result, we use conditional probability. Given a possible world D_i from source S_1, and D_j from source S_2, the probability of integrated possible world $\mathrm{P}(D_{ij}) = \mathrm{P}(D_i) * \mathrm{P}(D_j|D_i) = \mathrm{P}(D_j) * \mathrm{P}(D_i|D_j)$. When D_i and D_j are incompatible, $\mathrm{P}(D_j|D_i) = \mathrm{P}(D_i|D_j) = 0$. If D_i and D_j only connect to each other, $\mathrm{P}(D_j|D_i) = \mathrm{P}(D_i|D_j) = 1$. Otherwise the probability is distributed over all possible combinations. For instance, if D_i connects to D_j and D_k, then $\mathrm{P}(D_j|D_i) = \frac{P(D_j)}{P(D_j) + P(D_k)}$ and $\mathrm{P}(D_k|D_i) = \frac{P(D_k)}{P(D_j) + P(D_k)}$.

Example 2.6. Continue with Example 2.2 in previous section, suppose the probability distribution is as follows: $\mathrm{PW}(D_1) = 0.2$, $\mathrm{PW}(D_2) = 0.4$, $\mathrm{PW}(D_3) = 0.4$, $\mathrm{PW}(D'_1) = 0.1$, $\mathrm{PW}(D'_2) = 0.5$, $\mathrm{PW}(D'_3) = 0.2$, $\mathrm{PW}(D'_4) = 0.2$.

Figure 2.1 shows the bipartite graph of Example 2.1.

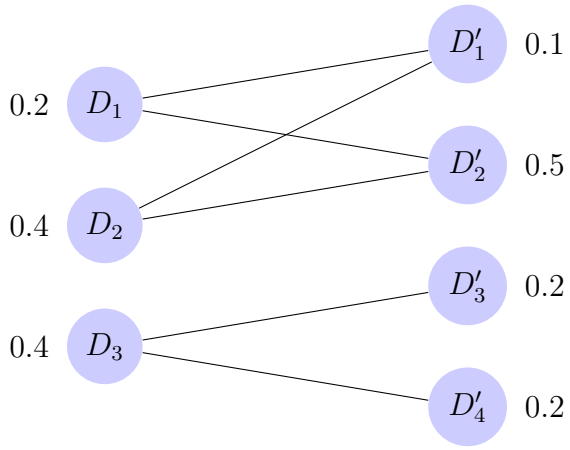


Figure 2.1: Consistency graph of the integration of S_1 and S_2

Note that $P(D_1) + P(D_2) = P(D'_1) + P(D'_2)$ and $P(D_3) = P(D'_3) + P(D'_4)$. The consistency constraints are satisfied, then the integration result has a probability distribution as shown below:

$$P(D_1 \wedge D'_1) = P(D_1) * \frac{P(D'_1)}{P(D'_1) + P(D'_2)} = 0.03$$

$$P(D_1 \wedge D'_2) = P(D_1) * \frac{P(D'_2)}{P(D'_1) + P(D'_2)} = 0.17$$

$$P(D_2 \wedge D'_1) = P(D_2) * \frac{P(D'_1)}{P(D'_1) + P(D'_2)} = 0.07$$

$$P(D_2 \wedge D'_2) = P(D_2) * \frac{P(D'_2)}{P(D'_1) + P(D'_2)} = 0.33$$

$$P(D_3 \wedge D'_3) = P(D_3) * \frac{P(D'_3)}{P(D'_3) + P(D'_4)} = 0.2$$

$$P(D_3 \wedge D'_4) = P(D_3) * \frac{P(D'_4)}{P(D'_3) + P(D'_4)} = 0.2$$

Given this integration result, we could further interpret it based on the application needs. For instance, if we only consider the most possible world, then $P(D_2 \wedge D_2')$ has the highest probability, so it is most likely that Bob takes all the three courses CS100, CS101, and CS201.

Integrating Uncertain Relations in Compact Model

While logical representation model reduces the number of union operations across sources, it is not yet efficient because of dealing with the possible worlds. The goal of [5] is to integrate uncertain data sources based on the possible world semantics but represented in a compact form. The data sources in their proposal are represented as pr-relations, defined in Section 2.1. Since the inputs are uncertain or probabilistic data sources, they are first converted into the compact model, done as follows. If the input source is uncertain data without probabilities, the conversion still applies, but without the step for probability calculation, and event variables become pure boolean variables with no associated probabilities.

Let S be an uncertain data source with the tuple set $T(S)$ and the set of possible worlds $PW(S) = \{D_1, ..., D_n\}$. Let $P(D_i)$ be the probability of the possible world D_i. Then $\sum_{i=1}^{n} P(D_i) = 1$. Also let $\mathrm{P}(x_i)$ be the probability of the event variable x_i. We need $N - 1$ event variables to represent N possible worlds. Consider the set of 2^{n-1} interpretations of the event variables (truth assignment). They use the given

probabilities of the possible worlds as the probabilities of the event variables.

- Assign D_1 with all the truth assignments, in which e_1 is true. Then the logical expression of D_1 is e_1.
- Assign D_2 with all the truth assignments, in which e_1 is false and e_2 is true. The expression of D_2 is then $\neg e_1 \wedge e_2$.
- Continue with the truth assignments. The logical expression for the possible world D_i is then $\neg e_1 \wedge ... \wedge \neg e_{i-1} \wedge e_i$.
- The last possible world D_n is assigned the truth assignment, in which all the event variables are false. Thus, the logical expression for D_n is $\neg e_1 \wedge ... \wedge \neg e_{i-1} \wedge \neg e_{n-1}$.

Given the probability of each possible world, the probability of the event attribute E for a tuple $t \in r$ is obtained as follows.

- If t is not in any possible world, then the value of E attribute is false.
- Otherwise, this value is $f(t) = \bigvee\{(f(D_i) \mid t \in D_i)\}$.

The probability of the event variables are:

$$P(e_1) = p_1 = d_1$$

$$P(e_2) = p_2 = d_2/(1 - d_1)$$

$$P(e_i) = p_i = d_i/(1 - d_1 - d_2 - ... - d_{i-1})$$

These truth assignments are then used to convert the compact pr-relations back to possible worlds, so that the representations of the sources are consistent for both input and output. A value true of an attribute E associated with a tuple t indicates that t exists in this possible world under the current truth assignment. Each truth assignment is equivalent to one possible world, and each possible world is mapped by one or multiple truth assignments. With these two conversion algorithms, we are able to transform data between the possible worlds model and pr-relations.

Uncertain Data Sources without Probabilities

During the integration process, dependencies between the input data sources need to be captured. These dependencies are considered constraints in the integrated data. The work in [5] defines the notion of extended pr-relations to represent the integration result along with the dependencies, as follows.

Definition 2.7. Extended Probabilistic Relations: An extended probabilistic relation (epr-relation) is a pr-relation together with a set of event constraints.

The importance of event constraints is that they eliminate invalid truth assignments. A *valid* assignment of truth values to event variables will satisfy all the event

constraints. Only valid truth assignments will be considered to generate possible worlds. The integration procedure is defined as follows, which is also applicable to ppr-relations.

Let r_1 and r_2 be the two pr-relations in data sources S_1 and S_2, respectively, and epr_I be the integrated epr-relation. The tuple set of epr_I is the union of tuple sets of r_1 and r_2. For each tuple t in $\mathrm{T}(epr_I)$ which appears only in either r_1 or r_2, copy the corresponding E value of t from that source. For each tuple t that appear in both r_1 and r_2, copy the corresponding E value from either r_1 or r_2, but keep a global event constraint to epr_I: $w_1 \equiv w_2$, where w_1 is the value of the E attribute for t in r_1 and w_2 is the value of the E attribute for that tuple in r_2. The two values are equivalent, meaning that the common tuple t is either not in the current possible worlds of any of the two data sources, or it appears in both. We use $w_1 \equiv w_2$ to represent the equivalence of w_1 and w_2, that is $(w_1 \rightarrow w_2) \wedge (w_2 \rightarrow w_1)$

To illustrate this, let us consider integration of the data sources in Example 2.1. First, we apply the conversion algorithm. The pr-relations for the possible worlds of sources in the example are shown in Figure 2.8 and 2.9, in which some complex expressions are simplified.

student	course	E
Bob	CS100	$x_1 \vee (\neg x_1 \wedge x_2)$
Bob	CS101	$\neg x_1$

Table 2.8: The pr-relation of source S_1

student	course	E
Bob	CS100	$x'_1 \vee (\neg x'_1 \wedge x'_2)$
Bob	CS201	$\neg x'_1$
Bob	CS202	$\neg x'_1 \wedge \neg x'_2 \wedge \neg x'_3$

Table 2.9: The pr-relation of source S_2

Following the integration procedure, Table 2.10 shows the integrated result represented in epr-relation.

student	course	E
Bob	CS100	$x_1 \vee (\neg x_1 \wedge x_2)$
Bob	CS101	$\neg x_1$
Bob	CS201	$\neg x'_1$
Bob	CS202	$\neg x'_1 \wedge \neg x'_2 \wedge \neg x'_3$
$x_1 \vee (\neg x_1 \wedge x_2) \equiv x'_1 \vee (\neg x'_1 \wedge x'_2)$		

Table 2.10: The epr-relation of the integration

According to the global event constraints, the truth assignments for (x_1, x_2, x'_1, x'_2) are shown in Table 2.11. in which the invalid ones are crossed out. Expanding the epr-relation based on the valid truth assignments will produce the same set of possible worlds obtained using the logical representation approach.

Probabilistic Data Sources

The work in [22] introduced *Event Variable Formula (EVF)* to calculate probability distribution of integrated possible worlds based on pr-relation integration. It benefits from the probability calculation proposed for pr-relations.

x_1	x_2	x_1'	x_2'	x_3'
0	0	0	0	0
0	0	0	0	1
~~0~~	~~0~~	~~0~~	~~1~~	~~0~~
~~0~~	~~0~~	~~0~~	~~1~~	~~1~~
~~0~~	~~0~~	~~1~~	~~0~~	~~0~~
~~0~~	~~0~~	~~1~~	~~0~~	~~1~~
~~0~~	~~0~~	~~1~~	~~1~~	~~0~~
~~0~~	~~0~~	~~1~~	~~1~~	~~1~~
~~0~~	~~1~~	~~0~~	~~0~~	~~0~~
~~0~~	~~1~~	~~0~~	~~0~~	~~1~~
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1
~~1~~	~~0~~	~~0~~	~~0~~	~~0~~
~~1~~	~~0~~	~~0~~	~~0~~	~~1~~
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
~~1~~	~~1~~	~~0~~	~~0~~	~~0~~
~~1~~	~~1~~	~~0~~	~~0~~	~~1~~
1	1	0	1	0
1	1	0	1	1
1	1	1	0	0
1	1	1	0	1
1	1	1	1	0
1	1	1	1	1

Table 2.11: Truth assignment table for integration result

Definition 2.8. An Event Variable Formula (EVF) is a logical formula obtained by applying Boolean operations over multiple event variables. Let r be the pr-relation

on the schema $R \cup \{E\}$ and T(r) $= \{t_1, ...t_n\}$ be the tuple set of r. The value of the E attribute associated with t_i is f_i, $1 \leqslant i \leqslant n$. The formula corresponding to a possible world D is represented by the formula of the tuples it includes, that is: $f_e(D) = \wedge_{t_i \in D} f_i \wedge_{t_j \notin D} f_j$.

The event variables in the same data source are assured to be independent. The event variables from different sources may not be independent, because of the global event constraints. If these variables are not independent, conditional probability is used to compute the probability.

We illustrate this using the same probabilistic data sources of Example 2.6 with the probability distribution defined earlier in Section 2.2. The first step is to convert this data sources to pr-relations using the conversion algorithm. The resulting pr-relations are the same as shown in Table 2.8 and 2.9. We now calculate the probability of each event variable as follows:

$$\text{P}(x_1) = d_1 = 0.2,\ \text{P}(x_2) = \frac{d_2}{1 - d_1} = 0.5,\ \text{P}(x'_1) = d'_1 = 0.1,\ \text{P}(x'_2) = d'_2/(1 - d'_1)$$
$$= 0.5/0.9 = 0.56,\ \text{P}(x'_3) = d'_3/(1 - d'_1 - d'_2) = 0.5$$

Next, we integrate pr-relations based on algorithm explained in Section 2.2. This produces the same epr-relations shown in Table 2.10. The epr-relation represents the same set of possible worlds shown in Table 2.7.

Finally, we generate the EVF for each possible world in the integration result and compute its probability using the EVFs. For instance, the possible world of

integration of D_2 and D'_1 consists of two tuples: t_1 = (Bob, CS100) and t_2 =(Bob, CS101). The tuple set also include the tuples t_3 = (Bob, CS201) and t_4 =(Bob, CS202). The EVF formula of this integrated possible world is $EVF(D_2, D'_1) = f_1 \wedge f_2 \wedge \neg f_3 \wedge \neg \wedge f_4 = (x_1 \vee (\neg x_1 \wedge x_2)) \wedge \neg x_1 \wedge x'_1 \wedge \neg(\neg x'_1 \wedge \neg x'_2 \wedge \neg x'_3) =$, which would be further simplified to obtain $P(EVF) = (\neg x_1 \wedge x_2) \wedge x'_1$. According to the global constraint, $\mathrm{P}(x'_1|(\neg x_1 \wedge x_2)) = P(x'_1)/(P(x'_1) + P(\neg x'_1 \wedge x'_2)) = 1/6$. So $P(EVF) = 0.8 * 0.5 * 1/6 = 0.07$. Probability calculation for the rest of the possible worlds in the integration result follows the same steps. This result is consistent with the probabilities calculated in the integration using the possible worlds.

Motivation

Even though data integration is a popular and widely studied topic over the past two decades, not much work has done on the integration of uncertain data, especially the uncertain data that are not independent. Existing work takes independent assumption that tuples are independent from each other. Integrating uncertain data with dependencies is an interesting topic that we would like to explore.

We present two proposed approaches to integrate both uncertain data and probabilistic data in this chapter. Since possible worlds model is the semantics of uncertain data, the proposed uncertain data integration approach in possible worlds model helps

us understand the integration result. However, due to the exponential size of the possible worlds, it is not an efficient and practical approach to present and integrate uncertain data.

Previous work of probabilistic data integration is limited to two sources [19][5][22]. It provides a basis to further study and explore the problem of integration for more than two data sources. We begin by introducing the properties of the integration process, using which we then define an algorithm for integration of multiple data sources. One of the challenges for integrating probabilistic data represented by the possible worlds model is to extend the bipartite graph proposed in [19] in Figure 2.1. The bipartite graph works for two data sources only. For multiple data sources of more than two, the probability constraints change with the order in which the sources are integrated. The probability calculation for the integrated data needs to be extended as well. When considering the compact approach of integration process for more than two sources, the equivalence has to be re-establish for both of our extended integration solutions. Besides, the compact approach does not have probability consistency defined.

As we can see, the probability constraints may not be satisfied during the integration process. In fact, as the number of data sources increases, the closer and more similar the data sources are, the stricter and more complicated the probability constrains will be. However, the probability adjustment techniques proposed in [19]

works only for two data sources. A more general adjustment algorithm should be devised along with the extended integration frameworks.

Chapter 3

Integrating Multiple Uncertain Data Sources

In this chapter, we introduce the semantics and procedure of integrating multiple uncertain data sources.

We start with introducing the desired properties of the integration operations reviewed in Chapter 2. According to our findings, we use these properties as the basis for integration of more than two sources. We demonstrated the limitations and issues with existing integration operations, and propose an extended integration approach for integrating multiple sources. Our approach includes a generalized conversion algorithm of uncertain data from possible worlds model to compact probabilistic relations, the integration procedure, the definition of probability constraints, and a

probability adjustment method for violated constraints.

3.1 Properties of Integration Operations

Recall the integration process for two sources studied in existing work [19], [5] in Chapter 2. We use I_{pw} to denote the integration of uncertain data in possible worlds model without probabilities, and use I_{pw}^{P} for when probabilities are present. Integration of uncertain data in the compact form (epr-relation) without probabilities is denoted by I_{epr}, and correspondingly, I_{epr}^{P} denotes the integration when the sources are probabilistic data. Each data source is represented as a set of possible worlds.

We study the idempotent, commutativity, and associativity of integration operations. These properties form a basis for extending the integration of two sources to multiple-source integration. We start with integration of pure uncertain data sources without probability (I_{pw} and I_{epr}).

Theorem 3.1. *I_{pw} is idempotent, commutative and associative.*

Proof. The proofs are based on the logical representation of the sources and the integration operation. Suppose $f(S)$ is the logical expression representing the set of possible worlds of source S. From the literature we know that for two sources S_1 and S_2, $f(I_{pw}(S_1, S_2)) = f(S_1) \wedge f(S_2)$. Let S be a source that $S = \{D_1, D_2, ..., D_m\}$, the logical expression $f(S) = \bigvee_{D_i \in S} f(D_i)$. If $f(S_1) \equiv f(S_2)$, meaning S_1 and S_2 contain

the same set of possible worlds, then $S_1 = S_2$.

- Idempotent: For any uncertain data source S, I_{pw} is idempotent if $I_{pw}(S, S) = S$.

  Suppose two sources $S_1 = S_1'$. We have that $f(S_1) = f(S_1')$. The logical expression of the integration result is as follows:

  $f(I_{pw}(S_1, S_1')) = f(S_1) \wedge f(S_1') = f(S_1)$. That is, $I_{pw}(S_1, S_1') = S_1$. The property holds.

- Commutativity: Given uncertain data sources S_1, S_2, if $I_{pw}(S_1, S_2) = I_{pw}(S_2, S_1)$, this property holds.

  The integration result represented by the logical expression is $f(I_{pw}(S_1, S_2)) = f(S_1) \wedge f(S_2)$. Because logical conjunction is commutative, thus $f(I_{pw}(S_1, S_2)) = f(S_1) \wedge f(S_2) = f(S_2) \wedge f(S_1) = f(I_{pw}(S_2, S_1))$.

- Associativity: Let S_1, S_2 and S_3 be the data sources. If $f(I_{pw}(I_{pw}(S_1, S_2), S_3)) = f(I_{pw}(S_1, I_{pw}(S_2, S_3)))$, I_{pw} is associative. To be consistent with the notions, we use $f^-(f(S))$ to represent the conversion from logical expressions $f(S)$ to a set of possible worlds of S.

  If we integrate S_1 and S_2, then integrate S_3 with the first result:

  $f(I_{pw}(I_{pw}(S_1, S_2), S_3)) = f(I_{pw}(f^-(f(S_1) \wedge f(S_2))), S_3)) = (f(S_1) \wedge f(S_2)) \wedge f(S_3)$

If S_2 and S_3 are integrated first, then integrate with S_1:

$f(I_{pw}(I_{pw}(S_2, S_3), S_1)) = f(I_{pw}(f^{-}(f(S_2) \wedge f(S_3))), S_1)) = (f(S_2) \wedge f(S_3)) \wedge f(S_1)$.

Since logical conjunction is associative and commutative,

$(f(S_1) \wedge f(S_2)) \wedge f(S_3) = f(S_1) \wedge ((f(S_2) \wedge f(S_3))$. Therefore,

$f(I_{pw}(I_{pw}(S_1, S_2), S_3)) = f(I_{pw}(S_1, I_{pw}(S_2, S_3)))$. I_{pw} is associative.

□

Theorem 3.2. *I_{epr} is idempotent, commutative and associative.*

Proof. Let S_1, S_2 be two uncertain data sources. Their pr-relation representations are ppr_1 and ppr_2.

- Idempotent: The event attribute value of a tuple t in ppr_1 is $f_e(t)$. Recall the integration process in Chapter 2, each distinct tuple in the input sources are copied into the table along with its even attribute value. Common tuples would add a global constraint to the result. For every tuple t in the integration result of $I_{epr}(S_1, S_1)$, there is a global constraint $f_e(t) \equiv f_e(t)$. We observe that since all these constraints are always satisfied, they can be eliminated. The integration result is simplified to a ppr-relation ppr_I. $T(ppr_I) = T(S_1)$ and $f_e(t) = f_e(t')$ for every t in ppr_I and corresponding t' in S_1 where $t = t'$. The event variable set $V_I = V_{S_1}$. Thus $ppr_I = S_1$. $PW(ppr_I) = PW(S_1)$, the property is satisfied.

- Commutativity: Suppose the common tuple set of S_1 and S_2 is T_c. $I_{epr}(S_1, S_2) = ppr_{I1}$, $I_{epr}(S_2, S_1) = ppr_{I2}$. It is obvious that $T(ppr_{I1}) = T(ppr_{I2}) = T(S_1) \cup T(S_2)$. The event variable set $V_{I1} = V_{V_{I2}}$. For every common tuple $t \in T_c$, the value of event attribute in ppr_{I1} is $f_e(t_i)$, with a global constraint $f_e() \equiv f'_e(t)$. The event attribute value in ppr_{I2} is $f'_e(t)$, with a global constraint $f'_e(t) \equiv f_e(t)$. Tuples that are unique in either sources will not bring global constraints, and their event attribute values are the same in ppr_{I1} and ppr_{I2}. Thus, ppr_{I1} and ppr_{I2} are equivalent, they represent the same set of possible worlds. $I_{epr}(S_1, S_2) = I_{epr}(S_1, S_2)$.

- Associativity: Given S_1, S_2 and S_3 as uncertain data sources, their representation as ppr-relations are ppr_1, ppr_2 and ppr_3. If the integration order is S_1, S_2, and then S_3, the integration result $ppr_{I1} = I_{epr}(I_{epr}(S_1, S_2), S_3)$. If S_2 and S_3 are integrated first, then $ppr_{I2} = I_{epr}(I_{epr}(S_2, S_3), S_1)$. According to the integration process, we have that $T(ppr_{I1}) = T(ppr_{I2})$, and $V_{I1} = V_{V_{I2}}$. There are three types of tuples in the integration result: tuples that exists in every data source, tuples that exists in two or more sources, but not all the sources, and tuples that exists only in one source. We show the associativity of integration using a simple example as data sources in Table 3.1. In the three sources, t_1 is a common tuple of S_1, S_2 and S_3. t_2 is a common tuple of S_1 and S_2 while t_3 is a common tuple of S_1 and S_3. The other tuples are unique among all the

T	E
t_1	$f_e(t_1)$
t_2	$f_e(t_2)$
t_3	$f_e(t_3)$
t_4	$f_e(t_4)$

$ppr_1(S_1)$

T	E
t_1	$f'_e(t_1)$
t_2	$f'_e(t_2)$
t_5	$f_e(t_5)$

$ppr_2(S_2)$

T	E
t_1	$f''_e(t_1)$
t_3	$f'_e(t_3)$
t_6	$f_e(t_6)$

$ppr_3(S_3)$

Table 3.1: Uncertain Data Sources

sources. Table 3.2 shows the integration results ppr_{I1} and ppr_{I2} following different orders. For unique tuples that appear only in one data source, the order

T	E
t_1	$f_e(t_1)$
t_2	$f_e(t_2)$
t_3	$f_e(t_3)$
t_4	$f_e(t_4)$
t_5	$f_e(t_5)$
t_6	$f_e(t_6)$
$f_e(t_1) \equiv f'_e(t_1) \equiv f''_e(t_1))$	
$f_e(t_2) \equiv f'_e(t_2)$	
$f_e(t_3) \equiv f'_e(t_3)$	

ppr_{I1}

T	E
t_1	$f'_e(t_1)$
t_2	$f'_e(t_2)$
t_3	$f'_e(t_3)$
t_4	$f_e(t_4)$
t_5	$f_e(t_5)$
t_6	$f_e(t_6)$
$f'_e(t_1) \equiv f''_e(t_1) \equiv f_e(t_1))$	
$f'_e(t_2) \equiv f_e(t_2)$	
$f'_e(t_3) \equiv f_e(t_3)$	

ppr_{I2}

Table 3.2: Integration of S_1, S_2 and S_3 with different integration orders

of integration does not affect its event value. For any common tuple, however, its event value is assigned by the first source that includes the tuple. On the basis of the global constraints, different integration results obtained by different orders are equivalent, as they represent the same set of possible worlds. That is, $I_{epr}(I_{epr}(S_1, S_2), S_3) = I_{epr}(S_1, I_{epr}(S_2, S_3))$.

□

As explained above, we note that to integrate uncertain data without probability associated, both I_{pw} and I_{epr} are idempotent, commutative and associative operations. The integration result depends only on the set of uncertain data sources, and is not affected by the order in which they are involved in the integration process. Based on these properties, we extend the integration algorithm as follows.

Algorithm 3.1 Integration algorithm for multiple uncertain data sources without probabilities

Let I be the binary integration operation either I_{pw} or I_{epr}, and s_k be a non-empty set of uncertain data sources S_1 to S_k, which are to be integrated. Let $F(s_k)$ be the function to integrate a set of uncertain data sources:

$$F(s_k) = \begin{cases} S_1 & \text{if } k = 1 \\ I(F(s_{k-1}), S_k) & \text{if } k > 1 \end{cases} \tag{1}$$

Now we focus on the properties of integration operation I^P_{pw} for probabilistic data, and present the following result.

Theorem 3.3. *The I^P_{pw} is idempotent and commutative, but not associative.*

Proof. To examine the equivalence of two probabilistic data set, it is necessary to compare both possible worlds set and their probability distribution. Let S_1, S_2 and S_3 be the probabilistic data sources. Given I_{pw} is idempotent, commutative and associative, we only need to compare the probability distribution of the integration results.

- Idempotent: Let S_1 and S_1' be two data sources, $S_1 = S_1'$, $PW(S_1) = \{D_1, D_2, ..., D_m\}$, $PW(S_1') = \{D_1', D_2', ..., D_m'\}$, $D_i = D_i'$. Integrate the two sources and compute probability distribution with the bipartite graph. Each possible world in S_1 can only integrate with the possible world that has the same set of tuples in S_1'. There are m connected components in the bipartite graph, each component contains two nodes connecting to each other. Node N_i connects only to N_i'. $P(D_i) = P(D_i')$, the probability constraints are satisfied. According to the probability calculation algorithm, for every possible world pair D_i, D_i' in the input sources, the probability of integrated possible world $P(D_i \wedge D_i') = P(D_i) = P(D_i')$. Probability distribution is the same as source S_1 and S_1', the property holds.

- Commutativity: Base on the commutativity of I_{pw}, the connected components in the corresponding bipartite graph are not changed by the integration order. Only *consistent* possible worlds are connected in the graph. Within each connected component G, if the probability constraints $\sum_{D \in G \wedge D \in S_1} P(D) = \sum_{D' \in G \wedge D' \in S_2} P(D')$ are not satisfied, the adjustment algorithm reviewed in Chapter 2 will adjust probability of data sources so that the total probability of the connected component is equal to the average of probability on both sides: $P(G) = (\sum_{D \in G \wedge D \in S_1} P(D) + \sum_{D' \in G \wedge D' \in S_2} P(D'))/2$. The adjustment is not affected by integration order. Given D, the conditional probability

$P(D'|D)$ is the percentage of $P(D')$ in all possible worlds that can integrate with D. Such possible worlds are these in the same connected component. Therefore $P(D'|D) = P(D')/\sum_{D_x \in G \wedge D_x \in S_2} P(D_x) = P(D')/P(G)$. Similarly, $P(D|D') = P(D)/\sum_{D_y \in G \wedge D_y \in S_1} P(D_y) = P(D)/P(G)$. We can see that the probability of every integrated possible world can be calculated regardless of integration order: $P(D \wedge D') = P(D) * P(D'|D) = P(D') * P(D|D')$. Thus, $I^P_{pw}(S_1, S_2) = I^P_{pw}(S_2, S_1)$.

- Associativity: We show that I^P_{pw} is not associative using a counter example. That is, different integration order may generate different groups of probability constraints, and that having all the probability constraints satisfied under one integration order does not guarantee probabilistic consistency under another integration order.

  We will then show that if the probability constraints are not satisfied, then probability adjustment should apply. Because the probability adjustment is done locally, in this case, different integration order produces different results, and hence the process is not associative.

  Example 3.4. Let us consider data sources S_1 and S_2 defined in Example 2.1. Now consider a third source S_3, defined as Table 3.3.

  The probability distribution of the possible worlds in S_1, S_2, and S_3 are shown

in Figure 3.1. The bipartite graph in [19] is used to calculate probability distribution of integration result applies to two sources. There are three different orders to integrate S_1, S_2 and S_3: $I^P_{pw}(I^P_{pw}(S_1, S_2), S_3)$, $I^P_{pw}(I^P_{pw}(S_1, S_3), S_2)$ and $I^P_{pw}(I^P_{pw}(S_2, S_3), S_1)$. Note that the constraints checking and probability distribution calculation of the integration result in the examples are calculated by our implementation justified in the next chapter.

Registration (student, course)
(Bob, CS300)

D''_1

Registration (student, course)
(Bob, CS101)

D''_2

Table 3.3: Possible Worlds of source S_3

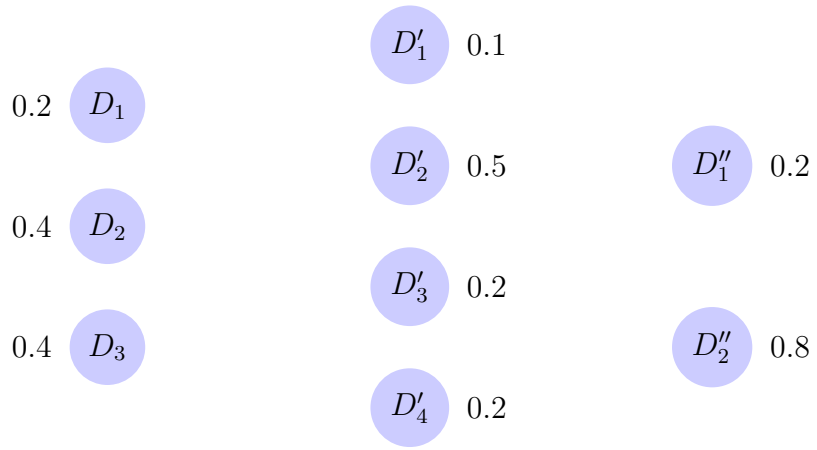


Figure 3.1: Nodes of S_1, S_2 and S_3

Let us consider the integration of S_1, S_2 and S_3 in the order expressed in $I^P_{pw}(I^P_{pw}(S_1, S_2), S_3)$. The consistency graph of the integration of S_1 and S_2

was shown in Figure 2.1. First we check that the probability constraints are satisfied:

$$P(D_1) + P(D_2) = P(D'_1) + P(D'_2) \tag{2}$$

$$P(D_3) = P(D'_3) + P(D'_4) \tag{3}$$

D_{ij} is the integration of the possible worlds D_i from S_1 and D'_j from S_2. Probability distribution of $I^P_{pw}(S_1, S_2)$ is calculated as in Section 2.2. The next step is to integrate the result with the third data source S_3. Figure 3.2 shows the consistency graph of the integration result.

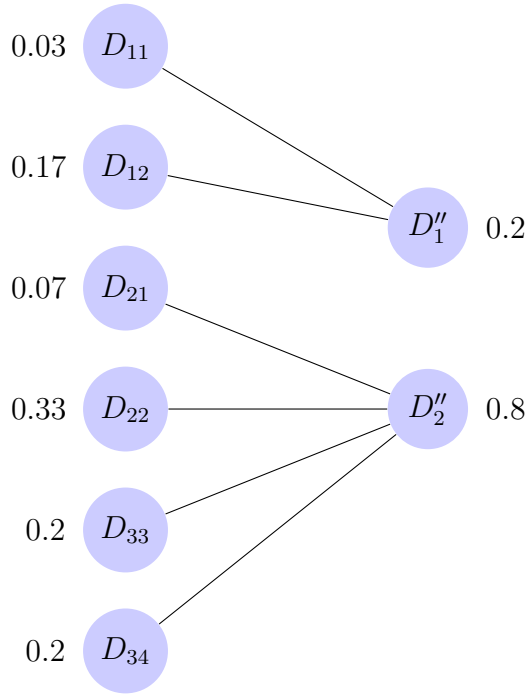


Figure 3.2: Consistency graph of the integration of $I^P_{pw}(S_1, S_2)$ and S_3

A set of probability constraints that must be satisfied in this step includes the following:

$$P(D_{11}) + P(D_{12}) = P(D''_1) \tag{4}$$

$$P(D_{21}) + P(D_{22}) + P(D_{33}) + P(D_{34}) = P(D''_2) \tag{5}$$

Since all constraints (2) to (5) are satisfied, this integration does not need probability adjustment. Thus integration result is as follows, where D_{ijk} is the possible world obtained by integrating D_i, D'_j and D''_k.

$P(D_{111}) = 0.03$; $P(D_{121}) = 0.17$; $P(D_{212}) = 0.07$; $P(D_{222}) = 0.33$; $P(D_{332}) = 0.2$; $P(D_{342}) = 0.2$.

Let us consider a different order $I^P_{pw}(I^P_{pw}(S_1, S_3), S_2)$ for the integration. The consistent graph of the integration of S_1 and S_3 is shown in Figure 3.3.

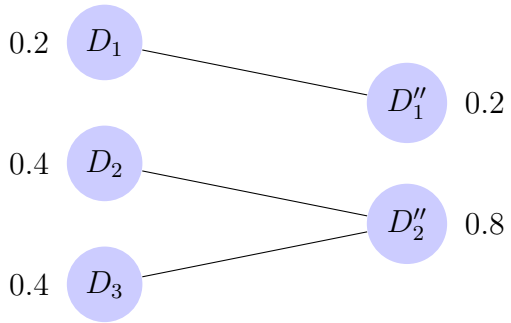


Figure 3.3: Consistency graph for the integration of S_1 and S_3

The probability constraints below, introduced in Figure 3.3, are satisfied:

$$P(D_1) = P(D_1'') \tag{6}$$

$$P(D_2) + P(D_3) = P(D_2'') \tag{7}$$

Figure 3.4 shows the probabilities of integration of S_1 and S_2 in the first step, along with the consistency graph of the second integration.

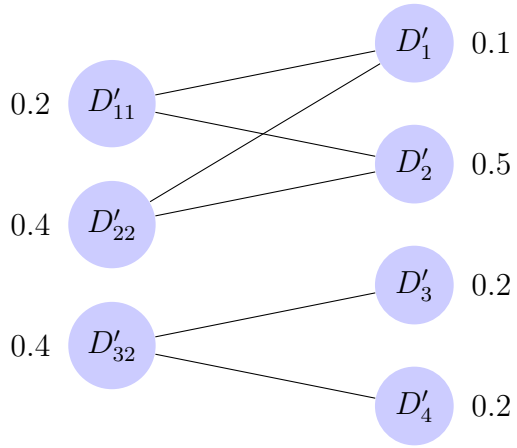


Figure 3.4: Consistency graph of the integration of $I^P_{pw}(S_1, S_3)$ and S_2

The probability constraints are satisfied:

$$P(D'_{11}) + P(D'_{22}) = P(D'_1) + P(D'_2) \tag{8}$$

$$P(D'_{32}) = P(D'_3) + P(D'_4) \tag{9}$$

Thus, the integration result is calculated as follows:

$P(D'_{111}) = 0.03$, $P(D'_{121}) = 0.17$, $P(D'_{212}) = 0.07$, $P(D'_{222}) = 0.33$, $P(D'_{332}) = 0.2$, $P(D'_{342}) = 0.2$.

Finally, let us consider the third order $I^P_{pw}(I^P_{pw}(S_2, S_3), S_1)$ of integrating the three sources S_1, S_2 and S_3. If S_2 and S_3 are integrated first, since they do not have common tuples, the two sources are considered independent, and hence, no probability constraints is introduced. In this case, all pairs of the possible worlds from S_2 and S_3 can integrate. Thus yields eight possible worlds of the integration result: $D''_{11} = 0.02$; $D''_{12} = 0.08$; $D''_{21} = 0.1$; $D''_{22} = 0.4$; $D''_{31} = 0.04$; $D''_{32} = 0.16$; $D''_{41} = 0.04$; $D''_{42} = 0.16$.

If we integrate the result with S_1, we obtain the consistency graph shown in Figure 3.5.

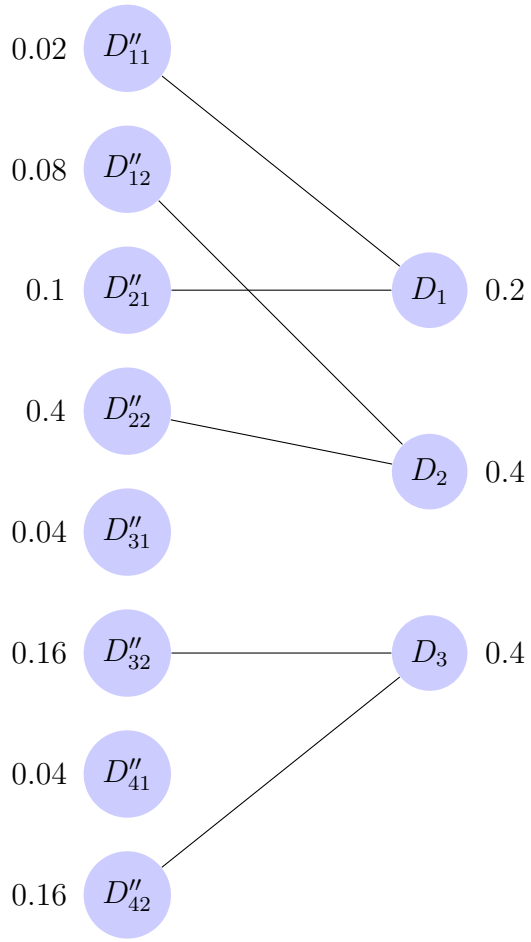


Figure 3.5: Consistency graph of the integration of $I^P_{pw}(S_2, S_3)$ and S_1

According to the graph, we have the following probability constraints:

$$P(D''_{11}) + P(D''_{21}) = P(D_1) \tag{10}$$

$$P(D''_{12}) + P(D''_{22}) = P(D_2) \tag{11}$$

$$P(D''_{32}) + P(D''_{42}) = P(D_3) \tag{12}$$

$$P(D''_{31}) = P(D''_{41}) = 0 \tag{13}$$

However, these constraints are not satisfied. We thus need to adjust the probabilities based on the algorithm proposed in [19]. For this, we first distribute the probability 0.08 from D''_{31} and D''_{41} over the possible worlds on the left side of the graph based on the proportion of existing probability, and then adjust the probabilities of the three connected components. We use D''_{ijk} to represent each possible world of the integration result as follows.

$P(D''_{111}) = 0.03$; $P(D''_{121}) = 0.13$; $P(D''_{212}) = 0.08$; $P(D''_{222}) = 0.38$; $P(D''_{332}) = 0.19$; $P(D''_{342}) = 0.19$.

As can be seen from the above three different orders of performing the integration operation, each order may requires a set of probability constraints to be satisfied. We remark that there is no justified preference among these three orders and different results. Even though $I^P_{pw}(I^P_{pw}(S_1, S_2), S_3)$ and $I^P_{pw}(I^P_{pw}(S_1, S_3), S_2)$ both have their constraints satisfied, and they produce the same probability distribution over the integration result, the constraints are not satisfied in the third order $I^P_{pw}(I^P_{pw}(S_2, S_3), S_1)$. After probability adjustment, the probability distribution of the third integration result is different from the other orders. This shows that the integration operation I^P_{pw} is not associative.

□

Due to the limitation of the bipartite graph, when integrating multiple data sources, only two sets of possible worlds are integrated at each step of the integration process, which produces an intermediate result for the next step. As shown, probability constraints based on the intermediate result obtained at each step does not reflect the general, global dependency among the input data sources. It is difficult to tell the dependency between S_1 and S_2 or between S_1 and S_3 because after each step, information about the input sources are their relationships are not carried over. Besides, enumerating the integration results of all possible orders is unrealistic. This disadvantage prevents probability adjustment to be applied based on the *local* proportion of the possible worlds within data sources.

To avoid these problems, we introduce I^P_{EPRs}, a compact probabilistic data integration procedure for multiple sources. We will then revisit the above example and illustrate how the procedure provides the idea of integration of multiple uncertain data sources in possible world model.

3.2 Extended Uncertain Data Integration in Compact Model

As shown in the previous section, the integration operation I^P_{pw} is not associative and hence not suitable for multiple probabilistic data sources because of the probability

calculation and adjustments involved. In this section, we introduce I^P_{EPRs}, an integration algorithm which extends I^P_{epr}, proposed in [5] for two sources. For this, we first consider a representation of the possible worlds in compact forms (pr-relations) and propose a generalized conversion algorithm by identifying the basic ideas. Following these principles, there can be different algorithms to generate a pr-relation from a set of possible worlds. We define Interpreted Event Variable to track the relationship between event variables and the possible worlds. We also need also to define probability consistency constraints and corresponding probability adjustment algorithm for data sources with inconsistent probabilities. We will show that our integration is idempotent, commutative, and associative. To validate our work and to show its feasibility, we have also implemented the proposed integration process together with routines to check constraints and calculate probabilities.

Generalized Conversion Algorithm

To convert a set of possible worlds into a pr-relation, each possible world is first represented by a conjunction of event variables. Consider each event variable e as an axis that divides the space of the set PW^e_{sum} of possible worlds into two parts: e and $\neg e$. The possible worlds that are on the right hand side of the axis is denoted by PW_e, and those on the other side is denoted by $PW_{\neg e}$. For every possible world D

in PW_e, we have that:

$$f_e(D) = f_e(D) \wedge e \tag{1}$$

Similarly, for every possible world D' in $PW_{\neg e}$, we have:

$$f_e(D') = f_e(D') \wedge \neg e \tag{2}$$

We know that $PW^e_{sum} = PW_e \cup PW_{\neg e}$ and $PW_e \cap PW_{\neg e} = \emptyset$. The probability $P(e)$ of event variable e is defined as:

$$P(e) = \sum_{D \in PW_e} P(D) / \sum_{D' \in PW^e_{sum}} P(D') \tag{3}$$

A set of independent event variable axes divides the space into several parts. Possible worlds take up parts of the graph, and their expression formulas are represented by axis' values covering that part. Each possible world needs to be uniquely represented, thus each part contains one and only one possible world. Therefore, for n possible worlds, $n - 1$ event variables are needed to represent them. Note that PW^{ei}_{sum} is updated each time we add a new event variable e_i. We illustrate the idea with an example, and show it coincides with the conversion algorithm presented in Chapter 2.

Example 3.1. Let $D_1, ..., D_n$ be a set of possible worlds. Since the expression associated with D_1 is $f_e(D_1) = e_1$, D_1 is on the right hand side of the axis e_1 as shown in Figure 3.6 (a). We then have:

$f_e(D_1) = e_1$

$PW^{e1}_{sum} = \{D_1, D_2, ..., D_n\}$

$PW_{e1} = \{D_1\}$

$PW_{\neg e1} = \{D_2, ..., D_n\}$

$P(e_1) = \sum_{D \in PW_{e1}} P(D) \ / \ \sum_{D' \in PW^{e1}_{sum}} P(D') = P(D_1)$.

Next, since the second possible world does not have its own part, we add another event variable e_2 to divide the current set of possible worlds that are, i.e., $PW^{e2}_{sum} = PW_{\neg e1}$. The part of $\neg e_1$ is considered a complete part for e_2. Thus, $f_e(D_2) = \neg e_1 \wedge e_2$

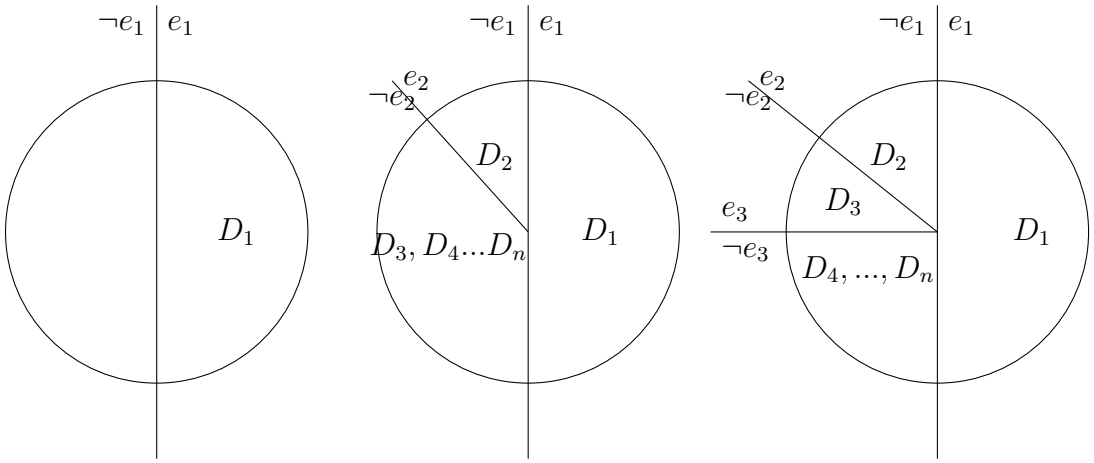
$PW^{e2}_{sum} = \{D_2, ..., D_n\}$

$PW_{e2} = \{D_2\}$

$PW_{\neg e1} = \{D_3, ..., D_n\}$

$P(e_2) = P(D_2) / \sum_{i=2}^{n} P(D_i)$.

This is shown as in Figure 3.6 (b).



(a) Add e_1 **(b) Add e_2** **(c) Add e_3**

Figure 3.6: Steps to build the conversion graph

We repeat the process of dividing the set of the possible worlds and obtain the corresponding formulas along with the probabilities of the event variables introduced until PW^e_{sum} cannot be further divided. Figure 3.7 shows the completed graph.

$f_e(D_{n-1}) = \neg e_1 \wedge \neg e_2 \wedge ... \wedge \neg e_{n-2} \wedge e_{n-1}$

$P(e_{n-1}) = \sum_{i=2}^{n-1} P(D_i) / \sum_{i=2}^{n} P(D_i)$

We verify that $P(f_e D_n) = P(D_n)$:

$P(f_e(D_n)) = P(\neg e_1 \wedge \neg e_2 \wedge ... \wedge \neg e_{n-2} \wedge \neg e_{n-1}) = (D_2 + D_3 + ... + D_n) * ((D_3 + D_4 + ... + D_n)/(D_2 + D_3 + ... + D_n)) * ((D_4 + D_5 + ... + D_n)/(D_3 + D_4 + ... + D_n)) * ... * D_n/(D_{n-1} + D_n) = D_n.$

As mentioned, the result of the conversion of all the possible worlds we obtained is the same as the conversion algorithm proposed in [5], described in Chapter 2.

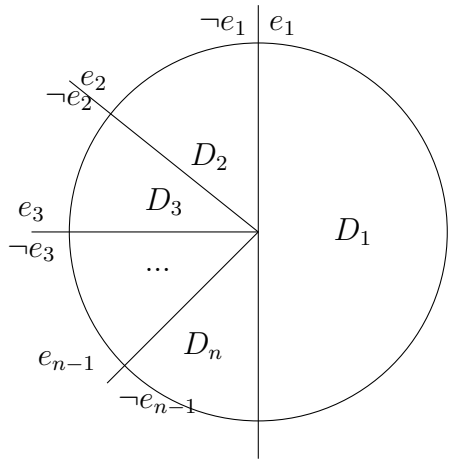


Figure 3.7: Conversion graph for Example 3.1

Intuitively, Example 3.1 adds the event variables in the counter clockwise order, and each time divides a possible world into PW_e. However, this is not the only possible approach. A conversion approach is valid if these three conditions are satisfied:

1. n-1 event variables are used
2. each possible world is mapped to a unique part
3. the probabilities of the event variables and possible world formulas are calculated based on (1) to (3) as described at the beginning of this section.

We will use the following possible conversion algorithms based on [8]. Intuitively, each added event variable e recursively divides PW^e_{sum} evenly into PW_e and $PW_{\neg e}$, until PW^e_{sum} contains only one possible world.

In every iteration, e is a new event variable that is added to the graph, and $P(e) =$

Algorithm 3.2 Convert a set of possible worlds to a pr-relation

Let PW^e_{sum}, PW_e, and $PW_{\neg e}$ be as defined above. Consider a set of $n-1$ independent event variables $E = \{e_1, e_2, ...e_{n-1}\}$. We obtain the event formula of each possible world recursively as follows. For every possible world D in PW^e_{sum}:

$$F(D) = \begin{cases} 1 & \text{if } |PW^e_{sum}| = 1 \\ e \wedge F(D) with PW^e_{sum} = PW_e & \text{if } D \in PW_e \\ \neg e \wedge F(D) with PW^e_{sum} = PW_{\neg e} & \text{if } D \in PW_{\neg e} \end{cases} \quad (4)$$

$\sum_{D \in PW_e} P(D)$ / $\sum_{D' \in PW^e_{sum}} P(D')$. For the set of possible worlds $\{D_i, D_{i+1}, ..., D_j\}$ in PW^e_{sum}, we have that:

$PW_e = \{D_i, D_{i+1}, ..., D_{(i+j)/2}\}$

$PW_{\neg e} = \{D_{1+(i+j)/2}, D_{2+(i+j)/2}, ..., D_j\}$.

That is, every newly added event variable e divides the existing set of possible worlds PW^e_{sum} into the subsets PW_e and $PW_{\neg e}$ of equal size.

Once we get $f_e(D)$ for every possible world, the event formula $f_e(t)$ for each tuple t in the tuple set is defined as $f_e(t) = \bigvee(f_e(D_i)|t \in D_i)$, obtained by the disjunction of the formulas of each possible world D_i that includes t.

Even though this approach also uses $n-1$ event variables, the length of the event formula for each possible world is logarithmic, as opposed to being linear, as shown in Example 3.1. It is easier to manipulate the formula or replace the event variables. For instance, converting the four possible worlds D_1, D_2, D_3 and D_4 with e_1, e_2, e_3. With the algorithm shown in Example 3.1, $f_e(D_1) = e_1$, $f_e(D_2) = \neg e_1 \wedge e_2$, $f_e(D_3) = \neg e_1 \wedge \neg e_2 \wedge e_3$, $f_e(D_4) = \neg e_1 \wedge \neg e_2 \wedge \neg e_3$, whereas with Algorithm 3.2, each

event variable is in the formulas of a subset of the possible worlds: $f_e(D_1) = e_1 \wedge e_2$, $f_e(D_2) = e_1 \wedge \neg e_2$, $f_e(D_3) = \neg e_1 \wedge e_3$, $f_e(D_4) = \neg e_1 \wedge \neg e_3$. If e_2 can be eliminated by the rest of the event variables, with the first conversion algorithm, the length of the event formula of most ($n + 1 - i$ for e_i) possible world will further increase, while in the second conversion algorithm, only formulas of a logarithmic number of possible worlds will be updated.

According to the above conversion algorithms, the probabilities of the event variables are contributed by the probabilities of the possible worlds. In return, the combination of event variables represents the probability of each possible world. To keep track of the internal relationships between the probabilities of the possible worlds and the event variables, we define *Interpreted Event Variable(IEV)* as follows.

Definition 3.2. Interpreted Event Variable (IEV): Let e be an event variable, PW^e_{sum}, PW_e, $PW_{\neg e}$, and $P(e)$ be in the generalized conversion algorithm defined above. The IEV of e is the probability formula represented by the possible worlds: $IEV(e) = \bigcup_{D \in PW_e} D \,/\, \bigcup_{D' \in PW^e_{sum}} D'$. Since the possible worlds are mutual exclusive, we have that $P(e) = P(IEV(e)) = \sum_{D \in PW_e} P(D) \;/\; \sum_{D' \in PW^e_{sum}} P(D')$. This is consistent with the probability calculation formula of e.

IEVs keep track of the probability assignment of event variables, when the probability distribution of the input data sources changes or is adjusted, the probability of the corresponding event variables can be updated accordingly.

Probability Consistency and Adjustment

Previous work in [5] and [22] focused on integration of two pr-relations, and on probability calculation of the integration result through EVFs. However, they did not consider probability constraints. When the data sources are probabilistic, the event formula of each tuple t, which is a boolean expression indicating the existence of the tuple under truth values, is also used to calculate probability of t and the probability associated with the possible worlds of the integration result.

Definition 3.3. (Probabilistic Consistency for Integrated epr-relation) Let epr_I be the epr-relation of the integration result obtained by I^P_{epr}. Then the set $GC(epr_I)$ of global constraints (GC) of epr_I, is defined as $\{w_1 \equiv w'_1, w_2 \equiv w'_2, ..., w_n \equiv w'_n\}$, where w_i and w'_i are the event attribute values of the same tuple t_i in different sources. Then $P(w_i) = P(w'_i)$ must be satisfied for all constraints in $GC(epr_I)$. Replacing every event variable e by $IEV(e)$, we obtain the consistency constraints $GC_{iev}(epr_I)$ which are the dependencies of the probability of the original input data sources.

Within each global constraint, w_i and w'_i are equivalent boolean formulas for any tuple t_i, and hence, either of them can be considered as the event attribute value of t_i. Since the probability $P(f_e(t))$ of a tuple t should be consistent regardless of the representation of $f_e(t)$, we have that $P(w_i) = P(w'_i)$.

Essentially, the dependency among data sources exists because of the common tuples. In other words, for every common tuple t, the sum $P(t)$ of the probabilities of the possible worlds that includes tuple t should be equal in all data sources.

Another constraint is that for every possible world D that is incompatible with possible worlds in other sources, $P(D) = 0$. Recall that a possible world D_i in S is compatible with the possible worlds from other sources if there is at least one possible world D_j in each other source S_j that makes every possible world in $PW_{NI} = \{D_i, D_1, D_2, ..., D_j\}$ compatible with all the other possible worlds in PW_{NI}. In other words, D_i is compatible if there is at lease one possible world D_{ijk} in the integration result to which D_i has contributed. Otherwise, the probability of a possible world that does not contribute to an integration result should be 0, for consistency reason. A more formal way to put it is as follows.

Definition 3.4. (Incompatible Possible World) Let D be a possible world in a probabilistic data source S_i. Let I be the set of possible worlds representing the integration result of a set of probabilistic data sources including S_i. $T(S)$ is the tuple set of S. If for every possible world $D' \in PW(D_I)$, $D \neq T(S_i) \cap D'$, then D is incompatible with possible worlds from other sources, and D does not contribute to the integration result. Thus, $P(D) = 0$.

.

The probability constraints from a system of equations that has to be solved. It

justifies the constraints considered in [19] reviewed in Chapter 2. When the number of input data sources is two, the interdependencies of tuples yield the probabilistic consistency constraints of the epr-relation of the integration result.

For out Example 2.1, the consistency constraints $GC_{iev}(epr_I)$ are as follows:

$$P(D_1) + P(D_2) = P(D'_1) + P(D'_2) \tag{5}$$

Since possible worlds in the same source sum to 1, and all the possible worlds contribute to the integration result, we know that $P(D_1) + P(D_2) + P(D_3) = 1$, and hence $P(D_3) = P(D'_3) + P(D'_4)$. These constraints are consistent with the equation (2) and (3) in I^P_{pw}.

The probabilistic consistency constraints in out approach is equivalent to the proposed constraints in [19], however, the advantage of the constraints in epr-relation is that they keep track of the probability relationships of the possible worlds of the data sources, instead of the intermediate integration results. In fact, they are the minimum set of constraints needed. When probability constraints are violated, probability adjustment applies to the possible worlds of the input data sources.

We propose a heuristic probability adjustment method as a possible solution to violated probability constraints.

(Heuristic probability adjustment) Let epr-relation epr_I be the integration

result of a set of data sources $\{S_1, S_2, ..., S_n\}$, and $GC_{iev}(epr_I)$ represents the probabilist consistency constraints induced by the possible worlds of the input data sources. In addition, the sum of the probabilities of the possible worlds in a data source is 1. The adjustment applies to the probability of possible worlds in the data sources, so that probability constraints can be satisfied.

1. We identify a set of possible worlds PW_{NI} such that for every possible world D, $P(D) = 0$, meaning that the possible worlds in PW_{NI} do not contribute to the integration result.

2. For each data source, the probabilities of possible worlds in PW_{NI} should be redistributed to other possible worlds in the same data source according to the proportion of their probabilities.

3. Simplify and solve equations so that a possible world does not appear twice in the equations, thus provides a unique adjustment of its probability. For each constraint $IEV(w_i) = IEV(w_i')$ if $P(IEV(w_i)) \neq P(IEV(w_i'))$, we assign to $P(IEV(w_i))$ and $P(IEV(w_i'))$ the average $0.5 * (P(IEV(w_i)) + P(IEV(w_i'))$. The probability changes reflected on each individual possible world are based on the proportion of their probabilities.

Integration Procedure

We next introduce the extended uncertain data integration algorithm, and illustrate how it works using an example. In the algorithm, the data sources are probabilistic. However, uncertain data sources without probabilities can still follow the integration algorithm without probability calculation step.

Algorithm 4 (Extended uncertain data integration in compact model) Let s_k be a set of uncertain data sources D_1 to D_k, and $P(D_i)$ represents the probability of D_i.

1. Using Algorithm 3.2, we convert data sources in s_k to a set PR_k of pr-relations. That is, $PR_k = \{pr_1, ...pr_k\}$.

2. Integrate PR_k using Algorithm 3.1, where the base integration algorithm is I_{epr}. The integration result is epr_I, containing a set of global constraints $GC(epr_I)$.

3. Verify the probabilistic constraints of epr_I, and adjust the probabilities when the constraints are not satisfied.

The integration result epr_I can be converted to a set of possible worlds by truth assignments. Probability distribution of each possible world D_i is calculated by $EVF(D_i)$.

Theorem 3.5. *Extended uncertain data integration in the compact model is idempotent, commutative and associative.*

Proof. Let S_1, S_2 be two uncertain data sources, pr_1 and pr_2 represent their pr-relations. As discussed earlier, the base two-source integration algorithm I_{epr} is idempotent, commutative and associative. We only need to verify probability distribution of the integration result.

- Idempotent: Let pr_1 be a ppr-relation ppr_1 without probability distribution, so that I_{epr} can be used to integrate the possible worlds. Since $I_{epr}(ppr_1, ppr_1) = ppr_1$, for each possible world D in the integration result, there is a possible world D' in the data source S_1 such that $T(D) = T(D')$, $EVF(D) = EVF(D')$, and $P(D) = P(D')$. That is, the probability distribution of the integration result is the same as that in the data source.

- Commutativity: As shown in Theorem 3.2, for every common tuple $t = t_i = t_j$, $f_e(t_i) = f_e(t_j)$. For every tuple t' that is unique in either S_1 or S_2, $f(t')$ is the same as in ppr_{I1} and ppr_{I2}. Thus for each pair of possible worlds D_i in ppr_{I1} and D_j in ppr_{I2} such that $T(D_i) = T(D_j)$, we have that $EVF(D_i) = EVF(D_j)$, and hence the probability distribution of ppr_{I1} and ppr_{I2} are the same.

- Associative: Let S_3 be a third data source. The integration algorithm I makes use of the associativity of I_{epr} to recursively integrate the possible worlds of a set of pr-relations. Probability validation and adjustment are done as the last step of the integration. Therefore, $I(I(S_1, S_2), S_3)$ and $I(I(S_2, S_3), S_1)$ generate

equivalent pr-relations pr_{I1} and pr_{I2} in that $f_e(t) = f_e(t')$ for every tuple $t \in pr_{I1}$, $t' \in pr_{I2}$ and $t = t'$. Their event variable sets are both the union of the variables from input data sources. Thus convert this compact relation to a set of possible worlds, for every possible worlds D and D', $EVF(D) = EVF(D')$, $P(D) = P(D')$, and $PW(I(I(S_1, S_2), S_3)) = PW(I(I(S_2, S_3), S_1))$. Different integration orders generate equivalent results.

□

We demonstrate the integration process with the example used for Theorem 3.3. The constraints checking and probability calculation for tuples are computed by our implemented system.

1) Convert S_1, S_2 and S_3 into pr-relations. $PW(S_1) = \{D_1, D_2, D_3\}$. There are three possible worlds, two event variables are needed. $V(S_1) = \{e_1, e_2\}$. $PW_{e1} = \{D_1, D_2\}$, $PW_{\neg e1} = \{D_3\}$. $PW^{e2}_{sum} = PW_{e1} = \{D_1, D_2\}$, $PW_{e2} = \{D_1\}$, $PW_{\neg e2} = \{D_2\}$. Thus $f_e(D_1) = e_1 \wedge e_2$, $f_e(D_2) = e_1 \wedge \neg e_2$, $f_e(D_3) = \neg e_1$. We convert S_2 and S_3 in a similar way. The pr-relations are shown in Figures 3.4, 3.5 and 3.6.

student	course	E
Bob	CS100	e_1
Bob	CS101	$\neg e_1 \vee (e_1 \wedge \neg e_2)$

Table 3.4: pr_1

$IEV(e_1) = D_1 \cup D_2$, $IEV(e_2) = D_1/(D_1 \cup D_2)$, $IEV(e'_1) = D'_1 \cup D'_2$, $IEV(e'_2) =$

student	course	E
Bob	CS100	e'_1
Bob	CS201	$\neg e'_1 \vee (e'_1 \wedge \neg e'_2)$
Bob	CS202	$\neg e'_1 \wedge \neg e'_3$

Table 3.5: pr_2

student	course	E
Bob	CS300	e''_1
Bob	CS101	$\neg e''_1$

Table 3.6: pr_3

$D'_1/(D'_1 \cup D'_2)$, $IEV(e'_3) = D'_3/(D'_3 \cup D'_4)$, $IEV(e''_1) = D''_1$.

$P(e_1) = 0.6$, $P(e_2) = 0.33$, $P(e'_1) = 0.6$, $P(e'_2) = 0.17$, $P(e'_3) = 0.5$, $P(e''_1) = 0.2$.

2) Integrate pr_1, pr_2 and pr_3 recursively. The integration result is shown in Figure 3.7.

student	course	E
Bob	CS100	e_1
Bob	CS101	$\neg e_1 \vee (e_1 \wedge \neg e_2)$
Bob	CS201	$\neg e'_1 \vee (e'_1 \wedge \neg e'_2)$
Bob	CS202	$\neg e'_1 \wedge \neg e'_3$
Bob	CS300	e''_1
$e_1 \equiv e'_1$		
$\neg e_1 \vee (e_1 \wedge \neg e_2) \equiv \neg e''_1$		

Table 3.7: epr-relation of the integration result

3) Probabilistic consistency checking: validate $P(w_i) = P(w'_i)$ or represent the global constraints in the form of IEV. This yields the following two probabilistic constraints:

$P(D_1) + P(D_2) = P(D'_1) + P(D'_2)$

$P(D_2) + P(D_3) = P(D''_2)$

These constraints are satisfied. We convert the integration result to a set of possible worlds associated with probabilities computed by EVF.

$T(D_{111}) = \{(Bob, CS100), (Bob, CS300)\}$.

$P(D_{111}) = P(e_1 \wedge e''_1 \wedge \neg(\neg e_1 \vee (e_1 \wedge \neg e_2)) \wedge \neg(\neg e'_1 \vee (e'_1 \wedge \neg e'_2)) \wedge \neg(\neg e'_1 \vee \neg e'_3)) = 0.03$.

Similarly $P(D_{121}) = 0.17$, $P(D_{212}) = 0.07$, $P(D_{222}) = 0.33$, $P(D_{332}) = 0.2$, $P(D_{342}) = 0.2$.

We now revisit the example of integrating multiple sources with I^P_{pw}. We define the semantics of multiple-source integration:

1. Consider data sources as pure uncertain data without probabilities. We then recursively integrate data sources using I_{pw}.

2. Check the probability consistency constraints that probabilities of every common tuple t is equal in all the data sources.

3. Probability adjustment.

Since on-the-fly probability adjustment may lead to different results in general, we propose to integrate sources as pure uncertain data, and postponed constraints checking and probability adjustment to the end. Given the probability constraints, the probability of each possible world in the integration result can be computed by the

probabilities of data sources. For instance, in the previous example, $P(D_{111}) = P(D_1 \wedge D_1' \wedge D_1'')$. Using conditional probability, $P(D_{111}) = P(D_1) * P(D_1')/(P(D_1') + P(D_2')) * 1 = 0.2/6 = 0.03$. This result is consistent with our calculation in the example.

Chapter 4

System Architecture and Implementation

This chapter presents the technical details of the running prototypes we developed for uncertain data integration proposed in Chapter 3. It includes two systems: a compact uncertain data integration framework, called InPRS, over multiple probabilistic data sources, and an integration-query system over multiple *pure* uncertain data sources presented in the possible worlds model without probabilities associated. The goal of the implementation of InPRS is to illustrate the feasibility of the ideas proposed uncertain data integration framework. The integration-query system builds a foundation to understand the semantics of queries over the integrated framework. The various calculation in our examples in Chapter 3 and the results are produced

by these systems. This system is developed using the Java programming language. An overview of the system architecture is provided in Figures 4.1 and 4.2.

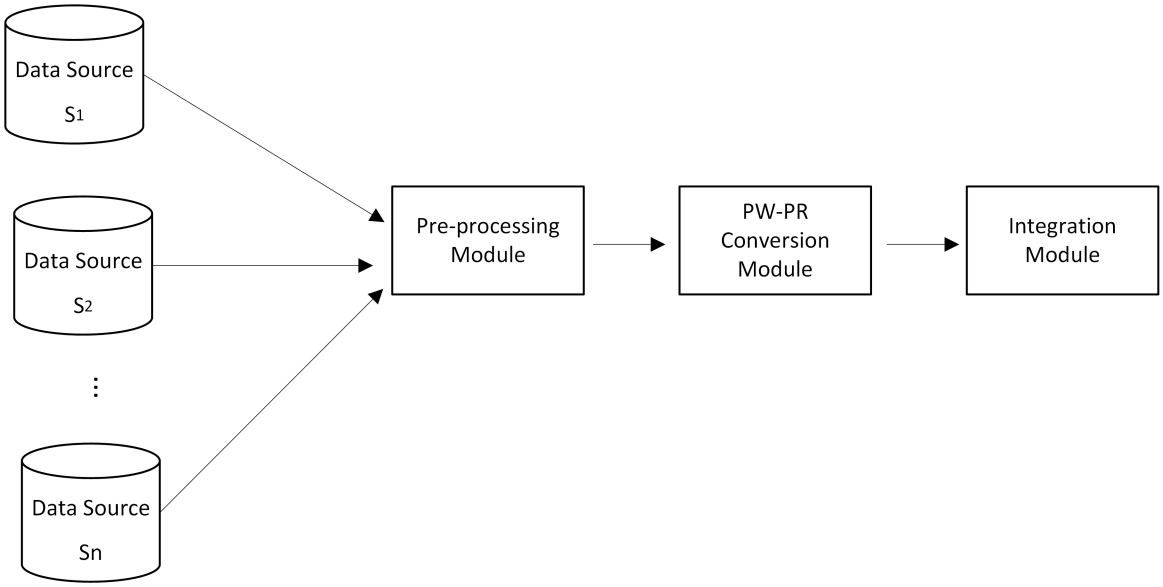


Figure 4.1: System Architecture of InPRS

4.1 InPRS

As shown in Figure 4.1, the system includes three modules: pre-processing module, PW-PR conversion module and integration module. These modules interact with each other to complete the integration process.

Input Data Sources

In the extended integration operation discussed in Chapter 3, the set of input data sources are probabilistic data each of which is represented as a set of possible worlds.

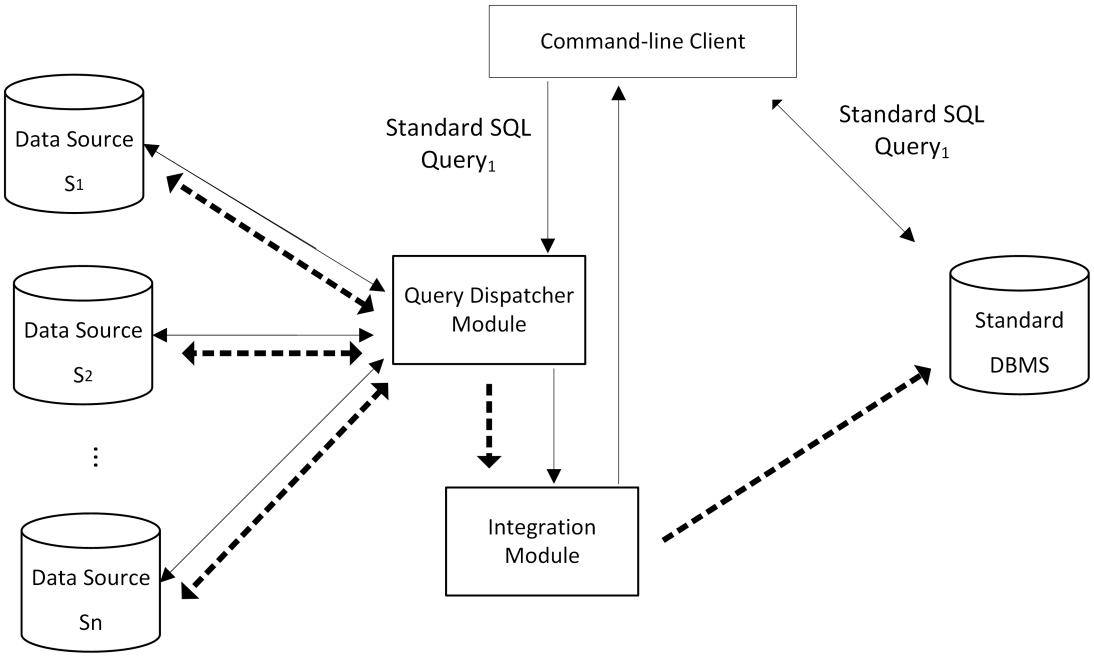


Figure 4.2: System Architecture of the Integration-Query system

In our system, each source is stored and managed by a standard relational DBMS. Within each source, each possible world is stored as a standard relation. There is an extra relation that records the probability distribution of all the possible worlds. The detailed information of the data sources, such as IP address and credentials to the remote server, relation name and schema of the databases, are in a property file.

Pre-processing Module

Given the property file, the pre-processing module connects to each data source and loads the data into an internal data structure in the main memory. This module is designed and implemented based on our conventions and structure of the possible worlds of the input data sources. For instance, an uncertain relation called *Register*

is represented as a set of of possible worlds, each of which is stored on a standard relation in the DBMS.

PW-PR Conversion Module

This module takes a set of possible worlds generated by the pre-processing module, and convert each one into a pr-relation, in which each tuple is annotated by a formula of event variables. The probability of each tuple is calculated by the formula. Each event variable E maps to a formula over the input possible worlds, shown by $IEV(e)$. In this case, if the probabilities of input possible worlds are changed, the change will propagate and update the probability of each tuple.

Integration Module

The integration module iteratively integrates the set of pr-relations passed by the PW-PR Conversion Module and eventually generate an epr-relation along with a group of IEV mappings for every event variable.

Details of the internal design and implementation are described below.

Pre-processing Module

The pre-processing module is responsible for loading data into the internal data structure and objects. When a user launches the Java program, the user needs to specify the property file containing information of the data sources. A sample property file is shown in Figure 4.3.

```
registration=(STUDENT CHAR(30) NOT NULL,COURSE CHAR(50) NOT NULL,
primary key(student, course))
relationName=register;registration
register=(student character(30) NOT NULL,course character(50) NOT
NULL,grade integer NOT NULL)
```

Figure 4.3: Sample property file

The assumption is that the schema of each common relation is the same in all the sources. Before describing the work flow of loading data based on these information, we first explain the data structure to represent the data sources. We take advantage of the object oriented design principles and set the representation models as follows.

Data Structure

- Each data source contains a collection of mappings from a relation name to the relation object. It also keeps a collection of schemas, mapping from a relation name to its schema. The information of remote server are stored in the data source object as they are properties.

- Each relation is represented as a set of possible worlds.

- Each possible world contains a set of tuples, an *EventVariables* object, the name of this possible world in the source database, and the probability of this possible world.

- Each tuple is represented as a list of *string*s. Each for a value of the corresponding attribute/column. These values together form a tuple as we know in

databases. We rewrite the object comparison function so that tuples with the same string value are considered to be the same entities. In this case they are treated as a common tuple.

- Each EventVariables object represents an event variable formula for a possible world. It consists of a set of positive variables string and a set of negative variables string. Thus, the formula it represents is the conjunction of each element in the positive variable set, and the negation of the elements in the negative variable set.

- PRDataSource stores the converted pr-relations for each data source. Each PRDataSource has a collection of mappings from relation names to pr-relation objects. A data source may contain a number of relations, each of which will be converted into a single pr-relation

- Each pr-relation object maintains a collection of mapping. The key is a Tuple object and the value is a set of EventVariables. It means that the event formula of each tuple is the disjunction of the event formula represented by EventVariables. There is also a collection of mappings from the event variable to a *ProbInPWs* object. The mapping represents IEV function to keep track of the relationship between an event variable and the possible worlds in the data source.

- ProbInPWs is used to calculate probabilities based on the possible worlds. It has a *Numerator* member, a *Denominator* member, each of which is a list of possible world objects. A *calculateProb()* function will divide the sum of the probabilities of the possible worlds in Numerator by the probabilities of those in the Denominator, and assigns this value to a variable member called *prob*. Storing this value avoids the overhead of recalculation when the sources are not changed.

- Epr-relation class extends pr-relation class, so that all the members in pr-relation will be in epr-relation. In addition, a HashMap keeps all the global constraints. The key is a Tuple object, and the value is a set of sets. Within the inner set is the EventVariables object. As mentioned in pr-relation object, the event formula of a tuple is a set of EventVariables object. Here the value of the map means a set of such event formula of the same tuple that are equivalent to each other.

When the property file is specified, the system loads source information to each source object. Given this information, an instance of the class called DBConnector connects to each data source. In each data source, the system loads the names of all tables, goes to each table and fill the data into a new possible world object. It also retrieves the probability of this possible world from the data source. When all the information of a possible world are loaded, the possible world is added to the relation hash map

in the data source object. In this way, all data sources are loaded into the specified data structures in memory.

PW-PR Conversion Module

After data is loaded into each data source object, the following three steps are taken in this module.

1. Compute event formula for each possible world and IEV, for each event variable and every data source.

2. With the event formula of each possible world generated, each relation in the data sources is converted from a set of possible worlds to a pr-relation.

3. Resolve and simplify the event formula of the tuples in each relation.

The algorithm to compute event formula for possible worlds is shown in Algorithm 4.1. Event variables are added to the EventVariables object in each possible world iteratively. The hash map structure to represent IEV of event variables for each relation in each data source is also filled during the process. We use the system to run and calculate for Example 3.4. After performing this step, we set the output of the system shown in Figure 4.4.

Algorithm 4.1 Compute Event Formula for Possible Worlds

1: **function** ComputeEV(pws, evs)
Input: pws: a list of possible worlds, evs: a hash map with the key as event variable mapping to its probabilistic (a ProbInPWs object)
2: **if** the size of pws is 1 **then**
3: return
4: **else**
5: generate new event variable e_i
6: declare a new ProbInPWs object ▷ $ProbInPWs$ represents the $IEV(e_i)$
7: $ProbInPWs.Denominator \leftarrow pws$
8: $size \leftarrow$ the number of possible worlds in pws
9: $mid = size$ / 2
10: **for** $j = 0$ to mid **do**
11: add e_i to positive variable set of PW_j
12: $ProbInPWs.Numerator \leftarrow PW_j$
▷ add the jth possible world to the Numerator set of $ProbInPWs$
13: **end for**
14: **for** $l = mid + 1$ to $size$ **do**
15: add e_i to negative variable set of PW_l
16: $PW_{\neg sum} \leftarrow PW_l$
▷ add the lth possible world to a list of possible world called $PW_{\neg sum}$
17: **end for**
18: put (e_i, $ProbInPWs$) pair in evs
19: ComputeEV($ProbInPWs.Numerator, evs$)
20: ComputeEV($PW_{\neg sum}, evs$)
21: **end if**
22: **end function**

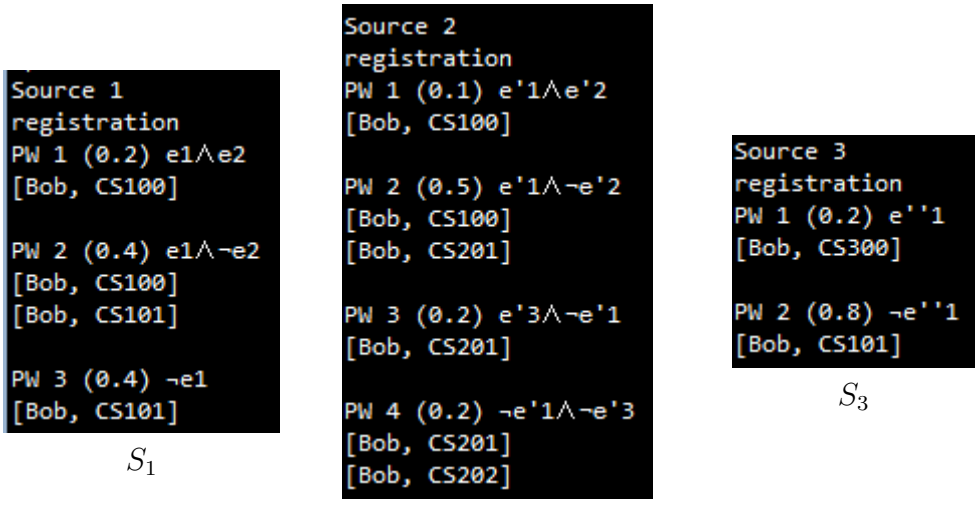


Figure 4.4: Event Formula of Possible Worlds in the Sources

Next, the module creates a pr-relation from a set of possible worlds. The algorithm is shown in Algorithm 4.2. This is an implementation to the evenly conversion algorithm in Chapter 3. The hash map from the event variable names to ProbInPWs objects is an empty input object to be filled when each new event variable is initialized along the process. The event formula of a tuple t is the disjunction of the formula of a set of possible worlds that includes t. In our representation model, there is a set of EventVariables objects for each tuple. The set represents the disjunction of the formula an EventVariables stands for. The converted pr-relations for each data source are stored in a PRDataSource object.

To reduce the length of the event formula for a tuple and simplify probability calculation if possible, where a function to solve formula takes a pr-relation and

Algorithm 4.2 Convert a Set of Possible Worlds to a Pr-relation

1: **function** PWStoProbRelation(pws, evs)
Input: pws: a set of possible worlds, evs: a hash map with the key as event variable mapping to its probabilistic (a ProbInPWs object)
Output: a pr-relation object
2: declare an empty pr-relation pr
3: **for all** possible world p in pws **do**
4: **for all** tuple t in p **do**
5: **if** t not in pr **then**
6: create a set of EventVariables $evSet$
7: pr.Tuples ← (t, evSet)
▷ $pr.Tuples$ is a the hash map of tuple to a set of EventVariables representing the event formula of that tuple
8: **end if**
9: ev.PositiveVariables ← positive variable set of the EventVariables in p
10: ev.NegativeVariables ← negative variable set of the EventVariables in p ▷ ev is a new EventVariables object, it copies the EventVariables of p
11: the value of the key t in pr.Tuples ← key-value pair (t, ev)
12: **end for**
13: **end for**
14: set evs to pr
▷ set evs to the hash map of (event variable map, EventVariables) in pr
15: **end function**

merges the event formula of a tuple in the following situation:

If $f_e(t) = (f_1 \wedge e_i) \vee (f_1 \wedge \neg e_i)$, then $f_e(t) = f_1$. f_1 is a subset of the formula. We compare every pair (EV_i, EV_j) of EventVariables objects for each tuple. If the number of positive and negative variables are equal in EV_i and EV_j, and the positive variables set of EV_i contains all positive and negative variables in EV_j, we remove EV_j and the common negative variable of EV_j in EV_i, and vice versa. At the end, if in a EventVariables object, both the positive variable set and the negative variable set are empty, we remove this EventVariables from the hash map. The following examples illustrate the simplification process above.

Suppose $f_e(t) = (e_i \wedge e_j) \vee (e_i \wedge \neg e_j) = e_i$, $f_e(t') = e_k \vee \neg e_k = true$.

Following the three steps above in this module, the converted relations in Example 3.4 are shown in Figures 4.5, 4.6, and 4.7.

```
[Bob, CS101](e1∧¬e2)∨(¬e1) 0.8
[Bob, CS100](e1) 0.6
e1 = P(registration_d1s1) + P(registration_d2s1)    = 0.6
e2 = P(registration_d1s1)  / (P(registration_d1s1) +P(registration_d2s1)) = 0.33
```

Figure 4.5: Pr-relation of Source S_1

Integration Module

The converted PRDataSources are passed to the integration module. As mentioned earlier in the data structure of the system, each epr-relation is a pr-relation with the

```
[Bob, CS202](¬e'1∧¬e'3) 0.2
[Bob, CS201](e'1∧¬e'2)∨(¬e'1) 0.9
[Bob, CS100](e'1) 0.6
e'1 = P(registration_d1s2) + P(registration_d2s2)    = 0.6
e'2 = P(registration_d1s2)  / (P(registration_d1s2) +P(registration_d2s2)) = 0.17
e'3 = P(registration_d3s2)  / (P(registration_d3s2) +P(registration_d4s2)) = 0.5
```

Figure 4.6: Pr-relation of Source S_2

```
[Bob, CS101](¬e''1) 0.8
[Bob, CS300](e''1) 0.2
e''1 = P(registration_d1s3)    = 0.2
```

Figure 4.7: Pr-relation of Source S_3

addition of a hash map called *gConstraints*, to maintain all the global constraints using a key which is the tuple object and the value which is a set of EventVariables sets. The hash map *gConstraints* represents the equivalence of a set of event formula of the same tuple. Given a set of PRDataSources, we iteratively integrate two PRDataSource. First we convert each source to an epr-relation with an empty hash map *gConstraints*. Then we merge two epr-relations using Algorithm 4.3.

Given a set of epr-relations, we iteratively integrate two sources at a time, iteratively until we have integrated all the sources. We then merge tuple sets of the sources and their event variables, and add the common tuple's alternative event formula representation to the hash map of the global constraints. After the conversion, we get an epr-relation. The epr-relation of Example 3.4 is shown in Figure 4.8. The probability of each tuple t is calculated in the process based on the event formula associated with

Algorithm 4.3 Integrate Two Epr-relations to a epr-relation

1: **function** PWStoProbRelation(epr_1, epr_2)
Input: epr_1, epr_2: two epr-relations
Output: epr_I: an epr-relation object
2: declare an empty epr-relation epr_I
3: **for all** tuple t in pr_1 **do**
4: $epr_I.Tuples \leftarrow epr_1.Tuples$
▷ $epr.Tuples$ is the hash map of tuple to a set of EventVariables representing the event formula of that tuple
5: **end for**
6: $epr_I.IEV \leftarrow epr_1.IEV$
▷ $epr.IEV$ is the hash map in epr contains the mapping from an event variable to a ProbInPWs object representing the IEV of the event variable
7: **for all** Tuple t' in epr_2 **do**
8: **if** t' is not in $epr_I.Tuples$ **then**
9: add the pair (t', set of EventVariables of t') in $epr_2.Tuples$ to $epr_I.Tuples$
10: **else**
11: **if** t' is not in $epr_I.gConstraints$ **then**
12: declare a set of sets called *constraint* containing EventVariables.
13: *constraint* $\leftarrow$ the value of t' in $epr_2.Tuples$
14: add the pair of t' and *constraint* to $epr_I.gConstraints$
15: **else**
16: add the set of EventVariables of t' in $epr_2.Tuples$ to the value of entry t' in $epr_I.gConstraints$
17: **end if**
18: **end if**
19: **end for**
20: add $epr_2.IEV$ to $epr_I.IEV$
21: **end function**

```
Integration Result:
Relation registration
[Bob, CS202](¬e'1∧¬e'3) 0.2
[Bob, CS201](e'1∧¬e'2)∨(¬e'1) 0.9
[Bob, CS101](e1∧¬e2)∨(¬e1) 0.8
[Bob, CS100](e1) 0.6
[Bob, CS300](e''1) 0.2
(e1∧¬e2)∨(¬e1) == ¬e''1
e1 == e'1

e''1 = P(registration_d1s3)   = 0.2
e1 = P(registration_d1s1) + P(registration_d2s1)   = 0.6
e2 = P(registration_d1s1)  / (P(registration_d1s1) +P(registration_d2s1)) = 0.33
e'1 = P(registration_d1s2) + P(registration_d2s2)   = 0.6
e'2 = P(registration_d1s2)  / (P(registration_d1s2) +P(registration_d2s2)) = 0.17
e'3 = P(registration_d3s2)  / (P(registration_d3s2) +P(registration_d4s2)) = 0.5
```

Figure 4.8: Integration Result of S_1, S_2 and S_3

t. We store the probability value, but re-evaluate it and ensure the value is up-to-date when there is any changes in the input possible worlds. In stead, we use IEV to calculate the probability of each event variable. For instance, the ProbInPWs of an event variable e contains D_1, D_2 as numerator and D_1, D_2, D_3, D_4 as denominator. Then we have $P(e) = (P(D_1) + P(D_2))/(P(D_1) + P(D_2) + P(D_3) + P(D_4))$.

User can further interpret the integration result to verify probability consistency as follows:

1. Find possible worlds in the data sources that do not contribute to the integration result, and set their probability to 0. Distribute their probability value within the same source.

2. Call the *checkConstraints*() function in the epr-relation object and compare

the probability in each entry of the hash map $gConstraints$. That is, checking if for tuple t, there are the constraints $f_e(t) \equiv f_{e_1}(t) \equiv f_{e_2}(t) \equiv ..$, $P(f_{e_1}(t)) = P(f_{e_2}(t)) = ...$

3. If probability consistency checking fails, further probability adjustment should be applied to input possible worlds. The adjustment should be based on the interdependency represented by the global constraints, represented in the form of possible worlds, GC_{iev}. That is, use $IEV(e)$ to replace the event variables in the global constraints and get a set of equations represented by the possible worlds from data sources.

4.2 Integration-query System

We also build a prototype integration-query system over multiple pure uncertain data sources to evaluate the semantics of queries over the integration framework. Since possible worlds model is the semantics of uncertain data source, our system also uses possible worlds model as the model for data source representation. The data sources are pure uncertain data without probability distribution. Querying over probabilistic data can use this as the foundation and focus more on the probability distribution of the result.

The architecture of this system is shown in Figure 4.2. The three-tier system

contains a command line user interface, an application tier, and a data tier. The application tier consists of a query dispatcher module and an integration module over a set of possible worlds. The data tier is a standard DBMS, for which we used PostgreSQL in our implementation.

The system is used to compare the results of *integration first* structure and *query first* structure. In the *Integration first* approach, we first integrate the data sources into the database, on which we evaluate the query result, as opposed to evaluate query over each individual data source. The *query first* approach is used on the other hand, when a query q is rewritten according to the relations in data sources, and then the system integrates the query results from different data sources and integrate them into a single result to the query q. Using two data sources S_1 and S_2, and a query q, the *query first* and *integration first* approaches of query processing in integration frameworks means $Q(I(S_1, S_2))$ versus $I(Q(S_1), Q(S_2))$. The question now is how the two compares. Later this section we will present the examples, investigate this problem and the challenges posed.

A property file with the same structure shown in Figure 4.3 is loaded when initializing the system. The system then connects to data sources, loads all the data and integrates them. The integration result is stored in the database I. When user inputs a query q to the system, q is rewritten and sent separately to query dispatcher module and the database I. Query dispatcher rewrites and sends q to each data source, and

collects the query results from each sources. The query results are then passed to the integration module, before returning to the user. On the other hand, database I executes q and returns the query result to the user. We then integrate the results and compare with the previous result.

We present the data structure in the system. The structure of the input data sources is the same as in InPRS system, i.e. each possible world is stored as a standard relation in the database.

- Each data source object contains a hash map, called *relations*, with the key being the relation name, and value being a relation object. It also contains information about the data source.

- Each relation object consists of a set of possible world objects, a hash map structure consisting of tuple object and propositional variable name pairs, representing the tuple set and a set of mappings from a string variable name to its boolean value, called $logicalExpression$. The set represents the disjunction relationship of the logical expressions that the hash map returns.

- A possible world D has a string name indicating the relation in the DBMS that represents, a set of tuple objects, and a hash map of (variable name, boolean value) pairs as the logical expression of D.

- A tuple object is the same as the tuple in our InPRS system, introduced before.

We next describe technical details of the internal design and implementation of each module.

Query Dispatcher Module

This module takes a query q and identifies the relation names mentioned in the query that are in our data sources. For each data source, the module generates a set of queries, one for each concrete possible world in the matched relation. The module dispatches the rewritten queries to the corresponding source(s), collects query results, and returns a set of data source objects containing the query results. The data loading for the *integration first* approach also uses this module and the query is a projection on all the columns of each relation involved.

For instance, if the query is

q: **SELECT** * **FROM** registration **WHERE** student = 'Bob';

Suppose the data sources S_1 has two possible worlds for relation *registration*: $registration_d1$, $registration_d2$. Then the rewritten queries q passed to S_1 are:

q = **SELECT** * **FROM** *registration_d*1 **WHERE** student = 'Bob';

q = **SELECT** * **FROM** *registration_d*2 **WHERE** student = 'Bob';

Now suppose the relation *student_record* in S_1 has three possible worlds, and a new query q' involves both relations *registration* and *student_record*. The query dispatcher in this case will generate 6 queries for S_1.

Integration Module

The input of Integration Module is a set of DataSource objects, and its output is the object DataSource containing the integration result. The workflow of this module is as follows.

1. Based on the relation involved, the corresponding relation objects from different data sources are collected into a set. Recall that relation object consists of a set of possible world objects.

2. The set of the relation objects collected are passed to function $integratePWs()$ described in Algorithm 4.4. This is an implementation of Algorithm 3.1 in Chapter 3. The output is a single relation object containing a set of the possible worlds.

3. Steps 1 and 2 are repeated for each common relation in the data sources, and the result of the integration is kept in the same DataSource object.

In line 13 of Algorithm 4.4, a function is called to integrate two relation objects, each containing a set of possible worlds. To integrate two sets of possible worlds, we should identify and use the compatible possible worlds from different data sources. For this, we first calculate the union and intersection of the tuple sets in the two relation objects. The union is the tuple set of the integration result, denoted by a relation object R_I. The intersection is the set of common tuples. If the intersection

Algorithm 4.4 Integrate Possible Worlds from Data Sources

1: **function** INTEGRATEPWS(*srcs*)
Input: *srcs*: a set of relation objects
Output: a relation object
2: **if** the size of srcs is 1 **then**
3: return the relation object in srcs
4: **end if**
5: take the first relation object r in srcs
6: compute tuple set of r
▷ generate a variable for each tuple, and store the variable in the tuple-variable mapping
7: **for all** possible world pw in r **do**
8: generate logical expression of p
▷ traverse tuple set of r and add (variable of tuple, $true$) to logical expression of pw if pw contains the tuple. add (variable of tuple, $false$) otherwise
9: add the logical expression of pw to the $logicalExpression$ hash map of r
10: **end for**
11: remove the first relation from *src*
12: *tempRelation* ← recursiveIntegratePWs(srcs)
13: return integrate(r, *tempRelation*)
14: **end function**

of tuple sets is empty, then there is no common tuples. Therefore, we go through the two sets of possible worlds and for each pair of possible worlds, we union the logical expression of the possible worlds, and add the set to R_I. The union set represents the logical expression of one possible world in the integration result. If there are common tuples, we update the variable for the common tuples in one source to the variables in the other, so that common tuples are uniformly represented by the same variable in both sources. We then go through the input possible worlds. If the existence of common tuples in the possible worlds pair is consistent, meaning common tuples exist in both of them, or none, we union the two possible worlds and add the result to R_I.

When the integration module is used in the integration of the data sources, the system writes the output result to PostgreSQL according to the $logicalExpression$ in the integration result, as possible world is saved as a standard relation in the database, this module is also used to integrate query results from the data sources using the proposed $query\ first$ approach.

The Integration-Query system implements the *query first* structure and *integration first* structure so that we could focus more on comparison and understanding of the results. The InPRS system shows the feasibility of the integration framework, and performs probability calculations. As mentioned before, all the probability calculation in the examples provided in this thesis were done by the system and also manually verified to be as expected. An important phase in our implementation is choosing the

"right" data structure. We design the classes so that their structure and relationships resemble the semantics of the data model considered. Our choices helped a better understanding of the requirements and implementation of the modules used in our system prototype.

Data integration in general is motivated by the desire of combining databases and data sources, so that they can work together and offer more information and better understanding of the data in overall. The integration system also provides a unified interface for query. Uncertain data integration is different from standard, exact data integration by its uncertain nature. For instance, in standard data integration, conflicting data in two sources means they cannot integrate before solving the inconsistency. However, in uncertain data integration, if two possible worlds are not consistent, they are considered incompatible, and they each may be able to integrate with other possible worlds in the other sources.

In this sense, many problems that have been well studied in standard data integration may remain a challenge for uncertain data integration. With the help of the Integration-Query system prototype, we compare and explore the *query first* and *integration first* approaches. This structural exploration helps us better understand the semantics of uncertain data integration. Consider the following Example 4.1:

Example 4.1. Suppose S_1, S_2 are two uncertain data sources, each of which contains a relation $Registration(student, course)$ and a relation $Tutorial(course, TA)$. The

possible worlds in S_1 and S_2 are as shown in Table 4.1, 4.2, 4.3 and 4.4. There is a query $q = Registration \bowtie Tutorial$.

Registration (student, course)
(Bob, CS100)

D_1

Registration (student, course)
(Bob, CS101)

D_2

Table 4.1: Possible Worlds of *Registration* in Source S_1

Tutorial (course, TA)
(CS100, Sam)

$D3_1$

Table 4.2: Possible Worlds of *Tutorial* in Source S_1

Registration (student, course)
(Bob, CS100)

D'_1

Table 4.3: Possible Worlds of *Registration* in Source S_2

Tutorial (course, TA)
(CS100, Jon)

D'_2

Tutorial (course, TA)
(CS101, Josh)

D'_3

Table 4.4: Possible Worlds of *Tutorial* in Source S_2

We follow the *integration first* approach and the compute the result of $Q(I(S_1, S_2))$, which is shown in Table 4.5. The result relation is named $Reference$.

Reference (student, course, TA)
(Bob, CS100, Sam)
(Bob, CS100, Jon)

D_{I1}

Reference (student, course, TA)
(Bob, CS100, Sam)

D_{I2}

Table 4.5: The Result of *Integration First* of q over S_1 and S_2

Reference (student, course, TA)
(Bob, CS100, Sam)
(Bob, CS100, Jon)

D'_{I1}

Reference (student, course, TA)
(Bob, CS100, Jon)

D'_{I2}

Reference (student, course, TA)
(Bob, CS100, Sam)

D'_{I3}

$\varnothing$

D'_{I4}

Table 4.6: The Result of *Query First* of q over S_1 and S_2

The query result of the *query first* approach is shown in Table 4.6.

We notice the difference in the query results between these two approaches. There are two more possible worlds in the result of *query first* approach than *integration first*. Note that data integration in general is a process of enriching knowledge, and the data sources complement each other. According to information theory [7], given the same set of tuples, an uncertain database with fewer possible worlds contains more information. Having more possible worlds indicates less precision in the knowledge. When the number of possible worlds is one, it becomes a standard, exact relation. Intuitively, *integration first* approach is similar to the concept of *data warehouse*

structure [12] [15] in standard data integration, in the sense that data are also loaded and materialized. While *query first* is similar to *mediator-based* integration architecture, that the physical independence of data sources is respected. *Integration first approach* is able to combine data from multiple sources to create new data beyond each source.

To avoid missing pieces of data combined from difference sources, standard mediator-based system answer queries using views [13]. The idea is that, data sources are defined as a view in terms of the global schema, these views give the system an idea of the relationships between data sources. The mediator-based system can combine these views to find all possible ways to answer a query. This view definition is also known as *local-as-view*.

In our *query first* approach, the system may rewrite the query as these sub-queries: $q_1 = Registration_1 \bowtie Tutorial_1$, $q_2 = Registration_1 \bowtie Tutorial_2$, $q_3 = Registration_2 \bowtie Tutorial_1$, $q_4 = Registration_2 \bowtie Tutorial_2$, then integrate the query results of them. We then get as in Table 4.7:

As we compare the result of Table 4.7 with 4.5, the solution for standard data integration cannot solve the problem in uncertain data integration, for the reason that, instead of keeping all the information in standard case, uncertain data integration removes incompatible possible worlds, thus, some tuples may be removed during the integration process. Simply combining data across different sources only solves

Reference (student, course, TA)
(Bob, CS100, Sam)
(Bob, CS100, Jon)

D''_{I1}

Reference (student, course, TA)
(Bob, CS100, Sam)

D''_{I3}

Reference (student, course, TA)
(Bob, CS100, Sam)
(Bob, CS101, Josh)

D''_{I2}

Table 4.7: The Result of $Query\ First$ of sub-queries of q over S_1 and S_2

the problem that sources are isolated and data in different sources do not interact, however, more reasonable query evaluation and processing algorithms are needed.

Another interesting problem in the query processing of uncertain data integration system is related to the tuple set. As we know, each data source contains a set of tuples, indicating the presence and absence of the tuples in each possible world. However, the tuple set cannot be carried over during query answering. Suppose a query q projects on some of the attributes of relation R, a tuple $t \in S_1$ cannot coexist with another tuple $t' \in S_2$. t, t' are the common tuples of S_1 and S_2. After the projection, the two tuples become the same, thus compatible. This as well causes inconsistency between *query first* and *integration first* approaches.

Chapter 5

Conclusion and Future Research

This research is motivated by the fact that, even though data integration is a popular and widely studied topic over the past two decades, not much work has done on integration of uncertain data, especially the uncertain, probabilistic data without the independent assumption. Our goal is to explore the semantics and techniques of uncertain data integration.

We study the existing work on uncertain and probabilistic data integration frameworks. We demonstrate the limitations and issues of existing solutions to integrate two data sources, and show that a simple extension does not generate deterministic integration result. We then study the properties of the proposed integration operations, and use these properties as the foundation of our framework. We propose an integration framework over multiple uncertain and probabilistic data sources. Within

the framework, we generalize the conversion algorithm from possible worlds model to the compact probabilistic relation model. We define the integration procedure, the IEV to track the relationship of event variables and data sources, the probability consistency among data sources, and a heuristic probability adjustment method when the constraints are violated. Our integration framework equally treats data sources, provides and enhances the overall view of multiple uncertain data sources. Our solution reduces to the existing technique for integrating two sources.

We build a running prototype of the proposed framework to show its feasibility and to automate the probability calculation. Our work can contribute towards uncertain data integration applications on a large-scale. We also build a running prototype to help better understand the relationships of query and integration. It can be a starting point to more general and complex query evaluation solution.

As the purpose of data integration is perhaps to be able to query the integrated system, an interesting direction for future work is to evaluate query over our compact integration result: an epr-relation.

Note that as the uncertain data is collected separately in different sources, the probability consistency requirement may not be satisfied in practice. In our proposed framework, the probability consistency constraints equations need to be solved before probability adjustment procedure kicks in. A more practical probability redistribution algorithm is an important next step to explore.

The prototype to demonstrate our proposed integration framework is a single-user, in-memory system. In real world applications, the data size is much larger. One way to speed up the process is to use parallelism for large uncertain data sets.

References

[1] Serge Abiteboul, Paris Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. *Theoretical Computer Science*, 78(1):159–187, 1991.

[2] Charu C Aggarwal. *Managing and Mining Uncertain Data: 3, A.*, volume 35. Springer Science & Business Media, 2010.

[3] Parag Agrawal. Incorporating uncertainty in data management and integration. August 2012.

[4] Parag Agrawal, Anish Das Sarma, Jeffrey Ullman, and Jennifer Widom. Foundations of uncertain-data integration. *Proceedings of the VLDB Endowment*, 3(1-2):1080–1090, 2010.

[5] Amir Dayyan Borhanian and Fereidoon Sadri. A compact representation for efficient uncertain-information integration. In *Proceedings of the 17th International Database Engineering & Applications Symposium*, pages 122–131. ACM, 2013.

[6] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Information integration: Conceptual modeling and reasoning support. In *Cooperative Information Systems, 1998. Proceedings. 3rd IFCIS International Conference on*, pages 280–289. IEEE, 1998.

[7] Thomas M Cover and Joy A Thomas. Elements of information, 1991.

[8] Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007.

[9] AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of data integration*. Elsevier, 2012.

[10] Xin Dong, Alon Y Halevy, and Cong Yu. Data integration with uncertainty. In *Proceedings of the 33rd international conference on Very large data bases*, pages 687–698. VLDB Endowment, 2007.

[11] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Integrating conflicting data: the role of source dependence. *Proceedings of the VLDB Endowment*, 2(1):550–561, 2009.

[12] Alon Halevy, Anand Rajaraman, and Joann Ordille. Data integration: the teenage years. In *Proceedings of the 32nd international conference on Very large data bases*, pages 9–16. VLDB Endowment, 2006.

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

[14] Ali Kiani and Nematollaah Shiri. A framework for information integration with uncertainty. In *Advanced Distributed Systems*, pages 194–206. Springer, 2005.

[15] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246. ACM, 2002.

[16] Jayant Madhavan, S Jeffery, Shirley Cohen, Xin Dong, David Ko, Cong Yu, and Alon Halevy. Web-scale data integration: You can only afford to pay as you go. CIDR, 2007.

[17] Matteo Magnani and Danilo Montesi. Uncertainty in data integration: current approaches and open problems. In *MUD*, pages 18–32, 2007.

[18] Matteo Magnani and Danilo Montesi. A survey on uncertainty management in data integration. *Journal of Data and Information Quality (JDIQ)*, 2(1):5, 2010.

[19] Fereidoon Sadri. On the foundations of probabilistic information integration. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 882–891. ACM, 2012.

[20] Fereidoon Sadri. Belief revision in uncertain data integration. In *Databases Theory and Applications*, pages 78–90. Springer, 2015.

[21] Anish Das Sarma. *Managing uncertain data*. PhD thesis, Stanford InfoLab, 2009.

[22] Gayatri Tallur. *Uncertain data integration with probabilities*. The University of North Carolina at Greensboro, 2013.

[23] Jennifer Widom. Trio: A system for integrated management of data, accuracy, and lineage. *Technical Report*, 2004.