THE RELATIONAL DATABASE THEORY OF BAYESIAN NETWORKS

A Thesis
Submitted to the Faculty of Graduate Studies and Research
In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in
Computer Science
University of Regina

By
Cory James Butz
Regina, Saskatchewan
January 2000

Abstract

Based on the elegant theory of relational databases, the present investigation establishes a unified model for both relational databases and Bayesian networks. This is in contradiction to the argument that relational databases and Bayesian networks are *different*, where it was shown that the implication problem does not coincide for *embedded multivalued dependency* (EMVD) and *probabilistic conditional independence* (CI). The main result of this thesis, however, is that the implication problem coincides on the *solvable* subclasses of EMVD and CI, but differs on the *unsolvable* general classes of EMVD and CI. This means that there is *no* practical difference between relational databases and Bayesian networks, since only the *solvable* subclasses are useful in the design of both of these knowledge systems.

A unified model provides the opportunity for cross-research. Recently, attempts have been made to generalize the standard Bayesian network model with contextual dependencies, an object-oriented Bayesian network model, and a multi-agent Bayesian network model. In this thesis, we demonstrate the usefulness of our unified model by making significant contributions to these extensions. By drawing from the highly developed relational database model, we propose more general probabilistic dependencies as well as several consistency results in the object-oriented and multi-agent models.

Acknowledgements

I would like to thank Dr. S.K.M. Wong for showing me how research is properly done. His blunt assessment of both my written work and ideas was at times hard to accept, but was a necessary component towards becoming a more mature writer and researcher. I will state for the record that any of my future achievements would not have been possible without the early guidance of Dr. Wong.

Many thanks to my external examiner, Dr. E. Neufeld, and the other members of my Ph.D. committee: Dr. H. Hamilton, Dr. S. Kirkland, Dr. M. Liu, Dr. R.B. Maguire, and Dr. L.V. Saxton. Their insightful comments, suggestions, and constructive criticisms resulted in an improved thesis.

I would also like to thank the Faculty of Graduate Studies, the Department of Computer Science, and Dr. S.K.M. Wong for financial support. It is important to have periods without distraction in order to entirely devote oneself to research.

Dr. Y.Y. Yao and Dr. R.B. Maguire were always very kind to me. I appreciate the support and advice they gave me.

I am also grateful to Irene Butz for just being Mom, and Connie Maguire for just being Connie. I also received support from Rod, Melanie, Jiahua, Chengyan, Chi, Yan, and OO.

Finally, I would like to thank my beautiful wife, Pei Song, for her love and devotion. Her steady encouragement played an important part in my being able to maintain a high level of enthusiasm.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

Owing to the dependence of application programs on the physical storage of data in existing database models, Codd [19] proposed the *relational database model* for the logical management of large data banks in transaction processing systems. Thus, one advantage of the relational database model is that it allows data to be viewed at an abstract level rather than a physical level. Another advantage of the relational database model stems from the principle that one is not interested in storing duplicate information in the management of data. The relational database model is ideally suited for this purpose since each relation is defined as a set of tuples. That is, all data is viewed as being stored in relations (tables), with each row summarizing some object or relationship in the real world. All duplicate tuples are then *ignored* in the manipulation of the relations. To complete the specification of the relational database model, Codd proposed abstract *query languages* [20, 22], and the first *data dependency* [19, 21] as a semantic tool to design better databases from a universal relation. An important result [18] was that an abstract query could be mapped onto the physical database in an efficient manner. At this point, the relational database model was about to have a remarkable impact on the study of database systems.

The development of the relational database model soon flourished into an elegant and robust framework, called the *theory of relational databases* [56]. In fact, several topics in the theory of relational databases developed theories of their own. These include the design of the database scheme, query processing, and data dependencies. *Data dependencies*, such as *embedded multivalued dependency* (EMVD), *multivalued* *dependency* (MVD) and *join dependency* (JD), can be used to provide an economical representation of a universal relation. Two of the most important results were: (i) the ability to specify the universal relation as the join of several smaller relations, and (ii) the development of an effective method to optimize an input query in order to only access the relevant portions of the database. In general, a culminating result by Beeri, Fagin, Maier and Yannakakis [9] was that the special class of *acyclic join dependencies* (AJDs) possess a number of desirable properties in database applications. The *theory of data dependencies* [29, 48, 84] continued for many years to produce numerous dependencies to express various properties of data. Thus, the relational database model provided a rigorous mathematical foundation for the management of data.

More generally, the relational database model served as a solid basis for the development of more general database models including complex-relational databases [1, 2], object-relational databases [2, 5], distributed databases [17], deductive databases [16], and temporal databases [81]. *Complex-relational databases* allow the value of an attribute to be itself a relation. *Object-relational databases* are an attempt to incorporate the object-oriented notions successfully applied in programming languages. *Distributed databases* model the situation where the data is not stored in a single repository, but is instead stored in multiple sites connected via a network. Relational databases and logic programming were unified in *deductive databases*. Finally, *temporal databases* [81] are an attempt to introduce the notion of parametric data. Each of these database models could adopt the established *theories of concurrency control and recovery* [10, 63] in addressing the problems of asynchronous access to shared data and system failure in database systems. Thus, although a relative newcomer to the field of computer science, database systems became arguably its most highly developed subject area.

Another important subject in computer science involves the development of probabilistic reasoning systems [38, 62, 66] for the management of uncertain knowledge. In particular, Pearl's [66] *Bayesian network model* has become a proven framework for uncertainty management in many artificial intelligence applications including medical diagnosis. In this approach, one assumes that all knowledge can be represented as a joint probability distribution. As in the relational database model, each tuple summarizes some observation in the real world. However, it is essential to *count* the number of duplicate tuples in the probabilistic approach to uncertainty management. At the same time, Pearl's two major contributions parallel major relational database results: (i) the ability to specify the joint probability distribution as the product of conditional probability distributions, and (ii) the development of an effective method to optimize an input query in order to only access the relevant portions of the Bayesian network. Thus, probability theory provided a rigorous foundation for the development of Bayesian networks for uncertainty management.

The main argument in this thesis is that our *Bayesian database model* [89, 96] provides a *unified* model for both Bayesian networks and relational databases. This single model, providing more insight than mere similarities between relational databases and Bayesian networks [35, 43, 53, 59, 87], demonstrates that there is a direct correspondence between the operators and dependencies used in both knowledge systems. More specifically, a distribution can be viewed as a probabilistic relation. It can then be explicitly shown that the *projection* and *natural join* operators in relational databases are intrinsic components in the definition of the *marginalization* and *multiplication* operations in Bayesian networks, respectively. Furthermore, the only dependency used in Bayesian networks, namely, *probabilistic conditional independence*, can be expressed as *Bayesian embedded multivalued dependency* (BEMVD) in our model. In sacrificing the embedded conditional independencies, a Markov network can be seen as a joint distribution satisfying a *Bayesian acyclic join dependency* (BAJD). That is, the representation of probabilistic knowledge in practice (a Markov network) is a general form of the desired representation of data in practice (an AJD). Consequently, it can be explicitly demonstrated [96] how probabilistic query processing can be performed in existing relational database management systems using SQL. Thus, our model serves as a single framework for the knowledge representation in both knowledge systems.

The usefulness of our main argument is further emphasized by demonstrating the many exciting theoretical possibilities a Bayesian database model offers. A unified model provides the opportunity for cross-research. In this study, we examine the fundamental problem of testing the implication of dependencies, and the recent attempts to *generalize* [13, 33, 34, 49, 101] the standard Bayesian network model with more general dependencies and models.

The Implication Problem

The *implication problem* [35, 36, 37, 65, 68, 78, 79, 88, 98] is to test whether a given input set **C** of probabilistic conditional independencies (BEMVDs) logically implies another probabilistic conditional independency **c**. We say **C** *logically implies* **c** if whenever any distribution that satisfies all the conditional independencies in **C**, then the distribution also satisfies **c**. It has been pointed out [79] that the implication problems for the *general* classes of EMVD and BEMVD do not coincide. Consequently, it is suggested [79] that the Bayesian network model is different from the relational database model. This remark, however, does not take into consideration one important issue, namely, the *solvability* of the implication problem. In this comprehensive study of the implication problem for probabilistic conditional independence, we show that the implication problem for BEMVD is not solvable. This is a stronger result than the absence of a finite complete axiomatization for probabilistic conditional independencies [79]: not only is there no complete axiomatization, but *no* method exists to finitely decide the implication problem for probabilistic conditional independence. Furthermore, we also present a *nonaxiomatic* method for testing the implication of a special class of BEMVD, namely, *nonembedded* probabilistic conditional independence (BMVD). Most importantly of all, our results suggest that the implication problems in the Bayesian network model and the relational database model coincide on *solvable* classes of dependencies as illustrated in Figure 1.1. While being part of a model, *unsolvable* classes of dependencies have no practical value. That is, both models are confined to using solvable classes of dependencies in the design and implementation of knowledge systems in practice. These observations suggest that there is no real difference between the relational database model and the Bayesian network model in practice.

Recently, attempts have been made to *generalize* the standard Bayesian network model using: (i) *contextual* dependencies [13, 33, 34], (ii) an *object-oriented Bayesian*

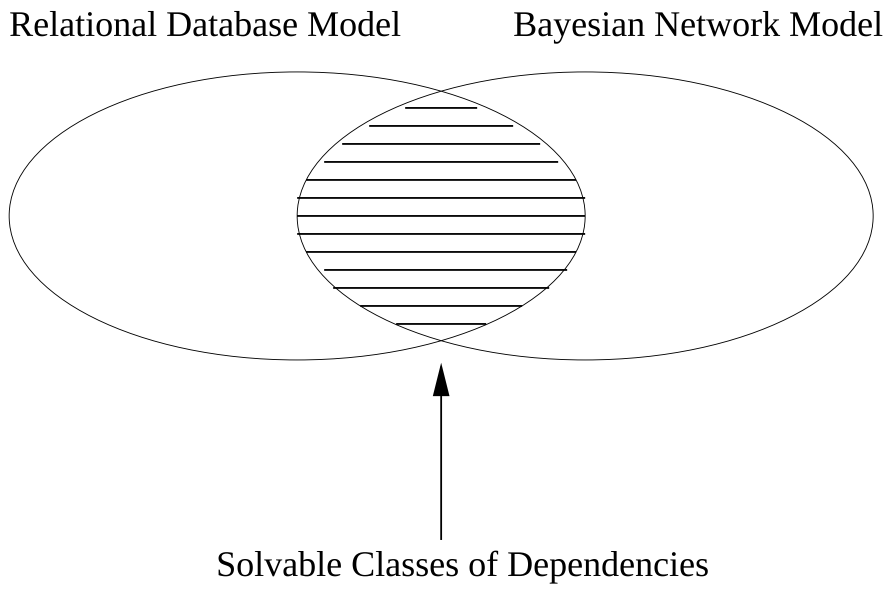

Figure 1.1: The implication problems in the relational database model and the Bayesian network model coincide on the *solvable* classes of dependencies.

network model [49], and (iii) a *distributed* Bayesian network model [101]. Part (i) follows the attempt in database theory to develop more general data dependencies. Parts (ii) and (iii) parallel the above mentioned generalizations of the standard relational database model. The usefulness of our Bayesian database model, as well as support for our main argument, is evident by our ability to quickly extend and formalize these initial generalizations of the standard Bayesian network model.

More General Dependencies

Bayesian networks are defined using only a *single* type of dependency, namely, BEMVD. *Contextual* dependencies [13, 33, 34], such as *context-specific independence* (CSI) [13], were introduced as a more general dependency than BEMVD. That is, contextual independencies can be used to capture independencies that remain undetected by BEMVD. In this exposition, we introduce a more general contextual dependency than CSI [94]. In fact, we introduce a more general *noncontextual* dependency than

BEMVD, called *weak* BEMVD (WBEMVD). *Complete* axiomatizations are stated for the special class of WBMVD alone as well as the mixture of WBMVD and BMVD.

Complex-Value Bayesian Networks

Another attempt to generalize Bayesian networks is the so-called *object-oriented Bayesian network* [49]. Essentially the extension is to allow for the *refining* or *coarsening* of parts of a Bayesian network. That is, in modeling a large Bayesian network, one is allowed to focus in or out on parts of the network depending on whether they are of interest or not. These capabilities correspond to the complex-relational database model [1, 2]. Here we suggest a complex-probabilistic-relational database model in which the value of an attribute is allowed to be a distribution itself [94]. We introduce a *nest* operator to coarsen the network and an *unnest* to refine the network. It is demonstrated how *inconsistency* may arise in performing inference in complex-value probabilistic networks. More importantly, we demonstrate that WBMVD is a necessary and sufficient condition for ensuring *consistency*.

Multi-Agent Bayesian Networks

The notion of *distributed* Bayesian networks, called *multi-agent* Bayesian networks [101], was recently suggested to generalize the standard Bayesian network model. It is useful to partition the discussion of distributed Bayesian networks into two parts: (i) constructing the dependency structure, and (ii) performing probabilistic inference. For part (i), it is not realistic to expect the domain experts to manually construct the dependency structure since the problem domain may be much larger and perhaps distributed. Furthermore, it is not entirely clear how those learning methods [44, 45, 66, 97, 99] developed for the single-agent environment can be applied, let alone how one can obtain a reliable sample. Here we suggest a more robust algorithm for constructing the dependency structure of a multi-agent probabilistic network [90, 92]. For part (ii), the innovative model proposed in [101] exhibits several undesirable characteristics. Consequently, we propose asynchronous control protocols and a method for processing evidence to ensure global consistency at all times. We then introduce the notion of a *distributed* query. Techniques are developed to reduce the amount of data transmitted between agents in answering distributed requests.

The important point to realize in the above discussion is that our Bayesian database model facilitates cross-research between the Bayesian network model and the relational database model. As this thesis explicitly demonstrates, several significant contributions can be made by taking advantage of the many highly developed database models. On the other hand, ignorance of these database models may lead to duplication of results and slowed development.

This thesis is organized into ten chapters. Chapter 2 presents the traditional relational database model. In Chapter 3, we review the two kinds of probabilistic network used for uncertainty management. In Chapter 4, we introduce our Bayesian database model. It is useful in Chapter 5 to have a thorough discussion on the relationship between the operators and dependencies used in the above models. Chapter 6 discusses the use of more general dependencies in Bayesian networks. Chapter 7 investigates a complex-value Bayesian network model. The implication problem is studied in Chapter 8. Chapters 9 examines multi-agent Bayesian networks. Conclusions and possible future work are discussed in Chapter 10.

Chapter 2

THE RELATIONAL DATABASE MODEL

In this chapter, we review the elegant relational database model [56] with an emphasis on data dependencies. The concepts presented in this chapter form the foundation for the discussion in this thesis.

2.1 Relations and Relation Schemes

A *relation scheme* R is a finite set of *attributes* (attribute names) $R = \{A_1, A_2, \ldots, A_m\}$. Corresponding to each attribute A_i is a nonempty finite set D_i, $1 \leq i \leq m$, called the *domain* of A_i. Let $D = D_1 \cup D_2 \ldots \cup D_m$. A *relation* r on the relation scheme R, written $r(R)$, is a finite set of mappings $\{t_1, t_2, \ldots, t_s\}$ from R to D with the restriction that for each mapping $t \in r$, $t(A_i)$ must be in D_i, $1 \leq i \leq k$, where $t(A_i)$ denotes the value obtained by restricting the mapping t to A_i. A relation r on $R = \{A_1, A_2, \ldots, A_m\}$ is depicted in Figure 2.1. The mappings are called *tuples* and $t(A)$ is called the A-value of t. We use $t(X)$ in the obvious way and call it the X-value of t.

Mappings are used in our exposition to avoid any explicit ordering of the attributes in the relation scheme. To simplify the notation, however, we will henceforth denote relations by writing the attributes in a certain order and the tuples as lists of values in the same order. Furthermore, the following relational database conventions will be

$$r = \begin{array}{|cccc|}\hline A_1 & A_2 & \dots & A_m \\ \hline t_1(A_1) & t_1(A_2) & \dots & t_1(A_m) \\ t_2(A_1) & t_2(A_2) & \dots & t_2(A_m) \\ \vdots & \vdots & \vdots & \vdots \\ t_s(A_1) & t_s(A_2) & \dots & t_s(A_m) \\ \hline \end{array}$$

Figure 2.1: A relation r on the scheme $R = \{A_1, A_2, \dots, A_m\}$.

$$r(ABC) = \begin{array}{|ccc|}\hline A & B & C \\ \hline 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ \hline \end{array}$$

Figure 2.2: A relation r on the scheme $R = ABC$.

adopted for simplified notation. Uppercase letters A, B, C from the beginning of the alphabet may be used to denote attributes. A relation scheme $R = \{A_1, A_2, \dots, A_m\}$ may be written as simply $A_1A_2 \dots A_m$. A relation r on scheme R is then written as either $r(R)$ or $r(A_1A_2 \dots A_m)$. The singleton set $\{A\}$ is sometimes written as A and concatenation XY may be used to denote set union $X \cup Y$.

Example 2.1 A relation r on $R = ABC$ is shown in Figure 2.2, where $D_A = D_B = D_C = \{0, 1\}$.

2.2 Relational Operators

In this section, we introduce relational operators to manipulate relations. The select, project, and natural join operators form the class of conjunctive queries. As will be shown in the next section, probabilistic inference in belief networks can be seen as the processing of conjunctive queries.

$$\sigma_{C=0}(r) = \begin{array}{|ccc|} \hline A & B & C \\ \hline 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline \end{array}$$

Figure 2.3: The relation $\sigma_{C=0}(r)$, where $r(ABC)$ is the relation in Figure 2.2.

2.2.1 The Select Operator

When the *select* operator σ is applied to a relation r, it yields another relation that is a subset of tuples of r with a certain value on a specified attribute.

Let r be a relation on scheme R, $A \in R$, and $a \in D_A$. Then

$$\sigma_{A=a}(r) = \{t \mid t \in r \text{ and } t(A) = a\}.$$

Example 2.2 Let $r(ABC)$ be the relation in Figure 2.2. The relation $\sigma_{C=0}(r)$ is depicted in Figure 2.3.

2.2.2 The Project Operator

Whereas the select operator chooses a subset of tuples in a relation, the *project* operator π chooses a subset of attributes.

Let r be a relation on R and X a subset of R. The *projection of* r *onto* X, written $\pi_X(r)$, is defined as

$$\pi_X(r) = \{\ t(X) \mid t \in r\ \}. \tag{2.1}$$

That is, $\pi_X(r)$ is the set of all tuples $t(X)$ such that t is in r.

Example 2.3 Let $r(ABC)$ be the relation in Figure 2.2. The projection $\pi_{AB}(r)$ of r onto AB is the relation illustrated in Figure 2.4.

$$\pi_{AB}(r) = \begin{array}{|cc|} \hline A & B \\ \hline 0 & 0 \\ 1 & 1 \\ \hline \end{array}$$

Figure 2.4: The projection $\pi_{AB}(r)$ of relation r in Figure 2.2 onto AB.

$$\begin{array}{|cc|} \hline A & B \\ \hline 1 & 1 \\ 2 & 1 \\ 1 & 2 \\ \hline \end{array} \bowtie \begin{array}{|cc|} \hline B & C \\ \hline 1 & 1 \\ 1 & 2 \\ 3 & 1 \\ \hline \end{array} = \begin{array}{|ccc|} \hline A & B & C \\ \hline 1 & 1 & 1 \\ 1 & 1 & 2 \\ 2 & 1 & 1 \\ 2 & 1 & 2 \\ \hline \end{array}$$

Figure 2.5: The natural join $r(AB) \bowtie r(BC)$ of relations $r(AB)$ and $r(BC)$.

2.2.3 The Natural Join Operator

Natural join $\bowtie$ is a binary operator for combining two relations[1] .

The *natural join* of two relations $r_1(X)$ and $r_2(Y)$, written $r_1(X) \bowtie r_2(Y)$, is defined as

$$r_1(X) \bowtie r_2(Y) = \{ \ t(XY) \mid t(X) \in r_1(X) \text{ and } t(Y) \in r_2(Y) \ \}. \qquad (2.2)$$

That is, $r_1(X) \bowtie r_2(Y)$ denotes the set of tuples $t(XY)$ such that $t(X)$ is in r_1 and $t(Y)$ is in r_2.

Example 2.4 The natural join $r(AB) \bowtie r(BC)$ of relations $r(AB)$ and $r(BC)$ is shown in Figure 2.5.

Let $r_1(R_1)$, $r_2(R_2)$,...,$r_n(R_n)$ be relations with $R = R_1 \cup R_2 \cup \ldots \cup R_n$. Let $t_1, t_2, \ldots, t_n$ be a sequence of tuples (not necessarily distinct) with $t_i \in r_i$, $1 \leq i \leq n$. We say tuples $t_1, t_2, \ldots, t_n$ are *joinable on* the sequence $R_1, R_2, \ldots, R_n$ if there is a tuple t on R such that $t_i = t(R)$, $1 \leq i \leq n$. Tuple t is the *result* of joining $t_1, t_2, \ldots, t_n$ on $R_1, R_2, \ldots, R_n$.

[1] Relational database theory includes several variations of the natural join operator. These other join operators are not required in this discussion.

2.3 Data Dependencies

Data dependencies were extensively studied since they played a key role in database scheme design[2] . A *data dependency* is a restriction on the permissible sets of data that are allowed to appear in the database. Such constraints are useful in attenuating data redundancy and avoiding update anomalies. In particular, a data dependency can be used to provide an economical representation of a relation using projections of that relation. In this section, we examine three fundamental data dependencies in relational database theory, namely, functional dependency, multivalued dependency, and acyclic join dependency.

2.3.1 Functional Dependencies

Let $r(R)$ be a relation and $X, Y \subseteq R$. We say that the *functional dependency* (FD) $X \rightarrow Y$ is satisfied by $r(R)$ if every two tuples of $r(R)$ which have the same projection on X also have the same projection on Y. That is, the FD $X \rightarrow Y$ is satisfied by $r(R)$ if and only if each X-value in $r(R)$ is associated with precisely one Y-value.

The FD $X \rightarrow Y$ is a *sufficient* condition for $r(R)$ to be losslessly decomposed as

$$r(R) = \pi_{XY}(r) \bowtie \pi_{XZ}(r). \tag{2.3}$$

Example 2.5 Consider again the relation $r(ABC)$ in Figure 2.2. The FD $B \rightarrow A$ is satisfied by r. By equation (2.3), relation $r(ABC)$ can then be losslessly decomposed into two relations $\pi_{AB}(r)$ and $\pi_{BC}(r)$ as shown in Figure 2.6. That is, $r(ABC) = \pi_{AB}(r) \bowtie \pi_{BC}(r)$.

2.3.2 Multivalued Dependencies

In Section 2.3.1, we saw that the presence of a functional dependency in a relation scheme means that the scheme can be decomposed to eliminate redundancy while

[2] Comprehensive reviews of this work can be found in [29, 48, 84].

$$
\begin{array}{|ccc|}
\hline
A & B & C \\
\hline
0 & 0 & 0 \\
0 & 0 & 1 \\
1 & 1 & 1 \\
\hline
\end{array}
=
\begin{array}{|cc|}
\hline
A & B \\
\hline
0 & 0 \\
1 & 1 \\
\hline
\end{array}
\bowtie
\begin{array}{|cc|}
\hline
B & C \\
\hline
0 & 0 \\
0 & 1 \\
1 & 1 \\
\hline
\end{array}
$$

Figure 2.6: Since the FD $B \rightarrow A$ is satisfied by r, relation $r(ABC)$ can be losslessly decomposed as $r(ABC) = \pi_{AB}(r) \bowtie \pi_{BC}(r)$.

preserving information. However, it is not necessary that an FD hold before such a decomposition may take place. Consider the relation $r(ABC)$ in Figure 2.7. Neither the functional dependency $B \rightarrow A$ nor $B \rightarrow C$ is satisfied by relation $r(ABC)$, yet $R(ABC)$ still losslessly decomposes as $r(R) = \pi_{AB}(r) \bowtie \pi_{BC}(r)$.

$$
\begin{array}{|ccc|}
\hline
A & B & C \\
\hline
0 & 0 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\hline
\end{array}
=
\begin{array}{|cc|}
\hline
A & B \\
\hline
0 & 0 \\
1 & 0 \\
1 & 1 \\
\hline
\end{array}
\bowtie
\begin{array}{|cc|}
\hline
B & C \\
\hline
0 & 0 \\
0 & 1 \\
1 & 1 \\
\hline
\end{array}
$$

Figure 2.7: Neither the functional dependency $B \rightarrow A$ or $B \rightarrow C$ is satisfied by relation $r(ABC)$, yet $R(ABC)$ still losslessly decomposes as $r(R) = \pi_{AB}(r) \bowtie \pi_{BC}(r)$.

Let R be a relation scheme, X and Y be subsets of R, and $Z = R - XY$. A relation $r(R)$ satisfies the *multivalued dependency* (MVD) $X \rightarrow\rightarrow Y$ if, for any two tuples t_1 and t_2 in r with $t_1(X) = t_2(X)$, there exists a tuple t_3 in r with

$$t_3(XY) = t_1(XY) \text{ and } t_3(Z) = t_2(Z). \tag{2.4}$$

The MVD $X \rightarrow\rightarrow Y$ is a *necessary* and *sufficient* condition for $r(R)$ to be losslessly decomposed as

$$r(R) = \pi_{XY}(r) \bowtie \pi_{XZ}(r). \tag{2.5}$$

$$
\begin{array}{|ccc|}\hline A & B & C \\ \hline 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ \hline \end{array}
\neq
\begin{array}{|cc|}\hline A & B \\ \hline 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ \hline \end{array}
\bowtie
\begin{array}{|cc|}\hline B & C \\ \hline 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ \hline \end{array}
=
\begin{array}{|ccc|}\hline A & B & C \\ \hline 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ \hline \end{array}
$$

Figure 2.8: Relation $r(R)$ does not satisfy the MVD $B \rightarrow\rightarrow A$.

Example 2.6 Equation (2.5) indicates that the relation $r(ABC)$ in Figure 2.7 satisfies the MVD $B \rightarrow\rightarrow A$ since $r(R) = \pi_{AB}(r) \bowtie \pi_{BC}(r)$. On the other hand, the relation $r(ABC)$ in Figure 2.8 does not satisfy the MVD $B \rightarrow\rightarrow A$ since $r(R) \neq \pi_{AB}(r) \bowtie \pi_{BC}(r)$.

2.3.3 Acyclic Join Dependencies

Acyclic join dependencies can be used to losslessly decompose a relation into more than two projections.

Let $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ be an *acyclic hypergraph*[3] on the set of attributes $R = R_1 \cup R_2 \cup \ldots \cup R_n$. We say that a relation $r(R)$ satisfies the *acyclic join dependency* (AJD) $\bowtie \{R_1, R_2, \ldots, R_n\}$ if

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_n}(r), \tag{2.6}$$

that is, if r decomposes losslessly onto $\mathcal{R}$. We also write $\bowtie \{R_1, R_2, \ldots, R_n\}$ as $\bowtie \mathcal{R}$. It follows that a relation r satisfies the acyclic join dependency $\bowtie \{R_1, R_2, \ldots, R_n\}$ if and only if r contains each tuple t for which there are tuples $t_1, t_2, \ldots, t_n$ of r (not necessarily distinct) such that $t_i(R_i) = t(R_i)$ for each i $(1 \leq i \leq n)$.

Example 2.7 Relation $r(R)$ in Figure 2.9 satisfies the AJD $\bowtie \mathcal{R}$, where $\mathcal{R} = \{R_1, R_2, R_3, R_4\}$ is the acyclic hypergraph depicted in Figure 3.2. That is,

$$r = \pi_{A_1A_2A_3}(r) \bowtie \pi_{A_2A_3A_4}(r) \bowtie \pi_{A_2A_3A_5}(r) \bowtie \pi_{A_5A_6}(r).$$

[3]See Section A.2 in Appendix A.

A_1	A_2	A_3	A_4	A_5	A_6
0	0	0	0	1	0
0	0	0	1	1	0
1	0	1	1	0	1
1	1	0	1	1	0

Figure 2.9: The relation $r(R)$ satisfies the AJD $\bowtie \mathcal{R}$, where $\mathcal{R} = \{\ R_1 = A_1A_2A_3,$ $R_2 = A_2A_3A_4,\ R_3 = A_2A_3A_5,\ R_4 = A_5A_6\}$ is the acyclic hypergraph depicted in Figure 3.2.

AJDs are one class of *join dependency*. Join dependencies in general, however, may exhibit undesirable characteristics [3]. A culminating result by Beeri, Fagin, Maier and Yannakakis [9] was that the special class of AJDs possess a number of desirable properties in database applications.

It should be clear that multivalued dependency is a special case of acyclic join dependency, namely, when $n = 2$. That is, saying that a relation $r(XYZ)$ satisfies the AJD $\bowtie \{XY, XZ\}$ is the same as saying that relation r satisfies the MVD $X \rightarrow\rightarrow Y$.

2.3.4 Embedded Multivalued Dependencies

As with conditional independence in Bayesian networks, it is important to distinguish between *embedded* and *nonembedded* MVDs. That is, a given relation r may not satisfy a MVD, yet a projection of r does.

Example 2.8 Consider the relation $r(ABCD)$ in Figure 2.10. It is easily verified that $r(ABCD)$ does not satisfy the MVD $B \rightarrow\rightarrow A$:

$$r(ABCD) \neq \pi_{AB}(r) \bowtie \pi_{BCD}(r).$$

However, the projection $\pi_{ABC}(r)$ does satisfy the MVD $B \rightarrow\rightarrow A$:

$$r(ABC) = \pi_{AB}(r) \bowtie \pi_{BC}(r).$$

A	B	C	D
0	0	0	0
0	0	1	1
1	0	0	0
1	0	1	0
1	1	1	1

Figure 2.10: The relation $r(ABCD)$ does not satisfy the MVD $B \rightarrow\rightarrow A$. However, the projection $\pi_{ABC}(r)$ does.

Example 2.8 indicates that MVDs are sensitive to context. It follows that the specification of the set of attributes is an integral part of the MVD [8]. It would then be more appropriate to use the notation $X \rightarrow\rightarrow Y \mid Z$ to stress the fact that the MVD involves the set of attributes XYZ.

We say relation $r(R)$ satisfies the *embedded multivalued dependency* (EMVD) $X \rightarrow\rightarrow Y \mid Z$ if $\pi_{XYZ}(r)$ satisfies the MVD $X \rightarrow\rightarrow Y$, that is,

$$\pi_{XYZ}(r) \quad = \quad \pi_{XY}(r) \bowtie \pi_{XZ}(r). \tag{2.7}$$

For notational convenience, however, we will continue to write the *nonembedded* MVD $X \rightarrow\rightarrow Y \mid (R - XY)$ as $X \rightarrow\rightarrow Y$ with the context R understood.

2.4 Summary

In this section, we reviewed the standard relational database model [56] with an emphasis on dependencies. The reader is referred to [29, 48, 84] for more recent work on dependencies. A well rounded discussion on some of the other more general database models can be found in [2].

Chapter 3

PROBABILISTIC MODELS

In this section, we introduce two frameworks for the representation of probabilistic knowledge, namely, Bayesian networks and Markov networks.

3.1 Bayesian Networks

Let $R = \{A_1, A_2, \ldots, A_k\}$ denote a finite set of discrete variables. Each variable A_i is associated with a finite domain D_i. Let D be the Cartesian product of the domains D_i, $1 \leq i \leq k$. A *joint probability distribution* (jpd) [38, 66] on D is a function p on D, assigning to each tuple $t \in D$ a real number $0 \leq p(t) \leq 1$ such that $\sum_{t \in D} p(t) = 1$. In general, a *potential* [38] is a function q on D such that $q(t)$ is a nonnegative real number and $\sum_{t \in D} q(t)$ is positive, i.e., at least one $q(t) > 0$. Each potential q can be transformed to a joint probability distribution p through *normalization*, that is, by setting $p(t) = q(t)/\sum_{v \in D} q(v)$. We say the distribution p is on R, and sometimes write p as $p(R)$, if the domain D is understood.

We say Y and Z are *conditionally independent* given X under p, denoted $I_p(Y, X, Z)$, if

$$p(y \mid x, z) \quad = \quad p(y \mid x), \tag{3.8}$$

whenever $p(x, z) > 0$. This conditional independency $I_p(Y, X, Z)$ can be equivalently written as

$$p(y, x, z) \quad = \quad \frac{p(y, x) \cdot p(x, z)}{p(x)}. \tag{3.9}$$

We write $I_p(Y, X, Z)$ as $I(Y, X, Z)$ if the joint probability distribution p is understood. In the special case where $Y \cup X \cup Z = R$, we call the probabilistic conditional independence $I(Y, X, Z)$ *nonembedded*; otherwise $I(Y, X, Z)$ is called *embedded*.

Example 3.1 Let p be a joint probability distribution over the set of variables R, where X, Y, Z and W are binary variables in R. Consider the conditional probability distribution shown in Table 3.1. By definition (3.8), variables X and ZW are conditionally independent given Y.

X	Y	Z	W	$p(X\|Y, Z, W)$
0	0	0	0	p_1
0	0	0	1	p_1
0	0	1	0	p_1
0	0	1	1	p_1
1	0	0	0	p_2
1	0	0	1	p_2
1	0	1	0	p_2
1	0	1	1	p_2
0	1	0	0	p_3
0	1	0	1	p_3
0	1	1	0	p_3
0	1	1	1	p_3
1	1	0	0	p_3
1	1	0	1	p_3
1	1	1	0	p_3
1	1	1	1	p_3

Table 3.1: Variables X and ZW are *conditionally independent* given Y.

By the chain rule, a joint probability distribution p over $R = \{A_1, A_2, \ldots, A_m\}$ can always be written as:

$$p(R) = p(A_1) \cdot p(A_2|A_1) \cdot p(A_3|A_1, A_2) \cdot \ldots \cdot p(A_m|A_1, A_2, \ldots, A_{m-1}). \qquad (3.10)$$

The above equation is an identity. However, one can use conditional independencies that are assumed to hold in the problem domain to obtain a simpler representation of a joint distribution. For example, consider a joint distribution $p(A_1, A_2, A_3, A_4, A_5, A_6)$. By equation (3.10), the joint distribution $p(R)$ can be written as

$$\begin{aligned} p(R) &= p(A_1) \cdot p(A_2|A_1) \cdot p(A_3|A_1, A_2) \cdot p(A_4|A_1, A_2, A_3) \cdot p(A_5|A_1, A_2, \ldots, A_4) \\ & \quad \cdot p(A_6|A_1, A_2, \ldots, A_5). \end{aligned} \tag{3.11}$$

Consider the following set $\mathbf{C}$ of conditional independencies:

$$\begin{aligned} p(A_3|A_1, A_2) &= p(A_3|A_1), \\ p(A_4|A_1, A_2, A_3) &= p(A_4|A_2, A_3), \\ p(A_5|A_1, A_2, A_3, A_4) &= p(A_5|A_2, A_3), \\ p(A_6|A_1, A_2, A_3, A_4, A_5) &= p(A_6|A_5), \end{aligned}$$

namely,

$$\begin{aligned} \mathbf{C} = \{ & I(A_3, A_1, A_2),\ I(A_4, A_2A_3, A_1),\ I(A_5, A_2A_3, A_1A_4), \\ & I(A_6, A_5, A_1A_2A_3A_4)\}. \end{aligned} \tag{3.12}$$

Utilizing these conditional independencies, the joint distribution $p(R)$ written using the chain rule in equation (3.11) can be expressed in a simpler form, namely:

$$p(R) = p(A_1) \cdot p(A_2|A_1) \cdot p(A_3|A_1) \cdot p(A_4|A_2, A_3) \cdot p(A_5|A_2, A_3) \cdot p(A_6|A_5). \tag{3.13}$$

We can represent the dependency structure of this joint distribution by a DAG as shown in Figure 3.1. This DAG, together with the conditional probability tables $p(A_1)$, $p(A_2|A_1)$, $p(A_3|A_1)$, $p(A_4|A_2, A_3)$, $p(A_5|A_2, A_3)$, and $p(A_6|A_5)$, define a Bayesian network. Such a network provides an economical representation of a joint probability distribution.

3.2 Markov Networks

Even though Bayesian networks provide an economical representation of a joint probability distribution, it may still be difficult to compute marginal distributions. Thus, several efficient local computation algorithms [47, 51, 76] were developed for

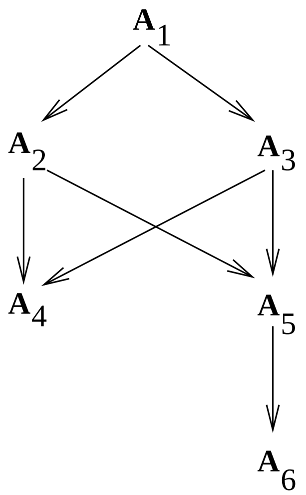

Figure 3.1: The directed acyclic graph reflecting the conditional independencies defined in equation (3.13).

computing marginal distributions in *Markov networks* [38]. It should be noted that a Markov network defined by Hajek, Havranek and Jirousek [38] is called a *decomposable* Markov network by Pearl [66]. That is, the definition of Markov network in [66] is different from the one used here.

To facilitate probabilistic inference, it is useful to transform a Bayesian network into a Markov network. The DAG representing the dependency structure of a Bayesian network can be converted by the moralization and triangulation procedures [38, 66] into an acyclic hypergraph. (An acyclic hypergraph in fact represents a chordal undirected graph. Each maximal clique in the graph corresponds to a hyperedge in the acyclic hypergraph.) For example, by applying these procedures to the DAG in Figure 3.1, we obtain the acyclic hypergraph depicted in Figure 3.2. Such an acyclic hypergraph represents the dependency structure of a Markov network. The joint probability distribution defined by equation (3.13) can be rewritten in terms of marginal distributions as:

$$p(R) \quad = \quad \frac{p(A_1, A_2, A_3) \cdot p(A_2, A_3, A_4) \cdot p(A_2, A_3, A_5) \cdot p(A_5, A_6)}{p(A_2, A_3) \cdot p(A_2, A_3) \cdot p(A_5)}. \qquad (3.14)$$

A Bayesian network is more expressive than a Markov network. Consider the

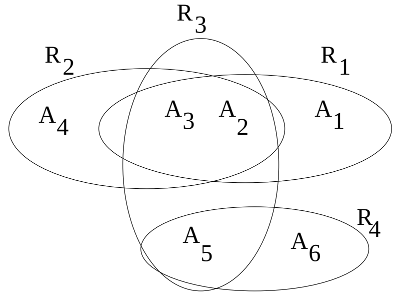

Figure 3.2: A graphical representation of the acyclic hypergraph $\mathcal{R} = \{ R_1 = A_1A_2A_3,\ R_2 = A_2A_3A_4,\ R_3 = A_2A_3A_5,\ R_4 = A_5A_6\}$.

marginal distribution $p(A_1, A_2, A_3)$ obtained from the Bayesian network in equation (3.13):

$$
\begin{aligned}
& p(A_1, A_2, A_3) \\
= & \sum_{A_4, A_5, A_6} p(A_1, A_2, A_3, A_4, A_5, A_6) \\
= & \sum_{A_4, A_5, A_6} p(A_1) \cdot p(A_2|A_1) \cdot p(A_3|A_1) \cdot p(A_4|A_2, A_3) \cdot p(A_5|A_2, A_3) \cdot p(A_6|A_5) \\
= & \ p(A_1) \cdot p(A_2|A_1) \cdot p(A_3|A_1) \\
= & \ \frac{p(A_1, A_2) \cdot p(A_1, A_3)}{p(A_1)}.
\end{aligned}
$$

Thus, the *embedded* probabilistic conditional independence $p(A_3|A_1, A_2) = p(A_3|A_1)$ is reflected by the Bayesian network in equation (3.13). However, this same independency is *not* reflected by the Markov network in equation (3.14):

$$
\begin{aligned}
p(A_1, A_2, A_3) &= \sum_{A_4, A_5, A_6} p(A_1, A_2, A_3, A_4, A_5, A_6) \\
&= \frac{p(A_1, A_2, A_3) \cdot p(A_2, A_3, A_4) \cdot p(A_2, A_3, A_5) \cdot p(A_5, A_6)}{p(A_2, A_3) \cdot p(A_2, A_3) \cdot p(A_5)} \\
&= \frac{p(A_1, A_2, A_3) \cdot p(A_2, A_3) \cdot p(A_2, A_3)}{p(A_2, A_3) \cdot p(A_2, A_3)} \\
&= p(A_1, A_2, A_3).
\end{aligned}
$$

3.3 Summary

In this section, we examined two types of probabilistic networks, namely, Bayesian and Markov. A Bayesian network is defined by a DAG and corresponding conditional probability distributions. On the other hand, a Markov network is defined by an acyclic hypergraph and corresponding marginal distributions. Bayesian networks facilitate the acquisition of probabilistic knowledge, while Markov networks facilitate the inference process in practice.

Chapter 4

A BAYESIAN DATABASE MODEL

In this section, we review our Bayesian database model [89, 96] with an emphasis on probabilistic dependencies.

4.1 Probabilistic Relations

A joint probability distribution can be represented as a relation. The relation $\mathbf{r}$ representing the jpd p has attributes $R \cup \{A_p\}$, where the column labelled by A_p stores the probability value. For example, the relation $\mathbf{r}$ representing the jpd p on the set of variables $R = \{A_1, A_2, \ldots, A_m\}$ is shown in Figure 4.1. Each tuple $\mathbf{t} \in \mathbf{r}$ is defined by $\mathbf{t}(R) = t \in D$ and $\mathbf{t}(A_p) = p(t)$. That is, $\mathbf{t} = \{t, p(t)\}$. For convenience we will say relation $\mathbf{r}$ is on R with the attribute A_p understood by context. That is, relations denoted by boldface represent probability distributions.

$$\mathbf{r} \;=\; \begin{array}{|ccccc|}
\hline
A_1 & A_2 & \ldots & A_m & A_p \\
\hline
\mathbf{t}_1(A_1) & \mathbf{t}_1(A_2) & \ldots & \mathbf{t}_1(A_m) & \mathbf{t}_1(A_p) = p(t_1) \\
\mathbf{t}_2(A_1) & \mathbf{t}_2(A_2) & \ldots & \mathbf{t}_2(A_m) & \mathbf{t}_2(A_p) = p(t_2) \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{t}_s(A_1) & \mathbf{t}_s(A_2) & \ldots & \mathbf{t}_s(A_m) & \mathbf{t}_s(A_p) = p(t_s) \\
\hline
\end{array}$$

Figure 4.1: A joint distribution p expressed as a relation $\mathbf{r}$.

$$\mathbf{r}(A_1A_2A_3) \;=\; \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_p \\ \hline 0 & 0 & 0 & 0.1 \\ 0 & 0 & 1 & 0.6 \\ 1 & 0 & 0 & 0.3 \\ \hline \end{array}$$

Figure 4.2: A relation $\mathbf{r}$ on the scheme $R = A_1A_2A_3$.

$$\sigma_{A_3=0}(\mathbf{r}) \;=\; \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_p \\ \hline 0 & 0 & 0 & 0.1 \\ 1 & 1 & 0 & 0.3 \\ \hline \end{array}$$

Figure 4.3: The relation $\sigma_{A_3=0}(\mathbf{r})$, where $r(A_1A_2A_3)$ is the relation in Figure 4.2.

4.2 Probabilistic Relational Operators

In this section, we introduce operators to manipulate the probabilistic relations.

4.2.1 The Select Operator

The *select* operator σ works exactly the same on probabilistic relations as it does on standard relations.

Let $\mathbf{r}$ be a relation on scheme R, $A \in R$, and $a \in D_A$. Then

$$\sigma_{A=a}(\mathbf{r}) \;=\; \{\mathbf{t} \mid \mathbf{t} \in r \text{ and } \mathbf{t}(A) = a\}.$$

Whenever evidence is collected, the distribution is refined to reflect only those tuples which agree with the evidence.

Example 4.1 Let $\mathbf{r}(A_1A_2A_3)$ be the relation in Figure 4.2. The relation $\sigma_{A_3=0}(r)$ is depicted in Figure 4.3.

$$\tau_{A_1A_2}(\mathbf{r}) \;=\; \begin{array}{|ccc|}\hline A_1 & A_2 & A_{p(A_1A_2)} \\ \hline 0 & 0 & 0.7 \\ 1 & 0 & 0.3 \\ \hline \end{array}$$

Figure 4.4: The marginalization $\tau_{A_1A_2}(\mathbf{r})$ of relation $\mathbf{r}$ in Figure 4.2 onto A_1A_2.

4.2.2 The Marginalization Operator

Let $\mathbf{r}$ be a relation on R and X be a subset of R. The *marginalization of* $\mathbf{r}$ *onto* X, written $\tau_X(\mathbf{r})$, is defined as

$$\tau_X(\mathbf{r}) \;=\; \{\ \mathbf{t}(XA_{p(X)})\ \mid\ \mathbf{t}(X) \in \pi_X(\mathbf{r}) \text{ and } \mathbf{t}(A_{p(X)}) = \sum_{\mathbf{t}' \in \mathbf{r}, \mathbf{t}'(X) = \mathbf{t}(X)} \mathbf{t}'(A_p)\ \} \tag{4.15}$$

That is, $\tau_X(\mathbf{r})$ represents a probability distribution on the traditional projection $\pi_X(\mathbf{r}) = \{\ \mathbf{t} \mid \mathbf{t} \in \mathbf{r}\ \}$ of $\mathbf{r}$ onto X. In the literature [38, 66], the relation $\tau_X(\mathbf{r})$ is called the *marginal distribution* $p(X)$ of $p(R)$ onto X.

Example 4.2 Given the relation $\mathbf{r}(A_1A_2A_3)$ in Figure 4.2, the marginalization of $\mathbf{r}$ onto A_1A_2 is the relation $\tau_{A_1A_2}(\mathbf{r})$ depicted in Figure 4.4. □

4.2.3 The Product Join Operator

The *product join* of two relations $\mathbf{r}_1(X)$ and $\mathbf{r}_2(Y)$, written $\mathbf{r}_1(X) \times \mathbf{r}_2(Y)$, is defined as

$$\begin{aligned} & \mathbf{r}_1(X) \times \mathbf{r}_2(Y) \\ = \;& \{\mathbf{t}(XYA_{p(X)\cdot p(Y)}) \mid \mathbf{t}(XY) \in \pi_X(\mathbf{r}_1) \bowtie \pi_Y(\mathbf{r}_2) \text{ and } \mathbf{t}(A_{p(X)\cdot p(Y)}) = \mathbf{t}(A_{p(X)}) \cdot \mathbf{t}(A_{p(Y)})\}. \end{aligned} \tag{4.16}$$

Thus, $\mathbf{r}_1(X) \times \mathbf{r}_2(Y)$ denotes the product $p(X) \cdot p(Y)$ of the two distributions $p(X)$ and $p(Y)$.

Example 4.3 The product join $\mathbf{r}(A_1A_2) \bowtie \mathbf{r}(A_2A_3)$ of relations $\mathbf{r}(A_1A_2)$ and $\mathbf{r}(A_2A_3)$ is shown in Figure 4.5. □

A_1	A_2	$A_{p(A_1A_2)}$
1	1	0.2
2	1	0.4
1	2	0.4

$\times$

A_2	A_3	$A_{p(A_2A_3)}$
1	1	0.2
1	2	0.5
3	1	0.3

$=$

A_1	A_2	A_3	$A_{p(A_1A_2)\cdot p(A_2A_3)}$
1	1	1	0.04
1	1	2	0.10
2	1	1	0.08
2	1	2	0.20

Figure 4.5: The product join $\mathbf{r}(A_1A_2) \bowtie \mathbf{r}(A_2A_3)$ of relations $\mathbf{r}(A_1A_2)$ and $\mathbf{r}(A_2A_3)$.

4.2.4 The Markov Join Operator

The Markov Join operator is introduced for shorthand notation. We first need the following preliminary definition.

Let $\mathbf{r}(R)$ be a joint distribution and $\tau_X(\mathbf{r})$ the marginalization onto the subset of attributes X. The *reciprocal* relation $\tau_X(\mathbf{r})^{-1}$ is defined as

$$\begin{aligned} &\tau_X(\mathbf{r})^{-1} \\ = \; &\{\mathbf{t}(XA_{1/p(X)}) \mid \mathbf{t}(X) \in \pi_X(\mathbf{r}) \text{ and } \mathbf{t}(A_{1/p(X)}) = \begin{cases} 1/\mathbf{t}(A_{p(X)}) & \text{if } \mathbf{t}(A_{p(X)}) > 0, \\ \mathbf{t}(A_{p(X)}) & \text{otherwise} \end{cases} \}. \end{aligned}$$

That is, the reciprocal relation contains the reciprocal of the positive probability values in $\tau_X(\mathbf{r})$. We now define the Markov join operator.

Let $\mathbf{r}(XY)$ and $\mathbf{r}(XZ)$ be two *consistent* relations, that is, $\tau_X(\mathbf{r}(XY)) = \tau_X(\mathbf{r}(XZ))$. The *Markov join* $\tau_{XY}(\mathbf{r}) \otimes \tau_{XZ}(\mathbf{r})$ of relations $\mathbf{r}(XY)$ and $\mathbf{r}(XZ)$ is defined as

$$\mathbf{r}(XY) \otimes \mathbf{r}(XZ) \;=\; \mathbf{r}(XY) \times \mathbf{r}(XZ) \times \tau_X(\mathbf{r}(XY))^{-1}. \tag{4.17}$$

Example 4.4 The Markov join $\mathbf{r}(A_1A_2) \otimes \mathbf{r}(A_2A_3)$ of the two consistent relations $\mathbf{r}(A_1A_2)$ and $\mathbf{r}(A_2A_3)$ is shown in Figure 4.6. □

4.3 Probabilistic Dependencies

Probabilistic dependencies are introduced since they can be used to provide an economical representation of a probabilistic relation using marginalizations of that

A_1	A_2	$A_{p(A_1A_2)}$
0	0	0.3
0	1	0.3
1	0	0.4

$\otimes$

A_2	A_3	$A_{p(A_2A_3)}$
0	0	0.4
0	1	0.3
1	1	0.3

$=$

A_1	A_2	A_3	$A_{\frac{p(A_1A_2)\cdot p(A_2A_3)}{p(A_2)}}$
0	0	0	0.1714285
0	0	1	0.1285714
0	1	1	0.3000000
1	0	0	0.2285714
1	0	1	0.1714285

Figure 4.6: The Markov join $\mathbf{r}(A_1A_2) \otimes \mathbf{r}(A_2A_3)$ of relations $\mathbf{r}(A_1A_2)$ and $\mathbf{r}(A_2A_3)$.

relation. In this section, we examine three fundamental probabilistic dependencies, namely, Bayesian multivalued dependency, Bayesian acyclic join dependency, and Bayesian embedded multivalued dependency.

4.3.1 Bayesian Multivalued Dependencies

The important notion of Bayesian MVD is now introduced. Let R be a relation scheme, X and Y be subsets of R, and $Z = R - XY$. A relation $\mathbf{r}(R)$ satisfies the *Bayesian multivalued dependency* (BMVD) $X \Rightarrow\Rightarrow Y$ if and only if

$$\begin{aligned} \mathbf{r}(R) &= \tau_{XY}(\mathbf{r}) \times \tau_{XZ}(\mathbf{r}) \times \tau_X(\mathbf{r})^{-1} \\ &= \tau_{XY}(\mathbf{r}) \otimes \tau_{XZ}(\mathbf{r}). \end{aligned} \tag{4.18}$$

Thus, the BMVD $X \Rightarrow\Rightarrow Y$ is a necessary and sufficient condition for a relation $\mathbf{r}(R)$ to be losslessly decomposed as $\mathbf{r}(R) = \tau_{XY}(\mathbf{r}) \otimes \tau_{XZ}(\mathbf{r})$.

Example 4.5 Equation (4.18) indicates that the relation $\mathbf{r}(A_1A_2A_3)$ in Figure 4.7 satisfies the BMVD $A_2 \Rightarrow\Rightarrow A_1$ since

$$\mathbf{r}(R) = \tau_{A_1A_2}(\mathbf{r}) \otimes \tau_{A_2A_3}(\mathbf{r}).$$

On the other hand, the two relations $\mathbf{r}(A_1A_2A_3)$ in Figures 4.8 (left, right) do not satisfy the BMVD $A_2 \Rightarrow\Rightarrow A_1$ since

$$\mathbf{r}(R) \quad \neq \quad \tau_{A_1A_2}(\mathbf{r}) \otimes \tau_{A_2A_3}(\mathbf{r}),$$

as shown in Figures 4.6 and 4.9, respectively. $\Box$

$\mathbf{r}$ =

A_1	A_2	A_3	A_p
0	0	0	0.3
0	0	1	0.3
0	1	1	0.2
1	0	0	0.1
1	0	1	0.1

=

A_1	A_2	$A_{p(A_1,A_2)}$
0	0	0.6
0	1	0.2
1	0	0.2

$\otimes$

A_2	A_3	$A_{p(A_2,A_3)}$
0	0	0.4
0	1	0.4
1	1	0.2

Figure 4.7: Relation $\mathbf{r}(A_1A_2A_3)$ satisfies the BMVD $A_2 \Rightarrow\Rightarrow A_1$.

A_1	A_2	A_3	A_p
0	0	0	0.1
0	0	1	0.2
0	1	1	0.3
1	0	0	0.3
1	0	1	0.1

A_1	A_2	A_3	A_p
0	0	0	0.1
0	0	1	0.2
1	0	0	0.3
1	1	1	0.4

Figure 4.8: Relation $\mathbf{r}(A_1A_2A_3)$ (left) does not satisfy the BMVD $A_2 \Rightarrow\Rightarrow A_1$ as shown in Figure 4.6. Relation $\mathbf{r}(A_1A_2A_3)$ (right) does not satisfy the BMVD $A_2 \Rightarrow\Rightarrow A_1$ as shown in Figure 4.9.

A_1	A_2	$A_{p(A_1A_2)}$
0	0	0.3
1	0	0.3
1	1	0.4

$\otimes$

A_2	A_3	$A_{p(A_2A_3)}$
0	0	0.4
0	1	0.2
1	1	0.4

=

A_1	A_2	A_3	$A_{\frac{p(A_1A_2)\cdot p(A_2A_3)}{p(A_2)}}$
0	0	0	0.2
0	0	1	0.1
1	0	0	0.2
1	0	1	0.1
1	1	1	0.4

Figure 4.9: Relation $\mathbf{r}(A_1A_2A_3)$ does not satisfy the BMVD $A_2 \Rightarrow\Rightarrow A_1$.

4.3.2 Bayesian Acyclic Join Dependencies

BMVD is a special case of a more general kind of data dependency, called probabilistic acyclic join dependency. Let $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ be an acyclic hypergraph on the set of attributes $R = R_1 \cup R_2 \cup \ldots \cup R_n$. We say a *Bayesian acyclic join dependency* (BAJD) $\otimes\mathcal{R}$ is satisfied by a relation $\mathbf{r}$, if

$$\mathbf{r}(R) = (\ldots((\tau_{R_1}(\mathbf{r}) \otimes \tau_{R_2}(\mathbf{r})) \otimes \tau_{R_3}(\mathbf{r}))\ldots) \otimes \tau_{R_n}(\mathbf{r}), \tag{4.19}$$

where the sequence $R_1, R_2, \ldots, R_n$ is a hypertree construction ordering for $\mathcal{R}$.

4.3.3 Bayesian Embedded Multivalued Dependencies

As with dependencies in both relational databases and Bayesian networks, it is important to distinguish between *embedded* and *nonembedded* dependencies in our Bayesian database model. As expected, a given relation $\mathbf{r}$ may not satisfy a BMVD, yet a marginalization of $\mathbf{r}$ does.

Example 4.6 Consider the relation $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 4.10. It can be verified that $\mathbf{r}(A_1A_2A_3A_4)$ does not satisfy the BMVD $A_2 \Rightarrow\Rightarrow A_1$:

$$\mathbf{r}(A_1A_2A_3A_4) \neq \tau_{A_1A_2}(r) \otimes \tau_{A_2A_3A_4}(\mathbf{r}).$$

However, the marginalization $\tau_{A_1A_2A_3}(\mathbf{r})$ does satisfy the BMVD $A_2 \Rightarrow\Rightarrow A_1$:

$$\tau_{A_1A_2A_3}(\mathbf{r}) = \tau_{A_1A_2}(r) \otimes \tau_{A_2A_3A_4}(\mathbf{r})$$

as shown in Figure 4.7. □

Example 4.6 indicates that BMVDs are sensitive to context. It follows that the specification of the set of attributes is an integral part of the BMVD. It would then be more appropriate to use the notation $X \Rightarrow\Rightarrow Y \mid Z$ to stress the fact that the BMVD involves the set of attributes XYZ.

We say relation $\mathbf{r}(R)$ satisfies the *Bayesian embedded multivalued dependency* (BEMVD) $X \Rightarrow\Rightarrow Y \mid Z$ if $\tau_{XYZ}(\mathbf{r})$ satisfies the BMVD $X \Rightarrow\Rightarrow Y$, that is,

$$\tau_{XYZ}(\mathbf{r}) \quad = \quad \tau_{XY}(\mathbf{r}) \otimes \tau_{XZ}(\mathbf{r}). \tag{4.20}$$

A_1	A_2	A_3	A_4	A_p
0	0	0	0	0.3
0	0	1	1	0.3
1	0	0	0	0.1
1	0	1	0	0.1
0	1	1	1	0.2

Figure 4.10: The relation $\mathbf{r}(A_1A_2A_3A_4)$ does not satisfy the BMVD $A_2 \Rightarrow\Rightarrow A_1$. However, the marginalization $\tau_{A_1A_2A_3}(\mathbf{r})$ does as shown in Figure 4.7.

For notational convenience, however, we will continue to write the *nonembedded* BMVD $X \Rightarrow\Rightarrow Y \mid (R - XY)$ as $X \Rightarrow\Rightarrow Y$ with the context R understood.

It is important to realize that BEMVD is simply probabilistic conditional independence expressed as a database dependency. Saying that the generalized relation $\mathbf{r}(R)$ satisfies the BEMVD $X \Rightarrow\Rightarrow Y|Z$ in equation (4.20) is equivalent to stating that Y and Z are conditionally independent given X in equation (3.9). We will then use the terms BEMVD and probabilistic conditional independence interchangeably.

4.4 Summary

It should be made clear that the Bayesian database model proposed here is quite different from other probabilistic relational databases [6, 26, 70]. The most important distinction is that our model is *not* confined to a probabilistic interpretation. Furthermore, the focus here is not on missing values [26] or missing probability values [6]. Whereas it is suggested in [70] that the database scheme need not reflect the dependency structure of a Bayesian/Markov network, here we show the exact opposite: the dependency structure of the probabilistic network precisely defines the database scheme.

Chapter 5

COMPARISON OF THE THREE DATA MODELS

In this section we provide a brief exposition on the relationship between the traditional relational database model and our Bayesian database model. Our goal is to demonstrate that the main difference between these two models is the choice of operators. Furthermore, the traditional relational operators are intrinsic components of the corresponding probabilistic operators. This realization will pave the way to adopt relational database techniques for probabilistic problems, as shown in subsequent sections.

5.1 Relations

Given a probability distribution $\mathbf{r}(R)$, the set of tuples with $p(t) > 0$ will be called the *support* [59] of $\mathbf{r}(R)$. For example, the traditional relation $r(A_1A_2A_3)$ in Figure 5.1 (left) is the support of the relation $\mathbf{r}(A_1A_2A_3)$ in Figure 5.1 (right).

5.2 Relational Operators

In this section, we draw a direct correspondence between the project and marginalization operators, as well as the natural join and product join operators.

$$r(A_1A_2A_3) = \begin{array}{|ccc|} \hline A_1 & A_2 & A_3 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ \hline \end{array} \qquad \mathbf{r}(A_1A_2A_3) = \begin{array}{|cccc|} \hline A_1 & A_2 & A_3 & A_p \\ \hline 0 & 0 & 0 & 0.1 \\ 0 & 0 & 1 & 0.6 \\ 1 & 0 & 0 & 0.3 \\ \hline \end{array}$$

Figure 5.1: The traditional relation $r(A_1A_2A_3)$ (left) is the support of the probability distribution $\mathbf{r}(A_1A_2A_3)$ (right).

$$\pi_{A_1A_2}(r) = \begin{array}{|cc|} \hline A_1 & A_2 \\ \hline 0 & 0 \\ 1 & 0 \\ \hline \end{array} \qquad \tau_{A_1A_2}(\mathbf{r}) = \begin{array}{|ccc|} \hline A_1 & A_2 & A_{p(A_1A_2)} \\ \hline 0 & 0 & 0.7 \\ 1 & 0 & 0.3 \\ \hline \end{array}$$

Figure 5.2: Given the relations $r(A_1A_2A_3)$ and $\mathbf{r}(A_1A_2A_3)$ in Figure 5.1, the projection $\pi_{A_1A_2}(r)$ (left) and the marginalization $\tau_{A_1A_2}(\mathbf{r})$ (right) onto A_1A_2, respectively.

5.2.1 The Project and Marginalization Operators

Given a relation, a set of attributes R, in choosing a subset of R, the traditional model applies the *projection* π operator, whereas the probabilistic model applies the *marginalization* τ operator. The project operator π ignores duplicate tuples, while the marginalization operator counts them. For example, given the relations $r(A_1A_2A_3)$ and $\mathbf{r}(A_1A_2A_3)$ in Figure 5.1, the projection $\pi_{A_1A_2}(r)$ (left) and the marginalization $\tau_{A_1A_2}(\mathbf{r})$ (right) onto A_1A_2 are illustrated in Figure 5.2, respectively.

5.2.2 The Natural Join and Product Join Operators

The traditional model uses the *natural join* operator $\bowtie$ to combine two relations, while the probabilistic model uses the *product join* operator $\times$ to combine two relations. In fact, the product join operator defines a distribution precisely over the tuples in the natural join [96]. For example, Figure 5.3 illustrates the natural join $r(A_1A_2) \bowtie r(A_2A_3)$ of relations $r(A_1A_2)$ and $r(A_2A_3)$ (top), and the product join $\mathbf{r}(A_1A_2) \bowtie \mathbf{r}(A_2A_3)$ of relations $\mathbf{r}(A_1A_2)$ and $\mathbf{r}(A_2A_3)$ (bottom).

A_1	A_2
1	1
2	1
1	2

$\bowtie$

A_2	A_3
1	1
1	2
3	1

$=$

A_1	A_2	A_3
1	1	1
1	1	2
2	1	1
2	1	2

A_1	A_2	$A_{p(A_1A_2)}$
1	1	0.2
2	1	0.4
1	2	0.4

$\times$

A_2	A_3	$A_{p(A_2A_3)}$
1	1	0.2
1	2	0.5
3	1	0.3

$=$

A_1	A_2	A_3	$A_{p(A_1A_2)\cdot p(A_2A_3)}$
1	1	1	0.04
1	1	2	0.10
2	1	1	0.08
2	1	2	0.20

Figure 5.3: The natural join $r(A_1A_2) \bowtie r(A_2A_3)$ of relations $r(A_1A_2)$ and $r(A_2A_3)$ (top). The product join $\mathbf{r}(A_1A_2) \bowtie \mathbf{r}(A_2A_3)$ of relations $\mathbf{r}(A_1A_2)$ and $\mathbf{r}(A_2A_3)$ (bottom).

5.3 Multivalued Dependency and Bayesian Multivalued Dependency

Consider the dependencies used to losslessly decompose relations into two projections (marginals). A traditional relation $r(R)$ is losslessly decomposed as

$$r(R) = \pi_{XY}(r) \bowtie \pi_{XZ}(r)$$

if and only if the MVD $X \rightarrow\rightarrow Y$ holds. On the other hand, a probabilistic relation $\mathbf{r}(R)$ is losslessly decomposed as

$$\mathbf{r}(R) = \tau_{XY}(\mathbf{r}) \otimes \tau_{XZ}(\mathbf{r})$$

if and only if the BMVD $X \Rightarrow\Rightarrow Y$ holds.

5.4 Acyclic Join Dependency and Bayesian Acyclic Join Dependency

In this section, we demonstrate a subtle difference in denoting the corresponding notions of AJD and BAJD.

It is well-known in relational database theory, that the natural join operator $\bowtie$ is both *commutative* and *associative* [56]. This means that we can write an unparenthesized string of joins without ambiguity. The Markov join is commutative by definition. However, the following example demonstrates that the Markov join operator $\otimes$ is not associative.

Example 5.1 Consider the acyclic hypergraph $\mathcal{R} = \{R_1 = ABC, R_2 = ABD, R_3 = ACE, R_4 = BCF\ \}$, where R_1, R_2, R_3, R_4 is a hypertree construction ordering and R_2, R_3, R_4, R_1 is not. Consider the relation $\mathbf{r}(ABCDEF)$ in Figure 5.4 and its support $r(R)$. As expected,

$$r \;=\; (\ (\pi_{R_1}(r) \bowtie \pi_{R_2}(r)) \bowtie \pi_{R_3}(r)\ ) \bowtie \pi_{R_4}(r) \tag{5.21}$$

and

$$r \;=\; (\ (\pi_{R_2}(r) \bowtie \pi_{R_3}(r)) \bowtie \pi_{R_4}(r)\ ) \bowtie \pi_{R_1}(r). \tag{5.22}$$

It can also be verified that

$$\begin{aligned} \mathbf{r} &= (\ (\tau_{R_1}(\mathbf{r}) \otimes \tau_{R_2}(\mathbf{r})) \otimes \tau_{R_3}(\mathbf{r})\ ) \otimes \tau_{R_4}(\mathbf{r}) \\ &= \tau_{R_1}(\mathbf{r}) \times \tau_{R_2}(\mathbf{r}) \times \tau_{AB}(\mathbf{r})^{-1} \times \tau_{R_3}(\mathbf{r}) \times \tau_{AC}(\mathbf{r})^{-1} \tau_{R_4}(\mathbf{r}) \times \tau_{BC}(\mathbf{r})^{-1}. \end{aligned} \tag{5.23}$$

However, as shown in Figure 5.5,

$$\begin{aligned} \mathbf{r} &\neq (\ (\tau_{R_2}(\mathbf{r}) \otimes \tau_{R_3}(\mathbf{r})) \otimes \tau_{R_4}(\mathbf{r})\ ) \otimes \tau_{R_1}(\mathbf{r}) \\ &= \tau_{R_2}(\mathbf{r}) \times \tau_{R_3}(\mathbf{r}) \times \tau_{A}(\mathbf{r})^{-1} \times \tau_{R_4}(\mathbf{r}) \times \tau_{BC}(\mathbf{r})^{-1} \times \tau_{R_1}(\mathbf{r}) \times \times \tau_{ABC}(\mathbf{r})^{-1}. \end{aligned} \tag{5.24}$$

Example 5.1 indicates that parentheses are required in defining a BAJD. However, it is also well known that a join path must be followed in an AJD to guarantee that intermediate joins are correct projections of the universal relation [9].

Example 5.2 Consider again the relation $\mathbf{r}(R)$ in Figure 5.4, its support $r(R)$, and the acyclic hypergraph $\mathcal{R} = \{R_1, R_2, R_3, R_4\}$. Even though $r(R)$ satisfies the AJD

A	B	C	D	E	F	A_p
0	0	0	0	0	0	0.1
0	0	1	0	1	1	0.2
0	1	0	1	0	2	0.3
1	0	0	2	2	0	0.4

Figure 5.4: A relation $\mathbf{r}(ABCDEF)$.

A	B	C	D	E	F	A_p
0	0	0	0	0	0	0.22222
0	0	1	0	1	1	0.11111
0	1	0	1	0	2	0.22222
1	0	0	2	2	0	0.44444

Figure 5.5: The relation $(\ (\tau_{R_2}(\mathbf{r}) \otimes \tau_{R_3}(\mathbf{r})) \otimes \tau_{R_4}(\mathbf{r})\ ) \otimes \tau_{R_1}(\mathbf{r})$, where $\mathbf{r}$ is the relation in Figure 5.4.

$\bowtie \mathcal{R}$, the intermediate join $\pi_{R_2}(r) \bowtie \pi_{R_3}(r)$ is not a correct projection of r onto the attributes $R_2 \cup R_3$ as shown in Figure 5.6. That is,

$$\begin{aligned} \pi_{R_2}(r) \bowtie \pi_{R_3}(r) &= \pi_{ABD}(r) \bowtie \pi_{ACE}(r) \\ &\neq \pi_{ABCDE}(r). \end{aligned} \tag{5.25}$$

Example 5.2 indicates that a join path in a join tree must be followed to guarantee intermediate joins are correct. In other words, it is meaningful to include parentheses in the definition of an AJD. Therefore, it would be appropriate to include parentheses in the definition of both AJDs and PAJDs. However, we will henceforth omit the parentheses in writing both dependencies and assume that a join path is always followed. This concludes our discussion on the terminology used in the three models. We summarize the main points of this section in Table 5.1.

A	B	C	D	E
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	2	2

Figure 5.6: Even though the support $r(R)$ of the relation $\mathbf{r}(R)$ in Figure 5.4 satisfies the AJD $\bowtie \{R_1, R_2, R_3, R_4\}$, the intermediate join $\pi_{R_2}(r) \bowtie \pi_{R_3}(r)$ is not a correct projection of r onto the attributes $R_2 \cup R_3$.

Relational Database	Bayesian Network	Bayesian Database
relation	distribution	relation
$r(R)$	$p(R)$	$\mathbf{r}(R)$
projection	marginal	marginal
$\pi_X(r)$	$p(X)$	$\tau_X(\mathbf{r})$
natural join	multiplication	product join
$\bowtie$	$\cdot$	$\times$
EMVD	conditional independence	BEMVD
$X \rightarrow\rightarrow Y \vert Z$	$p(Y \vert X, Z) = p(Y \vert X)$	$X \Rightarrow\Rightarrow Y \vert Z$
AJD	Markov Network	BAJD
$R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$	$\frac{p(R_1) \cdot p(R_2) \cdot \ldots \cdot p(R_n)}{p(R_1 \cap R_2) \cdot \ldots \cdot p(R_{n-1} \cap R_n)}$	$\tau_{R_1}(\mathbf{r}) \otimes \tau_{R_2}(\mathbf{r}) \otimes \ldots \otimes \tau_{R_n}(\mathbf{r})$

Table 5.1: The terminology used for corresponding notions in the traditional relational database model, Bayesian networks, and our Bayesian database model.

5.5 In Uniform Distributions

We are now able to directly contrast the fundamental notions of BMVD and MVD. We first consider the special case where the joint probability distribution p is *uniform* (constant). That is, the probability value for every configuration is either a constant or zero, i.e., $p(t) = k$ or $p(t) = 0$, for all tuples t.

Theorem 5.1 [59, 89] Let $\mathbf{r}(R)$ be a *uniform* distribution and $r(R)$ its support. Then relation $\mathbf{r}(R)$ satisfies the BMVD $X \Rightarrow\Rightarrow Y$ if and only if its support $r(R)$ satisfies the MVD $X \rightarrow\rightarrow Y$.

For a *uniform* distribution, Theorem 5.1 indicates that

$$\mathbf{r}(R) = \tau_{XY}(\mathbf{r}) \otimes \tau_{XZ}(\mathbf{r})$$

if and only if

$$r(R) = \pi_{XY}(r) \bowtie \pi_{XZ}(r).$$

In other words, in a uniform distribution, the MVD $X \rightarrow\rightarrow Y$ implies the BMVD $X \Rightarrow\Rightarrow Y$, and the BMVD $X \Rightarrow\Rightarrow Y$ implies the MVD $X \rightarrow\rightarrow Y$.

5.6 In Nonuniform Distributions

In a *nonuniform* distribution, however, MVD does not imply BMVD.

Example 5.3 Consider the relation $\mathbf{r}(A_1A_2A_3)$ in Figure 4.8 (left). It can be verified that the support $r(A_1A_2A_3)$ of $\mathbf{r}$ satisfies the MVD $A_2 \rightarrow\rightarrow A_1$. That is,

$$r(A_1A_2A_3) \quad = \quad \pi_{A_1A_2}(r) \bowtie \pi_{A_2A_3}(r)$$

as shown in Figure 2.7. However, the relation $\mathbf{r}(A_1A_2A_3)$ does not satisfy the BMVD $A_2 \Rightarrow\Rightarrow A_1$ as shown in Figure 4.6. $\Box$

On the other hand, Wen [87] demonstrated that BMVD implies MVD. In fact, Lee [53] summarized the above by showing that MVD is a necessary but not sufficient condition for BMVD. These remarks are formalized in our notation with the following result.

Theorem 5.2 [53, 87, 89] For any relation $\mathbf{r}(R)$ satisfying a set $\mathbf{C}$ of BMVDs, the support $r(R)$ must satisfy the corresponding set $C = \{X \rightarrow\rightarrow Y \mid X \Rightarrow\Rightarrow Y \in \mathbf{C}\}$ of MVDs.

We illustrate Theorem 5.2 with an example.

Example 5.4 As shown in Example 4.5, the relation $\mathbf{r}(A_1A_2A_3)$ in Figure 4.7 satisfies the BMVD $A_2 \Rightarrow\Rightarrow A_1$. It can be verified that the traditional relation $r(A_1A_2A_3)$ satisfies the MVD $A_2 \rightarrow\rightarrow A_1$, namely, $r(A_1A_2A_3) = \pi_{A_1A_2}(r) \bowtie \pi_{A_2A_3}(r)$, as required by Theorem 5.2. □

5.7 Functional Dependency and Probabilistic Multivalued Dependency

In the last section, we saw that past research has shown multivalued dependency is a necessary but not sufficient condition for conditional independence. In this section, we show that functional dependency is a sufficient but not necessary condition for probabilistic conditional independence.

Theorem 5.3 [15] Let $\mathbf{r}(R)$ be a joint probability distribution. Let $X, Y \subseteq R$ and $Z = R - XY$. Let $r(R)$ be the support of $\mathbf{r}(R)$. The *functional dependency* $X \rightarrow Y$ being satisfied by $r(R)$ is a sufficient but not necessary condition for the BMVD $X \Rightarrow\Rightarrow Y$ to be satisfied by $\mathbf{r}(R)$.

Proof: We first show that FD is sufficient for BMVD. Suppose the FD $X \rightarrow Y$ is satisfied by the support $r(R)$. This means that every X-value is associated with precisely one Y-value. In other words, if $t_i(X) = t_j(X)$, then $t_i(Y) = t_j(Y)$, $t_i, t_j \in \mathbf{r}(R)$. Since $XYZ = R$, it immediately follows that there are no duplicate XZ-values in $r(R)$, namely, $t_i(XZ) \neq t_j(XZ)$ for all $t_i, t_j \in r(R)$. Consider any tuple $t = < x\ y\ z >$ in $r(R)$. The probability value $p(x)$ of the X-value x in the marginal distribution $\tau_X(\mathbf{r})$ is defined by

$$p(x) = \sum_{t'(Y), t'(Z)} p(x\ t'(Y)\ t'(Z)). \tag{5.26}$$

By the initial assumption, equation (5.26) can be rewritten as

$$p(x) = \sum_{t'(Z)} p(x\ y\ t'(Z)) = p(xy). \tag{5.27}$$

Using equation (5.27) we obtain

$$\frac{p(xy) \cdot p(xz)}{p(x)} = p(xz).$$

However, $p(xz) = p(xyz)$ since there are no duplicate XZ-values in $r(R)$. By definition, Y and Z are conditionally independent given X under p. That is, the BMVD $X \Rightarrow\Rightarrow Y$ is satisfied by $\mathbf{r}$.

We now show that FD is not a necessary condition for BMVD. The BMVD $A_2 \Rightarrow\Rightarrow A_1$ is satisfied by the generalized relation $\mathbf{r}(A_1A_2A_3)$ as shown in Figure 4.7. However, no nontrivial FDs are satisfied by $\mathbf{r}(R)$. $\Box$

Theorem 5.3 indicates that functional dependency is a sufficient but not necessary condition for probabilistic conditional independence. In other words, the FD $X \rightarrow Y$ implies the BMVD $X \Rightarrow\Rightarrow Y$. However, the BMVD $X \Rightarrow\Rightarrow Y$ does not imply the FD $X \rightarrow Y$.

5.8 Summary

There are two cosmetic differences that tend to confuse the relationship between these two models. First, constraints in the relational database model have traditionally been called *dependencies*. On the other hand, constraints in the Bayesian network model are called *independencies*. The important point, however, is that dependencies and independencies serve the same purpose in both models. A second source of confusion stems from the context of the dependency. The relational database model tends to focus on *nonembedded* dependencies. It is then customary to speak of a dependency with the nonembedded context understood. That is, it will be emphasized when a discussion involves *embedded* dependencies. On the contrary, the Bayesian network model focuses on *embedded* dependencies. Consequently, it will be emphasized in the probabilistic literature when the discussion involves *nonembedded* dependencies.

Many researchers have noticed similarities between the above two knowledge systems including conditional independence versus embedded multivalued dependency, probabilistic dependencies versus knowledge dependencies, and the use of DAGs to represent dependencies in both systems [25, 51, 69, 66]. This is perhaps best summarized by Lee [52] and more recently by Hill [43]. In their research Lauritzen and Spiegelhalter [51] discussed how to transform representations of a joint probability distribution. This discussion, on the other hand, transforms the probabilistic model into a probabilistic relational database model by maintaining all conditional independencies in order to represent the joint probability distribution. In [77], Shafer, Shenoy and Mellouli pointed out *qualitative* conditional independence is equivalent to embedded multivalued dependency in relational theory. We would like to emphasize that qualitative conditional independence is *not* probabilistic conditional independence. Dechter [25] also stated that conditional independence parallels the notion of embedded multivalued dependency. However, conditional independence as defined by Dechter is *not* probabilistic conditional independence. Shafer [77] commented that the work in relational theory has not brought out the analogy with probabilistic conditional independence. We believe this work does exactly that. Whereas Pearl and Verma [69] suggested that probabilistic conditional independence can be captured by the notion of embedded multivalued dependencies in relational databases, we have shown that embedded multivalued dependency is a necessary but not sufficient condition for probabilistic conditional independence. Finally, it should perhaps be mentioned here that *nonembedded* probabilistic conditional independencies can be characterized in many equivalent ways [91].

Chapter 6

MORE PROBABILISTIC DEPENDENCIES

In this chapter, we introduce dependencies that generalize probabilistic conditional independence. In particular, we study contextual and noncontextual dependencies. *Contextual* dependencies are dependencies which hold for some (but not necessarily all) tuples in a relation. *Noncontextual* dependencies are dependencies which hold for all the tuples in the relation. We then examine ways to represent these more general dependencies.

6.1 Contextual Dependencies

Two kinds of contextual dependencies are studied, namely, contextual strong dependencies and contextual weak dependencies. It is shown that contextual strong independence is a special case of contextual weak dependence.

6.1.1 Contextual Strong Dependencies

Many researchers including [13, 33, 34] have pointed out that the notion of probabilistic conditional independence (BEMVD) is too restrictive to capture independencies that hold in some but not necessarily all contexts. This contextual independency, referred to as *context-specific independence* (CSI) [13], *asymmetric independence* (ASI) [34], or *probabilistic causal irrelevance* (PCI) [33], can be used to facilitate the acquisition, representation, and inference of probabilistic knowledge.

A Bayesian network only reflects conditional independencies $p(x|y,z) = p(x|y)$ which hold for *all* $y \in D_Y$. In some situations, however, the conditional independence may only hold for certain *specific* values in D_Y. Geiger and Heckerman [34] studied the notion of *asymmetric independence* (ASI), which states that variables are independent for some but not necessarily for all of their values.

Example 6.1 Variables X and ZW are *not* conditionally independent given Y in the CPD depicted in Table 6.1, namely,

$$p(X|Y,Z,W) \quad \neq \quad p(X|Y).$$

For instance,

$$p(X=1|Y=1,Z=0,W=0) = p_3 \quad \neq \quad p(X=1|Y=1,Z=1,W=1) = p_4.$$

However, variables X and ZW are conditionally independent given the *context* $Y = 0$. That is,

$$p(X|Y=0,Z,W) \quad = \quad p(X|Y=0).$$

Example 6.1 indicates that in some situations the conditional independence may hold in *some* but not necessarily all values of Y.

Boutilier, Friedman, Goldszmidt, and Koller [13] formalized the notion of contextual independence with *context-specific independence* (CSI). Let X, Y, C, Z be pairwise disjoint sets of variables. We say X and Z are *contextually independent* given Y and the *fixed context* $C = c$, denoted $I(X \perp Z|Y, C = c)$, if:

$$p(X|Y,C=c,Z) = p(X|Y,C=c), \text{ whenever } p(Y,C=c,Z) > 0. \tag{6.28}$$

It should be clear from equation (6.28) that probabilistic conditional independence is a *special case* of CSI, namely, CSI becomes CI when the CSI holds for all $c \in V_C$. That is, if the context-specific independence of X and Z given Y and $C = c$ holds for all $c \in V_C$, then we simply say X and Z are conditionally independent given YC.

X	Y	Z	W	$p(X\|Y,Z,W)$
0	0	0	0	p_1
0	0	0	1	p_1
0	0	1	0	p_1
0	0	1	1	p_1
1	0	0	0	p_2
1	0	0	1	p_2
1	0	1	0	p_2
1	0	1	1	p_2
0	1	0	0	p_3
0	1	0	1	p_3
0	1	1	0	p_3
0	1	1	1	p_3
1	1	0	0	p_3
1	1	0	1	p_3
1	1	1	0	p_4
1	1	1	1	p_4

Table 6.1: Variables X and ZW are *not* conditionally independent given Y. However, variables X and ZW are conditionally independent given the *context* $Y = 0$, namely, $p(X|Y=0, Z, W) = p(X|Y=0)$.

In [33], Galles and Pearl studied the notion of *probabilistic causal irrelevance* (PCI). We say Z is *probabilistically causally irrelevant* to X given Y if for a *fixed* $y \in V_Y$, the following relationship holds:

$$p(X|Y=y, Z) = p(X|Y=y). \tag{6.29}$$

Equation (6.29) is read "Once we hold Y fixed at y, changing the value of Z will not change the probability of X". Thus, PCI tries to capture the same contextual independencies as defined by equation (6.28). It follows that if the causal irrelevance holds for all $y \in V_Y$, then X and Z are probabilistically conditionally independent given Y in the usual sense.

We now express the notions of ASI, CSI and PCI in terms of our probabilistic relational database model. These notions can be represented as *contextual Bayesian multivalued dependency* (CBMVD) in our model.

$$\mathbf{r}(A_1A_2A_3) \;=\; \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_p \\ \hline 0 & 0 & 0 & 0.2 \\ 0 & 0 & 1 & 0.2 \\ 1 & 0 & 0 & 0.2 \\ 1 & 0 & 1 & 0.2 \\ 0 & 1 & 0 & 0.1 \\ 1 & 1 & 1 & 0.1 \\ \hline \end{array}$$

Figure 6.1: Relation $\mathbf{r}(A_1A_2A_3)$ satisfies the *contextual* BMVD $A_2 = 0 \Rightarrow\Rightarrow A_1$.

Let R be a relation scheme, X and Y be subsets of R, and $Z = R - XY$. A relation $\mathbf{r}(R)$ satisfies the *contextual Bayesian multivalued dependency* (CBMVD)

$$(X = x) \Rightarrow\Rightarrow Y$$

if and only if

$$\sigma_{X=x}(\mathbf{r}) \;=\; \tau_{XY}(\sigma_{X=x}(\mathbf{r})) \otimes \tau_{XZ}(\sigma_{X=x}(\mathbf{r})). \tag{6.30}$$

We will write the CBMVD $(X = x) \Rightarrow\Rightarrow Y$ as $X = x \Rightarrow\Rightarrow Y$ to simplify the notation.

Example 6.2 Equation (6.30) indicates that the relation $\mathbf{r}(A_1A_2A_3)$ in Figure 6.1 satisfies the CBMVD $A_2 = 0 \Rightarrow\Rightarrow A_1$ since

$$\sigma_{A_2=0}(\mathbf{r}) = \tau_{A_1A_2}(\sigma_{A_2=0}(\mathbf{r})) \otimes \tau_{A_2A_3}(\sigma_{A_2=0}(\mathbf{r})), \tag{6.31}$$

as illustrated in Figure 6.2 (top). On the other hand, the relation $\mathbf{r}(A_1A_2A_3)$ does not satisfy the CBMVD $A_2 = 1 \Rightarrow\Rightarrow A_1$ since

$$\sigma_{A_2=1}(\mathbf{r}) \;\neq\; \tau_{A_1A_2}(\sigma_{A_2=1}(\mathbf{r})) \otimes \tau_{A_2A_3}(\sigma_{A_2=1}(\mathbf{r})), \tag{6.32}$$

as illustrated in Figure 6.2 (bottom).

Henceforth we will use the terms *contextual* BMVD (CBMVD) and the contextual dependencies ASI, CSI and PCI interchangeably.

$$
\sigma_{A_2=0}(\mathbf{r}) \;=\; \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_p \\ \hline 0 & 0 & 0 & 0.2 \\ 0 & 0 & 1 & 0.2 \\ 1 & 0 & 0 & 0.2 \\ 1 & 0 & 1 & 0.2 \\ \hline \end{array} \;=\; \begin{array}{|ccc|}\hline A_1 & A_2 & A_{p(A_1A_2)} \\ \hline 0 & 0 & 0.4 \\ 1 & 0 & 0.4 \\ \hline \end{array} \otimes \begin{array}{|ccc|}\hline A_2 & A_3 & A_{p(A_2A_3)} \\ \hline 0 & 0 & 0.4 \\ 0 & 1 & 0.4 \\ \hline \end{array}
$$

$$
\sigma_{A_2=1}(\mathbf{r}) \;=\; \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_p \\ \hline 0 & 1 & 0 & 0.1 \\ 1 & 1 & 1 & 0.1 \\ \hline \end{array} \neq \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_{\frac{p(A_1A_2)p(A_2A_3)}{p(A_2)}} \\ \hline 0 & 1 & 0 & 0.05 \\ 0 & 1 & 1 & 0.05 \\ 1 & 1 & 0 & 0.05 \\ 1 & 1 & 1 & 0.05 \\ \hline \end{array}
$$

Figure 6.2: By equation (6.31), the relation in Figure 6.1 satisfies the CBMVD $A_2 = 0 \Rightarrow\Rightarrow A_1$ (top). However, equation (6.32) indicates that $A_2 = 1 \Rightarrow\Rightarrow A_1$ is not satisfied (bottom).

Lemma 6.1 Let $\mathbf{r}(XYZ)$ be a relation, where X, Y, Z are pairwise disjoint. If $\mathbf{r}$ satisfies the BMVD $X \Rightarrow\Rightarrow Y$, then $\mathbf{r}$ also satisfies the CBMVD $X = x \Rightarrow\Rightarrow Y$, for any X-value x appearing in $\mathbf{r}$. However, if $\mathbf{r}$ satisfies the CBMVD $X = x \Rightarrow\Rightarrow Y$, for some X-value x appearing in $\mathbf{r}$, relation $\mathbf{r}$ does not necessarily satisfy the BMVD $X \Rightarrow\Rightarrow Y$.

Proof: Straightforward. □

It should be clear that the notion of CBMVD $X = x \Rightarrow\Rightarrow Y$ tries to capture an independence in which the partition is defined by $X = x$. (We refer to the CBMVD $X = x \Rightarrow\Rightarrow Y$ as a *strong* dependency because the partition of $\mathbf{r}$ is defined precisely by $X = x$.) In the next section, however, we introduce contextual *weak* BMVD in which the partition is *not* explicitly defined by the context $X = x$.

6.1.2 Contextual Weak Dependencies

In this section, we introduce the notion of *contextual weak Bayesian multivalued dependency* (CWBMVD). In short, the generalization to the CBMVD $X = x \Rightarrow\Rightarrow Y$

is to *further* partition the tuples with X-value x. We show that CBMVD is a special case of CWBMVD.

The notions of ASI, CSI and PCI are too strong to capture a weaker form of independence.

Example 6.3 Let p be a joint probability distribution over the set of variables R, where $X, Y, Z, W \in R$. Let $D_X = \{0,1,2\}$, $D_Y = \{0,1\}$ and $D_Z = D_W = \{0,1,2,3\}$. Consider the CPD shown in Table 6.2, where configurations c with $p(c) = 0$ are not shown. The important point is that there is *no* contextual independency between variables X and ZW defined precisely by the fixed context $Y = 0$ or $Y = 1$. For example, if $Y = 0$,then

$$p(X=0|Y=0, Z=0, W=0) = p_1 \neq p(X=0|Y=0, Z=0, W=3) = 0, \quad (6.33)$$

and if $Y = 1$, then

$$p(X=0|Y=1, Z=0, W=0) = p_5 \neq p(X=0|Y=1, Z=0, W=1) = 0. \quad (6.34)$$

Equations (6.33) and (6.34) indicate that X and ZW are not conditionally independent given $Y = 0$ or $Y = 1$.

Example 6.3 indicates that the notions of ASI, CSI, and PCI do *not* capture any independency between variables X and ZW. We now demonstrate a contextual *weak* independence between variables X and ZW given the fixed context $Y = 0$.

Example 6.4 Let us focus on the configurations where $Y = 0$ in Table 6.2, namely, the set of configurations $(x, Y = 0, z, w)$, namely, $\{t_1, t_2, \ldots, t_{12}\}$. Recall that the domain of variables X, Y, Z, and W are $D_X = \{0,1,2\}$, $D_Y = \{0\}$, and $D_Z = D_W = \{0,1,2,3\}$. We make a few observations. When $Y = 0$, only specific values $x \in D_X$ appear with specific values $z \in D_Z$ and $w \in D_W$. That is,

$$X = 0,1 \qquad \Longleftrightarrow \qquad Z = 0,1 \text{ and } W = 0,1,$$

and

$$X = 2 \qquad \Longleftrightarrow \qquad Z = 2,3 \text{ and } W = 2,3.$$

	X	Y	Z	W	$p(X\|Y,Z,W)$
t_1	0	0	0	0	p_1
t_2	0	0	0	1	p_1
t_3	0	0	1	0	p_1
t_4	0	0	1	1	p_1
t_5	1	0	0	0	p_2
t_6	1	0	0	1	p_2
t_7	1	0	1	0	p_2
t_8	1	0	1	1	p_2
t_9	2	0	2	2	p_3
t_{10}	2	0	2	3	p_3
t_{11}	2	0	3	2	p_4
t_{12}	2	0	3	3	p_4
t_{13}	0	1	0	0	p_5
t_{14}	1	1	0	2	p_6
t_{15}	2	1	1	3	p_7

Table 6.2: Variables X and ZW are *not* conditionally independent given $Y = 0$, nor given $Y = 1$.

In other words, when $X = 0$ then Z is never 2 or 3 and so forth. Based on this observation let us partition the CPD in Table 6.2 into three separate CPDs as shown in Table 6.3.

Consider the CPD in Table 6.3 (top). Let us define *new* domains D'_X, D'_Y, D'_Z, D'_W, based on the values that appear. We obtain

$$D'_X = \{0,1\}, \quad D'_Y = \{0\}, \quad D'_Z = \{0,1\}, \quad D'_W = \{0,1\}. \tag{6.35}$$

With respect to the new domains in equation (6.35), by definition variables X and ZW are conditionally independent given Y in the CPD in Table 6.3 (top).

Now consider the CPD in Table 6.3 (middle). Similarly, we define new domains D''_X, D''_Y, D''_Z, D''_W, for the variables based on the values that appear. Hence,

$$D''_X = \{2\}, \quad D''_Y = \{0\}, \quad D''_Z = \{2,3\}, \quad D''_W = \{2,3\}. \tag{6.36}$$

With respect to the domains in equation (6.36), however, variables X and ZW are

	X	Y	Z	W	$p(X\|Y,Z,W)$
t_1	0	0	0	0	p_1
t_2	0	0	0	1	p_1
t_3	0	0	1	0	p_1
t_4	0	0	1	1	p_1
t_5	1	0	0	0	p_2
t_6	1	0	0	1	p_2
t_7	1	0	1	0	p_2
t_8	1	0	1	1	p_2

	X	Y	Z	W	$p(X\|Y,Z,W)$
t_9	2	0	2	2	p_3
t_{10}	2	0	2	3	p_3
t_{11}	2	0	3	2	p_4
t_{12}	2	0	3	3	p_4

	X	Y	Z	W	$p(X\|Y,Z,W)$
t_{13}	0	1	0	0	p_5
t_{14}	1	1	0	2	p_6
t_{15}	2	1	1	3	p_7

Table 6.3: Partitioning the CPD in Table 6.2 into three CPDs corresponding to (top,middle,bottom), respectively.

not conditionally independent given Y since

$$p(X=2|Y=0,Z=2,W=2)=p_3 \neq p(X=2|Y=0,Z=3,W=3)=p_4.$$

Lastly, consider the CPD in Table 6.3 (bottom). As with the other contextual dependencies, variables X and ZW are *not* weakly independent given the fixed context $Y=1$.

Unlike ASI, CSI and PCI in Example 6.3, Example 6.4 indicates that the notion of *contextual weak independence* captures a more general independency between variables X and ZW given $Y=0$. We now formalize this idea in our probabilistic relational database model.

We begin by recalling some familiar notions about relations [72]. Given a distribution over the set of variables $R = XYZ$, let $\mathbf{r}(R)$ be its support (see Section 5). Given any subset $V \subseteq R$, we can define an equivalence relation $\theta(V)$ on $\mathbf{r}$: for all $\mathbf{t}_i, \mathbf{t}_j \in \mathbf{r}$,

$$\mathbf{t}_i \ \theta(V) \ \mathbf{t}_j, \quad \text{if } \mathbf{t}_i(V) = \mathbf{t}_j(V). \tag{6.37}$$

(Recall that $\mathbf{t}_k(V)$ denotes the value of V in the tuple $\mathbf{t}_k$.)

Example 6.5 Consider the relation $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 6.3. By equation (6.37),

$$\theta(A_1) \ = \ \{ \ \{\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_4\}, \ \{\mathbf{t}_5, \mathbf{t}_6, \mathbf{t}_7, \mathbf{t}_8\}, \ \{\mathbf{t}_9, \mathbf{t}_{10}, \mathbf{t}_{11}, \mathbf{t}_{12}\}, \ \{\mathbf{t}_{13}\}, \ \{\mathbf{t}_{14}\}, \ \{\mathbf{t}_{15}\} \ \}.$$

$\mathbf{r}(A_1A_2A_3A_4) =$

	A_1	A_2	A_3	A_4	A_p
$\mathbf{t}_1$	0	0	0	0	0.05
$\mathbf{t}_2$	0	0	0	1	0.05
$\mathbf{t}_3$	0	0	1	0	0.05
$\mathbf{t}_4$	0	0	1	1	0.05
$\mathbf{t}_5$	1	0	0	0	0.10
$\mathbf{t}_6$	1	0	0	1	0.10
$\mathbf{t}_7$	1	0	1	0	0.10
$\mathbf{t}_8$	1	0	1	1	0.10
$\mathbf{t}_9$	2	0	2	2	0.05
$\mathbf{t}_{10}$	2	0	2	3	0.05
$\mathbf{t}_{11}$	2	0	3	2	0.05
$\mathbf{t}_{12}$	2	0	3	3	0.05
$\mathbf{t}_{13}$	0	1	0	0	0.05
$\mathbf{t}_{14}$	1	1	0	2	0.05
$\mathbf{t}_{15}$	2	1	1	3	0.10

Figure 6.3: Relation $\mathbf{r}(A_1A_2A_3A_4)$ satisfies the *contextual weak* BMVD $A_2 = 0 \gtrapprox\!\!> \gtrapprox\!\!> A_1$.

Let $T = \{t_1, t_2, \ldots, t_s\}$ denote a finite set of objects. Consider two equivalence relations θ_1 and θ_2 on T. The binary operator $\circ$, called the *composition*, is defined by: for $t_i, t_k \in T$,

$$t_i(\theta_1 \circ \theta_2)t_k, \text{ if for some } t_j \in T \text{ both } t_i\theta_1 t_j \text{ and } t_j\theta_2 t_k. \tag{6.38}$$

Lemma 6.2 Let θ_1 and θ_2 be two equivalence relations on T. Then $\theta_1 \circ \theta_2$ is an equivalence relation if and only if $\theta_1 \circ \theta_2 = \theta_2 \circ \theta_1$.

Proof: ($\Leftarrow$) By assumption, $t\theta_1 t$ and $t\theta_2 t$ for all $t \in T$. We immediately obtain $t(\theta_1 \circ \theta_2)t$ for all $t \in T$. Thus, $\theta_1 \circ \theta_2$ is reflexive. Consider any $t_i, t_j \in T$ such that $t_i(\theta_1 \circ \theta_2)t_j$. By definition, there exists some $t_k \in T$ such that $t_i\theta_1 t_k$ and $t_k\theta_2 t_j$. By assumption, we also have $t_j\theta_2 t_k$ and $t_k\theta_1 t_i$. By definition, we obtain $t_j(\theta_2 \circ \theta_1)t_i$. Since $\theta_2 \circ \theta_1 = \theta_1 \circ \theta_2$ we have $t_j(\theta_2 \circ \theta_1)t_i$. Thus, $\theta_2 \circ \theta_1$ is symmetric. Consider any $t_i, t_j, t_k \in T$ such that $t_i(\theta_1 \circ \theta_2)t_j$ and $t_j(\theta_1 \circ \theta_2)t_k$. By definition, $t_i((\theta_1 \circ \theta_2) \circ (\theta_1 \circ \theta_2))t_k$. By the *associative property* [72], we obtain $t_i(\theta_1 \circ (\theta_2 \circ \theta_1) \circ \theta_2))t_k$. Since $\theta_1 \circ \theta_2 = \theta_2 \circ \theta_1$ we write $t_i(\theta_1 \circ (\theta_1 \circ \theta_2) \circ \theta_2))t_k$. Another application of the associative property gives $t_i((\theta_1 \circ \theta_1) \circ (\theta_2 \circ \theta_2))t_k$. Since $\theta \circ \theta = \theta$ for any binary relation on T, we obtain the desired result $t_i(\theta_1 \circ \theta_2)t_k$. Thus, $\theta_1 \circ \theta_2$ is transitive. Therefore, $\theta_1 \circ \theta_2$ is an equivalence relation.

($\Rightarrow$) Since $\theta_1 \circ \theta_2$ is an equivalence relation on T, we immediately obtain $\theta_1 \circ \theta_2 = (\theta_1 \circ \theta_2)^{-1}$. The inverse of an equivalence relation on T is equal to the transpose, namely, $(\theta_1 \circ \theta_2)^{-1} = (\theta_1 \circ \theta_2)'$. The *reversal rule* [72] states $(\theta_1 \circ \theta_2)' = (\theta_2' \circ \theta_1')$. Since θ_1 and θ_2 are themselves equivalence relations, we obtain our desired result $(\theta_2' \circ \theta_1') = (\theta_2 \circ \theta_1)$. $\Box$

Lemma 6.2 indicates that $\theta_1 \circ \theta_2$ is an equivalence relation (a partition) if and only if $\theta_1 \circ \theta_2 = \theta_2 \circ \theta_1$. We now define *contextual weak Bayesian multivalued dependency* (CWBMVD) as follows.

Let R be a relation scheme, X and Y disjoint subsets of R, and $Z = R - XY$. A relation $\mathbf{r}(R)$ satisfies the *contextual weak Bayesian multivalued dependency* (CWB-MVD) $X = x \approx\!\!>\!\!\approx\!\!> Y$, if the following two conditions hold:

(i) $\theta(X = xY) \circ \theta(X = xZ) \;=\; \theta(X = xZ) \circ \theta(X = xY)$,

(ii) there *exists* an equivalence class π in the equivalence relation $\theta(X = xY) \circ \theta(X = xZ)$ which satisfies the BMVD $X \Rightarrow\Rightarrow Y$.

Condition (i) says that the composite relation $\theta(X = xY) \circ \theta(X = xZ)$ is an equivalence relation. This is the necessary condition for $X = x \gtrapprox\gtrapprox Y$ to hold. Condition (ii) says that the BMVD $X \Rightarrow\Rightarrow Y$ holds in the usual sense for that particular equivalence class.

Example 6.6 The relation $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 6.3 satisfies the CWBMVD $A_2 = 0 \gtrapprox\gtrapprox A_1$. By equation (6.37), we obtain the following equivalence relations:

$$\theta(A_1A_2 = 0) \quad = \quad \{\ \{\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_4\},\ \{\mathbf{t}_5, \mathbf{t}_6, \mathbf{t}_7, \mathbf{t}_8\},\ \{\mathbf{t}_9, \mathbf{t}_{10}, \mathbf{t}_{11}, \mathbf{t}_{12}\}\ \},$$

and

$$\theta(A_2 = 0A_3A_4) \quad = \quad \{\ \{\mathbf{t}_1, \mathbf{t}_5\}, \{\mathbf{t}_2, \mathbf{t}_6\}, \{\mathbf{t}_3, \mathbf{t}_7\}, \{\mathbf{t}_4, \mathbf{t}_8\}, \{\mathbf{t}_9\}, \{\mathbf{t}_{10}\}, \{\mathbf{t}_{11}\}, \{\mathbf{t}_{12}\}\ \}.$$

Applying equation (6.38), we obtain:

$$\begin{aligned} \theta(A_1A_2 = 0) \circ \theta(A_2 = 0A_3A_4) &= \{\ \pi_1 = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_8\},\ \pi_2 = \{\mathbf{t}_9, \mathbf{t}_{10}, \mathbf{t}_{11}, \mathbf{t}_{12}\}\ \} \\ &= \theta(A_2 = 0A_3A_4) \circ \theta(A_1A_2 = 0). \end{aligned} \tag{6.39}$$

Equation (6.39) indicates that condition (i) in the definition of CWBMVD is satisfied.

As shown in Figure 6.4, both equivalence classes $\pi_1 = \{t_1, t_2, \ldots, t_8\}$ and $\pi_2 = \{\mathbf{t}_9, \mathbf{t}_{10}, \mathbf{t}_{11}, \mathbf{t}_{12}\}$ satisfy the BMVD $A_2 \Rightarrow\Rightarrow A_1$. Since there is at least one equivalence class which satisfies $A_2 \Rightarrow\Rightarrow A_1$, condition (ii) in the definition of CWBMVD is satisfied. By definition, relation $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 6.3 satisfies the CWBMVD $A_2 = 0 \gtrapprox\gtrapprox A_1$.

The important point is that the notion of contextual weak Bayesian multivalued dependency (CWBMVD) is more general than the notion of CBMVD discussed in the last section. For example, the relation $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 6.3 does not satisfy the CBMVD

$$A_2 = 0 \Rightarrow\Rightarrow A_1,$$

$$
\pi_1 \;=\; \begin{array}{|ccccc|}\hline A_1 & A_2 & A_3 & A_4 & A_p \\ \hline 0 & 0 & 0 & 0 & 0.05 \\ 0 & 0 & 0 & 1 & 0.05 \\ 0 & 0 & 1 & 0 & 0.05 \\ 0 & 0 & 1 & 1 & 0.05 \\ 1 & 0 & 0 & 0 & 0.10 \\ 1 & 0 & 0 & 1 & 0.10 \\ 1 & 0 & 1 & 0 & 0.10 \\ 1 & 0 & 1 & 1 & 0.10 \\ \hline \end{array} \;=\; \begin{array}{|ccc|}\hline A_1 & A_2 & A_{p(A_1A_2)} \\ \hline 0 & 0 & 0.2 \\ 1 & 0 & 0.4 \\ \hline \end{array} \;\otimes\; \begin{array}{|cccc|}\hline A_2 & A_3 & A_4 & A_{p(A_2A_3A_4)} \\ \hline 0 & 0 & 0 & 0.15 \\ 0 & 0 & 1 & 0.15 \\ 0 & 1 & 0 & 0.15 \\ 0 & 1 & 1 & 0.15 \\ \hline \end{array}
$$

$$
\pi_2 \;=\; \begin{array}{|ccccc|}\hline A_1 & A_2 & A_3 & A_4 & A_p \\ \hline 2 & 0 & 2 & 2 & 0.05 \\ 2 & 0 & 2 & 3 & 0.05 \\ 2 & 0 & 3 & 2 & 0.05 \\ 2 & 0 & 3 & 3 & 0.05 \\ \hline \end{array} \;=\; \begin{array}{|ccc|}\hline A_1 & A_2 & A_{p(A_1A_2)} \\ \hline 2 & 0 & 0.2 \\ \hline \end{array} \;\otimes\; \begin{array}{|cccc|}\hline A_2 & A_3 & A_4 & A_{p(A_2A_3A_4)} \\ \hline 0 & 2 & 2 & 0.05 \\ 0 & 2 & 3 & 0.05 \\ 0 & 3 & 2 & 0.05 \\ 0 & 3 & 3 & 0.05 \\ \hline \end{array}
$$

Figure 6.4: The equivalence classes $\pi_1 = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_8\}$ (top) and $\pi_2 = \{\mathbf{t}_9, \mathbf{t}_{10}, \mathbf{t}_{11}, \mathbf{t}_{12}\}$ (bottom) both satisfy the BMVD $A_2 \Rightarrow\!\Rightarrow A_1$.

namely,

$$\sigma_{A_2=0}(\mathbf{r}) \;\neq\; \tau_{A_1A_2}(\sigma_{A_2=0}(\mathbf{r})) \otimes \tau_{A_2A_3A_4}(\sigma_{A_2=0}(\mathbf{r})).$$

That is, the set of tuples $\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{12}\}$ does not satisfy the BMVD $A_2 \Rightarrow\!\Rightarrow A_1$. However, the tuples $\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{12}\}$ can be further partitioned such that at least one equivalence class satisfies the BMVD $A_2 \Rightarrow\!\Rightarrow A_1$. As shown in example (6.6), $\mathbf{r}(A_1A_2A_3A_4)$ satisfies the CWBMVD

$$A_2 = 0 \;\approx\!>\!\approx\!>\; A_1.$$

It should be noted that the CWBMVD

$$X = x \;\approx\!>\!\approx\!>\; Y$$

becomes the CBMVD

$$X = x \Rightarrow\!\Rightarrow Y$$

when there is precisely one equivalence class in the equivalence relation $\theta(X = xY) \circ \theta(X = xZ)$.

Lemma 6.3 Let $\mathbf{r}(XYZ)$ be a relation, where X, Y, Z are pairwise disjoint. If $\mathbf{r}$ satisfies the CBMVD $X = x \Rightarrow\Rightarrow Y$, then $\mathbf{r}$ satisfies the CWBMVD $X = x \approx\!>\approx\!> Y$. The converse, however, is not necessarily true.

Proof: Straightforward. $\Box$

6.2 Noncontextual Dependencies

As we saw in the last section, contextual dependencies generalize BMVD by allowing the conditional independence to only hold on part of the relation. However, it may be more desirable to generalize BMVD with a *noncontextual* dependency, namely, a dependency which applies to the entire relation just like BMVD does. In this section, we introduce *weak Bayesian multivalued dependency* (WBMVD) as a more general *noncontextual* dependency than probabilistic conditional independence.

Let R be a relation scheme, X and Y disjoint subsets of R, and $Z = R - XY$. A relation $\mathbf{r}(R)$ satisfies the *weak Bayesian multivalued dependency* (WBMVD) $X \approx\!>\approx\!> Y$, if the following two conditions hold:

(i) $\theta(XY) \circ \theta(XZ) \;=\; \theta(XZ) \circ \theta(XY)$,

(ii) *every* equivalence class π in the equivalence relation $\theta(XY) \circ \theta(XZ)$ satisfies the BMVD $X \Rightarrow\Rightarrow Y$.

Example 6.7 Relation $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 6.5 does not satisfy the BMVD $A_2 \Rightarrow\Rightarrow A_1$. However, $\mathbf{r}(A_1A_2A_3A_4)$ does satisfy the WBMVD $A_2 \approx\!>\approx\!> A_1$. By equation (6.37), we obtain the equivalence relations $\theta(A_1A_2)$ and $\theta(A_2A_3A_4)$:

$$\begin{aligned}\theta(A_1A_2) \;=\; & \{\ \{\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_4\}, \{\mathbf{t}_5, \mathbf{t}_6, \mathbf{t}_7, \mathbf{t}_8\}, \{\mathbf{t}_9, \mathbf{t}_{10}, \mathbf{t}_{11}, \mathbf{t}_{12}\}, \{\mathbf{t}_{13}, \mathbf{t}_{14}, \mathbf{t}_{15}, \mathbf{t}_{16}\}, \\ & \{\mathbf{t}_{17}, \mathbf{t}_{18}, \mathbf{t}_{19}, \mathbf{t}_{20}\}, \{\mathbf{t}_{21}, \mathbf{t}_{22}, \mathbf{t}_{23}, \mathbf{t}_{24}\}, \{\mathbf{t}_{25}, \mathbf{t}_{26}, \mathbf{t}_{27}, \mathbf{t}_{28}\}, \{\mathbf{t}_{29}, \mathbf{t}_{30}, \mathbf{t}_{31}, \mathbf{t}_{32}\}\ \},\end{aligned}$$

and

$$\begin{aligned}\theta(A_2A_3A_4) \quad = \quad & \{\ \{\mathbf{t}_1,\mathbf{t}_5\},\{\mathbf{t}_2,\mathbf{t}_6\},\{\mathbf{t}_3,\mathbf{t}_7\},\{\mathbf{t}_4,\mathbf{t}_8\},\{\mathbf{t}_9,\mathbf{t}_{13}\},\{\mathbf{t}_{10},\mathbf{t}_{14}\},\{\mathbf{t}_{11},\mathbf{t}_{15}\},\\ & \{\mathbf{t}_{12},\mathbf{t}_{16}\},\{\mathbf{t}_{17},\mathbf{t}_{21}\},\{\mathbf{t}_{18},\mathbf{t}_{22}\},\{\mathbf{t}_{19},\mathbf{t}_{23}\},\{\mathbf{t}_{20},\mathbf{t}_{24}\},\{\mathbf{t}_{25},\mathbf{t}_{29}\},\\ & \{\mathbf{t}_{26},\mathbf{t}_{30}\},\{\mathbf{t}_{27},\mathbf{t}_{31}\},\{\mathbf{t}_{28},\mathbf{t}_{32}\}\ \}.\end{aligned}$$

Applying equation (6.38), we obtain:

$$\begin{aligned}& \theta(A_1A_2)\circ\theta(A_2A_3A_4)\\ = \quad & \{\ \pi_1=\{\mathbf{t}_1,\mathbf{t}_2,\mathbf{t}_3,\mathbf{t}_4\},\mathbf{t}_5,\mathbf{t}_6,\mathbf{t}_7,\mathbf{t}_8\},\ \ \pi_2=\{\mathbf{t}_9,\mathbf{t}_{10},\mathbf{t}_{11},\mathbf{t}_{12}\},\mathbf{t}_{13},\mathbf{t}_{14},\mathbf{t}_{15},\mathbf{t}_{16}\},\\ & \pi_3=\{\mathbf{t}_{17},\mathbf{t}_{18},\mathbf{t}_{19},\mathbf{t}_{20},\mathbf{t}_{21},\mathbf{t}_{22},\mathbf{t}_{23},\mathbf{t}_{24}\},\ \ \pi_4=\{\mathbf{t}_{25},\mathbf{t}_{26},\mathbf{t}_{27},\mathbf{t}_{28},\mathbf{t}_{29},\mathbf{t}_{30},\mathbf{t}_{31},\mathbf{t}_{32}\}\ \}\\ = \quad & \theta(A_2A_3A_4)\circ\theta(A_1A_2). \qquad (6.40)\end{aligned}$$

Equation (6.40) indicates that condition (i) of the definition of WBMVD is satisfied. It can be verified that each equivalence class $\pi_1 = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_8\}$, $\pi_2 = \{\mathbf{t}_9, \mathbf{t}_{10}, \ldots, \mathbf{t}_{16}\}$, $\pi_3 = \{\mathbf{t}_{17}, \mathbf{t}_{18}, \ldots, \mathbf{t}_{24}\}$, and $\pi_4 = \{\mathbf{t}_{25}, \mathbf{t}_{26}, \ldots, \mathbf{t}_{32}\}$ satisfies the BMVD $A_2 \Rightarrow\Rightarrow A_1$. That is, condition (ii) in the definition of WBMVD is satisfied. By definition, relation $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 6.5 satisfies the WBMVD $A_2 \gtrapprox\gtrapprox A_1$.

	A_1	A_2	A_3	A_4	A_p	
$\mathbf{t}_1$	0	0	0	0	p_1	
$\mathbf{t}_2$	0	0	0	1	p_1	
$\mathbf{t}_3$	0	0	1	0	p_1	
$\mathbf{t}_4$	0	0	1	1	p_1	$= \tau_{A_1A_2}(\pi_1) \otimes \tau_{A_2A_3A_4}(\pi_1)$
$\mathbf{t}_5$	1	0	0	0	p_1	
$\mathbf{t}_6$	1	0	0	1	p_1	
$\mathbf{t}_7$	1	0	1	0	p_1	
$\mathbf{t}_8$	1	0	1	1	p_1	
$\mathbf{t}_9$	2	0	2	2	p_1	
$\mathbf{t}_{10}$	2	0	2	3	p_1	
$\mathbf{t}_{11}$	2	0	3	2	p_1	
$\mathbf{t}_{12}$	2	0	3	3	p_1	$= \tau_{A_1A_2}(\pi_2) \otimes \tau_{A_2A_3A_4}(\pi_2)$
$\mathbf{t}_{13}$	3	0	2	2	p_1	
$\mathbf{t}_{14}$	3	0	2	3	p_1	
$\mathbf{t}_{15}$	3	0	3	2	p_1	
$\mathbf{t}_{16}$	3	0	3	3	p_1	
$\mathbf{t}_{17}$	0	1	0	0	p_2	
$\mathbf{t}_{18}$	0	1	0	3	p_2	
$\mathbf{t}_{19}$	0	1	3	0	p_2	
$\mathbf{t}_{20}$	0	1	3	3	p_2	$= \tau_{A_1A_2}(\pi_3) \otimes \tau_{A_2A_3A_4}(\pi_3)$
$\mathbf{t}_{21}$	2	1	0	0	p_2	
$\mathbf{t}_{22}$	2	1	0	3	p_2	
$\mathbf{t}_{23}$	2	1	3	0	p_2	
$\mathbf{t}_{24}$	2	1	3	3	p_2	
$\mathbf{t}_{25}$	1	1	1	1	p_3	
$\mathbf{t}_{26}$	1	1	1	2	p_3	
$\mathbf{t}_{27}$	1	1	2	1	p_3	
$\mathbf{t}_{28}$	1	1	2	2	p_3	$= \tau_{A_1A_2}(\pi_4) \otimes \tau_{A_2A_3A_4}(\pi_4)$
$\mathbf{t}_{29}$	3	1	1	1	p_4	
$\mathbf{t}_{30}$	3	1	1	2	p_4	
$\mathbf{t}_{31}$	3	1	2	1	p_4	
$\mathbf{t}_{32}$	3	1	2	2	p_4	

Figure 6.5: Relation $\mathbf{r}(A_1A_2A_3A_4)$ satisfies the WBMVD $A_2 \gtrapprox\!\!\gtrapprox A_1$.

We now provide an equivalent way to view WBMVDs.

Example 6.8 Consider the relation $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 6.6 (left). Suppose we wish to verify that the WBMVD $A_1 \ggg\!\!\ggg A_2$ is satisfied by $\mathbf{r}(R)$. The tuples of $\mathbf{r}(R)$ are partitioned by

$$\theta(A_1A_2) = \{\ \{\mathbf{t}_1, \mathbf{t}_3\}, \{\mathbf{t}_2\}\ \}$$

and

$$\theta(A_1A_3A_4) = \{\ \{\mathbf{t}_1\}, \{\mathbf{t}_2\}, \{\mathbf{t}_3\}\ \}.$$

We then obtain

$$\theta(A_1A_2) \circ \theta(A_1A_3A_4) \quad = \quad \theta(A_1A_3A_4) \circ \theta(A_1A_2) = \{\ \{\mathbf{t}_1, \mathbf{t}_3\}, \{\mathbf{t}_2\}\ \}. \quad (6.41)$$

Thus condition (i) of the definition of WBMVD is satisfied. The dummy attribute D can be added to $\mathbf{r}(A_1A_2A_3A_4)$ resulting in the relation $\mathbf{r}(A_1A_2A_3A_4D)$ shown in Figure 6.6 (right), where the values of D are assigned according to the partition defined in equation (6.41). The reader can verify that the traditional BMVD $A_1D \Rightarrow\!\!\Rightarrow A_2$ is satisfied by $\mathbf{r}(A_1A_2A_3A_4D)$, namely,

$$\mathbf{r}(A_1A_2A_3A_4D) \quad = \quad \tau_{A_1A_2D}(\mathbf{r}) \otimes \tau_{A_1A_3A_4D}(\mathbf{r}).$$

A_1	A_2	A_3	A_4	A_p
0	0	0	0	0.7
0	1	0	1	0.2
0	0	1	0	0.1

A_1	A_2	A_3	A_4	D	A_p
0	0	0	0	0	0.7
0	1	0	1	1	0.2
0	0	1	0	0	0.1

Figure 6.6: Relation $\mathbf{r}(A_1A_2A_3A_4)$ satisfying the WBMVD $A_1 \ggg\!\!\ggg A_2$ (left) can be equivalently stated as relation $\mathbf{r}(A_1A_2A_3A_4D)$ satisfying the BMVD $A_1D \Rightarrow\!\!\Rightarrow A_2$ (right), where the values of the dummy attribute D are assigned according to the partition defined in equation (6.41).

Lemma 6.4 Let $\mathbf{r}(XYZ)$ be a relation, where X, Y, Z are pairwise disjoint. If $\mathbf{r}$ satisfies the BMVD $X \Rightarrow\Rightarrow Y$, then $\mathbf{r}$ also satisfies the WBMVD $X \approx\!>\approx\!> Y$. The converse, however, is not necessarily true.

Proof: Straightforward. $\Box$

We conclude this section by comparing the noncontextual dependency WBMVD with the contextual dependencies CWBMVD and CBMVD.

Lemma 6.5 Let $\mathbf{r}(XYZ)$ be a relation, where X, Y, Z are pairwise disjoint. If $\mathbf{r}$ satisfies the WBMVD $X \approx\!>\approx\!> Y$, then $\mathbf{r}$ also satisfies the CWBMVD $X = x \approx\!>\approx\!> Y$.

Proof: Straightforward. $\Box$

It should be noted that the WBMVD $X \approx\!>\approx\!> Y$ is *not* the same as the CWBMVD $X = x \approx\!>\approx\!> Y$ holding for all X-values x. In condition (ii) of the definition of CWBMVD, only a *single* equivalence class is required to satisfy the BMVD $X \Rightarrow\Rightarrow Y$. On the other hand, condition (ii) of the definition of WBMVD requires *every* equivalence class in every X-value x to satisfy $X \Rightarrow\Rightarrow Y$. To avoid confusion between CWBMVD and WBMVD, it would perhaps be more appropriate to introduce another dependency which requires at least one equivalence class in every X-value x to satisfy the BMVD $X \Rightarrow\Rightarrow Y$. However, this proposed dependency can be seen as simply the CWBMVD $X = x \approx\!>\approx\!> Y$ holding for every X-value x.

Lemma 6.6 Let $\mathbf{r}(XYZ)$ be a relation, where X, Y, Z are pairwise disjoint. If $\mathbf{r}$ satisfies the WBMVD $X \approx\!>\approx\!> Y$, then $\mathbf{r}$ does not necessarily satisfy the CBMVD $X = x \Rightarrow\Rightarrow Y$, for some X-value x appearing in $\mathbf{r}$. Similarly, if $\mathbf{r}$ satisfies the CBMVD $X = x \Rightarrow\Rightarrow Y$, for some X-value x appearing in $\mathbf{r}$, then $\mathbf{r}$ does not necessarily satisfy the WBMVD $X \approx\!>\approx\!> Y$.

Proof: Straightforward. $\Box$

Even though Lemmas 6.3 and 6.5 indicate that both CBMVD and WBMVD imply CWBMVD, Lemma 6.6 states that no implication holds between the contextual dependency CBMVD and the noncontextual dependency WBMVD.

6.3 Representing These Dependencies

In this section, we examine two ways to represent these more general dependencies. One method involves the use of *multiple* Bayesian networks, while the other incorporates *dummy* nodes into a single Bayesian network.

Recall the conditional probability distribution in Table 6.1. This distribution does not satisfy the BMVD $Y \Rightarrow\Rightarrow X$. That is, variables X and ZW are not conditionally independent given Y. The Bayesian network in Figure 6.7 constructed using the notion of BMVD does not reflect any independency between variables X and ZW given Y.

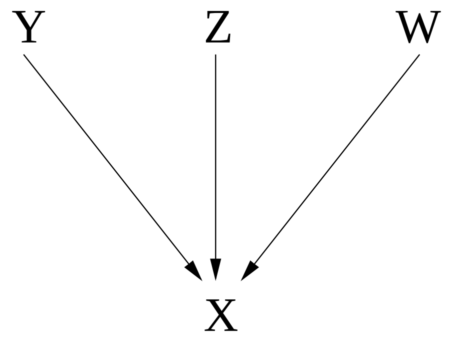

Figure 6.7: The constructed Bayesian network reflects the fact that the CPD in Table 6.1 does not satisfy the BMVD $Y \Rightarrow\Rightarrow X$.

Geiger and Heckerman [34] proposed the use of multiple Bayesian networks (*Bayesian multinets*) to represent and reason with CBMVDs. That is, each Bayesian network reflects the CBMVD $X = x \Rightarrow\Rightarrow Y$ statements that hold for that particular X-value x. The particular Bayesian network used during inference is determined by the evidence.

Example 6.9 Even though the conditional probability distribution in Table 6.1 does not satisfy the BMVD $Y \Rightarrow\Rightarrow X$, Example 6.1 demonstrated that the distribution does satisfy the CBMVD $Y = 0 \Rightarrow\Rightarrow X$. As shown in Figure 6.8 (left), one Bayesian network can be constructed to reflect $Y = 0 \Rightarrow\Rightarrow X$ and another to indicate that the distribution does not satisfy the CBMVD $Y = 1 \Rightarrow\Rightarrow X$.

Example 6.9 demonstrates that one Bayesian network can be constructed to capture the conditional independence of X and ZW given $Y = 0$, and another to show their dependence when $Y = 1$.

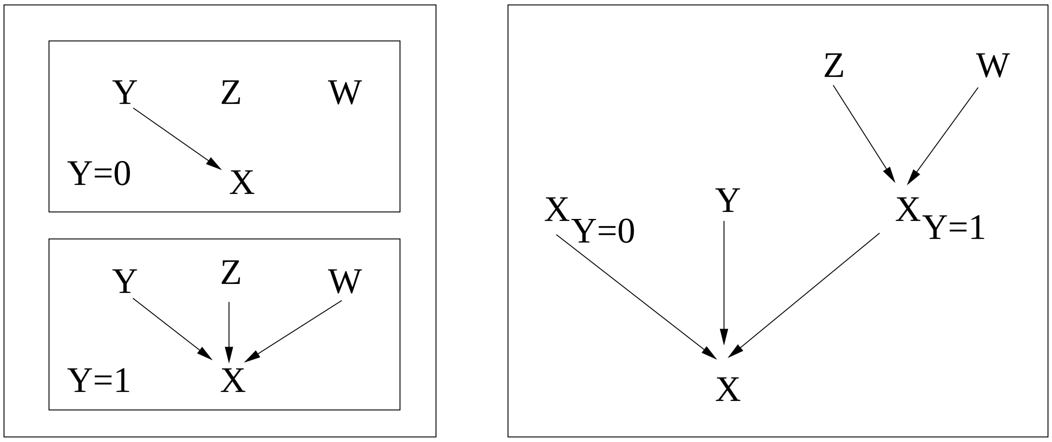

Figure 6.8: The use *Bayesian multinets* (left) and auxiliary nodes in a single Bayesian network (right) to capture the CPD in Table 6.1 satisfying the CBMVD $Y = 0 \Rightarrow\Rightarrow X$.

An alternative method for representing CBMVDs, suggested by Boutilier, Friedman, Goldszmidt, and Koller [13], is to incorporate auxiliary nodes into a single Bayesian network. In this case, dummy nodes such as $Y_{X=x}$ are used to reflect the CBMVDs $X = x \Rightarrow\Rightarrow Y$ that hold for that particular X-value x. During the inference process, if the X-value is x_1, then the node Y in the Bayesian network will become the dummy node $Y_{X=x_1}$. For more details see [13].

Example 6.10 Recall the CPD in Table 6.1. As shown in Figure 6.8 (right), the CBMVD $Y = 0 \Rightarrow\Rightarrow X$ is captured with the auxiliary node $X_{Y=0}$. On the other hand, the distribution not satisfying the CBMVD $Y = 1 \Rightarrow\Rightarrow X$ is reflected with the auxiliary node $X_{Y=1}$.

The notion of CWBMVD $X = x \approx\!>\!\approx\!> Y$ generalizes the CBMVD $X = x \Rightarrow\!\Rightarrow Y$ by *further* partitioning the set of tuples with X-value x. Thus, it is straightforward to represent CWBMVD with multiple Bayesian networks.

Example 6.11 Recall that the conditional probability distribution in Table 6.2 does not satisfy the CBMVD $Y = 0 \Rightarrow\!\Rightarrow X$. However, the distribution does satisfy the CWBMVD $Y = 0 \approx\!>\!\approx\!> X$. This dependency is captured by the use of multiple Bayesian networks as shown in Figure 6.9.

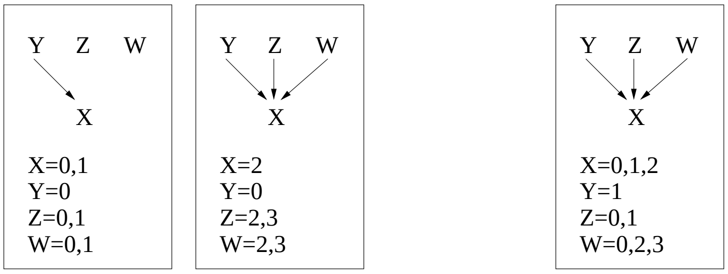

Figure 6.9: The use multiple Bayesian networks to capture the CPD in Table 6.2 satisfying the CWBMVD $Y = 0 \approx\!>\!\approx\!> X$.

There are two alternative ways to represent the *noncontextual* dependency WBMVD. One is in terms of multiple Bayesian networks as in [34]. The difference here is that each Bayesian network has the *same* dependency structure (DAG). As with the Bayesian multinets approach [34], the particular CPD to be used is determined by the evidence. An alternative method to represent WBMVD is to use a single Bayesian network as in [13]. The difference in this case is that *no* dummy nodes need to be incorporated into the Bayesian network. Instead each node is simply associated with a set of CPDs, compared with a single CPD using the notion of BMVD in traditional Bayesian networks.

Example 6.12 Recall that relation $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 6.5 does not satisfy the BMVD $A_2 \Rightarrow\!\Rightarrow A_1$. However, $\mathbf{r}(A_1A_2A_3A_4)$ does satisfy the WBMVD $A_2 \approx\!>\!\approx\!> A_1$. This dependency is captured by the use of multiple Bayesian networks with the same dependency structure as shown in Figure 6.10.

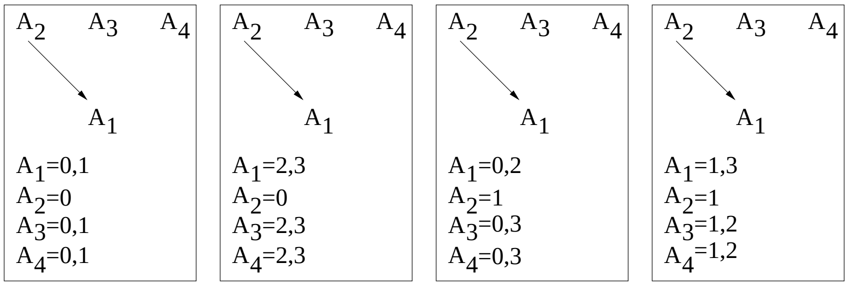

Figure 6.10: Relation $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 6.5 satisfying the WBMVD $A_2 \approx\!\!>\!\!\approx\!\!> A_1$ is captured by multiple Bayesian networks with the same structure.

6.4 Summary

Suppose a relation $\mathbf{r}(XYZ)$ satisfies the BMVD $X \Rightarrow\!\Rightarrow Y$. It can be verified that the equivalence classes in the equivalence relation $\theta(XY) \circ \theta(XZ)$ are precisely those defined by $\theta(X)$. That is,

$$\theta(X) \;=\; \theta(XY) \circ \theta(XZ) \;=\; \theta(XZ) \circ \theta(XY). \qquad (6.42)$$

In other words, the relation $\mathbf{r}(XYZ)$ is partitioned according to the X-values and every equivalence class in $\theta(X)$ satisfies the BMVD $X \Rightarrow\!\Rightarrow Y$.

The notion of CBMVD $X = x \Rightarrow\!\Rightarrow Y$ generalizes the BMVD $X \Rightarrow\!\Rightarrow Y$ by only requiring the partition with X-value x in $\theta(X)$ in equation (6.42) to satisfy $X \Rightarrow\!\Rightarrow Y$. That is, the other partitions in $\theta(X)$ may or may not satisfy $X \Rightarrow\!\Rightarrow Y$.

The notion of CWBMVD $X = x \approx\!\!>\!\!\approx\!\!> Y$ generalizes the CBMVD $X = x \Rightarrow\!\Rightarrow Y$ by only requiring a single equivalence class in a *further* partition of the X-value x partition in $\theta(X)$ to satisfy the BMVD $X \Rightarrow\!\Rightarrow Y$. In other words, the tuples with X-value x may themselves be partitioned such that $X \Rightarrow\!\Rightarrow Y$ holds in at least one equivalence class.

On the other hand, the *noncontextual* dependency WBMVD $X \approx\!\!>\!\!\approx\!\!> Y$ generalizes the BMVD $X \Rightarrow\!\Rightarrow Y$ by only requiring the $X \Rightarrow\!\Rightarrow Y$ to hold in every equivalence class in the equivalence relation

$$\theta(XY) \circ \theta(XZ) \;=\; \theta(XZ) \circ \theta(XY). \qquad (6.43)$$

That is, equation (6.43) indicates that WBMVD $X \approx\!>\approx\!> Y$ does not require the equivalence classes to be defined precisely by $\theta(X)$ as the BMVD $X \Rightarrow\Rightarrow Y$ does in equation (6.42).

The notion of weak Bayesian multivalued dependency was motivated by *weak multivalued dependency* introduced by Jaeschke and Scheck [46]. In the next section, we examine a useful property of WBMVD.

Other work on general forms of BMVD can be found in [39, 66, 71]. Poole [71] suggested a framework for probabilistic Horn-clause abduction in which new hypotheses can be invented to represent any general form of dependence between variables. In [39, 66], the notion of *causal* independence is suggested to facilitate knowledge acquisition and increase the speed of inference.

Chapter 7

COMPLEX-VALUE BAYESIAN NETWORKS

In this chapter, we present a brief discussion of complex-value probabilistic networks. That is, the value of an attribute is allowed to be another distribution itself. We introduce operators called *nest* and *unnest* to *coarsen* and *refine* parts of a probabilistic network, respectively. We show that WBMVD is a necessary and sufficient condition for ensuring *consistency* in such complex-value networks. Lastly, we demonstrate that the nest operator can be applied locally to a marginal distribution with the same effect as if applied to the joint distribution itself.

Koller and Pfeffer [49] suggested a framework for the modeling of and inference in a large Bayesian network. In this framework, parts of the network can be coarsened and refined.

Example 7.1 [49] Consider the Bayesian network for a car accident as shown in Figure 7.1. One can *refine* the node Car to reveal the internal structure as shown in Figure 7.2, where the rest of the network is not illustrated due to space limitations.

These ideas are now formalized in our probabilistic relational database model.

7.1 Complex-Value Operators

Our purpose here is to present two operators, called *nest* and *unnest*, for coarsening and refining a probabilistic network, respectively.

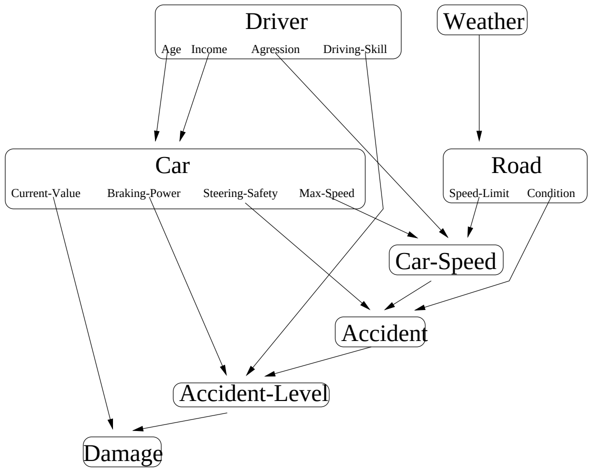

Figure 7.1: A Bayesian network for a car accident.

7.1.1 The Nest Operator

The *nest* operator ϕ is used to *coarsen* a relation $\mathbf{r}(XY)$. Intuitively, $\phi_{B=Y}(\mathbf{r})$ groups together all the Y-values into a nested relation given the same X-value.

Let $\mathbf{r}(R)$ be a relation on $R = A_1 A_2 \ldots A_m$. Let X and Y be disjoint sets of attributes such that $XY = R$. Coarsening attributes Y in $\mathbf{r}$ as attribute B is the relation $\phi_{B=Y}(\mathbf{r})$ with attributes XB defined as

$$\phi_{B=Y}(\mathbf{r}) \quad = \quad \{\mathbf{t} \mid \mathbf{t}(X) = \mathbf{u}(X) \text{ and } \mathbf{t}(B) = \{\mathbf{u}(YA_p)\} \text{ and } \mathbf{t}(A_{p(XB)}) = \sum_{\mathbf{u}} \mathbf{u}(A_p)\},$$

where $\mathbf{u} \in \mathbf{r}$. The attribute A_p in the B-value is relabelled $A_{p(Y)}$ and the values normalized.

Example 7.2 Consider the relation $\mathbf{r}(A_1A_2A_3)$ as depicted in Figure 7.3. Performing $\phi_{B=(A_2A_3)}(\mathbf{r})$ results in the relation shown in Figure 7.4.

7.1.2 The Unnest Operator

Whereas nest coarsens a relation, the *unnest* operator ψ *refines* a relation. Intuitively, $\psi_{B=Y}(\mathbf{r})$ joins each X-value with each Y-value in the nested B-value.

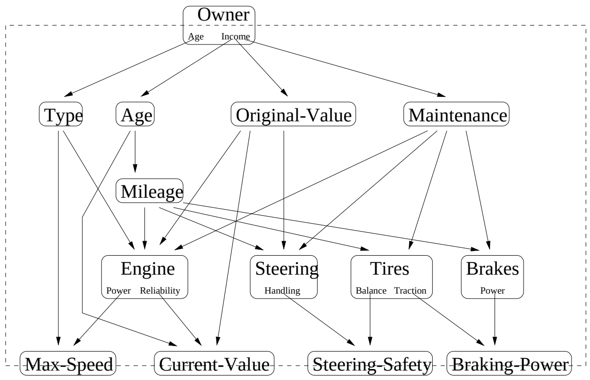

Figure 7.2: Refining the Bayesian network in Figure 7.1 to reveal the internal structure of the variable Car.

$$\mathbf{r}(A_1A_2A_3) = \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_p \\ \hline 1 & 1 & 2 & 0.125 \\ 1 & 3 & 4 & 0.250 \\ 1 & 5 & 6 & 0.125 \\ 2 & 1 & 3 & 0.125 \\ 2 & 2 & 4 & 0.125 \\ 3 & 0 & 0 & 0.125 \\ 3 & 0 & 1 & 0.125 \\ \hline \end{array}$$

Figure 7.3: A relation $\mathbf{r}(A_1A_2A_3)$.

Revealing the nested variables Y in attribute B of relation $\mathbf{r}(XB)$ is the relation $\psi_{B=Y}(\mathbf{r})$ with attributes XY defined as

$$\psi_{B=Y}(\mathbf{r}) \quad = \quad \{\mathbf{t} \mid \mathbf{t}(X) = \mathbf{u}(X) \text{ and } \mathbf{t}(Y) \in \mathbf{u}(B) \text{ and } \mathbf{t}(A_p) = \mathbf{v}(A_{p(Y)}) \cdot \mathbf{u}(A_p)\},$$

where $\mathbf{u} \in \mathbf{r}$, $\mathbf{v} \in \mathbf{u}(B)$, and $\mathbf{v}(Y) = \mathbf{t}(Y)$. Note that we may write $\psi_{B=Y}(\mathbf{r})$ as $\psi_B(\mathbf{r})$ since $\mathbf{Y}$ is implied by B.

Example 7.3 Revealing the nested attributes A_2A_3 in the coarsened relation $\mathbf{r}(XB)$ in Figure 7.4 results in the refined relation $\psi_B(\mathbf{r})$ shown in Figure 7.3.

$\phi_{B=A_2A_3}(\mathbf{r}) \; =$

<table>
<tr><th>A_1</th><th>B</th><th>$A_{p(A_1B)}$</th></tr>
<tr><td>1</td><td><table><tr><th>A_2</th><th>A_3</th><th>$A_{p(A_2A_3)}$</th></tr><tr><td>1</td><td>2</td><td>0.25</td></tr><tr><td>3</td><td>4</td><td>0.50</td></tr><tr><td>5</td><td>6</td><td>0.25</td></tr></table></td><td>0.50</td></tr>
<tr><td>2</td><td><table><tr><th>A_2</th><th>A_3</th><th>$A_{p(A_2A_3)}$</th></tr><tr><td>1</td><td>3</td><td>0.5</td></tr><tr><td>2</td><td>4</td><td>0.5</td></tr></table></td><td>0.25</td></tr>
<tr><td>3</td><td><table><tr><th>A_2</th><th>A_3</th><th>$A_{p(A_2A_3)}$</th></tr><tr><td>0</td><td>0</td><td>0.5</td></tr><tr><td>0</td><td>1</td><td>0.5</td></tr></table></td><td>0.25</td></tr>
</table>

Figure 7.4: The relation $\phi_{B=A_2A_3}(\mathbf{r})$ obtained from the relation $\mathbf{r}(A_1A_2A_3)$ in Figure 7.3 by coarsening attributes A_2A_3 as B.

7.2 Maintaining Consistency

The main result of this section is that WBMVD is a necessary and sufficient condition for ensuring *consistency* in such complex-value networks.

The following example demonstrates that inconsistency may arise due to the fact that the nest operator is not commutative. (In contrast, the unnest operator is commutative.)

Example 7.4 Consider the relation $\mathbf{r}(A_1A_2A_3)$ in Figure 7.5. Suppose we wish to coarsen the level of detail for A_1 and A_3. Computing $\phi_{B_1=A_1}(\phi_{B_3=A_3}(\mathbf{r}))$ is shown in Figure 7.6 (top). On the other hand, computing $\phi_{B_3=A_3}(\phi_{B_1=A_1}(\mathbf{r}))$ is illustrated in Figure 7.6 (bottom). Clearly, $\phi_{B_1=A_1}(\phi_{B_3=A_3}(\mathbf{r})) \neq \phi_{B_3=A_3}(\phi_{B_1=A_1}(\mathbf{r}))$.

Example 7.4 demonstrates that the nest operator does not necessarily commute. On the other hand, the following example demonstrates that the nest operator may commute without the relation satisfying the corresponding BMVD.

$$\mathbf{r}(A_1A_2A_3) \;=\; \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_p \\ \hline 0 & 0 & 0 & 0.4 \\ 1 & 0 & 0 & 0.4 \\ 1 & 0 & 1 & 0.2 \\ \hline \end{array}$$

Figure 7.5: A relation $\mathbf{r}(A_1A_2A_3)$.

Example 7.5 The BMVD $A_2 \Rightarrow\Rightarrow A_1$ is not satisfied by the relation $\mathbf{r}(A_1A_2A_3)$ shown in Figure 7.7. That is, A_1 and A_3 are not conditionally independent given A_2. However, $\phi_{B_1=A_1}(\phi_{B_3=A_3}(\mathbf{r})) = \phi_{B_3=A_3}(\phi_{B_1=A_1}(\mathbf{r}))$ as shown in Figure 7.8.

Example 7.5 indicates that probabilistic conditional independence is not a necessary condition for nest to commute. We now show that WBMVD is a necessary and sufficient condition for nest to commute. In other words, WBMVD ensures *consistency* in complex-value networks.

Theorem 7.1 Let $\mathbf{r}(R)$ be a relation, and X, Y, Z be pairwise disjoint subsets such that $XYZ = R$. Relation $\mathbf{r}(R)$ satisfies the WBMVD $X \approx\!>\approx\!> Y$ if and only if

$$\phi_{B_1=Y}(\phi_{B_2=Z}(\mathbf{r})) = \phi_{B_2=Z}(\phi_{B_1=Y}(\mathbf{r})).$$

Proof: ($\Rightarrow$) Suppose relation $\mathbf{r}(R)$ satisfies the WBMVD $X \approx\!>\approx\!> Y$. By definition, the set of tuples defining each equivalence class in the equivalence relation $\theta(YX) \circ \theta(XZ)$ satisfies the BMVD $X \Rightarrow\Rightarrow Y$. Without loss of generality, one equivalence class π is shown in Figure 7.9, where $c = a_1 + a_2 = b_1 + b_2$.

Given $X \approx\!>\approx\!> Y$, it is clear that computing $\phi_{B_2=Z}(\mathbf{r})$ only groups together tuples in the same equivalence class in the equivalence relation $\theta(YX) \circ \theta(XZ)$. Computing $\phi_{B_2=Z}(\mathbf{r})$ modifies the equivalence class π in Figure 7.9 into $\phi_{B_2=Z}(\pi)$ depicted in Figure 7.10. The expressions for probability values of Z can be simplified since, for example

$$\frac{(a_1b_1)c}{(a_1b_1)c + (a_1b_2)c} = \frac{b_1}{b_1 + b_2}.$$

$\phi_{B_1=A_1}(\phi_{B_3=A_3}(\mathbf{r})) =$

B_1	A_2	B_3	$A_{p(B_1A_1B_3)}$
$\begin{array}{\|cc\|}\hline A_1 & A_{p(A_1)} \\ \hline 0 & 1.0 \\ \hline\end{array}$	0	$\begin{array}{\|cc\|}\hline A_3 & A_{p(A_3)} \\ \hline 0 & 1.0 \\ \hline\end{array}$	0.6
$\begin{array}{\|cc\|}\hline A_1 & A_{p(A_1)} \\ \hline 1 & 1.0 \\ \hline\end{array}$	0	$\begin{array}{\|cc\|}\hline A_3 & A_{p(A_3)} \\ \hline 0 & 0.66 \\ 1 & 0.33 \\ \hline\end{array}$	0.4

$\phi_{B_3=A_3}(\phi_{B_1=A_1}(\mathbf{r})) =$

B_1	A_2	B_3	$A_{p(B_1A_1B_3)}$
$\begin{array}{\|cc\|}\hline A_1 & A_{p(A_1)} \\ \hline 0 & 0.5 \\ 1 & 0.5 \\ \hline\end{array}$	0	$\begin{array}{\|cc\|}\hline A_3 & A_{p(A_3)} \\ \hline 0 & 1.0 \\ \hline\end{array}$	0.8
$\begin{array}{\|cc\|}\hline A_1 & A_{p(A_1)} \\ \hline 1 & 1.0 \\ \hline\end{array}$	0	$\begin{array}{\|cc\|}\hline A_3 & A_{p(A_3)} \\ \hline 0 & 1.0 \\ \hline\end{array}$	0.2

Figure 7.6: The relations $\phi_{B_1=A_1}(\phi_{B_3=A_3}(\mathbf{r}))$ (top) and $\phi_{B_3=A_3}(\phi_{B_1=A_1}(\mathbf{r}))$ (bottom) indicate that the nest operation does not commute, where $\mathbf{r}$ is the relation in Figure 7.5.

Since the B_2-values are identical, computing $\phi_{B_1=Y}(\phi_{B_2=Z}(\mathbf{r}))$ modifies the equivalence class $\phi_{B_2=Z}(\pi)$ in Figure 7.10 into $\phi_{B_1=Y}(\phi_{B_2=Z}(\pi))$ illustrated in Figure 7.11.

This demonstrates that each equivalence class in $\theta(YX) \circ \theta(XZ)$ is reduced into a single tuple with all the Y-values and Z-values represented in one B_1-value and B_2-value, respectively. Since the conditional independence of Y and Z given X is symmetric, this argument can be applied to show that $\phi_{B_2=Z}(\phi_{B_1=Y}(\mathbf{r}))$ also reduces the initial equivalence class π in Figure 7.9 into the single tuple shown in Figure 7.11. This argument holds for each equivalence class in the equivalence relation $\theta(YX) \circ \theta(XZ)$.

($\Leftarrow$) Let $\mathbf{r}$ be a relation such that $\phi_{B_1=Y}(\phi_{B_2=Z}(\mathbf{r})) = \phi_{B_2=Z}(\phi_{B_1=Y}(\mathbf{r}))$. Consider one tuple in $\phi_{B_1=Y}(\phi_{B_2=Z}(\mathbf{r}))$ as shown in Figure 7.11. The equality holding indicates that

$$\mathbf{r}(A_1A_2A_3) \; = \; \begin{array}{|cccc|} \hline A_1 & A_2 & A_3 & A_p \\ \hline 0 & 0 & 0 & 0.225 \\ 0 & 0 & 1 & 0.375 \\ 1 & 0 & 0 & 0.075 \\ 1 & 0 & 1 & 0.125 \\ 2 & 0 & 2 & 0.200 \\ \hline \end{array}$$

Figure 7.7: The BMVD $A_2 \Rightarrow\Rightarrow A_1$ is not satisfied by the relation $\mathbf{r}(A_1A_2A_3)$. However, $\phi_{B_1=A_1}(\phi_{B_3=A_3}(\mathbf{r})) = \phi_{B_3=A_3}(\phi_{B_1=A_1}(\mathbf{r}))$.

<table>
<tr><th>B_1</th><th>A_2</th><th>B_3</th><th>$A_{p(B_1A_2B_3)}$</th></tr>
<tr><td><table><tr><th>A_1</th><th>$A_{p(A_1)}$</th></tr><tr><td>0</td><td>0.75</td></tr><tr><td>1</td><td>0.25</td></tr></table></td><td>0</td><td><table><tr><th>A_3</th><th>$A_{p(A_3)}$</th></tr><tr><td>0</td><td>0.375</td></tr><tr><td>1</td><td>0.675</td></tr></table></td><td>0.8</td></tr>
<tr><td><table><tr><th>A_1</th><th>$A_{p(A_1)}$</th></tr><tr><td>1</td><td>1.0</td></tr></table></td><td>0</td><td><table><tr><th>A_3</th><th>$A_{p(A_3)}$</th></tr><tr><td>0</td><td>1.0</td></tr></table></td><td>0.2</td></tr>
</table>

Figure 7.8: The coarsened relation representing both $\phi_{B_1=A_1}(\phi_{B_3=A_3}(\mathbf{r}))$ and $\phi_{B_3=A_3}(\phi_{B_1=A_1}(\mathbf{r}))$, where $\mathbf{r}(A_1A_2A_3)$ is shown in Figure 7.7.

$c = a_1 + a_2 = b_1 + b_2$, as otherwise the order of nesting becomes relevant. The result of $\psi_{B_1}(\phi_{B_1=Y}(\phi_{B_2=Z}(\mathbf{r}))$ is depicted in Figure 7.10. Attributes Y and Z are conditionally independent given X in the normalized distribution $\psi_{B_2}(\psi_{B_1}(\phi_{B_1=Y}(\phi_{B_2=Z}(\mathbf{r}))$ as shown in Figure 7.9. This argument holds for each tuple in $\phi_{B_1=Y}(\phi_{B_2=Z}(\mathbf{r}))$. By definition, relation $\mathbf{r}$ satisfies the WBMVD $X \gtrapprox\gtrapprox Y$. $\square$

It follows that Lemma 6.4 indicates that BMVD is only a *sufficient* condition for the nest operator to commute. It should be noted that Theorem 7.1 can be extended to more than two nest operations in a similar fashion as in [31].

In practice, a nested relation is represented as a *value*. Thus, in addition to a domain expert specifying BMVDs and WBMVDs that hold, we also assume a

$$\pi = \begin{array}{|cccc|}\hline Y & X & Z & A_p \\ \hline y_1 & x_1 & z_1 & (a_1b_1)/c \\ y_1 & x_1 & z_2 & (a_1b_2)/c \\ y_2 & x_1 & z_1 & (a_2b_1)/c \\ y_2 & x_1 & z_2 & (a_2b_2)/c \\ \hline \end{array} = \begin{array}{|ccc|}\hline Y & X & A_{p(YX)} \\ \hline y_1 & x_1 & a_1 \\ y_2 & x_1 & a_2 \\ \hline \end{array} \otimes \begin{array}{|ccc|}\hline Y & Z & A_{p(XZ)} \\ \hline x_1 & z_1 & b_1 \\ x_1 & z_2 & b_2 \\ \hline \end{array}$$

Figure 7.9: The set of tuples defining each equivalence class π in the equivalence relation $\theta(YX) \circ \theta(XZ)$ satisfies the BMVD $X \Rightarrow\Rightarrow Y$.

$$\phi_{B_2=Z}(\pi) = \begin{array}{|cccc|}\hline Y & X & B_2 & A_{p(YXB_2)} \\ \hline & & & \\ y_1 & x_1 & \begin{array}{|cc|}\hline Z & A_{p(Z)} \\ \hline z_1 & (a_1b_1)/c = b_1 \\ z_2 & (a_1b_2)/c = b_2 \\ \hline \end{array} & a_1(b_1+b_2)/c = a_1 \\ & & & \\ y_2 & x_1 & \begin{array}{|cc|}\hline Z & A_{p(Z)} \\ \hline z_1 & (a_2b_1)/c = b_1 \\ z_2 & (a_2b_2)/c = b_2 \\ \hline \end{array} & a_2(b_1+b_2)/c = a_2 \\ & & & \\ \hline \end{array}$$

Figure 7.10: The nested equivalence class $\phi_{B_2=Z}(\pi)$ in $\phi_{B_2=Z}(\mathbf{r})$.

labelling is specified. We use the following contrived example to illustrate this idea.

Example 7.6 Consider the relation $\mathbf{r}$ in Figure 7.12 representing the probabilistic knowledge regarding the large pulmonary and cardiovascular domains in medical diagnosis. For simplicity we only represent attributes $Angina\ Pulse$ and $Bronchitus\ Smoker$ for the pulmonary and cardiovascular subdomains, respectively. In reaching a diagnosis a physician may not be interested in the fine details of the pulmonary and cardiovascular subdomains. Even though the pulmonary and cardiovascular subdomains are *not* conditionally independent given $Excercise$, the WBMVD $Excercise \approx\!\!> \approx\!\!> AnginaPulse$ is satisfied by $\mathbf{r}$. Thus, the subdomains may be coarsened as shown in Figure 7.13, where for example the nested distribution

$$\phi_{B_1=Y}(\phi_{B_2=Z}(\pi)) \;=\; \begin{array}{|cccc|}\hline B_1 & X & B_2 & A_{p(B_1XB_2)} \\ \hline \begin{array}{|cc|}\hline Y & A_{p(Y)} \\ \hline y_1 & a_1 \\ y_2 & a_2 \\ \hline \end{array} & x_1 & \begin{array}{|cc|}\hline Z & A_{p(Z)} \\ \hline z_1 & b_1 \\ z_2 & b_2 \\ \hline \end{array} & c \\ \hline \end{array}$$

Figure 7.11: The modified equivalence class $\phi_{B_1=Y}(\phi_{B_2=Z}(\pi))$ in $\phi_{B_1=Y}(\phi_{B_2=Z}(\mathbf{r}))$.

Bronchitis	Smoker	$A_{p(BronchitisSmoker)}$
never	no	0.8
slight	no	0.2

is represented as the Lung-value *average*. If necessary the details of either or both subdomains may be refined to reveal the internal structure.

$\mathbf{r} =$

Angina	Pulse	Exercise	Bronchitis	Smoker	A_p
slight	high	poor	slight	never	0.100
no	regular	average	never	no	0.240
no	regular	average	slight	no	0.060
yes	regular	average	never	no	0.160
yes	regular	average	slight	no	0.040
severe	irregular	average	severe	daily	0.200
no	low	good	never	occasional	0.125
no	low	good	never	never	0.075

Figure 7.12: A relation $\mathbf{r}(Angina\ Pulse\ Exercise\ Bronchitis\ Smoker)$.

Example 7.6 illustrates that complex-value probabilistic networks allow the joint probability distribution to be viewed at different granular levels. In this example, the depiction of the joint distribution in Figure 7.13 is coarser than the one in Figure 7.12.

$$\phi_{Heart=AnginaPulse}(\phi_{Lung=BronchitusSmoker}(\mathbf{r})) \quad = $$

Heart	Exercise	Lung	A_p
poor	poor	good	0.1
average	average	average	0.5
severe	average	severe	0.2
good	good	excellent	0.2

Figure 7.13: In $\phi_{Heart=AnginaPulse}(\phi_{Lung=BronchitusSmoker}(\mathbf{r}))$ the complex-values are represented as *values*, where $\mathbf{r}$ is the relation in Figure 7.12.

7.3 Coarsening the Joint Probability Distribution Locally

The results presented thus far have been defined with respect to the joint probability distribution $\mathbf{r}(R)$. In practice, however, $\mathbf{r}(R)$ is represented as a BAJD as defined by equation (4.19). That is, the joint probability distribution is factorized in terms of marginal distributions over an acyclic hypergraph. Thus, it is not entirely clear at this point whether the nest operation can be applied *locally* to marginal distributions with the same result as if applied to $\mathbf{r}(R)$ itself. This important issue is now considered.

Fortunately, the nest operation can be applied *locally* to a marginal distribution $\tau_{R_i}(\mathbf{r})$ in a BAJD with the same effect as if applied to $\mathbf{r}(R)$ itself.

Theorem 7.2 Let $\mathbf{r}(R)$ satisfy a BAJD $\otimes\mathcal{R}$. Let $Z \subseteq R_i$ be such that Z does not contain any attribute in any J-key of $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$. Then

$$\phi_{B=Z}(\mathbf{r}) \quad = \quad \tau_{R_1}(\mathbf{r}) \otimes \tau_{R_2}(\mathbf{r}) \otimes \ldots \otimes \phi_{B=Z}(\tau_{R_i}(\mathbf{r})) \otimes \ldots \otimes \tau_{R_n}(\mathbf{r}).$$

Proof: Suppose $\mathbf{r}(R)$ satisfies a BAJD $\otimes\mathcal{R}$. Without loss of generality, let $\mathbf{r}(XYZ) = \tau_{XY}(\mathbf{r}) \otimes \tau_{XZ}(\mathbf{r})$, as shown in Figure 7.14. Since the BAJD is satisfied, we have $c = a_1 + a_2 = b_1 + b_2$. Computing $\phi_{B=Z}(\mathbf{r})$ using the joint distribution $\mathbf{r}$ is shown in Figure 7.15 (top). On the other hand, $\tau_{XY}(\mathbf{r}) \otimes \phi_{B=Z}(\tau_{XZ}(\mathbf{r}))$, where $\phi_{B=Z}(\tau_{XZ}(\mathbf{r}))$ is computed using the marginal distribution $\tau_{XZ}(\mathbf{r})$, is illustrated in Figure 7.15 (bottom). As desired,

$$\phi_{B=Z}(\mathbf{r}) \quad = \quad \tau_{XY}(\mathbf{r}) \otimes \phi_{B=Z}(\tau_{XZ}(\mathbf{r})). \ \Box$$

$$\mathbf{r}(XYZ) = \begin{array}{|cccc|}\hline Y & X & Z & A_p \\ \hline y_1 & x_1 & z_1 & a_1b_1/c \\ y_1 & x_1 & z_2 & a_1b_2/c \\ y_2 & x_1 & z_1 & a_2b_1/c \\ y_2 & x_1 & z_2 & a_2b_2/c \\ \hline \end{array} = \begin{array}{|ccc|}\hline Y & X & A_{p(YX)} \\ \hline y_1 & x_1 & b_1 \\ y_2 & x_1 & b_2 \\ \hline \end{array} \otimes \begin{array}{|ccc|}\hline X & Z & A_{p(XZ)} \\ \hline x_1 & z_1 & a_1 \\ x_1 & z_2 & a_2 \\ \hline \end{array}$$

Figure 7.14: Relation $\mathbf{r}(XYZ)$ satisfies the BAJD $\otimes\{XY, XZ\}$.

This is an important result as otherwise the joint distribution would have to be computed before performing the nest operation. As in the traditional probabilistic model, it is not always feasible to store the joint probability distribution. Theorem 7.2 indicates that performing the nest operation locally on the economical representation of the joint distribution is equivalent to performing the nest operation on the joint distribution itself. We illustrate Theorem 7.2 with an example.

Example 7.7 The relation $\mathbf{r}(A_1A_2A_3)$ in Figure 7.16, satisfies the BAJD $\otimes\{R_1 = A_1A_2, R_2 = A_2A_3\}$. That is, $\mathbf{r} = \tau_{R_1}(\mathbf{r}) \otimes \tau_{R_2}(\mathbf{r})$. Computing $\phi_{B=A_3}(\mathbf{r})$ is shown in Figure 7.17, i.e., performing the nest operation on the joint distribution. The same relation is obtained by computing $\tau_{R_1}(\mathbf{r}) \otimes \phi_{B=A_3}(\tau_{R_2}(\mathbf{r}))$, as shown in Figure 7.18.

$$
\phi_{B=Z}(\mathbf{r}) \;=\; \begin{array}{|cccc|}
\hline
Y & X & B & A_{p(YXB)} \\
\hline
y_1 & x_1 & \begin{array}{|cc|} \hline Z & A_{p(Z)} \\ \hline z_1 & a_1 b_1/c = b_1 \\ z_2 & a_1 b_2/c = b_2 \\ \hline \end{array} & a_1(b_1+b_2)/c = a_1 \\
 & & & \\
y_2 & x_1 & \begin{array}{|cc|} \hline Z & A_{p(Z)} \\ \hline z_1 & a_2 b_1/c = b_1 \\ z_2 & a_2 b_2/c = b_2 \\ \hline \end{array} & a_2(b_1+b_2)/c = a_2 \\
\hline
\end{array}
$$

$$
\begin{array}{|ccc|}
\hline
Y & X & A_{p(YX)} \\
\hline
y_1 & x_1 & a_1 \\
y_2 & x_1 & a_2 \\
\hline
\end{array}
\;\otimes\;
\begin{array}{|ccc|}
\hline
X & B & A_{p(XB)} \\
\hline
x_1 & \begin{array}{|cc|} \hline Z & A_{p(Z)} \\ \hline z_1 & b_1 \\ z_2 & b_2 \\ \hline \end{array} & c \\
\hline
\end{array}
\;=\;
\begin{array}{|cccc|}
\hline
Y & X & B & A_{p(YXB)} \\
\hline
y_1 & x_1 & \begin{array}{|cc|} \hline Z & A_{p(Z)} \\ \hline z_1 & b_1 \\ z_2 & b_2 \\ \hline \end{array} & a_1 c/c = a_1 \\
 & & & \\
y_2 & x_1 & \begin{array}{|cc|} \hline Z & A_{p(Z)} \\ \hline z_1 & b_1 \\ z_2 & b_2 \\ \hline \end{array} & a_2 c/c = a_2 \\
\hline
\end{array}
$$

Figure 7.15: Computing $\phi_{B=Z}(\mathbf{r})$ using the joint distribution $\mathbf{r}(R)$ (top). Computing $\phi_{B=Z}(\tau_{XZ}(\mathbf{r}))$ using the marginal distribution $\tau_{XZ}(\mathbf{r})$. It can be seen that $\tau_{XY}(\mathbf{r}) \otimes \phi_{B=Z}(\tau_{XZ}(\mathbf{r})) = \phi_{B=Z}(\mathbf{r})$ (bottom).

$$
\begin{array}{|cccc|}
\hline
A_1 & A_2 & A_3 & A_p \\
\hline
0 & 0 & 0 & 0.375 \\
0 & 0 & 1 & 0.125 \\
1 & 0 & 0 & 0.225 \\
1 & 0 & 1 & 0.075 \\
1 & 1 & 0 & 0.200 \\
\hline
\end{array}
\;=\;
\begin{array}{|ccc|}
\hline
A_1 & A_2 & A_{p(A_1 A_2)} \\
\hline
0 & 0 & 0.5 \\
1 & 0 & 0.3 \\
1 & 1 & 0.2 \\
\hline
\end{array}
\;\otimes\;
\begin{array}{|ccc|}
\hline
A_2 & A_3 & A_{p(A_2 A_3)} \\
\hline
0 & 0 & 0.6 \\
0 & 1 & 0.2 \\
1 & 0 & 0.2 \\
\hline
\end{array}
$$

Figure 7.16: Relation $\mathbf{r}(A_1 A_2 A_3)$ satisfies the BAJD $\otimes\{R_1 = A_1 A_2, R_2 = A_2 A_3\}$.

$\phi_{B=A_3}(\mathbf{r}) \ =$

<table>
<tr><th>A_1</th><th>A_2</th><th>B</th><th>$A_{p(A_1A_2B)}$</th></tr>
<tr><td>0</td><td>0</td><td><table><tr><th>A_3</th><th>$A_{p(A_3)}$</th></tr><tr><td>0</td><td>0.75</td></tr><tr><td>1</td><td>0.25</td></tr></table></td><td>0.5</td></tr>
<tr><td>1</td><td>0</td><td><table><tr><th>A_3</th><th>$A_{p(A_3)}$</th></tr><tr><td>0</td><td>0.75</td></tr><tr><td>1</td><td>0.25</td></tr></table></td><td>0.3</td></tr>
<tr><td>1</td><td>1</td><td><table><tr><th>A_3</th><th>$A_{p(A_3)}$</th></tr><tr><td>0</td><td>1.0</td></tr></table></td><td>0.2</td></tr>
</table>

Figure 7.17: The relation $\phi_{B=A_3}(\mathbf{r})$.

A_1	A_2	$A_{p(A_1A_2)}$
0	0	0.5
1	0	0.3
1	1	0.2

$\otimes$

<table>
<tr><th>A_2</th><th>B</th><th>$A_{p(A_2B)}$</th></tr>
<tr><td>0</td><td><table><tr><th>A_3</th><th>$A_{p(A_3)}$</th></tr><tr><td>0</td><td>0.75</td></tr><tr><td>1</td><td>0.25</td></tr></table></td><td>0.8</td></tr>
<tr><td>1</td><td><table><tr><th>A_3</th><th>$A_{p(A_3)}$</th></tr><tr><td>0</td><td>1.0</td></tr></table></td><td>0.2</td></tr>
</table>

$=$

<table>
<tr><th>A_1</th><th>A_2</th><th>B</th><th>$A_{p(A_1A_2B)}$</th></tr>
<tr><td>0</td><td>0</td><td><table><tr><th>A_3</th><th>$A_{p(A_3)}$</th></tr><tr><td>0</td><td>0.75</td></tr><tr><td>1</td><td>0.25</td></tr></table></td><td>0.5</td></tr>
<tr><td>1</td><td>0</td><td><table><tr><th>A_3</th><th>$A_{p(A_3)}$</th></tr><tr><td>0</td><td>0.75</td></tr><tr><td>1</td><td>0.25</td></tr></table></td><td>0.3</td></tr>
<tr><td>1</td><td>1</td><td><table><tr><th>A_3</th><th>$A_{p(A_3)}$</th></tr><tr><td>0</td><td>1.0</td></tr></table></td><td>0.2</td></tr>
</table>

Figure 7.18: $\tau_{R_1}(\mathbf{r}) \otimes \phi_{B=A_3}(\tau_{R_2}(\mathbf{r})) = \phi_{B=A_3}(\mathbf{r})$.

7.4 Summary

In this chapter, we presented a preliminary look at complex-value probabilistic networks, namely, joint probability distributions in which the values of variables may themselves be distributions. We introduced the *nest* and *unnest* operators to coarsen and refine a joint distribution. It was explicitly demonstrated that inconsistency may arise due to the fact that the nest operation does not commute. WBMVD was shown to be a necessary and sufficient condition for ensuring consistency. (Probabilistic conditional independence is only a sufficient condition.) Finally, it was shown that the nest operation can be applied to a marginal distribution with the same effect as if applied to the joint distribution itself. This is an important result since it may not be feasible to store the joint probability distribution directly.

It should be emphasized that the notions introduced here were motivated by corresponding notions already developed in database theory. Long before the so-called *object-oriented Bayesian network* [49] was proposed in 1997, let alone the Bayesian network [66] in 1988, Makinouchi [58] introduced the idea of *nested* relations in 1977. Jaeschke and Schek [46] introduced *weak multivalued dependency* (WMVD) to characterize when two nest operations on single attributes would commute. This result was extended to arbitrary sets of attributes by Thomas [82]. Fischer and Van Gucht [31] further extended this characterization to more than two nest operations. The interaction between dependencies and the nest operation was examined by Fischer, Saxton, Thomas and Van Gucht [30].

The important point is that the initial research into developing complex-value Bayesian networks [49, 94] can take full advantage of the extensive literature on complex-value and object-oriented database systems, including [1, 2, 5, 30, 46, 58, 82].

It is also worth mentioning, that the notion of nested distributions can also be applied to Information Retrieval [100].

Chapter 8

THE IMPLICATION PROBLEM

In the design and implementation of probabilistic reasoning or database systems, a *crucial* issue to consider is the *implication problem*. The implication problem has been extensively studied in both relational databases, including [4, 8, 57, 60, 61], and in Bayesian networks [35, 36, 37, 65, 68, 78, 79, 88, 98]. The implication problem is to test whether a given input set $\sum$ of independencies logically implies another independency σ. Traditionally, *axiomatization* was studied in an attempt to solve the implication problem for data and probabilistic conditional independencies. In this approach, a finite set of inference axioms are used to generate symbolic proofs for a particular independency in a manner analogous to the proof procedures in mathematical logics.

In this paper, we use our Bayesian database model to present a comprehensive study of the implication problem for probabilistic conditional independencies. In particular, we examine four classes of independencies, namely:

(1a) BEMVD,

(1b) Conflict-free BEMVD,

(2a) BMVD,

(2b) Conflict-free BMVD.

Class (**1a**) is the *general* class of probabilistic conditional independencies called *Bayesian embedded multivalued dependency* (BEMVD) in our unified model. It is important to realize that (**1b**), (**2a**) and (**2b**) are *special* subclasses of (**1a**). Subclass (**2a**) contains those probabilistic conditional independencies involving *all* variables, called *Bayesian (nonembedded) multivalued dependency* (BMVD) in our approach. BMVD is also known as *full* probabilistic conditional independency [60], or *fixed context* probabilistic conditional independency [35]. Thus, (**2a**) is a subclass of probabilistic conditional independency since (**1a**) may include a set containing the mixture of embedded and nonembedded (full) probabilistic conditional independencies, whereas (**2a**) can only include sets of nonembedded (full) probabilistic conditional independencies. Nonembedded probabilistic conditional independencies are graphically represented by acyclic hypergraphs, while the mixture of embedded and nonembedded probabilistic conditional independencies are graphically represented by DAGs. However, as already mentioned, there are some sets of probabilistic conditional independencies which do *not* have a perfect-map. Thus, we use the term *conflict-free* for those sets of conditional independencies which do have a perfect-map. Consequently, class (**2b**) contains those sets of nonembedded (full) probabilistic conditional independencies which can be *faithfully* represented by a *single* acyclic hypergraph. Similarly, class (**1b**) contains those sets of embedded and nonembedded probabilistic conditional independencies which can be *faithfully* represented by a *single* DAG. It is important to realize that (**1b**) is a special subclass of (**1a**), and that (**2b**) is a special subclass of (**2a**) (and of course (**1a**)). The subclass (**1b**) of conflict-free BEMVDs is important since it is used in the construction of Bayesian networks. That is, subclass (**1b**) allows a human expert to indirectly specify a joint distribution as a product of conditional probability distributions. The subclass (**2b**) of conflict-free BMVDs is also important since it is used in the construction of Markov networks.

Let **C** denote an arbitrary set of probabilistic dependencies belonging to one of the above four classes, and **c** denote any dependency from the same class. We desire a means to test whether **C** logically implies **c**, namely:

$$\mathbf{C} \quad \models \quad \mathbf{c}. \tag{8.44}$$

In our approach, for any arbitrary sets $\mathbf{C}$ and $\mathbf{c}$ of *probabilistic* dependencies, there are *corresponding* sets C and c of *data* dependencies. More specifically, for each of the above four classes of probabilistic dependencies, there is a corresponding class of data dependencies in the relational database model:

(1a) EMVD,

(1b) Conflict-free EMVD,

(2a) MVD,

(2b) Conflict-free MVD,

as depicted in Figure 8.1. Since we advocate that the Bayesian database model is a *generalization* of the relational database model, an immediate question to answer is:

Do the implication problems coincide in these two database models?

That is, we would like to know whether the proposition:

$$\mathbf{C} \models \mathbf{c} \iff C \models c, \tag{8.45}$$

holds for the individual pairs (**1a**, 1a), (**1b**, 1b), (**2a**, 2a), and (**2b**, 2b). For example, we wish to know whether proposition (8.45) holds for the pair (BEMVD, EMVD), where $\mathbf{C}$ is a set of BEMVDs, $\mathbf{c}$ is any BEMVD, and C and c are the *corresponding* EMVDs.

We will show that

$$\{\text{ BMVDs }\} \models \mathbf{c} \iff \{\text{ MVDs }\} \models c$$

holds for the pair (BMVD, BMVD). Since (conflict-free BMVD, conflict-free MVD) are special classes of (BMVD, BMVD), respectively, proposition (8.45) is obviously true for the pair (**2b**, 2b), namely:

$$\{\text{ conflict-free BMVDs }\} \models \mathbf{c} \iff \{\text{ conflict-free MVDs }\} \models c.$$

It can also be shown that:

$$\{ \text{ conflict-free BEMVDs } \} \models \mathbf{c} \quad \Longleftrightarrow \quad \{ \text{ conflict-free EMVDs } \} \models c$$

holds for the pair (conflict-free BEMVD, conflict-free EMVD). However, it is important to note that proposition (8.45) is *not* true for the pair (BEMVD, EMVD). That is, the implication problem does not coincide for the general classes of probabilistic conditional independencies and embedded multivalued dependency. In [79], it was pointed out that:

$$\{ \text{ BEMVDs } \} \models \mathbf{c} \;\not\Leftarrow\; \{ \text{ EMVDs } \} \models c, \tag{8.46}$$

and

$$\{ \text{ BEMVDs } \} \models \mathbf{c} \;\not\Rightarrow\; \{ \text{ EMVDs } \} \models c. \tag{8.47}$$

(A double solid arrow in Figure 8.1 represents the fact that proposition (8.45) holds, while a double dashed arrow indicates that proposition (8.45) does not hold.) Since the implication problems do not coincide in the pair (BEMVD, EMVD), it was suggested in [79] that Bayesian networks are intrinsically *different* from relational databases. This remark, however, does not take into consideration one important issue, namely, the *solvability* of the implication problem for a particular class of dependencies.

The question naturally arises as to why the implication problem coincides for some classes of dependencies but not for others. One important result in relational databases is that the implication problem for the general class of EMVDs is *unsolvable* [41]. (By solvability, we mean there exists a method which in a finite number of steps can decide whether $\sum \models \sigma$ holds for an arbitrary instance $(\sum, \sigma)$ of the implication problem.) Therefore, the observation in Equation (8.46) is not too surprising, since EMVD is an *unsolvable* class of dependencies. Furthermore, the implication problem for the BEMVD class of probabilistic conditional independencies is also *unsolvable*. One immediate consequence of this result is the observation in Equation (8.47). Therefore, the fact that the implication problems in Bayesian networks and relational databases do not coincide is based on *unsolvable* classes of dependencies.

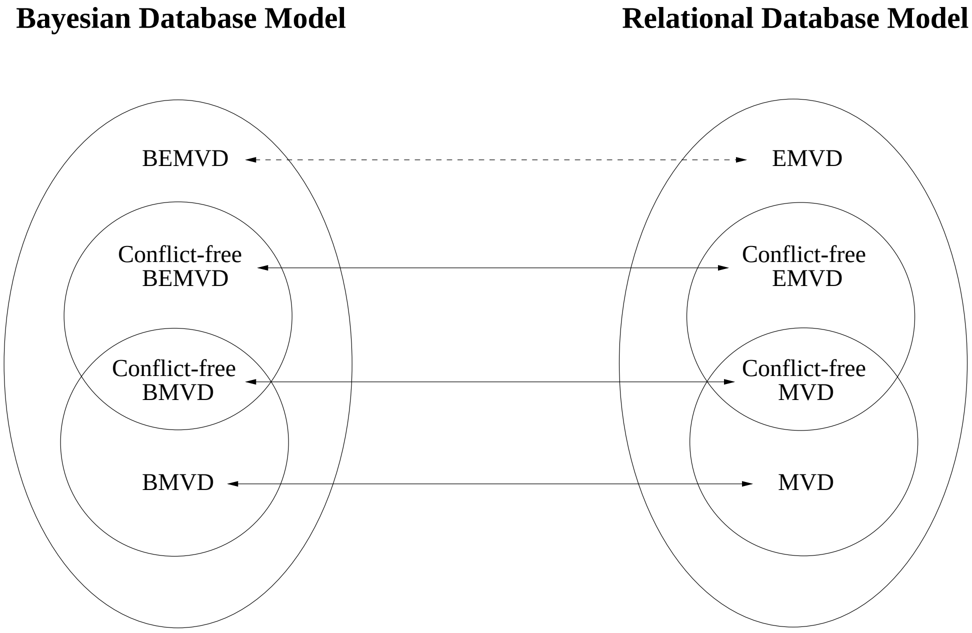

Figure 8.1: The four classes of *probabilistic* dependencies (BEMVD, conflict-free BEMVD, BMVD, conflict-free BMVD) traditionally found in the Bayesian database model are depicted on the left. The corresponding class of *data* dependencies (EMVD, conflict-free EMVD, MVD, conflict-free MVD) in the standard relational database model are depicted on the right.

This supports our argument that there is no *real* difference between Bayesian networks and standard relational databases in a practical sense, since only *solvable* classes of dependencies are useful in the design and implementation of both knowledge systems.

8.1 Subclasses of Probabilistic Conditional Independency

In this section, we emphasize the fact that probabilistic networks are constructed using special *conflict-free* subclasses within the general class of probabilistic conditional independencies. That is, Bayesian networks are not constructed using *arbitrary* sets of probabilistic conditional independencies, just as Markov networks are not constructed using *arbitrary* sets of nonembedded (full) probabilistic conditional independencies.

Probabilistic conditional independency is called *Bayesian embedded multivalued dependency* (BEMVD) in our approach. We define the general BEMVD class as follows:

$$\begin{aligned} (\mathbf{1a})\ \text{BEMVD} & \\ = & \ \{\ \mathbf{C} \mid \mathbf{C} \text{ is a set of probabilistic conditional independencies}\}. \end{aligned} \tag{8.48}$$

Bayesian networks are defined by a *directed acyclic graph* (DAG) and a corresponding set of conditional probability distributions. Such a DAG encodes probabilistic conditional independencies satisfied by a particular joint distribution. The method of *d-separation* [66] is used to infer conditional independencies from a DAG. For example, the conditional independency of A_1 and A_5 given $A_2A_3A_4$, i.e., $I(A_5, A_2A_3A_4, A_1)$, can be inferred from the DAG in Figure 3.1 using the d-separation method. However, it is important to realize that there are some sets of probabilistic conditional independencies that cannot be *faithfully* encoded by a single DAG.

Example 8.1 Consider the following set $\mathbf{C}$ of probabilistic conditional independencies on $\{A, B, C, D\}$:

$$\mathbf{C} \ = \ \{\ I(A, B, C),\ I(A, C, B),\ I(AB, C, D)\ \}. \tag{8.49}$$

There is no *single* DAG that can simultaneously encode the independencies in $\mathbf{C}$. □

Example 8.1 clearly indicates that Bayesian networks are defined only using a subclass of probabilistic conditional independencies. In order to label this subclass of independencies, we first recall the notion of perfect-map. A graphical structure is called a *perfect-map* [9, 66] of a given set $\mathbf{C}$ of probabilistic conditional independencies, if every conditional independency logically implied by $\mathbf{C}$ can be inferred from the graphical structure, and every conditional independency that can be inferred from the graphical structure is logically implied by $\mathbf{C}$. (We say $\mathbf{C}$ *logically implies* $\mathbf{c}$ and write $\mathbf{C} \models \mathbf{c}$, if whenever any distribution that satisfies all the conditional independencies in $\mathbf{C}$, then the distribution also satisfies $\mathbf{c}$.) A set $\mathbf{C}$ of probabilistic conditional independencies is called *conflict-free* if there exists a perfect-map DAG of $\mathbf{C}$.

We now can define the *conflict-free BEMVD* subclass used by Bayesian networks as follows:

$$\textbf{(1b)}\ \text{Conflict-free BEMVD} \tag{8.50}$$
$$= \ \{\ \mathbf{C} \mid \text{there exists a DAG which is a } \textit{perfect-map} \text{ of } \mathbf{C}\}.$$

It should be clear that a causal input list is a *minimum cover* [56] of a conflict-free set of conditional independencies. (A *causal input list* [67] or a *stratified protocol* [85] over a set $R = A_1A_2\ldots A_m$ of variables would contain precisely m conditional independency statements $I(X,Y,Z)$. For example, the set $\mathbf{C}$ of conditional independencies in Equation (3.12) is an example of a causal input list since $\mathbf{C}$ precisely defines the DAG in Figure 3.1. Since the conditional independency $I(A_5, A_2A_3A_4, A_1)$ can be inferred from the DAG in Figure 3.1, $\mathbf{C} \cup \{I(A_5, A_2A_3A_4, A_1)\}$ is still a conflict-free set but not a causal input list.)

As illustrated in Figure 8.1 (left), the main point is that the conflict-free BEMVD class is a subclass within the BEMVD class. For example, the set $\mathbf{C}$ of conditional independencies in Equation (8.49) belongs to the general BEMVD class in Equation (8.49) but does not belong to conflict-free BEMVD subclass in Equation (8.51).

Another subclass within the general BEMVD class are the *nonembedded* probabilistic conditional independencies. Nonembedded probabilistic conditional independency is also called *full* [60] or *fixed context* [35]. *Nonembedded* conditional independencies are those which involve *all* variables, i.e., $I(Y,X,Z)$ where $XYZ = R$.

Example 8.2 Let $R = \{A,B,C,D\}$. Consider the following set $\mathbf{C}$ of probabilistic conditional independencies:

$$\mathbf{C} \ = \ \{\ I(A,BC,D),\ I(A,B,C)\ \}.$$

The first independency $I(A,BC,D)$ is *nonembedded* (full) since $\{A,B,C,D\} = R$, but the second independency $I(A,B,C)$ is not full because $\{A,B,C\} \subset R$. $\Box$

The class of nonembedded probabilistic conditional independencies is called *Bayesian multivalued dependency* (BMVD) in our approach. We define the BMVD class as follows:

$$\textbf{(2a)}\ \text{BMVD} \tag{8.51}$$

$$= \{ \mathbf{C} \mid \mathbf{C} \text{ is a set of } \textit{nonembedded} \text{ probabilistic conditional independencies}\}.$$

Nonembedded (full) independencies are important since Markov networks do not reflect *embedded* conditional independencies.

The *separation* method [9] is used to infer nonembedded probabilistic conditional independencies from an acyclic hypergraph. Let $\mathcal{R}$ be an acyclic hypergraph on the set R of attributes and $X, Y \subseteq R$. The BMVD $X \Rightarrow\Rightarrow Y$ is inferred from the acyclic hypergraph $\mathcal{R}$, if and only if Y is the union of some disconnected components of the hypergraph $\mathcal{R}$ with the set of nodes X deleted.

Example 8.3 Consider the following acyclic hypergraph $\mathcal{R}$ on $R = ABCDEFGH$:

$$\mathcal{R} \;=\; \{R_1 = AB,\ R_2 = BCD,\ R_3 = DE,\ R_4 = DFG,\ R_5 = DFH\}.$$

Deleting the node D, we obtain:

$$\mathcal{R}' \;=\; \{R_1' = AB,\ R_2' = BC,\ R_3' = E,\ R_4' = FG,\ R_5' = FH\}.$$

The disconnected components in $\mathcal{R}'$ are:

$$S_1 = ABC, \qquad S_2 = E, \qquad S_3 = FGH.$$

By definition, the BMVDs $D \Rightarrow\Rightarrow ABC$, $D \Rightarrow\Rightarrow E$, $D \Rightarrow\Rightarrow FGH$, and $D \Rightarrow\Rightarrow ABCE$ can be inferred from $\mathcal{R}$. On the other hand, the BMVD $D \Rightarrow\Rightarrow BC$ is *not* inferred from $\mathcal{R}$ since BC is not equal to the union of some of the sets in $\{S_1, S_2, S_3\}$. □

Just as Bayesian networks are not constructed using arbitrary sets of BEMVDs, Markov networks are not constructed using *arbitrary* sets of BMVDs. That is, there are sets of nonembedded independencies which cannot be *faithfully* encoded by a single acyclic hypergraph.

Example 8.4 Consider the following set $\mathbf{C}$ of nonembedded probabilistic conditional independencies on $\{A, B, C\}$:

$$\mathbf{C} \;=\; \{\ I(A, B, C),\ I(A, C, B)\ \}. \qquad (8.52)$$

There is no *single* acyclic hypergraph that can simultaneously encode both nonembedded independencies in $\mathbf{C}$. □

Example 8.4 clearly indicates that Markov networks are defined only using a subclass of nonembedded probabilistic conditional independencies. The notion of conflict-free is again used to label this subclass. A set $\mathbf{C}$ of nonembedded probabilistic conditional independencies is called *conflict-free* if there exists an acyclic hypergraph which is a perfect-map of $\mathbf{C}$.

We now can define the *conflict-free BMVD* subclass used by Markov networks as follows:

$$\begin{aligned} (\mathbf{2b})\ & \text{Conflict-free BMVD} \\ =\ & \{\ \mathbf{C} \mid \text{there exists an acyclic hypergraph which is a } \textit{perfect-map} \text{ of } \mathbf{C}\}. \end{aligned} \tag{8.53}$$

As illustrated in Figure 8.1 (left), the main point is that the conflict-free BMVD class is a subclass within the BMVD class. For example, the set $\mathbf{C}$ of nonembedded probabilistic conditional independencies in Equation (8.52) belongs to the BMVD class in Equation (8.52) but not to the conflict-free BMVD class in Equation (8.54).

8.2 Problem Description

Before we study the implication problem in detail, let us first introduce some basic notions. Here we will use the terms *relation* and *joint probability distribution* interchangeably.

Let $\sum$ be a set of dependencies defined on a set of attributes R. By $SAT_R(\sum)$, we denote the set of all relations on (R) that satisfy all of the dependencies in $\sum$. We write $SAT_R(\sum)$ as $SAT(\sum)$ when R is understood, and $SAT(\sigma)$ for $SAT(\{\sigma\})$, where σ is a single dependency. We say $\sum$ *logically implies* σ, written $\sum \models \sigma$, if $SAT(\sum) \subseteq SAT(\sigma)$. In other words, σ is logically implied by $\sum$ if every relation which satisfies $\sum$ also satisfies σ. That is, there is no counter-example relation such that all of the dependencies in $\sum$ are satisfied but σ is not.

The *implication problem* is to test whether a given set $\sum$ of dependencies logically implies another dependency σ, namely,

$$\sum \ \models \ \sigma. \tag{8.54}$$

Clearly, the first question to answer is whether such a problem is *solvable*, i.e., whether there exists some method to provide a positive or negative answer for any given instance of the implication problem. We consider two methods for answering this question.

A method for testing implication is by axiomatization. An *inference axiom* is a rule that states if a relation satisfies certain dependencies, then it must satisfy certain other dependencies. Given a set $\sum$ of dependencies and a set of inference axioms, the *closure* of $\sum$, written $\sum^+$, is the smallest set containing $\sum$ such that the inference axioms cannot be applied to the set to yield a dependency not in the set. More specifically, the set $\sum$ *derives* a dependency σ, written $\sum \vdash \sigma$, if σ is in $\sum^+$. A set of inference axioms is *sound* if whenever $\sum \vdash \sigma$, then $\sum \models \sigma$. A set of inference axioms is *complete* if the converse holds, that is, if $\sum \models \sigma$, then $\sum \vdash \sigma$. In other words, saying a set of axioms are complete means that if $\sum$ logically implies the dependency σ, then $\sum$ derives σ. A sequence s of dependencies over R is a *derivation sequence* on $\sum$ if every dependency in s is either

(i) a member of $\sum$, or

(ii) follows from previous dependencies in s by an application of one of the given inference axioms.

Note that R is the set of attributes which appear in $\sum$. If the axioms are complete, to solve the implication problem we can simply compute $\sum^+$ and then test whether $\sigma \in \sum^+$.

Another approach for testing implication is to use a nonaxiomatic technique such as the *chase* algorithm [56]. The chase algorithm in relational database model is a powerful tool to obtain many nontrivial results. We will show that the chase algorithm can also be applied to the implication problem for a particular class of probabilistic conditional independencies.

The rest of this paper is organized as follows. Since nonembedded dependencies are best understood, we therefore choose to analyze the pair (BMVD, MVD), and their subclasses (conflict-free BMVD, conflict-free MVD) before the others. Next we consider the embedded dependencies. First we study the pair of (conflict-free BEMVD, conflict-free EMVD). The conflict-free BEMVD class has been studied extensively as these dependencies form the basis for the construction of Bayesian networks. Finally, we analyze the pair (BEMVD, EMVD). This pair subsumes all the other previously studied pairs. This pair is particularly important to our discussion here, as its implication problems are *unsolvable* in contrast to the other *solvable* pairs such as (BMVD, MVD) and (conflict-free BEMVD, conflict-free EMVD).

8.3 Nonembedded Dependency

In this section, we study the implication problem for the class of nonembedded (full) probabilistic conditional independency, called BMVD in our Bayesian database model. One way to demonstrate that the implication problem for BMVDs is solvable is to directly prove that a sound set of BMVD axioms are also *complete*. This is exactly the approach taken by Geiger and Pearl [35]. Here we take a different approach. Instead of directly demonstrating that the BMVD implication problem is solvable, we do it by establishing a one-to-one relationship between the implication problems of the pair (BMVD,MVD).

8.3.1 Full Multivalued Dependency in General

The MVD class of dependencies in the pair (BMVD,MVD) has been extensively studied in the standard relational database model. As mentioned before, MVD is the necessary and sufficient conditions for a lossless (binary) decomposition of a database relation. In this section, we review *two* methods for solving the implication problem of MVDs, namely, the *axiomatic* and *nonaxiomatic* methods.

(i) Axiomatization

It is well known [8] that MVDs have a finite complete axiomatization.

Theorem 8.1 The following inference axioms (MVD1)-(MVD7) are both sound and complete for multivalued dependencies (MVDs):

$(MVD1)$ If $Y \subseteq X$, then $X \rightarrow\rightarrow Y$.

$(MVD2)$ If $X \rightarrow\rightarrow Y$ and $Y \rightarrow\rightarrow Z$, then $X \rightarrow\rightarrow Z - Y$.

$(MVD3)$ If $X \rightarrow\rightarrow Y$, and $X \rightarrow\rightarrow Z$, then $X \rightarrow\rightarrow YZ$.

$(MVD4)$ If $X \rightarrow\rightarrow Y$ and $X \rightarrow\rightarrow Z$, then $X \rightarrow\rightarrow Y \cap Z$, $X \rightarrow\rightarrow Y - Z$.

$(MVD5)$ If $X \rightarrow\rightarrow Y$, then $XZ \rightarrow\rightarrow Y$.

$(MVD6)$ If $X \rightarrow\rightarrow Y$ and $YW \rightarrow\rightarrow Z$, then $XW \rightarrow\rightarrow Z - (YW)$.

$(MVD7)$ If $X \rightarrow\rightarrow Y$, then $X \rightarrow\rightarrow R - (XY)$.

It should perhaps be noted that axioms (MVD1) and (MVD2) form a *minimal* set [61], i.e., all other axioms can be derived from these two axioms. Axioms (MVD1)-(MVD7) are called *reflexivity*, *transitivity*, *union*, *decomposition*, *augmentation*, *pseudotransitivity*, and *complementation*, respectively.

The usefulness of a *sound* axiomatization lies in the ability to derive new dependencies from a given set.

Example 8.5 Consider the following set C of MVDs:

$$C = \{AB \rightarrow\rightarrow D,\ AE \rightarrow\rightarrow F,\ BD \rightarrow\rightarrow G\},$$

on the set of attributes $R = ABDEFG$. The following is a *derivation sequence* of the MVD $AB \rightarrow\rightarrow G$:

s_1. $AB \rightarrow\rightarrow D$ (given)

s_2. $AB \rightarrow\rightarrow B$ (MVD1)

s_3. $AB \rightarrow\rightarrow BD$ (MVD3) from s_1 and s_2

s_4. $BD \rightarrow\rightarrow G$ (given)

s_5. $AB \rightarrow\rightarrow G$ (MVD2) from s_3 and s_4.

Since the above derivation sequence $s = (s_1, s_2, s_3, s_4, s_5)$ is constructed based on sound axioms, this means that C *logically implies* $AB \rightarrow\rightarrow G$, written:

$$\{AB \rightarrow\rightarrow D,\ AE \rightarrow\rightarrow F,\ BD \rightarrow\rightarrow G\} \ \models \ AB \rightarrow\rightarrow G. \ \Box$$

The above example demonstrates that whenever a dependency is derived using sound axioms, the inferred dependency is logically implied by the given input set. However, if the inference axioms are *not* complete, then there is no guarantee that the axioms will derive *all* of the logically implied dependencies. Thus, in this approach the main task in solving the implication problem for a class of dependencies is to construct a set of complete inference axioms.

(ii) A Nonaxiomatic method - the Chase

Here we want to discuss an alternative method to solve the implication problem for the MVD class of dependencies. The discussion presented here follows closely the description given in [56].

We begin by examining what it means for a relation to decompose losslessly. Let r be a relation on R, and $R_1 \cup R_2 \cup \ldots \cup R_n = R$. We say relation r *decomposes losslessly* onto a database scheme $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ if

$$r \ = \ \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_n}(r). \tag{8.55}$$

It can be easily verified that:

$$r \ \subseteq \ \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_n}(r)$$

holds for any decomposition. In other words, every tuple $t \in r$ will also appear in the expression $\pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_n}(r)$. Thereby, for lossless decomposition it is sufficient to show:

$$r \ \supseteq \ \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_n}(r)$$

That is, to show that *every* tuple in the natural join of the projections is also a tuple in r. For example, the relation $r'(ABC)$ in Figure 2.8 does not decompose losslessly onto database scheme $\mathcal{R} = \{R_1 = AB, R_2 = BC\}$ since the tuple $t = <1\ 0\ 1>$ is in $\pi_{AB}(r) \bowtie \pi_{BC}(r)$ but is not an element of $r'(ABC)$.

The notion of lossless decomposition can be conveniently expressed by the *project-join mapping* $m_{\mathcal{R}}$ which is a function on relations on R defined by:

$$m_{\mathcal{R}}(r) = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_n}(r).$$

The important point to notice is that saying a relation $r(R)$ decomposes losslessly onto scheme $\mathcal{R}$ is the same as saying that $m_{\mathcal{R}}(r) = r$. Project-join mappings can be represented in tabular form called tableaux.

A *tableau* T is both a tabular means of representing a project-join mapping and a template for a relation r on R. Whereas a relation contains tuples of values, a tableau contains rows of subscripted variables (symbols). The a and b variables are called *distinguished* and *non-distinguished* variables, respectively. We restrict the variables in a tableau to appear in only one column. We make the further restriction that at most one distinguished variable may appear in any column. By convention, if the scheme of a tableau is $A_1A_2 \ldots A_m$, then the distinguished variable appearing in the A_i-column will be a_i. For example, a tableau T on scheme $R = A_1A_2A_3A_4$ is shown in Figure 8.2. We obtain a relation from the tableau by substituting domain values for variables. Let T be a tableau and let $V = \{\ a_1, a_2, \ldots, a_m, b_1, b_2, \ldots\}$ denote the set of its variables. A *valuation* ρ for T is a mapping from V to the Cartesian product $D_1 \times D_2 \times \ldots \times D_m$ such that $\rho(v)$ is in D_i when v is a variable appearing in the A_i-column. We extend the valuation from variables to rows and thence to the entire tableau. If $w = <v_1\ v_2 \ldots v_m>$ is a row in a tableau, we let $\rho(w) = < \rho(v_1)\ \rho(v_2) \ldots \rho(v_m) >$. We then let:

$$\rho(T) = \{\ \rho(w) \mid w \text{ is a row in } T\ \}.$$

Example 8.6 Consider the following valuation ρ:

$$\rho(a_1) = 1, \quad \rho(a_2) = 3, \quad \rho(a_3) = 5, \quad \rho(a_4) = 7,$$
$$\rho(b_1) = 4, \quad \rho(b_2) = 8, \quad \rho(b_3) = 2, \quad \rho(b_4) = 7, \quad \rho(b_5) = 4,$$

$$T = \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_4 \\ \hline a_1 & b_1 & a_3 & b_2 \\ b_3 & a_2 & a_3 & b_4 \\ a_1 & b_5 & a_3 & a_4 \\ \hline \end{array}$$

Figure 8.2: A tableau T on the scheme $A_1A_2A_3A_4$.

$$r = \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_4 \\ \hline 1 & 4 & 5 & 8 \\ 2 & 3 & 5 & 7 \\ 1 & 4 & 5 & 7 \\ \hline \end{array}$$

Figure 8.3: The relation r obtained as the result of applying ρ in Equation (8.56) to the tableau T in Figure 8.2.

The result of applying ρ to the tableau T in Figure 8.2 is the relation r in Figure 8.3. $\square$

Similar to a project-join mapping, a tableau T on scheme R can be interpreted as a function on relations $r(R)$. In this interpretation we require that T have a distinguished variable in every column. Let w_d be the row of all distinguished variables. That is, if $R = A_1A_2 \ldots A_m$, then $w_d = < a_1\ a_2\ \ldots\ a_m >$. Row w_d is not necessarily in T. If r is a relation on scheme R, we let:

$$T(r) = \{\ \rho(w_d) \mid \rho(T) \subseteq r\ \},$$

That is, if we find any valuation ρ that maps every row in T to a tuple in r, then $\rho(w_d)$ is in $T(r)$.

It is always possible to find a tableau $T_{\mathcal{R}}$ for representing a project-join mapping $m_{\mathcal{R}}$ defined by:

$$m_{\mathcal{R}}(r) = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_n}(r),$$

where $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$, and $R = R_1 \cup R_2 \cup \ldots \cup R_n$. The tableau $T_{\mathcal{R}}$ for $m_{\mathcal{R}}$ is defined as follows. The scheme for $T_{\mathcal{R}}$ is R. $T_{\mathcal{R}}$ has n rows, $w_1, w_2, \ldots, w_n$.

$$T = \begin{array}{|cccc|} \hline A_1 & A_2 & A_3 & A_4 \\ \hline a_1 & a_2 & b_1 & b_2 \\ b_3 & a_2 & a_3 & b_4 \\ b_5 & b_6 & a_3 & a_4 \\ \hline \end{array}$$

Figure 8.4: The tableau T on $R = A_1A_2A_3A_4$.

Row w_i has the distinguished variable a_j in the A_j-column exactly when $A_j \in R_i$. The remaining nondistinguished variables in w_i are unique and do not appear in any other row of $T_{\mathcal{R}}$. For example, let $\mathcal{R} = \{ R_1 = A_1A_2, R_2 = A_2A_3, R_3 = A_3A_4 \}$ and R_1, R_2, R_3 be a hypertree construction for $\mathcal{R}$. The tableau $T_{\mathcal{R}}$ for $m_{\mathcal{R}}$ is depicted in Figure 8.4.

Lemma 8.1 [56] Let $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ be a set of relation schemes, where $R = R_1R_2 \ldots R_n$. The project-join mapping $m_{\mathcal{R}}$ and the tableau $T_{\mathcal{R}}$ define the same function between relations $r(R)$. That is, $m_{\mathcal{R}}(r) = T_{\mathcal{R}}(r)$ for all $r(R)$.

Lemma 8.1 indicates that saying that a relation $r(R)$ decomposes losslessly onto scheme $\mathcal{R}$ is the same as saying that $T_{\mathcal{R}}(r) = r$.

Example 8.7 Consider the relation $r(A_1A_2A_3A_4)$ as shown on the left side of Figure 8.5. The valuation ρ, defined as:

$$\begin{array}{llllll} \rho(a_1) = 1, & \rho(a_2) = 3, & \rho(a_3) = 6, & \rho(a_4) = 8, & & \\ \rho(b_1) = 5, & \rho(b_2) = 7, & \rho(b_3) = 2, & \rho(b_4) = 8, & \rho(b_5) = 2, & \rho(b_6) = 3, \end{array}$$

indicates that $< 1\ 3\ 6\ 8 >$ is in $T_{\mathcal{R}}(r)$. All of $T_{\mathcal{R}}(r)$ is depicted on the right side of Figure 8.5. It is easily verified that applying the project-join mapping $m_{\mathcal{R}}$ to the relation $r(R)$ in Figure 8.5 also produces the relation on the right side of Figure 8.5. That is, $T_{\mathcal{R}}(r) = m_{\mathcal{R}}(r)$. □

The notion of what it means for two tableaux to be equivalent is now described. Let T_1 and T_2 be tableaux on scheme R. We write $T_1 \sqsubseteq T_2$ if $T_1(r) \subseteq T_2(r)$ for all

$r =$

A_1	A_2	A_3	A_4
1	3	5	7
1	4	5	7
2	3	6	8

$T(r) =$

A_1	A_2	A_3	A_4
1	3	5	7
1	3	6	8
1	4	5	7
2	3	5	7
2	3	6	8

Figure 8.5: A relation $r(A_1A_2A_3A_4)$ on the left. On the right, the relation $T(r)$, where T is the tableau in Figure 8.4.

relations $r(R)$. Tableaux T_1 and T_2 are *equivalent*, written $T_1 \equiv T_2$, if $T_1 \sqsubseteq T_2$ and $T_2 \sqsubseteq T_1$. That is, $T_1 \equiv T_2$ if $T_1(r) = T_2(r)$ for every relation $r(R)$. Let $SAT(C)$ denote the set of relations $r(R)$ that satisfy all the constraints in C. If T_1 and T_2 are tableaux on R, then we say T_1 is *contained* by T_2 on $SAT(C)$, written $T_1 \sqsubseteq_{SAT(C)} T_2$, if $T_1(r) \subseteq T_2(r)$ for every relation r in $SAT(C)$. We say T_1 and T_2 are *equivalent* on $SAT(C)$, written $T_1 \equiv_{SAT(C)} T_2$, if $T_1 \sqsubseteq_{SAT(C)} T_2$ and $T_2 \sqsubseteq_{SAT(C)} T_1$.

We now consider a method for modifying tableaux while preserving equivalence. A *M-rule* for a set C of AJDs is a means to modify an arbitrary tableau T to a tableau T' such that $T \equiv_{SAT(C)} T'$. Let $\mathcal{R} = \{R_1, R_2, \ldots, R_q\}$ be a set of relation schemes and let $\bowtie \mathcal{R}$ be a AJD on R. Let T be a tableau on R and let $w_1, w_2, \ldots, w_q$(not necessarily distinct) be rows of T that are joinable on $\mathcal{R}$ with result w. Applying the M-rule for $\bowtie \mathcal{R}$ to tableau T allows us to form the tableau:

$$T' = T \cup \{w\}.$$

If we view the tableau T as a relation, the generated row w can be expressed as:

$$w \quad = \quad w_1(R_1) \bowtie w_2(R_2) \bowtie \ldots \bowtie w_n(R_n). \tag{8.56}$$

Example 8.8 Let $C = \{ \bowtie \{A_1A_2,\ A_2A_3A_4\} \}$ and T be the tableau in Figure 8.6. Rows w_1 and w_2 are joinable on A_2. We can then apply the M-rule for $\bowtie \{A_1A_2, A_2A_3A_4\}$ in C to rows $w_1 = < a_1\ a_2\ b_1\ b_2 >$ and $w_2 = < b_3\ a_2\ a_3\ b_4 >$ of T to generate the new row:

$$w \quad = \quad w_1(A_1A_2) \bowtie w_2(A_2A_3A_4)$$

$$T = \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_4 \\ \hline a_1 & a_2 & b_1 & b_2 \\ b_3 & a_2 & a_3 & b_4 \\ b_5 & b_6 & a_3 & a_4 \\ \hline \end{array}$$

Figure 8.6: The tableau T on $R = A_1A_2A_3A_4$.

$$T' = \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_4 \\ \hline a_1 & a_2 & b_1 & b_2 \\ b_3 & a_2 & a_3 & b_4 \\ b_5 & b_6 & a_3 & a_4 \\ a_1 & a_2 & a_3 & b_4 \\ \hline \end{array}$$

Figure 8.7: The tableau $T' = T \cup \{< a_1\ a_2\ a_3\ b_4 >\}$, where T is the tableau in Figure 8.6.

$$\begin{aligned} &= \ < a_1\ a_2\ > \bowtie < a_2\ a_3\ b_4 > \\ &= \ < a_1\ a_2\ a_3\ b_4 > . \end{aligned}$$

Tableau $T' = T \cup \{w\}$ in Figure 8.7 is the result of this application. Even though rows $w = < a_1\ a_2\ a_3\ b_4 >$ and $w_3 = < b_5\ b_6\ a_3\ a_4 >$ are joinable on A_3, we cannot construct the new row $< a_1\ a_2\ a_3\ a_4 >$ since no M-rule exists in C which applies to attribute A_3. $\Box$

It is worth mentioning that M-rule is also applicable to MVDs since MVD is a special case of AJD.

Theorem 8.2 [56] Let $\mathcal{R} = \{R_1, R_2, \ldots, R_q\}$ and T' be the result of applying the M-rule for $\bowtie \mathcal{R}$ to tableau T. Tableaux T and T' are equivalent on $SAT(\bowtie \mathcal{R})$.

The *chase* algorithm can now be described. Given T and C, apply the M-rules associated with the AJDs in C, *until no further change is possible.* The resulting tableau, written $chase_C(T)$, is equivalent to T on all relations in $SAT(C)$, i.e., $T \equiv_{SAT(C)} chase_C(T)$, and $chase_C(T)$ considered as a relation is in $SAT(C)$.

Theorem 8.3 [56] $C \models \bowtie \mathcal{R}$ if and only if $chase_C(T_{\mathcal{R}})$ contains the row of all distinguished variables.

Theorem 8.3 states that the chase algorithm is equivalent to logical implication. We illustrate Theorem 8.3 with the following example.

Example 8.9 Suppose we wish to test the implication problem $C \models c$ on scheme $R = A_1A_2A_3A_4$, where $C = \{ \ A_2 \rightarrow\rightarrow A_1, \ \ A_3 \rightarrow\rightarrow A_4 \ \}$ is a set of MVDs and $c = \bowtie \{A_1A_2, \ A_2A_3, \ A_3A_4 \ \}$ is an AJD. We construct the initial tableau $T_{\mathcal{R}}$ in Figure 8.4 according to the database scheme $\mathcal{R}$ defined by c. Rows w_1 and w_2 are joinable on A_2. We can then apply the M-rule for $A_2 \rightarrow\rightarrow A_1$ in C to rows $w_1 = < a_1 \ a_2 \ b_1 \ b_2 >$ and $w_2 = < b_3 \ a_2 \ a_3 \ b_4 >$ of $T_{\mathcal{R}}$ to generate the new row:

$$\begin{aligned} w_4 &= w_1(A_1A_2) \bowtie w_2(A_2A_3A_4) \\ &= < a_1 \ a_2 \ a_3 \ b_4 > . \end{aligned}$$

Tableau $T_{\mathcal{R}} \cup \{w_4\}$ is depicted in Figure 8.7. Similarly, rows w_4 and w_3 are joinable on A_3. We can then the M-rule for $A_3 \rightarrow\rightarrow A_4$ in C to rows $w_4 = < a_1 \ a_2 \ a_3 \ b_4 >$ and $w_3 = < b_5 \ b_6 \ a_3 \ a_4 >$ to generate the new row:

$$\begin{aligned} w_d &= w_4(A_1A_2A_3) \bowtie w_3(A_3A_4) \\ &= < a_1 \ a_2 \ a_3 \ a_4 > \end{aligned}$$

as shown in Figure 8.9. Row w_d is the row of all distinguished variables. By Theorem 8.3, C logically implies c. That is, any relation that satisfies the MVDs in C must also satisfy the AJD c. $\Box$

It should be noted that the resulting tableau in the chase algorithm is *unique* regardless of the order in which the M-rules were applied.

Theorem 8.4 [56] The chase computation for a set of AJDs is a *finite Church-Rosser* replacement system. Therefore, $chase_C(T_{\mathcal{R}})$ is always a singleton set.

This completes the review of the implication problem for relational data dependencies.

$T_{\mathcal{R}} =$

A_1	A_2	A_3	A_4
a_1	a_2	b_1	b_2
b_3	a_2	a_3	b_4
b_5	b_6	a_3	a_4
a_1	a_2	a_3	b_4

Figure 8.8: Since $T_{\mathcal{R}}$ satisfies the MVD $A_2 \rightarrow\rightarrow A_1$ in C, by definition, rows w_1 and w_2 being joinable on A_2 imply that row $w_4 = < a_1\ a_2\ a_3\ b_4 >$ is also in $T_{\mathcal{R}}$.

$T_{\mathcal{R}} =$

A_1	A_2	A_3	A_4
a_1	a_2	b_1	b_2
b_3	a_2	a_3	b_4
b_5	b_6	a_3	a_4
a_1	a_2	a_3	b_4
a_1	a_2	a_3	a_4

Figure 8.9: Since $T_{\mathcal{R}}$ satisfies the MVD $A_3 \rightarrow\rightarrow A_4$ in C, by definition, rows w_4 and w_3 being joinable on A_3 imply that row $w_d = < a_1\ a_2\ a_3\ a_4 >$ is also in $T_{\mathcal{R}}$.

8.3.2 Full Probabilistic Conditional Independency in General

We now turn our attention to the class of nonembedded probabilistic conditional independency (BMVD) in the pair (BMVD, MVD). As in the MVD case, we will consider both the axiomatic and nonaxiomatic methods to solve the implication problem for the BMVD class of probabilistic dependencies. However, we first show an immediate relationship between the inference of BMVDs and that of MVDs.

Lemma 8.2 [59, 89] Let $\mathbf{C}$ be a set of BMVDs on R and $\mathbf{c}$ a single BMVD on R. Then

$$\mathbf{C} \models \mathbf{c} \implies C \models c,$$

where $C = \{X \rightarrow\rightarrow Y \mid X \Rightarrow\Rightarrow Y \in \mathbf{C}\}$ is the set of MVDs corresponding to the BMVDs in $\mathbf{C}$, and c is the MVD corresponding to the BMVD $\mathbf{c}$.

Proof: Suppose $\mathbf{C} \models \mathbf{c}$. We will prove the claim by contradiction. That is, suppose that $C \not\models c$. By definition, there exists a relation $r(R)$ such that $r(R)$ satisfies all of the MVDs in C, but $r(R)$ does not satisfy the MVD c. Let k denote the number of tuples in $r(R)$. We construct a probabilistic relation $\mathbf{r}(R)$ from $r(R)$ by appending the attribute A_p. For each of the k tuples in $\mathbf{r}(R)$, set $\mathbf{t}(A_p) = 1/k$. Thus, $\mathbf{r}(R)$ represents a *uniform* distribution. In the uniform case [59, 89], $\mathbf{r}(R)$ satisfies $\mathbf{C}$ if and only if $r(R)$ satisfies C. Again using the uniform case, $\mathbf{r}(R)$ does not satisfy $\mathbf{c}$ since $r(R)$ does not satisfy c. By definition, $\mathbf{C}$ does not logically imply $\mathbf{c}$, namely, $\mathbf{C} \not\models \mathbf{c}$. A contradiction to the initial assumption that $\mathbf{C} \models \mathbf{c}$. Therefore, $C \models c$. $\Box$

With respect to the pair (BMVD,MVD) of *nonembedded* dependencies, Lemma 8.2 indicates that the statement:

$$\mathbf{C} \models \mathbf{c} \implies C \models c$$

is a *tautology*. We now consider ways to solve the implication problem $\mathbf{C} \models \mathbf{c}$.

(i) BMVD Axiomatization

It can be easily shown that the following inference axioms for BMVDs are *sound*:

$(BMVD1)$ If $Y \subseteq X$, then $X \Rightarrow\Rightarrow Y$.

$(BMVD2)$ If $X \Rightarrow\Rightarrow Y$ and $Y \Rightarrow\Rightarrow Z$, then $X \Rightarrow\Rightarrow Z - Y$.

$(BMVD3)$ If $X \Rightarrow\Rightarrow Y$, and $X \Rightarrow\Rightarrow Z$, then $X \Rightarrow\Rightarrow YZ$.

$(BMVD4)$ If $X \Rightarrow\Rightarrow Y$ and $X \Rightarrow\Rightarrow Z$, then $X \Rightarrow\Rightarrow Y \cap Z$, $X \Rightarrow\Rightarrow Y - Z$.

$(BMVD5)$ If $X \Rightarrow\Rightarrow Y$, then $XZ \Rightarrow\Rightarrow Y$.

$(BMVD6)$ If $X \Rightarrow\Rightarrow Y$ and $YW \Rightarrow\Rightarrow Z$, then $XW \Rightarrow\Rightarrow Z - (YW)$,

$(BMVD7)$ If $X \Rightarrow\Rightarrow Y$, then $X \Rightarrow\Rightarrow R - (XY)$.

Axiom (BMVD1) holds trivially for any relation $\mathbf{r}(R)$ with $XY \subseteq R$. We now show that axiom (BMVD2) is sound. Recall that

$$X \Rightarrow\Rightarrow Y \iff X \Rightarrow\Rightarrow Y - X.$$

Thus, without loss of generality, let $R = XYZW$, where X, Y, Z and W are pairwise disjoint. By definition, the BMVDs $X \Rightarrow\Rightarrow Y$ and $Y \Rightarrow\Rightarrow Z$ mean

$$p(XYZW) = \frac{p(XY) \cdot p(XZW)}{p(X)}, \tag{8.57}$$

and

$$p(XYZW) = \frac{p(YZ) \cdot p(XYW)}{p(Y)}, \tag{8.58}$$

respectively. Computing the marginal distribution $p(XYZ)$ from both Equations (8.57) and (8.58), we respectively obtain:

$$p(XYZ) = \frac{p(XY) \cdot p(XZ)}{p(X)}, \tag{8.59}$$

and

$$p(XYZ) = \frac{p(YZ) \cdot p(XY)}{p(Y)}. \tag{8.60}$$

By Equations (8.59) and (8.60), we have:

$$\frac{p(XZ)}{p(X)} = \frac{p(YZ)}{p(Y)}. \tag{8.61}$$

By Equations (8.61) and (8.58), we obtain:

$$p(XYZW) = \frac{p(XZ) \cdot p(XYW)}{p(X)}. \tag{8.62}$$

Equation (8.62) is the definition of the BMVD $X \Rightarrow\Rightarrow Z$. The other axioms can be shown sound in a similar fashion.

Note that there is a one-to-one correspondence between the above inference rules for BMVDs and those MVD inference axioms (MVD1)-(MVD7) in Theorem 8.1. Since the BMVD axioms (BMVD1)-(BMVD7) are *sound*, it can immediately be shown that the implication problems coincide in the pair (BMVD,MVD).

Theorem 8.5 Given the *complete* axiomatization (MVD1)-(MVD7) for the MVD class. Then

$$\mathbf{C} \models \mathbf{c} \iff C \models c,$$

where $\mathbf{C}$ is a set of BMVDs, $C = \{X \rightarrow\rightarrow Y \mid X \Rightarrow\Rightarrow Y \in \mathbf{C}\}$ is the corresponding set of MVDs, and c is the MVD corresponding to a BMVD $\mathbf{c}$.

Proof: ($\Rightarrow$) Holds by Lemma 8.2.

($\Leftarrow$) Let $C \models c$. By Theorem 8.1, $C \models c$ implies that $C \vdash c$. That is, there exists a derivation sequence s of the MVD c by applying the MVD axioms to the MVDs in C. On the other hand, each MVD axiom has a corresponding BMVD axiom. This means there exists a derivation sequence $\mathbf{s}$ of the BMVD $\mathbf{c}$ using the BMVDs axioms on the BMVDs in $\mathbf{C}$, which parallels the derivation sequence s of the MVD c. That is, $\mathbf{C} \vdash \mathbf{c}$. Since the BMVD axioms are sound, $\mathbf{C} \vdash \mathbf{c}$ implies that $\mathbf{C} \models \mathbf{c}$. $\square$

Theorem 8.5 indicates that the implication problems coincide in the pair (BMVD,MVD), as indicated in Figure 8.1. The following result is an immediate consequence and is stated without proof.

Corollary 8.1 The axioms (BMVD1)-(BMVD7) are both *sound* and *complete* for the class of nonembedded probabilistic conditional independency.

By Corollary 8.1, it is not surprising then that Geiger and Pearl [35] showed that their alternative complete axioms for BMVDs were also complete for MVDs. In fact, the implication problem for BMVD has also been studied from an information-theoretic point of view. Yeung et al. [104] came to the same conclusion that the implication problem for nonembedded (full) conditional independencies is purely set-theoretic.

The main point of this section is to foster the notion that the Bayesian database model is intrinsically related to the standard relational database model. For example, by examining the implication problem for BMVD in terms of MVD, it is clear and immediate that the implication problems coincide in the pair (BMVD,MVD).

(ii) A Nonaxiomatic Method

We now present a *nonaxiomatic* method for testing the implication problem for nonembedded probabilistic conditional independencies. The standard chase algorithm can be modified for such a purpose by appropriately defining the manipulation of tableaux. However, we will then demonstrate that such a generalization is not necessary.

We briefly outline how a probabilistic chase can be formulated. A more complete description is given in [88]. The standard tableau T on a set of attributes $R = A_1A_2 \ldots A_m$ is augmented with attribute A_p. Each traditional row $w = < a_1a_2 \ldots a_m >$ is appended with probability symbol $p(a_1, a_2, \ldots, a_m)$. That is, a probabilistic tableau $\mathbf{T}$ contains rows $\mathbf{w} = < w, p(w) >$. In testing whether $\mathbf{C} \models \mathbf{c}$, we construct the initial tableau $\mathbf{T}_{\mathcal{R}}$ in the same fashion as in testing $C \models c$, where C and c are the corresponding MVDs, and $\mathcal{R}$ is the acyclic hypergraph corresponding to $\mathbf{c}$ (and c).

We now consider a method to modify probabilistic tableaux. We generalize the notion of M-rule for a MVD $X \rightarrow\rightarrow Y$ as follows. Let $\mathbf{T}$ be a probabilistic tableau on XYZ, $X \Rightarrow\Rightarrow Y$ a BMVD in a given set $\mathbf{C}$ of BMVDs, and $\mathbf{w}_1, \mathbf{w}_2$ be two *joinable* rows on X. A *B-rule* rule for the BMVD $X \Rightarrow\Rightarrow Y$ is a means to add the new row $\mathbf{w} = < w, p(w) >$ to $\mathbf{T}$, where w is defined in the usual sense according the M-rule for the corresponding MVD $X \rightarrow\rightarrow Y$, and the probability symbol $p(w)$ is defined as:

$$p(w) \quad = \quad \frac{p(w_1(XY)) \cdot p(w_2(XZ))}{p(w_1(X))}. \tag{8.63}$$

Example 8.10 Let $\mathbf{C} = \{A_2 \Rightarrow\Rightarrow A_1,\ A_3 \Rightarrow\Rightarrow A_4\}$ and consider the tableau $\mathbf{T}_{\mathcal{R}}$ in Figure 8.10. It can be seen that rows

$$\mathbf{w}_1 = < a_1\ a_2\ b_1\ b_2\ p(a_1a_2b_1b_2) >$$

and

$$\mathbf{w}_2 = < b_3\ a_2\ a_3\ b_4\ p(b_3a_2a_3b_4) >$$

are joinable on A_2. We can then apply the B-rule for the BMVD $A_2 \Rightarrow\Rightarrow A_1$ in $\mathbf{C}$ to generate a new row $\mathbf{w}_4 = < a_1, a_2, a_3, b_4, p(a_1, a_2, a_3, b_4) >$, where by Equation (8.63),

$$p(a_1, a_2, a_3, b_4) \quad = \quad \frac{p(a_1a_2) \cdot p(a_2a_3b_4)}{p(a_2)}.$$

The new row $\mathbf{w}_4$ is added to $\mathbf{T}_{\mathcal{R}}$ as shown in Figure 8.11 on the left. Similarly, rows $\mathbf{w}_3 = < b_3\ b_5\ a_3\ a_4\ p(b_3b_5a_3a_4) >$ and $\mathbf{w}_4 = < a_1\ a_2\ a_3\ b_4\ \frac{p(a_1a_2)p(a_2a_3b_4)}{p(a_2)} >$ are joinable on A_3. By Equation (8.63), the B-rule for the BMVD $A_3 \Rightarrow\Rightarrow A_4$ in $\mathbf{C}$ can be applied to rows $\mathbf{w}_3$ and $\mathbf{w}_4$ to generate the new row

$$\mathbf{w}_5 \quad = \quad < a_1\ a_2\ a_3\ a_4\ \frac{p(a_1a_2)p(a_2a_3)p(a_3a_4)}{p(a_2)p(a_3)} > .$$

The tableau $\mathbf{T}_{\mathcal{R}} \cup \{\mathbf{w}_4\} \cup \{\mathbf{w}_5\}$ is shown in Figure 8.10 on the left. $\Box$

The *probabilistic* chase algorithm is now introduced. Given $\mathbf{T}$ and $\mathbf{C}$, apply the B-rules associated with the BMVDs in $\mathbf{C}$, *until no further change is possible.* The resulting tableau, written $chase_{\mathbf{C}}(\mathbf{T})$, is equivalent to $\mathbf{T}$ on all relations in $SAT(\mathbf{C})$. That is, $\mathbf{T}(\mathbf{r}) = chase_{\mathbf{C}}(\mathbf{T})(\mathbf{r})$, for every probabilistic relation $\mathbf{r}$ satisfying every BMVD in $\mathbf{C}$. Furthermore, $chase_{\mathbf{C}}(\mathbf{T})$ considered as a relation is in $SAT(\mathbf{C})$. The next result indicates that the probabilistic chase algorithm is a *nonaxiomatic* method for testing the implication problem for the BMVD class.

$$\mathbf{T}_{\mathcal{R}} = \begin{array}{|ccccc|}\hline A_1 & A_2 & A_3 & A_4 & A_p \\ \hline a_1 & a_2 & b_1 & b_2 & p(a_1a_2b_1b_2) \\ b_3 & a_2 & a_3 & b_4 & p(b_3a_2a_3b_4) \\ b_5 & b_6 & a_3 & a_4 & p(b_5b_6a_3a_4) \\ \hline \end{array} \qquad T_{\mathcal{R}} = \begin{array}{|cccc|}\hline A_1 & A_2 & A_3 & A_4 \\ \hline a_1 & a_2 & b_1 & b_2 \\ b_3 & a_2 & a_3 & b_4 \\ b_5 & b_6 & a_3 & a_4 \\ \hline \end{array}$$

Figure 8.10: The initial tableau $\mathbf{T}_{\mathcal{R}}$ constructed according to the BAJD $\mathbf{c} = \otimes\{A_1A_2,\ A_2A_3,\ A_3A_4\}$ is shown on the left of the figure. The initial tableau $T_{\mathcal{R}}$ constructed according to the AJD $c = \bowtie \{A_1A_2,\ A_2A_3,\ A_3A_4\}$ is shown on the right.

$$\begin{array}{c|ccccc|}\hline & A_1 & A_2 & A_3 & A_4 & A_p \\ \hline \mathbf{w}_1 & a_1 & a_2 & b_1 & b_2 & p(a_1a_2b_1b_2) \\ \mathbf{w}_2 & b_3 & a_2 & a_3 & b_4 & p(b_3a_2a_3b_4) \\ \mathbf{w}_3 & b_5 & b_6 & a_3 & a_4 & p(b_5b_6a_3a_4) \\ \mathbf{w}_4 & a_1 & a_2 & a_3 & b_4 & \frac{p(a_1a_2)p(a_2a_3b_4)}{p(a_2)} \\ \mathbf{w}_5 & a_1 & a_2 & a_3 & a_4 & \frac{p(a_1a_2)p(a_2a_3)p(a_3a_4)}{p(a_2)p(a_3)} \\ \hline \end{array} \qquad \begin{array}{c|cccc|}\hline & A_1 & A_2 & A_3 & A_4 \\ \hline w_1 & a_1 & a_2 & b_1 & b_2 \\ w_2 & b_3 & a_2 & a_3 & b_4 \\ w_3 & b_5 & b_6 & a_3 & a_4 \\ w_4 & a_1 & a_2 & a_3 & b_4 \\ w_5 & a_1 & a_2 & a_3 & a_4 \\ \hline \end{array}$$

Figure 8.11: The tableaux obtained by adding the new rows $\mathbf{w}_4$ and $\mathbf{w}_5$ is shown on the left of the figure. The standard use of the corresponding M-rules is shown on the right.

Theorem 8.6 Let $\mathbf{C}$ be a set of BMVDs on $R = A_1A_2\ldots A_m$, and $\mathbf{c}$ be the BMVD $X \Rightarrow\Rightarrow Y$ on R. Then

$$\mathbf{C} \models \mathbf{c} \iff \ < a_1\ a_2 \ldots a_m\ p(a_1, a_2, \ldots, a_m) = \frac{p(xy)\cdot p(xz)}{p(x)} > \text{ is a row in } chase_{\mathbf{C}}(\mathbf{T}_{\mathcal{R}}),$$

where $\mathcal{R} = \{XY, XZ\}$ is the acyclic hypergraph corresponding to $\mathbf{c}$.

Proof: ($\Rightarrow$) We first show that the row of all distinguished variables $< a_1\ a_2 \ldots a_m$ $p(a_1, a_2, \ldots, a_m) >$ must appear in $chase_{\mathbf{C}}(\mathbf{T}_{\mathcal{R}})$. Given $\mathbf{C} \models \mathbf{c}$. By contradiction, suppose that the row $< a_1\ a_2 \ldots a_m\ p(a_1, a_2, \ldots, a_m) >$ does not appear in $chase_{\mathbf{C}}(\mathbf{T}_{\mathcal{R}})$. This means that the B-rules corresponding to the BMVDs in $\mathbf{C}$ cannot be applied to the joinable rows to generate the row $< a_1\ a_2 \ldots a_m\ p(a_1, a_2, \ldots, a_m) >$. This implies that the M-rules corresponding to the MVDs in $C = \{V \rightarrow\rightarrow W \mid V \Rightarrow\Rightarrow W \in \mathbf{C}\}$ cannot be applied to the joinable rows in $T_{\mathcal{R}}$ to generate the row $< a_1\ a_2 \ldots a_m >$ of all distinguished variables, where c is the MVD corresponding to the BMVD $\mathbf{c}$. By Theorem 8.3, the row $< a_1\ a_2 \ldots a_m >$ not appearing in $chase_C(T_{\mathcal{R}})$ means that $C \not\models c$, where $chase_C(T_{\mathcal{R}})$ is the result of chasing c under C. By Theorem 8.5, $C \not\models c$ implies that $\mathbf{C} \not\models \mathbf{c}$. A contradiction. Therefore, the row $< a_1\ a_2 \ldots a_m\ p(a_1, a_2, \ldots, a_m) >$ must appear in $chase_{\mathbf{C}}(\mathbf{T}_{\mathcal{R}})$.

We now show that $p(a_1, a_2, \ldots, a_m)$ can be factorized as desired. By contradiction, suppose that:

$$p(a_1, a_2, \ldots, a_m) \quad \neq \quad \frac{p(xy) \cdot p(xz)}{p(x)}.$$

This means that $chase_{\mathbf{C}}(\mathbf{T}_{\mathcal{R}})$, considered as a probabilistic relation, satisfies the BMVDs in $\mathbf{C}$ but does not satisfy the BMVD $\mathbf{c}$. By definition, $\mathbf{C} \not\models \mathbf{c}$. A contradiction. Therefore,

$$p(a_1, a_2, \ldots, a_m) \quad = \quad \frac{p(xy) \cdot p(xz)}{p(x)}.$$

($\Leftarrow$) Given the row $< a_1\ a_2 \ldots a_m\ p(a_1, a_2, \ldots, a_m) >$ appears in $chase_{\mathbf{C}}(\mathbf{T}_{\mathcal{R}})$. This means that the B-rules corresponding to the BMVDs in $\mathbf{C}$ can be applied to $\mathbf{T}_{\mathcal{R}}$ to generate the row $< a_1\ a_2 \ldots a_m\ p(a_1, a_2, \ldots, a_m) >$. This implies that the M-rules corresponding to the MVDs in $C = \{V \rightarrow\rightarrow W \mid V \Rightarrow\Rightarrow W \in \mathbf{C}\}$ can be applied to the joinable rows in $T_{\mathcal{R}}$ to generate the row $< a_1\ a_2 \ldots a_m >$ of all distinguished variables, where c is the MVD corresponding to the BMVD $\mathbf{c}$. By Theorem 8.3, the row $< a_1\ a_2 \ldots a_m >$ appearing in $chase_C(T_{\mathcal{R}})$ means that $C \models c$, where $chase_C(T_{\mathcal{R}})$ is the result of chasing c under C. By Theorem 8.5, $C \models c$ implies that $\mathbf{C} \models \mathbf{c}$. $\Box$

Theorem 8.6 indicates that $\mathbf{C} \models \mathbf{c}$ if and only if the row of all distinguished variables appears in $chase_{\mathbf{C}}(\mathbf{T}_{\mathcal{R}})$, i.e., $p(a_1, a_2, \ldots, a_m)$ can always be factorized according to the BMVD being tested.

As promised, we now show that developing a probabilistic chase algorithm for the Bayesian network model is not necessary because of the intrinsic relationship between the Bayesian and relational database models.

Theorem 8.7 Let $\mathbf{C}$ be a set of BMVDs on $R = A_1 A_2 \ldots A_m$, and $\mathbf{c}$ be a single BMVD on R. Then

$$\mathbf{C} \models \mathbf{c} \iff < a_1\ a_2 \ldots a_m > \text{ is a row in } chase_C(T_{\mathcal{R}}),$$

where $C = \{X \rightarrow\rightarrow Y \mid X \Rightarrow\Rightarrow Y \in \mathbf{C}\}$ is the set of MVDs corresponding to $\mathbf{C}$, c is the MVD corresponding to $\mathbf{c}$, and $chase_C(T_{\mathcal{R}})$ is the result of chasing c under C.

Proof: By Theorem 8.5,

$$\mathbf{C} \models \mathbf{c} \iff C \models c.$$

By Theorem 8.3,

$$C \models c \iff < a_1\ a_2 \ldots a_m > \text{ is a row in } chase_C(T_{\mathcal{R}}).$$

The claim follows immediately. □

Theorem 8.7 indicates that the standard chase algorithm, developed for testing the implication of *data* dependencies, can in fact be used to test the implication of nonembedded probabilistic conditional independency.

8.3.3 The Subclass of Conflict-free Full Dependencies

In this section, we examine the pair (conflict-free BMVD, conflict-free MVD). Recall that conflict-free BMVD is a subclass within the BMVD class. Similarly, conflict-free MVD is a subclass of MVD. Since we have already shown that the implication problems coincide in the pair (BMVD, MVD), obviously the implication problems coincide in the pair (conflict-free BMVD, conflict-free MVD) as mentioned in [60]. However, here we would like to take this opportunity to show that every conflict-free set $\mathbf{C}$ of BMVDs is *equivalent* to a Bayesian acyclic-join dependency (BAJD), $\otimes\mathcal{R}$. That is, whenever any probabilistic relation satisfies all the BMVDs in $\mathbf{C}$, then it also satisfies the BAJD $\otimes\mathcal{R}$, and vice versa.

Theorem 8.8 Let $\mathbf{C}$ denote a conflict-free set of BMVDs. Let $C = \{X \rightarrow\rightarrow Y \mid X \Rightarrow\Rightarrow Y \in \mathbf{C}\}$ be the conflict-free set of MVDs corresponding to $\mathbf{C}$. Then $\mathbf{C}$ and C have the same perfect-map $\mathcal{R}$.

Proof: The same separation method is used to infer both BMVDs and MVDs from acyclic hypergraphs. Therefore, for any given acyclic hypergraph $\mathcal{R}$, the BMVD $X \Rightarrow\Rightarrow Y$ can be inferred from $\mathcal{R}$ if and only if the corresponding MVD $X \rightarrow\rightarrow Y$ can be inferred from $\mathcal{R}$. Let $\mathcal{R}_1$ be the acyclic hypergraph which is a perfect-map of the conflict-free set $\mathbf{C}$ of BMVDs. Let $\mathcal{R}_2$ the perfect-map of C. We need to show that $\mathcal{R}_1$ and $\mathcal{R}_2$ denote the same acyclic hypergraph. Since a conflict-free set of MVDs has a unique perfect-map [9], it suffices to show that $\mathcal{R}_1$ is a perfect-map of the set C of MVDs.

Suppose $C \models X \rightarrow\rightarrow Y$. By Theorem 8.5, $\mathbf{C} \models \mathbf{c}$ if and only if $C \models c$. Thus, $\mathbf{C} \models X \Rightarrow\Rightarrow Y$. Since $\mathcal{R}_1$ is a perfect-map of $\mathbf{C}$, $X \Rightarrow\Rightarrow Y$ can be inferred from $\mathcal{R}_1$ using the separation method. By the above observation, this means that the MVD $X \rightarrow\rightarrow Y$ can be inferred from $\mathcal{R}_1$.

Suppose the MVD $X \rightarrow\rightarrow Y$ can be inferred from $\mathcal{R}_1$ using the separation method. By the above observation, this means that the BMVD $X \Rightarrow\Rightarrow Y$ can be inferred from $\mathcal{R}_1$. Since $\mathcal{R}_1$ is a perfect-map of $\mathbf{C}$, $\mathbf{C} \models X \Rightarrow\Rightarrow Y$. By Theorem 8.5, this implies that $C \models X \rightarrow\rightarrow Y$. $\Box$

Theorem 8.8 indicates that every conflict-free set of nonembedded probabilistic dependencies is equivalent to a Bayesian acyclic join dependency.

8.4 Embedded Dependency

We now examine the implication problem for *embedded* dependencies. As shown in Figure 8.1, the class of conflict-free BEMVD is a subclass of BEMVD, and conflict-free EMVD is a subclass of EMVD. We choose to first discuss the pair (conflict-free BEMVD, conflict-free EMVD) since the implication problems for these two classes are *solvable*. We then conclude our discussion by looking at the implication problem for the pair (BEMVD, EMVD) which represent the general classes of probabilistic conditional independency and embedded multivalued dependency.

8.4.1 The Subclass of Conflict-free Embedded Dependencies

Here we study the implication problem for the pair (conflict-free BEMVD, conflict-free EMVD). We begin the conflict-free BEMVD class.

The class of conflict-free BEMVDs plays a key role in the design of Bayesian networks. Recall that a set of BEMVDs is *conflict-free* if they can be faithfully represented by a *single* directed acyclic graph (DAG). We can use the *d-separation* method [66] to infer BEMVDs from a DAG. One desirable property of the conflict-free BEMVD class is that every conflict-free set of BEMVDs has a DAG as its *perfect-map*.

A set of conflict-free BEMVDs can be characterized as a join dependency, called *Bayesian embedded acyclic join dependency* (BEAJD). We say a relation $\mathbf{r}(R)$ satisfies the BEAJD, written $\times D$, if $\mathbf{r}(R)$ can be expressed as:

$$\mathbf{r}(R) \quad = \quad \mathbf{r}(A_1) \times \mathbf{r}(A_2|pa(A_2)) \times \mathbf{r}(A_3|pa(A_3)) \times \ldots \times \mathbf{r}(A_m|pa(A_m)),$$

where D is a DAG, $pa(A_m)$ is the parent set of A_m in D, and $\mathbf{r}(A_i|pa(A_i))$ is the probabilistic relation representing the *conditional probability distribution* $p(A_i|pa(A_i))$. It should be clear that a BEAJD in our terminology represents a *Bayesian network* [66].

Example 8.11 Let $\mathbf{r}(A_1A_2A_3A_4A_5A_6)$ be the relation representing the joint distribution $p(A_1, A_2, A_3, A_4, A_5, A_6)$ in Equation 3.10. Thus, $\mathbf{r}$ satisfies the BEAJD, $\times D$, where D is the DAG in Figure 3.1. That is, $\mathbf{r}(A_1A_2A_3A_4A_5A_6)$ can be losslessly decomposed as:

$$\begin{aligned}\mathbf{r}(A_1A_2A_3A_4A_5A_6) \quad &= \quad \mathbf{r}(A_1) \times \mathbf{r}(A_2|A_1) \times \mathbf{r}(A_3|A_1) \times \mathbf{r}(A_4|A_2A_3) \times \mathbf{r}(A_5|A_2A_3) \\ &\quad\quad \times \mathbf{r}(A_6|A_5). \ \Box\end{aligned}$$

The class of conflict-free BEMVD is a *special case* of the general BEMVD class, as shown in Figure 8.1. This special class of probabilistic dependencies has several desirable properties [66] including a complete axiomatization.

Theorem 8.9 [66] The class of *conflict-free BEMVD* has a *complete* axiomatization. Let X, Y, Z, W be pairwise disjoint subsets of R such that $XYZW = R$.

(CF-BEMVD1) $\quad X \Rightarrow\Rightarrow Y$ if and only if $X \Rightarrow\Rightarrow ZW$, (CF-BEMVD2) If $X \Rightarrow\Rightarrow YW \mid Z$, then $X \Rightarrow\Rightarrow Y \mid Z$,

(CF-BEMVD3) If $X \Rightarrow\Rightarrow YZ$, then $XZ \Rightarrow\Rightarrow Y$,

(CF-BEMVD4) If $X \Rightarrow\Rightarrow Y \mid Z$ and $XZ \Rightarrow\Rightarrow Y \mid W$, then $X \Rightarrow\Rightarrow Y \mid ZW$.

The axioms (CF-BEMVD1)-(CF-BEMVD4) are respectively called *symmetry*, *decomposition*, *weak union*, and *contraction*. Clearly, Theorem 8.9 indicates that the implication problem for the conflict-free BEMVD class is solvable.

We now turn our attention to the other class of dependency in the pair (conflict-free BEMVD, conflict-free EMVD), namely, conflict-free EMVD. In order to solve the implication problem for the class of *conflict-free* EMVD, we again use the method of drawing a one-to-one correspondence between the classes of conflict-free BEMVD and conflict-free EMVD.

Consider the following EMVD inference axioms, where X, Y, Z, W are pairwise disjoint subsets of R such that $XYZW = R$.

(CF-EMVD1) $X \rightarrow\rightarrow Y$ if and only if $X \rightarrow\rightarrow ZW$,

(CF-EMVD2) If $X \rightarrow\rightarrow YW \mid Z$, then $X \rightarrow\rightarrow Y \mid Z$,

(CF-EMVD3) If $X \rightarrow\rightarrow YZ$, then $XZ \rightarrow\rightarrow Y$,

(CF-EMVD4) If $X \rightarrow\rightarrow Y \mid Z$ and $XZ \rightarrow\rightarrow Y \mid W$, then $X \rightarrow\rightarrow Y \mid ZW$.

Inference axioms (CF-EMVD1), (CF-EMVD2) and (CF-EMVD3) are well-known properties of EMVDs [8]. More importantly, the database community was aware in 1979 [80] that the contraction axiom (CF-EMVD4) for EMVD, like the symmetry axiom (CF-EMVD1), is bidirectional:

(CF-EMVD4)$'$ $X \rightarrow\rightarrow Y \mid Z$ and $XZ \rightarrow\rightarrow Y \mid W$ if and only if $X \rightarrow\rightarrow Y \mid ZW$.

Thus, it should perhaps be mentioned here that the contraction axiom (CF-BEMVD4) for probabilistic conditional independence should be written properly as the following result indicates.

Lemma 8.3 The contraction axiom (CF-BEMVD4) in Pearl's semi-graphoid axioms [66] is bidirectional for probabilistic conditional independence:

(CF-BEMVD4)′ $\quad X \Rightarrow\Rightarrow Y \mid Z$ and $XZ \Rightarrow\Rightarrow Y \mid W$ if and only if $X \Rightarrow\Rightarrow Y \mid ZW$.

Proof: ($\Rightarrow$) This direction is the usual contraction axiom (CF-BEMVD4).
($\Leftarrow$) The BEMVD $X \Rightarrow\Rightarrow Y \mid Z$ can be obtained from $X \Rightarrow\Rightarrow Y \mid ZW$ using the decomposition axiom (CF-BEMVD2). Similarly, $XZ \Rightarrow\Rightarrow Y \mid W$ can be obtained from $X \Rightarrow\Rightarrow Y \mid ZW$ using the weak union axiom (CF-BEMVD3). $\Box$

We now return to our comparison of the implication problem for the pair (conflict-free BEMVD, conflict-free EMVD).

Theorem 8.10 Given the *complete* axiomatization (CF-BEMVD1)-(CF-BEMVD4) for the CF-BEMVD class. Then

$$\mathbf{C} \models \mathbf{c} \implies C \models c,$$

where $\mathbf{C}$ is a conflict-free set of BEMVDs, $C = \{X \rightarrow\rightarrow Y|Z \;\mid\; X \Rightarrow\Rightarrow Y|Z \in \mathbf{C}\ \}$ is the corresponding conflict-free set of EMVDs, and c is the EMVD corresponding to the BEMVD $\mathbf{c}$.

Proof: Suppose that $\mathbf{C} \models \mathbf{c}$. By Theorem 8.9, $\mathbf{C} \models \mathbf{c}$ implies that $\mathbf{C} \vdash \mathbf{c}$. That is, there exists a derivation sequence $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{c})$ of the BEMVD $\mathbf{c}$ from the conflict-free set $\mathbf{C}$ of BEMVDs using the inference axioms (CF-BEMVD1)-(CF-BEMVD4). The above discussion demonstrates that the corresponding inference axioms (CF-EMVD1)-(CF-EMVD4) are *sound* for deriving new EMVDs. This means that there is a derivation sequence $s = (s_1, s_2, \ldots, c)$ of the EMVD c from the conflict-free set C of EMVDs using the inference axioms (CF-EMVD1)-(CF-EMVD4), such that s parallels $\mathbf{s}$. That is, $C \vdash c$. We obtain our desired result since $C \vdash c$ implies that $C \models c$. $\Box$

Theorem 8.10 indicates that

$$\mathbf{C} \models \mathbf{c} \implies C \models c,$$

holds in the pair (conflict-free BEMVD, conflict-free EMVD). Conversely, we want to know whether

$$\mathbf{C} \models \mathbf{c} \impliedby C \models c,$$

is also true for this pair of dependencies. It was shown that there exists a complete axiomatization for conflict-free EMVDs [66].

Theorem 8.11 [66] The axioms (CF-EMVD1)-(CF-EMVD4) are *complete* for the class of *conflict-free* EMVD.

Based on this theorem, the following result is immediate.

Theorem 8.12 Given the complete axiomatization (CF-EMVD1)-(CF-EMVD4) for the CF-EMVD class. Then

$$\mathbf{C} \models \mathbf{c} \impliedby C \models c,$$

where C is a conflict-free set of BEMVDs, $\mathbf{C} = \{X \Rightarrow\Rightarrow Y|Z \mid X \to\to Y|Z \in C\}$ is the corresponding conflict-free set of EMVDs, and $\mathbf{c}$ is the BEMVD corresponding to the EMVD c.

Proof: The proof follows from a similar argument given in the proof of Theorem 8.10. □

The important point to remember is that Theorems 8.10 and 8.12 together indicate that

$$\mathbf{C} \models \mathbf{c} \iff C \models c \tag{8.64}$$

holds for the pair (conflict-free BEMVD, conflict-free EMVD). As already mentioned, the class of conflict-free BEMVDs is the basis for constructing a Bayesian network. However, conflict-free EMVDs have traditionally been ignored in relational databases. The above observation indicates that the special class of *conflict-free* EMVDs is equally useful in the design and implementation of traditional database applications.

8.4.2 Embedded Dependencies in General

The last pair of dependencies we study is (BEMVD, EMVD). All of the previously studied classes of probabilistic dependencies are a subclass of BEMVD (probabilistic conditional independency). Similarly, EMVD is the general class of multivalued dependencies. Before we study BEMVDs, we first examine the implication problem for EMVDs.

Theorem 8.13 [64, 75] The general EMVD class does not have a *finite* complete axiomatization.

The chase algorithm also does *not* solve the implication problem for the EMVD class. If $C \not\models c$, then the chase algorithm can continue forever. The reason is that, by definition, a M-rule for an EMVD $X \rightarrow\rightarrow Y|Z$ in a given set C of EMVDs would only generate a *partial* new row. To modify the chase algorithm for EMVDs, the partial row is padded out with *unique* nondistinguished variables in the remaining attributes. Thus, in using an EMVD the chase adds a new row containing new symbols. This enables further applications of EMVDs in C, which will add more new rows with new symbols, and this process does not terminate and can continue forever. (With MVDs, on the other hand, a new row consists only of existing symbols meaning that eventually there are no new rows to generate.)

The chase algorithm, however, is a *proof procedure* for implication of EMVDs [29]. This means that if $C \models c$, then the row of all distinguished variables will eventually be generated. The generation of the row of all a's can be used as a stopping criterion.

Example 8.12 Suppose we wish to verify that $C \models c$, where C and c are defined as:

$$\{A_1 \rightarrow\rightarrow A_3 \mid A_4,\ A_2 \rightarrow\rightarrow A_3 \mid A_4,\ A_3A_4 \rightarrow\rightarrow A_1 \mid A_2\} \models A_1A_2 \rightarrow\rightarrow A_3. \quad (8.65)$$

The initial tableau $T_{\mathcal{R}}$ is constructed according to c as shown in Figure 8.12 (left). We can apply the M-rule corresponding to the EMVD $A_1 \rightarrow\rightarrow A_3 \mid A_4$ in C to joinable rows $w_1 = < a_1\ a_2\ a_3\ b_1 >$ and $w_2 = < a_1\ a_2\ b_2\ a_4 >$ to generate the new row $w_3 = < a_1\ b_3\ a_3\ a_4 >$ as shown in Figure 8.12 (right). Similarly, we can apply the M-rule corresponding to the EMVD $A_2 \rightarrow\rightarrow A_3 \mid A_4$ in C to joinable

$$T_{\mathcal{R}} = \begin{array}{|cccc|} \hline A_1 & A_2 & A_3 & A_4 \\ \hline a_1 & a_2 & a_3 & b_1 \\ a_1 & a_2 & b_2 & a_4 \\ \hline \end{array} \qquad \begin{array}{c|cccc|} \hline & A_1 & A_2 & A_3 & A_4 \\ \hline w_1 & a_1 & a_2 & a_3 & b_1 \\ w_2 & a_1 & a_2 & b_2 & a_4 \\ w_3 & a_1 & b_3 & a_3 & a_4 \\ w_4 & b_4 & a_2 & a_3 & a_4 \\ w_5 & a_1 & a_2 & a_3 & a_4 \\ \hline \end{array}$$

Figure 8.12: On the left, the initial tableau $T_{\mathcal{R}}$ constructed according to the EMVD c defined as $A_1A_2 \to\to A_3$. The row $< a_1\ a_2\ a_3\ a_4 >$ of all distinguished variables appears in $chase_{\mathbf{C}}(T_{\mathcal{R}})$ indicating $C \models c$.

rows $w_1 = < a_1\ a_2\ a_3\ b_1 >$ and $w_2 = < a_1\ a_2\ b_2\ a_4 >$ to generate the new row $w_4 = < b_4\ a_2\ a_3\ a_4 >$ as shown in Figure 8.12 (right). Finally, we can obtain the row $< a_1\ a_2\ a_3\ a_4 >$ of all distinguished variables by applying the M-rule corresponding to the MVD $A_3A_4 \to\to A_1 \mid A_2$ in C to joinable rows w_3 and w_4. Therefore, $C \models c$. $\square$

For over a decade, considerable effort was put forth in the database research community to show that the implication problem for EMVDs is in fact *unsolvable.* Herrmann [41] recently succeeded in showing this elusive result.

Theorem 8.14 [41] *The implication problem for the general EMVD class is unsolvable.*

Theorem 8.14 is important since it indicates that *no* method exists for deciding the implication problem for the EMVD class. This concludes our discussion on the EMVD class.

We now study the corresponding class of probabilistic dependencies in the pair (BEMVD, EMVD), namely, the general class of probabilistic conditional independency. Pearl [66] conjectured that the semi-graphoid axioms (CF-BEMVD1)-(CF-BEMVD4) could solve the implication problem for probabilistic conditional independency (BEMVD) in general. This conjecture was refuted [79, 98].

Theorem 8.15 [79, 98] *BEMVDs do not have a finite complete axiomatization.*

$$r(A_1A_2A_3A_4) \ = $$

A_1	A_2	A_3	A_4
0	0	0	0
0	0	0	1
0	1	0	0
1	0	0	0
1	1	0	0
1	1	1	0

Figure 8.13: Relation r satisfies all of the EMVDs in C but does not satisfy the EMVD c, where C and c are defined in Example 8.13. Therefore, $C \not\models c$.

Theorem 8.15 indicates that it is not possible to solve the implication problem for the BEMVD class using a finite axiomatization. This result does not rule out the possibility that some alternative method exists for solving this implication problem.

As with the other classes of probabilistic dependencies, we now examine the relationship between $\mathbf{C} \models \mathbf{c}$ and $C \models c$ in the pair (BEMVD,EMVD). The following two examples [79] indicate that the implication problems for EMVD and BEMVD do not coincide.

Example 8.13 Consider the set $\mathbf{C} = \{A_3A_4 \Rightarrow\Rightarrow A_1|A_2,\ A_1 \Rightarrow\Rightarrow A_3|A_4,\ A_2 \Rightarrow\Rightarrow A_3|A_4,\ \emptyset \Rightarrow\Rightarrow A_1|A_2\}$ of BEMVDs, and $\mathbf{c}$ the single BEMVD $\emptyset \Rightarrow\Rightarrow A_3|A_4$. In [78], Studeny showed that $\mathbf{C} \models \mathbf{c}$. Now consider the set $C = \{X \rightarrow\rightarrow Y|Z \ \mid \ X \Rightarrow\Rightarrow Y|Z \in \mathbf{C}\}$ of EMVDs corresponding to the set $\mathbf{C}$ of BEMVDs, and the single EMVD $\emptyset \rightarrow\rightarrow A_3|A_4$ corresponding to the BEMVD $\mathbf{c}$. Consider the relation $r(A_1A_2A_3A_4)$ in Figure 8.13. It can be verified that $r(A_1A_2A_3A_4)$ satisfies all of the EMVDs in C but does not satisfy the EMVD c. That is, $C \not\models c$. $\Box$

Example (8.13) indicates that

$$\mathbf{C} \models \mathbf{c} \not\Rightarrow \ C \models c. \tag{8.66}$$

Example 8.14 Consider the set $C = \{A_1 \rightarrow\rightarrow A_3 \mid A_4,\ A_2 \rightarrow\rightarrow A_3 \mid A_4,\ A_3A_4 \rightarrow\rightarrow A_1 \mid A_2\}$ of EMVDs, and let c be the single EMVD $A_1A_2 \rightarrow\rightarrow A_3$. The chase algorithm was used in Example 8.12 to show that $C \models c$. Now consider the corresponding

$$\mathbf{r}(A_1A_2A_3A_4) \;=\; \begin{array}{|ccccc|} \hline A_1 & A_2 & A_3 & A_4 & A_p \\ \hline 0 & 0 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 1 & 0.2 \\ 0 & 0 & 1 & 0 & 0.2 \\ 0 & 0 & 1 & 1 & 0.1 \\ 0 & 1 & 1 & 1 & 0.1 \\ 1 & 0 & 1 & 1 & 0.1 \\ 1 & 1 & 1 & 1 & 0.1 \\ \hline \end{array}$$

Figure 8.14: Relation $\mathbf{r}$ satisfies all of the BEMVDs in $\mathbf{C}$ but does not the BEMVD $\mathbf{c}$, where $\mathbf{C}$ and $\mathbf{c}$ are defined in Example 8.14. Therefore, $\mathbf{C} \not\models \mathbf{c}$.

set of BEMVDs $\mathbf{C} = \{A_1 \Rightarrow\Rightarrow A_3 \mid A_4,\ A_2 \Rightarrow\Rightarrow A_3 \mid A_4,\ A_3A_4 \Rightarrow\Rightarrow A_1 \mid A_2\}$ and $\mathbf{c}$ is the BMVD $A_1A_2 \Rightarrow\Rightarrow A_3$. It is easily verified that relation $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 8.14 satisfies all of the BEMVDs in $\mathbf{C}$ but does not satisfy the BEMVD $\mathbf{c}$. Therefore, $\mathbf{C} \not\models \mathbf{c}$. $\Box$

Example 8.14 indicates that

$$\mathbf{C} \models \mathbf{c} \not\Leftarrow \;\; C \models c. \tag{8.67}$$

In the next section, we attempt to answer why the implication problems coincide for some classes but not for others.

8.5 The Role of Solvability

We have shown that $\mathbf{C} \models \mathbf{c} \iff C \models c$ holds:

for the pair (BMVD, MVD) in Theorem 8.5,

for the pair (Conflict-free BMVD, Conflict-free MVD) in Theorem 8.5,

for the pair (Conflict-free BEMVD, Conflict-free EMVD) in Equation 8.64.

That is, the implication problems coincide in these three pairs of classes. However, Examples (8.13) and (8.14) demonstrate that

$$\mathbf{C} \models \mathbf{c} \not\Leftrightarrow C \models c \qquad \text{for the pair (BEMVD, EMVD).}$$

The implication problems for each class in the first three pairs are *solvable*. However, the implication problem for the general EMVD class in the pair (BEMVD, EMVD) is *unsolvable*. These observations lead us to make the following conjecture.

Conjecture 8.1 Consider any pair (BD-class, RD-class), where BD-class is a class of probabilistic dependencies in the Bayesian database model and RD-class is the corresponding class of data dependencies in the relational database model. Let $\mathbf{C}$ be a set of probabilistic dependencies chosen from BD-class, and $\mathbf{c}$ a single dependency in BD-class. Let C and c denote the corresponding set of data dependencies of $\mathbf{C}$ and $\mathbf{c}$, respectively, in RD-class.

(i) If the implication problem is *solvable* for the class BD-class, then

$$\mathbf{C} \models \mathbf{c} \quad \Longrightarrow \quad C \models c.$$

(ii) If the implication problem is *solvable* for the class RD-class, then

$$\mathbf{C} \models \mathbf{c} \quad \Longleftarrow \quad C \models c.$$

In [79], Studeny studied the relationship between the implication problems in the pair (BEMVD, EMVD), namely, probabilistic conditional independency (BEMVD) and embedded multivalued dependency. Based on Conjecture 8.1 (i), his observation:

$$\mathbf{C} \models \mathbf{c} \;\not\Longrightarrow\; C \models c,$$

would indicate that the implication problem for the general class of probabilistic conditional independency is *unsolvable*. Similarly, based on Conjecture 8.1 (ii), his observation:

$$\mathbf{C} \models \mathbf{c} \;\not\Longleftarrow\; C \models c,$$

would indicate that the implication problem for the class of EMVD is *unsolvable*.

A successful proof of this conjecture would provide a proof that the implication problems for EMVD and BEMVD (probabilistic conditional independency) are both unsolvable.

8.6 New Solvable Classes of Dependencies

In this section, we introduce two new solvable classes of probabilistic dependencies. The first is WBMVD alone, while the other is the mixture of BMVD and WBMVD together.

The complete axiomatization for WBMVD is derived from the complete axiomatization of *weak multivalued dependency* (WMVD) [46] in database theory.

Theorem 8.16 [31] The axioms WMVD1-WMVD4 are complete for the implication of WMVDs.

$(WMVD1)$	If $Y \subseteq X$, then $X \rightsquigarrow\!\!\rightsquigarrow Y$.
$(WMVD2)$	If $X \rightsquigarrow\!\!\rightsquigarrow Y$, then $X \rightsquigarrow\!\!\rightsquigarrow Y - Z$ and $X \rightsquigarrow\!\!\rightsquigarrow YZ$, where $Z \subseteq X$.
$(WMVD3)$	If $X \rightsquigarrow\!\!\rightsquigarrow Y$, then $XZ \rightsquigarrow\!\!\rightsquigarrow Y$.
$(WMVD4)$	If $X \rightsquigarrow\!\!\rightsquigarrow Y$, then $X \rightsquigarrow\!\!\rightsquigarrow R - (XY)$.

Now consider the corresponding axioms for the class of WBMVD.

Theorem 8.17 The axioms WBMVD1-WBMVD4 are complete for the implication of WBMVDs.

$(WBMVD1)$	If $Y \subseteq X$, then $X \approx\!\!>\!\approx\!\!> Y$.
$(WBMVD2)$	If $X \approx\!\!>\!\approx\!\!> Y$, then $X \approx\!\!>\!\approx\!\!> Y - Z$ and $X \approx\!\!>\!\approx\!\!> YZ$, where $Z \subseteq X$.
$(WBMVD3)$	If $X \approx\!\!>\!\approx\!\!> Y$, then $XZ \approx\!\!>\!\approx\!\!> Y$.
$(WBMVD4)$	If $X \approx\!\!>\!\approx\!\!> Y$, then $X \approx\!\!>\!\approx\!\!> R - (XY)$.

The proof of Theorem 8.17 can be shown following the one given in [31]. The important point is that Theorem 8.16 and Theorem 8.17 indicate that the implication problems coincide for these two *solvable* classes of dependencies, as illustrated in Figure 1.1.

It should be noted that there is no transitivity axiom for WBMVDs. The WBMVDs $A_1 \approx\!\!>\approx\!\!> A_2$ and $A_2 \approx\!\!>\approx\!\!> A_3$ are satisfied by $\mathbf{r}(A_1A_2A_3A_4)$ in Figure 6.6 (left), but $A_1 \approx\!\!>\approx\!\!> A_3$ is *not*.

The second new class we study is the mixture of BMVDs and WBMVDs together. Again, the motivation stems from a complete axiomatization for the mixture of MVDs and WMVDs together in database theory.

Van Gucht and Fischer [83] have shown that the class of MVDs and WMVDs together have a complete axiomatization.

Theorem 8.18 [83] The axioms MVD1-MVD7, WMVD1-WMVD4, (MVD & WMVD1) and (MVD & WMVD2) form a complete axiomatization for the class of MVDs and WMVDs:

$$(MVD\&WMVD1) \qquad \text{If } X \rightarrow\rightarrow Y, \text{ then } X \leadsto\leadsto Y.$$

$$(MVD\&WMVD2) \qquad \text{If } XZ_1 \leadsto\leadsto Y, XZ_2 \leadsto\leadsto Y, \text{ and } XY \rightarrow\rightarrow Z_1, \text{ then } X \leadsto\leadsto Y,$$

where X, Y, Z_1, Z_2 are pairwise disjoint subsets of R such that $XYZ_1Z_2 = R$.

We first show that the axioms (MVD & WMVD1) and (MVD & WMVD2) are sound under a probabilistic interpretation.

Lemma 8.4 Axioms (BMVD & WBMVD1) and (BMVD & WBMVD2) are *sound* for the class of BMVDs and WBMVDs together.

$$(BMVD\&WBMVD1) \qquad \text{If } X \Rightarrow\Rightarrow Y, \text{ then } X \leadsto\leadsto Y.$$

$$(BMVD\&WBMVD2) \qquad \text{If } XZ_1 \approx\!\!>\approx\!\!> Y, XZ_2 \approx\!\!>\approx\!\!> Y, \text{ and } XY \Rightarrow\Rightarrow Z_1, \text{ then } X \approx\!\!>\approx\!\!> Y,$$

where X, Y, Z_1, Z_2 are pairwise disjoint subsets of R such that $XYZ_1Z_2 = R$.

Proof: Axiom (BMVD & WBMVD1) follows directly from Lemma 6.4. By definition, the WBMVDs $XZ_1 \approx\!\!>\approx\!\!> Y$, $XZ_2 \approx\!\!>\approx\!\!> Y$, and the BMVD $XY \Rightarrow\Rightarrow Z_1$, imply

$$\theta(XZ_1Y) \circ \theta(XZ_1Z_2) = \theta(XZ_1Z_2) \circ \theta(XZ_12Y), \tag{8.68}$$

$$\theta(XZ_2Y) \circ \theta(XZ_1Z_2) = \theta(XZ_1Z_2) \circ \theta(XZ_2Y), \tag{8.69}$$

and

$$\theta(XY) = \theta(XYZ_1) \circ \theta(XYZ_2) = \theta(XYZ_2) \circ \theta(XYZ_1), \tag{8.70}$$

respectively. Using equations (8.68)-(8.70) it follows

$$\begin{aligned}
\theta(XY) \circ \theta(XZ_1Z_2) &= \theta(XYZ_1) \circ \theta(XYZ_2) \circ \theta(XZ_1Z_2) \\
&= \theta(XYZ_1) \circ \theta(XZ_1Z_2) \circ \theta(XYZ_2) \\
&= \theta(XZ_1Z_2) \circ \theta(XYZ_1) \circ \theta(XYZ_2) \\
&= \theta(XZ_1Z_2) \circ \theta(XY). \qquad (8.71)
\end{aligned}$$

Equation (8.71) indicates that condition (i) is satisfied. Condition (ii) is also satisfied by Lemma 6.4. By definition, $X \approx\!> \approx\!> Y$. □

Lemma 8.4 indicates that axioms (BMVD & WBMVD1) and (BMVD & WB-MVD2) are sound.

Theorem 8.19 The axioms BMVD1-BMVD7, WBMVD1-WBMVD4, (BMVD & WBMVD1) and (BMVD & WBMVD2) form a complete axiomatization for the class of BMVDs and WBMVDs.

The proof of Theorem 8.19 can be shown by following the proof given by Van Gucht and Fischer [83].

As illustrated in Figure 1.1, the important point is that Theorem 8.18 and Theorem 8.19 indicate that the implication problems coincide for these two *solvable* classes of dependencies.

8.7 Summary

In this chapter, we made several significant contributions to the study of the implication problem in the Bayesian network model. Most importantly, it was demonstrated that the implication problems in the relational database model and the

Bayesian network model coincide on five *solvable* classes of dependencies. The work discussed on the three main classes of dependencies found in the probabilistic literature is reported in [95]. The results on the two new solvable classes of dependencies are in preparation. These results suggest that the implication problems coincide on *all* solvable classes of dependencies as illustrated in Figure 1.1. Studeny's remark [79] that the relational database model and the Bayesian network model are *different* was shown to be based on two *unsolvable* classes of dependencies.

Whereas axiomatization has been traditionally used to test the implication of probabilistic dependencies in the Bayesian network model, a *nonaxiomatic* method called the *chase* was introduced to test the implication of nonembedded probabilistic conditional independence.

It should be mentioned that Galles and Pearl [33] develop axioms for inferring contextual BMVDs.

Chapter 9

MULTI-AGENT BAYESIAN NETWORKS

There is increasing interest in developing *multi-agent* (distributed) Bayesian networks. In these situations, a group of agents work together to reach a common goal. In this chapter, we discuss two fundamental issues pertaining to multi-agent Bayesian networks, namely, constructing the multi-agent Bayesian network and reasoning in a multi-agent environment.

9.1 Constructing a Multi-Agent Bayesian Network

In this section, we propose an automated process for constructing the combined dependency structure of a *multi-agent* probabilistic network [92, 90]. Each domain expert supplies any known conditional independency information and not necessarily an explicit dependency structure. Our method determines a succinct representation of all the supplied independency information called a *minimal cover*. This process involves detecting all *inconsistent* information, and removing all *redundant* information. A *unique* dependency structure of the multi-agent probabilistic network can be constructed directly from this minimal cover. The main result of this section is that the constructed dependency structure is a *perfect-map* of the minimal cover. That is, every probabilistic conditional independency logically implied by the minimal cover can be inferred from the dependency structure, and every probabilistic conditional independency inferred from the dependency structure is logically implied by the minimal cover.

Traditionally, probabilistic knowledge is represented and reasoned with by a *single* agent. Even in the centralized environment, manually constructing a Bayesian network has been regarded as a difficult procedure [49], especially when the conditional independency information regarding the problem domain is not fully understood. Many learning methods have subsequently been developed for constructing the dependency structure of a probabilistic network using independency information mined from observed data including [12, 23, 40, 42, 44, 45, 50, 97, 99].

Recently, there is an emerging interest in extending the centralized probabilistic environment into a *multi-agent* environment. In these situations, a number of individual agents are willing to cooperate and share their knowledge to reach a common goal. (It is also possible that the agents are physically separated.) An example of this situation can be found in medical applications. Each agent could represent a medical specialist. These specialists pool their knowledge together to make a diagnosis. Another example can be found in military applications. Each agent could represent a unit commander in a battle. Each commander makes decisions with the information he possesses together with the information supplied by the other unit commanders.

In a multi-agent environment, we assume that the knowledge of each agent is represented by a marginal distribution of a *common* joint probability distribution. To define such a multi-agent probabilistic network, we need to explicitly specify the dependency structure representing the conditional independency information known to hold in the multi-agent problem domain. It is not realistic to expect the domain experts to manually construct the dependency structure since the problem domain may be much larger and perhaps distributed. One suggestion would be to learn the dependency structure from observed data. It is not entirely clear, however, how those learning methods developed for the single-agent environment can be applied, let alone how one might obtain a reliable sample. Thus, constructing the multi-agent dependency structure amounts to finding a method to combine the known conditional independency information supplied by the individual domain experts. One previously proposed method [102] constructs the dependency structure of a multi-agent

Bayesian network by verifying that the dependency structure formed by *connecting* the individual agent DAGs is *acyclic*. This method is straightforward, but we show in Section 9.1.2 that it may be too restrictive for constructing the multi-agent dependency structure in some situations.

In this chapter, we suggest a more robust algorithm for constructing the dependency structure of a multi-agent probabilistic network. Each domain expert supplies any known conditional independency information and not necessarily an explicit dependency structure. Our automated process computes a succinct representation of all the supplied independency information called a *minimal cover*. This process involves detecting all *inconsistent* information, and removing all *redundant* information. A *unique* dependency structure of the multi-agent probabilistic network can be constructed directly from this minimal cover. In fact, it is shown that the constructed dependency structure is a *perfect-map* [66] of the minimal cover. That is, every probabilistic conditional independency logically implied by the minimal cover can be inferred from the dependency structure, and every probabilistic conditional independency inferred from the dependency structure is logically implied by the minimal cover. Our method takes advantage of the fact that there exists a *complete* axiomatization for *nonembedded* conditional independencies.

9.1.1 Constructing the Qualitative Component

Our main goal here is to develop a process for the construction of the dependency structure (an acyclic hypergraph) of a multi-agent Markov network. We assume that the input to such a construction process is a set of probabilistic conditional independencies supplied by the different domain experts. We will first outline two problems of such a process caused by redundant and conflicting independency information in the initial input set. A complete set of axioms for nonembedded BMVDs will be subsequently applied to remove redundant and to detect inconsistent independency information. The remaining set of BMVDs is used to systematically construct the dependency structure of the desired Markov network. (Note that the resulting acyclic hypergraph is in fact a *perfect-map*.)

Given a set G of probabilistic conditional independencies supplied by the individual domain experts over a set R of attributes, a *construction algorithm* factorizes R into two sets of attributes on the basis of a known conditional independency in G. That is, given $XYZ = R$, the set of attributes R is replaced by $R \cap XY$ and $R \cap XZ$, where the BMVD $X \Rightarrow\Rightarrow Y \mid Z$ is in G. Each of these new subsets may be further factorized on the basis of another known BMVD in G. Before formally defining the construction algorithm, let us first highlight two desirable properties such a construction process should satisfy:

(i) Every conditional independency provided should contribute in the construction process, and

(ii) A unique dependency structure is constructed.

Unfortunately, without refining the conditional independencies supplied by the individual domain experts, it is not always possible to meet these desirable properties. To illustrate a failure of (i), consider the following example.

Example 9.1 Let G be the following set of BMVDs on $R = A_1A_2A_3A_4$:

$$G = \{A_1A_2 \Rightarrow\Rightarrow A_3,\ A_3A_4 \Rightarrow\Rightarrow A_1\}.$$

Factorizing R with the BMVD $A_1A_2 \Rightarrow\Rightarrow A_3$ produces the hypergraph $\mathcal{R} = \{A_1A_2A_3, A_1A_2A_4\}$. The problem now is that the other BMVD $A_3A_4 \Rightarrow\Rightarrow A_1$ in the input set G cannot be applied to refine the dependency structure $\mathcal{R}$. A similar argument holds if the BMVD $A_3A_4 \Rightarrow\Rightarrow A_1$ is applied first. To illustrate a failure of property (ii), consider the set G of BMVDs on $R = A_1A_2 \ldots A_7$:

$$\begin{aligned} G \;=\; & \{A_1 \Rightarrow\Rightarrow A_7,\ A_1A_2 \Rightarrow\Rightarrow A_4, A_1A_2 \Rightarrow\Rightarrow A_5,\ A_1A_2 \Rightarrow\Rightarrow A_3A_6A_7, \\ & A_1A_3 \Rightarrow\Rightarrow A_4,\ A_1A_3 \Rightarrow\Rightarrow A_6, A_1A_3 \Rightarrow\Rightarrow A_2A_5A_7\}. \end{aligned}$$

Factorizing R with the BMVD $A_1 \Rightarrow\Rightarrow A_7$ produces the hypergraph $\mathcal{R}_1$:

$$\mathcal{R}_1 = \{R_{11} = A_1A_7, R_{12} = A_1A_2A_3A_4A_5A_6\}.$$

The set of attributes $R_{12} = A_1A_2A_3A_4A_5A_6$ can be further refined with the BMVD $A_1A_2 \Rightarrow\Rightarrow A_4$. (Since the BMVD $A_1A_2 \Rightarrow\Rightarrow A_4$ holds on R, then the BMVD $A_1A_2 \Rightarrow\Rightarrow A_4$ holds on $A_1A_2A_3A_4A_5A_6$.) Factorizing $R_{12} = A_1A_2A_3A_4A_5A_6$ as such produces the sets $A_1A_2A_4$ and $A_1A_2A_3A_5A_6$. The latter can also be factorized using the BMVD $A_1A_2 \Rightarrow\Rightarrow A_5$ to produce the new dependency structure $\mathcal{R}_2$:

$$\mathcal{R}_2 = \{R_{11} = A_1A_7, R_{21} = A_1A_2A_4, R_{22} = A_1A_2A_5, R_{23} = A_1A_2A_3A_6\}.$$

The set of attributes $R_{23} = A_1A_2A_3A_6$ can be factorized with the BMVD $A_1A_3 \Rightarrow\Rightarrow A_6$ constructing the output dependency structure $\mathcal{R}_3$:

$$\mathcal{R}_3 = \{R_{11} = A_1A_7, R_{21} = A_1A_2A_4, R_{22} = A_1A_2A_5, R_{31} = A_1A_2A_3, R_{32} = A_1A_3A_6\}.$$

On the other hand, factorizing $R_{12} \in \mathcal{R}_1$ with the BMVD $A_1A_3 \Rightarrow\Rightarrow A_4$ produces the attribute sets $A_1A_3A_4$ and $A_1A_2A_3A_5A_6$. The latter of which can be be further factorized with the BMVD $A_1A_3 \Rightarrow\Rightarrow A_6$ to produce the dependency structure $\mathcal{R}_{2'}$:

$$\mathcal{R}_{2'} = \{R_{11} = A_1A_7, R_{21'} = A_1A_3A_4, R_{22'} = A_1A_3A_6, R_{23'} = A_1A_2A_3A_5\}.$$

Factorizing the set of attributes $R_{23'} = A_1A_2A_3A_5$ using the BMVD $A_1A_2 \Rightarrow\Rightarrow A_5$ produces the output dependency structure $\mathcal{R}_{3'}$:

$$\begin{aligned}\mathcal{R}_{3'} &= \{R_{11} = A_1A_7, R_{21'} = A_1A_3A_4, R_{22'} = A_1A_3A_6, R_{31'} = A_1A_2A_3,\\ &\quad R_{32'} = A_1A_2A_5\}.\end{aligned}$$

Obviously, the output constructed dependency structures $\mathcal{R}_3$ and $\mathcal{R}_{3'}$ are not the same.

The problem in example 9.1 is that the order in which the BMVDs were applied to factorize R affected the output dependency structure of the multi-agent Markov network. Another undesirable characteristic can be illustrated by the following example.

Example 9.2 Let $R = A_1A_2A_3A_4A_5A_6$ and G be the set of BMVDs

$$G = \{A_1 \Rightarrow\Rightarrow A_5,\ A_2 \Rightarrow\Rightarrow A_6,\ A_5A_6 \Rightarrow\Rightarrow A_3, A_5A_6 \Rightarrow\Rightarrow A_4\}.$$

Factorizing R first with the BMVD $A_1 \Rightarrow\Rightarrow A_5$, followed by the BMVD $A_2 \Rightarrow\Rightarrow A_6$ produces the output dependency structure $\mathcal{R}$:

$$\mathcal{R} = \{R_{11} = A_1A_5, R_{21} = A_1A_2A_3A_4, R_{22} = A_2A_6\}.$$

The remaining BMVDs in the input set G $A_5A_6 \Rightarrow\Rightarrow A_3$ and $A_5A_6 \Rightarrow\Rightarrow A_4$ cannot be applied to factorize any set of attributes in $\mathcal{R}$. However, $\mathcal{R}$ can be further factorized by the BMVD $A_1A_2 \Rightarrow\Rightarrow A_3$ as

$$\mathcal{R} = \{R_{11} = A_1A_5, R_{22} = A_2A_6, R_3 = A_1A_2A_3, R_4 = A_1A_2A_4\},$$

since G logically implies the BMVD $A_1A_2 \Rightarrow\Rightarrow A_3$.

Example 9.2 demonstrates that it may be possible to factorize the output dependency structure of the multi-agent Markov network with a BMVD $X \Rightarrow\Rightarrow Y$ not explicitly stated, but logically implied by G.

The above problems in examples 9.1 and 9.2 are the result of redundant and conflicting (inconsistent) conditional independency information in the initial input set. These two problems can be resolved by removing all redundant independencies, and subsequently identifying any conflicting conditional independencies.

Similar to the relational database theory [8, 28], it is useful to introduce the notion of a dependency basis. A dependency basis is used to summarize a set of BMVDs that all have the same left-hand side. Given $X \subseteq R$, the *dependency basis* of X, written $Dep(X)$, is defined as follows:

$$Dep(X) = \{W_1, W_2, \ldots, W_m\},$$

where $X \cap W_i = \emptyset$ for $i = 1, 2, \ldots, m$ and $\{W_1, W_2, \ldots, W_m\}$ forms a partition of $R - X$. With this notation, it is understood that the set $\{A_i \mid A_i \in X\}$ is implicitly included in $Dep(X)$. The usefulness of introducing $Dep(X)$ lies in the fact that for any BMVD $X \Rightarrow\Rightarrow Y$ which is logically implied by an given set G of BMVDs, the set Y is a *union* of some elements in $Dep(X)$. That is, if $G \models X \Rightarrow\Rightarrow Y$, then Y is equal to the union of some sets in $Dep(X)$; whereas, for each nonempty proper subset W of W_i $(1 \leq i \leq m)$, the BMVD $X \Rightarrow\Rightarrow W$ is not in G^+ (i.e., $G \not\models X \Rightarrow\Rightarrow W$).

Sometimes, it is more convenient to express the dependency basis $Dep(X)$ as in [28], namely:

$$X \Rightarrow\Rightarrow W_1 \mid W_2 \mid \ \ldots \ \mid W_m.$$

These two notations will be used interchangeably in the following exposition.

We will first present an algorithm to construct the dependency basis for a given $X \subseteq R$. Beeri [7] originally proposed a polynomial time algorithm to compute the dependency basis for MVDs in relational databases. (Faster algorithms have since been proposed [32, 74].) Here we adopt Beeri's procedure to our problem by using BMVDs instead of MVDs. Our method is outlined in Algorithm 9.1 by replacing the complete axiomatization for multivalued dependencies with the complete axiomatization for BMVDs. Given a set G of BMVDs on a set of attributes R and a set $X \subseteq R$, Algorithm 9.1 constructs the dependency basis of X as follows:

Algorithm 9.1

procedure Dependency-Basis(G, X)

$Dep(X) = \{\{A\} \ \mid \ A \in \{X\}\} \ \cup \ \{R - X\}$

repeat until $Dep(X)$ is not changed

{

for each BMVD $W \Rightarrow\Rightarrow Z$ in G

{

$Y = \cup \ \{R \ \mid \ R \in Dep(X),$ and $R \cap W \neq \emptyset\}$

$Z' = Z - Y$

if $Z' \neq \emptyset$ and $Z' \neq \cup R_i$ for some $R_i \in Dep(X)$

{

$Dep(X) = \{R \mid R \in Dep(X)$ and $R \cap Z' = \emptyset\} \cup$

$\{R \cap Z', R - Z', Z' - R \mid R \in Dep(X)$ and $R \cap Z' \neq \emptyset\}$

}

}

}

return $(Dep(X))$

end Dependency-Basis

Let us use an example to demonstrate how Algorithm 9.1 works.

Example 9.3 Let $R = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}\}$. Let the input set of nonembedded BMVDs be $G = \{\{A_1, A_2\} \Rightarrow\Rightarrow \{A_4, A_5, A_6, A_7\}, \{A_3, A_7, A_{10}\} \Rightarrow\Rightarrow \{A_1, A_4, A_8, A_9\}\}$ on R. Suppose we want to compute the dependency basis $Dep(X)$ for the subset $X = \{A_1, A_3, A_7, A_{10}\} \subseteq R$. By (BMVD1), it follows:

$$\begin{aligned} \{A_1, A_3, A_7, A_{10}\} &\Rightarrow\Rightarrow \{A_1\}, \\ \{A_1, A_3, A_7, A_{10}\} &\Rightarrow\Rightarrow \{A_3\}, \\ &\vdots \\ \{A_1, A_3, A_7, A_{10}\} &\Rightarrow\Rightarrow \{A_1, A_3, A_7, A_{10}\}. \end{aligned}$$

By applying (BMVD4) repeatedly, we obtain the *initial* dependency basis:

$$Dep(X) = \{\{A_1\}, \{A_3\}, \{A_7\}, \{A_{10}\}, \{A_2, A_4, A_5, A_6, A_8, A_9\}\}.$$

First, we check if the BMVD $\{A_1, A_2\} = W \Rightarrow\Rightarrow Z = \{A_4, A_5, A_6, A_7\}$ in G can be used to refine $Dep(X)$. The members $\{A_1\}$ and $\{A_2, A_4, A_5, A_6, A_8, A_9\}$ in $Dep(X)$ intersect W. Thus,

$$Y = \{A_1\} \cup \{A_2, A_4, A_5, A_6, A_8, A_9\} = \{A_1, A_2, A_4, A_5, A_6, A_8, A_9\}.$$

By (BMVD3), we can immediately conclude:

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Y = \{A_1, A_2, A_4, A_5, A_6, A_8, A_9\}. \tag{9.72}$$

On the other hand, by (BMVD5), $W \Rightarrow\Rightarrow Z$ implies that:

$$W \cup \{A_4, A_5, A_6, A_8, A_9\} \Rightarrow\Rightarrow Z \cup \{A_4, A_5, A_6, A_8, A_9\}.$$

That is,

$$\{A_1, A_2, A_4, A_5, A_6, A_8, A_9\} \Rightarrow\Rightarrow \{A_4, A_5, A_6, A_7, A_8, A_9\}. \tag{9.73}$$

By applying (BMVD2) to equations (9.72) and (9.73), we obtain:

$$\begin{aligned}\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Z' &= Z - Y \\ &= \{A_4, A_5, A_6, A_7, A_8, A_9\} - \{A_1, A_2, A_4, A_5, A_6, A_8, A_9\} \\ &= \{A_7\}.\end{aligned}$$

However, the current dependency basis $Dep(X)$ cannot be refined since $Z' = \{A_7\}$ is already an element in $Dep(X)$.

Next we consider the BMVD $\{A_3, A_7, A_{10}\} = W \Rightarrow\Rightarrow Z = \{A_1, A_4, A_8, A_9\}$ in G. Then, $Y = \{A_3\} \cup \{A_7\} \cup \{A_{10}\} = \{A_3, A_7, A_{10}\}$, as only $\{A_3\}, \{A_7\}$, and $\{A_{10}\}$ intersect W. Thus, by (BMVD3), we have:

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Y = \{A_3, A_7, A_{10}\}.$$

By applying (BMVD2), this BMVD together with $W \Rightarrow\Rightarrow Z$ implies that:

$$\begin{aligned}\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Z' &= Z - Y \\ &= \{A_1, A_4, A_8, A_9\} - \{A_3, A_7, A_{10}\} = \{A_1, A_4, A_8, A_9\}.\end{aligned}$$

Now (BMVD4) is used to *refine* the above $Dep(X)$. From:

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Y = \{A_2, A_4, A_5, A_6, A_8, A_9\},$$

and

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Z = \{A_1, A_4, A_8, A_9\},$$

it follows:

$$\begin{aligned}\{A_1, A_3, A_7, A_{10}\} = X &\Rightarrow\Rightarrow Y \cap Z = \{A_4, A_8, A_9\}, \\ \{A_1, A_3, A_7, A_{10}\} = X &\Rightarrow\Rightarrow Y - Z = \{A_2, A_5, A_6\}, \\ \{A_1, A_3, A_7, A_{10}\} = X &\Rightarrow\Rightarrow Z - Y = \{A_1\}.\end{aligned}$$

Thus, the new $Dep(X)$ becomes:

$$\begin{aligned}Dep(X) &= \{\{A_3\}, \{A_7\}, \{A_{10}\}\} \cup \{\{A_1\}, \{A_2, A_5, A_6\}, \{A_4, A_8, A_9\}\} \\ &= \{\{A_1\}, \{A_3\}, \{A_7\}, \{A_{10}\}, \{A_2, A_5, A_6\}, \{A_4, A_8, A_9\}\}.\end{aligned}$$

As the dependency basis has been updated, the above computation is repeated using the input set G. The BMVD $\{A_1, A_2\} = W \Rightarrow\Rightarrow Z = \{A_4, A_5, A_6, A_7\}$ and $Dep(X)$ imply that:

$$Y = \{A_1\} \cup \{A_2, A_5, A_6\} = \{A_1, A_2, A_5, A_6\}.$$

By (BMVD2) and (BMVD5),

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Z' = Z - Y = \{A_4, A_7\}.$$

Applying (BMVD4) to the BMVDs,

$$\begin{aligned} \{A_1, A_3, A_7, A_{10}\} &\Rightarrow\Rightarrow \{A_7\}, \\ \{A_1, A_3, A_7, A_{10}\} &\Rightarrow\Rightarrow \{A_4, A_7\}, \end{aligned}$$

and

$$\{A_1, A_3, A_7, A_{10}\} \Rightarrow\Rightarrow \{A_4, A_8, A_9\},$$

we obtain:

$$\begin{aligned} \{A_1, A_3, A_7, A_{10}\} &\Rightarrow\Rightarrow \{A_4\}, \\ \{A_1, A_3, A_7, A_{10}\} &\Rightarrow\Rightarrow \{A_7\}, \end{aligned}$$

and

$$\{A_1, A_3, A_7, A_{10}\} \Rightarrow\Rightarrow \{A_8, A_9\}.$$

Hence, $Dep(X)$ is changed to:

$$Dep(X) = \{\{A_1\}, \{A_3\}, \{A_7\}, \{A_{10}\}, \{A_4\}, \{A_8, A_9\}, \{A_2, A_5, A_6\}\}.$$

One can verify that the above $Dep(X)$ cannot be further refined. $Dep(X)$ is therefore the desired dependency basis for $X = \{A_1, A_3, A_7, A_{10}\}$ which can be written as:

$$X \Rightarrow\Rightarrow \{A_4\} \mid \{A_8, A_9\} \mid \{A_2, A_5, A_6\},$$

or

$$Dep(X) = \{\{A_4\}, \{A_8, A_9\}, \{A_2, A_5, A_6\}\}.$$

Theorem 9.1 Given a set G of BMVDs on a set of attributes R and $X \subseteq R$, Algorithm 9.1 always terminates and $Dep(X)$ is the dependency basis of X.

The proof of Theorem 9.1 follows the corresponding proof [7] in relational database theory and is shown in Appendix C.

The individual domain experts may initially supply redundant and conflicting conditional independency information. Our proposed algorithm for constructing the dependency structure of probabilistic networks requires the input independency information in a more refined format. In particular all redundant independency information must be removed and the remaining information expressed in terms of its dependency basis. The last task is to identify any conflicting independency information.

The individual domain experts may initially supply a set G of BMVDs containing *redundant* ones. That is, those BMVDs that can be derived from the rest of the BMVDs in G using the inference axioms (BMVD1) and (BMVD2). We say that a set G_1 of BMVDs is a *cover* of G if $G^+ = G_1^+$. If a cover G_1 of G contains no proper subset G_2 such that G_2 is also a cover of G, i.e., $G^+ = G_2^+$, then G_1 is a *minimal* cover of G. A minimal cover contains no redundant independency information.

We now give a procedure to compute a minimal cover for a given set G of BMVDs supplied by the individual agents. Take any BMVD g in G, say $X \Rightarrow\Rightarrow Y$, and compute $Dep(X)$ from the set $G - \{g\}$ of BMVDs. If Y is a union of some sets in $Dep(X)$, i.e., $G - \{g\} \vdash X \Rightarrow\Rightarrow Y$, then remove g from G; otherwise g remains in G. Repeat this step for every BMVD in G. If we adopt Sagiv's faster algorithm [74] for computing the dependency basis using multivalued dependencies instead of Beeri's [7], a minimum cover of a given set G of BMVDs can be computed in time $O(k^2\ |R^2|)$, where k is the number of BMVDs originally in G [54].

Consider a computed minimum cover G of the collective set of BMVDs supplied by the agents. The left sides of the BMVDs of G are called the *keys* of G. The *left set* of G, written $\mathbf{X}$, is the set of all keys of G. A *full* minimum cover G_1 is a minimum cover which contains all BMVDs in the dependency basis of $\mathbf{X}$. That is, if the dependency basis of a key $X \in \mathbf{X}$ is $Dep(X) = \{W_1, W_2, \ldots, W_m\}$, then G_1 contains the BMVDs $X \Rightarrow\Rightarrow W_i$ $(1 \leq i \leq m)$. A full minimum cover contains no redundant information and is expressed in terms of its dependency basis. For example, given the minimum cover G of a set of BMVDs over $R = \{A_1, A_2, A_3, A_4\}$

$$G \;=\; \{\{A_1\} \Rightarrow\Rightarrow \{A_2\} \mid \{A_3, A_4\},\; \{A_1, A_2\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_4\}\}, \tag{9.74}$$

the full minimum cover G_1 of G is computed by repeatedly applying Algorithm 9.1 with G and each key in $\mathbf{X}$ as input:

$$\begin{aligned} G_1 &= \{\; Dep(\{A_1\}) = \{\; \{A_2\}, \{A_3\}, \{A_4\} \;\},\; Dep(\{A_1, A_2\}) = \{\; \{A_3\}, \{A_4\} \;\} \;\} \\ &= \{\{A_1\} \Rightarrow\Rightarrow \{A_2\} \mid \{A_3\} \mid \{A_4\},\; \{A_1, A_2\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_4\} \;\}. \end{aligned} \tag{9.75}$$

It should be noted that such a full minimum cover may contain conflicting BMVDs. A *conflict-free* full minimum cover is derived by removing the conflicting BMVDs from the full minimum cover. This conflict-free full minimum cover is used to construct the multi-agent dependency structure.

The notion of conflict-free multivalued dependencies was originally introduced by Lien [55] in the study of the relationship between various database models. We extend this notion to BMVDs in our probabilistic relational data model. We say that a BMVD $X \Rightarrow\Rightarrow Y$ *splits* two attributes A_i and A_j if one of them in in Y and the other is in $R - XY$, where R is the set of all attributes. A set G of BMVDs splits two attributes A_i and A_j if some BMVD in G splits them. We then say that a BMVD splits a set W if it splits two attributes in W, and that a set G of BMVDs splits a set W if some BMVD in G splits two attributes in W. A set G of BMVDs is *conflict-free* if

$$\begin{aligned} &(i) \quad G \text{ does not split its keys, and} \\ &(ii) \quad Dep(X) \cap Dep(Y) \subseteq Dep(X \cap Y). \end{aligned}$$

If conflicting BMVDs are detected, we have to rely on the domain experts to resolve these conflicts. Henceforth, we may assume that a conflict-free full minimum cover has been determined from the BMVDs supplied by the individual domain experts. Here we suggest an algorithm (i.e., Algorithm 9.2) to generate a unique dependency structure (an acyclic hypergraph) from a conflict-free full minimum cover of BMVDs. This algorithm is a modified version of Lien's algorithm [55].

Let $\mathbf{X}$ denote the set of keys in the conflict-free full minimum cover. The keys in $\mathbf{X}$ can be arranged in a *p-ordering sequence* $(X_1, X_2, \ldots, X_p)$ such that $X_i \subset X_j$ implies $i < j$. Given a conflict-free full minimum cover G and a p-ordering sequence $(X_1, X_2, \ldots, X_p)$ of the keys $\mathbf{X}$ of G, Algorithm 9.2 constructs an acyclic hypergraph representing the dependency structure of the input BMVDs as follows:

Algorithm 9.2

procedure $Construction(G, (X_1, X_2, \ldots, X_p))$
$\mathcal{R}^0 := \{R\};$
for $i := 1$ to p
{
    $\mathcal{R}^{i-1} := \mathcal{R}^{i-1} - \{R_j\}$, where $X_i \subseteq R_j;$
    $\mathcal{R}^i := \mathcal{R}^{i-1} \cup \{X_i(R_j \cap W) \mid W \in Dep(X_i)$ and $R_j \cap W \neq \emptyset\};$
}
return $(\mathcal{R}^p)$
end $Construction$

Example 9.4 Let $R = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\}$ and let $(\{A_1\}, \{A_2\}, \{A_3\}, \{A_1, A_2\}, \{A_1, A_3\}, \{A_2, A_3\})$ be a p-ordering sequence of the keys in the following conflict-free minimum cover G:

$$
\begin{aligned}
G = \{\ \{A_1\} &\Rightarrow\Rightarrow \{A_7\} \mid \{A_2, A_3, A_4, A_5, A_6, A_8, A_9\},\\
\{A_2\} &\Rightarrow\Rightarrow \{A_8\} \mid \{A_1, A_3, A_4, A_5, A_6, A_7, A_9\},\\
\{A_3\} &\Rightarrow\Rightarrow \{A_9\} \mid \{A_1, A_2, A_4, A_5, A_6, A_7, A_8\},\\
\{A_1, A_2\} &\Rightarrow\Rightarrow \{A_4\} \mid \{A_7\} \mid \{A_8\} \mid \{A_3, A_5, A_6, A_9\},\\
\{A_1, A_3\} &\Rightarrow\Rightarrow \{A_5\} \mid \{A_7\} \mid \{A_9\} \mid \{A_2, A_4, A_6, A_8\},\\
\{A_2, A_3\} &\Rightarrow\Rightarrow \{A_6\} \mid \{A_8\} \mid \{A_9\} \mid \{A_1, A_4, A_5, A_7\}\ \}.
\end{aligned}
$$

The dependency structure is $\mathcal{R}^0 = \{\ \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\}\ \}$ in the initialization step. For $i = 1$, the hyperedge $R_j \in \mathcal{R}^0$ is $R_j = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7,$

$A_8, A_9\}$ since $X_1 = \{A_1\} \subseteq \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\}$. The first step removes the hyperedge R_j from $\mathcal{R}^0$ to obtain $\mathcal{R}^0 = \emptyset$. The second step adds hyperedges to construct $\mathcal{R}^1$. Since $W = \{A_7\} \in Dep(\{A_1\})$ and $R_j \cap \{A_7\} \neq \emptyset$, the hyperedge

$$X_1(R_j \cap \{A_7\}) = \{A_1, A_7\},$$

is added to $\mathcal{R}^1$. Similarly, $W = \{A_2, \ldots, A_9\} \in Dep(\{A_1\})$ and $R_j \cap \{A_2, \ldots, A_9\} \neq \emptyset$. Thus, $X_1(R_j \cap W) = \{A_1, \ldots, A_6, A_8, A_9\}$ is added to $\mathcal{R}^1$. Thus, the intermediate dependency structure $\mathcal{R}^1$ is:

$$\mathcal{R}^1 = \{\{A_1, A_7\}, \{A_1, A_2, A_3, A_4, A_5, A_6, A_8, A_9\}\}.$$

For $i = 2$, the hyperedge $R_j \in \mathcal{R}^1$ is $R_j = \{A_1, \ldots, A_6, A_8, A_9\}$ since $X_2 = \{A_2\} \subseteq R_j$. The first step removes R_j from $\mathcal{R}^1$ obtaining $\mathcal{R}^1 = \{ \{A_1, A_7\} \}$. The second step then adds the hyperedges $\{A_2, A_8\}$ and $\{A_1, \ldots, A_6, A_9\}$ to $\mathcal{R}^1$ to construct the intermediate dependency structure $\mathcal{R}^2$ as follows:

$$\mathcal{R}^2 = \{\{A_1, A_7\}, \{A_2, A_8\}, \{A_1, A_2, A_3, A_4, A_5, A_6, A_9\}\}.$$

The subsequent intermediate dependency structures generated by Algorithm 9.2 are:

$$\begin{aligned}
\mathcal{R}^3 &= \{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\}, \{A_1, A_2, A_3, A_4, A_5, A_6\}\}, \\
\mathcal{R}^4 &= \{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\}, \{A_1, A_2, A_4\}, \{A_1, A_2, A_3, A_5, A_6\}\}, \\
\mathcal{R}^5 &= \{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\}, \{A_1, A_2, A_4\}, \{A_1, A_3, A_5\}, \{A_1, A_2, A_3, A_6\}\}, \\
\mathcal{R}^6 &= \{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\}, \{A_1, A_2, A_4\}, \{A_1, A_3, A_5\}, \{A_2, A_3, A_6\}, \\
&\quad \{A_1, A_2, A_3\}\}.
\end{aligned}$$

It can be easily verified that the output $\mathcal{R} = \mathcal{R}^6$ is an acyclic hypergraph. A hypertree construction ordering of this schema is $R_1 = \{A_1, A_2, A_3\}$, $R_2 = \{A_1, A_2, A_4\}$, $R_3 = \{A_1, A_3, A_5\}$, $R_4 = \{A_2, A_3, A_6\}$, $R_5 = \{A_1, A_7\}$, $R_6 = \{A_2, A_8\}$, $R_7 = \{A_3, A_9\}$.

Thus, the J-keys of $\mathcal{R}$ are:

$$\begin{aligned}
R_2 \cap R_{b(2)} &= \{A_1, A_2\}, & b(2) &= 1,\\
R_3 \cap R_{b(3)} &= \{A_1, A_3\}, & b(3) &= 1,\\
R_4 \cap R_{b(4)} &= \{A_2, A_3\}, & b(4) &= 1,\\
R_5 \cap R_{b(5)} &= \{A_1\}, & b(5) &= 2,\\
R_6 \cap R_{b(6)} &= \{A_2\}, & b(6) &= 4,\\
R_7 \cap R_{b(7)} &= \{A_3\}, & b(7) &= 3.
\end{aligned}$$

Algorithm 9.2 is executed $\mathbf{X} = p$ times. Each iteration i has W_{i,m_i} steps, where $Dep(X_i)$ is

$$X_i \Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ \ldots \ \mid W_{i,m_i}.$$

Thus, the computational complexity of Algorithm 9.2 is $O(|G|)$.

It is worth mentioning that we are not simply using Lien's algorithm [55] with a different kind of independency as input. Lien's input type is called "MVDs with nulls" (NMVDs) with which some properties of BMVDs (probabilistic conditional independencies) do not hold. In particular, NMVDs have the following two characteristics:

(i) For any key X, $Dep(X)$ can be computed by those NMVDs with a key $Y \subseteq X$,

(ii) Two logically equivalent minimum covers have the same set of keys $\mathbf{X}$.

Characteristics (i) and (ii) do not hold for BMVDs. A counter-example of (i) is given by G in equation (9.74). The dependency basis of key $\{A_1\}$ is $Dep(\{A_1\}) = \{\{A_2\}, \{A_3\}, \{A_4\}\}$ as shown in equation (9.75). However, the BMVD $\{A_1\} \Rightarrow\Rightarrow \{A_3\}$ cannot be derived without using the key $\{A_1, A_2\} \not\subseteq \{A_1\}$. A counterexample of characteristic (ii) is given by consideration of the sets G_1 and G_2 of BMVDs over $R = \{A_1, A_2, A_3, A_4, A_5\}$, where

$$\begin{aligned}
G_1 &= \{\{A_5\} \Rightarrow\Rightarrow \{A_2\},\ \{A_1, A_5\} \Rightarrow\Rightarrow \{A_3\}\}\\
G_2 &= \{\{A_5\} \Rightarrow\Rightarrow \{A_2\},\ \{A_1, A_2, A_5\} \Rightarrow\Rightarrow \{A_3\}\}.
\end{aligned}$$

We have $G_1^+ = G_2^+$, and G_1 and G_2 are both minimal covers of G_1^+. However, the keys $\mathbf{X}_1$ of G_1 are not the same as the keys $\mathbf{X}_2$ of G_2, i.e., $\mathbf{X}_1 = \{\{A_5\}, \{A_1, A_5\}\} \neq \mathbf{X}_2 = \{\{A_5\}, \{A_1, A_2, A_5\}\}$.

Because of the above differences between "MVDs with nulls" and probabilistic conditional independencies, the construction algorithm (Algorithm 9.2) requires the input set of dependencies in different forms. For applications involving NMVDs, the relational database scheme is constructed from a conflict-free minimum cover. On the other hand, for applications involving conditional independencies, the dependency structure of the probabilistic network is constructed from the more refined conflict-free full minimum cover.

Our goal now is to demonstrate that the acyclic hypergraph $\mathcal{R}$ constructed as output by Algorithm 9.2 is a perfect-map of the given input set G of BMVDs. That is, every BMVD logically implied by G can be inferred from $\mathcal{R}$, and every BMVD inferred from $\mathcal{R}$ is logically implied by G. If $\mathcal{R}$ is a hypergraph, then the set of BMVDs (MVDs) *generated by* $\mathcal{R}$ is the set of BMVDs $X \Rightarrow\Rightarrow Y$, where Y is the union of some disconnected components of the hypergraph $\mathcal{R} - X$ obtained from $\mathcal{R}$ by deleting the set X of nodes. That is, $\mathcal{R} - X = \{h - X \mid h \text{ is a hyperedge of } \mathcal{R}\} - \{\emptyset\}$. We then say that X *separates off* Y from the rest of the nodes.

Example 9.5 Consider the acyclic hypergraph $\mathcal{R}$ in Figure 3.2. Let $X = \{A_2, A_3\}$. The disconnected components of $\mathcal{R}$ obtained by deleting the set X is the set

$$\mathcal{R} - X = \{\{A_1\}, \{A_4\}, \{A_5, A_6\}\}.$$

Three BMVDs generated by $\mathcal{R}$ are $\{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\}$, $\{A_2, A_3\} \Rightarrow\Rightarrow \{A_4\}$, and $\{A_2, A_3\} \Rightarrow\Rightarrow \{A_5, A_6\}$.

We first turn our attention to relationship between any two keys X_i and X_k in $\mathbf{X}$. Consider the dependency bases of X_i and X_k:

$$X_i \Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,m_i-1} \mid W_{i,m_i}, \tag{9.76}$$

$$X_k \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,l} \mid \ldots \mid W_{k,m_k-1} \mid W_{k,m_k}. \tag{9.77}$$

Case 1. $X_i - X_k \neq \emptyset$ and $X_k - X_i \neq \emptyset$. Since keys are not split, we know precisely one $X_i W_{i,j}$ contains X_k. Without loss of generality, we may choose $j = 1$, namely,

$$X_k \subseteq X_i W_{i,1}.$$

Applying augmentation on equation (9.77), we derive the BMVDs:

$$X_i W_{i,1} \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,m_k-1} \mid W_{k,m_k}.$$

By transitivity, we obtain:

$$\begin{aligned}
X_i &\Rightarrow\Rightarrow & W_{k,1} - X_i W_{i,1} = W_{k,1} - W_{i,1},\\
X_i &\Rightarrow\Rightarrow & W_{k,2} - X_i W_{i,1} = W_{k,2} - W_{i,1},\\
&\vdots\\
X_i &\Rightarrow\Rightarrow & W_{k,m_k-1} - X_i W_{i,1} = W_{k,m_k-1} - W_{i,1},\\
X_i &\Rightarrow\Rightarrow & W_{k,m_k} - X_i W_{i,1} = W_{k,m_k} - W_{i,1}.
\end{aligned} \tag{9.78}$$

Similarly, we also know precisely one $X_k W_{k,j}$ contains X_i. Without loss of generality, we may assume $j = l$, namely,

$$X_i \subseteq X_k W_{k,l}.$$

Note that $W_{k,l}$ must contain at least one attribute belonging to X_i; otherwise, $X_i \subseteq X_k$ contradicting our initial assumption.

Applying augmentation on equation (9.76), we derive the BMVDs

$$X_k W_{k,l} \Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,m_i-1} \mid W_{i,m_i}.$$

Applying transitivity using these BMVDs and equation (9.77) we obtain:

$$\begin{aligned}
X_k &\Rightarrow\Rightarrow & W_{i,1} - X_k W_{k,l} = W_{i,1} - W_{k,l},\\
X_k &\Rightarrow\Rightarrow & W_{i,2} - X_k W_{k,l} = W_{i,2} - W_{k,l},\\
&\vdots\\
X_k &\Rightarrow\Rightarrow & W_{i,m_i} - X_k W_{k,l} = W_{i,m_i} - W_{k,l}.
\end{aligned} \tag{9.79}$$

We now make some observations which will be used in showing the main result of this section. (To improve readability the proofs have been moved to Appendix C.)

Proposition 9.1 No $W_{k,j}$ in $Dep(X_k)$, $j \neq l$, partially intersects $W_{i,1}$.

Proposition 9.2 If $W_{k,j} \cap W_{i,1} = \emptyset$, $j \neq l$, then $W_{k,j}$ must also belong to $Dep(X_i \cap X_k)$.

For notational convenience, let

$$Dep'(X_i) \;=\; Dep(X_i) - \{W_{i,1}\}.$$

Proposition 9.3 If $W_{i,s}$ in $Dep'(X_i)$ belongs to $Dep(X_i \cap X_k)$, then $W_{i,s}$ must also belong to $Dep(X_k)$.

The only remaining element in $Dep(X_k)$ which we have not considered in detail so far is $W_{k,l}$. The following observations are in order.

Proposition 9.4 $W_{k,l} - X_i W_{i,1} \neq \emptyset$. More specifically,

$$W_{k,l} - X_i W_{i,1} \;=\; \cup \; \{ \; W \mid W \in Dep'(X_i) \text{ and } W \notin Dep(X_i \cap X_k) \; \}.$$

Proposition 9.5 $X_i - X_k \neq \emptyset$ and $X_k - X_i \neq \emptyset$. If $Dep(X_i)$ and $Dep(X_k)$ satisfy propositions (1) - (4), then $Dep(X_i)$ and $Dep(X_k)$ cannot be refined.

Case 2. $X_i \subseteq X_k$. We have the following observations on the elements in $Dep(X_k)$.

Proposition 9.6 No $W_{k,j}$ partially intersects $W_{i,1}$.

Proposition 9.7 Every $W_{i,s}$ in $Dep'(X_i)$ also belongs to $Dep(X_k)$.

Proposition 9.8 If $W_{k,j} \cap W_{i,1} = \emptyset$, then $W_{k,j}$ is also an element in $Dep'(X_i)$.

Let us now explicitly demonstrate the implications of the above results. Let X_i and X_k be two keys that are not subsets of each other. Recall the dependency basis of X_i and X_k in equations (9.76) and equations (9.77), respectively, where $X_k \subseteq X_i W_{i,1}$

and $X_i \subseteq X_k W_{k,l}$. Since proposition 9.1 states that no $W_{k,j}$ in $Dep(X_k)$, $j \neq l$, partially intersects $W_{i,1}$, we can rewrite $Dep(X_k)$ in equation (9.77) as:

$$X_k \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,l-1} \mid W_{k,l} \mid W_1 \mid \ldots \mid W_s, \tag{9.80}$$

where $W_{k,j} \cap W_{i,1} = W_{k,j}$, $j = 1, \ldots, l-1$, and $W_j \cap W_{i,1} = \emptyset$, $j = 1, \ldots, s$. By proposition 9.2, each element $W_1, \ldots, W_s$ also belongs to $Dep(X_i \cap X_k)$, i.e., $\{W_1, \ldots, W_s\} \subseteq Dep(X_i \cap X_k)$. By proposition 9.4, we can write $Dep(X_i)$ in equation (9.76) and $Dep(X_k)$ in equation (9.80) as:

$$X_i \Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,r} \mid W_{i,r+1} \mid \ldots \mid W_{i,m_i}, \tag{9.81}$$

and

$$X_k \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,l-1} \mid ((Z)(W_{i,2} \cdots W_{i,r})) \mid W_1 \mid \ldots \mid W_s, \tag{9.82}$$

where $W_{i,j} \notin Dep(X_i \cap X_k)$, $j = 2, \ldots, r$, and $W_{k,l} = ((Z)(W_{i,2} \cdots W_{i,r}))$. Since every element in $Dep'(X_i)$ is either an element in $Dep(X_i \cap X_k)$ or not an element in $Dep(X_i \cap X_k)$, by proposition 9.3 the elements $W_{i,r+1}, \ldots, W_{i,m_i}$ also belong to $Dep(X_k)$. This means the elements $W_{i,r+1}, \ldots, W_{i,m_i}$ are precisely the elements $W_1, \ldots, W_s$ in $Dep(X_k)$. Thus, we rewrite $Dep(X_i)$ in equation (9.81) as:

$$X_i \Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,r} \mid W_1 \mid \ldots \mid W_s. \tag{9.83}$$

Since $W_{k,j} \cap W_{i,1} = W_{k,j}$, $j = 1, \ldots, l-1$, we further clarify $Dep(X_i)$ in equation (9.83) as:

$$X_i \Rightarrow\Rightarrow ((Y)(W_{k,1} \cdots W_{k,l-1})) \mid W_{i,2} \mid \ldots \mid W_{i,r} \mid W_1 \mid \ldots \mid W_s, \tag{9.84}$$

where $W_{i,1} = ((Y)(W_{k,1} \cdots W_{k,l-1}))$. By substituting for Y in the above equation and Z in equation (9.82), we obtain the most detailed description of $Dep(X_i)$ and $Dep(X_k)$:

$$X_i \Rightarrow\Rightarrow (((X_k - X_i)(V))(W_{k,1} \cdots W_{k,l-1})) \mid W_{i,2} \mid \ldots \mid W_{i,r} \mid W_1 \mid \ldots \mid W_s, \tag{9.85}$$

and

$$X_k \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,l-1} \mid (((X_i - X_k)(V))(W_{i,2} \cdots W_{i,r})) \mid W_1 \mid \ldots \mid W_s, \tag{9.86}$$

where $\{W_1, \ldots, W_s\} \subseteq Dep(X_i \cap X_k)$, $Y = ((X_k - X_i)(V))$, $Z = ((X_i - X_k)(V))$, and V may be the empty set.

Similarly, in the second case where $X_i \subseteq X_k$, by propositions 9.6 $-$ 9.8, $Dep(X_i)$ in equation (9.76) and $Dep(X_k)$ in equation (9.77) take the detailed form:

$$X_i \Rightarrow\Rightarrow (X_k - X_i)(W_{k,1} \cdots W_{k,l-1}) \mid W_1 \mid \ldots \mid W_s, \tag{9.87}$$

and

$$X_k \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,l-1} \mid W_1 \mid \ldots \mid W_s. \tag{9.88}$$

This completes our analysis of the relationship between any two keys X_i and X_k in $\mathbf{X}$.

We now focus our attention on the graphical properties of the dependency structure constructed by Algorithm 9.2.

Recall that $\mathcal{R}^{k-1}$ denotes the intermediate dependency structure constructed after $k-1$ iterations of Algorithm 9.2. We want to show that X_k is a subset of a hyperedge h in $\mathcal{R}^{k-1}$, where

$$h \;=\; X_1 W_{1,1} \cap X_2 W_{2,1} \cap \ldots \cap X_{k-1} W_{k-1,m_k-1},$$

and we have assumed that $X_k \subseteq X_j W_{j,1}$, $j = 1, \ldots, k-1$.

We first make some observations in the case where the dependency structure $\mathcal{R}^{k-1}$ is a perfect-map of the following BMVDs:

$$\begin{aligned}
X_1 \;&\Rightarrow\Rightarrow\; W_{1,1} \mid W_{1,2} \mid \ldots \mid W_{1,m_1}, \\
&\vdots \\
X_i \;&\Rightarrow\Rightarrow\; W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,m_i}, \\
&\vdots \\
X_{k-1} \;&\Rightarrow\Rightarrow\; W_{k-1,1} \mid W_{k-1,2} \mid \ldots \mid W_{k-1,m_{k-1}}.
\end{aligned} \tag{9.89}$$

From the assumption that $\mathcal{R}^{k-1}$ is a perfect-map of equation (9.89), X_i is a J-key of $\mathcal{R}^{k-1}$ and $X_i W_{i,1}$ is a disconnected component when X_i is deleted from $\mathcal{R}^{k-1}$. (The same can be said about all the X_i's in equation (9.89).) Clearly each $X_i W_{i,1}$ can be characterized by a unique set $\overline{X_i W_{i,1}}$ of hyperedges in $\mathcal{R}^{k-1}$, namely:

$$\overline{X_i W_{i,1}} = \{h \mid h \in \mathcal{R}^{k-1} \text{ and } h \cap W_{i,1} \neq \emptyset\}.$$

Proposition 9.9 X_k is a subset of exactly one hyperedge h in $\mathcal{R}^{k-1}$.

Proposition 9.10 Let h be the hyperedge in $\mathcal{R}^{k-1}$ containing X_k. Then

$$\{h\} \;=\; \overline{X_1 W_{1,1}} \cap \ldots \cap \overline{X_{k-1} W_{k-1,1}}.$$

This concludes our analysis of the graphical properties of the dependency structure constructed by Algorithm 9.2. In propositions (9.1) - (9.6), we proved a number of axiomatic properties for $Dep(X_k)$ based on the fact that we are given a conflict-free full minimum cover. Before stating the main result of this section, we need to derive one additional property about $Dep(X_k)$ using the graphical properties in propositions (9.9) and (9.10).

Recall that

$$\overline{X_i W_{i,1}} = \{h \mid h \in \mathcal{R}^{k-1} \text{ and } h \cap W_{i,1} \neq \emptyset\},$$

and from proposition (9.10),

$$\{h\} \;=\; \overline{X_1 W_{1,1}} \cap \ldots \cap \overline{X_{k-1} W_{k-1,1}}.$$

Since

$$X_i W_{i,1} \;=\; \cup_{h \in \overline{X_i W_{i,1}}} h,$$

it follows that $X_k \subseteq h$, where

$$h \;=\; X_1 W_{1,1} \cap \ldots \cap X_{k-1} W_{k-1,1}.$$

We are now ready to state the main result of this chapter.

Theorem 9.2 The dependency structure $\mathcal{R}^k$ constructed from Algorithm 9.2 is a perfect-map of the BMVDs:

$$\begin{array}{ccc} X_1 & \Rightarrow\Rightarrow & W_{1,1} \mid W_{1,2} \mid \ \ldots \ \mid W_{1,m_1}, \\ & \vdots & \end{array}$$

$$
\begin{aligned}
X_i &\Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,m_i}, \\
&\vdots \\
X_{k-1} &\Rightarrow\Rightarrow W_{k-1,1} \mid W_{k-1,2} \mid \ldots \mid W_{k-1,m_{k-1}}. \\
X_k &\Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,m_k}.
\end{aligned} \tag{9.90}
$$

Proof: We will prove this claim by induction.
(i) Basic step: $h = \{R\}$,

$$X_1 \Rightarrow\Rightarrow W_{1,1} \mid W_{1,2} \mid \ldots \mid W_{1,m_1}.$$

Obviously, $\mathcal{R}^1$ is a perfect-map for the above BMVDs, and X_1 is the J-key of $\mathcal{R}^1$. Inductive hypothesis: $\mathcal{R}^{k-1}$ is a perfect-map for equation (9.89), and $X_1, X_2, \ldots, X_{k-1}$ are the J-keys of $\mathcal{R}^{k-1}$.

Based on previous discussions, we have the following observations. Proposition (9.9) states there exists one and only one hyperedge $h \in \mathcal{R}^{k-1}$ that contains X_k. Proposition (9.10) indicates that this hyperedge h can be expressed as:

$$h = X_1W_{1,1} \cap \ldots \cap X_{k-1}W_{k-1,1} = \cap_{i=1}^{k-1} X_iW_{i,1}.$$

We have shown in proposition (9.7) that

$$Dep'(X_i \cap X_k) \subseteq Dep(X_k),$$

for $i = 1, \ldots, k-1$. Since $W_{i,1}$ and $Dep'(X_i \cap X_k)$ are disjoint,

$$(\cap_{i=1}^{k-1} X_iW_{i,1}) \cap (\cup_{i=1}^{k-1}(\cup \{W \mid W \in Dep'(X_i \cap X_k)\})) = \emptyset.$$

By definition, $h = \cap_{i=1}^{k-1} X_iW_{i,1}$ and thereby:

$$h \cap (\cup_{i=1}^{k-1}(\cup \{W \mid W \in Dep'(X_i \cap X_k)\})) = \emptyset. \tag{9.91}$$

On the other hand, we have shown in proposition (9.2) that each $X_iW_{i,1}$ has a *nonempty* intersection with every element in

$$D \equiv Dep(X_k) - Dep'(X_i), \; i = 1, \ldots, k-1.$$

Thus, h has a nonempty intersection with every $W_{k,j} \in D$.

As a result of applying the dependencies

$$X_k \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,m_k}$$

to the hypergraph $\mathcal{R}^{k-1}$, we obtain the hypergraph $\mathcal{R}^k$ which contains the following *new* hyperedges from h:

$$\{ \ R_{k,j} = X_k(h \cap W_{k,j}) \mid W_{k,j} \in D \ \}.$$

Obviously, $\mathcal{R}^k$ is an acyclic hypergraph if $\mathcal{R}^{k-1}$ is acyclic. Moreover, $X_1, X_2, \ldots, X_{k-1}$ are J-keys of $\mathcal{R}^k$ and when X_i $(1 \leq i \leq k-1)$ is deleted from $\mathcal{R}^k$, the disconnected components of X_i are the same as those when X_i is deleted from $\mathcal{R}^{k-1}$.

For every $W_{k,j} \in D$ such that $X_k W_{k,j}$ does not contain any key,

$$h \cap W_{k,j} = W_{k,j}.$$

Whenever $X_k W_{k,j}$ contains some key(s), $W_{k,j} \in D$, then $W_{k,j}$ contains all the hyperedges in:

$$\cup_{i=1}^{k-1} \ (Dep'(X_i) - Dep'(X_i \cap X_k) \ )). \tag{9.92}$$

Obviously, every hyperedge in $\mathcal{R}^{k-1}$ must be connected to h through some key(s) in h. In particular, every hyperedge in equation (9.92) is connected to h by some key(s) in h. It immediately follows that all attributes in $h - W_{k,j}$ are connected to all attributes in $h \cap W_{k,j}$ by some key(s) in h. In other words, all the attributes in $W_{k,j}$ will comprise one and only one disconnected component when X_k is deleted from $\mathcal{R}^k$. We can immediately conclude that when X_k is deleted from $\mathcal{R}^k$, the resulting disconnected components are exactly equal to the elements in $Dep(X_k)$. By the inductive hypothesis that $\mathcal{R}^{k-1}$ is a perfect-map for equation (9.89), it immediately follows that $\mathcal{R}^k$ is a perfect-map for equation (9.90). $\Box$

Theorem 9.2 indicates that the output acyclic hypergraph is a *perfect-map* of the input conflict-free full minimal cover. That is, every BMVD logically implied by the cover can be inferred from the acyclic hypergraph, and every BMVD inferred from the acyclic hypergraph is logically implied by the cover.

We now show that a conflict-free full minimum cover G has a unique dependency structure $\mathcal{R}$. Let $q = (X_1, X_2, \ldots, X_k, X_l, \ldots, X_p)$ be a p-ordering sequence of the keys $\mathbf{X}$ of G. If two keys X_k and X_l are not subsets of each other, then $q = (X_1, X_2, \ldots, X_l, X_k, \ldots, X_p)$ is also a valid p-ordering sequence. We say that q' is obtained from q by a *2-permutation* [55], namely, by permuting a neighbouring pair of keys which are not subsets of each other.

Let $\mathbf{X}$ be the keys of a conflict-free full minimum cover G. We write $\mathcal{R}_q$ to denote the dependency structure constructed as output by Algorithm 9.2 using G with p-ordering sequence q.

Lemma 9.1 Let G be a conflict-free full minimum cover and $q = (X_1, X_2, \ldots, X_{k-1}, X_k, X_l, \ldots, X_p)$ be a p-ordering sequence of the keys $\mathbf{X}$ of G. Let $q' = (X_1, X_2, \ldots, X_{k-1}, X_l, X_k, \ldots, X_p)$ be another p-ordering sequence obtained from q by a 2-permutation. Then $\mathcal{R}_q = \mathcal{R}_{q'}$.

Proof: Let $\mathcal{R}^{k-1}$ be the intermediate dependency structure constructed by Algorithm 9.2 after $k-1$ iterations. If X_k and X_l are contained in distinct hyperedges of $\mathcal{R}^{k-1}$, then the claim follows trivially. Suppose X_k and X_l are contained in the same hyperedge $h \in \mathcal{R}^{k-1}$. Recall that propositions (1) - (5) explicitly state the form that $Dep(X_k)$ and $Dep(X_l)$ have in relationship to each other. That is,

$$X_k \Rightarrow\Rightarrow W_1 \mid \ldots \mid W_s \mid W_{k,1} \mid \ldots \mid W_{k,i} \mid (X_l - X_k)(Z)(W_{l,1} \cdots W_{l,j}),$$

and

$$X_l \Rightarrow\Rightarrow W_1 \mid \ldots \mid W_s \mid W_{l,1} \mid \ldots \mid W_{l,j} \mid (X_k - X_l)(Z)(W_{k,1} \cdots W_{k,i}),$$

where each $W_1, \ldots, W_s$ is in $Dep(X_k \cap X_l)$, and Z may be empty.

Suppose q is the p-ordering sequence used. Obviously, $W_1 \cdots W_s \cap h = \emptyset$. By definition, $h \in \mathcal{R}^{k-1}$ is replaced with the hyperedges:

$$R_1 = X_k(W_{k,1} \cap h),$$

$$
\begin{aligned}
R_2 &= X_k(W_{k,2} \cap h), \\
&\vdots \\
R_i &= X_k(W_{k,i} \cap h), \\
R_{i+1} &= X_k(((X_l - X_k)(Z)(W_{l,1} \cdots W_{l,j})) \cap h) = (X_k X_l Z)(W_{l,1} \cdots W_{l,j}) \cap h).
\end{aligned}
$$

Now $X_l \subseteq R_{i+1} \in \mathcal{R}^k$. By definition, R_{i+1} is replaced with the hyperedges:

$$
\begin{aligned}
R'_1 &= X_l(W_{l,1} \cap R_{i+1}), \\
R'_2 &= X_l(W_{l,2} \cap R_{i+1}), \\
&\vdots \\
R'_j &= X_l(W_{l,j} \cap R_{i+1}), \\
R'_{j+1} &= X_l(((X_k - X_l)(Z)(W_{k,1} \cdots W_{k,i})) \cap R_{i+1}) \\
&= (X_l X_k Z)(W_{k,1} \cdots W_{k,i}) \cap R_{i+1}) \\
&= (X_l X_k Z)(W_{k,1} \cdots W_{k,i}) \cap (X_k X_l Z)(W_{l,1} \cdots W_{l,j}) \cap h = (X_l X_k Z) \cap h.
\end{aligned}
$$

Thus, $h \in \mathcal{R}^{k-1}$ is replaced with the set of hyperedges

$$
R_1, R_2, \ldots, R_i, R'_1, R'_2, \ldots, R'_j, (X_l X_k Z) \cap h \tag{9.93}
$$

in $\mathcal{R}^{k+1}$.

Now suppose, on the other hand, that q' is the p-ordering used in Algorithm 9.2. We will show that the intermediate constructed dependency structure $\mathcal{R}^{k+1}$ is the same. By definition, $h \in \mathcal{R}^{k-1}$ is replaced with the hyperedges:

$$
\begin{aligned}
R'_1 &= X_l(W_{l,1} \cap h), \\
R'_2 &= X_l(W_{l,2} \cap h), \\
&\vdots \\
R'_j &= X_l(W_{l,j} \cap h), \\
R'_{j+1} &= X_l(((X_k - X_l)(Z)(W_{k,1} \cdots W_{k,i})) \cap h) = (X_l X_k Z)(W_{k,1} \cdots W_{k,i}) \cap h).
\end{aligned}
$$

Now $X_k \subseteq R'_{j+1} \in \mathcal{R}^k$. By definition, R'_{j+1} is replaced with the hyperedges:

$$
R_1 = X_k(W_{k,1} \cap R'_{j+1}),
$$

$$
\begin{aligned}
R_2 &= X_k(W_{k,2} \cap R'_{j+1}), \\
&\vdots \\
R_i &= X_k(W_{k,i} \cap R'_{j+1}), \\
R_{i+1} &= X_k(((X_l - X_k)(Z)(W_{l,1} \cdots W_{l,j})) \cap R'_{j+1}) \\
&= (X_k X_l Z)(W_{l,1} \cdots W_{l,j}) \cap R'_{j+1}) \\
&= (X_k X_l Z)(W_{l,1} \cdots W_{l,j}) \cap (X_l X_k Z)(W_{k,1} \cdots W_{k,i}) \cap h = (X_k X_l Z) \cap h.
\end{aligned}
$$

Thus, $h \in \mathcal{R}^{k-1}$ is replaced with the set of hyperedges

$$R'_1, R'_2, \ldots, R'_j, R_1, R_2, \ldots, R_i, (X_k X_l Z) \cap h \tag{9.94}$$

in $\mathcal{R}^{k+1}$.

The desired result is obtained since equation (9.93) is identical to equation (9.94). $\Box$

Theorem 9.3 A conflict-free full minimum cover G has a unique dependency structure. That is, the particular p-ordering sequence used in Algorithm 9.2 is immaterial.

Proof: We show by induction that any valid p-ordering can be transformed into any other valid p-ordering sequence through a series of 2-permutations. Let $q = (X_1, X_2, \ldots, X_p)$ be any valid p-ordering sequence of the keys $\mathbf{X}$ of a conflict-free full minimum cover. Obviously, any valid p-ordering sequence with two keys in q transposed can be obtained from q by a 2-permutation. Suppose q_k is any valid p-ordering sequence obtainable from q by k 2-permutations. Let q_{k+1} be any valid p-ordering sequence with two keys in q_k transputed. By definition, q_{k+1} can be obtained from q_k by a 2-permutation. It easily follows that q_{k+1} is obtainable from q by $k+1$ 2-permutations. Lemma 9.1 demonstrates that if two neighbouring keys X_k and X_l are not subsets of each other, then they can be interchanged in the p-ordering sequence without affecting the output dependency structure $\mathcal{R}$. Thus, the constructed dependency structure $\mathcal{R}_q$ is identical to $\mathcal{R}_{q_{k+1}}$ $\Box$

9.1.2 Eliciting the Quantitative Component

In order to completely define a probabilistic network, one must specify the dependency structure and the corresponding probability tables. In the last section, we explicitly demonstrated a robust automated method for constructing the dependency structure of a Markov network over a multi-agent problem domain. However, the corresponding potentials are not necessarily uniquely definable. On the other hand, the conditional probability tables corresponding to the dependency structure of a Bayesian network can be uniquely specified. Therefore, it is useful to transform the constructed acyclic hypergraph into a DAG in order to elicit the quantitative component of the probabilistic network. The potentials of the constructed acyclic hypergraph in the last section can then be defined in terms of the elicited conditional probability tables.

We first motivate the need for developing an automated process for constructing the acyclic hypergraph of a Markov network, and subsequently transforming it into a DAG. The reason is the inherent difficulty in directly constructing the DAG of a Bayesian network in a multi-agent environment. As already mentioned, it is not realistic to expect the domain experts to manually construct the dependency structure since the problem domain may be much larger than the single-agent case, and perhaps distributed. One suggestion would be to learn the dependency structure from observed data. It is not entirely clear, however, how those learning methods developed for the single-agent environment can be applied, let alone how one might obtain a reliable sample. Thus, constructing the multi-agent dependency structure amounts to finding a method to combine the known conditional independency information supplied by the individual domain experts. One previously proposed method [102] constructs the dependency structure of a multi-agent Bayesian network by verifying that the dependency structure formed by *connecting* the individual agent DAGs is *acyclic*. However, we now demonstrate that the acyclicity condition is too restrictive.

Consider the situation where two agents wish to form a probabilistic multi-agent reasoning system. According to [102], each agent supplies a respective DAG as shown in Figure 9.1. It can be easily verified that the combined dependency structure contains the cycle $A_2 \rightarrow A_5 \rightarrow A_2$. One might wonder, since the same conditional

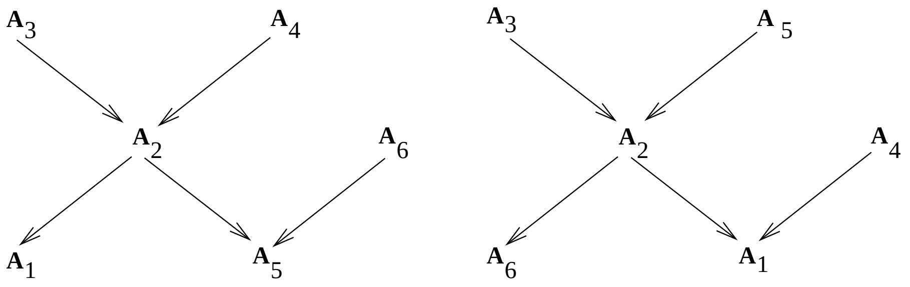

Figure 9.1: The respective DAGs of two agents who wish to form a multi-agent system.

independency may be expressed in a variety of DAGs, if the agents can supply other equivalent DAGs such that the combined dependency structure is in fact a DAG. Our example explicitly demonstrates that this does not always work. DAGs which express precisely the same conditional independencies have the same links and uncoupled head-to-head nodes [86]. (An uncoupled head-to-head node is a node A with two incoming arrows (B, A) and (C, A) with no link between B and C.) By construction, each DAG in Figure 9.1 has no other equivalent DAG other than itself. Thus, Xiang's method would state that these two agents cannot form a multi-agent system. However, consider the acyclic hypergraphs in Figure 9.2 obtained by sacrificing the *embedded* probabilistic conditional independencies represented in the respective DAGs in Figure 9.1.

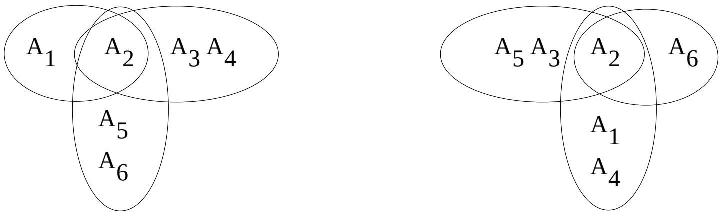

Figure 9.2: The agent dependency structures, reflecting only nonembedded probabilistic conditional independencies, obtained from the respective DAGs in Figure 9.1.

The first agent would supply the independency information

$$\{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3, A_4\} \mid \{A_5, A_6\},$$

while the second agents supplies

$$\{A_2\} \Rightarrow\Rightarrow \{A_1, A_4\} \mid \{A_3, A_5\} \mid \{A_6\}.$$

Note that neither agent has knowledge of the BMVD $A_2 \Rightarrow\Rightarrow A_3$. The dependency basis $Dep(A_2)$ can be determined by applying Algorithm 9.1 with these BMVDs as input:

$$Dep(A_2) = \{\{A_1\}, \{A_3\}, \{A_4\}, \{A_5\}, \{A_6\}\}.$$

Supplying this conflict-free full minimum cover as input to Algorithm 9.2 results in the multi-agent dependency structure $\mathcal{R}$:

$$\mathcal{R} \;\; = \;\; \{\{A_1, A_2\}, \{A_2, A_3\}, \{A_2, A_4\}, \{A_2, A_5\}, \{A_2, A_6\}\}.$$

Note that the BMVD $A_2 \Rightarrow\Rightarrow A_3$, which was previously unknown to each agent, can be inferred from the combined multi-agent dependency structure $\mathcal{R}$. The important point to realize is that the BMVD $A_2 \Rightarrow\Rightarrow A_3$ was logically implied by the *combination* of the individual domain expert independency information. Thereby, not only would our method allow the agents to form a multi-agent system, but our method may detect independencies that are logically implied by the combination of all independencies.

One may also suggest directly constructing a multi-agent DAG by simply using the semi-graphoid axioms [66] for probabilistic conditional independencies in place of the BMVD axiomatization. However, we remind the reader that the semi-graphoid axioms are not complete for probabilistic conditional independencies. Thus, our approach to constructing a multi-agent probabilistic network is to first construct a Markov network, and then transform it into a Bayesian network. (In fact, the same approach was recently taken in [45] in terms of a *learning* approach.) We now turn our attention to incorporating the BEMVDs.

Since the individual domain experts may supply embedded independency information, and the contraction axiom (BEMVD-IL4) is *sound* for inferring new nonembedded independencies from given embedded and nonembedded independencies, we show here how the supplied embedded conditional independencies can be used in constructing the dependency structure of a Markov network. This process will utilize the fact that the BMVD axioms are not only complete for nonembedded BMVDs, but are in fact complete for deriving all embedded BMVDs from other embedded BMVDs as long as the embedded BMVDs are defined over the same *fixed* set of attributes. The contraction axiom (BEMVD-IL4) can then be applied to derive BMVDs logically implied by other BMVDs defined over a *mixed* set of attributes.

Let G be a set of all BMVDs (not necessarily nonembedded) over the set R of all attributes in the multi-agent problem domain supplied by the individual domain experts. For every BMVD $X \Rightarrow\Rightarrow Y|Z$ in G, construct the set G^{XYZ} of *nonembedded* BMVDs over XYZ as follows:

$$G^{XYZ} = \{\ X \Rightarrow\Rightarrow Y|Z \ \ | \ \ X \Rightarrow\Rightarrow V|W \text{ in } G \ \text{ and } Y \subseteq V,\ Z \subseteq W\ \}.$$

For example, consider the set G of BMVDs over $R = \{A_1, A_2, A_3, A_4\}$:

$$G \ = \ \{\ \{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3\},\ \{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\}\ \}.$$

We then construct the sets $G^{\{A_1,A_2,A_3,A_4\}}$ and $G^{\{A_1,A_2,A_3\}}$ of respective nonembedded BMVDs as follows:

$$G^{\{A_1,A_2,A_3,A_4\}} \ = \ \{\ \{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\}\ \}, \tag{9.95}$$

$$G^{\{A_1,A_2,A_3\}} \ = \ \{\ \{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3\},\ \{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\}\ \}. \tag{9.96}$$

We apply the process described earlier to remove redundancy, detect inconsistency, and compute a conflict-free full minimum cover for each of the G^{XYZ} above. This set of conflict-free full minimum covers reflects all logically implied conditional independency information for each fixed $XYZ \subset R$. For each set of nonembedded BMVDs over XYZ in equations (9.95) and (9.96), the respective conflict-free full minimum cover, written G_c^{XYZ}, is:

$$G_c^{\{A_1,A_2,A_3,A_4\}} = \{\ Dep(\{A_2, A_3\}) \ = \ \{\{A_1\}, \{A_4\}\}\ \}, \tag{9.97}$$

$$G_c^{\{A_1,A_2,A_3\}} = \{\ Dep(\{A_2\}) \ = \ \{\{A_1\}, \{A_3\}\}\ \}. \tag{9.98}$$

The contraction axiom (BEMVD-IL4) is now applied to derive new conditional independencies from other conditional independencies defined on *mixed* sets of attributes. Applying (BEMVD-IL4) on the BMVD $\{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3\}$ over $\{A_1, A_2, A_3\}$ in equation (9.98) and the BMVD $\{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\}$ over $\{A_1, A_2, A_3, A_4\}$ in equation (9.97), we derive the new nonembedded BMVD $\{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3, A_4\}$ over $\{A_1, A_2, A_3, A_4\}$. We add this BMVD to the set $G^{\{A_1,A_2,A_3,A_4\}}$ in equation (9.95), namely,

$$
\begin{aligned}
& G^{\{A_1,A_2,A_3,A_4\}} \cup \{\ \{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3, A_4\}\ \} \\
= \ & \{\{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\}\ \} \cup \{\ \{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3, A_4\}\ \} \\
= \ & \{\{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\},\ \{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3, A_4\}\ \}.
\end{aligned}
$$

An updated conflict-free full minimum cover $G_c^{\{A_1,A_2,A_3,A_4\}}$ for the nonembedded BMVDs over $\{A_1, A_2, A_3, A_4\}$ is then:

$$G_c^{\{A_1,A_2,A_3,A_4\}} = \{\ Dep(\{A_2\}) \ = \ \{\{A_1\}, \{A_3, A_4\}\}\ \}. \tag{9.99}$$

Note that the BMVD $\{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\}$ in equation (9.97) is now redundant in equation (9.99). This demonstrates that the contraction inference axiom can derive BMVDs not derivable using the BMVD axiomatization.

The conflict-free full minimum cover G_c^R and a p-ordering sequence can be supplied as input to Algorithm 9.2. The output dependency structure of the multi-agent Markov network $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ is a perfect-map of G_c^R. We construct a DAG which reflects precisely the same probabilistic conditional independencies as $\mathcal{R}$ as follows. Define a hypertree construction ordering $R_1, R_2, \ldots, R_n$. Let a branching function $b(i)$ be $b(i) = i - 1$. The attributes R in $\mathcal{R}$ are *sequentially* labelled following the hypertree construction ordering $R_1, R_2, \ldots, R_n$ satisfying the following two conditions:

(i) Let $A_j, A_k \in R_1$. If $A_j \notin R_2$ and $A_k \in R_2$, then $j < k$.

(i) Let $A_j, A_k \in R_i$, $i = 2, \ldots, n$. If $A_j \in R_{b(i)}$ and $A_k \notin R_{b(i)}$, then $j < k$.

Clearly, such a labelling is not unique. Construct a DAG $D = (R, E)$, where

$$E = \{A_j \rightarrow A_k \mid j < k,\ A_j, A_k \in R_i\},$$

for all $R_i \in \mathcal{R}$. For example, consider the acyclic hypergraph $\mathcal{R} = \{R_1, R_2, R_3, R_4\}$ in Figure 3.2. Following the hypertree construction ordering R_1, R_2, R_3, R_4, the DAG in Figure 9.3 could be constructed as the dependency structure of the multi-agent Bayesian network. (The directed edge (A_2, A_3) could be equivalently replaced by (A_3, A_2).)

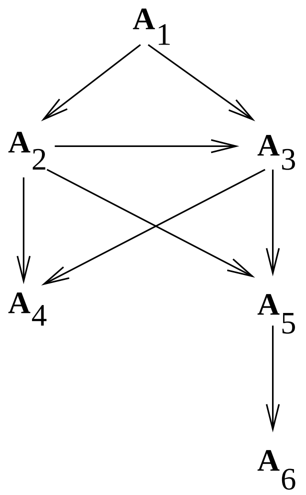

Figure 9.3: The DAG representing the dependency structure of the multi-agent Bayesian network.

Notice that the only probabilistic conditional independencies inferred from the DAG using d-separation [66] are exactly those nonembedded conditional independencies inferred from the acyclic hypergraph $\mathcal{R}$. It is assumed that the domain experts are able to specify the conditional probability tables corresponding to the constructed DAG, namely, $p(A_3)$, $p(A_1|A_3)$, $p(A_2|A_1, A_3)$, $p(A_4|A_1, A_2)$, $p(A_5|A_1, A_2)$, and $p(A_6|A_5)$. These elicited conditional probability tables are represented in our probabilistic relational database model as probabilistic relations. At this point, the multi-agent Bayesian network is completely defined.

Since each agent reasons with a particular subset of variables in the entire network, it is necessary to section the constructed multi-agent dependency structure. The multiply sectioned Bayesian network technique [103] can be applied for this purpose. Thus each agent initially is given a *portion* of the constructed DAG representing the dependency structure of the multi-agent Bayesian network. However, in practice, it is useful to transform the Bayesian network into a Markov network in order to take advantage of the techniques developed for computing marginal distributions. Even though the original DAG has been sectioned, this transformation can still be accomplished through cooperation of the agents [103]. Thereby, the dependency structure of the multi-agent system is again a hypergraph, but the hyperedges are distributed amongst the agents. At this point, the multi-agent system is ready for user input. In the next section, we discuss techniques for probabilistic reasoning in a distributed multi-agent environment.

9.2 Multi-Agent Reasoning

In this section, we turn our attention to performing inference in a multi-agent environment. We demonstrate that inconsistency may arise due to asynchronous access to the knowledge base and also hardware and software failure. Protocols are subsequently introduced to ensure *consistency*.

9.2.1 Asynchronous Control Protocols

We propose a model for multi-agent probabilistic reasoning in a distributed environment [93]. Unlike other methods, this model is capable of processing input in a truly asynchronous fashion. Asynchronous control protocols and a method for processing evidence are developed to ensure global consistency at all times. The proposed system then extends beyond an interpretive system since the now well-defined concept of a distributed request can be introduced. Techniques are also suggested to reduce data transmission in answering this type of request.

Probabilistic reasoning has become an accepted framework for managing uncertain knowledge in a centralized environment. There is, however, an emerging interest in extending the probabilistic formulism into a distributed multi-agent environment [101]. This environment consists of multiple agents perhaps located at distinct sites. In this section, we assume that each agent's knowledge is represented by a Markov distribution [38]. A system which accepts evidence and produces a higher level description of the environment is called *interpretive*. One such cooperative, interpretive system which applies probabilistic techniques for managing uncertainty in a distributed multi-agent environment was proposed in [101]. That system, while innovative, exhibits several undesirable characteristics including: (i) each agent processes evidence and answers local probabilistic requests *serially*; (ii) each agent's knowledge base is *private*. That is, the agents may share evidence entering their respective local sites, but they do not share their knowledge bases. Consequently, each agent can only answer *local* requests. There is no concept of a distributed request; (iii) the entire system has to be brought *off-line* periodically such that *global* consistency can be restored. Although techniques were suggested to reduce this off-line time, shutting down the system even for a relatively short time could be unacceptable to many multi-agent systems; (iv) no provisions are made for transaction or agent *failure*.

Aside from these undesirable characteristics, the claim was made in [101] that the proposed system processes evidence "asynchronously." While the claim is true, it should be somewhat qualified. Even though the agents process evidence "asynchronously" with respect to each other, each agent processes local evidence *serially*. More importantly, each agent only updates the *local* knowledge base and does not immediately share evidence with the other agents. Thereby, knowledge inconsistency of two agents having unique belief on common variables is allowed to occur. Thus, processing evidence in this "asynchronous" fashion avoids the fundamental problem of concurrent access to *one* knowledge base, as well as leads to an inconsistent global knowledge base.

In this section, we propose a multi-agent system which maintains global consistency at all times. As it may be necessary in many cooperative multi-agent systems for one agent to access the knowledge base of another, we remove the restriction of private knowledge bases made in [101]. That is, we allow the agents to access the knowledge base of one another. The inconsistency problems associated with asynchronous access to a knowledge base *must* now be addressed. Asynchronous control protocols and a method for processing evidence are suggested to ensure global consistency at all times, without the need of ever bringing any agent in the system off-line. By assuming public knowledge bases and maintaining global consistency, the concept of a *distributed* request is now well defined extending our model beyond merely an interpretive one. Techniques are developed to reduce the amount of data transmitted between agents in answering distributed requests.

Initially, the local propagation techniques are applied to bring the multi-agent system into a globally consistent state. (For now global consistency can be viewed as all agents having the same belief on all common variables.) The system is then ready for user interaction and sensory input. Whereas in [101], each agent processes input from its *single* user and multiple sensors *serially*, our probabilistic relational model is capable of processing input from *multiple* users and multiple sensors *asynchronously*. The situation is further complicated by assuming that the knowledge base of each agent is *public*. The objective of this section is to explicitly demonstrate the types of problems that may arise once an agent extends from processing input *serially* to processing input *asynchronously*. In the next section, asynchronous control protocols will be developed which will maintain global consistency.

Sensory input or user interaction can be categorized into two cases, namely, collected evidence and probabilistic requests. Probabilistic requests involve only read operations on a knowledge base, while performing belief update with collected evidence may require read and write operations. In [101], processing evidence "asynchronously" means each agent processes evidence *serially* and does *not* immediately pass the evidence to the other private knowledge bases. In this section, on the other hand, the focus is on processing both evidence and probabilistic requests in a truly *asynchronous* fashion. The following examples explicitly demonstrate the types of problems which may arise in meeting this objective.

It is important to note that if no asynchronous protocols are present, inconsistency can arise from the asynchronous execution of the input *only*. That is, this inconsistency is a result of executing the input asynchronously and is not a consequence of invalid input. Also, these examples introduce the concept of a *transaction*. For now a transaction can be simply viewed as the necessary *read* and *write* operations required to execute the input.

The following example demonstrates that input may be lost due to asynchronous access to a knowledge base.

Example 9.6 Consider the situation of a patient undergoing two medical examinations: one for his heart in the pulmonary ward and another for his lungs in the respiratory ward. Suppose we have a distribution $\mathbf{r}(S)$ on $S = \{heart, lung\}$ for this patient as shown in Figure 9.4. The values 0 and 1 represent a negative, positive result, respectively. Let the input of the pulmonary ward be represented by transaction T_1 and the input of the respiratory ward by transaction T_2. Let T_1 have the evidence $heart = 0$ while T_2 has evidence $lung = 0$. If both transactions enter the system asynchronously, the following execution could occur:

$$Read_1(S); Read_2(S); Write_1(S); Write_2(S).$$

T_1 first reads $\mathbf{r}(S)$, then so does T_2. Transaction T_1 then enters its collected evidence and derives the distribution $\mathbf{r}(S)'$ as shown in Figure 9.5 (left). It then writes $\mathbf{r}(S)'$ back to memory overwriting $\mathbf{r}(S)$. At the same time, T_2 enters its collected evidence and derives the distribution $\mathbf{r}(S)''$ shown in Figure 9.5 (right). Transaction T_2 then writes $\mathbf{r}(S)''$ back to memory overwriting $\mathbf{r}(S)'$. The result of this execution is as if only T_2 had entered evidence. However, the final result of all the collected evidence regarding the patient should be the distribution containing the single tuple $< 0\ 0\ 1.0 >$. That is, we have lost the evidence $heart = 0$ contained in T_1.

$\mathbf{r}(S) =$

$heart$	$lung$	A_p
0	0	0.2
0	1	0.2
1	0	0.3
1	1	0.3

Figure 9.4: A probability distribution $\mathbf{r}(S)$ on $S = \{heart, lung\}$.

Example 9.6 demonstrates that evidence may be lost when two transactions both read the same current distribution and subsequently both update the distribution. On the other hand, the following example demonstrates that an incorrect value may be returned.

$$\mathbf{r}(S)' = \begin{array}{|ccc|} \hline heart & lung & A_{p'} \\ \hline 0 & 0 & 0.5 \\ 0 & 1 & 0.5 \\ \hline \end{array} \qquad \mathbf{r}(S)'' = \begin{array}{|ccc|} \hline heart & lung & A_{p''} \\ \hline 0 & 0 & 0.4 \\ 1 & 0 & 0.6 \\ \hline \end{array}$$

Figure 9.5: Distributions $\mathbf{r}(S)'$ (left) representing the evidence entered by T_1 only, and $\mathbf{r}(S)''$ (right) representing the evidence entered by T_2 only.

Example 9.7 Consider a distribution $\mathbf{r}(S)$ on $S = A_1A_2$ as depicted in Figure 9.6 (left). Suppose one transaction T_1 is computing the marginal of $\mathbf{r}(S)$ onto a subset of variables A_1, namely $\tau_{A_1}(\mathbf{r})$. After T_1 starts reading tuples from $\mathbf{r}(S)$, an asynchronous transaction T_2 updates $\mathbf{r}(S)$ with evidence, say $A_1 = 1$ and $A_2 = 1$, by writing $< 1\ 1\ 1.0 >$ and then writing the probability value for all other tuples to 0.0. That is, we have the detailed execution (we omit the last three writes of T_2):

$$R_1(< 0\ 0\ 0.1 >); R_1(< 0\ 1\ 0.2 >); R_1(< 1\ 0\ 0.3 >); W_2(< 1\ 1\ 1.0 >); R_1(< 1\ 1\ 1.0 >),$$

where R and W denote *Read* and *Write*, respectively. That is, if T_1 reads the first three tuples of relation $\mathbf{r}(S)$, then T_2 updates the fourth tuple, then it is possible for T_1 to compute the incorrect marginal distribution $\tau_{A_1}(\mathbf{r})$ depicted in Figure 9.6 (right).

$$\mathbf{r}(S) = \begin{array}{|ccc|} \hline A_1 & A_2 & A_p \\ \hline 0 & 0 & 0.1 \\ 0 & 1 & 0.2 \\ 1 & 0 & 0.3 \\ 1 & 1 & 0.4 \\ \hline \end{array} \qquad \tau_{A_1}(\mathbf{r}) = \begin{array}{|cc|} \hline A_2 & A_{p(A_2)} \\ \hline 0 & 0.3 \\ 1 & 1.3 \\ \hline \end{array}$$

Figure 9.6: A probability distribution $\mathbf{r}(S)$ (left). The incorrect marginal distribution $\tau_{A_1}(\mathbf{r})$ (right).

Example 9.7 demonstrates that inconsistent retrieval may occur when a retrieval transaction reads some tuples in a relation $\mathbf{r}(S)$ (before another transaction containing collected evidence modifies the relation) and subsequently reads other tuples which have been modified.

The following example demonstrates that an incorrect value may be read as a result of improper belief updating in a distributed environment.

Example 9.8 Consider two agents $\mathcal{A}_1$ and $\mathcal{A}_2$ in a multi-agent environment. The knowledge bases of agents $\mathcal{A}_1$ and $\mathcal{A}_2$ represent the relations $\mathbf{r}(A_1A_2)$ and $\mathbf{r}(A_2A_3)$ as shown in Figure 9.7. Suppose evidence $A_1 = 0, A_2 = 0$ is collected at $\mathcal{A}_1$, and belief update is performed by agent $\mathcal{A}_1$ on the local knowledge base $\mathbf{r}(A_1A_2)$ *only*. Thus, the knowledge base of the multi-agent system is inconsistent. Suppose further that the request $p(x_2 = 1)$ is issued by a user in the multi-agent system. If the request is submitted to agent $\mathcal{A}_1$, then the answer $p(x_2 = 1) = 0.0$ will be returned. On the other hand, if the request is submitted to agent $\mathcal{A}_2$, then the answer $p(x_2 = 1) = 0.9$ will be returned. The user will not be confident that the answer to the request is correct.

$\mathbf{r}(A_1A_2) =$

A_1	A_2	$A_{p(A_1A_2)}$
0	0	0.1
1	1	0.9

$\mathbf{r}(A_2A_3) =$

A_2	A_3	$A_{p(A_2A_3)}$
0	1	0.1
1	0	0.4
1	1	0.5

Figure 9.7: Relations $\mathbf{r}(A_1A_2)$ (left) and $\mathbf{r}(A_1A_2)$ (right) of agents $\mathcal{A}_1$ and $\mathcal{A}_2$.

We now develop asynchronous control protocols to resolve the above problems. These protocols maintain a consistent global knowledge base and return correct answers to probabilistic requests. We first clarify the key concept of a *transaction* in our discussion. A transaction is a mathematical structure which allows us to model asynchronous knowledge base operations for a given input. Asynchronous execution of several transactions are then modelled in a mathematical structure termed a *history*. Properties which a history must satisfy in order to ensure consistency are stated. In short, the effects of an asynchronous execution of transactions must be equivalent to some serial execution of the same transactions. Finally, a practical method to implement this theory is described. While vast amounts of research on concurrency control exist in relational theory, the discussion in this section draws from Bernstein, Hadzilacos and Goodman [10].

The central notion in our discussion of asynchronous access to a shared knowledge base is the *transaction*. Each input, whether a probabilistic request or collected evidence, is represented as a transaction. The transaction is translated into the necessary read and write operations to be performed on the knowledge base. The agent must support *Read* and *Write* operations to facilitate access to the knowledge base. $Read(X)$ (or $R(X)$) returns the marginal distribution $\mathbf{r}(X)$. $Write(X, val)$ (or $W(X, val)$) will overwrite the distribution $\mathbf{r}$ on X with the tuple(s) and the associated probability values in the new distribution $\mathbf{r}(X)$. (We will write $W(X, val)$ as $W(X)$ if the actual distribution $\mathbf{r}(X)$ is unimportant.)

An agent must execute the read and write operations atomically. From this point of view, it seems as if the read and write operations are executed serially. In the previously proposed system [101], the operations must be executed in a *serial* fashion. However, in our model we can execute knowledge base operations *asynchronously* to improve computer resource utilization and provide faster user response time. To ensure correctness, however, the result of this asynchronous execution of the read and write operations must be the *same* as one from some serial execution of the same operations.

The goal of controlling asynchronous access to a shared knowledge base is to ensure that transactions execute *atomically*. That is, the transactions must not interfere with one another. In our asynchronous system, as read and write operations may execute in parallel, we model transactions as partial orders. That is, a transaction need not specify the relative order of every pair of knowledge base operations that appear in it. We denote the Read (or Write) issued by transaction T_i on the distribution $\mathbf{r}(X)$ by $R_i(X)$ (or $W_i(X)$). For simplified notation, we assume that no transaction reads or writes the same distribution $\mathbf{r}(X)$ more than once. (Note that none of the results in this discussion depend on this assumption.) We now formalize the concept of a transaction.

A *transaction* T_i is a partial order with an ordering relation $<_i$, where

(i) $T_i \subseteq \{R_i(X), W_i(X) \mid \mathbf{r}(X) \text{ is a distribution } \}$;

(ii) If $R_i(X), W_i(X) \in T_i$ then either $R_i(X) <_i W_i(X)$ or $W_i(X) <_i R_i(X)$.

To illustrate asynchronous control concepts, we can naturally depict a transaction T_i as a directed acyclic graph (DAG) with the arrows indicating the ordering defined by $<_i$. That is, the arrows indicate the order in which the knowledge base operations execute.

The concept of a transaction allows us to model asynchronous execution of the operations of a *single* transaction. We now extend this idea to modeling asynchronous transactions. In other words, we can now model asynchronous access of several transactions to a shared knowledge base. Similar to depicting the operations of a transaction, we represent a history as a DAG and do not depict arrows implied by transitivity. Consider the following three transactions:

$$
\begin{aligned}
T_1 &= R_1(X) \to W_1(X), \\
T_2 &= R_2(X) \to W_2(Y) \to W_2(X), \\
T_3 &= R_3(Y) \to W_3(X) \to W_3(Y) \to W_3(Z).
\end{aligned}
$$

A *history* H over $\{T_1, T_2, T_3\}$ may be depicted as the DAG shown in Figure 9.8.

$$
H = \begin{tikzcd}
R_1(X) \arrow[r] & W_1(X) \arrow[d] & & \\
R_3(Y) \arrow[r] & W_3(X) \arrow[r] \arrow[d] & W_3(Y) \arrow[r] \arrow[d] & W_3(Z) \\
 & R_2(X) \arrow[r] & W_2(Y) \arrow[r] & W_2(X)
\end{tikzcd}
$$

Figure 9.8: A history over three transactions $\{T_1, T_2, T_3\}$.

Our goal here is to define when an asynchronous execution of transactions is correct. Executing transactions asynchronously may lead to inconsistency (see Section 3). These undesirable characteristics are entirely the result of the interleaving of the knowledge base operations and are not the result of incorrect transactions. The simplest solution to this problem is not to allow any interleaving of transactions and merely execute the transactions serially as in [101]. A better solution is to permit asynchronous execution of transactions and ensure that the result of this asynchronous execution of transactions is identical to the one obtained from some serial execution of the same transactions. An execution of transactions is *serializable* if it returns the same distribution to the user and has the same effect on the knowledge base as some serial execution of the same transactions. Since serial executions are correct [101], and serializable executions of asynchronous transactions have the same effect as some serial execution, we conclude that serializable executions are also correct. We now formalize these notions in terms of histories.

In order to define serializable histories, we first define the concept of equivalent histories as well as a serial history. We then introduce the notion of a serializable history.

Two histories H and H' are *equivalent*, denoted $H \equiv H'$, if

(i) H and H' are defined over the same set of transactions and have the same knowledge base operations; and

(ii) For any conflicting knowledge base operations P_i and Q_j belonging to transactions T_i and T_j, respectively, if $P_i <_H Q_j$ then $P_i <_{H'} Q_j$.

A history H is *serial* if for every two transactions T_i and T_j that appear in H, either all knowledge base operations of T_i appear before all knowledge base operations of T_j or vice versa.

Thus, a serial history can be seen as an execution of transactions in some order [101].

A history H is *serializable*, if it is equivalent to a serial history H_S.

Determining whether an asynchronous execution of transactions is correct means determining whether the history representing the asynchronous execution of transactions is serializable. The history is deemed serializable or not by examining a *serialization graph* derived from the history.

Let H be a history over $T = \{T_1, T_2, \ldots, T_n\}$. The *serialization graph* (SG) for H, denoted $SG(H)$, is a directed graph whose nodes are the transactions in T that are in H and whose edges are all $T_i \rightarrow T_j$ $(i \neq j)$ such that one of T_i's knowledge base operations precedes and conflicts with one of T_j's knowledge base operations in H.

Theorem 9.4 [10] A history H is *serializable* iff $SG(H)$ is acyclic.

A practical solution to multi-user access on an agent's knowledge base is to use locking. With this tool, each distribution (not necessarily marginal) has a lock associated with it. Before a transaction T_j can execute operations on the distribution, it must first control the associated lock. If another transaction T_i currently controls the lock, then T_j must wait until T_i releases the lock. The concept of locking is used to ensure serializable executions.

Each distribution $\mathbf{r}(X)$ has two locks associated with it, namely a read lock operation $Rl(X)$ and a write lock operation $Wl(X)$. We denote transaction T_i obtaining a read lock (or write lock) on the distribution $\mathbf{r}(X)$ by $Rl_i(X)$ (or $Wl_i(X)$). In the situation where the type of lock is not important, we will denote the lock operation with O, P or Q. Transaction T_i obtaining an arbitrary lock on distribution $\mathbf{r}(X)$ is then denoted by $Ol_i(X)$. (Note that we will sometimes write $Rl_i(X)$ (or $Wl_i(X)$) to denote the operation of setting the lock. The meaning will always be clear from the context.)

An important concept to be used in locking is that of two locks conflicting. Two locks $Ol_i(X)$ and $Pl_j(Y)$ *conflict* if $i \neq j$, $X = Y$, and at least one of Ol and Pl is a write lock. Thereby, two locks on distinct distributions do not conflict, nor do two Read locks by distinct transactions on one distribution, nor a read and a write lock on the same distribution held by the same transaction.

After a transaction T has performed its Read (or Write) operation on the distribution $\mathbf{r}(X)$, it then *releases* the lock, denoted by the *unlock* operation $Ru_i(X)$ (or $Wu_i(X)$).

It is the responsibility of the agent to obtain and release the locks on behalf of the transactions. We are now ready to describe a locking protocol called *two phase locking* which the agent can abide by to ensure consistency.

Two Phase Locking

(i) When a transaction T_i intends to perform an operation $P_i(X)$ on a distribution $\mathbf{r}(X)$, the agent tests if $Pl_i(X)$ conflicts with another lock $Ql_j(X)$ that is already held. If $Pl_i(X)$ conflicts with any other lock, then the agent delays $P_i(X)$ until the conflicting lock is released. On the other hand, if $Pl_i(X)$ does not conflict with any other lock, then the agent sets $Pl_i(X)$ on behalf of T_i and executes the operation $P_i(X)$.

(ii) Once a transaction T_i has released a lock, it may not subsequently obtain any more locks on any distribution.

Part (i) of the two phase locking protocol ensures that conflicting operations on a distribution are executed in the same order as the locks are obtained. Part (ii) is required to ensure that all pairs of conflicting operations of two transactions are executed in the same order. Otherwise, transaction T_j may appear to precede transaction T_i with respect to distribution $\mathbf{r}(X)$, yet follow transaction T_i with respect to another distribution $\mathbf{r}(Y)$. Such an execution is not serializable.

Theorem 9.5 [10] Every two phase locking history H is serializable.

We now discuss some performance issues on answering probabilistic requests by stressing their importance in a distributed environment. In distributed systems, the data transmission between agents should be minimized. We assume here that transmission speeds may be orders of magnitude lower than the transfer rates between disks and main memory (i.e., local I/O operations). Therefore, it is important to reduce the data transmission costs in distributed request processing. We conclude by introducing a method for processing evidence which maintains global consistency at all times.

There are two classes of probabilistic requests, namely, unconditional and conditional. Conditional requests involve selection of certain configurations whereas unconditional requests do not. It is straightforward to adopt the centralized processing techniques for unconditional requests $p(X)$ [96]. Hence, we focus the discussion on the more interesting techniques for conditional requests.

Conditional requests involve selection of certain tuples in computing the answer. In this section we will present a request processing technique to reduce the amount of data transmitted between agents in answering conditional probabilistic requests.

Consider what is involved in computing $\mathbf{r}(X) \otimes \mathbf{r}(Y)$ to answer a conditional probabilistic request, where $\mathbf{r}(X)$ and $\mathbf{r}(Y)$ are represented in the knowledge bases of two different agents. One agent could transmit the *entire* distribution to the other, which could then compute the answer. However, since the Markov join operator $\otimes$ is defined in terms of the product join operator $\times$ (which is partly defined [96] in terms of the natural join operator $\bowtie$), we can take advantage of the query processing techniques developed for standard relational databases to answer conditional probabilistic requests.

Let $\mathbf{r}(X)$ and $\mathbf{r}(Y)$ be two relations. The *semijoin* of $\mathbf{r}(X)$ with $\mathbf{r}(Y)$, written $\mathbf{r}(X) \triangleright \mathbf{r}(Y)$, is the relation

$$\mathbf{r}(X) \triangleright \mathbf{r}(Y) \quad = \quad \pi_{X A_{p(X)}}(\mathbf{r}(X) \bowtie \mathbf{r}(Y)).$$

That is, the semijoin $\mathbf{r}(X) \triangleright \mathbf{r}(Y)$ are the tuples in relation $\mathbf{r}(X)$ that are joinable with tuples in relation $\mathbf{r}(Y)$.

Note that the entire relation $\mathbf{r}(Y)$ is not required in computing $\mathbf{r}(X) \triangleright \mathbf{r}(Y)$ because

$$\mathbf{r}(X) \triangleright \mathbf{r}(Y) \quad = \quad \pi_{X A_{p(X)}}(\mathbf{r}(X) \bowtie \mathbf{r}(Y)) = \mathbf{r}(X) \bowtie \pi_{X \cap Y}(\mathbf{r}(Y)).$$

Now the useful property of semijoin is that

$$\mathbf{r}(X) \bowtie \mathbf{r}(Y) \quad = \quad (\mathbf{r}(X) \triangleright \mathbf{r}(Y)) \bowtie \mathbf{r}(Y). \qquad (9.100)$$

Since $\mathbf{r}(X) \bowtie \mathbf{r}(Y)$ is a necessary step in computing $\mathbf{r}(X) \times \mathbf{r}(Y)$ [96], equation (9.100) indicates that computing $\mathbf{r}(X) \bowtie \mathbf{r}(Y)$ as such will reduce transmission costs between agents whenever $\mathbf{r}(X)$ has a greater cardinality than that of $\pi_{X \cap Y}(\mathbf{r})$ and $\mathbf{r}(X) \triangleright \mathbf{r}(Y)$ together.

Example 9.9 Consider the acyclic hypergraph $\mathcal{R} = \{R_1 = A_1A_2A_3,\ R_2 = A_3A_4A_5\}$. Let relation $\mathbf{r}(A_1A_2A_3A_4A_5)$ satisfy the PAJD $\otimes\mathcal{R}$

$$\mathbf{r}(A_1A_2A_3A_4A_5) \quad = \quad \tau_{A_1A_2A_3}(\mathbf{r}) \otimes \tau_{A_3A_4A_5}(\mathbf{r}),$$

where $\tau_{A_1A_2A_3}(\mathbf{r})$ and $\tau_{A_3A_4A_5}(\mathbf{r})$, depicted in Figure 9.9, represent the knowledge bases of agents $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. Suppose that the probabilistic request

$$p(A_1, A_3, A_5 \mid A_2 = 0, A_4 = 0) \tag{9.101}$$

is issued by a user at $\mathcal{A}_2$. Agent $\mathcal{A}_1$ could transmit the entire relation $\tau_{A_1A_2A_3}(\mathbf{r})$ to $\mathcal{A}_2$ so that $\mathcal{A}_2$ could compute the answer. However, the request can be answered more efficiently using the semijoin operation. First each agent performs local processing to reduce the tuples involved in answering the request. In this case, agent $\mathcal{A}_1$ computes

$$\sigma_{A_2=0}(\tau_{A_1A_2A_3}(\mathbf{r})) \tag{9.102}$$

and agent $\mathcal{A}_2$ determines

$$\sigma_{A_4=0}(\tau_{A_3A_4A_5}(\mathbf{r})) \tag{9.103}$$

The resultant relations in equations (9.102) and (9.103), called *views* [56] in the relational model, are shown in Figure 9.10.

Next agent $\mathcal{A}_2$ transmits $\pi_{A_3}(\sigma_{A_4=0}(\mathbf{r}(A_3A_4A_5)))$ to agent $\mathcal{A}_1$. Agent A_1 computes the semijoin $\sigma_{A_2=0}(\tau_{A_1A_2A_3}(\mathbf{r})) \rhd \sigma_{A_4=0}(\tau_{A_3A_4A_5}(\mathbf{r}))$, namely,

$$\sigma_{A_2=0}(\tau_{A_1A_2A_3}(\mathbf{r})) \bowtie \pi_{A_3}(\sigma_{A_4=0}(\mathbf{r}(A_3A_4A_5))). \tag{9.104}$$

The semijoin $\sigma_{A_2=0}(\tau_{A_1A_2A_3}(\mathbf{r})) \rhd \sigma_{A_4=0}(\tau_{A_1A_2A_3}(\mathbf{r}))$ is shown in Figure 9.11 (left). Agent $\mathcal{A}_1$ transmits this semijoin to agent $\mathcal{A}_2$.

Agent $\mathcal{A}_2$ in turn computes the semijoin

$$\sigma_{A_4=0}(\mathbf{r}(A_3A_4A_5)) \quad \rhd \quad (\sigma_{A_2=0}(\tau_{A_1A_2A_3}(\mathbf{r})) \bowtie \pi_{A_3}(\sigma_{A_4=0}(\tau_{A_1A_2A_3}(\mathbf{r})))), \tag{9.105}$$

namely,

$$\sigma_{A_4=0}(\mathbf{r}(A_3A_4A_5)) \quad \bowtie \quad \tau_{A_3}(\sigma_{A_2=0}(\tau_{A_1A_2A_3}(\mathbf{r})) \bowtie \pi_{A_3}(\sigma_{A_4=0}(\tau_{A_1A_2A_3}(\mathbf{r})))).$$

The semijoin in equation (9.105) is illustrated in Figure 9.11 (right).

Finally, agent $\mathcal{A}_2$ answers the conditional probabilistic request $p(A_1, A_3, A_5 \mid A_2 = 0, A_4 = 0)$ in equation (9.101) by computing the Markov join $\otimes$ of the two semijoins in Figure 9.11.

$\tau_{R_1}(\mathbf{r}) =$

A_1	A_2	A_3	$A_{p(R_1)}$
0	0	0	0.1
1	0	1	0.2
2	1	2	0.3
3	2	3	0.4

$\tau_{R_2}(\mathbf{r}) =$

A_3	A_4	A_5	$A_{p(R_2)}$
0	0	1	0.1
1	1	2	0.2
2	0	3	0.3
3	1	4	0.4

Figure 9.9: Relations $\tau_{R_1}(\mathbf{r})$ (left) and $\tau_{R_2}(\mathbf{r})$ (right) of agents $\mathcal{A}_1$ and $\mathcal{A}_2$.

$\sigma_{A_2=0}(\tau_{R_1}(\mathbf{r})) =$

A_1	A_2	A_3	$A_{p(R_1)}$
0	0	0	0.1
1	0	1	0.2

$\sigma_{A_4=0}(\tau_{R_2}(\mathbf{r})) =$

A_3	A_4	A_5	$A_{p(R_2)}$
0	0	1	0.1
2	0	3	0.3

Figure 9.10: The relations $\sigma_{A_2=0}(\tau_{A_1A_2A_3}(\mathbf{r}))$ (left) and $\sigma_{A_4=0}(\tau_{A_3A_4A_5}(\mathbf{r}))$ (right), where $\tau_{A_1A_2A_3}(\mathbf{r})$ and $\tau_{A_3A_4A_5}(\mathbf{r})$ are shown in Figure 9.9, respectively.

The important point in example 9.9 is that no marginal distribution was transmitted in its entirety even though the multi-agent system is globally consistent.

In [101], the method for processing evidence may lead to global inconsistency. In this section our goal is to develop a method for processing evidence which maintains global consistency at all times. The proposed method for maintaining global consistency takes advantage of the following theorem which relates *pairwise* consistency and *global* consistency in a hypertree.

Theorem 9.6 [24] Let R be a set of variables. Let $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ be a hypertree, where $\bigcup_{i=1}^{n} R_i = R$, and $R_1, R_2, \ldots, R_n$ is a hypertree construction ordering with branching function $b(i) = i-1$, $i = 2, 3, \ldots, n$. Let $\mathbf{r}(R_i)$ be a relation such that

A_1	A_2	A_3	$A_{p(R_1)}$
0	0	0	0.1

A_3	A_4	A_5	$A_{p(R_2)}$
0	0	1	0.1

Figure 9.11: The semijoins $\sigma_{A_2=0}(\tau_{A_1A_2A_3}(\mathbf{r})) \triangleright \sigma_{A_4=0}(\tau_{A_3A_4A_5}(\mathbf{r}))$ (left) in equation (9.104) and $\sigma_{A_4=0}(\mathbf{r}(A_3A_4A_5)) \triangleright (\sigma_{A_2=0}(\tau_{A_1A_2A_3}(\mathbf{r})) \bowtie \pi_{A_3}(\sigma_{A_4=0}(\tau_{A_1A_2A_3}(\mathbf{r}))))$ (right) in equation (9.105).

the relations are *pairwise consistent*, namely,

$$\tau_{R_i \cap R_j}(\mathbf{r}(R_i)) \;\; = \;\; \tau_{R_i \cap R_j}(\mathbf{r}(R_j)),$$

for each pair R_i, R_j with $R_i \cap R_j \neq \emptyset$, $i \neq j$. Then there exists a unique joint probability distribution $\mathbf{r}(R)$ such that $\tau_{R_i}(\mathbf{r}) = \mathbf{r}(R_i)$.

Theorem 9.6 indicates that maintaining pairwise consistency on a hypertree ensures global consistency. In other words, the distribution represented by the knowledge base of each agent is in fact a marginal distribution derived from the same joint probability distribution. Thereby, since our multi-agent system is defined on a hypertree, global consistency can be achieved by developing a method for processing evidence which maintains pairwise consistency.

The method for processing evidence which maintains pairwise consistency is now described. Let $\mathbf{r}(R_i)$ be the marginal distribution represented by the knowledge base of agent A_i. Let $\mathbf{T}$ be the set of tuples to be deleted from $\mathbf{r}(R_i)$ as a result of the collected evidence. Let $Y \subseteq R_i$ be the set of variables which are also represented in the knowledge base of at least one other agent. For each $\mathbf{t} \notin \mathbf{T}$, set the probability value $p'_{R_i}(\mathbf{t})$ as follows:

$$p'_{R_i}(\mathbf{t}) \;\; = \;\; p_{R_i}(\mathbf{t}) \; + \; (\sum_{\mathbf{u} \in \mathbf{T}} p_{R_i}(\mathbf{u}))(\frac{p_{R_i}(\mathbf{t})}{\sum_{\mathbf{v} \notin \mathbf{T}} p_{R_i}(\mathbf{v})}), \qquad \text{if} \; \sum_{\mathbf{v} \notin \mathbf{T}} p_{R_i}(\mathbf{v}) > 0,$$

where $\mathbf{u}[Y] = \mathbf{t}[Y]$, and $\mathbf{v}[Y] = \mathbf{t}[Y]$. Delete the set $\mathbf{T}$ of tuples from $\mathbf{r}(R_i)$.

Let us now demonstrate that performing belief update in this fashion maintains a pairwise consistent knowledge base. By the above construction, we obtain:

$$\begin{aligned}
\frac{p'_{R_i}(\mathbf{t})}{\sum_{\mathbf{v} \notin \mathbf{T}} p'_{R_i}(\mathbf{v})} &= \frac{p_{R_i}(\mathbf{t}) + [\sum_{\mathbf{u} \in \mathbf{T}} p_{R_i}(\mathbf{u})](\frac{p_{R_i}(\mathbf{t})}{\sum_{\mathbf{v} \notin \mathbf{T}} p_{R_i}(\mathbf{v})})}{\sum_{\mathbf{v} \notin \mathbf{T}}(\sum_{\mathbf{u} \in \mathbf{T}} p_{R_i}(\mathbf{u}))} \\
&= \frac{p_{R_i}(\mathbf{t})\frac{\sum_{\mathbf{v} \notin \mathbf{T}} p_{R_i}(\mathbf{v})}{\sum_{\mathbf{v} \notin \mathbf{T}} p_{R_i}(\mathbf{v})} + [\sum_{\mathbf{u} \in \mathbf{T}} p_{R_i}(\mathbf{u})](\frac{p_{R_i}(\mathbf{t})}{\sum_{\mathbf{v} \notin \mathbf{T}} p_{R_i}(\mathbf{v})})}{\sum_{\mathbf{w}} p_{R_i}(\mathbf{w})} \\
&= \frac{\frac{p_{R_i}(\mathbf{t})}{\sum_{\mathbf{v} \notin \mathbf{T}} p_{R_i}(\mathbf{v})}[\sum_{\mathbf{v} \notin \mathbf{T}} p_{R_i}(\mathbf{v}) + \sum_{\mathbf{u} \in \mathbf{T}} p_{R_i}(\mathbf{u})]}{\sum_{\mathbf{w}} p_{R_i}(\mathbf{w})}
\end{aligned}$$

$$= \frac{\frac{p_{R_i}(\mathbf{t})}{\sum_{\mathbf{v} \notin \mathbf{T}} p_{R_i}(\mathbf{v})} \sum_{\mathbf{w}} p_{R_i}(\mathbf{w})}{\sum_{\mathbf{w}} p_{R_i}(\mathbf{w})}$$
$$= \frac{p_{R_i}(\mathbf{t})}{\sum_{\mathbf{v} \notin \mathbf{T}} p_{R_i}(\mathbf{v})}.$$

Hence, pairwise consistency is maintained after the evidence has been processed by agent $\mathcal{A}_i$. That is, for any knowledge base R_j with $R_i \cap R_j \neq \emptyset$, the equality

$$\tau_{R_i \cap R_j}(\mathbf{r}(R_i)') \quad = \quad \tau_{R_i \cap R_j}(\mathbf{r}(R_j))$$

still holds. By Theorem 9.6, the knowledge base of the entire multi-agent system remains *globally* consistent.

Example 9.10 Let $\mathcal{R} = \{R_1 = A_1A_2, R_2 = A_2A_3A_4A_5, R_3 = A_5A_6\}$ be an acyclic hypergraph. Consider a multi-agent system consisting of three agents $\mathcal{A}_1, \mathcal{A}_2$, and $\mathcal{A}_3$ with respective knowledge bases $\tau_{R_1}(\mathbf{r})$, $\tau_{R_2}(\mathbf{r})$, and $\tau_{R_3}(\mathbf{r})$ depicted in Figure 9.12. Suppose agent $\mathcal{A}_2$ collects evidence and consequently must delete the set of tuples

$$\mathbf{T} \quad = \quad \{ < 0\ 1\ 0\ 1\ 0.050 >,\ < 0\ 1\ 1\ 1\ 0.050 > \}.$$

The set of variables that are also represented in the knowledge bases of at least one other agent is $Y = A_2A_5$. The resulting marginal distribution $\tau_{R_2}(\mathbf{r})'$ obtained by the proposed method for processing evidence is shown in Figure 9.13. It can easily verified that the proposed method for processing evidence maintains pairwise consistency.

As our multi-agent system is defined on a hypertree, by Theorem 9.6 the knowledge base is also globally consistency. Assuming public knowledge bases and maintaining global consistency at all times allows the introduction of *distributed* requests into the distributed probabilistic reasoning framework, as well as ensuring that all agents have consistent belief on all common variables.

9.2.2 Failure and Recovery Control Protocols

In this section, we incorporate recovery protocols to ensure consistency in recovering from transaction and agent *failure* [14]. While it is acknowledged that the actual

A_1	A_2	$A_{p(R_1)}$
0	0	0.1
0	1	0.2
1	0	0.3
1	1	0.4

A_2	A_3	A_4	A_5	$A_{p(R_2)}$
0	0	0	0	0.050
0	0	0	1	0.025
0	0	1	0	0.050
0	0	1	1	0.075
0	1	0	0	0.050
0	1	0	1	0.050
0	1	1	0	0.050
0	1	1	1	0.050
1	0	0	0	0.100
1	0	0	1	0.100
1	0	1	0	0.100
1	0	1	1	0.100
1	1	0	0	0.050
1	1	0	1	0.050
1	1	1	0	0.050
1	1	1	1	0.050

A_5	A_6	$A_{p(R_3)}$
0	0	0.2
0	1	0.3
1	0	0.3
1	1	0.2

Figure 9.12: Marginal distributions $\tau_{R_1}(\mathbf{r})$, $\tau_{R_2}(\mathbf{r})$, and $\tau_{R_3}(\mathbf{r})$ representing the knowledge bases of agents $\mathcal{A}_1, \mathcal{A}_2$, and $\mathcal{A}_3$, respectively.

extension to the relational database recovery technique is not significant in itself, the main contribution of this section is to demonstrate multi-agent uncertainty management as an area of distributed relational database research and implementation.

We use examples from [10] to demonstrate the problems involved in managing agent failure and recovery. (For convenience, we will use the terms agent and site interchangeably.)

There is a simple method to manage replicated attributes if agents never fail. Process Read(X) (or $R(X)$) by reading any copy of $\tau_X(\mathbf{r})$ and process Write(X) (or $W(X)$) by writing all copies of $\tau_X(\mathbf{r})$. Asynchronous control is achieved by *distributed two-phase locking* [27].

In practice, however, agents will fail and subsequently recover. Agent failure can be managed by extending the simple method as follows. Process $R(X)$ by reading any copy of $\tau_X(\mathbf{r})$ at an "up" agent. Process $W(X)$ by updating all copies of $\tau_X(\mathbf{r})$ at

A_1	A_2	$A_{p(R_1)}$
0	0	0.1
0	1	0.2
1	0	0.3
1	1	0.4

A_2	A_3	A_4	A_5	$A_{p(R_2)'}$
0	0	0	0	0.050
0	0	0	1	0.050
0	0	1	0	0.050
0	0	1	1	0.150
0	1	0	0	0.050
0	1	1	0	0.050
1	0	0	0	0.100
1	0	0	1	0.100
1	0	1	0	0.100
1	0	1	1	0.100
1	1	0	0	0.050
1	1	0	1	0.050
1	1	1	0	0.050
1	1	1	1	0.050

A_5	A_6	$A_{p(R_3)}$
0	0	0.2
0	1	0.3
1	0	0.3
1	1	0.2

Figure 9.13: Pairwise consistency is maintained with the new marginal distribution $\tau_{R_2}(\mathbf{r})'$ after agent $\mathcal{A}_2$ has processed evidence.

"up" agents and ignoring copies at "down" agents. This simple method of managing agent failure is not necessarily correct.

Example 9.11 Consider a distributed multi-agent environment representing distributions $\tau_X(\mathbf{r})$ and $\tau_Y(\mathbf{r})$ with copies $\tau_{X_{S_1}}(\mathbf{r})$, $\tau_{X_{S_2}}(\mathbf{r})$, $\tau_{Y_{S_3}}(\mathbf{r})$ and $\tau_{Y_{S_4}}(\mathbf{r})$. Consider two *transactions* [93] T_1 and T_2, where T_1 reads $\tau_X(\mathbf{r})$ then writes $\tau_Y(\mathbf{r})$ and T_2 reads $\tau_Y(\mathbf{r})$ then writes $\tau_X(\mathbf{r})$. The simple method *allows* the following execution:

$$R_1(X_{S_1}), R_2(Y_{S_4}), S_1\sqcap, S_4\sqcap, W_1(Y_{S_3}), W_2(X_{S_2}).$$

where $S_i\sqcap$ denotes the failure of agent S_i.

T_1 and T_2 each set a read-lock on the corresponding copy and then perform their reads asynchronously. Next agents S_1 and S_4 fail. Finally, T_1 and T_2 perform their writes asynchronously. T_1 writes $\tau_Y(\mathbf{r})$ by locking all copies at "up" agents, namely, $\tau_{Y_{S_3}}(\mathbf{r})$. Note that T_1's write-lock on $\tau_{Y_{S_3}}(\mathbf{r})$ does not conflict with T_2's read-lock on $\tau_{Y_{S_4}}(\mathbf{r})$ because $\tau_{Y_{S_3}}(\mathbf{r})$ and $\tau_{Y_{S_4}}(\mathbf{r})$ are different copies at different agents. T_2 writes $\tau_X(\mathbf{r})$ similarly.

The execution in example 9.11 is *not* correct since T_2 does not read the distribution $\tau_Y(\mathbf{r})$ written by T_1, and T_1 does not read the distribution $\tau_X(\mathbf{r})$ written by T_2. One of these two cases must occur in a serial execution of T_1 and T_2 in a single copy environment.

Managing agent recovery is slightly more complex. Consider a distribution $\tau_X(\mathbf{r})$ with copies $\tau_{X_{S_1}}(\mathbf{r})$ and $\tau_{X_{S_2}}(\mathbf{r})$ at agents S_1 and S_2, respectively. Suppose agent S_1 fails. Copy $\tau_{X_{S_1}}(\mathbf{r})$ may become outdated while S_1 is down since copy $\tau_{X_{S_2}}(\mathbf{r})$ may be updated. Copy $\tau_{X_{S_1}}(\mathbf{r})$ must be updated when agent S_1 recovers before users access it. A simple method to accomplish this is to copy the value of $\tau_{X_{S_2}}(\mathbf{r})$ into $\tau_{X_{S_1}}(\mathbf{r})$. This simple method of handling agent recovery is not necessarily correct.

Example 9.12 Consider a distributed multi-agent environment with four agents S_1, S_2, S_3, and S_4. Let $\tau_X(\mathbf{r})$ and $\tau_Y(\mathbf{r})$ be distributions with copies $\tau_{X_{S_1}}$, $\tau_{X_{S_2}}$, $\tau_{X_{S_3}}$, and $\tau_{Y_{S_4}}(\mathbf{r})$. Consider two transactions T_1 and T_2, where T_1 writes $\tau_X(\mathbf{r})$ and then reads $\tau_Y(\mathbf{r})$, and T_2 reads $\tau_X(\mathbf{r})$ and then writes $\tau_Y(\mathbf{r})$. The simple method for handling agent recovery *allows* the following execution:

$$W_1(X_{S_3}), R_{in}(X_{S_1}), W_{in}(X_{S_2}), W_1(X_{S_1}), R_2(X_{S_2}),$$
$$R_1(Y_{S_4}), W_2(Y_{S_4}),$$

where $Read_{in}(X_{S_1})$ and $Write_{in}(X_{S_2})$ are issued by agent S_2 to update copy $\tau_{X_{S_2}}(\mathbf{r})$.

Since agent S_2 has failed when the execution begins, T_1 only updates copies $\tau_{X_{S_1}}(\mathbf{r})$ and $\tau_{X_{S_3}}(\mathbf{r})$. However, S_2 recovers in parallel with T_1 by reading the outdated copy $\tau_{X_{S_1}}(\mathbf{r})$. Since S_2 has recovered by the time T_2 starts, T_2 subsequently reads the outdated copy $\tau_{X_{S_2}}(\mathbf{r})$.

The execution in example 9.12 is *not* correct since it is not equivalent to a serial execution of T_1 and T_2 in a single copy environment. In a serial execution where T_1 is executed first, T_2 reads the distribution $\tau_X(\mathbf{r})$ written by T_1. However, T_2 reads a prior value of $\tau_X(\mathbf{r})$ in the above execution. A similar argument holds if T_2 is executed first in a serial execution.

We now state recovery protocols which only produce *correct* executions. While vast amounts of distributed and currency control techniques have been developed for relational databases, the exposition here draws from Bernstein, Hadzilacos and Goodman [10].

There are two correctness criteria for multi-agent probabilistic environments, namely, replication control and asynchronous control. *Replication control* states that multiple copies of a distribution should behave as a single copy as far as the user can tell. *Asynchronous control* states that the asynchronous execution of transactions must be equivalent to some serial execution of those transactions. Executing transactions serially in a single copy environment is the correctness criteria, called *1-serializability*, for asynchronous access with replicated attributes. Unlike the model in [101] (see Section 2.4) which violates *both* correctness criteria, our goal here is to define a multi-agent probabilistic reasoning system that exhibits 1-serializability.

We begin by defining some mathematical structures pertinent to our discussion. A transaction is a mathematical structure used to model asynchronous read and write operations for one user. Asynchronous execution of transactions are then modelled in a *history*.

An agent *translates* transaction T's read and write operations on distributions into operations on the replicated copies of those distributions. Hence, we define a function h that maps $R(X)$ into $R(X_S)$, where $\tau_{X_S}(\mathbf{r})$ is a copy of $\tau_X(\mathbf{r})$; each $W(X)$ into $W(X_{S_1}), \ldots, W(X_{S_q})$ for some copies $\tau_{X_{S_i}}(\mathbf{r})$, $0 < i \leq q$; each transaction commit into c_i; and each transaction abort into a_i.

A *complete replicated data* (RD) history H over $T = \{T_1, \ldots, T_n\}$ is a partial order with ordering relation $<$ where

(i) $H = h(\cup_{i=1}^{l} T_i)$ for some translation function h;

(ii) For each T_i and all operations P_i, Q_i in T_i, if $P_i <_i Q_i$, then every operation in $h(P_i)$ is related by $<$ to every operation in $h(Q_i)$;

(iii) For every $R_j(X_A)$, there is at least one $W_i(X_A) < R_j(X_A)$;

(iv) All pairs of conflicting operations are related by $<$, where two operations *conflict* if they operate on the same *copy* and at least one of them is a write operation; and

(v) If $W_i(X) <_i R_i(X)$ and $h(R_i(X)) = R_i(X_A)$ then $W_i(X_A) \in h(W_i(X))$.

The nodes in a *serialization graph* (SG) [10] for an RD history correspond to committed transactions in the history, and there is an edge $T_i \rightarrow T_j$ if there are conflicting operations P_i in T_i and Q_j in T_j such that $P_i < Q_j$. Unfortunately, standard SGs for RD histories are too weak for our purposes.

Example 9.13 Consider a multi-agent environment representing distributions $\tau_X(\mathbf{r})$ and $\tau_Y(\mathbf{r})$ with copies $\tau_{X_{S_1}}(\mathbf{r})$, $\tau_{X_{S_2}}(\mathbf{r})$, $\tau_{Y_{S_3}}(\mathbf{r})$, and $\tau_{Y_{S_4}}(\mathbf{r})$. Suppose we have three transactions T_1, T_2 and T_3: T_1 wants to update $\tau_X(\mathbf{r})$ and $\tau_Y(\mathbf{r})$; T_2 wants to read $\tau_X(\mathbf{r})$ then update $\tau_Y(\mathbf{r})$; T_3 wants to read $\tau_Y(\mathbf{r})$ then update $\tau_X(\mathbf{r})$. If the agent uses two-phase locking and ignores failed copies, history H_1 depicted in Figure 9.14 is *allowed* to occur, where $X_{S_i}\sqcap$ denotes the failure of copy $\tau_{X_{S_i}}(\mathbf{r})$. (Here we find it convenient to speak of copies, rather than agents, failing.)

Transactions T_2 and T_3 begin by reading $\tau_{X_{S_1}}(\mathbf{r})$ and $\tau_{Y_{S_4}}(\mathbf{r})$. The copies that they read fail after they complete their reads. The agent translates $W_2(Y)$ into $W_2(Y_{S_3})$ since $\tau_{Y_{S_3}}(\mathbf{r})$ is the only available copy of $\tau_Y(\mathbf{r})$. Similarly, the agent translates $W_3(X)$ into $W_3(X_{S_2})$. The agent has no trouble locking each copy that T_2 and T_3 access because no two operations of these transactions access the same *copy* of any distribution.

The important point is that H_1 is not correct since it is not 1-serializable, namely, H_1 is not equivalent to a serial history over $\{T_1, T_2, T_3\}$ on a single copy environment. A serial execution of T_1, T_2, and T_3 on a single copy environment would have either T_2 reading the value of $\tau_X(\mathbf{r})$ written by T_3, or T_3 reading the value of $\tau_Y(\mathbf{r})$ written by T_2. However, standard SGs are too weak for our purposes since it can be verified that $SG(H_1)$ is still *acyclic*. (Acyclicity in SGs is regarded as a correctness criteria [10].)

Example 9.13 explicitly demonstrates that simply translating $R(X)$ into $R(X_S)$ for some copy $\tau_{X_S}(\mathbf{r})$ and $W(X)$ into $W(X_S)$ for all available copies $\tau_{X_S}(\mathbf{r})$ may lead to an incorrect execution even if failed copies never recover. The problem here is that the logical conflicting access of T_2 and T_3 on $\tau_X(\mathbf{r})$ and $\tau_Y(\mathbf{r})$ was never manifested into conflicting access on *copies* of $\tau_X(\mathbf{r})$ and $\tau_Y(\mathbf{r})$. The task now is to develop a

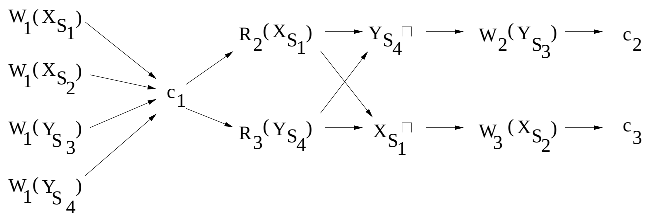

Figure 9.14: History H_1 models an *incorrect* execution even though $SG(H_1)$ is acyclic.

method to ensure that any two transactions that have conflicting access to the same distribution also have conflicting access to some copy of that distribution.

The *available copies algorithm* [10] handles replicated attributes by using a validation protocol, in conjunction with this read-any and write-all-available approach, to ensure correctness. The *validation protocol* consists of two steps:

(i) Missing writes validation: T_i makes sure that all copies it was unable to write are still unavailable, and

(ii) Access validation: T_i makes sure that all copies it read or wrote are still available.

This algorithm also assumes that the agent uses *strict two-phase locking*. The following two examples illustrate how the available copies algorithm ensures 1-serializability.

Let t_i denote the moment when transaction T_i *begins* its access validation step. By the available copies algorithm, t_i must follow all $R_i(X)$ and $W_i(X)$ as well as the missing writes validation step (if present), and must precede c_i.

Example 9.14 Consider the incorrect history H_1 in Figure 9.14 redrawn as history H_1' in Figure 9.15. Since T_2 read $\tau_{X_{S_1}}(\mathbf{r})$, the failure of $\tau_{X_{S_1}}(\mathbf{r})$ must have occurred after T_2 started its access validation. Otherwise, T_2 would have found $\tau_{X_{S_1}}(\mathbf{r})$ to be unavailable and would therefore abort. Therefore, $t_2 < X_{S_1}\sqcap$ and, similarly, $t_3 < Y_{S_4}\sqcap$. Since T_2 wrote $\tau_{Y_{S_3}}(\mathbf{r})$ but not $\tau_{Y_{S_4}}(\mathbf{r})$, it must have carried out the missing writes validation and found that $\tau_{Y_{S_4}}(\mathbf{r})$ is still unavailable. Since $\tau_{Y_{S_4}}(\mathbf{r})$ had already been initialized, it must be that $\tau_{Y_{S_4}}(\mathbf{r})$ failed before the completion of T_i's missing writes validation and thus before the beginning of access validation. Hence $Y_{S_4}\sqcap < t_2$. The precedence $X_{S_1}\sqcap < t_3$ is justified on similar grounds. Given these precedences we have a cycle in H_1': $t_2 < X_{S_1}\sqcap < t_3 < Y_{S_4}\sqcap < t_2$. This is impossible since H_1', being a history, is supposed to be a partial order. This means that H_1' could not have happened.

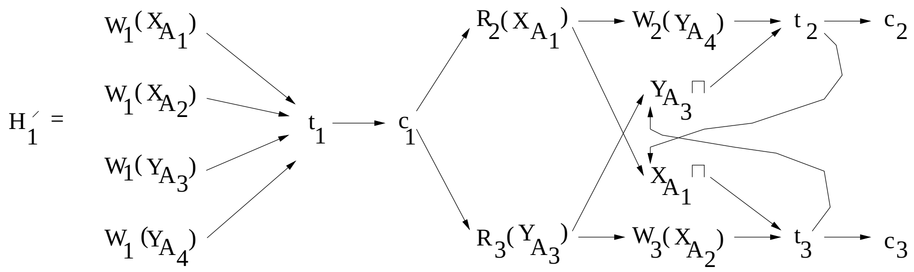

Figure 9.15: The incorrect history H_1 in Figure 9.14 redrawn as history H_1'.

The next example illustrates the significance that the agent uses strict two-phase locking.

Example 9.15 Let T_1, T_2 and T_3 be the same transactions as in H_1 except where there is only one copy $\tau_{Y_{S_3}}(\mathbf{r})$ of $\tau_Y(\mathbf{r})$. Consider history H_2 in Figure 9.16 ignoring for the moment the broken arrow from c_3 to $W_2(Y_{S_3})$. H_2 is not 1-serializable since T_2 does not read the distribution $\tau_X(\mathbf{r})$ written by T_3, and T_3 does not read the distribution $\tau_Y(\mathbf{r})$ written by T_2. One of these two cases must occur in a serial single copy environment. However, the validation protocol in the available copies algorithm does not prevent H_2 from occurring. Instead the assumption that the agent uses two-phase locking prohibits history H_2 from happening. Since T_3 read Y_{S_3} before T_2 updated it, T_3 must have locked $\tau_{Y_{S_3}}(\mathbf{r})$ before T_2 did. By strict two-phase locking, T_2 won't lock $\tau_{Y_{S_3}}(\mathbf{r})$ until after T_3 has committed. That is, we have the precedence $c_3 < W_2(Y_{S_3})$. Given these precedences we have a cycle in H_2: $t_3 < c_3 < W_2(Y_{S_3}) <$

$t_2 < X_{S_1}\sqcap < t_3$. This is impossible since H_2, being a history, is supposed to be a partial order. This means that H_2 could not have happened.

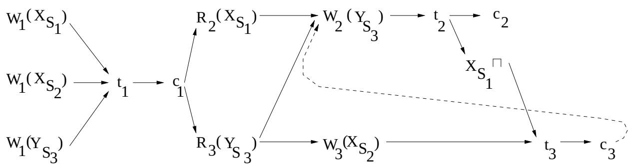

Figure 9.16: History H_1 in Figure 9.14 with only one copy of $\tau_Y(\mathbf{r})$ redrawn as history H_2.

Theorem 9.7 [10] The available copies algorithm only produces 1-serializable histories.

The model proposed in [101] for probabilistic reasoning in a multi-agent environment does not include provisions for transaction or agent failure, nor does it exhibit 1-serializability. In fact, it violates *both* correctness criteria, namely, replication control and asynchronous control. On the other hand, Theorem 9.7 indicates that our generalized relational data model will only produce 1-serializable histories even in light of transaction and agent failure.

9.3 Summary

The dependency structure of a probabilistic network is a graphical representation of the conditional independencies that are known to hold in the problem domain. It is not realistic to expect the domain experts to directly construct the dependency structure of a multi-agent probabilistic network since the problem domain may be significantly larger and perhaps distributed. It is also not entirely clear how the single-agent techniques for learning the dependency structure can be applied, let alone obtaining a reliable sample. Thus, constructing the multi-agent dependency structure amounts to finding a method to combine the known conditional independency information supplied by the individual domain experts. We have shown that the method of simply connecting the individual dependency structures in [102] may be too restrictive in some situations.

In this section, an automated procedure was proposed for directly constructing the dependency structure (an acyclic hypergraph) of a multi-agent Markov network. The individual domain experts can supply any known probabilistic conditional independency information and not necessarily an explicit dependency structure. Our method is capable of detecting all *inconsistent* and *redundant* independencies. The resulting *minimum* cover is used to systematically construct a *unique* dependency structure. The main result of this section is that the constructed acyclic hypergraph is in fact a *perfect-map* of the probabilistic conditional independencies in the minimal cover. This method takes full advantage of the fact that nonembedded conditional independencies have a complete axiomatization.

In this section, techniques were proposed for multi-agent probabilistic reasoning in a distributed environment. Unlike other methods [101], this model is capable of processing input in a *truly* asynchronous fashion. Asynchronous control protocols and a method for processing evidence are developed to ensure global consistency at *all* times. The proposed system then extends beyond an interpretive system since the now well-defined concept of a *distributed* request can be introduced. Techniques are also suggested to reduce data transmission in answering this type of request. Finally, techniques were introduced to ensure consistency in managing transaction and agent (site) failure.

Chapter 10

CONCLUDING REMARKS

One goal in science is to develop more general theories. In this thesis, we have developed a single theory for the individual theories for relational databases [19] and Bayesian networks [66]. A unified model is important since it provides the opportunity for cross-research. However, the relational database model is much more developed than the Bayesian network model. We then focused on adopting the established database techniques for uncertainty management purposes. In particular, we focused on three general areas, namely, the implication problem, more general dependencies, and more general models.

10.1 The Implication Problem

It has been pointed out that [79] the implication problems for embedded multi-valued dependence and probabilistic conditional independence do not coincide. Consequently, it is suggested that the Bayesian network model is different from the relational database model. In this thesis, we have shown that this observation uses two *unsolvable* classes of dependencies. On the other hand, it was demonstrated that the implication problems in the Bayesian network model and the relational database model coincide on *solvable* classes of dependencies. While there are unsolvable classes of dependencies in each model, these classes have no practical value. That is, both models are confined to using solvable classes of dependencies in the design and implementation of knowledge systems. This means that there is no real difference between the Bayesian network model and the relational database model in a practical sense.

10.2 Dependencies

Bayesian networks are defined using only a *single* type of dependency, namely, probabilistic conditional independence. *Contextual* dependencies, such as *context-specific independence* (CSI) [13], *asymmetric independence* (ASI) [34], and *probabilistic causal irrelevance* (PCI) [33], have been recently introduced to capture conditional independencies which hold in parts of a distribution. In this thesis, we introduced a more general contextual dependency. In fact, we introduced a more general dependency than probabilistic conditional independence which applies to the entire distribution.

10.3 Models

Instead of using more general dependencies, an alternative suggestion is to generalize the Bayesian network model itself. The *object-oriented Bayesian network* [49] models a large Bayesian network by allowing one to focus in or out on parts of the network depending on whether they are of interest or not. In this thesis, we proposed a complex-probabilistic-relational database model in which the value of an attribute can be itself a distribution [94]. We introduced a *nest* operator to coarsen the network and an *unnest* to refine the network. It was demonstrated how *inconsistency* may arise in performing inference in complex-value probabilistic networks. More importantly, a new dependency was introduced as a necessary and sufficient condition for ensuring *consistency*.

The notion of *distributed* Bayesian networks, called *multi-agent* Bayesian networks [101], was recently suggested to generalize the standard Bayesian network model. In this discussion, we proposed an automated process to construct the dependency structure of a multi-agent probabilistic network. Furthermore, several protocols were introduced to ensure *consistency* in managing asynchronous access as well as transaction and agent failure. The notion of a distributed query was introduced along with techniques to minimize the data transmitted in answering this type of probabilistic request.

10.4 Future Research

Although the focus of this thesis has been Bayesian networks, the results of the present study may be useful in other related problem domains [76, 96], including belief functions [77], dynamic programming [11], solving sparse linear equations [73], and constraint propagation [25].

The most exciting topic of future research lies in extending the discussion on complex-values. As already mentioned, the notion of complex-value has been extensively studied in the database framework. Thus, it will be very important to adopt these techniques for probabilistic purposes.

The work on multi-agent Bayesian networks focuses on private knowledge bases [101]. On the other hand, the work on distributed databases uses public knowledge bases to the best of our knowledge. The query processing techniques developed for private knowledge bases may be useful in some distributed database applications where the data is considered confidential.

Appendix A

GRAPHICAL STRUCTURES

A.1 Directed Acyclic Graphs

A *directed graph* $D = (R, E)$ consists of a set $R = \{A_1, A_2, \ldots, A_m\}$ of vertices and a set E of ordered pairs of vertices called directed edges. A directed graph is depicted with points representing the vertices and an arrow from point A_i to A_j representing the directed edge (A_i, A_j). Given $R = \{A_1, A_2, \ldots, A_6\}$ and

$$E = \{(A_1, A_2), (A_1, A_3), (A_2, A_4), (A_2, A_5), (A_3, A_4), (A_3, A_5), (A_5, A_6)\} \quad \text{(A.106)}$$

the directed graph $D = (R, E)$ is depicted in Figure 3.1.

A *path* in a directed graph is a sequence $A_1, A_2, \ldots, A_k$ of distinct vertices such that $(A_i, A_{i+1}) \in E$, for $1 \leq i < k$. A *cycle* is a path $A_1, A_2, \ldots, A_k$ for which $A_1 = A_k$. A directed graph is called a *directed acyclic graph* (DAG) if it contains no cycles. Thus, the directed graph in Figure 3.1 is acyclic, namely, a DAG.

A.2 Acyclic Hypergraphs

Let $R = \{A_1, A_2, \ldots, A_m\}$ be a finite set of attributes. A *hypergraph* $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ is a set of subsets $R_i \subseteq R$, namely, $\mathcal{R} \subseteq 2^R$. We say that $\mathcal{R}$ has the *running intersection property* if there is an ordering $R_1, R_2, \ldots, R_n$ of $\mathcal{R}$ such that for $2 \leq i \leq n$ there exists a branching function $b(i) < i$ such that $R_i \cap (R_1 \cup \ldots \cup R_{i-1}) \subseteq R_{b(i)}$. We call $\mathcal{R}$ an *acyclic hypergraph* if and only if $\mathcal{R}$ has the running intersection property [9]. Given an ordering $R_1, R_2, \ldots, R_n$ for an acyclic hypergraph $\mathcal{R}$, and a branching function $b(i)$ for this ordering, the set $\mathcal{J}$ of *J-keys* is defined as:

$$\mathcal{J} = \{R_{b(2)} \cap R_2, R_{b(3)} \cap R_3, \ldots, R_{b(n)} \cap R_n\}. \tag{A.107}$$

This set $\mathcal{J}$ of J-keys is in fact independent of the ordering, i.e., $\mathcal{J}$ is the same for any ordering of a given acyclic hypergraph.

Example A.1 Let $R = \{A_1, A_2, \ldots, A_6\}$ and $\mathcal{R} = \{\ R_1 = \{A_1, A_2, A_3\},\ R_2 = \{A_2, A_3, A_4\},\ R_3 = \{A_2, A_3, A_5\},\ R_4 = \{A_5, A_6\}\ \}$ denote the hypergraph illustrated in Figure 3.2. The ordering R_1, R_2, R_3, R_4 of $\mathcal{R}$ with branching function $b(2) = 1$, $b(3) = 2$ and $b(4) = 3$ satisfies the definition of the running intersection property:

$$\begin{aligned} R_2 \cap R_1 &= \{A_2, A_3\} \subseteq R_1; \quad b(2) = 1, \\ R_3 \cap (R_1 \cup R_2) &= \{A_2, A_3\} \subseteq R_2; \quad b(3) = 2, \\ R_4 \cap (R_1 \cup R_2 \cup R_3) &= \{A_5\} \subseteq R_3; \quad b(4) = 3. \end{aligned}$$

Thus, $\mathcal{R}$ is an acyclic hypergraph. The set $\mathcal{J}$ of J-keys for the acyclic hypergraph $\mathcal{R}$ is

$$\mathcal{J} = \{R_1 \cap R_2, R_1 \cap R_3, R_3 \cap R_4\} = \{\{A_2, A_3\}, \{A_5\}\}. \ \Box$$

A.3 Jointrees

Saying that $\mathcal{R}$ is an acyclic hypergraph is the same as saying that $\mathcal{R}$ has a jointree. Given an acyclic hypergraph $\mathcal{R}$ and a branching function b, a *jointree* $\mathcal{JT}$ for $\mathcal{R}$ is a tree with set $\mathcal{R}$ of nodes, such that

(i) Each edge $(R_i, R_{b(i)})$ is labelled by the set of attributes $R_i \cap R_{b(i)}$, and

(ii) For every pair R_i, R_j ($R_i \neq R_j$) and every A in $R_i \cap R_j$, each edge along the unique path between R_i and R_j includes A.

Example A.2 Consider the acyclic hypergraph $\mathcal{R}$ and branching function b in Example A.1. A jointree $\mathcal{JT}$ for $\mathcal{R}$ is shown in Figure A.1. □

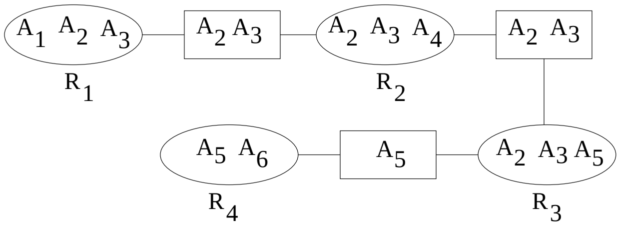

Figure A.1: A graphical representation of the jointree corresponding to the acyclic hypergraph $\mathcal{R} = \{ R_1, R_2, R_3, R_4 \}$ in Figure 3.2, where $b(2) = 1$, $b(3) = 2$ and $b(4) = 3$.

A.4 Moralization and Triangulation Procedures

The reader is referred to [38] for a more complete discussion of moralization and triangulation.

A directed acyclic graph $D = (R, E)$ can be transformed into a *moralized* graph D_M as follows:

(i) For each vertex $A \in R$, add an undirected edge between all nonadjacent parents of A.

(ii) Replace each directed edge (A_i, A_j) with the undirected edge $\{A_i, A_j\}$.

A graph is *chordal* if every cycle with at least four distinct nodes has a *chord*, that is, an edge connecting two nonconsecutive nodes on the cycle. Chordal graphs are sometimes called *triangulated*.

Let $G = (R, E)$ be an undirected graph. If $X \subseteq R$, then the subgraph induced by X in $G = (R, E)$ is $G_X = (X, E_X)$, where E_X contains all edges from E with both vertices in X. An undirected graph is *complete* if there is an edge between every two distinct vertices in X. A *maximal clique* in a graph G is a set X of vertices that induces a maximal complete subgraph.

We are now ready to describe how a DAG D can be transformed into a chordal undirected graph. The first step is to *moralize* the DAG resulting in D_M. A chordal undirected graph is constructed by triangulating D_M. The *maximum cardinality search* (MCS) algorithm can then be used to determine the maximal cliques of the chordal undirected graph.

Example A.3 Let D be the DAG in Figure 3.1. The moralized graph D_M is depicted in Figure A.2. The undirected graph D_M is also chordal. The maximal cliques of D_M are $\{A_1, A_2, A_3\}$, $\{A_2, A_3, A_4\}$, $\{A_2, A_3, A_5\}$, and $\{A_5, A_6\}$. $\Box$

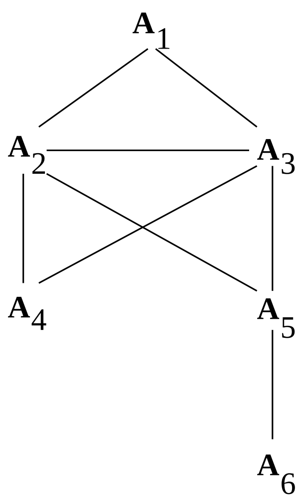

Figure A.2: The moralized graph D_M, where D is the DAG in Figure 3.1.

Appendix B

A SOUND AXIOMATIZATION OF BMVDS

We can use the semi-graphoid axioms in Theorem 8.9 to show that axioms BMVD1-BMVD7 are *sound* for nonembedded probabilistic conditional independencies.

Lemma B.1 Axiom BMVD1 is sound for the implication of nonembedded probabilistic conditional independence.

Proof: Axiom BMVD1 holds for any relation $\mathbf{r}(R)$ with $XY \subseteq R$. $\Box$

Lemma B.2 Axiom BMVD2 is sound for the implication of nonembedded probabilistic conditional independence.

Proof: Let X, Y, Z, and W be pairwise disjoint subsets of R.

1. $X \Rightarrow\Rightarrow Y$ (given),
2. $X \Rightarrow\Rightarrow Y \mid Z$ (CF − BEMVD2 from 1),
3. $X \Rightarrow\Rightarrow Z \mid Y$ (CF − BEMVD1 from 2),
4. $Y \Rightarrow\Rightarrow Z$ (given),
5. $Y \Rightarrow\Rightarrow XW$ (CF − BEMVD1 from 4),
6. $XY \Rightarrow\Rightarrow W$ (CF − BEMVD3 from 5),
7. $XY \Rightarrow\Rightarrow Z$ (CF − BEMVD1 from 6),
8. $X \Rightarrow\Rightarrow Z$ (CF − BEMVD4 from 3 and 7). $\Box$

Lemma B.3 Axiom BMVD3 is sound for the implication of nonembedded probabilistic conditional independence.

Proof: Let X, Y, Z, and W be pairwise disjoint subsets of R.

1. $X \Rightarrow\Rightarrow Y$ (given),
2. $X \Rightarrow\Rightarrow ZW$ (CF − BEMVD1 from 1),
3. $XZ \Rightarrow\Rightarrow W$ (CF − BEMVD3 from 2),
4. $XZ \Rightarrow\Rightarrow Y$ (CF − BEMVD1 from 3). □

Lemma B.4 Axiom BMVD4 is sound for the implication of nonembedded probabilistic conditional independence.

Proof: Let X, Y, Z, and W be pairwise disjoint subsets of R.

1. $X \Rightarrow\Rightarrow Y$ (given),
2. $X \Rightarrow\Rightarrow Y \mid W$ (CF − BEMVD2 from 1),
3. $X \Rightarrow\Rightarrow W \mid Y$ (CF − BEMVD1 from 2),
4. $X \Rightarrow\Rightarrow Z$ (given),
5. $X \Rightarrow\Rightarrow YW$ (CF − BEMVD1 from 4),
6. $XY \Rightarrow\Rightarrow W$ (CF − BEMVD3 from 5),
7. $X \Rightarrow\Rightarrow W$ (CF − BEMVD4 from 3 and 6),
8. $X \Rightarrow\Rightarrow YZ$ (CF − BEMVD1 from 7). □

Lemma B.5 Axiom BMVD5 is sound for the implication of nonembedded probabilistic conditional independence.

Proof: Axiom BMVD5 is axiom CF-BEMVD3 in the semi-graphoid axiomatization. □

Lemma B.6 Axiom BMVD6 is sound for the implication of nonembedded probabilistic conditional independence.

Proof: Since the axioms (BMVD1)-(BMVD5) have been shown sound, it is convenient to follow a similar proof in [8].

1.	$X \Rightarrow\Rightarrow Y$	(given),
2.	$XW \Rightarrow\Rightarrow YW$	(BMVD5 from 1),
3.	$YW \Rightarrow\Rightarrow Z$	(given),
4.	$XW \Rightarrow\Rightarrow Z - (YW)$	(BMVD2 from 2 and 3). $\Box$

Lemma B.7 Axiom BMVD7 is sound for the implication of nonembedded probabilistic conditional independence.

Proof: Axiom BMVD7 is the semi-graphoid axiom CF-BEMVD1. $\Box$

Theorem B.1 The BMVD axioms BMVD1-BMVD7 are sound for the implication of probabilistic conditional independence.

Proof: The claim follows directly from Lemmas (B.1)-(B.7). $\Box$

Appendix C

DERIVATION OF MULTI-AGENT RESULTS

In this appendix, we prove many results used in our discussion on multi-agent Bayesian networks in Chapter 9.

Theorem 9.1 Given a set G of BMVDs on a set of attributes R and $X \subseteq R$, Algorithm 9.1 always terminates and $Dep(X)$ is the dependency basis of X.

Proof: This proof follows the corresponding proof [7] in relational database theory. We will first show that Algorithm 9.1 always terminates, and then demonstrate $Dep(X)$ is in fact the dependency basis of X given G. Notice that at all times the value of $Dep(X)$ is a partition of all the attributes R. Clearly, every iteration of the *repeat until* construct (except the last) refines the value of $Dep(X)$. Each refinement increases the number of elements in $Dep(X)$ by at least one. Therefore, the *repeat until* construct is executed at most $|R|$ times.

We now demonstrate that the value of $Dep(X)$ is independent of the order using the BMVDs of G in executing the *for* construct. Given a fixed order of the BMVDs in G, consider the constructed value $Dep(X)$. Let $Dep(X) = \{Y_1, Y_2, \ldots, Y_k, Y_{k+1}, \ldots, Y_{k+|X|}\}$, where $Y_1 \cup Y_2 \cup \ldots \cup Y_k = R - X$ and $Y_{k+1}, \ldots, Y_{k+|X|}$ are singleton sets whose union is X. We now demonstrate that if $Y_i \in Dep(X)$, then $X \Rightarrow\Rightarrow Y_i$ is in G^+.

By the reflexivity axiom (BMVD1), $X \Rightarrow\Rightarrow A_i$ is in G^+ for each $A_i \in X$. Hence,

$X \Rightarrow\Rightarrow X$ is in G^+, and by the definition of BMVD so is $X \Rightarrow\Rightarrow R - X$. Thus for every set Y in the initial value of $Dep(X)$, $X \Rightarrow\Rightarrow Y$ is in G^+. We show this claim is true after each iteration by induction on the iterations of the *while* construct. Suppose the claim is true after $j (j \geq 0)$ iterations of the *while* construct. Let $W \Rightarrow\Rightarrow Z$ in G be a BMVD used in the $j+1$ iteration and suppose the value of $Dep(X)$ is changed. Let Y be the union of the sets of $Dep(X)$ after the jth iteration that intersect W. Since $Dep(X)$ is a partition of R and $W \subseteq Y$, by (BMVD5) the BMVD $W \Rightarrow\Rightarrow Z$ can be augmented to $Y \Rightarrow\Rightarrow Z$. Applying (BMVD2) to $X \Rightarrow\Rightarrow Y$ and $Y \Rightarrow\Rightarrow Z$ we derive that $X \Rightarrow\Rightarrow Z - Y$ is in G^+. It easily follows by (BMVD4) that $X \Rightarrow\Rightarrow Y$ is in G^+ for every Y in the value of $Dep(X)$ after the $j+1$ iteration.

After Algorithm 9.1 has terminated, $X \Rightarrow\Rightarrow Y$ is in G^+ for every Y in $Dep(X)$. Thus, every element of $Dep(X)$ is a union of the elements of the dependency basis of X. To complete the proof that $Dep(X)$ is in fact equal to the dependency basis of X, we now show that each element in the dependency basis of X is a union of the elements of $Dep(X)$. Equality is then implied since both sets are partitions of R. We shall construct a relation $\mathbf{r}(R)$ with the following two properties:

(i) Every BMVD $W \Rightarrow\Rightarrow Z$ in G is satisfied by $\mathbf{r}(R)$, and

(ii) A BMVD $X \Rightarrow\Rightarrow Y$ is satisfied by $\mathbf{r}(R)$ if and only if Y is a union of elements of $Dep(X)$.

Since every BMVD of G is satisfied by $\mathbf{r}(R)$, so is every BMVD in G^+. Hence for every Y in the dependency basis of X, the BMVD $X \Rightarrow\Rightarrow Y$ is satisfied by $\mathbf{r}(R)$. By property (ii), we can conclude that $Dep(X)$ is in fact equal to the dependency basis of X.

We now construct the relation $\mathbf{r}(R)$ defined by a distribution $p(R)$. We assume that each attribute $A_i \in R$ has the domain $\{0, 1\}$. The relation $\mathbf{r}(R)$ has 2^k rows, one row for each sequence of zeroes and ones of length k. (Recall that k is the number of sets in $Dep(X)$ such that $Y_1 \cup Y_2 \cup \ldots \cup Y_k = R - X$.) In the row corresponding to a sequence $a_1, \ldots, a_k$, each of the attributes Y_i is assigned the value a_i, where $a_i \in \{0, 1\}$ and $i = 1, 2, \ldots, k$. Each attribute of X is assigned the value 1 in all rows of $\mathbf{r}(R)$. The attribute A_p is assigned the value $1/(2^k)$ in all rows of $\mathbf{r}(R)$.

We now make some observations regarding the constructed relation $\mathbf{r}(R)$. The BMVD $\emptyset \Rightarrow\Rightarrow Y_i$ is satisfied by $\mathbf{r}(R)$ $(1 \leq i \leq k)$. By (BMVD5), the BMVD $\emptyset \Rightarrow\Rightarrow Y_i$ can be augmented to $W \Rightarrow\Rightarrow Y_i$ for each set $W \subseteq R$ and $1 \leq i \leq k$.

Our second observation is that if a set W intersects Y_i, then the BMVD $W \Rightarrow\Rightarrow V_i$ is satisfied by $\mathbf{r}(R)$, for each $V_i \subseteq Y_i$. Note that the attributes in $W \cap Y_i$ always have the same value as the rest of the attributes in Y_i for every tuple of $\mathbf{r}(R)$. It follows that the FD $W \rightarrow V_i$ is satisfied by the relation $\pi_R(\mathbf{r})$, for every $V_i \subseteq Y_i$. The FD $W \rightarrow V_i$ implies the MVD $W \rightarrow\rightarrow V_i$ is satisfied by $\pi_R(\mathbf{r})$. The MVD $W \rightarrow\rightarrow V_i$ implies that the BMVD $W \Rightarrow\Rightarrow V_i$ is satisfied by $\mathbf{r}(R)$, since $\mathbf{r}(R)$ is a uniform distribution.

Our last observation is that if W does not intersect Y_i, then for each nonempty proper subset $\hat{Y}_i$ of Y_i, i.e., $\emptyset \subset \hat{Y}_i \subset Y_i$, the BMVD $W \Rightarrow\Rightarrow \hat{Y}_i$ is *not* satisfied by $\mathbf{r}(R)$.

We can now show that $\mathbf{r}(R)$ has property (i) above. Let $W \Rightarrow\Rightarrow Z$ be in G and let Y be the union of the sets from $Dep(X)$ that intersect W. Since Algorithm 9.1 has terminated, we know that $Z - Y$ is either empty or a union of some $Y_i \in Dep(X)$. Therefore, $W \Rightarrow\Rightarrow Z - Y$ is satisfied by $\mathbf{r}(R)$. The fact that $W \Rightarrow\Rightarrow Z \cap Y$ is satisfied by $\mathbf{r}(R)$ follows easily from our observation that if W intersects some Y_i, then for each $\hat{Y}_i \subseteq Y_i$, $W \Rightarrow\Rightarrow \hat{Y}_i$ is satisfied by $\mathbf{r}(R)$. Since $W \Rightarrow\Rightarrow Z - Y$ and $W \Rightarrow\Rightarrow Z \cap Y$ are satisfied by $\mathbf{r}(R)$, by (BMVD3) the BMVD $W \Rightarrow\Rightarrow Z$ is satisfied by $\mathbf{r}(R)$.

To show property (ii), suppose that the BMVD $X \Rightarrow\Rightarrow Y$ is satisfied by $\mathbf{r}(R)$. By construction the BMVD $X \Rightarrow\Rightarrow Y_i$ $(1 \leq i \leq k)$ is satisfied by $\mathbf{r}(R)$. By (BMVD4) the BMVD $X \Rightarrow\Rightarrow Y \cap Y_i$ is satisfied by $\mathbf{r}(R)$ $(1 \leq i \leq k)$. However, since X does not intersect any Y_i, by observation the BMVD $X \Rightarrow\Rightarrow Y \cap Y_i$ is satisfied by $\mathbf{r}(R)$ if and only if $Y \cap Y_i$ is either empty or equal to Y_i. Thus, $Y - X$ is a union of some of the Y_i's $(1 \leq i \leq k)$, and Y is a union of elements of $Dep(X)$. $\Box$

We now prove several results that are used in deriving the main result of our paper stated in Theorem 9.2.

Case 1: $X_i \not\subseteq X_k$.

Proposition 9.1 No $W_{k,j}$ in $Dep(X_k)$, $j \neq l$, partially intersects $W_{i,1}$.

Proof: Suppose there exists a $W_{k,j}$ that intersects partially with $W_{i,1}$, $j \neq l$. That is:

$$W_{k,j} \cap W_{i,1} \neq \emptyset \text{ and } W_{k,j} - W_{i,1} \subset W_{k,j}.$$

Since $X_i \subseteq X_k W_{k,l}$, we have

$$X_k \Rightarrow\Rightarrow X_k W_{k,l} = X_i \overline{W}_{k,l}, \text{ where } \overline{W}_{k,l} = X_k W_{k,l} - X_i.$$

By augmentation on the GMVD $X_i \Rightarrow\Rightarrow W_{k,j} - W_{i,1}$ in equation (9.78), we obtain:

$$X_i \overline{W}_{k,l} \Rightarrow\Rightarrow W_{k,j} - W_{i,1}.$$

By transitivity, it follows:

$$X_k \Rightarrow\Rightarrow (W_{k,j} - W_{i,1}) - X_i \overline{W}_{k,l}.$$

However, the right side can be simplified as $(W_{k,j} - W_{i,1}) - X_i \overline{W}_{k,l} = (W_{k,j} - W_{i,1}) - X_k W_{k,l} = (W_{k,j} - W_{i,1}) - W_{k,l} = (W_{k,j} - W_{i,1})$. Since $(W_{k,j} - W_{i,1}) \subset W_{k,j}$, then $Dep(X_k)$ can be refined. This is of course a contradiction. $\Box$

Proposition 9.2 If $W_{k,j} \cap W_{i,1} = \emptyset$, $j \neq l$, then $W_{k,j}$ must also belong to $Dep(X_i \cap X_k)$.

Proof: We first show that $W_{k,j}$ must belong to $Dep(X_i)$. From equation (9.78), we obtain:

$$X_i \Rightarrow\Rightarrow W_{k,j} - W_{i,1} = W_{k,j}.$$

Suppose $W_{k,j} \neq W_{i,s}$, for any $s \geq 2$. We can show either $Dep(X_i)$ or $Dep(X_k)$ can be refined. Obviously, if $W_{k,j}$ is not equal to a union of some W_i's, then $Dep(X_i)$ can be refined. A contradiction. On the other hand, if $W_{k,j}$ is equal to a union of more than one $W_{i,s}$, then $W_{k,j}$ can be written as:

$$W_{k,j} = W_{i,s_1} W_{i,s_2} \ldots W_{i,s_q}.$$

By definition $W_{k,j} \cap W_{k,l} = \emptyset$, it then follows:

$$W_{i,s_1} - W_{k,l} = W_{i,s_1} \text{ and } W_{i,s_2} - W_{k,l} = W_{i,s_2}.$$

By equation (9.79), we therefore obtain:

$$X_k \Rightarrow\Rightarrow W_{i,s_1} \text{ and } X_k \Rightarrow\Rightarrow W_{i,s_2}.$$

This means that $Dep(X_k)$ can be refined. A contradiction. From the above analysis, we can therefore conclude that $W_{k,j} = W_{i,s}$, for some $s \geq 2$. That is, $W_{k,j}$ is an element in $Dep(X_i)$. Hence,

$$W_{k,j} \in Dep(X_i) \cap Dep(X_k).$$

By the second condition of conflict-free, we have

$$Dep(X_i) \cap Dep(X_k) \subseteq Dep(X_i \cap X_k).$$

It follows $W_{k,j} \in Dep(X_i \cap X_k)$. $\Box$

Proposition 9.3 If $W_{i,s}$ in $Dep'(X_i)$ belongs to $Dep(X_i \cap X_k)$, then $W_{i,s}$ must also belong to $Dep(X_k)$.

Proof: Since $W_{i,s} \in Dep(X_i \cap X_k)$, by augmentation we have $X_i \Rightarrow\Rightarrow W_{i,s}$ and $X_k \Rightarrow\Rightarrow W_{i,s}$. Suppose $W_{i,s}$ is not equal to some $W_{k,j}$ disjoint from $W_{i,1}$. Then there are two possibilities:

(i) $W_{i,s}$ is a union of more than one $W_{k,j}$ disjoint from $W_{i,1}$. That is,

$$W_{i,s} = W_{k,j_1} W_{k,j_2} \ldots W_{k,j_q}.$$

From equation (9.78), we obtain:

$$X_i \Rightarrow\Rightarrow W_{k,j_1}, \text{ and } X_i \Rightarrow\Rightarrow W_{k,j_2}.$$

This means that $W_{i,s}$ in $Dep(X_i)$ can be made "smaller". This is a contradiction.

(ii) $W_{i,s}$ is not a union of some $W_{k,j}$'s.

From the fact that $X_k \Rightarrow\Rightarrow W_{i,s}$, we can immediately conclude that some $W_{k,j}$'s in $Dep(X_k)$ can be made "smaller". This is a contradiction.

Therefore, $W_{i,s}$ must be equal to some $W_{k,j}$ outside $W_{i,1}$. $\square$

Proposition 9.4 $W_{k,l} - X_i W_{i,1} \neq \emptyset$. More specifically,

$$W_{k,l} - X_i W_{i,1} \quad = \quad \cup \ \{ \ W \mid W \in Dep'(X_i) \text{ and } W \notin Dep(X_i \cap X_k) \ \}.$$

Proof: $W_{k,l} - W_{i,1} \neq \emptyset$ since by assumption $W_{k,l}$ contains at least one attribute in X_i and of course $W_{i,1}$ does not. Since $\mathcal{N}$ is a fixed set of attributes, the claim follows immediately. $\square$

Proposition 9.5 $X_i - X_k \neq \emptyset$ and $X_k - X_i \neq \emptyset$. If $Dep(X_i)$ and $Dep(X_k)$ satisfy propositions (1) - (4), then $Dep(X_i)$ and $Dep(X_k)$ cannot be refined.

Proof: Obviously, $X_i \Rightarrow\Rightarrow W_{k,l} - W_{i,1}$ in equation (9.78) leads to no refinement of $Dep(X_i)$. Next we want to show that $X_i \Rightarrow\Rightarrow W_{k,l} - W_{i,1}$ leads to no refinement of $Dep(X_k)$ either. Since $X_k \Rightarrow\Rightarrow W_{k,l}$, we have

$$X_k \Rightarrow\Rightarrow X_k W_{k,l} = X_i \overline{W}_{k,l}, \text{ where } \overline{W}_{k,l} = X_k W_{k,l} - X_i.$$

By augmentation, $X_i \Rightarrow\Rightarrow W_{k,l} - W_{i,1}$ becomes

$$X_i \overline{W}_{k,l} \Rightarrow\Rightarrow W_{k,l} - W_{i,1}.$$

By transitivity, we obtain:

$$X_k \Rightarrow\Rightarrow (W_{k,l} - W_{i,1}) - X_i \overline{W}_{k,l}. \tag{C.108}$$

Since $X_i \overline{W}_{k,l} = X_k W_{k,l}$ and $(W_{k,l} - W_{i,1}) \subseteq X_k W_{k,l}$, the GMVD in equation (C.108) is $X_k \Rightarrow\Rightarrow \emptyset$. Thus, $Dep(X_k)$ cannot be refined with this GMVD.

We now show that the GMVD $X_k \Rightarrow\Rightarrow W_{i,1} - W_{k,l}$ in equation (9.79) leads to no refinement of $Dep(X_k)$. By proposition (1), $W_{i,1} - W_{k,l}$ is a union of elements in $Dep(X_k)$. Thus, $X_k \Rightarrow\Rightarrow W_{i,1} - W_{k,l}$ cannot be used to refine $Dep(X_k)$. We now show that the GMVD $X_k \Rightarrow\Rightarrow W_{i,1} - W_{k,l}$ in equation (9.79) leads to no refinement of $Dep(X_i)$. Since $X_i \Rightarrow\Rightarrow W_{i,1}$, we have

$$X_i \Rightarrow\Rightarrow X_i W_{i,1} = X_k \overline{W}_{i,1}, \text{ where } \overline{W}_{i,1} = X_i W_{i,1} - X_k.$$

By augmentation, $X_k \Rightarrow\Rightarrow W_{i,1} - W_{k,l}$ becomes

$$X_k \overline{W}_{i,1} \Rightarrow\Rightarrow W_{i,1} - W_{k,l}.$$

By transitivity, we obtain:

$$X_i \Rightarrow\Rightarrow (W_{i,1} - W_{k,l}) - X_k \overline{W}_{i,1}. \tag{C.109}$$

Since $X_k \overline{W}_{i,1} = X_i W_{i,1}$ and $(W_{i,1} - W_{k,l}) \subseteq X_i W_{i,1}$, the GMVD in equation (C.109) is $X_i \Rightarrow\Rightarrow \emptyset$. Thus, $Dep(X_i)$ cannot be refined with this GMVD. $\square$

Case 2: $X_i \subseteq X_k$.

Proposition 9.6 No $W_{k,j}$ partially intersects $W_{i,1}$.

Proof: Suppose $W_{k,j} - W_{i,1} \subset W_{k,j}$. From equation (9.78),

$$X_i \Rightarrow\Rightarrow W_{k,j} - W_{i,1} \equiv W'_{k,j} \subset W_{k,j}.$$

Since $X_i \subseteq X_k$, by augmentation, we obtain the GMVD $X_k \Rightarrow\Rightarrow W'_{k,j}$. This means that $Dep(X_k)$ can be refined. This is a contradiction. $\square$

Proposition 9.7 Every $W_{i,s}$ in $Dep'(X_i)$ also belongs to $Dep(X_k)$.

Proof: In this case, $Dep(X_i \cap X_k) = Dep(X_i)$. The claim follows from proposition (9.3). $\square$

Proposition 9.8 If $W_{k,j} \cap W_{i,1} = \emptyset$, then $W_{k,j}$ is also an element in $Dep'(X_i)$.

Proof: In this case, $Dep(X_i \cap X_k) = Dep(X_i)$. The claim follows from proposition (9.2). $\Box$

Proposition 9.9 X_k is a subset of exactly one hyperedge h in $\mathcal{H}^{k-1}$.

Proof: Let h_i and h_j be two distinct hyperedges in $\mathcal{H}^{k-1}$. Suppose X_k is a subset of $h_i \cap h_j$. Since X_k is not a J-key in $\mathcal{H}^{k-1}$, X_k must be contained by some J-key of $\mathcal{H}^{k-1}$. This is a contradiction to the definition of a p-ordering sequence. On the other hand, suppose there are attributes A and B in X_k such that $A \in h_i - h_j$ and $B \in h_j - h_i$. This means X_k is split by some key. This contradicts the initial assumption that the keys are conflict-free. $\Box$

Proposition 9.10 Let h be the hyperedge in $\mathcal{H}^{k-1}$ containing X_k. Then

$$\{h\} = \overline{X_1 W_{1,1}} \cap \ldots \cap \overline{X_{k-1} W_{k-1,1}}.$$

Proof: Obviously, by proposition (9.9), we have

$$\{h\} \subseteq \overline{X_1 W_{1,1}} \cap \ldots \cap \overline{X_{k-1} W_{k-1,1}}.$$

It is also clear that any hyperedge $h' \in \mathcal{H}^{k-1}$, $h' \neq h$, h' is not an element of some $\overline{X_i W_{i,1}}$. Thus,

$$X_k \subseteq \{h\} = \overline{X_1 W_{1,1}} \cap \ldots \cap \overline{X_{k-1} W_{k-1,1}}. \ \Box$$

Bibliography

[1] S. Abiteboul, P. Fischer, and H. Schek. *Nested Relations and Complex Objects in Databases*, volume 361. Springer-Verlag, 1989.

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

[3] A. V. Aho, C. Beeri, and J. D. Ullman. The theory of joins in relational databases. *ACM Transactions on Database Systems*, 4(3):297–314, September 1979.

[4] W.W. Armstrong. Dependency structures of database relationships. In *Proceedings of IFIP 74*, pages 580–583, Amsterdam, 1974. North Holland.

[5] F. Bancilhon, C. Delobel, and P. Kanellakis. *Building an Object-Oriented Database System.* Morgan Kaufman, 1991.

[6] D. Barbará, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, 4(5):487–502, 1992.

[7] C. Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, 5(3):241–259, September 1980.

[8] C. Beeri, R. Fagin, and J.H. Howard. A complete axiomatization for functional and multivalued dependencies in database relations. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 47–61, 1977.

[9] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM*, 30(3):479–513, July 1983.

[10] P.A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, Don Mills, Ontario, 1987.

[11] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming.* Academic Press, 1990.

[12] R. Bouckaert. Properties of bayesian belief network learning algorithms. In *Tenth Conference on Uncertainty in Artificial Intelligence*, pages 102–109. Morgan Kaufmann Publishers, 1994.

[13] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in bayesian networks. In *Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 115–123. Morgan Kaufmann Publishers, 1996.

[14] C.J. Butz and S.K.M. Wong. Recovery protocols in multi-agent probabilistic reasoning systems. In *International Database Engineering and Applications Symposium*, pages 302–310. IEEE Press, 1999.

[15] C.J. Butz, S.K.M. Wong, and Y.Y. Yao. On data and probabilistic dependencies. In *IEEE Canadian Conference on Electrical and Computer Engineering*, pages 1692–1697. IEEE Press, 1999.

[16] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases.* Springer-Verlag, 1990.

[17] Stefano Ceri and Guiseppe Pelagatti. *Distributed Databases: Principles & Systems.* McGraw-Hill, 1984.

[18] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Eighth ACM Symposium on Theory of Computing*, pages 77–90, 1976.

[19] E.F. Codd. A relational model of data for large shared data banks. *Communication of ACM*, 13(6):377–387, June 1970.

[20] E.F. Codd. A database sublanguage founded on the relational calculus. *ACM SIGFIDET*, pages 35–61, 1971.

[21] E.F. Codd. Further normalization of the data base relational model. *Data Base Systems, Rustin(ed), Prentice-Hall publishers*, 1972.

[22] E.F. Codd. Relational completeness of data base sub-languages. *Data Base Systems, Rustin(ed), Prentice-Hall publishers*, 1972.

[23] G.F. Cooper and E.H. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning.*, 9:309–347, 1992.

[24] A.P. Dawid and S.L. Lauritzen. Hyper markov laws in the statistical analysis of decomposable graphical models. *Ann. Stat.*, 21:1272–1317, 1993.

[25] R. Dechter. Decomposing a relation into a tree of binary relations. *Journal of Computer and System Sciences*, 41:2–24, 1990.

[26] D. Dey and S. Sarkar. A probabilistic relational model and algebra. *ACM Transactions on Database Systems*, 21(3):339–369, September 1996.

[27] K.P. Eswaran, J.N. Gray, R.A. Lorie, and I.L. Traiger. The notions of consistency and predicate locks in a database system. *Communication of ACM*, 19(11):624–633, November 1976.

[28] R. Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Transactions on Database Systems*, 2(3):262–278, September 1977.

[29] R. Fagin and M.Y. Vardi. The theory of data dependencies: A survey. *Mathematics of Information Processing: Proceedings of Symposia in Applied Mathematics*, 34:19–71, 1986.

[30] P. Fischer, L.V. Saxton, S. Thomas, and D. Van Gucht. Interactions between dependencies and nested relational structures. *Journal of Computer and System Sciences*, 31(3):343–354, 1985.

[31] P. Fischer and D. Van Gucht. Weak multivalued dependencies. In *Proceedings of the Third ACM SIGACT-SIGMOD Symposium on the Principles of Database Systems*, pages 266–274, 1984.

[32] Z. Galil. An almost linear-time algorithm for computing a dependency basis in a relational database. *Journal of the ACM*, 29(1):96–102, January 1982.

[33] D. Galles and J. Pearl. Axioms of causal relevance. *Artificial Intelligence*, 97:9–43, 1997.

[34] D. Geiger and D. Heckerman. Advances in probabilistic reasoning. In *Seventh Conference on Uncertainty in Artificial Intelligence*, pages 118–126. Morgan Kaufmann Publishers, 1991.

[35] D. Geiger and J. Pearl. Logical and algorithmic properties of conditional independence. Technical Report R-97-II-L, University of California, 1989.

[36] D. Geiger and J. Pearl. Logical and algorithmic properties of conditional independence and graphical models. *The Annals of Statistics*, 21(4):2001–2021, 1993.

[37] D. Geiger, T. Verma, and J. Pearl. Identifying independence in bayesian networks. Technical Report R-116, University of California, 1988.

[38] P. Hajek, T. Havranek, and R. Jirousek. *Uncertain Information Processing in Expert Systems*. CRC Press, 1992.

[39] D. Heckerman and J. Breese. A new look at causal independence. In *Tenth Conference on Uncertainty in Artificial Intelligence*, pages 286–292. Morgan Kaufmann Publishers, 1994.

[40] D. Heckerman, D. Geiger, and D.M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning.*, 20:197–243, 1995.

[41] C. Herrmann. On the undecidability of implications between embedded multi-valued database dependencies. *Information and Computation*, 122(2):221–235, 1995.

[42] E.H. Herskovits and G.F. Cooper. Kutato: An entropy-driven system for construction of probabilistic expert systems from database. In *Sixth Conference on Uncertainty in Artificial Intelligence*, pages 54–62, 1990.

[43] J. Hill. Comment. *Statistical Science*, 8(3):258–261, 1993.

[44] J. Hu and Y. Xiang. Learning belief networks in domains with recursively embedded pseudo independent submodels. In *Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 258–265. Morgan Kaufmann Publishers, 1997.

[45] Y. Huang and Y. Xiang. Learning bayesian networks by learning decomposable markov networks first. In *IEEE Canadian Conference on Electrical and Computer Engineering*, pages 1704–1709. IEEE Press, 1999.

[46] G. Jaeschke and H.J. Schek. Remarks on the algebra on non first normal form relations. In *Proceedings of First ACM SIGACT-SIGMOD Symposium on the Principles of Database Systems*, pages 124–138, 1982.

[47] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, 4:269–282, 1990.

[48] P.C. Kanellakis. *Handbook of Theoretical Computer Science*, chapter Elements of Relational Database Theory, pages 1074–1156. Elsevier, 1991.

[49] D. Koller and A. Pfeffer. Object-oriented bayesian networks. In *Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 302–313. Morgan Kaufmann Publishers, 1997.

[50] W. Lam and F. Bacchus. Learning bayesian networks: An approach based on the mdl principle. *Computational Intelligence*, 10(3):269–293, 1994.

[51] S.L. Lauritzen and D.J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50:157–244, 1988.

[52] T.T. Lee. An algebraic theory of relational databases. *The Bell System Technical Journal*, 62(10):3159–3204, 1983.

[53] T.T. Lee. An information-theoretic analysis of relational databases-part i: Data dependencies and information metric. *IEEE Transactions on Software Engineering*, SE-13(10):1049–1061, 1987.

[54] Y.E. Lien. Hierarchical schemata for relational databases. *ACM Transactions on Database Systems*, 6(1):48–69, 1981.

[55] Y.E. Lien. On the equivalence of database models. *Journal of the ACM*, 29(2):336–362, October 1982.

[56] D. Maier. *The Theory of Relational Databases*. Principles of Computer Science. Computer Science Press, Rockville, Maryland, 1983.

[57] D. Maier, A.O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM Transactions on Database Systems*, 4(4):455–469, 1979.

[58] A. Makinouchi. A consideration on normal form of not-necessarily-normalized relation in the relational data model. In *Third International Conference on Very Large Data Bases*, pages 447–453, 1977.

[59] F. Malvestuto. A unique formal system for binary decompositions of database relations, probability distributions and graphs. *Information Sciences*, 59:21–52, 1992.

[60] F. Malvestuto. A complete axiomatization of full acyclic join dependencies. *Information Processing Letters*, 68(3):133–139, 1998.

[61] A. Mendelzon. On axiomatizing multivalued dependencies in relational databases. *Journal of the ACM*, 26(1):37–44, 1979.

[62] R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems.* Wiley, New York, 1990.

[63] C. Papadimitriou. *The Theory of Database Concurrency Control.* Principles of Computer Science. Computer Science Press, Rockville, Maryland, 1986.

[64] D. Parker and K. Parsaye-Ghomi. Inference involving embedded multivalued dependencies and transitive dependencies. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 52–57, 1980.

[65] A. Paz. Membership algorithm for marginal independencies. Technical Report CSD-880095, University of California, 1988.

[66] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, San Francisco, California, 1988.

[67] J. Pearl, D. Geiger, and T. Verma. Conditional independence and its representations. *Kybernetica*, 25(2):33–44, 1989.

[68] J. Pearl and A. Paz. Graphoids: Graph-based logic for reasoning about relevance relations. Technical Report R-53-L, University of California, 1985.

[69] J. Pearl and T. Verma. The logic of representing dependencies by directed graphs. In *AAAI87 Sixth National Conference on Artificial Intelligence*, volume 1, pages 374–379, 1987.

[70] M. Pittarelli. An algebra for probabilistic databases. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):293–303, 1994.

[71] D. Poole. Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[72] F. Preparata and R. Yeh. *Introduction to Discrete Structures for Computer Science and Engineering.* Addison-Wesley, 1973.

[73] D.J. Rose. Triangulated graphs and the elimination process. *J. Math. Anal. Appl.*, 32:597–609, 1970.

[74] Y. Sagiv. An algorithm for inferring multivalued dependencies with an application to propositional logic. *Journal of the ACM*, 27(2):250–262, April 1980.

[75] Y. Sagiv and F. Walecka. Subset dependencies and a complete result for a subclass of embedded multivalued dependencies. *Journal of the ACM*, 20(1):103–117, 1982.

[76] G. Shafer. An axiomatic study of computation in hypertrees. School of Business Working Papers 232, University of Kansas, 1991.

[77] G. Shafer, P.P. Shenoy, and K. Mellouli. Propagating belief functions in qualitative markov trees. *International Journal of Approximate Reasoning*, 1:349–400, 1987.

[78] M. Studeny. Multiinformation and the problem of characterization of conditional-independence relations. *Problems of Control and Information Theory*, 18(1):3–16, 1989.

[79] M. Studeny. Conditional independence relations have no finite complete characterization. In *Eleventh Prague Conference on Information Theory, Statistical Decision Foundation and Random Processes*, pages 377–396. Kluwer, 1990.

[80] K. Tanaka, Y. Kambayashi, and S. Yajima. Properties of embedded multivalued dependencies in relational databases. *Trans. of IECE of Japan*, E62(8):536–543, 1979.

[81] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass. *Temporal Databases: Theory, Design, and Implemenation.* Database Systems and Applications. Benjamin/Cummings, Don Mills, Ontario, 1993.

[82] S.J. Thomas. *A Non-First-Normal Form Relational Database Model.* PhD thesis, Vanderbilt University, 1983.

[83] D. Van Gucht and P. Fischer. Mvds, weak mvds and nested relational structures. Technical Report CS-84-19, Vanderbilt University, P.O. BOX 1679, Station B, Nashville, Tn., October 1984.

[84] M.Y. Vardi. *Trends in Theoretical Computer Science*, chapter Fundamentals of Dependency Theory, pages 171–224. Computer Science Press, 1987.

[85] T. Verma and J. Pearl. Causal networks: Semantics and expressiveness. In *Fourth Conference on Uncertainty in Artificial Intelligence*, pages 352–359, St. Paul, MN, 1988.

[86] T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227. GE Corporate Research and Development, 1990.

[87] W.X. Wen. From relational databases to belief networks. In *Seventh Conference on Uncertainty in Artificial Intelligence*, pages 406–413. Morgan Kaufmann Publishers, 1991.

[88] S.K.M. Wong. Testing implication of probabilistic dependencies. In *Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 545–553. Morgan Kaufmann Publishers, 1996.

[89] S.K.M. Wong. An extended relational data model for probabilistic reasoning. *Journal of Intelligent Information Systems*, 9:181–202, 1997.

[90] S.K.M. Wong and C.J. Butz. Constructing the dependency structure of a multi-agent probabilistic network. *Submitted to IEEE Transactions on Knowledge and Data Engineering*, 1998.

[91] S.K.M. Wong and C.J. Butz. Equivalent characterization of a class of conditional probabilistic independencies. In *First International Conference on Rough Sets and Current Trends in Computing*, pages 338–345. Springer, 1998.

[92] S.K.M. Wong and C.J. Butz. A method for constructing the dependency structure of a probabilistic network. In *Seventh International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems*, 1998.

[93] S.K.M. Wong and C.J. Butz. Probabilistic reasoning in a distributed multi-agent environment. In *Third International Conference on Multi-Agent Systems*, pages 341–348. IEEE Press, 1998.

[94] S.K.M. Wong and C.J. Butz. Contextual weak independence in bayesian networks. In *Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 670–679, 1999.

[95] S.K.M. Wong, C.J. Butz, and D. Wu. On the implication problem for probabilistic conditional independence. In *submitted for publication*, 1999.

[96] S.K.M. Wong, C.J. Butz, and Y. Xiang. A method for implementing a probabilistic model as a relational database. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 556–564. Morgan Kaufmann Publishers, 1995.

[97] S.K.M. Wong, C.J. Butz, and Y. Xiang. Automated database scheme design using mined data dependencies. *Journal of the American Society for Information Science*, 49(5):455–470, 1998.

[98] S.K.M. Wong and Z.W. Wang. On axiomatization of probabilistic conditional independence. In *Tenth Conference on Uncertainty in Artificial Intelligence*, pages 591–597. Morgan Kaufmann Publishers, 1994.

[99] S.K.M. Wong and Y. Xiang. Construction of a markov network from data for probabilistic inference. In *Third International Workshop on Rough Sets and Soft Computing*, pages 562–569, 1994.

[100] S.K.M. Wong, Y.Y. Yao, and C.J. Butz. *Soft Computing in Information Retrieval: Techniques and Applications*, chapter Granular Information Retrieval. Springer Verlag, 1999.

[101] Y. Xiang. A probabilistic framework for cooperative multi-agent distributed interpretation and optimization of communication. *Artificial Intelligence*, 87:295–342, 1996.

[102] Y. Xiang. Verification of dag structures in cooperative belief network based multi-agent systems. *Networks*, 31:183–191, 1998.

[103] Y. Xiang, D. Poole, and M.P. Beddoes. Multiply sectioned bayesian networks and junction forests for large knowledge based systems. *Computational Intelligence*, 9:171–220, 1993.

[104] R.W. Yeung, T.T. Lee, and Z. Ye. Information-theoretic characterizations of conditional mutual independence and markov random fields. *Submitted to IEEE Transactions on Information Theory*, 1999.