Constructing the Dependency Structure of a Multiagent Probabilistic Network

S.K. Michael Wong, *Member*, *IEEE*, and Cory J. Butz

Abstract—A probabilistic network consists of a dependency structure and corresponding probability tables. The *dependency structure* is a graphical representation of the conditional independencies that are known to hold in the problem domain. In this paper, we propose an automated process for constructing the combined dependency structure of a *multiagent* probabilistic network. 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 multiagent probabilistic network can be constructed directly from this minimal cover. The main result of this paper 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.

Index Terms—Probabilistic networks, dependency structure, probabilistic reasoning, conditional independence, data dependencies, multiagent systems.

1 Introduction

Probabilistic networks [11], [14], [19], [20] have become an established framework for representing and reasoning with uncertain knowledge. A probabilistic network consists of a dependency structure coupled with a corresponding set of probability tables. The *dependency structure* is a graphical representation of the conditional independencies that are known to hold in the problem domain. These conditional independencies are needed to provide an economical representation of a joint probability distribution over the problem domain. Clearly, probabilistic reasoning would not be practical without this independency information. Traditionally, there are two main types of probabilistic networks, namely, Bayesian and Markov. A *Bayesian* network [14], [19], [20] consists of a directed acyclic graph (DAG) and corresponding conditional probability tables. The DAG can represent conditional independencies that hold over *any* subset of variables in the problem domain. The other kind of probabilistic network is called a Markov network. A *Markov* network [11] consists of an acyclic hypergraph and corresponding potentials [11]. Unlike a DAG, which is capable of representing independencies over any subset of variables in the problem domain, an acyclic hypergraph can only represent known conditional independencies that involve *all* the variables in the problem domain. We refer to these conditional independencies as *nonembedded*. In spite of not being capable of representing independencies involving proper subsets of variables, a Markov network can take advantage of the many efficient propagation techniques [13], [15], [24] developed for computing marginal distributions.

Traditionally, probabilistic knowledge is represented and reasoned with by a *single* agent. Manually constructing a Bayesian network has been regarded as a difficult procedure, especially when the conditional independency information regarding the problem domain is not fully understood. Several learning methods have subsequently been developed for constructing the dependency structure of a probabilistic network using independency information mined from observed data [12], [20], [31].

Recently, there is emerging interest in extending the traditional single-agent probabilistic environment into a *multiagent* 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 multiagent 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 multiagent probabilistic network, we need to explicitly specify the dependency structure representing the conditional independency information known to hold in the multiagent 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 authors are with the Department of Computer Science, University of Regina, Regina, Saskatchewan, Canada, S4S 0A2.*
E-mail: {wong, butz}@cs.uregina.ca.

Manuscript received 26 Oct. 1996; revised 27 Oct. 1998; accepted 15 Dec. 1999.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 108121.

1041-4347/01/$10.00 © 2001 IEEE

the dependency structure from observed data. It is not entirely clear, however, how those learning methods [12], [20], [31] developed for the single-agent environment can be applied. For example, it may not be possible to obtain a reliable sample. On the other hand, if the learning methods are applied to collected samples for each individual agent, then independence assumptions must be made rendering the samples independent. Thus, constructing the multiagent dependency structure amounts to finding a method to combine the known conditional independency information supplied by the individual domain experts. One previously proposed method [37] constructs the dependency structure of a multiagent Bayesian network. That method verifies whether the dependency structure formed by *connecting* the individual agent DAGs is *acyclic*. This method is straightforward, but may be too restrictive for constructing the multiagent dependency structure in some situations. (See the discussion on related work in Section 3.)

In this paper, we suggest a more robust algorithm for constructing the dependency structure of a multiagent 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 multiagent probabilistic network can be constructed directly from this minimal cover. In fact, it is shown that the constructed dependency structure is a *perfect-map* [20] 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 [10], [20], [29].

Our formulism here is based on a generalized relational data model [28], [35] we developed for probabilistic reasoning. In fact, our data model can be applied to other applications involving local propagation on an acyclic hypergraph [35], including dynamic programming [5], solving sparse linear equations [21], and constraint propagation [7]. The process of constructing an acyclic hypergraph proposed in this paper can then be applied to these other applications by defining the supplied independency information accordingly.

This paper is organized as follows: The basic notions are defined in Section 2. As the reader may not be familiar with the *generalized relational data model*, we have included a review of this model. In Section 3, we discuss related research. The proposed method for constructing the dependency structure of a multiagent Markov network is described in Section 4. We show in Section 5 how the dependency structure of a Markov network may be further refined using the *mixture* of embedded and nonembedded conditional independencies. The conclusion is presented in Section 6.

2 Basic Notions

We begin by defining some pertinent notions: hypergraphs, relational databases, and our generalized relational data model [28], [29], [35]. A detailed description of this model is given here as it forms the basis of our subsequent discussion.

2.1 Hypergraphs

Let $\mathcal{N}$ be a finite set of variables $\{A_1, A_2, \ldots, A_m\}$. A *hypergraph*, denoted $\mathcal{H}$, is a family of subsets of variables in $\mathcal{N}$, i.e., $\mathcal{H} \subseteq 2^{\mathcal{N}}$. An element in $\mathcal{H}$ is called a *hyperedge*. We call an element $t \in \mathcal{H}$, a *twig*, if there exists another distinct element $b \in \mathcal{H}$, such that

$$t \cap \big(\cup \{h \mid h \in \mathcal{H} \text{ and } h \neq t\}\big) = t \cap b.$$

(By this definition, the hyperedge in a hypergraph consisting of a single hyperedge is not a twig.) This means that the intersection of t and the hypergraph $\cup(\mathcal{H} - \{t\})$ is contained in the hyperedge b. We call any such b a *branch* for the twig t and note that a twig t may have many possible branches. A hypergraph $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ is called an *acyclic* hypergraph (a hypertree) [4], [24] if its elements $h_1, h_2, \ldots, h_i$ can be ordered such that h_i is a twig in the subhypergraph, $\{h_1, h_2, \ldots, h_i\}$, $i = 2, \ldots, n$. We call any ordering satisfying this condition a *hypertree construction ordering* for $\mathcal{H}$. (A hypertree construction ordering can also be represented as a *join tree* [4].) Given a particular hypertree construction ordering, we can choose an integer $b(i)$, for $i = 2, \ldots, n$, such that $1 \leq b(i) \leq i - 1$ and $h_{b(i)}$ is a branch for h_i in $\{h_1, h_2, \ldots, h_i\}$. We call $b(i)$ a *branching function* for this ordering. It is possible that a hypertree construction ordering may have more than one branching function. Given a hypertree construction ordering $h_1, h_2, \ldots, h_n$ for a hypertree $\mathcal{H}$, and a branching function $b(i)$ for this ordering, the set $\mathcal{J}$ of *J-keys* is defined as

$$\mathcal{J} = \{h_{b(2)} \cap h_2, h_{b(3)} \cap h_3, \ldots, h_{b(n)} \cap h_n\}.$$

This set $\mathcal{J}$ is in fact independent of the hypertree construction ordering, i.e., $\mathcal{J}$ is the same for any hypertree construction ordering of a given acyclic hypergraph. In the probabilistic reasoning theory, the set $\mathcal{J}$ of the hypertree $\mathcal{H}$ is referred to as the *separation* set.

Example 1. Consider the case where $\mathcal{N} = \{A_1, A_2, \ldots, A_6\}$. Let

$$\mathcal{H} = \big\{h_1 = \{A_1, A_2, A_3\}, h_2 = \{A_2, A_3, A_4\}, h_3 = \{A_2, A_3, A_5\}, h_4 = \{A_5, A_6\}\big\}$$

denote the hypergraph shown in Fig. 1. Since we can define a hypertree construction ordering h_1, h_2, h_3, h_4, by definition this hypergraph is a hypertree. One possible branching function for this ordering h_1, h_2, h_3, h_4 is $b(2) = 1, b(3) = 1, b(4) = 3$. The set $\mathcal{J}$ of J-keys for the acyclic hypergraph $\mathcal{H}$ is

$$\mathcal{J} = \{h_1 \cap h_2, h_1 \cap h_3, h_3 \cap h_4\} \;=\; \big\{\{A_2, A_3\}, \{A_5\}\big\}.$$

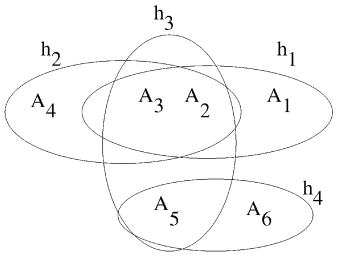

Fig. 1. A graphical representation of the hypergraph $\mathcal{H} = \{h_1, h_2, h_3, h_4\}$.

2.2 Relational Databases

In this section, we review the basic concepts of the standard relational database model with emphasis on data dependencies [1], [18]. These relational concepts are extended in the next section to express similar probabilistic concepts and dependencies.

Let $\mathcal{N} = \{A_1, A_2, \ldots, A_m\}$ be a finite set of attributes (variables). Each attribute A_i is associated with a finite set D_{A_i}, $1 \leq i \leq m$, called the domain of A_i. We define an $\mathcal{N}$-tuple t (or simply a *tuple* if $\mathcal{N}$ is understood) to be a function from $\mathcal{N}$ to $D_{A_1} \cup D_{A_2} \cup \ldots \cup D_{A_m}$ with the restriction that $t[A_i] \in D_{A_i}$ for all $A_i \in \mathcal{N}$, where $t[A_i]$ denotes the value obtained by restricting the mapping to A_i. Thus, a tuple is a mapping that associates a value with each attribute in $\mathcal{N}$, i.e., $t[\mathcal{N}] = < t[A_1], t[A_2], \ldots, t[A_m] >$. If $X \subseteq \mathcal{N}$ and t is a $\mathcal{N}$-tuple, then $t[X]$ denotes the X-tuple obtained by restricting the mapping to X. A *relation over* X (or a *relation* if X is understood) is a finite set of X-tuple. We will sometimes find it convenient to add subscripts in denoting a relation and write the relation r over X as $r[X]$.

The relational operators, select, project, and natural join are defined as follows: Let r be a relation over $\mathcal{N}$ with $A \in \mathcal{N}$ and a an element in the domain of A, i.e., $a \in D_A$. The *select* operator σ is a unary operator on relations. That is, $\sigma_{A=a}(r) = \{t \in r \mid t[A] = a\}$ defines the set of tuples t in r such that $t[A] = a$. The *projection* of $r[\mathcal{N}]$ onto $X \subseteq \mathcal{N}$ is defined as $r[X] = \{t[X] \mid t \in r[\mathcal{N}]\}$. The *natural join* of two relations $r_1[X]$ and $r_2[Y]$ is defined as

$$r_1[X] \bowtie r_2[Y] = \big\{t[XY] \mid t[X] \in r_1[X], t[Y] \in r_2[Y]\big\},$$

where we have written $X \cup Y$ as XY.

Let $r[\mathcal{N}]$ be a relation over a set of attributes $\mathcal{N}$ and $X, Y \subseteq \mathcal{N}$. We say that the *functional dependency* (FD) $X \rightarrow Y$ is satisfied by $r[\mathcal{N}]$ if every two tuples of $r[\mathcal{N}]$ which have the same projection on X also have the same projection on Y. That is, an FD $X \rightarrow Y$ is satisfied by $r[\mathcal{N}]$ if and only if each X-value in $r[\mathcal{N}]$ is associated with precisely one Y-value. The FD $X \rightarrow Y$ is a sufficient but not a necessary condition for $r[\mathcal{N}]$ to be written as $r[\mathcal{N}] = r[XY] \bowtie r[X(\mathcal{N} - XY)]$.

Let $X, Y, Z \subseteq \mathcal{N}$ such that $Y \cap Z \subseteq X$ and $r[\mathcal{N}]$ a relation over $\mathcal{N}$. We say that the *multivalued dependency* (MVD), written $X \twoheadrightarrow Y \mid Z$, is satisfied by $r[\mathcal{N}]$ if and only if $r[XYZ] = r[XY] \bowtie r[XZ]$. The MVD $X \twoheadrightarrow Y \mid Z$ is called *nonembedded* in the special case where $XYZ = \mathcal{N}$. If $XYZ \subset \mathcal{N}$, then the MVD $X \twoheadrightarrow Y \mid Z$ is called *embedded*. Suppose the MVD $X \twoheadrightarrow Y \mid ZW$ is satisfied by the relation $r[\mathcal{N}]$, where X, Y, Z, and W are disjoint subsets of $\mathcal{N}$ (i.e., $r[XYZW] = r[XY] \bowtie r[XZW]$). Obviously, the MVD $X \twoheadrightarrow Y \mid Z$ is satisfied by $r[\mathcal{N}]$, that is, the *smaller* projection $r[XYZ]$ of $r[\mathcal{N}]$ onto XYZ can be written $r[XYZ] = r[XY] \bowtie r[XZ]$. However, the converse is not necessarily true. The fact that the MVD $X \twoheadrightarrow Y \mid Z$ is satisfied by $r[\mathcal{N}]$ (i.e., $r[XYZ] = r[XY] \bowtie r[XZ]$) does *not* necessarily imply that $X \twoheadrightarrow Y \mid ZW$ or $X \twoheadrightarrow YW \mid Z$ would be satisfied by $r[\mathcal{N}]$ (i.e., the *larger* projection $r[XYZW]$ can be expressed as $r[XYZW] = r[XY] \bowtie r[XZW]$ or $r[XYZW] = r[XYW] \bowtie r[XZ]$). For example, it can be verified that the MVD $\{A_1\} \twoheadrightarrow \{A_2\} \mid \{A_3\}$ is satisfied by the relation $r[\mathcal{N}]$ over $\mathcal{N} = \{A_1, A_2, A_3, A_4\}$, as shown in Fig. 2, i.e.,

$$r\big[\{A_1, A_2, A_3\}\big] = r\big[\{A_1, A_2\}\big] \bowtie r\big[\{A_1, A_3\}\big].$$

However, this MVD does not imply that either the MVD $\{A_1\} \twoheadrightarrow \{A_2, A_4\} \mid \{A_3\}$ or $\{A_1\} \twoheadrightarrow \{A_2\} \mid \{A_3, A_4\}$ is also satisfied by the relation $r[\mathcal{N}]$, i.e.,

$$r[\mathcal{N}] \neq r\big[\{A_1, A_2, A_4\}\big] \bowtie r\big[\{A_1, A_3\}\big]$$

and

$$r[\mathcal{N}] \neq r\big[\{A_1, A_2\}\big] \bowtie r\big[\{A_1, A_3, A_4\}\big].$$

The MVD $X \twoheadrightarrow Y \mid (\mathcal{N} - XY)$ is a necessary and sufficient condition for $r[\mathcal{N}]$ to be losslessly decomposed as $r[\mathcal{N}] = r[XY] \bowtie r[X(\mathcal{N} - XY)]$. Thereby, the FD $X \rightarrow Y$ logically implies the MVD $X \twoheadrightarrow Y \mid (\mathcal{N} - XY)$, but the converse is not necessarily true.

Multivalued dependency is a special case of a more general kind of data dependency, called join dependency. We say that the *join dependency* (JD), written $\bowtie \mathcal{H}$, is satisfied by a relation $r[\mathcal{N}]$ if

$$r[\mathcal{N}] = r[h_1] \bowtie r[h_2] \bowtie \ldots \bowtie r[h_n],$$

where $h_i \subseteq \mathcal{N}$ and $\cup_{i=1}^{n} h_i = \mathcal{N}$. The database scheme $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ is in fact a hypergraph. We say that $\bowtie \mathcal{H}$ is an *acyclic* join dependency (AJD) if $\mathcal{H}$ is an acyclic hypergraph. It has been demonstrated that an AJD has many desirable properties and plays an important role in database design [4].

$r[\mathcal{N}] =$

A_1	A_2	A_3	A_4
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	1	1	1

Fig. 2. A *relation* $r[\mathcal{N}]$ over $N = \{A_1, A_2, A_3, A_4\}$.

2.3 The Generalized Relational Data Model

We have shown that probabilistic networks can be viewed as a generalization of the standard relational database model [28], [29], [35]. This model is referred to as the generalized relational data model in which probabilistic concepts can be conveniently expressed in familiar relational terminologies. One of the advantages of this *unified* model is that techniques developed for one particular subdomain can be appropriately modified such that they will become applicable to other subdomains [27], [30], [33], [34], [36], [32].

$\Phi_{\mathcal{N}} =$

A_1	A_2	...	A_m	$f_{\phi_{\mathcal{N}}}$
$t_1[A_1]$	$t_1[A_2]$	...	$t_1[A_m]$	$t_1[f_{\phi_{\mathcal{N}}}]$
$t_2[A_1]$	$t_2[A_2]$	...	$t_2[A_m]$	$t_2[f_{\phi_{\mathcal{N}}}]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$t_s[A_1]$	$t_s[A_2]$	...	$t_s[A_m]$	$t_s[f_{\phi_{\mathcal{N}}}]$

Fig. 3. A joint distribution $\phi_{\mathcal{N}}$ expressed as a *relation* $\Phi_{\mathcal{N}}$, where $t_i[f_{\phi_{\mathcal{N}}}] = \phi_{\mathcal{N}}(t_i[\mathcal{N}])$.

Let $\mathcal{N} = \{A_1, A_2, \ldots, A_m\}$ denote a finite set of variables. A joint probability distribution (jpd) over $\mathcal{N}$, written as $\phi_{\mathcal{N}}$ or $\phi(\{A_1, A_2, \ldots, A_m\})$, is a normalized nonnegative real-valued function. We can express a jpd $\phi_{\mathcal{N}}$ as a generalized relation $\Phi_{\mathcal{N}}$ in our model. The relation $\Phi_{\mathcal{N}}$ is defined by the set of attributes $\{A_1, A_2, \ldots, A_m, f_{\phi_{\mathcal{N}}}\}$. For convenience, we will say $\Phi_{\mathcal{N}}$ is a relation over $\mathcal{N} = \{A_1, A_2, \ldots, A_m\}$. It is understood that the attribute $f_{\phi_{\mathcal{N}}}$ is implicitly used for defining the relation $\Phi_{\mathcal{N}}$. Each row in $\Phi_{\mathcal{N}}$ is defined by a tuple t_i in a standard relation $r[\mathcal{N}]$, as shown in Fig. 3.

Having defined a joint probability distribution as a generalized relation, we can now define probabilistic operations on distributions as generalized relational operations. Computing the projection of a relation and the natural join of two relations in relational database theory corresponds to computing a marginal distribution and the product of two distributions in probabilistic reasoning theory, respectively.

If $\Phi_{\mathcal{N}}$ is a relation and $X \subseteq \mathcal{N}$, then the marginalization of $\Phi_{\mathcal{N}}$ onto X is the *marginal* relation, denoted $\Phi_{\mathcal{N}}^{\downarrow X}$, with attributes $X \cup \{f_{\phi_{\mathcal{N}}^{\downarrow X}}\}$, defined by

$$\Phi_{\mathcal{N}}^{\downarrow X} = \Big\{ t[X \cup \{f_{\phi_{\mathcal{N}}^{\downarrow X}}\}] \mid t[X] \in \Phi_{\mathcal{N}}[X], \text{ and } t[f_{\phi_{\mathcal{N}}^{\downarrow X}}] = \phi_{\mathcal{N}}^{\downarrow X}(t[X]) = \sum_{t' \in \Phi_{\mathcal{N}}, t'[X] = t[X]} \phi_{\mathcal{N}}(t'[\mathcal{N}]) \Big\}.$$

Note that if $\Phi_{\mathcal{N}}$ is a relation representing a distribution over $\mathcal{N}$ and $X \subseteq Y \subseteq \mathcal{N}$, then $((\Phi_{\mathcal{N}})^{\downarrow Y})^{\downarrow X} = \Phi_{\mathcal{N}}^{\downarrow X}$.

Let ϕ_X and ϕ_Y be two distributions over X and Y, respectively. We can express the product $\phi_X \cdot \phi_Y$ as the *product join* $\Phi_X \times \Phi_Y$ of the corresponding relations Φ_X and Φ_Y. That is, $\Phi_X \times \Phi_Y$ is a relation on the set of attributes $XY \cup \{f_{\phi_X \cdot \phi_Y}\}$ defined as follows:

$$\Phi_X \times \Phi_Y = \Big\{ t\big[XY \cup \{f_{\phi_X \cdot \phi_Y}\}\big] \mid t[XY] \in \big(\Phi_X[X] \bowtie \Phi_Y[Y]\big), \text{ and } t[f_{\phi_X \cdot \phi_Y}] = \phi_X\big(t[X]\big) \cdot \phi_Y\big(t[Y]\big) \Big\}.$$

Let $\Phi_{\mathcal{N}}$ be a relation. The inverse of $\Phi_{\mathcal{N}}$ is the *inverse* relation, denoted $(\Phi_{\mathcal{N}})^{-1}$, with attributes $\mathcal{N} \cup \{f_{(\phi_{\mathcal{N}})^{-1}}\}$, defined by

$$(\Phi_{\mathcal{N}})^{-1} = \Big\{ t\big[\mathcal{N} \cup \{f_{(\phi_{\mathcal{N}})^{-1}}\}\big] \mid t[\mathcal{N}] \in \Phi_{\mathcal{N}}[\mathcal{N}] \text{ and } t[f_{(\phi_{\mathcal{N}})^{-1}}] = 1/t[f_{\phi_{\mathcal{N}}}] \text{ if } t[f_{\phi_{\mathcal{N}}}] > 0, \text{ and } t[f_{(\phi_{\mathcal{N}})^{-1}}] = t[f_{\phi_{\mathcal{N}}}] \text{ otherwise} \Big\}.$$

$\Phi_{\mathcal{N}} =$

A_1	A_2	A_3	A_4	$f_{\phi_{\mathcal{N}}}$
0	0	0	0	0.2
0	0	1	1	0.2
0	1	0	0	0.2
0	1	1	0	0.2
1	1	1	1	0.2

Fig. 4. The relation $\Phi_{\mathcal{N}}$ over $N = \{A_1, A_2, A_3, A_4\}$.

Generalized relational data dependencies can now be introduced using the above generalized operators. The notion of probabilistic conditional independence is used to decompose a joint distribution into two marginal distributions, namely, conditional independence corresponds to MVD in relational database theory.

The fundamental notion of generalized multivalued dependency is introduced first. This generalized data dependency is equivalent to probabilistic conditional independence.

Let $X, Y, Z \subseteq \mathcal{N}$ such that $Y \cap Z \subseteq X$ and $\Phi_{\mathcal{N}}$ a relation over $\mathcal{N}$. We say that the *generalized multivalued dependency* (GMVD), written

$$X \Rightarrow\Rightarrow Y \mid Z,$$

is satisfied by the relation $\Phi_{\mathcal{N}}$ if and only if the marginal relation $\Phi_{\mathcal{N}}^{\downarrow XYZ}$ of $\Phi_{\mathcal{N}}$ can be factorized as follows:

$$\Phi_{\mathcal{N}}^{\downarrow XYZ} = \Phi_{\mathcal{N}}^{\downarrow XY} \otimes \Phi_{\mathcal{N}}^{\downarrow XZ} \equiv \Phi_{\mathcal{N}}^{\downarrow XY} \times \Phi_{\mathcal{N}}^{\downarrow XZ} \times (\Phi_{\mathcal{N}}^{\downarrow X})^{-1}. \quad (1)$$

We refer to the binary operation $\otimes$ defined above as the *generalized join*.

Note that it can be shown that

$$X \Rightarrow\Rightarrow Y \mid Z \text{ if and only if } X \Rightarrow\Rightarrow (Y - X) \mid (Z - X).$$

We distinguish the special case when $XYZ = \mathcal{N}$ by calling the GMVD $X \Rightarrow\Rightarrow Y \mid Z$ *nonembedded*. If $XYZ \subset \mathcal{N}$, then we call the GMVD $X \Rightarrow\Rightarrow Y \mid Z$ *embedded*. Suppose the GMVD $X \Rightarrow\Rightarrow Y \mid ZW$ is satisfied by the relation $\Phi_{\mathcal{N}}$, where X, Y, Z, and W are disjoint subsets of $\mathcal{N}$ (i.e., $\Phi_{\mathcal{N}}^{\downarrow XYZW} = \Phi_{\mathcal{N}}^{\downarrow XY} \otimes \Phi_{\mathcal{N}}^{\downarrow XZW}$). Clearly, the GMVD $X \Rightarrow\Rightarrow Y \mid Z$ is satisfied by $\Phi_{\mathcal{N}}$, that is, the *smaller* marginal relation $\Phi_{\mathcal{N}}^{\downarrow XYZ}$ of $\Phi_{\mathcal{N}}$ onto XYZ can be written $\Phi_{\mathcal{N}}^{\downarrow XYZ} = \Phi_{\mathcal{N}}^{\downarrow XY} \otimes \Phi_{\mathcal{N}}^{\downarrow XZ}$. Similar to MVDs in standard relational databases, however, the converse is not necessarily true. The fact that the GMVD $X \Rightarrow\Rightarrow Y \mid Z$ is satisfied by $\Phi_{\mathcal{N}}$ does *not* necessarily imply that $X \Rightarrow\Rightarrow Y \mid ZW$ or $X \Rightarrow\Rightarrow YW \mid Z$ would be satisfied by $\Phi_{\mathcal{N}}$ (i.e., the *larger* marginal relation $\Phi_{\mathcal{N}}^{\downarrow XYZW}$ may not necessarily be expressed as $\Phi_{\mathcal{N}}^{\downarrow XYZW} = \Phi_{\mathcal{N}}^{\downarrow XY} \otimes \Phi_{\mathcal{N}}^{\downarrow XZW}$ or $\Phi_{\mathcal{N}}^{\downarrow XYZW} = \Phi_{\mathcal{N}}^{\downarrow XYW} \otimes \Phi_{\mathcal{N}}^{\downarrow XZ}$). For example, consider the relation $\Phi_{\mathcal{N}}$ over $\mathcal{N} = \{A_1, A_2, A_3, A_4\}$, as shown in Fig. 4. It can be verified that the GMVD $\{A_1\} \Rightarrow\Rightarrow \{A_2\} \mid \{A_3\}$ is satisfied by $\Phi_{\mathcal{N}}$, i.e., $\Phi_{\mathcal{N}}^{\downarrow\{A_1, A_2, A_3\}} = \Phi_{\mathcal{N}}^{\downarrow\{A_1, A_2\}} \otimes \Phi_{\mathcal{N}}^{\downarrow\{A_1, A_3\}}$. However, this GMVD does not imply that either of the GMVDs $\{A_1\} \Rightarrow\Rightarrow \{A_2, A_4\} \mid \{A_3\}$ or $\{A_1\} \Rightarrow\Rightarrow \{A_2\} \mid \{A_3, A_4\}$ is satisfied by

$\Phi_{\mathcal{N}}$, as can also be verified, i.e., $\Phi_{\mathcal{N}} \neq \Phi_{\mathcal{N}}^{\downarrow\{A_1,A_2,A_4\}} \otimes \Phi_{\mathcal{N}}^{\downarrow\{A_1,A_3\}}$ or $\Phi_{\mathcal{N}} \neq \Phi_{\mathcal{N}}^{\downarrow\{A_1,A_2\}} \otimes \Phi_{\mathcal{N}}^{\downarrow\{A_1,A_3,A_4\}}$.

Let $\Phi_{\mathcal{N}}$ be a relation representing a joint probability distribution $\phi_{\mathcal{N}}$ over a set of variables $\mathcal{N}$, and $X, Y, Z \subseteq \mathcal{N}$ be disjoint subsets. We say Y and Z are *probabilistically conditionally independent* given X with respect to $\Phi_{\mathcal{N}}$ if

$$\left(\Phi_{\mathcal{N}}^{\downarrow YXZ} \times \left(\Phi_{\mathcal{N}}^{\downarrow XZ}\right)^{-1}\right)^{\downarrow XY} = \Phi_{\mathcal{N}}^{\downarrow XY} \times \left(\Phi_{\mathcal{N}}^{\downarrow X}\right)^{-1}. \tag{2}$$

Equivalently, conditional independence can be defined as

$$\Phi_{\mathcal{N}}^{\downarrow YXZ} = \Phi_{\mathcal{N}}^{\downarrow XY} \times \Phi_{\mathcal{N}}^{\downarrow XZ} \times \left(\Phi_{\mathcal{N}}^{\downarrow X}\right)^{-1}. \tag{3}$$

It should be obvious that the definition of probabilistic conditional independence given in (3) is equivalent to stating that the relation $\Phi_{\mathcal{N}}^{\downarrow YXZ}$ satisfies the GMVD $X \Rightarrow\Rightarrow Y \mid Z$ in (1), namely,

$$\Phi_{\mathcal{N}}^{\downarrow YXZ} = \Phi_{\mathcal{N}}^{\downarrow XY} \otimes \Phi_{\mathcal{N}}^{\downarrow XZ}.$$

One can also verify that (2) and (3) written in our generalized relational data model notation are equivalent, respectively, to the following more familiar definitions of probabilistic conditional independence:

$$\phi(Y \mid XZ) = \phi(Y \mid X), \tag{4}$$

and

$$\phi(YXZ) = \frac{\phi(YX) \cdot \phi(XZ)}{\phi(X)}. \tag{5}$$

(Pearl [20] writes the GMVD $X \Rightarrow\Rightarrow Y \mid Z$ as $I(Y, X, Z)$.)

Here, we should perhaps make one observation comparing GMVDs in probabilistic uncertainty management and MVDs in standard relational databases. Consider the relation $\Phi_{\mathcal{N}}$ representing a *uniform* joint distribution $\phi_{\mathcal{N}}$ (i.e., $t[f_{\phi_{\mathcal{N}}}] = c$ for all t in $\Phi_{\mathcal{N}}$), as shown in Fig. 4. By projecting $\Phi_{\mathcal{N}}$ onto the set of attributes $\mathcal{N}$, we obtain the standard relation $\Phi_{\mathcal{N}}[\mathcal{N}]$ shown in Fig. 2. It has been shown [28] that the nonembedded GMVD $X \Rightarrow\Rightarrow Y \mid Z$ is satisfied by a uniform $\Phi_{\mathcal{N}}$ if and only if the nonembedded MVD $X \rightarrow\rightarrow Y \mid Z$ is satisfied by $\Phi_{\mathcal{N}}[\mathcal{N}]$. This result will be used in a subsequent proof.

We now define the notions of generalized join dependencies in order to express a joint distribution as the product of *potentials* [11], i.e., probability tables which are not necessarily pairwise consistent. Recall that, in relational database theory, the notion of decomposing a relation into two projections with an MVD was generalized into decomposing a relation into two or more projections with a JD. Probabilistic networks can be expressed as generalized join dependencies in our data model. In particular, it will be shown that a Markov network is equivalent to a generalized relation satisfying a generalized acyclic join dependency.

By the chain rule, a joint probability distribution (jpd) ϕ over $\mathcal{N} = \{A_1, A_2, \ldots, A_m\}$ can always be written as

$$\begin{aligned}\phi = {} & \phi(\{A_1\}) \cdot \phi(\{A_2\}|\{A_1\}) \\ & \cdot \phi(\{A_3\}|\{A_1, A_2\}) \cdot \ldots \cdot \phi(\{A_m\}|\{A_1, A_2, \ldots, A_{m-1}\}).\end{aligned}$$

The above equation is an identity. However, one can use conditional independencies that are *known* to hold in the problem domain to obtain a simpler representation of a jpd. For example, consider a jpd $\phi(\{A_1, A_2, A_3, A_4, A_5, A_6\})$ and the following known conditional independencies:

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

namely, the following GMVDs

$$\begin{aligned}\{A_1\} &\Rightarrow\Rightarrow \{A_2\} \mid \{A_3\}, \\ \{A_2, A_3\} &\Rightarrow\Rightarrow \{A_1\} \mid \{A_4\}, \\ \{A_2, A_3\} &\Rightarrow\Rightarrow \{A_1, A_4\} \mid \{A_5\}, \\ \{A_5\} &\Rightarrow\Rightarrow \{A_1, A_2, A_3, A_4\} \mid \{A_6\}.\end{aligned}$$

Utilizing these conditional independencies, a jpd written using the chain rule can be expressed in a simpler form, namely,

$$\begin{aligned}&\phi(\{A_1, A_2, A_3, A_4, A_5, A_6\}) = \\ &\quad \phi(\{A_1\}) \cdot \phi(\{A_2\}|\{A_1\}) \cdot \phi(\{A_3\}|\{A_1\}) \\ &\quad \cdot \phi(\{A_4\}|\{A_2, A_3\}) \cdot \phi(\{A_5\}|\{A_2, A_3\}) \cdot \phi(\{A_6\}|\{A_5\}).\end{aligned} \tag{6}$$

We can represent the dependency structure of this jpd by a DAG, as shown in Fig. 5. This DAG, together with the conditional probability tables $\phi(\{A_1\})$, $\phi(\{A_2\}|\{A_1\})$, $\phi(\{A_3\}|\{A_1\})$, $\phi(\{A_4\}|\{A_2, A_3\})$, $\phi(\{A_5\}|\{A_2, A_3\})$, and $\phi(\{A_6\}|\{A_5\})$, defines a Bayesian network. Such a network provides an economical representation of a jpd.

A salient feature of the generalized relational data model is that a jpd can be equivalently expressed as a relation. For example, the jpd $\phi(\{A_1, A_2, A_3, A_4, A_5, A_6\})$ in (6), can be expressed as

$$\begin{aligned}\Phi_{\mathcal{N}} = {} & \Phi_{\{A_1\}} \times \Phi_{\{A_1,A_2\}} \times \Phi_{\{A_1,A_3\}} \\ & \times \Phi_{\{A_2,A_3,A_4\}} \times \Phi_{\{A_2,A_3,A_5\}} \times \Phi_{\{A_5,A_6\}},\end{aligned} \tag{7}$$

where $\mathcal{N} = \{A_1, A_2, A_3, A_4, A_5, A_6\}$. The relations $\Phi_{\{A_1\}}$, $\Phi_{\{A_1,A_2\}}$, $\Phi_{\{A_1,A_3\}}$, $\Phi_{\{A_2,A_3,A_4\}}$, $\Phi_{\{A_2,A_3,A_5\}}$, and $\Phi_{\{A_5,A_6\}}$ are respectively defined by the conditional probability tables, $\phi(\{A_1\})$, $\phi(\{A_2\}|\{A_1\})$, $\phi(\{A_3\}|\{A_1\})$, $\phi(\{A_4\}|\{A_2, A_3\})$, $\phi(\{A_5\}|\{A_2, A_3\})$, and $\phi(\{A_6\}|\{A_5\})$.

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 [11], [20] 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 Fig. 5, we obtain the acyclic hypergraph depicted in Fig. 1. Such an acyclic hypergraph represents the dependency structure of a Markov network. To define a Markov network, we need

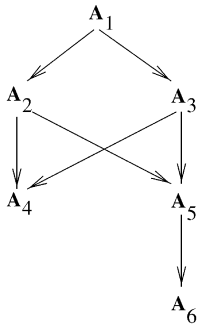

Fig. 5. A DAG, the dependency structure of a Bayesian network, representing the conditional independencies in (6).

to specify its potentials. The jpd defined by (6) can be rewritten as:

$$\Phi_{\mathcal{N}} = \Phi_{\{A_1,A_2,A_3\}} \times \Phi_{\{A_2,A_3,A_4\}} \times \Phi_{\{A_2,A_3,A_5\}} \times \Phi_{\{A_5,A_6\}},$$

where

$$\Phi_{\{A_1,A_2,A_3\}} \equiv \Phi_{\{A_1\}} \times \Phi_{\{A_1,A_2\}} \times \Phi_{\{A_1,A_3\}}, \tag{8}$$

The relations $\Phi_{\{A_1,A_2,A_3\}}$, $\Phi_{\{A_2,A_3,A_4\}}$, $\Phi_{\{A_2,A_3,A_5\}}$, and $\Phi_{\{A_5,A_6\}}$ are called potentials. These potentials can be transformed into marginal relations of $\Phi_{\mathcal{N}}$. In terms of marginals, we can express $\Phi_{\mathcal{N}}$ as

$$\Phi_{\mathcal{N}} = \left(\left(\left(\Phi_{\mathcal{N}}^{\downarrow\{A_1,A_2,A_3\}} \times \Phi_{\mathcal{N}}^{\downarrow\{A_2,A_3,A_4\}} \times \left(\Phi_{\mathcal{N}}^{\downarrow\{A_2,A_3\}}\right)^{-1}\right) \times \Phi_{\mathcal{N}}^{\downarrow\{A_2,A_3,A_5\}} \times \left(\Phi_{\mathcal{N}}^{\downarrow\{A_2,A_3\}}\right)^{-1}\right) \times \Phi_{\mathcal{N}}^{\downarrow\{A_5,A_6\}} \times \left(\Phi_{\mathcal{N}}^{\downarrow\{A_5\}}\right)^{-1}\right). \tag{9}$$

By the definition of the generalized join operator $\otimes$, (9) can be expressed as:

$$\Phi_{\mathcal{N}} = \left(\left(\Phi_{\mathcal{N}}^{\downarrow\{A_1,A_2,A_3\}} \otimes \Phi_{\mathcal{N}}^{\downarrow\{A_2,A_3,A_4\}}\right) \otimes \Phi_{\mathcal{N}}^{\downarrow\{A_2,A_3,A_5\}}\right) \otimes \Phi_{\mathcal{N}}^{\downarrow\{A_5,A_6\}}. \tag{10}$$

In our generalized relational data model, we say that the above $\Phi_{\mathcal{N}}$ satisfies the *generalized acyclic join dependency* (GAJD) [28], written $\otimes\mathcal{H} = \{h_1 = \{A_1, A_2, A_3\}, h_2 = \{A_2, A_3, A_4\}, h_2 = \{A_2, A_3, A_5\}, h_3 = \{A_5, A_6\}\}$. We call the pair $(\Phi_{\mathcal{N}}, \mathcal{H})$ a Markov network, $\Phi_{\mathcal{N}}$ is the relation defined by (10), and $\mathcal{H}$ is the acyclic hypergraph depicted in Fig. 1 representing the dependency structure of the network. In general, we say a GAJD $\otimes\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ is satisfied by a relation $\Phi_{\mathcal{N}}$ if $\Phi_{\mathcal{N}}$ can be written as

$$\Phi_{\mathcal{N}} = \left(\ldots\left(\left(\Phi_{\mathcal{N}}^{\downarrow h_1} \otimes \Phi_{\mathcal{N}}^{\downarrow h_2}\right) \otimes \Phi_{\mathcal{N}}^{\downarrow h_3}\right)\ldots \otimes \Phi_{\mathcal{N}}^{\downarrow h_n}\right), \tag{11}$$

where the sequence $h_1, h_2, \ldots, h_n$ is a hypertree construction ordering for $\mathcal{H}$.

A Bayesian network is more expressive than a Markov network. The structure of a Markov network only reflects

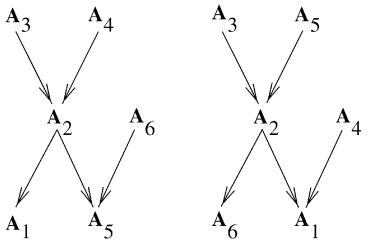

Fig. 6. The respective DAGs of two agents who wish to form a multiagent system.

nonembedded GMVDs. For instance, in the above example, the embedded GMVD $\{A_1\} \Rightarrow\Rightarrow \{A_2\} \mid \{A_3\}$ is not satisfied by the Markov relation in (10). In contrast, this GMVD is satisfied by the Bayesian relation in (7).

3 Related Research

Here, we motivate the need for developing an automated process for constructing the dependency structure of a Markov network. The reason is the inherent difficulty of constructing the dependency structure of a Bayesian network for a multiagent problem domain. 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 [12], [20], [31] developed for the single-agent environment can be applied, let alone obtaining a reliable sample. Thus, constructing the multiagent dependency structure amounts to finding a method to combine the known conditional independency information supplied by the individual domain experts. One previously proposed method [37] constructs the dependency structure of a multiagent Bayesian network. That method verifies whether 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 multiagent reasoning system. According to [37], each agent supplies a respective DAG, as shown in Fig. 6. 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 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 [26]. By construction, each DAG in Fig. 6 has no other equivalent DAG other than itself. Thus, Xiang's method would state that these two agents cannot form a multiagent system. However, consider the acyclic hypergraphs in Fig. 7, obtained by sacrificing the *embedded*

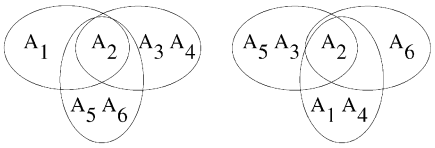

Fig. 7. The agent dependency structures, reflecting only nonembedded probabilistic conditional independencies, obtained from the respective DAGs in Fig. 6.

probabilistic conditional independencies represented in the respective DAGs in Fig. 6.

The first agent would supply the independency information

$$\begin{aligned}\big\{&\{A_2\} \Rightarrow\Rightarrow \{A_1\}|\{A_3, A_4, A_5, A_6\},\\ &\{A_2\} \Rightarrow\Rightarrow \{A_3, A_4\}|\{A_1, A_5, A_6\},\\ &\{A_2\} \Rightarrow\Rightarrow \{A_5, A_6\}|\{A_1, A_3, A_4\}\big\},\end{aligned}$$

while the second agents supplies

$$\begin{aligned}\big\{&\{A_2\} \Rightarrow\Rightarrow \{A_1, A_4\}|\{A_3, A_5, A_6\},\\ &\{A_2\} \Rightarrow\Rightarrow \{A_3, A_5\}|\{A_1, A_4, A_6\},\\ &\{A_2\} \Rightarrow\Rightarrow \{A_6\}|\{A_1, A_3, A_4, A_5\}\big\}.\end{aligned}$$

Note that neither agent has knowledge of the GMVD $\{A_2\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_1, A_4, A_5, A_6\}$. Given the combined set of GMVDs, our approach would produce the multiagent dependency structure $\mathcal{H}$:

$$\mathcal{H} = \big\{\{A_1, A_2\}, \{A_2, A_3\}, \{A_2, A_4\}, \{A_2, A_5\}, \{A_2, A_6\}\big\}.$$

Note that the GMVD $\{A_2\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_1, A_4, A_5, A_6\}$, which was previously unknown to each agent, can be inferred from the combined multiagent dependency structure $\mathcal{H}$. The important point to realize is that the GMVD $\{A_2\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_1, A_4, A_5, A_6\}$ 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 multiagent system, but our method may detect independencies that are logically implied by the combination of all independencies.

One may also suggest directly constructing a multiagent DAG from an arbitrary input set of probabilistic conditional independencies using Pearl's semigraphoid axioms [20]:

*SG*1:	$I(Y,X,Z)$ if and only if $I(Z,X,Y)$;	[symmetry]
*SG*2:	$I(Y,X,ZW)$ then $I(Y,X,Z)$ and $I(Y,X,W)$;	[decomposition]
*SG*3:	$I(Y,X,ZW)$ then $I(Y,XW,Z)$;	[weak union]
*SG*4:	$I(Y,X,Z)$ and $I(Y,XZ,W)$ then $I(Y,X,ZW)$.	[contraction].

(We express the conditional independence of Y and Z given X by the GMVD $X \Rightarrow\Rightarrow Y \mid Z$. Pearl denotes the same independency by $I(Y, X, Z)$.) Unfortunately, it has been shown [25], [32] that Pearl's semigraphoid axioms are not complete for probabilistic conditional independencies, as incorrectly conjectured by Pearl [20]. In fact, probabilistic conditional independencies have no finite complete axiomatization [25], [32]. That is, the semigraphoid axioms may not derive every conditional independency logically implied by an arbitrary set of probabilistic conditional independencies.

It is well-known [20], however, that the semigraphoid axioms are complete for *nonembedded* probabilistic conditional independencies. In fact, Geiger and Pearl [10] developed an alternative complete axiomatization for nonembedded probabilistic conditional independencies using only nonembedded inference axioms. In [29], yet another alternative complete axiomatization for nonembedded probabilistic conditional independencies was shown. This complete axiomatization (stated in Section 4.2) directly corresponds to a complete axiomatization for (nonembedded) multivalued dependency (MVD) in relational databases [2]. Therefore, we prefer to use the complete axiomatization in [29] to emphasize the intrinsic relationship between Bayesian networks and relational databases.

4 Constructing the Dependency Structure of a Markov Network

Representing probability distributions as relations in the generalized data model enables us to adopt various techniques developed in other areas such as relational databases for probabilistic reasoning systems. In particular, we have demonstrated how a Markov network can be represented as a generalized acyclic join dependency in our model. Furthermore, similar to standard relational databases, there exists a complete axiomatization for *nonembedded* generalized multivalued dependencies.

Our main goal here is to develop a process for the construction of the dependency structure (an acyclic hypergraph) of a multiagent 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 GMVDs will be subsequently applied to remove redundant and detect inconsistent independency information. The remaining set of GMVDs 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*.)

4.1 Scheme Design Problems

Given a set G of probabilistic conditional independencies supplied by the individual domain experts over a set $\mathcal{N}$ of attributes, a *construction algorithm* factorizes $\mathcal{N}$ into two sets of attributes on the basis of a known conditional independency in G. That is, given $XYZ = \mathcal{N}$, the set of attributes $\mathcal{N}$ is replaced by $\mathcal{N} \cap XY$ and $\mathcal{N} \cap XZ$, where the GMVD $X \Rightarrow\Rightarrow Y \mid Z$ is in G. Each of these new subsets may be further factorized on the basis of another known GMVD in G. Before formally defining the construction algorithm, let us first highlight two desirable properties such a construction process should satisfy:

1. Every conditional independency provided should contribute in the construction process.
2. 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 1, consider the set G of GMVDs on $\mathcal{N} = \{A_1, A_2, A_3, A_4\}$:

$$G = \big\{\{A_1, A_2\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_4\},\ \{A_3, A_4\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_2\}\big\}.$$

Factorizing $\mathcal{N}$ with the GMVD $\{A_1, A_2\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_4\}$ produces the hypergraph $\mathcal{H} = \{\{A_1, A_2, A_3\}, \{A_1, A_2, A_4\}\}$. The problem now is that the other GMVD $\{A_3, A_4\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_2\}$ in the input set G cannot be applied to refine the dependency structure $\mathcal{H}$. A similar argument holds if the GMVD $\{A_3, A_4\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_2\}$ is applied first. To illustrate a failure of Property 2, consider the set G of GMVDs on $\mathcal{N} = \{A_1, , A_2, \ldots, A_7\}$:

$$\begin{aligned} G = \big\{ \quad & \{A_1\} \Rightarrow\Rightarrow \{A_7\} \mid \{A_2, A_3, A_4, A_5, A_6\}, \\ & \{A_1, A_2\} \Rightarrow\Rightarrow \{A_4\} \mid \{A_3, A_5, A_6, A_7\}, \\ & \{A_1, A_2\} \Rightarrow\Rightarrow \{A_5\} \mid \{A_3, A_4, A_6, A_7\}, \\ & \{A_1, A_2\} \Rightarrow\Rightarrow \{A_3, A_6, A_7\} \mid \{A_4, A_5\}, \\ & \{A_1, A_3\} \Rightarrow\Rightarrow \{A_4\} \mid \{A_2, A_5, A_6, A_7\}, \\ & \{A_1, A_3\} \Rightarrow\Rightarrow \{A_6\} \mid \{A_2, A_4, A_5, A_7\}, \\ & \{A_1, A_3\} \Rightarrow\Rightarrow \{A_2, A_5, A_7\} \mid \{A_4, A_6\} \quad \big\}. \end{aligned}$$

Factorizing $\mathcal{N}$ with the GMVD

$$\{A_1\} \Rightarrow\Rightarrow \{A_7\} \mid \{A_2, A_3, A_4, A_5, A_6\}$$

produces the hypergraph $\mathcal{H}_1$:

$$\mathcal{H}_1 = \big\{h_{11} = \{A_1, A_7\}, h_{12} = \{A_1, A_2, A_3, A_4, A_5, A_6\}\big\}.$$

The set of attributes $h_{12} = \{A_1, A_2, A_3, A_4, A_5, A_6\}$ can be further refined with the GMVD

$$\{A_1, A_2\} \Rightarrow\Rightarrow \{A_4\} \mid \{A_3, A_5, A_6\}.$$

(Since the GMVD $\{A_1, A_2\} \Rightarrow\Rightarrow \{A_4\} \mid \{A_3, A_5, A_6, A_7\}$ holds on $\mathcal{N}$, then the GMVD $\{A_1, A_2\} \Rightarrow\Rightarrow \{A_4\} \mid \{A_3, A_5, A_6\}$ holds on $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ (see Section 2.3.)) Factorizing $h_{12} = \{A_1, A_2, A_3, A_4, A_5, A_6\}$ as such produces the sets $\{A_1, A_2, A_4\}$ and $\{A_1, A_2, A_3, A_5, A_6\}$. The latter can also be factorized using the GMVD $\{A_1, A_2\} \Rightarrow\Rightarrow \{A_5\} \mid \{A_3, A_6\}$ to produce the new dependency structure $\mathcal{H}_2$:

$$\mathcal{H}_2 = \big\{h_{11} = \{A_1, A_7\}, h_{21} = \{A_1, A_2, A_4\}, h_{22} = \{A_1, A_2, A_5\}, h_{23} = \{A_1, A_2, A_3, A_6\}\big\}.$$

The set of attributes $h_{23} = \{A_1, A_2, A_3, A_6\}$ can be factorized with the GMVD $\{A_1, A_3\} \Rightarrow\Rightarrow \{A_6\} \mid \{A_2\}$ constructing the output dependency structure $\mathcal{H}_3$:

$$\begin{aligned} \mathcal{H}_3 = \big\{ \quad & h_{11} = \{A_1, A_7\}, h_{21} = \{A_1, A_2, A_4\}, \\ & h_{22} = \{A_1, A_2, A_5\}, h_{31} = \{A_1, A_2, A_3\}, \\ & h_{32} = \{A_1, A_3, A_6\}\big\}. \end{aligned}$$

On the other hand, factorizing $h_{12} \in \mathcal{H}_1$ with the GMVD $\{A_1, A_3\} \Rightarrow\Rightarrow \{A_4\} \mid \{A_2, A_5, A_6\}$ produces the sets of attributes $\{A_1, A_3, A_4\}$ and $\{A_1, A_2, A_3, A_5, A_6\}$, the latter of which can be be further factorized with the GMVD $\{A_1, A_3\} \Rightarrow\Rightarrow \{A_6\} \mid \{A_2, A_5\}$ to produce the dependency structure $\mathcal{H}_{2'}$:

$$\mathcal{H}_{2'} = \big\{h_{11} = \{A_1, A_7\}, h_{21'} = \{A_1, A_3, A_4\}, h_{22'} = \{A_1, A_3, A_6\}, h_{23'} = \{A_1, A_2, A_3, A_5\}\big\}.$$

Factorizing the set of attributes $h_{23'} = \{A_1, A_2, A_3, A_5\}$ using the GMVD $\{A_1, A_2\} \Rightarrow\Rightarrow \{A_5\} \mid \{A_3\}$ produces the output dependency structure $\mathcal{H}_{3'}$:

$$\begin{aligned} \mathcal{H}_{3'} = \big\{ \quad & h_{11} = \{A_1, A_7\}, h_{21'} = \{A_1, A_3, A_4\}, \\ & h_{22'} = \{A_1, A_3, A_6\}, h_{31'} = \{A_1, A_2, A_3\}, \\ & h_{32'} = \{A_1, A_2, A_5\}\big\}. \end{aligned}$$

Obviously, the output constructed dependency structures $\mathcal{H}_3$ and $\mathcal{H}_{3'}$ are not the same. The problem, in this case, is that the order in which the GMVDs were applied to factorize $\mathcal{N}$ affected the output dependency structure of the multiagent Markov network.

Another undesirable characteristic can be illustrated by the following example: Let $\mathcal{N} = \{A_1, A_2, A_3, A_4, A_5, A_6\}$ and G be the set of GMVDs

$$G = \big\{\{A_1\} \Rightarrow\Rightarrow \{A_5\} \mid \{A_2, A_3, A_4, A_6\},\ \{A_2\} \Rightarrow\Rightarrow \{A_6\} \mid \{A_1, A_3, A_4, A_5\},\ \{A_5, A_6\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_1, A_2, A_4\},\ \{A_5, A_6\} \Rightarrow\Rightarrow \{A_4\} \mid \{A_1, A_2, A_3\}\big\}.$$

Factorizing $\mathcal{N}$ first with the GMVD

$$\{A_1\} \Rightarrow\Rightarrow \{A_5\} \mid \{A_2, A_3, A_4, A_6\},$$

followed by the GMVD $\{A_2\} \Rightarrow\Rightarrow \{A_6\} \mid \{A_1, A_3, A_4, A_5\}$ produces the output dependency structure $\mathcal{H}$:

$$\mathcal{H} = \big\{h_{11} = \{A_1, A_5\}, h_{21} = \{A_1, A_2, A_3, A_4\}, h_{22} = \{A_2, A_6\}\big\}.$$

The remaining GMVDs in the input set G $\{A_5, A_6\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_1, A_2, A_4\}$ and $\{A_5, A_6\} \Rightarrow\Rightarrow \{A_4\} \mid \{A_1, A_2, A_3\}$ cannot be applied to factorize any set of attributes in $\mathcal{H}$. The problem here is that $\mathcal{H}$ does not reflect *all* of the dependency information since $\mathcal{H}$ can be further factorized by the GMVD $\{A_1, A_2\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_4, A_5, A_6\}$ as

$$\mathcal{H} = \big\{h_{11} = \{A_1, A_5\},\ h_{22} = \{A_2, A_6\},\ h_3 = \{A_1, A_2, A_3\},\ h_4 = \{A_1, A_2, A_4\}\big\},$$

since G logically implies the GMVD

$$\{A_1, A_2\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_4, A_5, A_6\}.$$

That is, it may be possible to factorize the output dependency structure of the multiagent Markov network with a GMVD g not explicitly stated but logically implied by G.

The above problems 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.

4.2 Computing a Dependency Basis

We will consider primarily nonembedded GMVDs referring to a fixed (universal) relation $\Phi_{\mathcal{N}}$ defined on a particular set of attributes $\mathcal{N}$. Thus, to simplify the notation, we may subsequently write $X \Rightarrow\Rightarrow Y \mid \mathcal{N} - XY$ as $X \Rightarrow\Rightarrow Y$ if no confusion arises.

A set G of GMVDs *logically implies* the GMVD $X \Rightarrow\Rightarrow Y$, written $G \models X \Rightarrow\Rightarrow Y$, if $X \Rightarrow\Rightarrow Y$ is satisfied by every relation that satisfies all the GMVDs in G. That is, $X \Rightarrow\Rightarrow Y$ is logically implied by G if there is no counterexample relation such that all the GMVDs in G are satisfied but $X \Rightarrow\Rightarrow Y$ is not. An *inference axiom* is a rule that states if a relation Φ satisfies certain GMVDs, then it must satisfy certain other GMVDs. Given a set G of GMVDs and a set of inference axioms, the *closure* of G, written G^+, is the smallest set containing G such that the axioms cannot be applied to the set to yield a GMVD not in the set. More specifically, the set G *derives* a GMVD $X \Rightarrow\Rightarrow Y$, written $G \vdash X \Rightarrow\Rightarrow Y$, if $X \Rightarrow\Rightarrow Y$ is in G^+. A set of axioms is *sound* if, whenever $G \vdash X \Rightarrow\Rightarrow Y$, then $G \models X \Rightarrow\Rightarrow Y$. A set of axioms is *complete* if the converse holds, that is, if $G \models X \Rightarrow\Rightarrow Y$, then $G \vdash X \Rightarrow\Rightarrow Y$. In other words, if G logically implies the GMVD $X \Rightarrow\Rightarrow Y$, then G derives $X \Rightarrow\Rightarrow Y$. A complete set of axioms is *minimal* if no proper subset of axioms is also complete. A complete minimal set of inference axioms for nonembedded GMVDs is listed below [29] (assume symmetry):

GMVD1 : If $Y \subseteq X$, then $X \Rightarrow\Rightarrow Y$; [reflexivity]
GMVD2 : If $X \Rightarrow\Rightarrow Y$ and $Y \Rightarrow\Rightarrow Z$, then $X \Rightarrow\Rightarrow Z - Y$ [transitivity].

(Strictly speaking, (GMVD1) is an identity and not an inference axiom.) From this minimal set, one can derive additional inference axioms that will be used in subsequent discussions. We obtain

GMVD3 : If $Z \subseteq W$ and $X \Rightarrow\Rightarrow Y$, then $WX \Rightarrow\Rightarrow ZY$; [augmentation]
GMVD4 : If $X \Rightarrow\Rightarrow Y$ and $X \Rightarrow\Rightarrow Z$, then $X \Rightarrow\Rightarrow YZ$; [union]
GMVD5 : If $X \Rightarrow\Rightarrow Y$ and $X \Rightarrow\Rightarrow Z$, then $X \Rightarrow\Rightarrow Y \cap Z$, $X \Rightarrow\Rightarrow Y - Z$, and $X \Rightarrow\Rightarrow Z - Y$ [decomposition].

It should be noted that the axioms (GMVD1) and (GMVD2) are different from the semigraphoid axioms. The GMVD axioms are defined only with respect to nonembedded probabilistic conditional independencies. We do not incorporate axioms which mix embedded and nonembedded independencies such as the semigraphoid contraction axiom. (In fact, it has been shown [25], [27] that no finite complete axiomatization exists for *both* embedded and nonembedded conditional independencies.)

Similarly to the relational database theory [2], [8], it is useful to introduce the notion of a dependency basis. A dependency basis is used to summarize a set of GMVDs that all have the same lefthand side. Given $X \subseteq \mathcal{N}$, 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 $\mathcal{N} - X$. 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 GMVD $X \Rightarrow\Rightarrow Y$ which is logically implied by an given set G of GMVDs, 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 GMVD $X \Rightarrow\Rightarrow W$ 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 [8], 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 $X \subseteq \mathcal{N}$. Beeri [3] originally proposed a polynomial time algorithm to compute the dependency basis for MVDs in relational databases. (Faster algorithms have since been proposed [9], [22].) Here, we adopt Beeri's procedure to our problem by using GMVDs instead of MVDs. Our method is outlined in Algorithm 1 by replacing the complete axiomatization for multivalued dependencies with the complete axiomatization for GMVDs. Given a set G of GMVDs on a set of attributes $\mathcal{N}$ and a set $X \subseteq \mathcal{N}$, Algorithm 1 constructs the dependency basis of X as follows:

Algorithm 1

procedure Dependency-Basis(G, X)
$Dep(X) = \{\{A\} \mid A \in \{X\}\} \cup \{\mathcal{N} - X\}$
repeat until $Dep(X)$ is not changed
{
  for each GMVD $W \Rightarrow\Rightarrow Z$ in G
  {
    $Y = \cup \{R \mid R \in Dep(X), \text{ and } R \cap W \neq \emptyset\}$
    $Z' = Z - Y$
    if $Z' \neq \emptyset$ and $Z' \neq \cup_i R_i$, (i.e., if Z' is not a union of some R_i's in $Dep(X)$)
    {
      $Dep(X) = \{R \mid R \in Dep(X) \text{ and } R \cap Z' = \emptyset\} \cup \{R \cap Z', R - Z', Z' - R \mid R \in Dep(X) \text{ and } R \cap Z' \neq \emptyset\}$
    }
  }
}
return $(Dep(X))$
end Dependency − Basis

Let us use an example to demonstrate how Algorithm 1 works.

Example 2. Let $\mathcal{N} = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}\}$. Let the input set of nonembedded GMVDs be

$$G = \big\{\{A_1, A_2\} \Rightarrow\Rightarrow \{A_4, A_5, A_6, A_7\},\ \{A_3, A_7, A_{10}\} \Rightarrow\Rightarrow \{A_1, A_4, A_8, A_9\}\big\}$$

on $\mathcal{N}$. Suppose we want to compute the dependency basis $Dep(X)$ for the subset $X = \{A_1, A_3, A_7, A_{10}\} \subseteq \mathcal{N}$. According to the algorithm, we obtain the *initial* dependency basis:

$$Dep(X) = \big\{\{A_1\}, \{A_3\}, \{A_7\}, \{A_{10}\}, \{A_2, A_4, A_5, A_6, A_8, A_9\}\big\}.$$

First, we check if the GMVD

$$\{A_1, A_2\} = W \Rightarrow\Rightarrow Z = \{A_4, A_5, A_6, A_7\}$$

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 (GMVD4), we can immediately conclude

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Y = \{A_1, A_2, A_4, A_5, A_6, A_8, A_9\}. \tag{12}$$

On the other hand, by (GMVD3), $W \Rightarrow\Rightarrow Z$ implies that:

$$W \cup \{A_4, A_5, A_6, A_8, A_9\} \Rightarrow\Rightarrow Z,$$

namely,

$$\{A_1, A_2, A_4, A_5, A_6, A_8, A_9\} \Rightarrow\Rightarrow \{A_4, A_5, A_6, A_7\}. \tag{13}$$

By applying (GMVD2) to (12) and (13), we obtain:

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Z' = Z - Y = \{A_4, A_5, A_6, A_7\} - \{A_1, A_2, A_4, A_5, A_6, A_8, A_9\} = \{A_7\}.$$

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 GMVD

$$\{A_3, A_7, A_{10}\} = W \Rightarrow\Rightarrow Z = \{A_1, A_4, A_8, A_9\}$$

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 (GMVD4), we have:

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Y = \{A_3, A_7, A_{10}\}.$$

By applying (GMVD2), this GMVD, together with $W \Rightarrow\Rightarrow Z$, implies that:

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Z' = Z - Y = \{A_1, A_4, A_8, A_9\} - \{A_3, A_7, A_{10}\} = \{A_1, A_4, A_8, A_9\}.$$

Now, (GMVD5) is used to *refine* the above $Dep(X)$. From the GMVD

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow R = \{A_2, A_4, A_5, A_6, A_8, A_9\},$$

already in $Dep(X)$ and the above GMVD

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Z' = \{A_1, A_4, A_8, A_9\},$$

it follows:

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow R \cap Z' = \{A_4, A_8, A_9\},$$
$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow R - Z' = \{A_2, A_5, A_6\},$$
$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Z' - R = \{A_1\}.$$

Thus, the new $Dep(X)$ becomes:

$$Dep(X) = \big\{\{A_3\}, \{A_7\}, \{A_{10}\}\big\} \cup \big\{\{A_1\}, \{A_2, A_5, A_6\}, \{A_4, A_8, A_9\}\big\} = \big\{\{A_1\}, \{A_3\}, \{A_7\}, \{A_{10}\}, \{A_4, A_8, A_9\}, \{A_2, A_5, A_6\}\big\}.$$

As the dependency basis has been changed, a second iteration of the *repeat until* construct is performed. The GMVD

$$\{A_1, A_2\} = W \Rightarrow\Rightarrow Z = \{A_4, A_5, A_6, A_7\},$$

and $Dep(X)$ imply that:

$$Y = \{A_1\} \cup \{A_2, A_5, A_6\} = \{A_1, A_2, A_5, A_6\}.$$

By (GMVD2) and (GMVD3),

$$\{A_1, A_3, A_7, A_{10}\} = X \Rightarrow\Rightarrow Z' = Z - Y = \{A_4, A_7\}.$$

Hence, $Dep(X)$ is changed to:

$$Dep(X) = \big\{\{A_1\}, \{A_3\}, \{A_7\}, \{A_{10}\}, \{A_4\}, \{A_8, A_9\}, \{A_2, A_5, A_6\}\big\}.$$

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 1. *Given a set G of GMVDs on a set of attributes $\mathcal{N}$ and $X \subseteq \mathcal{N}$, Algorithm 1 always terminates and $Dep(X)$ is the dependency basis of X.*

Proof. This proof follows the corresponding proof [3] in relational database theory. We will first show that Algorithm 1 always terminates and then demonstrate that $Dep(X)$ 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 $\mathcal{N}$. 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 $|\mathcal{N}|$ times.

We now demonstrate that the value of $Dep(X)$ is independent of the order using the GMVDs of G in

executing the *for* construct. Given a fixed order of the GMVDs 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 $Y_1 \cup Y_2 \cup \ldots \cup Y_k = \mathcal{N} - X$ 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 (GMVD1), $X \Rightarrow\Rightarrow A_i$ is in G^+ for each $A_i \in X$. Hence, $X \Rightarrow\Rightarrow X$ is in G^+ and, by the definition of GMVD so is $X \Rightarrow\Rightarrow \mathcal{N} - X$. 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 GMVD 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 $\mathcal{N}$ and $W \subseteq Y$, by (GMVD3) the GMVD $W \Rightarrow\Rightarrow Z$ can be augmented to $Y \Rightarrow\Rightarrow Z$. Applying (GMVD2) 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 (GMVD5) that $X \Rightarrow\Rightarrow Y$ is in G^+ for every Y in the value of $Dep(X)$ after the $j+1$ iteration.

After Algorithm 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 $\mathcal{N}$. We shall construct a relation $\Phi_{\mathcal{N}}$ with the following two properties:

1. Every GMVD $W \Rightarrow\Rightarrow Z$ in G is satisfied by $\Phi_{\mathcal{N}}$.
2. A GMVD $X \Rightarrow\Rightarrow Y$ is satisfied by $\Phi_{\mathcal{N}}$ if and only if Y is a union of elements of $Dep(X)$.

Since every GMVD of G is satisfied by $\Phi_{\mathcal{N}}$, so is every GMVD in G^+. Hence, for every Y in the dependency basis of X, the GMVD $X \Rightarrow\Rightarrow Y$ is satisfied by $\Phi_{\mathcal{N}}$. By Property 2, we can conclude that $Dep(X)$ is in fact equal to the dependency basis of X.

We now construct the relation $\Phi_{\mathcal{N}}$ defined by a distribution $\phi_{\mathcal{N}}$. We assume that each attribute $A_i \in \mathcal{N}$ has the domain $\{0, 1\}$. The relation $\Phi_{\mathcal{N}}$ has 2^k rows, one row for each sequence of zeros and ones of length k. (Recall that k is the number of sets in $Dep(X)$ such that $Y_1 \cup Y_2 \cup \ldots \cup Y_k = \mathcal{N} - X$.) 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 $\Phi_{\mathcal{N}}$. The attribute $f_{\phi_{\mathcal{N}}}$ is assigned the value $1/(2^k)$ in all rows of $\Phi_{\mathcal{N}}$.

We now make some observations regarding the constructed relation $\Phi_{\mathcal{N}}$. The GMVD $\emptyset \Rightarrow\Rightarrow Y_i$ is satisfied by $\Phi_{\mathcal{N}}$ $(1 \leq i \leq k)$. By (GMVD3), the GMVD $\emptyset \Rightarrow\Rightarrow Y_i$ can be augmented to $W \Rightarrow\Rightarrow Y_i$ for each set $W \subseteq \mathcal{N}$ and $1 \leq i \leq k$.

Our second observation is that if a set W intersects Y_i, then the GMVD $W \Rightarrow\Rightarrow V_i$ is satisfied by $\Phi_{\mathcal{N}}$ 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 $\Phi_{\mathcal{N}}$. It follows that the FD $W \rightarrow V_i$ is satisfied by the relation $\Phi_{\mathcal{N}}[\mathcal{N}]$, for every $V_i \subseteq Y_i$. The FD $W \rightarrow V_i$ implies the MVD $W \rightarrow\rightarrow V_i$ is satisfied by $\Phi_{\mathcal{N}}[\mathcal{N}]$ (see Section 2.2). The MVD $W \rightarrow\rightarrow V_i$ implies that the GMVD $W \Rightarrow\Rightarrow V_i$ is satisfied by $\Phi_{\mathcal{N}}$ since $\Phi_{\mathcal{N}}$ is a uniform distribution (see Section 2.3).

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 GMVD $W \Rightarrow\Rightarrow \hat{Y}_i$ is *not* satisfied by $\Phi_{\mathcal{N}}$.

We can now show that $\Phi_{\mathcal{N}}$ has Property 1 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 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 $\Phi_{\mathcal{N}}$. The fact that $W \Rightarrow\Rightarrow Z \cap Y$ is satisfied by $\Phi_{\mathcal{N}}$ 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 $\Phi_{\mathcal{N}}$. Since $W \Rightarrow\Rightarrow Z - Y$ and $W \Rightarrow\Rightarrow Z \cap Y$ are satisfied by $\Phi_{\mathcal{N}}$, by (GMVD4) the GMVD $W \Rightarrow\Rightarrow Z$ is satisfied by $\Phi_{\mathcal{N}}$.

To show Property 2, suppose that the GMVD $X \Rightarrow\Rightarrow Y$ is satisfied by $\Phi_{\mathcal{N}}$. By construction, the GMVD $X \Rightarrow\Rightarrow Y_i$ $(1 \leq i \leq k)$ is satisfied by $\Phi_{\mathcal{N}}$. By (GMVD5), the GMVD $X \Rightarrow\Rightarrow Y \cap Y_i$ is satisfied by $\Phi_{\mathcal{N}}$ $(1 \leq i \leq k)$. However, since X does not intersect any Y_i, by observation, the GMVD $X \Rightarrow\Rightarrow Y \cap Y_i$ is satisfied by $\Phi_{\mathcal{N}}$ 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_is $(1 \leq i \leq k)$ and Y is a union of elements of $Dep(X)$. □

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 GMVDs containing *redundant* ones. That is, those GMVDs that can be derived from the rest of the GMVDs in G using the inference axioms (GMVD1) and (GMVD2). We say that a set G_1 of GMVDs 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 GMVDs supplied by the individual agents. Take any GMVD g in G, say, $X \Rightarrow\Rightarrow Y$, and compute $Dep(X)$ from the set $G - \{g\}$ of GMVDs. 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 GMVD in G. If we adopt Sagiv's faster algorithm [22] for computing the dependency basis using multivalued dependencies instead of Beeri's [3], a minimum cover of a given set G of GMVDs can be computed in

time $O(k^2\,|\mathcal{N}^2|)$, where k is the number of GMVDs originally in G [16].

Consider a computed minimum cover G of the collective set of GMVDs supplied by the agents. The left sides of the GMVDs 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 GMVDs 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 GMVDs $X \Rightarrow\Rightarrow W_i$ $(1 \le i \le m)$. 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 GMVDs over $\mathcal{N} = \{A_1, A_2, A_3, A_4\}$:

$$G = \big\{\{A_1\} \Rightarrow\Rightarrow \{A_2\} \mid \{A_3, A_4\},\ \{A_1, A_2\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_4\}\big\}, \tag{14}$$

the full minimum cover G_1 of G is computed by repeatedly applying Algorithm 1 with G and each key in X as input,

$$\begin{aligned} G_1 = \Big\{\ & Dep(\{A_1\}) = \big\{\ \{A_2\}, \{A_3\}, \{A_4\}\big\},\ Dep\big(\{A_1, A_2\}\big) \\ & = \big\{\ \{A_3\}, \{A_4\}\big\}\Big\} = \big\{\{A_1\} \Rightarrow\Rightarrow \{A_2\} \mid \{A_3\} \mid \{A_4\}, \\ & \quad \{A_1, A_2\} \Rightarrow\Rightarrow \{A_3\} \mid \{A_4\}\big\}. \end{aligned} \tag{15}$$

4.3 Conflict-Free Dependencies

A full minimum cover may contain conflicting GMVDs. A *conflict-free* full minimum cover is derived by removing the conflicting GMVDs from the full minimum cover. This conflict-free full minimum cover is needed to construct the multiagent dependency structure.

The notion of conflict-free multivalued dependencies was originally introduced by Lien [17] in the study of the relationship between various database models. We extend this notion to GMVDs in our generalized relational data model. We say that a GMVD $X \Rightarrow\Rightarrow Y$ *splits* two attributes A_i and A_j if one of them is in Y and the other is in $\mathcal{N} - XY$, where $\mathcal{N}$ is the set of all attributes. A set G of GMVDs splits two attributes A_i and A_j if some GMVD in G splits them. We then say that a GMVD splits a set W if it splits two attributes in W and that a set G of GMVDs splits a set W if some GMVD in G splits two attributes in W. A set G of GMVDs is *conflict-free* if

1. G does not split its keys and
2. $Dep(X) \cap Dep(Y) \subseteq Dep(X \cap Y)$.

For example, the set of GMVDs in the first example of Section 4.1 violates Condition 1. It can be verified that the GMVDs in the second example of Section 4.1 violate Condition 2.

If conflicting GMVDs 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 GMVDs supplied by the individual domain experts.

As in relational databases [4], conflict-free sets of generalized multivalued dependencies have several nice properties: 1) They allow a unique Markov network dependency structure and 2) all generalized multivalued dependencies participate in the decomposition process, that is, the phenomenon where decomposing according to one generalized multivalued dependency prevents another generalized multivalued dependency from being applied does not occur. Furthermore, enforcing conflict-freedom should not necessarily be seen as a restriction. On the contrary, it has been argued [23] that if a set of dependencies is not conflict-free, then part of the semantics is not adequately captured.

4.4 A Construction Algorithm

Here, we suggest an algorithm (i.e., Algorithm 2) to generate a unique dependency structure (an acyclic hypergraph) from a conflict-free full minimum cover of GMVDs. This algorithm is a modified version of Lien's algorithm [17].

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 2 constructs an acyclic hypergraph representing the dependency structure of the input GMVDs as follows:

Algorithm 2

procedure $Construction(G, (X_1, X_2, \ldots, X_p))$
$\mathcal{H}^0 := \{\mathcal{N}\}$;
for $i := 1$ to p
{
  $\mathcal{H}^{i-1} := \mathcal{H}^{i-1} - \{h_j\}$, where $X_i \subseteq h_j$;
  $\mathcal{H}^i := \mathcal{H}^{i-1} \cup \{X_i \cup (h_j \cap W) \mid W \in DEP(X_i)$
    and $h_j \cap W \neq \emptyset\}$;
}
return $(\mathcal{H}^p)$
end $Construction$

Example 3. Let $\mathcal{N} = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\}$ 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:

$$\begin{aligned} G = \big\{ & \{A_1\} \Rightarrow\Rightarrow \{A_7\} \mid \{A_2, A_3, A_4, A_5, A_6, A_8, A_9\}, \\ & \{A_2\} \Rightarrow\Rightarrow \{A_8\} \mid \{A_1, A_3, A_4, A_5, A_6, A_7, A_9\}, \\ & \{A_3\} \Rightarrow\Rightarrow \{A_9\} \mid \{A_1, A_2, A_4, A_5, A_6, A_7, A_8\}, \\ & \{A_1, A_2\} \Rightarrow\Rightarrow \{A_4\} \mid \{A_7\} \mid \{A_8\} \mid \{A_3, A_5, A_6, A_9\}, \\ & \{A_1, A_3\} \Rightarrow\Rightarrow \{A_5\} \mid \{A_7\} \mid \{A_9\} \mid \{A_2, A_4, A_6, A_8\}, \\ & \{A_2, A_3\} \Rightarrow\Rightarrow \{A_6\} \mid \{A_8\} \mid \{A_9\} \mid \{A_1, A_4, A_5, A_7\}\big\}. \end{aligned}$$

In the initialization step, the dependency structure is $\mathcal{H}^0 = \{\ \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\}\ \}$. For $i = 1$, the hyperedge $h_j \in \mathcal{H}^0$ is

$$h_j = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\}$$

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 h_j from $\mathcal{H}^0$ to obtain $\mathcal{H}^0 = \{\ \}$. The second step adds hyperedges

to construct $\mathcal{H}^1$. Since $W = \{A_7\} \in Dep(\{A_1\})$ and $h_j \cap \{A_7\} \neq \emptyset$, the hyperedge

$$X_1 \cup (h_j \cap \{A_7\}) = \{A_1, A_7\}$$

is added to $\mathcal{H}^1$. Similarly, $W = \{A_2, \ldots, A_9\} \in Dep(\{A_1\})$ and $h_j \cap \{A_2, \ldots, A_9\} \neq \emptyset$. Thus,

$$X_1(h_j \cap W) = \{A_1, \ldots, A_6, A_8, A_9\}$$

is added to $\mathcal{H}^1$. Thus, the intermediate dependency structure $\mathcal{H}^1$ is

$$\mathcal{H}^1 = \big\{\{A_1, A_7\}, \{A_1, A_2, A_3, A_4, A_5, A_6, A_8, A_9\}\big\}.$$

For $i = 2$, the hyperedge $h_j \in \mathcal{H}^1$ is

$$h_j = \{A_1, \ldots, A_6, A_8, A_9\}$$

since $X_2 = \{A_2\} \subseteq h_j$. The first step removes h_j from $\mathcal{H}^1$ obtaining $\mathcal{H}^1 = \{\{A_1, A_7\}\}$. The second step then adds the hyperedges $\{A_2, A_8\}$ and $\{A_1, \ldots, A_6, A_9\}$ to $\mathcal{H}^1$ to construct the intermediate dependency structure $\mathcal{H}^2$ as follows:

$$\mathcal{H}^2 = \{\{A_1, A_7\}, \{A_2, A_8\}, \{A_1, A_2, A_3, A_4, A_5, A_6, A_9\}\}.$$

The subsequent intermediate dependency structures generated by Algorithm 2 are:

$$\begin{aligned}
\mathcal{H}^3 =& \big\{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\}, \{A_1, A_2, A_3, A_4, A_5, A_6\}\big\},\\
\mathcal{H}^4 =& \big\{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\}, \{A_1, A_2, A_4\},\\
& \{A_1, A_2, A_3, A_5, A_6\}\big\},\\
\mathcal{H}^5 =& \big\{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\}, \{A_1, A_2, A_4\},\\
& \{A_1, A_3, A_5\}, \{A_1, A_2, A_3, A_6\}\big\},\\
\mathcal{H}^6 =& \big\{\{A_1, A_7\}, \{A_2, A_8\}, \{A_3, A_9\}, \{A_1, A_2, A_4\},\\
& \{A_1, A_3, A_5\}, \{A_2, A_3, A_6\}, \{A_1, A_2, A_3\}\big\}.
\end{aligned}$$

It can be easily verified that the output $\mathcal{H} = \mathcal{H}^6$ is an acyclic hypergraph. A hypertree construction ordering of this scheme is

$$\begin{aligned}
&h_1 = \{A_1, A_2, A_3\}, h_2 = \{A_1, A_2, A_4\},\\
&h_3 = \{A_1, A_3, A_5\}, h_4 = \{A_2, A_3, A_6\},\\
&h_5 = \{A_1, A_7\}, h_6 = \{A_2, A_8\}, h_7 = \{A_3, A_9\}.
\end{aligned}$$

Thus, the J-keys of $\mathcal{H}$ are

$$\begin{aligned}
h_2 \cap h_{b(2)} &= \{A_1, A_2\}, \quad b(2) = 1,\\
h_3 \cap h_{b(3)} &= \{A_1, A_3\}, \quad b(3) = 1,\\
h_4 \cap h_{b(4)} &= \{A_2, A_3\}, \quad b(4) = 1,\\
h_5 \cap h_{b(5)} &= \{A_1\}, \quad b(5) = 2,\\
h_6 \cap h_{b(6)} &= \{A_2\}, \quad b(6) = 4,\\
h_7 \cap h_{b(7)} &= \{A_3\}, \quad b(7) = 3.
\end{aligned}$$

Algorithm 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 2 is $O(|G|)$.

It is worth mentioning that we are not simply using Lien's algorithm [17] with a different kind of independency as input. Lien's input type is called "MVDs with nulls" (NMVDs) with which some properties of GMVDs (probabilistic conditional independencies) do not hold. In particular, NMVDs have the following two characteristics:

1. For any key X, $Dep(X)$ can be computed by those NMVDs with a key $Y \subseteq X$.
2. Two logically equivalent minimum covers have the same set of keys $\mathbf{X}$.

Characteristics 1 and 2 do not hold for GMVDs. A counterexample of 1 is given by G in (14). The dependency basis of key $\{A_1\}$ is $Dep(\{A_1\}) = \{\{A_2\}, \{A_3\}, \{A_4\}\}$, as shown in (15). However, the GMVD $\{A_1\} \Rightarrow\Rightarrow \{A_3\}$ cannot be derived without using the key $\{A_1, A_2\} \not\subseteq \{A_1\}$. A counterexample of characteristic 2 is given by consideration of the sets G_1 and G_2 of GMVDs over $\mathcal{N} = \{A_1, A_2, A_3, A_4, A_5\}$, where

$$\begin{aligned}
G_1 &= \big\{\{A_5\} \Rightarrow\Rightarrow \{A_2\},\ \{A_1, A_5\} \Rightarrow\Rightarrow \{A_3\}\big\}\\
G_2 &= \big\{\{A_5\} \Rightarrow\Rightarrow \{A_2\},\ \{A_1, A_2, A_5\} \Rightarrow\Rightarrow \{A_3\}\big\}.
\end{aligned}$$

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.,

$$\mathbf{X}_1 = \big\{\{A_5\}, \{A_1, A_5\}\big\} \neq \mathbf{X}_2 = \big\{\{A_5\}, \{A_1, A_2, A_5\}\big\}.$$

Because of the above differences between "MVDs with nulls" and probabilistic conditional independencies, the construction algorithm (Algorithm 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.

4.5 The Relationship between the Constructed Dependency Structure and the Refined Input Set of Dependencies

Our goal now is to demonstrate that the acyclic hypergraph $\mathcal{H}$ constructed as output by Algorithm 2 is a perfect-map of the given input set G of GMVDs. That is, every GMVD logically implied by G can be inferred from $\mathcal{H}$ and every GMVD inferred from $\mathcal{H}$ is logically implied by G. If $\mathcal{H}$ is a hypergraph, then the set of GMVDs *generated by* $\mathcal{H}$ is the set of GMVDs $X \Rightarrow\Rightarrow Y$, where Y is the union of some disconnected components of the hypergraph $\mathcal{H} - X$ obtained from $\mathcal{H}$ by deleting the set X of nodes. That is, $\mathcal{H} - X = \{h - X \mid h \text{ is a hyperedge of } \mathcal{H}\} - \{\emptyset\}$. We then say that X *separates off* Y from the rest of the nodes.

Example 4. Consider the acyclic hypergraph $\mathcal{H}$ in Fig. 1. Let $X = \{A_2, A_3\}$. The disconnected components of $\mathcal{H}$ obtained by deleting the set X is the set

$$\mathcal{H} - X = \big\{\{A_1\}, \{A_4\}, \{A_5, A_6\}\big\}.$$

Three GMVDs generated by $\mathcal{H}$ 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

$$X_i \Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,m_i-1} \mid W_{i,m_i}, \quad (16)$$
$$X_k \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,l} \mid \ldots \mid W_{k,m_k-1} \mid W_{k,m_k}. \quad (17)$$

Case 1. $X_i - X_k \neq \emptyset$ and $X_k - X_i \neq \emptyset$. Since keys are not split, we know precisely one $X_iW_{i,j}$ contains X_k. Without loss of generality, we may choose $j = 1$, namely,

$$X_k \subseteq X_iW_{i,1}.$$

Applying augmentation on (17), we derive the GMVDs

$$X_iW_{i,1} \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,m_k-1} \mid W_{k,m_k}.$$

By transitivity, we obtain:

$$\begin{aligned} X_i &\Rightarrow\Rightarrow W_{k,1} - X_iW_{i,1}, \\ X_i &\Rightarrow\Rightarrow W_{k,2} - X_iW_{i,1}, \\ &\vdots \\ X_i &\Rightarrow\Rightarrow W_{k,m_k-1} - X_iW_{i,1}, \\ X_i &\Rightarrow\Rightarrow W_{k,m_k} - X_iW_{i,1}. \end{aligned} \quad (18)$$

Since $X \Rightarrow\Rightarrow Y$ if and only if $X \Rightarrow\Rightarrow (Y - X)$ (see Section 2.3), (18) can be rewritten as:

$$\begin{aligned} X_i &\Rightarrow\Rightarrow W_{k,1} - W_{i,1}, \\ X_i &\Rightarrow\Rightarrow W_{k,2} - W_{i,1}, \\ &\vdots \\ X_i &\Rightarrow\Rightarrow W_{k,m_k-1} - W_{i,1}, \\ X_i &\Rightarrow\Rightarrow W_{k,m_k} - W_{i,1}. \end{aligned} \quad (19)$$

Similarly, we also know precisely one $X_kW_{k,j}$ contains X_i. Without loss of generality, we may assume $j = l$, namely,

$$X_i \subseteq X_kW_{k,l}.$$

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 (16), we derive the GMVDs

$$X_kW_{k,l} \Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,m_i-1} \mid W_{i,m_i}.$$

Applying transitivity using these GMVDs and (17), we obtain

$$\begin{aligned} X_k &\Rightarrow\Rightarrow W_{i,1} - X_kW_{k,l} = W_{i,1} - W_{k,l}, \\ X_k &\Rightarrow\Rightarrow W_{i,2} - X_kW_{k,l} = W_{i,2} - W_{k,l}, \\ &\vdots \\ X_k &\Rightarrow\Rightarrow W_{i,m_i} - X_kW_{k,l} = W_{i,m_i} - W_{k,l}. \end{aligned} \quad (20)$$

We now make some observations which will be used in showing the main result of our paper. (To improve readability, the proofs have been moved to the Appendix.)

Proposition 1. *No $W_{k,j}$ in $Dep(X_k)$, $j \neq l$, partially intersects $W_{i,1}$.*

Proposition 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 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 4. *$W_{k,l} - X_iW_{i,1} \neq \emptyset$. More specifically,*

$$W_{k,l} - X_iW_{i,1} = \cup \{W \mid W \in Dep'(X_i) \text{ and } W \notin Dep(X_i \cap X_k)\}.$$

Proposition 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, 2, 3, and 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 6. *No $W_{k,j}$ partially intersects $W_{i,1}$.*

Proposition 7. *Every $W_{i,s}$ in $Dep'(X_i)$ also belongs to $Dep(X_k)$.*

Proposition 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 (16) and (17), respectively, where $X_k \subseteq X_iW_{i,1}$ and $X_i \subseteq X_kW_{k,l}$. Since Proposition 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 (17) as

$$X_k \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,l-1} \mid W_{k,l} \mid W_1 \mid \ldots \mid W_s, \quad (21)$$

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 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 4, we can write $Dep(X_i)$ in (16) and $Dep(X_k)$ in (21) as

$$X_i \Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,r} \mid W_{i,r+1} \mid \ldots \mid W_{i,m_i} \quad (22)$$

and

$$X_k \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,l-1} \mid \big((Z)(W_{i,2} \cdots W_{i,r})\big) \mid W_1 \mid \ldots \mid W_s, \quad (23)$$

where $W_{i,j} \notin Dep(X_i \cap X_k)$,

$$j = 2, \ldots, r,$$

and

$$W_{k,l} = \big((Z)(W_{i,2} \cdots W_{i,r})\big).$$

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 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 (24) as

$$X_i \Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,r} \mid W_1 \mid \ldots \mid W_s. \tag{24}$$

Since $W_{k,j} \cap W_{i,1} = W_{k,j}$, $j = 1, \ldots, l-1$, we further clarify $Dep(X_i)$ in (24) as

$$X_i \Rightarrow\Rightarrow \big((Y)(W_{k,1} \cdots W_{k,l-1})\big) \mid W_{i,2} \mid \ldots \mid W_{i,r} \mid W_1 \mid \ldots \mid W_s, \tag{25}$$

where $W_{i,1} = ((Y)(W_{k,1} \cdots W_{k,l-1}))$. By substituting for Y in the above equation and Z in (23), we obtain the most detailed description of $Dep(X_i)$ and $Dep(X_k)$:

$$X_i \Rightarrow\Rightarrow \Big(\big((X_k - X_i)(V)\big)(W_{k,1} \cdots W_{k,l-1})\Big) \mid W_{i,2} \mid \ldots \mid W_{i,r} \mid W_1 \mid \ldots \mid W_s \tag{26}$$

and

$$X_k \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,l-1} \mid \Big(\big((X_i - X_k)(V)\big)(W_{i,2} \cdots W_{i,r})\Big) \mid W_1 \mid \ldots \mid W_s, \tag{27}$$

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 6, 7, and 8, $Dep(X_i)$ in (16) and $Dep(X_k)$ in (17) take the detailed form:

$$X_i \Rightarrow\Rightarrow (X_k - X_i)(W_{k,1} \cdots W_{k,l-1}) \mid W_1 \mid \ldots \mid W_s \tag{28}$$

and

$$X_k \Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,l-1} \mid W_1 \mid \ldots \mid W_s. \tag{29}$$

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 2.

Recall that $\mathcal{H}^{k-1}$ denotes the intermediate dependency structure constructed after $k-1$ iterations of Algorithm 2. We want to show that X_k is a subset of a hyperedge h in $\mathcal{H}^{k-1}$, 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 $\mathcal{H}^{k-1}$ is a perfect-map of the following GMVDs:

$$\begin{aligned} X_1 &\Rightarrow\Rightarrow W_{1,1} \mid W_{1,2} \mid \ldots \mid W_{1,m_1}, \\ &\vdots \\ X_i &\Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,m_i}, \\ &\vdots \\ X_{k-1} &\Rightarrow\Rightarrow W_{k-1,1} \mid W_{k-1,2} \mid \ldots \mid W_{k-1,m_{k-1}}. \end{aligned} \tag{30}$$

From the assumption that $\mathcal{H}^{k-1}$ is a perfect-map of (30), X_i is a J-key of $\mathcal{H}^{k-1}$ and $X_i W_{i,1}$ is a disconnected component when X_i is deleted from $\mathcal{H}^{k-1}$. (The same can be said about all the X_is in (30).) Clearly, each $X_i W_{i,1}$ can be characterized by a unique set $\overline{X_i W_{i,1}}$ of hyperedges in $\mathcal{H}^{k-1}$, namely,

$$\overline{X_i W_{i,1}} = \{h \mid h \in \mathcal{H}^{k-1} \text{ and } h \cap W_{i,1} \neq \emptyset\}.$$

Proposition 9. *X_k is a subset of exactly one hyperedge h in $\mathcal{H}^{k-1}$.*

Proposition 10. *Let h be the hyperedge in $\mathcal{H}^{k-1}$ containing X_k. Then,*

$$\{h\} = \overline{X_1 W_{1,1}} \cap \ldots \cap \overline{X_{k-1} W_{k-1,1}}.$$

This concludes our analysis of the graphical properties of the dependency structure constructed by Algorithm 2. In Propositions 1-6, 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 paper, we need to derive one additional property about $Dep(X_k)$ using the graphical properties in Propositions 9 and 10.

Recall that

$$\overline{X_i W_{i,1}} = \{h \mid h \in \mathcal{H}^{k-1} \text{ and } h \cap W_{i,1} \neq \emptyset\}$$

and from Proposition 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 paper.

Theorem 2. *The dependency structure $\mathcal{H}^k$ constructed from Algorithm 2 is a perfect-map of the GMVDs*

$$\begin{aligned} X_1 &\Rightarrow\Rightarrow W_{1,1} \mid W_{1,2} \mid \ldots \mid W_{1,m_1}, \\ &\vdots \\ X_i &\Rightarrow\Rightarrow W_{i,1} \mid W_{i,2} \mid \ldots \mid W_{i,m_i}, \\ &\vdots \\ X_{k-1} &\Rightarrow\Rightarrow W_{k-1,1} \mid W_{k-1,2} \mid \ldots \mid W_{k-1,m_{k-1}}, \\ X_k &\Rightarrow\Rightarrow W_{k,1} \mid W_{k,2} \mid \ldots \mid W_{k,m_k}. \end{aligned} \tag{31}$$

Proof. We will prove this claim by induction.

Basic step. $h = \{\mathcal{N}\}$,

$$X_1 \Rightarrow\Rightarrow W_{1,1} \mid W_{1,2} \mid \ldots \mid W_{1,m_1}.$$

Obviously, $\mathcal{H}^1$ is a perfect-map for the above GMVDs and X_1 is the J-key of $\mathcal{H}^1$. Inductive hypothesis: $\mathcal{H}^{k-1}$ is a perfect-map for (30) and $X_1, X_2, \ldots, X_{k-1}$ are the J-keys of $\mathcal{H}^{k-1}$.

Based on previous discussions, we have the following observations: Proposition 9 states there exists one and only one hyperedge $h \in \mathcal{H}^{k-1}$ that contains X_k. Proposition 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 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,

$$\left(\cap_{i=1}^{k-1} X_iW_{i,1}\right) \ \cap \ \left(\cup_{i=1}^{k-1}\left(\cup \{W \mid W \in Dep'(X_i \cap X_k)\}\right)\right) = \emptyset.$$

By definition, $h = \cap_{i=1}^{k-1} X_iW_{i,1}$ and, thereby,

$$h \ \cap \ \left(\cup_{i=1}^{k-1}\left(\cup \{W \mid W \in Dep'(X_i \cap X_k)\}\right)\right) = \emptyset. \qquad (32)$$

On the other hand, we have shown in Proposition 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 $\mathcal{H}^{k-1}$, we obtain the hypergraph $\mathcal{H}^k$ which contains the following *new* hyperedges from h:

$$\{\ h_{k,j} = X_k(h \cap W_{k,j}) \mid W_{k,j} \in D\ \}.$$

Obviously, $\mathcal{H}^k$ is an acyclic hypergraph if $\mathcal{H}^{k-1}$ is acyclic. Moreover, $X_1, X_2, \ldots, X_{k-1}$ are J-keys of $\mathcal{H}^k$ and, when X_i $(1 \leq i \leq k-1)$ is deleted from $\mathcal{H}^k$, the disconnected components of X_i are the same as those when X_i is deleted from $\mathcal{H}^{k-1}$.

For every $W_{k,j} \in D$ such that $X_kW_{k,j}$ does not contain any key,

$$h \cap W_{k,j} = W_{k,j}.$$

Whenever $X_kW_{k,j}$ contains some key(s), $W_{k,j} \in D$, then $W_{k,j}$ contains all the hyperedges in

$$\cup_{i=1}^{k-1} \ (Dep'(X_i) - Dep'(X_i \cap X_k))). \qquad (33)$$

Obviously, every hyperedge in $\mathcal{H}^{k-1}$ must be connected to h through some key(s) in h. In particular, every hyperedge in (33) 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 $\mathcal{H}^k$. We can immediately conclude that, when X_k is deleted from $\mathcal{H}^k$, the resulting disconnected components are exactly equal to the elements in $Dep(X_k)$. By the inductive hypothesis that $\mathcal{H}^{k-1}$ is a perfect-map for (30), it immediately follows that $\mathcal{H}^k$ is a perfect-map for (31). □

We now show that a conflict-free full minimum cover G has a unique dependency structure $\mathcal{H}$. 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* [17], namely, by permuting a neighboring 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 $\mathcal{H}_q$ to denote the dependency structure constructed as output by Algorithm 2 using G with p-ordering sequence q.

Lemma 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 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, $\mathcal{H}_q = \mathcal{H}_{q'}$.*

Proof. Let $\mathcal{H}^{k-1}$ be the intermediate dependency structure constructed by Algorithm 2 after $k-1$ iterations. If X_k and X_l are contained in distinct hyperedges of $\mathcal{H}^{k-1}$, then the claim follows trivially. Suppose X_k and X_l are contained in the same hyperedge $h \in \mathcal{H}^{k-1}$. 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, $h \in \mathcal{H}^{k-1}$ is replaced with the hyperedges

$$\begin{aligned} h_1 &= X_k(W_{k,1} \cap h), \\ h_2 &= X_k(W_{k,2} \cap h), \\ &\vdots \\ h_i &= X_k(W_{k,i} \cap h), \\ h_{i+1} &= X_k(((X_l - X_k)(Z)(W_{l,1} \cdots W_{l,j})) \cap h) \\ &= (X_kX_lZ)(W_{l,1} \cdots W_{l,j}) \cap h). \end{aligned}$$

Now, $X_l \subseteq h_{i+1} \in \mathcal{H}^k$. By definition, h_{i+1} is replaced with the hyperedges

$$\begin{aligned}
h'_1 &= X_l(W_{l,1} \cap h_{i+1}),\\
h'_2 &= X_l(W_{l,2} \cap h_{i+1}),\\
&\vdots\\
h'_j &= X_l(W_{l,j} \cap h_{i+1}),\\
h'_{j+1} &= X_l(((X_k - X_l)(Z)(W_{k,1} \cdots W_{k,i})) \cap h_{i+1})\\
&= (X_l X_k Z)(W_{k,1} \cdots W_{k,i}) \cap h_{i+1})\\
&= (X_l X_k Z)(W_{k,1} \cdots W_{k,i}) \cap (X_k X_l Z)(W_{l,1} \cdots W_{l,j}) \cap h\\
&= (X_l X_k Z) \cap h.
\end{aligned}$$

Thus, $h \in \mathcal{H}^{k-1}$ is replaced with the set of hyperedges

$$h_1, h_2, \ldots, h_i, h'_1, h'_2, \ldots, h'_j, (X_l X_k Z) \cap h \tag{34}$$

in $\mathcal{H}^{k+1}$.

Now suppose, on the other hand, that q' is the p-ordering used in Algorithm 2. We will show that the intermediate constructed dependency structure $\mathcal{H}^{k+1}$ is the same. By definition, $h \in \mathcal{H}^{k-1}$ is replaced with the hyperedges

$$\begin{aligned}
h'_1 &= X_l(W_{l,1} \cap h),\\
h'_2 &= X_l(W_{l,2} \cap h),\\
&\vdots\\
h'_j &= X_l(W_{l,j} \cap h),\\
h'_{j+1} &= X_l(((X_k - X_l)(Z)(W_{k,1} \cdots W_{k,i})) \cap h)\\
&= (X_l X_k Z)(W_{k,1} \cdots W_{k,i}) \cap h).
\end{aligned}$$

Now, $X_k \subseteq h'_{j+1} \in \mathcal{H}^k$. By definition, h'_{j+1} is replaced with the hyperedges

$$\begin{aligned}
h_1 &= X_k(W_{k,1} \cap h'_{j+1}),\\
h_2 &= X_k(W_{k,2} \cap h'_{j+1}),\\
&\vdots\\
h_i &= X_k(W_{k,i} \cap h'_{j+1}),\\
h_{i+1} &= X_k(((X_l - X_k)(Z)(W_{l,1} \cdots W_{l,j})) \cap h'_{j+1})\\
&= (X_k X_l Z)(W_{l,1} \cdots W_{l,j}) \cap h'_{j+1})\\
&= (X_k X_l Z)(W_{l,1} \cdots W_{l,j}) \cap (X_l X_k Z)(W_{k,1} \cdots W_{k,i}) \cap h\\
&= (X_k X_l Z) \cap h.
\end{aligned}$$

Thus, $h \in \mathcal{H}^{k-1}$ is replaced with the set of hyperedges

$$h'_1, h'_2, \ldots, h'_j, h_1, h_2, \ldots, h_i, (X_k X_l Z) \cap h \tag{35}$$

in $\mathcal{H}^{k+1}$.

The desired result is obtained since (34) is identical to (35). □

Theorem 3. *A conflict-free full minimum cover G has a unique dependency structure. That is, the particular p-ordering sequence used in Algorithm 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 transputed 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 transmuted. 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 1 demonstrates that if two neighboring 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 $\mathcal{H}$. Thus, the constructed dependency structure $\mathcal{H}_q$ is identical to $\mathcal{H}_{q_{k+1}}$. □

5 Refining the Dependency Structure Using the Mixture of Nonembedded and Embedded Independencies

In this section, we outline how the constructed dependency structure can be refined using the embedded independency information supplied by the domain experts. This process will utilize the fact that the GMVD axioms are not only complete for nonembedded GMVDs, but are, in fact, complete for deriving all embedded GMVDs from other embedded GMVDs as long as the embedded GMVDs are defined over the same *fixed* set of attributes. The contraction axiom (SG4) can then be applied to derive GMVDs logically implied by other GMVDs defined over a *mixed* set of attributes.

Let G be a set of all GMVDs (not necessarily nonembedded) over the set $\mathcal{N}$ of all attributes in the multiagent problem domain supplied by the individual domain experts. For every GMVD $X \Rightarrow\Rightarrow Y|Z$ in G, construct the set G_{XYZ} of *nonembedded* GMVDs over the *fixed* context XYZ as follows:

$$G_{XYZ} = \{X \Rightarrow\Rightarrow Y|Z \mid X \Rightarrow\Rightarrow V|W \text{ in } G, \text{ and } Y \subseteq V,\ Z \subseteq W\}.$$

For example, consider the set G of GMVDs over $\mathcal{N} = \{A_1, A_2, A_3, A_4\}$

$$G = \big\{\{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3\},\ \{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\}\big\}.$$

We then construct the sets $G_{\{A_1,A_2,A_3\}}$ and $G_{\{A_1,A_2,A_3,A_4\}}$ of respective nonembedded GMVDs as follows:

$$G_{\{A_1,A_2,A_3\}} = \big\{\{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3\},\ \{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\}\big\} \tag{36}$$

and

$$G_{\{A_1,A_2,A_3,A_4\}} = \big\{\{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\}\big\}. \tag{37}$$

We apply the process described in Section 4 to remove redundancy, detect inconsistency, and compute a conflict-free full minimum cover, written G^c_{XYZ}, for each G_{XYZ} above. This set of conflict-free full minimum covers reflects

all logically implied conditional independency information for each fixed $XYZ \subset \mathcal{N}$. For the sets $G_{\{A_1,A_2,A_3\}}$ and $G_{\{A_1,A_2,A_3,A_4\}}$ of nonembedded GMVDs in (36) and (37), the respective conflict-free full minimum covers are

$$G^c_{\{A_1,A_2,A_3\}} = \Big\{ Dep(\{A_2\}) = \big\{\{A_1\},\{A_3\}\big\} \Big\} \tag{38}$$

and

$$G^c_{\{A_1,A_2,A_3,A_4\}} = \Big\{ Dep(\{A_2,A_3\}) = \big\{\{A_1\},\{A_4\}\big\} \Big\}. \tag{39}$$

The contraction axiom (SG4) is now applied to derive new conditional independencies from other conditional independencies defined on *mixed* sets of attributes. Applying (SG4) on the GMVD $\{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3\}$ over $\{A_1, A_2, A_3\}$ in (38) and the GMVD $\{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\}$ over $\{A_1, A_2, A_3, A_4\}$ in (39), we derive the new nonembedded GMVD $\{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3, A_4\}$ over $\{A_1, A_2, A_3, A_4\}$. We add this GMVD to the set $G_{\{A_1,A_2,A_3,A_4\}}$ in (37), namely,

$$\begin{aligned} & G_{\{A_1,A_2,A_3,A_4\}} \ \cup \ \big\{\{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3,A_4\}\big\} \\ =& \big\{\{A_2,A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\}\big\} \cup \\ & \big\{\{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3,A_4\}\big\} \\ =& \big\{\{A_2,A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\},\ \{A_2\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_3,A_4\}\big\}. \end{aligned}$$

An updated conflict-free full minimum cover $G^c_{\{A_1,A_2,A_3,A_4\}}$ for the nonembedded GMVDs over $\{A_1, A_2, A_3, A_4\}$ is then,

$$G^c_{\{A_1,A_2,A_3,A_4\}} = \Big\{ Dep(\{A_2\}) = \big\{\{A_1\},\{A_3,A_4\}\big\} \Big\}. \tag{40}$$

Note that the GMVD $\{A_2, A_3\} \Rightarrow\Rightarrow \{A_1\} \mid \{A_4\}$ in (39) is now redundant in (40). This demonstrates that the contraction axiom can use the *mixture* of embedded and nonembedded dependencies to derive GMVDs not derivable using the GMVD axiomatization. The conflict-free full minimum cover $G^c_{\mathcal{N}}$ and a p-ordering sequence can be supplied as input to Algorithm 2. The output dependency structure of the multiagent Markov network $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ is a perfect-map of $G^{\mathcal{N}}_c$.

In order to completely define a probabilistic network, one must specify the dependency structure and the corresponding probability tables. However, the potentials corresponding to an acyclic hypergraph 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. It is thereby 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 can then be defined in terms of the elicited conditional probability tables.

It is always possible to construct a DAG which reflects precisely the same probabilistic conditional independencies as and acyclic hypergraph $\mathcal{H}$. For example, consider the acyclic hypergraph $\mathcal{H} = \{h_1, h_2, h_3, h_4\}$ in Fig. 1. The dependency structure of the multiagent Bayesian network can be defined by adding the directed edge (A_2, A_3) to the DAG in Fig. 5. (The directed edge (A_2, A_3) could be equivalently replaced by (A_3, A_2).) Notice that the only probabilistic conditional independencies inferred from the modified DAG using d-separation [20] are exactly those nonembedded conditional independencies inferred from acyclic hypergraph $\mathcal{H}$. It is assumed that the domain experts are able to specify the conditional probability tables corresponding to the constructed DAG, namely, $\phi(\{A_1\})$, $\phi(\{A_2\}|\{A_1\})$, $\phi(\{A_3\}|\{A_1, A_2\})$, $\phi(\{A_4\}|\{A_2, A_3\})$, $\phi(\{A_5\}|\{A_2, A_3\})$, and $\phi(\{A_6\}|\{A_5\})$. These elicited conditional probability tables are represented as the relations $\Phi_{\{A_1\}}$, $\Phi_{\{A_1,A_2\}}$, $\Phi_{\{A_1,A_2,A_3\}}$, $\Phi_{\{A_2,A_3,A_4\}}$, $\Phi_{\{A_2,A_3,A_5\}}$, $\Phi_{\{A_5,A_6\}}$, respectively. At this point, the multiagent 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 multiagent dependency structure. The multiply sectioned Bayesian network technique [38] can be applied for this purpose. Thus, each agent initially is given a portion of the constructed DAG representing the dependency structure of the multiagent 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 [38]. The dependency structure of the multiagent system is thereby again a hypergraph, but the hyperedges are distributed among the agents. At this point, the multiagent system is ready for user input. Techniques have already been proposed for probabilistic reasoning in a distributed multiagent environment [6], [33].

6 Conclusion

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 multiagent 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 multiagent 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 [37] may be too restrictive in some situations.

In this paper, an automated procedure was proposed for directly constructing the dependency structure (an acyclic hypergraph) of a multiagent 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 paper 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 [10], [20], [29].

APPENDIX

PROOFS OF PROPOSITIONS

Here, we prove several results that are used in deriving the main result of our paper stated in Theorem 2.

Case 1. $X_i \not\subseteq X_k$.

Proposition 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 (19), 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

$$\begin{aligned}(W_{k,j} - W_{i,1}) - X_i \overline{W}_{k,l} &= (W_{k,j} - W_{i,1}) - X_k W_{k,l} \\ &= (W_{k,j} - W_{i,1}) - W_{k,l} = (W_{k,j} - W_{i,1}).\end{aligned}$$

Since $(W_{k,j} - W_{i,1}) \subset W_{k,j}$, then $Dep(X_k)$ can be refined. This is a contradiction. □

Proposition 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 (19), 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 that 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_is, 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 (20), 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 $W_{k,j} \in Dep(X_i \cap X_k)$. □

Proposition 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:

1. $W_{i,s}$ 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 (19), 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.
2. $W_{i,s}$ 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 $W_{i,1}$. □

Proposition 4. *$W_{k,l} - X_i W_{i,1} \neq \emptyset$. More specifically,*

$$W_{k,l} - X_i W_{i,1} = \cup \{W \mid W \in Dep'(X_i) \text{ and } W \notin Dep(X_i \cap X_k)\}.$$

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. □

Proposition 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, 2, 3, and 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 (19) 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

$$X_k \Rightarrow\Rightarrow (W_{k,l} - W_{i,1}) - X_i \overline{W}_{k,l}.$$

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 (41) 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 (20) 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 (20) leads to no refinement of $Dep(X_i)$. Since $X_i \Rightarrow\Rightarrow W_{i,1}$, we have

$$X_i \Rightarrow\Rightarrow X_iW_{i,1} = X_k\overline{W}_{i,1}, \text{ where } \overline{W}_{i,1} = X_iW_{i,1} - X_k.$$

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

$$X_i \Rightarrow\Rightarrow (W_{i,1} - W_{k,l}) - X_k\overline{W}_{i,1}. \quad (42)$$

Since $X_k\overline{W}_{i,1} = X_iW_{i,1}$ and $(W_{i,1} - W_{k,l}) \subseteq X_iW_{i,1}$, the GMVD in (42) is $X_i \Rightarrow\Rightarrow \emptyset$. Thus, $Dep(X_i)$ cannot be refined with this GMVD. □

Case 2. $X_i \subseteq X_k$.

Proposition 6. *No $W_{k,j}$ partially intersects $W_{i,1}$.*

Proof. Suppose $W_{k,j} - W_{i,1} \subset W_{k,j}$. From (19),

$$X_i \Rightarrow\Rightarrow W_{k,j} - W_{i,1} \equiv W'_{k,j} \subset W_{k,j}.$$

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. □

Proposition 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 3. □

Proposition 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 2. □

Proposition 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. □

Proposition 10. *Let h be the hyperedge in $\mathcal{H}^{k-1}$ containing X_k. Then,*

$$\{h\} = \overline{X_1W_{1,1}} \cap \ldots \cap \overline{X_{k-1}W_{k-1,1}}.$$

Proof. Obviously, by Proposition 9, we have

$$\{h\} \subseteq \overline{X_1W_{1,1}} \cap \ldots \cap \overline{X_{k-1}W_{k-1,1}}.$$

It is also clear that any hyperedge $h' \in \mathcal{H}^{k-1}$, $h' \neq h$, h' is not an element of some $\overline{X_iW_{i,1}}$. Thus,

$$X_k \subseteq \{h\} = \overline{X_1W_{1,1}} \cap \ldots \cap \overline{X_{k-1}W_{k-1,1}}. \quad \square$$

REFERENCES

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

[2] C. Beeri, R. Fagin, and J. Howard, "A Complete Axiomatization for Functional and Multivalued Dependencies in Database Relations," *Proc. ACM SIGMOD,* pp. 47-61, 1977.

[3] C. Beeri, "On the Membership Problem for Functional and Multivalued Dependencies in Relational Databases," *ACM Trans. Database Systems,* vol. 5, no. 3, pp. 241-259, 1980.

[4] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, "On the Desirability of Acyclic Database Schemes," *J. ACM,* vol. 30, no. 3, pp. 479-513, 1983.

[5] U. Bertele and F. Brioschi, *Nonserial Dynamic Programming.* Academic Press, 1972.

[6] C.J. Butz and S.K.M. Wong, "Recovery Protocols in Multi-Agent Probabilistic Reasoning Systems," *Proc. Int'l Database Eng. and Applications Symp.,* pp. 302-310, 1999.

[7] R. Dechter, A. Dechter, and J. Pearl, "Optimization in Constraint Networks," *Influence Diagrams, Belief Nets, and Decision Analysis,* R.M. Oliver and J.Q. Smith, eds., pp. 411-425, Wiley, 1990.

[8] R. Fagin, "Multivalued Dependencies and a New Normal Form for Relational Databases," *ACM Trans. Database Systems,* vol. 2, no. 3, pp. 262-278, 1977.

[9] Z. Galil, "An Almost Linear-Time Algorithm for Computing a Dependency Basis in a Relational Database," *J. ACM,* vol. 29, no. 1, pp. 96-102, 1982.

[10] D. Geiger and J. Pearl, "Logical and Algorithmic Properties of Conditional Independence," Technical Report R-97-II-L, Univ. of California, Los Angeles, 1989.

[11] P. Hajek, T. Havranek, and R. Jirousek, *Uncertain Information Processing in Expert Systems.* CRC Press, 1992.

[12] J. Hu and Y. Xiang, "Learning Belief Networks in Domains with Recursively Embedded Pseudo Independent Submodels," *Proc. 13th Conf. Uncertainty in Artificial Intelligence,* pp. 258-265, 1997.

[13] F. Jensen, "Junction Tree and Decomposable Hypergraphs," technical report, JUDEX, Aalborg, Denmark, 1988.

[14] F. Jensen, *An Introduction to Bayesian Networks* UCL, 1996.

[15] S.L. Lauritzen and D.J. Spiegelhalter, "Local Computation with Probabilities on Graphical Structures and Their Application to Expert Systems," *J. Royal Statistical Soc. B,* vol. 50, pp. 157-244, 1988.

[16] Y.E. Lien, "Hierarchical Schemata for Relational Databases," *ACM Trans. Database Systems,* vol. 6, no. 1, pp. 48-69, 1981.

[17] Y.E. Lien, "On the Equivalence of Database Models," *J. ACM,* vol. 29, no. 2, pp. 336-362, 1982.

[18] D. Maier, *The Theory of Relational Databases.* Computer Science Press, 1983.

[19] R. Neapolitan, *Probabilistic Reasoning in Expert Systems.* Wiley, 1990.

[20] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[21] D.J. Rose, "Triangulated Graphs and the Elimination Process," *J. Math. Analysis Applications,* vol. 32, pp. 597-609, 1970.

[22] Y. Sagiv, "An Algorithm for Inferring Multivalued Dependencies with an Application to Propositional Logic," *J. ACM,* vol. 27, no. 2, pp. 250-262, 1980.

[23] E. Sciore, "Some Observations on Real-World Data Dependencies," *Proc. XP1 Workshop,* 1980.

[24] G. Shafer, "An Axiomatic Study of Computation in Hypertrees," School of Business Working Papers 232, Univ. of Kansas, Lawrence, 1991.

[25] M. Studeny, "Conditional Independence Relations Have No Finite Complete Characterization," *Proc. 11th Prague Conf. Information Theory, Statistical Decision Foundation, and Random Processes,* pp. 377-396, 1990.

[26] T. Verma and J. Pearl, "Equivalence and Synthesis of Causal Models," *Proc. Sixth Conf. Uncertainty in Artificial Intelligence,* pp. 220-227, 1990.

[27] S.K.M. Wong, "Testing Implication of Probabilistic Dependencies," *Proc. 12th Conf. Uncertainty in Artificial Intelligence,* pp. 545-553, 1996.

[28] S.K.M. Wong, "An Extended Relational Data Model for Probabilistic Reasoning," *J. Intelligent Information Systems,* vol. 9, pp. 181-202, 1997.

[29] S.K.M. Wong, "The Relational Structure of Belief Networks," *J. Intelligent Information Systems,* 1998.

[30] S.K.M. Wong, "A Rule-Based Language for Probabilistic Reasoning," *Information, Uncertainty, and Fusion,* B. Bouchon-Meunier, R. Yager, and L. Zadeh, eds., pp. 221-232, Kluwer, 2000.
[31] S.K.M. Wong and Y. Xiang, "Construction of a Markov Network from Data for Probabilistic Inference," *Proc. Third Int'l Conf. Rough Sets and Soft Computing,* pp. 562-569, 1994.
[32] S.K.M. Wong and Z. Wang, "On Axiomatization of Probabilistic Conditional Independence," *Proc. 10th Conf. Uncertainty in Artificial Intelligence,* pp. 591-597, 1994.
[33] S.K.M. Wong and C.J. Butz, "Probabilistic Reasoning in a Distributed Multi-Agent Environment," *Proc. Third Int'l Conf. Multi-Agent Systems,* pp. 341-348, 1998.
[34] S.K.M. Wong and C.J. Butz, "Contextual Weak Independence in Bayesian Networks," *Proc. 15th Conf. Uncertainty in Artificial Intelligence,* pp. 670-679, 1999.
[35] S.K.M. Wong, C.J. Butz, and Y. Xiang, "A Method for Implementing a Probabilistic Model as a Relational Database," *Proc. 11th Conf. Uncertainty in Artificial Intelligence,* pp. 556-564, 1995.
[36] S.K.M. Wong, C.J. Butz, and Y. Xiang, "Automated Database Scheme Design Using Mined Data Dependencies," *J. Am. Soc. Information Science,* vol. 49, no. 5, pp. 455-470, 1998.
[37] Y. Xiang, "Verification of DAG Structures in Cooperative Belief Network-Based Multi-agent Systems," *Networks,* vol. 31, pp. 183-191, 1998.
[38] Y. Xiang, D. Poole, and M.P. Beddoes, "Multiply Sectioned Bayesian Networks and Junction Forests for Large Knowledge-Based Systems," *Computational Intelligence,* vol. 9, pp. 171-220, 1993.

S.K. Michael Wong received the BSc degree from the University of Hong Kong in 1963. He received the MA and PhD degrees in theoretical physics from the University of Toronto, Ontario, Canada, in 1964 and 1968, respectively. Before he joined the Department of Computer Science at the University of Regina in 1982, he worked in various computer-related industries. Currently, he is a professor of computer science. His research interests include uncertainty reasoning, information retrieval, database systems, and data mining. He is a member of the IEEE and the IEEE Computer Society.

Cory J. Butz received the BSc, MSc, and PhD degrees in computer science from the University of Regina, Saskatchewan, Canada in 1994, 1996, and 2000, respectively. He has agreed to join the School of Information Technology and Engineering at the University of Ottawa, Ontario, Canada, as an assistant professor. His research interests include uncertainty reasoning, information retrieval, database systems, and data mining.

▷ **For further information on this or any computing topic, please visit our Digital Library at** http://computer.org/publications/dlib.