NORTH-HOLLAND

A Recovery Algorithm for Chain Graphs

Milan Studený
Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, Prague, Czech Republic.

ABSTRACT

The class of chain graphs (CGs) *involving both undirected graphs (= Markov networks) and directed acyclic graphs (= Bayesian networks) was introduced in middle eighties for description of probabilistic conditional independence structures. Every class of Markov equivalent* CGs *(that is,* CGs *describing the same conditional independence structure) has a natural representative, which is called the largest* CG. *The paper presents a recovery algorithm, which on the basis of the conditional independence structure given by a* CG *(in the form of a dependency model) finds the largest* CG *representing the corresponding class of Markov equivalent* CGs. *As a by-product a graphical characterization of graphs which are the largest* CGs *(for a class of Markov equivalent* CGs) *is obtained, and a simple algorithm changing every* CG *into the largest* CG *of the corresponding equivalence class is given.* © *1997 Elsevier Science Inc.*

KEYWORDS: ***chain graph, dependency model, Markov equivalence, pattern, largest chain graph, recovery algorithm***

1. INTRODUCTION

Classical graphical approaches to the description of probabilistic conditional independence structures use either undirected graphs (UGs), also called Markov networks, or directed acyclic graphs (DAGs), known as Bayesian networks or (probabilistic) influence diagrams. In middle eighties Lauritzen and Wermuth [12] introduced the class of *chain graphs* (CGs), which includes both UGs and DAGs, but not only them. In CGs both undirected edges, called *lines*, and directed edges, called *arrows*, are

Address correspondence to Milan Studený, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 4, 182 08 Prague, Czech Republic. E-*mail:* studeny@utia.cas.cz.

Received 1 December 1996; accepted 1 December 1996.

0888-613X/97/$17.00
PII S0888-613X(97)00018-2

simultaneously allowed, but directed cycles are for forbidden (nevertheless, undirected cycles are allowed). To establish semantics for CGs, Lauritzen [14], followed by Frydenberg [7], generalized the so-called *moralization criterion* for DAGs from [15] for reading independencies from a CG. It is an indirect criterion consisting of three steps: restricting the CG to a certain set of nodes, transforming it properly to an UG (called the *moral graph*), and using the separation criterion for UGs with respect to the moral graph. An equivalent *c-separation criterion*, testing trails in the original CG directly, was proposed in [3]. It generalizes the well-known d-separation criterion for DAGs introduced by Pearl [17]. The separation criterion for CGs helped lately to confirm Lauritzen's and Frydenberg's conjecture [14, 7] that for every CG there exists a probability distribution which exhibits exactly those independencies which can be read from the graph according to the moralization criterion. This generalizes analogous results for UGs [8, 10] and DAGs [9].

Several recent works show that CGs are attracting the attention of researchers. Whittaker [23] gave several examples of the use of CGs; Cox and Wermuth [6] considered a wider class of chain graphs having two further types of edges, namely "dashed lines" and "dashed arrows". Andersson, Madigan, and Perlman [1] used special CGs, called *essential graphs*, to represent uniquely classes of Markov equivalent DAGs and characterized the essential graphs in graphical terms. Meek [16] followed the method of [22] and proposed an algorithm which on basis of the conditional independence structure given by a DAG finds the above-mentioned essential graph. Buntine [4] gave an equivalent definition of a CG as a hierarchical combination of Markov and Bayesian networks.

In case of UGs, the conditional independence structure given by an UG uniquely determines the graph. This is not true in case of DAGs, where different DAGs can describe the same conditional independence structure (then we say that the DAGs are Markov equivalent). The same situation occurs in the case of CGs. Frydenberg [7] characterized Markov equivalent CGs in graphical terms, namely as CGs having the same underlying graph and the same occurrences of (minimal) complexes. This result generalizes an analogous characterization of Markov equivalence for DAGs from [21]. Unlike the case of DAGs, where the class of Markov equivalent DAGs has no distinguished member, every class of Markov equivalent CGs can be naturally represented by a special CG within the class. Frydenberg [7] showed that every class of Markov equivalent CGs has a CG with the greatest number of lines (or dually with the least number of arrows). This graph is called the *largest CG* of the corresponding class of Markov equivalent CGs. Note for explanation that the essential graph of a class of Markov equivalent DAGs does not coincide in general with the largest CG of the corresponding class of Markov equivalent CGs.

This paper describes a *recovery algorithm* which, on the basis of the conditional independence structure given by an unknown CG, finds the largest CG of the corresponding class of Markov equivalent CGs. Like analogous procedures from [22] and [16] for finding the essential graph of a class of Markov equivalent DAGs, the presented recovery algorithm has two stages. First, on the basis of special "elementary" statements obtained from the dependency model, one identifies the edges of the underlying graph and the occurrences of complexes, and forms the so-called *pattern* of the equivalence class. It is a special graph, having the required underlying graph and only arrows produced by the complexes (the other edges are lines). According to the above-mentioned Frydenberg's characterization of Markov equivalence [7], this pattern uniquely determines the class of Markov equivalent CGs. However, the pattern is not a CG in general, and some of its lines have to be changed into arrows to obtain the corresponding largest CG. This is done in the second stage of the recovery algorithm by repeated application of three special rules.

Moreover, the graphs which are the largest CGs of classes of Markov equivalent CGs are characterized in graphical terms. In fact, a simple algorithm which on the basis of a CG finds the corresponding largest CG is presented. It applies a so-called *pool-component rule* to the given CG, and a CG coincides with the corresponding largest CG iff this rule cannot be applied, or equivalently, iff the given CG has maximal connectivity components in certain sense.

The structure of the paper is as follows. In the second section basic definitions and important results are recalled. The third section describes the first stage of the recovery algorithm, namely how the above-mentioned pattern of the class of Markov equivalent CGs can be obtained on the basis of the conditional independence structure given by a CG. It is shown that the presented algorithm really yields the desired pattern. The fourth section deals with the second stage of the recovery algorithm. Three basic rules, namely the *transitivity rule*, the *necessity rule*, and the *double-cycle rule*, for changing the pattern into the corresponding largest CG are formulated, and the proof of completeness of those rules is given (that is, the proof that their application really yields the largest CG). A formal strengthening of the basic rules is also discussed in a subsection of the fourth section. The fifth section describes the pool-component rule for obtaining the largest CG of a class of Markov equivalent CGs directly from a member of the class. As a consequence of the preceding results a characterization of largest CGs is obtained. In the last (sixth) section, named Conclusions, a few remarks on presented results are given.

Note that several results of this paper were already formulated in the conference contribution [19], but the proof of the main result was omitted there owing to space limitation.

2. BASIC CONCEPTS

2.1. Graphs

A *hybrid graph* G over a nonempty finite set of *nodes* N is specified by a set of two-element subsets of N, called *edges*, where every edge $\{u, v\}$ is either a *line* (= undirected edge), denoted by $u — v$ or $v — u$, or an *arrow* (= directed edge) from u to v, denoted by $u \to v$ or $v \leftarrow u$, or an arrow from v to u, denoted by $u \leftarrow v$ or $v \to u$. An *undirected graph* (UG) is a graph containing only lines; a *directed graph* is a graph containing only arrows. The *underlying graph* of G is obtained from G by changing all edges of G into lines. The *induced subgraph* of G on a nonempty set $T \subset N$, denoted by G_T, is the graph over T which has exactly those edges in G which are subsets of T. *Connectivity components* of G are obtained by removing all arrows in G and taking the connectivity components of the remaining undirected graph.

A *route* in G is a sequence of its nodes $v_1, \dots, v_k$, $k \geq 1$, such that $\{v_i, v_{i+1}\}$ is an edge in G for every $i = 1, \dots, k-1$. It is called a *path* if it consists of distinct nodes. It is called a *pseudocycle* if $v_1 = v_k$, and a *cycle* if moreover $k \geq 4$ and $v_1, \dots, v_{k-1}$ are distinct. A cycle $v_1, \dots, v_k$, $k \geq 4$, is called a *chordless cycle* in G if there is no other edge in G between nodes $\{v_1, \dots, v_{k-1}\}$ except the mentioned edges of the cycle. Such an additional edge is called a *chord* of the cycle. A (pseudo)cycle is *undirected* if it consists of lines only. A (pseudo)cycle is *directed* if $v_i \to v_{i+1}$ or $v_i — v_{i+1}$ for $i = 1, \dots, k-1$, and $v_j \to v_{j+1}$ for at least one $j \in \{1, \dots, k-1\}$. A *directed acyclic graph* (DAG) is a directed graph without directed cycles.

A *complex* in G is a special induced subgraph of G, namely a path $v_1, \dots, v_k$, $k \geq 3$, such that $v_1 \to v_2$, $v_i — v_{i+1}$ for $i = 2, \dots, k-2$, $v_{k-1} \leftarrow v_k$ in G, and no additional edges between nodes of $\{v_1, \dots, v_k\}$ exist in G. The nodes v_1 and v_k are called the *parents* of the complex, the set $\{v_2, \dots, v_{k-1}\}$ the *region* of the complex, and the number $k - 2$ the *degree* of the complex. The set of parents of a complex κ will be denoted by $\mathrm{par}(\kappa)$. Note that the concept of complex is equivalent to Frydenberg's notion of "minimal complex" [7]. I decided to simplify the terminology because I believe that "nonminimal complexes" have no reasonable use.

Having nodes u, v with $u \to v$ in G, u is called a *parent* of v and v a *child* of u. In case $u — v$ they are *siblings*. Supposing A is a set of nodes of G, the set of parents [children] of nodes in A will be denoted by $\mathrm{pa}_G(A)$ [$\mathrm{ch}_G(A)$]. The boundary of A, denoted by $\mathrm{bd}_G(A)$, is the set of parents and siblings of nodes in A which are not in A. The symbol of the graph G is omitted when it is clear from context; a singleton $\{u\}$ will be often denoted by the symbol of the corresponding node u.

A route $v_1, \dots, v_k, k \geq 1$, is *descending* if $v_i \to v_{i+1}$ or $v_i — v_{i+1}$ for $1 \leq i \leq k-1$. In particular, an undirected path is considered to be a descending path. Another special descending path is a *slide*, that is, a path $v_1, \dots, v_k$, $k \geq 2$, such that $v_1 \to v_2$ and $v_i — v_{i+1}$ for $i = 2, \dots, k-1$. If there exists a descending path from a node u to a node v, then u is an *ancestor* of v, or dually v is a *descendant* of u. Having a set of nodes $A \subset N$, its *ancestral set*, denoted by $\text{an}_G(A)$, is the set of all ancestors of nodes in A (it contains A).

A *chain* for a hybrid graph G over N is a partition of N into ordered disjoint (nonempty) subsets $B_1, \dots, B_n$, $n \geq 1$, called *blocks*, such that if $\{u, v\}$ is an edge with $u, v \in B_i$ then $u — v$, and if $\{u, v\}$ is an edge with $u \in B_i$, $v \in B_j$, $i < j$, then $u \to v$. The original definition of CG (which also explains the terminology) is the following one.

DEFINITION 2.1 *A* chain graph (CG) *is a hybrid graph which admits a chain.*

However, there are several equivalent definitions of CG mentioned in the following lemma. They imply that CGs include both UGs and DAGs.

LEMMA 2.1 *The following conditions are equivalent for a hybrid graph G:*
- **(i)** *G is a chain graph,*
- **(ii)** *G has no directed pseudocycles,*
- **(iii)** *G has no directed cycles,*
- **(iv)** *G has no directed chordless cycles,*
- **(v)** *the set of connectivity components of G can be ordered to form a chain.*

Proof To verify (i) $\Rightarrow$ (ii) by contradiction let us consider a directed pseudocycle $v_1, \dots, v_k$, $k \geq 4$, in a CG G. Let us take the last block B_r of a chain $B_1, \dots, B_n$ which is hit by a node v_i of the pseudocycle (i.e. $v_i \in B_r$). As B_r is the last hit block, the possibility $v_i \to v_{i+1}$ is excluded. Therefore $v_i — v_{i+1}$ and $v_{i+1} \in B_r$. By repetition of this consideration we show that all nodes of the pseudocycle belong to B_r (recall that $v_k = v_1$), which contradicts the assumption that the pseudocycle contains at least one arrow.

To show (ii) $\Leftrightarrow$ (iii) it suffices to realize that a directed cycle can be made from a directed pseudocycle by successive removal of its parts between two different occurrences of the same node.

A similar principle holds for the proof of (iii) $\Leftrightarrow$ (iv). If a directed cycle $v_1, \dots, v_k$, $k \geq 4$, has a chord $\{v_i, v_j\}$ in G, $1 \leq i, j \leq k-1$, $2 \leq j - i \leq k-3$, then either $v_1, \dots, v_i, v_j, \dots, v_k$ or $v_i, \dots, v_j, v_i$ is a directed cycle in

G, and one obtains a chordless directed cycle by successive removal of chords.

The implication (ii) $\Rightarrow$ (v) can be proved by induction on the number of connectivity components of G. Supposing G satisfies (ii), the first observation is that there is no arrow between nodes of the same connectivity component of G, as otherwise the arrow together with an undirected path connecting the nodes would form a directed pseudocycle. Thus, the claim is evident if G has just one component. The second observation is that there exist a component C of G with $\mathrm{ch}_G(C) = \emptyset$. Indeed, otherwise one could construct a never-ending descending path owing to the fact that every component has a child. Owing to (ii), that path would never return to the same component, which contradicts the assumption that G has finitely many nodes. Let us take a component C of G with $\mathrm{ch}_G(C) = \emptyset$. Then the induced subgraph $G_{N \setminus C}$ also satisfies (ii) and has the same components as G with the exception of C. Moreover, all arrows entering C are directed into nodes of C; and C can be added as the last block to a chain of connectivity components of $G_{N \setminus C}$.

The implication (v) $\Rightarrow$ (i) is trivial. ■

A connectivity component C of a CG G is called *terminal* if $\mathrm{ch}_G(C) = \emptyset$. Lemma 2.1(v) implies that every CG has at least one terminal connectivity component, namely the last block of a chain of components. Note that one CG may admit several chains, but every block of a chain is a union of connectivity components of the graph. Thus, chains made of connectivity components cannot be refined.

Supposing C is a connectivity component of a CG G, the symbol $\mathscr{K}(C)$ will denote the class of complexes in G having their region in C.

2.2. Dependency Models and Markov Properties

Supposing N is a nonempty finite set of variables, let us denote by $T(N)$ the class of triplets $\langle X, Y \mid Z \rangle$ of disjoint subsets of N whose first two components X and Y are nonempty. These triplets will describe particular conditional (in)dependency statements. A *dependency model* over N is a decomposition of $T(N)$ into two parts, namely the *independence part* and the complementary *dependence part*. Let's write $I_M\langle X, Y \mid Z \rangle$ if a triplet $\langle X, Y \mid Z \rangle$ belongs to the independence part of a dependency model M, and otherwise write $D_M\langle X, Y \mid Z \rangle$. Conditional independence structures over N will be described by dependency models over N.

A *probability distribution* over N is specified by a collection of nonempty finite sets $\{\mathbf{X}_i;\ i \in N\}$ and by a function $P : \prod_{i \in N} \mathbf{X}_i \to [0, 1]$ with $\Sigma\{P(\mathbf{x});\ \mathbf{x} \in \prod_{i \in N} \mathbf{X}_i\} = 1$. If $P(\mathbf{x}) > 0$ for all $\mathbf{x} \in \prod_{i \in N} \mathbf{X}_i$, then P is called *strictly positive*. Whenever $\emptyset \neq X \subset N$ and P is a probability distribution over N, its *marginal distribution* on X is a probability distribution P^X (over X) defined as follows ($P^N \equiv P$):

$$P^X(\mathbf{x}) = \sum\left\{P(\mathbf{x},\mathbf{y});\, \mathbf{y} \in \prod_{i \in N \setminus X} \mathbf{X}_i\right\} \qquad \text{for} \quad \mathbf{x} \in \prod_{i \in X} \mathbf{X}_i.$$

Having $\langle X, Y \mid Z \rangle \in T(N)$ and a probability distribution P over N, we will say that X is *conditionally independent of Y given Z with respect to P* and write $X \perp\!\!\!\perp Y \mid Z\ (P)$ if

$$\forall \mathbf{x} \in \prod_{i \in X} \mathbf{X}_i, \mathbf{y} \in \prod_{i \in Y} \mathbf{X}_i, \mathbf{z} \in \prod_{i \in Z} \mathbf{X}_i,$$

$$P^{X \cup Y \cup Z}(\mathbf{x},\mathbf{y},\mathbf{z}) \cdot P^Z(\mathbf{z}) = P^{X \cup Z}(\mathbf{x},\mathbf{z}) \cdot P^{Y \cup Z}(\mathbf{y},\mathbf{z}),$$

where we accept the convention $P^{\varnothing}(-) \equiv 1$.

The dependency model *induced by* a probability distribution P over N has its independence part specified as the collection of all triples $\langle X, Y \mid Z \rangle \in T(N)$ such that $X \perp\!\!\!\perp Y \mid Z\ (P)$. Thus, the dependency model induced by a probability distribution P describes the probabilistic conditional independence structure of P.

Supposing G is a CG, its *moral graph* is obtained in two steps. First, the parents of every complex in G are joined by an edge. Second, the underlying graph of the resulting graph is taken. Frydenberg [7] gave another equivalent definition, namely to join the parents of every connectivity component of G which are not joined, and then to "forget" the orientations.

A triplet $\langle X, Y \mid Z \rangle \in T(N)$ is *represented* in a CG G according to the *moralization criterion* if every path in the moral graph of $G_{\mathrm{an}(X \cup Y \cup Z)}$ from a node of X to a node of Y contains a node of Z. Thus, the moralization criterion taken from [14, 7] has three steps: first, to take the induced subgraph of G on the corresponding ancestral set $\mathrm{an}_G(X \cup Y \cup Z)$; second, to find the moral graph of the induced subgraph; third, to apply the classical separation criterion for UGs to the moral graph. Note that in [3] we have introduced a direct separation criterion for CGs, which generalizes the concept of d-separation for DAGs from [17]. This *c-separation* criterion tests directly trails in the original CG whether they are blocked by a set of nodes. Nevertheless, we have proved in [18] (Consequence 4.1) that the moralization criterion and the separation criterion for CGs are equivalent.

The dependency model *induced by* a CG G has its independence part specified as the class of triplets represented in G according to the moralization criterion.

A probability distribution P over N is *Markovian* w.r.t. a CG G over N if every triplet from $T(N)$ represented in G belongs to the independence part of the dependency model induced by P. The use of CGs for description of probabilistic conditional independence structures is justified by the following result proved in [18, Theorem 7.2] by essential use of the c-separation criterion.

THEOREM 2.1 *For every* CG *G over N there exists a strictly positive probability distribution P over N such that the dependency models induced by G and P coincide.*

Two chain graphs G and H over N are *Markov equivalent* if their classes of Markovian distributions coincide. Frydenberg [7, Property 5.6] gave the following characterization of Markov equivalence which generalizes an analogous result for DAGs from [21].

THEOREM 2.2 *Two* CG*s are Markov equivalent iff they have the same underlying graph and complexes.*

Supposing G and H are CGs over the same set of variables with the same underlying graph, we say that G is *larger* than H if every arrow of G is an arrow in H with the same orientation. Note that Frydenberg [7] defined the relation "larger" for every pair of CGs, and I use only a restricted definition here. The following theorem reformulates a little bit a further result from [7, Proposition 5.7].

THEOREM 2.3 *For every* CG *G there exists a Markov equivalent* CG *G_∞ which is larger than every* CG *which is Markov equivalent to G.*

DEFINITION 2.2 *The graph G_∞ from the previous theorem is called the* largest CG *of the class of* CG*s which are Markov equivalent to G (or briefly, the largest* CG *corresponding to G).*

It is evident that the largest CG of every class of Markov equivalent CGs is uniquely determined.

3. THE FIRST STAGE: THE PATTERN

3.1. Description of the Pattern Recovery

The first step of the recovery algorithm is to obtain so-called pattern of the corresponding class of Markov equivalent CGs on the basis of the dependency model induced by a CG.

DEFINITION 3.1 *An arrow in a* CG *G is called a* complex arrow *in G if it belongs to a complex in G. The* pattern *of the class of* CG*s which are Markov equivalent to G (or briefly the pattern corresponding to G), denoted by G_0, is a hybrid graph obtained from the underlying graph of G by directing all edges which are complex arrows in G (with the same orientation).*

On can derive from Theorem 2.2 that two CGs are Markov equivalent iff they have the same pattern. However, the pattern may not be a CG, as the following example shows.

EXAMPLE 3.1 To illustrate the concept of pattern, let us consider the DAG G in Figure 1(i). It has only one complex: $a \to d \leftarrow c$. The corresponding pattern is in Figure 1(ii).

To reconstruct the pattern corresponding to a CG from its induced dependency model the following notation is suitable.

DEFINITION 3.2 *Let M be a dependency model over N, and $u, v, w \in N$ be distinct. The symbol $D_M\langle u, v \mid -\rangle$ will be used to replace an entire collection of statements, namely $D_M\langle u, v \mid Z\rangle$ for all $Z \subset N \setminus \{u, v\}$. Similarly, $D_M\langle u, v \mid +w\rangle$ will replace $D_M\langle u, v \mid Z\rangle$ for all $Z \subset N \setminus \{u, v\}$ with $w \in Z$.*

The algorithm presented here produces a sequence of hybrid graphs $H_0, H_1, \ldots$ with the same underlying graph as G, such that H_i has all complexes in G of degree at most i.

PATTERN RECOVERY ALGORITHM Let M be the dependency model induced by an (unknown) CG G over N.

(i) The starting iteration is an undirected graph H_0 over N defined by the following rule: $u — v$ in H_0 iff $D_M\langle u, v \mid -\rangle$.

(ii) For $l = 1, \ldots, \text{card}\, N - 2$ the iteration H_l is made from H_{l-1} by possible directing of some lines of H_{l-1}. Namely, in every situation when some sequence of distinct nodes $w_1, \ldots, w_{l+2}$ exists such that

- $w_1 \to w_2$ or $w_1 — w_2$ in H_{l-1},
- $w_{l+1} \leftarrow w_{l+2}$ or $w_{l+1} — w_{l+2}$ in H_{l-1},
- $w_i — w_{i+1}$ in H_{l-1} for $i = 2, \ldots, l$,
- no other edge exists in H_{l-1} among $\{w_1, \ldots, w_{l+2}\}$,
- $D_M\langle w_1, w_{l+2} \mid +w_2\rangle$, and
- $D_M\langle w_1, w_{l+2} \mid +w_{l+1}\rangle$,

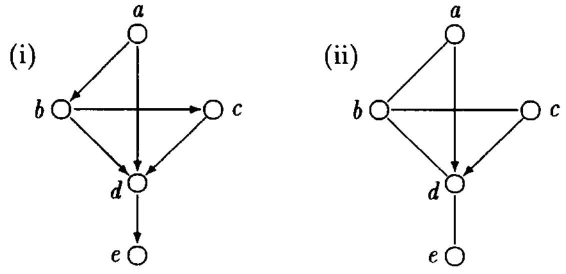

Figure 1. Example of a DAG and its pattern.

one has $w_1 \to w_2$ and $w_{l+1} \leftarrow w_{l+2}$ in H_l (note that these edges may possibly be directed already in H_{l-1}). All other edges of H_l keep their type and orientation from H_{l-1}.

The following result justifies the algorithm; its proof is the topic of the rest of this section.

THEOREM 3.1 *The last iteration of the pattern recovery algorithm is just the pattern corresponding to the considered (unknown)* CG G.

3.2 Several Lemmas

LEMMA 3.1 *Let G be a* CG *over N; M the dependency model induced by G; and $u, v \in N$ distinct such that $\{u, v\}$ is not an edge in G. Then* $I_M\langle u, v \mid \mathrm{bd}_G(u) \cup \mathrm{bd}_G(v)\rangle$.

Proof Let us apply the moralization criterion to $\langle u, v \mid T\rangle$, where $T = \mathrm{bd}_G(u) \cup \mathrm{bd}_G(v)$. Evidently $\mathrm{an}_G(\{u, v\} \cup T) = \mathrm{an}_G(\{u, v\})$ and one should consider the induced subgraph $H = G_{\mathrm{an}(\{u,v\})}$. Let us verify by contradiction that either $\mathrm{ch}_H(u)$ or $\mathrm{ch}_H(v)$ is empty. Indeed, if $u \to t$ in H for some t, then owing to $t \in \mathrm{an}_G(\{u, v\})$ there exists a descending path in G from t to $\{u, v\}$. It has to lead to v, as otherwise a directed cycle in G exists—see Lemma 2.1(iii). Similarly, $v \to s$ in H for some s implies that there is a descending path in G from s to u. Thus, if both $u \to t$ and $v \to s$, then $u \to t \cdots v \to s \cdots u$ is a directed pseudocycle in G, which contradicts Lemma 2.1. Therefore, one can suppose without loss of generality that $\mathrm{ch}_H(u) = \emptyset$. This implies that no edge leading to u is added when the moral graph of H is made, and u has $\mathrm{bd}_G(u)$ as the set of its neighbors in the moral graph. Thus, T interrupts every path between u and v in the moral graph of H. ■

LEMMA 3.2 *Let G be a* CG *over N; M the dependency model induced by G; and $u, v \in N$ distinct. Then $\{u, v\}$ is an edge in G iff $D_M\langle u, v \mid -\rangle$.*

Proof Supposing $\{u, v\}$ is an edge in G, for every $Z \subset N \setminus \{u, v\}$ the edge $\{u, v\}$ occurs in the moral graph of $G_{\mathrm{an}(\{u,v\} \cup Z)}$ and $\langle u, v \mid Z\rangle$ is not represented in G according to the moralization criterion. Thus, the necessity of the condition $D_M\langle u, v \mid -\rangle$ is verified; the sufficiency follows directly from Lemma 3.1. ■

LEMMA 3.3 *Let G be a* CG *over N, and M the dependency model induced by G. Suppose that $u, v, w \in N$ are distinct nodes of N such that $\{u, w\}$, $\{v, w\}$ are edges in G and $\{u, v\}$ is not an edge in G. Then $u \to w \leftarrow v$ is a complex in G iff $D_M\langle u, v \mid +w\rangle$.*

Proof Supposing $u \to w \leftarrow v$ in G and $w \in Z \subset N \setminus \{u, v\}$, the moral graph of $G_{\mathrm{an}(\{u,v\} \cup Z)}$ contains the edge $\{u, v\}$. Hence, necessarily

$D_M\langle u, v \mid +w\rangle$. For sufficiency one can suppose by contradiction that the induced subgraph on $\{u, w, v\}$ is not a complex in G. Then $w \in \mathrm{bd}_G(u) \cup \mathrm{bd}_G(v)$, and Lemma 3.1 leads to contradiction. ■

LEMMA 3.4 *Let G be a CG over N, and M the dependency model induced by G. Suppose that $w_1, \ldots, w_k$, $k \geq 4$, is a sequence of distinct nodes where*

- *$\{w_i, w_{i+1}\}$ is an edge in G for $i = 1, \ldots, k-1$,*
- *no other edge exists in G among $\{w_1, \ldots, w_k\}$,*
- *$w_i, w_{i+1}, \ldots, w_j$ is not a complex in G whenever $1 \leq i \leq j \leq k$, $2 \leq j - i < k - 1$.*

Then $w_1 \to w_2 — \cdots — w_{k-1} \leftarrow w_k$ is a complex in G iff

$$[D_M\langle w_1, w_k \mid +w_2\rangle \,\&\, D_M\langle w_1, w_k \mid +w_{k-1}\rangle].$$

Proof The necessity of $D_M\langle w_1, w_k \mid +w_2\rangle$: if $w_2 \in Z \subset N \setminus \{w_1, w_k\}$, then the complex $w_1 \to w_2 — \cdots — w_{k-1} \leftarrow w_k$ occurs in the induced subgraph of G on $\mathrm{an}_G(\{w_1, w_k\} \cup Z)$ and the edge $\{w_1, w_k\}$ occurs in the corresponding moral graph. The necessity of $D_M\langle w_1, w_k \mid +w_{k-1}\rangle$ is analogous.

To show the sufficiency of the conditions let us verify first by contradiction that $w_1 \to w_2$ in G. Indeed, otherwise $w_2 \in \mathrm{bd}_G(w_1)$ and one can use Lemma 3.1 for $u = w_1$, $v = w_k$ to get contradiction with $D_M\langle w_1, w_k \mid +w_2\rangle$. Similarly, $D_M\langle w_1, w_k \mid +w_{k-1}\rangle$ implies $w_{k-1} \leftarrow w_k$ in G. Supposing that $w_{j-1} \leftarrow w_j$ in G for some $3 \leq j \leq k-1$ let us consider the minimal such j and find maximal $1 \leq i \leq j-2$ with $w_i \to w_{i+1}$ in G. Then $w_i \to w_{i+1} — \cdots — w_{j-1} \leftarrow w_j$ is a complex in G what contradicts the assumption. Thus, no such j exists. Similarly, no $2 \leq i \leq k-2$ with $w_i \to w_{i+1}$ in G exists and $w_r — w_{r+1}$ for $r = 2, \ldots, k-1$. ■

A general sufficient condition for existence of a complex arrow, which will be utilized in sequel, is given by the following lemma.

LEMMA 3.5 *Let L be a CG over N, and $w_1, \ldots, w_k$, $k \geq 3$, a sequence of nodes in L (possibly not distinct) such that*

(i) $w_1 \neq w_k$,
(ii) $w_1 \to w_2$ *and* $w_{k-1} \leftarrow w_k$ *in* L,
(iii) *for all* $i = 2, \ldots, k-2$ *either* $w_i \to w_{i+1}$ *or* $w_i — w_{i+1}$ *in* L,
(iv) *there is no edge in* L *between* w_k *and* $\{w_i;\ 1 \leq i \leq k-2\}$.

Then there exists a complex in L containing $w_k \to w_{k-1}$ composed of nodes $\{w_i;\ 1 \leq i \leq k\}$.

Proof Conditions (i)–(iv) imply that w_k is distinct from the other nodes. Similarly, (i) and (ii), (iii) imply that w_1 is distinct from the other nodes (otherwise a directed pseudocycle in L exists). The route $w_1, \ldots, w_k$

can be shortened to consist of distinct nodes by removing those of its parts between different occurrences of the same node [conditions (i)–(iv) are preserved by that change].

Further possible modifications of $w_1, \ldots, w_k$ will ensure that for every $i = 1, \ldots, k-2$ there is no edge in L between w_i and $\{w_r;\ i+2 \le r \le k\}$. Indeed, as concerns w_1, take maximal $2 \le l \le k$ such that $\{w_1, w_l\}$ is an edge in L. By (iv) $l < k$, and by (ii) and (iii) (as L is a CG) $w_1 \to w_l$ in L. Thus, the path $w_1, \ldots, w_k$ can be shortened by possibly replacing the section $w_1, \ldots, w_l$ by this single arrow $w_1 \to w_l$ and by a natural change of notation ($\tilde{w}_1 = w_1$ and $\tilde{w}_i = w_{l-2+i}$ for $i = 2, \ldots, \tilde{k} = k - l + 2$). This ensures that conditions (i)–(iv) will be preserved. Then, an analogous consideration can be made for w_2 (already in the modified path) with the only difference that one has either $w_2 \to w_l$ or $w_2 - w_l$ in L for the corresponding w_l, $3 \le l \le k$. The result of the series of modifications is a path $w_1, \ldots, w_k$, $k \ge 3$, satisfying (ii)–(iii) such that no other edges among $\{w_1, \ldots, w_k\}$ in L exist. Take the maximal $1 \le s \le k-2$ with $w_s \to w_{s+1}$. Then $w_s \to w_{s+1} - \cdots - w_{k-1} \leftarrow w_k$ is a complex in L. ■

3.3. Proof of Correctness of Pattern Recovery

In this subsection the proof of Theorem 3.1 is given.

By Lemma 3.2 H_0 has the same underlying graph as G, and hence every H_i has the same underlying graph as G. Let us verify by induction on $l = 1, \ldots, \mathrm{card}\, N - 2$ the following two conditions:

(a) $u \to w$ in H_l implies $u \to w$ in G_0;

(b) every complex in G of degree at most l is also a complex in H_l.

To verify (a) for H_1, realize that $u \to w$ in H_1 implies the existence of a third node v with $u - w - v$ in H_0, $\neg(u - v)$ in H_0, and $D_M\langle u, v \mid +w\rangle$. Hence $\{u, w\}$, $\{v, w\}$ are edges in G while $\{u, v\}$ is not an edge in G, and by Lemma 3.3 (sufficiency) $u \to w \leftarrow v$ is a complex in G, which says $u \to w$ in G_0.

To verify (b) for H_1 suppose that $u \to w \leftarrow v$ is a complex in G and by Lemma 3.3 (necessity) derive $D_M\langle u, v \mid +w\rangle$. Moreover, evidently $u - w - v$ in H_0, $\{u, v\}$ is not an edge in H_0, and thus, by the construction of H_1, $u \to w \leftarrow v$ is a complex in H_1.

Supposing (a), (b) hold for H_{l-1}, $l \ge 2$, let us verify (a) for H_l. If $u \to w$ in H_l, then either $u \to w$ in H_{l-1} and one can use the induction assumption, or, by the construction of H_l, $u - w$ in H_{l-1} and there exists a sequence of nodes $w_1, \ldots, w_{l+2}$, where $w_1 = u$, $w_2 = w$, such that the collection of conditions from item (ii) of the algorithm is satisfied. As H_{l-1} has the same underlying graph as G, the first two assumptions of Lemma 3.4 for $k = l + 2$ are fulfilled. The third assumption of Lemma 3.4 then follows from condition (b) for H_{l-1}. Thus, by Lemma 3.4 (sufficiency),

$w_1 \to w_2 — \cdots — w_{l+1} \leftarrow w_{l+2}$ is a complex in G, which implies $u = w_1 \to w_2 = w$ in G_0.

Supposing (a), (b) hold for H_{l-1}, $l \geq 2$, and (a) holds for H_l, let us verify (b) for H_l. Let $w_1 \to w_2 — \cdots — w_{s+1} \leftarrow w_{s+2}$, $s \leq l$, be a complex in G.

The first observation is that $w_r — w_{r+1}$ in H_l for $r = 2, \ldots, s$. Indeed, one can suppose by contradiction, for instance, $w_r \leftarrow w_{r+1}$ in H_l for some $2 \leq r \leq s$; then by (a) for H_l, $w_r \leftarrow w_{r+1}$ in G_0 and therefore in G, which contradicts the assumption. Similar contradiction can be obtained if $w_r \to w_{r+1}$ in H_l for some $2 \leq r \leq s$.

Second, if $1 \leq s < l$, then $w_1 \to w_2 — \cdots — w_{s+1} \leftarrow w_{s+2}$ is a complex in H_{l-1} by (b) for H_{l-1}, which is saved in H_l by the previous observation.

Third, if $s = l$, then one derives by using (a) for H_{l-1} that $w_r — w_{r+1}$ in H_{l-1} for $r = 2, \ldots, l$, $w_1 \to w_2$ or $w_1 — w_2$ in H_{l-1}, and $w_{l+1} \leftarrow w_{l+2}$ or $w_{l+1} — w_{l+2}$ in H_{l-1}. Evidently no other edge exists in H_{l-1} among $\{w_1, \ldots, w_{l+2}\}$, and Lemma 3.4 (necessity) says $D_M\langle w_1, w_{l+2} \mid +w_2\rangle$ and $D_M\langle w_1, w_{l+2} \mid +w_{l+1}\rangle$. In short, the collection of conditions from the item (ii) of the pattern recovery algorithm is satisfied, and, by the construction of H_l, $w_1 \to w_2$ and $w_{l+1} \leftarrow w_{l+2}$ in H_l, and hence, by the first-mentioned observation, $w_1 \to w_2 — \cdots — w_{l+1} \leftarrow w_{l+2}$ is a complex in H_l.

Thus, the last iteration H_* of the algorithm has the same underlying graph as G, and, by (a) for H_*, $u \to v$ in H_* implies $u \to v$ in G_0. Conversely, the fact $u \to v$ in G_0 implies that there exists a complex in G which contains $u \to v$, which implies by (b) for H_* that there is a complex in H_* containing $u \to v$ and therefore $u \to v$ in H_*. Thus, $u \to v$ in H_* iff $u \to v$ in G_0 and hence $H_* = G_0$. ■

4. THE SECOND STAGE: LARGEST CHAIN GRAPH

4.1. Description of the LCG Recovery Algorithm

The pattern G_0 of the considered class of Markov equivalent CGs, obtained in the preceding section, should be changed into the corresponding largest CG G_∞. This is done by the *largest chain graph recovery algorithm*, or LCG recovery algorithm for short.

4.1.1. EXTENDED HYBRID GRAPH Iterations of the presented algorithm are not mere hybrid graphs, but *extended hybrid graphs*, some lines of which have "forbidden" potential orientations (by which we understand orientations in future iterations and therefore in G_∞). Let us write $\neg\{u \leftarrow v\}_\infty$ to denote that a line $u — v$ has the orientation $u \leftarrow v$ forbidden. In pictures, the fact $\neg\{u \leftarrow v\}_\infty$ will be depicted by a short thick perpendicular line crossing the line $u — v$ near the node u. A descending route $w_1, \ldots, w_k$,

$k \geq 2$, in such an extended hybrid graph G will be called *steady* if $\neg\{w_i \leftarrow w_{i+1}\}_\infty$ in G whenever $w_i — w_{i+1}$ in G for all $i = 1, \ldots, k-1$.

DEFINITION 4.1 *Let H be an extended hybrid graph over N, and K be a hybrid graph over N. We say that H is a* map *of K if the following conditions hold*:

(a) *H and K have the same underlying graph*;
(b) *whenever $u \leftarrow v$ in H, then $u \leftarrow v$ in K*;
(c) *H has the same complexes as K*;
(d) *whenever $\neg\{u \leftarrow v\}_\infty$ in H, then either $u \rightarrow v$ in K or $u — v$ in K.*

Note that it may happen in an extended hybrid graph that one line has both potential orientations forbidden, that is, one has both $\neg\{u \leftarrow v\}_\infty$ and $\neg\{v \leftarrow u\}_\infty$ for a line $u — v$.

The starting iteration of the algorithm will be the pattern G_0.

LEMMA 4.1 *Supposing G is a* CG *over N, the pattern G_0 is a map of G_∞.*

Proof We are to verify (a)–(d) from Definition 4.1 for $H = G_0$ and $K = G_\infty$. Owing to Theorem 2.2, the graphs G and G_∞ have the same underlying graph and complexes. Thus, both (a) and (b) are evident from the definition of G_0 (see Definition 3.1).

As concerns (c), to show that every complex in G_∞ (= in G) is a complex in G_0 it suffices to realize that every line of a complex in G remains a line in G_0. Conversely, let us consider a complex $v_1 \rightarrow v_2 — \cdots — v_{k-1} \leftarrow v_k$, $k \geq 3$, in G_0. Then by (b) $v_1 \rightarrow v_2$ and $v_{k-1} \leftarrow v_k$ in G_∞, and owing to (a) it remains to verify that $v_i — v_{i+1}$ in G_∞ for $i = 2, \ldots, k-2$. Suppose by contradiction that $v_i \leftarrow v_{i+1}$ in G_∞ for such i, take minimal such $i \in \{2, \ldots, k-2\}$, and apply Lemma 3.5 to $L = G_\infty$ and $v_1, \ldots, v_{i+1}$ to derive that $v_i \leftarrow v_{i+1}$ is a complex arrow in G_∞ and therefore in G_0 (as we have already mentioned). That contradicts the assumption that $v_i — v_{i+1}$ in G_0. An analogous contradiction can be derived in case $v_i \rightarrow v_{i+1}$ in G_∞ for some $i \in \{2, \ldots, k-2\}$. Thus, $v_1 \rightarrow v_2 — \cdots — v_{k-1} \leftarrow v_k$ is a complex in G_∞, and (c) is verified.

Condition (d) is empty for G_0, as it has no "forbidden" orientations. ■

4.1.2. BASIC RULES OF THE LCG RECOVERY ALGORITHM Further possible iterations G_m, $m \geq 1$, of the LCG recovery algorithm will be obtained from previous ones by application of three special rules. Each rule makes a single change and leaves the rest of the extended graph untouched. The transitivity rule just forbids potential orientations of lines; the necessity and the double-cycle rule change lines into arrows. Let us describe the changes made by the rules.

TRANSITIVITY RULE (See Figure 2) Suppose that $w_1 \rightarrow w_2 — \cdots — w_k$, $k \geq 3$, is a slide in an iteration G_m ($m \geq 0$), such that no other edge exists

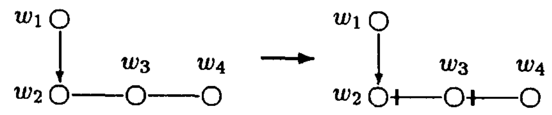

Figure 2. Transitivity rule.

in G_m among $\{w_1,\ldots,w_k\}$. Then the transitivity rule changes that slide in G_m into a steady slide in the next iteration G_{m+1}. Equivalently, one derives $\neg\{w_i \leftarrow w_{i+1}\}_\infty$ in G_{m+1} for all $i = 2,\ldots,k-1$.

Of course, the transitivity rule need not be applied if the corresponding slide is already steady (as it would not make any change).

LEMMA 4.2 (Soundness of the transitivity rule) *The application of the transitivity rule to a map of a* LCG G_∞ *yields a map of* G_∞.

Proof One has to verify that the resulting graph G_{m+1} satisfies the conditions in Definition 4.1. Conditions (a)–(c) are evident. To verify condition (d) one has to show that there is no $2 \le i \le k-1$ with $w_i \leftarrow w_{i+1}$ in G_∞. However, in such a case one takes minimal such i and applies Lemma 3.5 to $L = G_\infty$ and $w_1,\ldots,w_{i+1}$ to derive that $w_i \leftarrow w_{i+1}$ is a complex arrow in G_∞. Hence, $w_i \leftarrow w_{i+1}$ in G_m by the condition (c) for G_m, which contradicts the assumption. ■

NECESSITY RULE (See Figures 3 and 4) Suppose that $r_0,\ldots,r_k,r_0$, $k \ge 2$, is a chordless cycle in an iteration G_m ($m \ge 0$), such that $r_k — r_0$ in G_m and

either $r_0 \to r_1 — \cdots — r_k$ is a steady slide in G_m (that is the first variant)

or $r_0 \to r_1$ in G_m and $r_1 \to r_2 — \cdots — r_k$ is a steady slide in G_m (that is the second variant).

Then the necessity rule changes the line $r_k — r_0$ in G_m into the arrow $r_k \leftarrow r_0$ in the next iteration G_{m+1}.

LEMMA 4.3 (Soundness of the necessity rule) *The application of the necessity rule to a map of a* LCG G_∞ *yields a map of* G_∞.

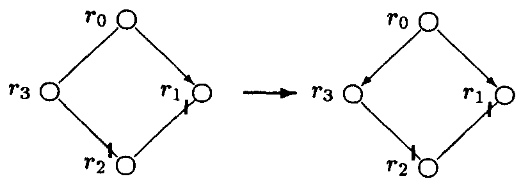

Figure 3. Necessity rule—the first variant.

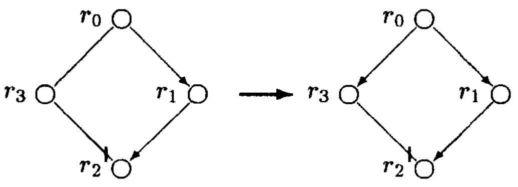

Figure 4. Necessity rule—the second variant.

Proof Condition (a) from Definition 4.1 for G_{m+1} is evident. To verify (b) let us suppose by contradiction that either $r_k \rightarrow r_0$ or $r_k — r_0$ in G_∞. Then the assumption that G_m is a map of G_∞ implies that $r_0, \ldots, r_k, r_0$ is a directed cycle in G_∞, which contradicts the assumption that G_∞ is a CG. Thus, $r_k \leftarrow r_0$ in G_∞.

The edge $\{r_0, r_k\}$ does not belong to a complex in G_∞. Indeed, if $r_k \leftarrow r_0$ is a complex arrow in G_∞, then by (c) it is a complex arrow in G_m, which contradicts the assumption.

The edge $\{r_0, r_k\}$ does not belong to a complex in G_{m+1}. Suppose by contradiction there exists a complex $w_1 \rightarrow w_2 — \cdots — w_{l-1} = r_k \leftarrow r_0 = w_l$, $l \geq 3$, in G_{m+1}. Thus, $w_1 \rightarrow w_2 — \cdots — w_{l-1} — w_l$ in G_m. As G_m is a map of G_∞ and $w_{l-1} \leftarrow w_l$ in G_∞, one can take minimal $2 \leq s \leq l-1$ with $w_s \leftarrow w_{s+1}$ in G_∞ and apply Lemma 3.5 to $L = G_\infty$ and $w_1, \ldots, w_{s+1}$ to derive that $w_s \leftarrow w_{s+1}$ is a complex arrow in G_∞, and therefore $w_s \leftarrow w_{s+1}$ in G_m, which contradicts the fact above.

Thus, condition (c) from Definition 4.1 is preserved in G_{m+1}. Condition (d) is evident. ■

DOUBLE-CYCLE RULE (See Figure 5) Suppose that $r_0, \ldots, r_k, r_0$, $k \geq 2$, is a chordless cycle in an iteration G_m ($m \geq 0$), such that $r_0 \rightarrow r_1 — \cdots — r_{k-1}$ is a steady slide in G_m and $r_0 — r_k$, $r_k — r_{k-1}$ in G_m. Moreover, suppose that $s_0 \rightarrow s_1 — \cdots — s_l = r_1$, $l \geq 1$, is a steady slide in G_m such that $s_0 \neq r_0$, and $\{r_k, s_0\}$ is an edge in G_m but $\{r_0, s_0\}$ is not an

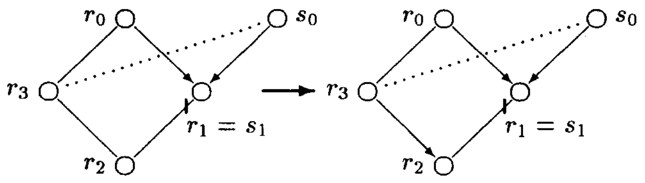

Figure 5. Double-cycle rule.

edge in G_m. Then the double-cycle rule changes the line $r_{k-1} - r_k$ in G_m into the arrow $r_{k-1} \leftarrow r_k$ in the next iteration G_{m+1}.

LEMMA 4.4 (Soundness of the double-cycle rule) *The application of the double-cycle rule to a map of* LCG G_∞ *yields a map of* G_∞.

Proof Condition (a) from Definition 4.1 for G_{m+1} is evident. To verify (b) let us consider by contradiction that either $r_{k-1} \rightarrow r_k$ or $r_{k-1} - r_k$ in G_∞. Then necessarily $r_k \leftarrow r_0$ in G_∞, as otherwise $r_0, \ldots, r_k, r_0$ is a directed cycle in G_∞. Similarly, $r_k \leftarrow s_0$ in G_∞, as otherwise $s_0, \ldots, s_l = r_1, \ldots, r_k, s_0$ is a directed pseudocycle in G_∞ (see Lemma 2.1). But then $r_0 \rightarrow r_k \leftarrow s_0$ is a complex in G_∞, and condition (c) for G_m implies that $r_k \leftarrow r_0$ in G_m, which contradicts the assumption.

To verify (c) for G_{m+1} it suffices to show that the edge $\{r_{k-1}, r_k\}$ does not belong to a complex in G_∞ or in G_{m+1}. This can be done by the same procedure as in the proof of Lemma 4.3 (where the edge $\{r_0, r_k\}$ was treated).

Condition (d) is evident. ■

4.1.3. LCG RECOVERY ALGORITHM The starting iteration of the algorithm is the pattern G_0 of a CG over N. Then the rules described in the preceding subsubsection are applied to produce further iterations G_m, $m \geq 0$, of the algorithm. The transitivity rule has the highest priority, then the necessity rule follows, and the double-cycle rule has the lowest priority.

That means that, having iteration G_m, $m \geq 0$, one first tries to apply the transitivity rule. It does not matter which slide satisfying the assumption of the transitivity rule is chosen. In case the transitivity rule cannot be applied to G_m (that means all apposite slides are already steady in G_m), one tries to apply the necessity rule to G_m. Again, it does not matter which cycle satisfying the assumptions of the necessity rule is chosen or which variant of the rule is considered. In case also the necessity rule cannot be applied to G_m (that means no apposite cycle exists in G_m), one tries to apply the double-cycle rule. It does not matter which cycle and slide satisfying the assumptions of the double-cycle rule are considered. If also the double-cycle rule cannot be applied, the algorithm stops and G_m will be the last iteration.

However, if one succeeds in applying one of the rules as described above, a new iteration G_{m+1} is obtained as a result (necessarily G_{m+1} differs from G_m). Then one tries to apply to G_{m+1} the same procedure as to G_m (that is, one starts by trying to apply the transitivity rule, etc.).

THEOREM 4.1 *Supposing G_0 is the pattern corresponding to a* CG *G over N, the last iteration of the* LCG *recovery algorithm is the largest* CG *corresponding to G.*

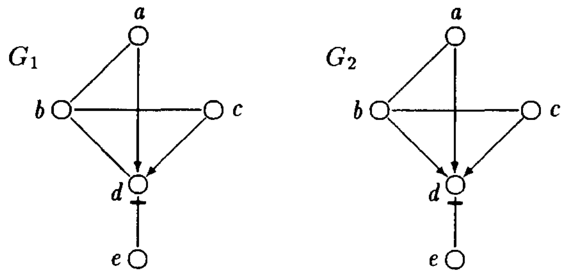

Figure 6. Iterations of the LCG recovery algorithm.

Of course, the last iteration of the LCG recovery algorithm is an extended graph, and we formally change it into an ordinary hybrid graph by ignoring information given by forbidden potential orientations of its lines.

The proof of Theorem 4.1 is given in the following subsection. Let us conclude this subsection with an example.

EXAMPLE 4.1 Let us consider the DAG from Figure 1(i). The largest CG of the corresponding class of Markov equivalent CGs is in Figure 7(i). It can be obtained from the pattern in Figure 1(ii) by application of the LCG recovery algorithm. First, the transitivity rule for $K = 3$, $w_1 = a$, $w_2 = d$, $w_3 = e$ (or alternatively $w_1 = c$) derives $\neg\{d \leftarrow e\}_\infty$ in G_1—see Figure 6, the left picture. However, neither the transitivity rule nor the necessity rule can be applied to G_1. But one can use the double-cycle rule where $k = 2, l = 1, r_0 = a, r_1 = s_1 = d, r_2 = b, s_0 = c$ to derive $b \rightarrow d$ in G_2—see Figure 6, the right picture (alternatively $r_0 = c$, $s_0 = a$). As no rule can be applied to G_2, it is the last iteration of the LCG recovery algorithm. Thus, the LCG of the mentioned class of Markov equivalent CGs is given by Figure 7(i), already without "forbidden" orientations of lines. The corresponding essential graph, called the "completed pattern" in [21], is given in

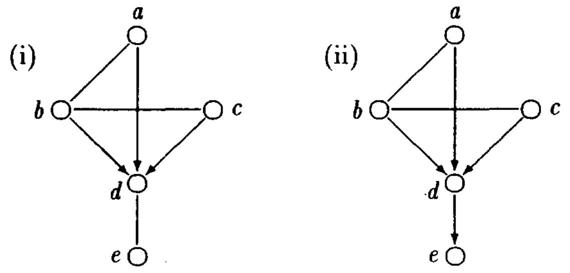

Figure 7. The largest chain graph versus the essential graph.

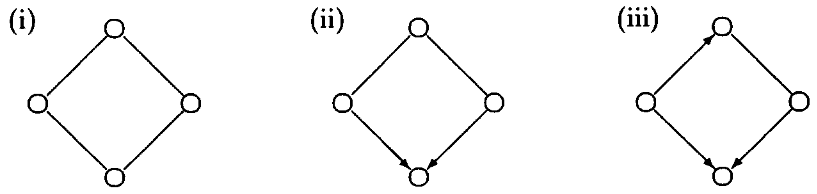
Figure 8. Cordless cycles—possibilities (A) and (B).

the Figure 7(ii). So the LCG and the essential graph corresponding to a DAG may differ.

4.2. Convergence of the Algorithm

The following lemma contains an analysis of how a chordless cycle in a CG looks.

LEMMA 4.5 *Suppose G is a CG and α is a chordless cycle in G. Then just one of the following possibilities occurs*:

(A) *α is an undirected cycle in G—see the illustrative Figure* 8(i);

(B) *α contains a complex in G (and possibly other arrows)—see Figure* 8(ii)–(iii);

(C) *α has the form $r_0, \ldots, r_k, r_0$, $k \geq 2$, where $r_0 \to r_1$, $r_0 \to r_k$, and the remaining edges of α are lines—see Figure* 9(i);

(D) *α has the form $r_0, \ldots, r_k, r_0$, $k \geq 2$, where $r_0 \to r_1$, $r_k \to r_{k-1}$, and the remaining edges of α except $\{r_0, r_k\}$ are lines—see Figure* 9(ii)–(iii), *which illustrate two possible subcases..*

Proof In case α contains no arrows in G, condition (A) holds.

In case $\alpha : r_0, \ldots, r_k, r_0$, $k \geq 2$, has an arrow in G, one can suppose without loss of generality that $r_0 \to r_1$, and by Lemma 2.1(iv) $r_s \leftarrow r_{s+1}$ for some $1 \leq s \leq k$ (with the convention $r_{k+1} = r_0$). Take minimal such s. Moreover, one can suppose that $r_i — r_{i+1}$ for all $i = 1, \ldots, s-1$. Indeed, in case $r_i \to r_{i+1}$ for some $1 \leq i \leq s-1$ (the orientation $r_i \leftarrow r_{i+1}$ is

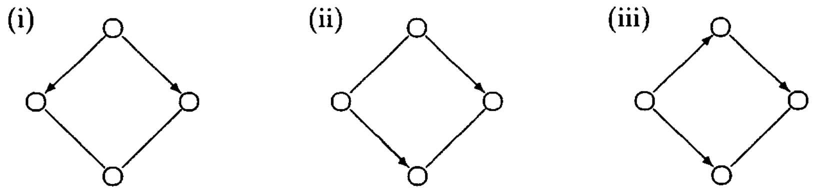
Figure 9. Cordless cycles—possibilities (C) and (D).

excluded owing to the definition of s), take maximal such i and change the notation (= indexing) of nodes of α into $\tilde{r}_0, \ldots, \tilde{r}_k, \tilde{r}_0$, where $\tilde{r}_j = r_t$ with $t = i + j \bmod k + 1$. The reindexed cycle then has the desired property.

Thus, if $s \leq k - 2$ (after the modification), then $r_0 \rightarrow r_1 - \cdots - r_s \leftarrow r_{s+1}$ is a complex in G and condition (B) holds. In case $s = k - 1$, case (D) occurs, while $s = k$ leads to condition (C). ■

LEMMA 4.6 *Let G be a* CG *over N, and C be a connectivity component of G such that the set $A = \mathrm{pa}_G(C) \cap \bigcap_{\kappa \in \mathscr{K}(C)} \bigcap_{v \in \mathrm{par}(\kappa)} \mathrm{ch}_G(v)$ is nonempty. Let B be a terminal connectivity component of the induced graph G_A. Let us make from G a hybrid graph H by changing all arrows from B to C into lines. Then H is a* CG *over N which is Markov equivalent to G.*

Note that a natural convention $\bigcap_{\kappa \in \varnothing} \bigcap_{v \in \mathrm{par}(\kappa)} \mathrm{ch}_G(v) = N$ is accepted in case $\mathscr{K}(C) = \varnothing$. Therefore in case $\mathscr{K}(C) = \varnothing$ the set A is nothing but $\mathrm{pa}_G(C)$.

Proof Evidently H and G have the same underlying graph.

Let $\kappa : w_1 \rightarrow w_2 - \cdots - w_{k-1} \leftarrow w_k$, $k \geq 3$, be a complex in G. To show that κ is a complex in H it suffices to show that $w_1 \rightarrow w_2$ and $w_{k-1} \leftarrow w_k$ in H. That is evident in case $\kappa \notin \mathscr{K}(C)$, because then $w_2, w_{k-1} \notin C$. However, in case $\kappa \in \mathscr{K}(C)$ one has $w_1, w_k \notin \mathrm{ch}_G(w_1) \cap \mathrm{ch}_G(w_k) = \bigcap_{v \in \mathrm{par}(\kappa)} \mathrm{ch}_G(v)$. Thus, $w_1, w_k \notin A$, which implies $w_1, w_k \notin B$, and the arrows $w_1 \rightarrow w_2$ and $w_{k-1} \leftarrow w_k$ will be saved in H.

Let $\lambda : v_1 \rightarrow v_2 - \cdots - v_{l-1} \leftarrow v_l$, $l \geq 3$, be a complex in H. Then necessarily $v_1 \rightarrow v_2$ and $v_{l-1} \leftarrow v_l$ in G, and to show that λ is a complex in G it suffices to verify that $v_i - v_{i+1}$ in G for $i = 2, \ldots, l-2$. For example, if $v_i \leftarrow v_{i+1}$ in G, take minimal such $i \in \{2, \ldots, l-2\}$ and apply Lemma 3.5 for $L = G$ and $v_1, \ldots, v_{i+1}$ to derive that $v_i \leftarrow v_{i+1}$ belongs to a complex in G. It was already verified that every complex in G is a complex in H. This implies that $v_i \leftarrow v_{i+1}$ in H, which contradicts the assumption. Similarly, the case $v_i \rightarrow v_{i+1}$ in G for $i \in \{2, \ldots, l-2\}$ can be excluded. Thus, H and G share complexes.

To show that H is a CG, according to Lemma 2.1(iv) it suffices to verify that no chordless cycle in H is directed. We already know that G and H share chordless cycles. Thus, let us consider a chordless cycle α in G and distinguish the cases mentioned in Lemma 4.5. We are to show that α is not a directed cycle in H.

Supposing (A) holds, α is an undirected cycle in G which is saved in H.

Supposing (B) holds, a complex in G contained in α remains in H, and again α is not a directed cycle.

In case (C), either both the arrows $r_0 \rightarrow r_1$ and $r_0 \rightarrow r_k$ are changed into lines in H (in case $r_0 \in B$ and $r_1, r_k \in C$) and α is an undirected cycle in H, or they both are saved in H (realize that $r_1 \in C$ iff $r_k \in C$). In both subcases α is not a directed cycle in H.

In case (D), one can suppose without loss of generality that either $r_0 \to r_k$ or $r_0 — r_k$ in G (otherwise one can "interchange" r_0 and r_k by reindexing α into $\tilde{r}_0, \dots, \tilde{r}_k, \tilde{r}_0$, where $\tilde{r}_j = r_t$ with $t = -j - 1 \bmod k + 1$). Let us start with the most complicated subcase: $r_1, r_{k-1} \in C$ and $r_0 \in B$. Then we show that α is an undirected cycle in H.

The first step is to verify that every complex in $\mathscr{K}(C)$ has both parents joined with r_k by an edge in G. Suppose by contradiction that there exists $\kappa \in \mathscr{K}(C)$ and $w \in \mathrm{par}(\kappa)$ such that $\{r_k, w\}$ is not an edge of G. Then one also finds $\lambda \in \mathscr{K}(C)$, with $r_k \in \mathrm{par}(\lambda)$. Indeed, in case $r_k \neq w$ take a path in G through $C \cup \{w\}$ connecting w and r_{k-1} which cannot be shortened —it must have the form $w = w_1 \to w_2 — \cdots — w_l = r_{k-1}$, $l \geq 2$, where $w_2, \dots, w_l \in C$ and no additional edges among $\{w_1, \dots, w_l\}$ exist in G. Then take minimal s such that $\{r_k, w_s\}$ is an edge in G. Necessarily, $s \geq 2$, $r_k \to w_s$ in G, and $\lambda : r_k \to w_s — \cdots — w_2 \leftarrow w_1$ is a complex in G. Evidently $\lambda \in \mathscr{K}(C)$. However, $r_0 \notin \bigcap_{v \in \mathrm{par}(\lambda)} \mathrm{ch}_G(\lambda)$ implies $r_0 \notin A$, which contradicts the assumption $r_0 \in B$.

The second step is to show directly $r_k \in A$. Suppose by contradiction that there exists $\kappa \in \mathscr{K}(C)$ and $w \in \mathrm{par}(\kappa)$ with $r_k \to w$ or $r_k — w$ in G [one already knows that $r_k \in \mathrm{pa}_G(C)$]. But then, owing to the assumption $r_0 \to r_k$ or $r_0 — r_k$ in G, r_0 is an ancestor of w in G, which excludes the possibility $r_0 \in \mathrm{ch}_G(w)$. This contradicts the assumption $r_0 \in B \subset A$.

Thus, the option $r_0 \to r_k$ in G is excluded, since then a node $r_0 \in B$ has a child $r_k \in A$, which contradicts the assumption that B is a terminal component of G_A. Therefore $r_0 — r_k$ in G what implies $r_k \in B$. This means that both $r_0 \to r_1$ and $r_k \to r_{k-1}$ are changed into lines in H, and α is an undirected cycle in H.

Well, only simple subcases remain. If $r_{k-1} \notin C$, then $r_1 \notin C$ and both $r_0 \to r_1$ and $r_k \to r_{k-1}$ are saved in H. If $r_1, r_{k-1} \in C$, $r_0 \notin B$, and $r_0 \to r_k$ in G, then both $r_0 \to r_1$ and $r_0 \to r_k$ are saved in H. In case $r_1, r_{k-1} \in C$, $r_0 \notin B$, $r_0 — r_k$ in G, and $r_k \notin B$, both $r_0 \to r_1$ and $r_k \to r_{k-1}$ are saved in H. In case $r_1, r_{k-1} \in C$, $r_0 \notin B$, $r_0 — r_k$ in G, and $r_k \in B$, the roles of r_0 and r_k are exchangeable (one can use the already mentioned reindexing of α) and by their possible interchange one obtains the first-mentioned subcase.

So in every case α is not a directed cycle in H, and we have verified that H is a CG. By Theorem 2.2 it is Markov equivalent to G. ■

COROLLARY 4.1 *Suppose G is a* LCG *of a class of Markov equivalent* CGs *over N. Then for every connectivity component C of G the set* $A = \mathrm{pa}_G(C) \cap \bigcap_{\kappa \in \mathscr{K}(C)} \bigcap_{v \in \mathrm{par}(\kappa)} \mathrm{ch}_G(v)$ *is empty.*

Proof Under the situation that A is nonempty, one can take a terminal connectivity component B of G_A and define the graph H from Lemma 4.6. Evidently, at least one arrow in G was changed into a line in H. Thus,

H is a CG, Markov equivalent to G, which is larger than G, but distinct from G. This contradicts the definition of the largest CG. ■

LEMMA 4.7 *Let G be a* CG *over N such that for every its connectivity component C the set $A = \mathrm{pa}_G(C) \cap \bigcap_{\kappa \in \mathscr{K}(C)} \bigcap_{v \in \mathrm{par}(\kappa)} \mathrm{ch}_G(v)$ is empty. Then the LCG recovery algorithm (see Section* 4.1) *applied to the pattern G_0 (corresponding to G) yields G.*

Proof Lemmas 4.1–4.4 imply that every iteration of the LCG recovery algorithm is a map of G_∞. So every arrow in any iteration G_m, $m \geq 0$, is an arrow in G_∞ (with the same orientation) and thus in G. In particular, no other edge, except an arrow in G, will be an arrow in the last iteration G_* of the algorithm.

To show the converse one needs to verify that every arrow in G will be directed by one of the mentioned rules. Let us consider a connectivity component C of G with $\mathrm{pa}_G(C) \neq \emptyset$. Then $\mathscr{K}(C) \neq \emptyset$, as otherwise the set A is nonempty. Thus, there exists a complex with a region in C, and at least two arrows in G_0 are directed into nodes of C. Our aim is to verify that every arrow $u \to t$ in G from $u \in \mathrm{pa}_G(C)$ into $t \in C$ will be directed by the rules (this is sufficient to prove the mentioned converse statement). Two basic cases will be distinguished.

I. *There exists $\kappa \in \mathscr{K}(C)$, $v \in \mathrm{par}(\kappa)$ such that $\{u, v\}$ is not an edge in G* [in particular, this includes the case when $\kappa \in \mathscr{K}(C)$ with $u \in \mathrm{par}(\kappa)$]. We can suppose $u \neq v$ as otherwise we replaced v by the other parent of κ. Let us consider a path between v and t in $G_{C \cup \{v\}}$ which cannot be shortened. Necessarily, it looks like $v = r_1 \to r_2 - \cdots - r_l = t$, $l \geq 2$. Take minimal $1 \leq i \leq l$ such that $\{u, r_i\}$ is an edge of G. Then $i \geq 2$, and $\lambda : r_1 \to r_2 - \cdots - r_i \leftarrow u$ is a complex from $\mathscr{K}(C)$ with $u \in \mathrm{par}(\lambda)$. Thus, $r_1 \to r_2$ in G_0, and one can apply the transitivity rule to $r_1, \ldots, r_l$ to derive $\neg\{r_i \leftarrow r_{i+1}\}_\infty$ for $i = 2, \ldots, l-1$ in a future iteration G_m, $m \geq 0$. In case $i = l$ one already has $u \to r_l = t$ in G_0. Otherwise take minimal $i + 1 \leq j \leq l$ such that $\{u, r_j\}$ is an edge in G. Necessarily $u \to r_j$ in G, and in case $u - r_j$ in some iteration G_m (otherwise it is already directed in G_m) one can use the necessity rule (the first variant) for the cycle $u \to r_i - r_{i+1} - \cdots - r_j - u$ (it is a chordless cycle) to derive $u \to r_j$ in a future iteration. In case $j \neq l$, repeat the procedure until the arrow $u \to r_l = t$ is directed in an iteration G_n, $n \geq m$.

II. *For every complex $\kappa \in \mathscr{K}(C)$ and $v \in \mathrm{par}(\kappa)$ the edge $\{u, v\}$ occurs in G.* Owing to the assumption that the set A is empty, there exists a complex $\lambda : v = w_1 \to w_2 - \cdots - w_{k-1} \leftarrow w_k$, $k \geq 3$, in $\mathscr{K}(C)$ such that either $u \to v$ or $u - v$ in G. Note that this implies that the possibility $u \leftarrow v$ in G_m is excluded for every $m \geq 0$. Let us again consider the "shortest" path $v = r_0 \to r_1 - \cdots - r_l = t$, $l \geq 1$, between v and t in $G_{C \cup \{v\}}$. It was already shown that $r_0 \to r_1$ in an iteration G_m, $m \geq 0$ [because $\lambda \in \mathscr{K}(C)$ and $v \in \mathrm{par}(\lambda)$—see I, where $u = v$ and $t = r_1$]. The application of the transitivity rule then allows us to derive $\neg\{r_i \leftarrow r_{i+1}\}_\infty$ for $i = 1, \ldots, l-1$ in another future iteration. Take minimal $1 \leq h \leq l$ such that $\{u, r_h\}$ is an edge of G. Then $r_0, r_1, \ldots, r_h, u, r_0$ is a chordless cycle in G. In case $u \to v$ in an apposite future iteration G_m, one can apply directly the necessity rule (the second variant applied to $u, v = r_0, r_1, \ldots, r_h, u$) in order to direct $u \to r_h$ in the next iteration G_{m+1}. In case $u — v$ in the apposite future iteration G_m one can use the double-cycle rule, where $w_k = s_0 \to s_1 — \cdots — s_j = r_1$ is the "shortest" path in G between w_k and r_1 belonging to $C \cup \{w_k\}$. One can be sure that $s_0 \to s_1$ will be directed in a future iteration (see I), and $\neg\{s_i \leftarrow s_{i+1}\}_\infty$ for all $i = 1, \ldots, j-1$ will be derived, too. Thus, the arrow $u \to r_h$ will be directed. By the same procedure as in case I one can show that the application of the necessity rule (the first variant) will derive $u \to t$ in the end. ■

Proof of Theorem 4.1 By Theorem 2.3 there exists a LCG for G. Because G and G_∞ are Markov equivalent, the pattern corresponding to G and the pattern corresponding to G_∞ coincide. According to Corollary 4.1 the assumptions of Lemma 4.7 for G_∞ (instead of G) are satisfied. ■

4.3. Formal Strengthening of the Rules

The original formulation of the rules used in the LCG recovery algorithm in [19] was much stronger than in Section 4.1. Let us recall it.

By a *semislide* from a node w_1 to a node w_k in an (extended) hybrid graph H will be understood a descending route $w_1, \ldots, w_k$, $k \geq 2$, with $w_1 \to w_2$. Evidently, any slide is a semislide.

EXTENDED TRANSITIVITY RULE Suppose that $w_1, \ldots, w_{k-1}$, $k \geq 3$, is a steady semislide in an iteration G_m $(m \geq 0)$, $w_{k-1} — w_k$ in G_m, and there is no edge in G_m between w_k and $\{w_1, \ldots, w_{k-2}\}$. Then $\neg\{w_{k-1} \leftarrow w_k\}_\infty$ in the next iteration G_{m+1} is derived.

In short, the preceding rule derives that $w_1, \ldots, w_k$ is a steady semislide in a future iteration. So a steady semislide $w_1, \ldots, w_{k-1}$ is prolonged in the next iteration. This motivated the name "transitivity rule."

Evidently, if the assumptions of the transitivity rule from Section 4.1 are satisfied, then by successive application of the extended transitivity rule the desired aim of the transitivity rule is derived in a future iteration. So the extended transitivity rule is formally stronger.

EXTENDED NECESSITY RULE Suppose that $r_0, \dots, r_k, r_0$, $k \geq 2$, is a pseudocycle in an iteration G_m $(m \geq 0)$, such that (under the convention $r_{k+1} = r_0$) one has $r_0 \rightarrow r_1$ in G_m, $r_j — r_{j+1}$ in G_m for some $1 \leq j \leq k$, and for every $i \in \{1, \dots, k\} \setminus \{j\}$ either $r_i \rightarrow r_{i+1}$ in G_m or $[r_i — r_{i+1}$ and $\neg\{r_i \leftarrow r_{i+1}\}_\infty$ in $G_m]$. Then the line $r_j — r_{j+1}$ is changed into the arrow $r_j \leftarrow r_{j+1}$ in G_{m+1}.

It is evident that the assumptions of the extended necessity rule are weaker than the assumptions of the necessity rule from Section 4.1 (both variants)—every chordless cycle is a pseudocycle, and one can take $j = k$ above. Therefore it generalizes the necessity rule.

EXTENDED DOUBLE-CYCLE RULE Suppose that $r_0, \dots, r_k, r_0$, $k \geq 2$, is a pseudocycle in an iteration G_m $(m \geq 0)$, such that $r_0, \dots, r_{k-1}$ is a steady semislide in G_m, and $r_{k-1} — r_k$, $r_k — r_{k+1} = r_0$ in G_m. Moreover let us suppose that $s_0, \dots, s_l$, $l \geq 1$, is a steady semislide to $r_1 = s_l$ such that $s_0 \neq r_0$ and there exists $0 \leq n \leq l - 1$ such that $\{r_k, s_n\}$ is an edge in G_m, but there is no edge in G_m between r_0 and $\{s_0, \dots, s_n\}$. Then the line $r_{k-1} — r_k$ in G_m is changed into the arrow $r_{k-1} \leftarrow r_k$ in G_{m+1}.

Note that the edge $\{r_{k-1}, r_k\}$ which is directed by the previous rule belongs to two (pseudo)cycles, namely to $r_0, \dots, r_k, r_0$ and $s_n, \dots, s_l = r_1, \dots, r_k, s_n$. This fact motivated the terminology.

Again, the extended double-cycle rule evidently generalizes the respective rule from Section 4.1: a chordless cycle is a pseudocycle, a slide is a semislide, and it suffices to put $n = 0$. Therefore, this rule is formally stronger.

However, all the extended rules are sound.

LEMMA 4.8 *The application of the extended transitivity rule, the extended necessity rule, and the extended double-cycle rule to a map of a* LCG G_∞ *yields a map of* G_∞.

Proof In fact, one can repeat the same arguments which were used in the proofs of Lemmas 4.2, 4.3, 4.4; the only problem is that one has to consider a more complex situation. It is left to the reader as an exercise. Let us give two hints.

In the case of the extended necessity rule one shows for condition (b) for G_{m+1} from Definition 4.1 that $r_j \leftarrow r_{j+1}$ in G_∞ and uses Lemma 2.1(ii) for that purpose.

In the case of the extended double-cycle rule one also uses Lemma 2.1(ii) to derive $r_k \leftarrow r_0$ in G_∞. To verify that it is a complex arrow in G_∞ three subcases should be distinguished. If $n = 0$, then one repeats the procedure from the proof of Lemma 4.4. In subcase $n \geq 1$ and $r_k \rightarrow s_n$ in G_∞ one derives a contradictory conclusion that $r_k, s_n, \dots, s_l = r_1, \dots, r_k$ is a directed cycle in G_∞. However, in subcase $n \geq 1$ and either $r_k \leftarrow s_n$ or $r_k — s_n$ in G_∞ one applies Lemma 3.5 to $L = G_\infty$ and $s_0, s_1, \ldots, s_n, r_k, r_0$ to derive that $r_k \leftarrow r_0$ is a complex arrow in G_∞. ■

Lemma 4.8 together with the proof of Theorem 4.1 implies that the modified LCG recovery algorithm, where the extended rules are taken into account, also yields the largest CG. In short:

THEOREM 4.2 *Supposing G_0 is a pattern corresponding to a* CG *G, the last iteration of the modified* LCG *recovery algorithm (with extended rules) is the largest* CG *corresponding to G.*

The previous result is nothing but Theorem 5.1 from [19]. Note for explanation that the original (longer) proof of convergence of the LCG recovery algorithm was based on the extended rules. But later, the proof was substantially simplified, and this change led to a more elegant formulation of the rules.

5. CHARACTERIZATION OF THE LCG

The preceding results allow us to derive as a by-product a graphical characterization of graphs which are the largest CGs of classes of Markov equivalent CGs.

COROLLARY 5.1 *A* CG *G over N is the largest* CG *of a class of Markov equivalent* CG*s iff for every its connectivity component C the set $A = \mathrm{pa}_G(C) \cap \bigcap_{\kappa \in \mathscr{K}(C)} \bigcap_{v \in \mathrm{par}(\kappa)} \mathrm{ch}_G(v)$ is empty.*

Proof The necessity of the condition follows from Corollary 4.1. For sufficiency one realizes that Lemma 4.7 says that the result of the LCG recovery algorithm is G, while Theorem 4.1 says that it is G_∞. Therefore $G = G_\infty$. ■

Moreover a simple algorithm changing every CG into the corresponding largest CG can be obtained. It consists in consecutive application of the following rule.

POOL-COMPONENT RULE Suppose that G is a CG, and C its connectivity component such that the set

$$A = \mathrm{pa}_G(C) \cap \bigcap_{\kappa \in \mathscr{K}(C)} \; \bigcap_{v \in \mathrm{par}(\kappa)} \mathrm{ch}_G(v)$$

is nonempty. Let us choose a terminal connectivity component B of the induced subgraph G_A. Then all arrows in G oriented from B to C will be changed into lines in the next iteration (all other edges will be saved).

In fact, the components B and C are pooled in the next iteration. This motivated the terminology.

THEOREM 5.1 *Supposing G is a* CG *over N, the consecutive application of the pool-component rule to G yields a sequence of* CG*s which is finite, and the last graph in the sequence is the largest* CG *of the class of* CG*s Markov equivalent to G.*

Proof It follows from Lemma 4.6 that the application of the pool-component rule yields a CG which is Markov equivalent to the original graph. So every iteration has that property. The pool-component rule can be applied until for every component (of an iteration) the corresponding set A will be empty. As the number of arrows to be changed into lines is finite, the procedure has to stop. However, this means by Corollary 5.1 that the last iteration is the largest CG of a class of Markov equivalent CGs. This class of course contains the last iteration, and therefore it contains also the starting iteration G. ■

6. CONCLUSIONS

Several remarks conclude this contribution. The first remark concerns the significance of the concept of largest CG. Markov networks have one big advantage: different UGs yield different dependency models. Bayesian networks have no such pleasant property: two different DAGs may represent the same dependency model, that is, be Markov equivalent. Moreover, the class of Markov equivalent DAGs has no natural representative, and one has to represent the class by a pattern or by an essential graph. However, then the problem arises whether such a representation allows one to identify the corresponding dependency model. As patterns and essential graphs are not DAGs in general, one cannot use the criteria for DAGs to obtain the dependency model. However, the concept of largest CG provides a reasonable solution even in the case of Bayesian networks. One can represent the class of Markov equivalent DAGs by the largest CG of the corresponding class of Markov equivalent CGs (which is, of course, wider, but represents the same dependency model). As the largest CG is a real member of the class of Markov equivalent CGs, one can identify the corresponding dependency model by some criterion for CGs, for example by the moralization criterion. In fact, the concept of essential graph also provides a solution of the mentioned problem, because it also belongs to the class of CGs which are Markov equivalent to the considered DAG, and one can use the criterion for CGs—see [1].

The second remark concerns the pattern recovery algorithm. It has an important feature: it depends only on predicates $\langle u, v \mid - \rangle$ and $\langle u, v \mid +w \rangle$

introduced in Definition 3.2. In particular, two CG models which coincide on these predicates must be equal. The number of such predicates is polynomial in the number of variables, unlike the exponential number of triplets in a general dependency model. This may give a more precise estimate of the number of CG models or DAG models. Perhaps a representation of DAG models in terms of these predicates would be more effective.

The third remark concerns the characterization of largest CGs and the pool-component rule. The simple algorithm described in the fifth section is of course more suitable in the considered situation than the combination of the moralization criterion for obtaining the induced dependency model with the pattern recovery algorithm and with the LCG recovery algorithm. It makes it possible to test for the Markov equivalence of CGs in alternative way (without finding complexes throughout both graphs—the complexes are "found" only locally). Evidently, two CGs are Markov equivalent iff their corresponding largest CGs coincide—that is, iff the application of the pool-component rule to both CGs gives the same result. Another future application might be a method to find the number of CG models over n variables. Perhaps an exact formula for the number of Markov equivalence classes will be obtained someday.

The complexity of all these algorithms remains a topic for further research.

Some final remarks are responses to reviewers' comments on the paper [19], which I could not include in that paper owing to space limitation. One of the reviewers encouraged me to compare CG models and models induced by embedded Bayesian networks treated in [21] and [11]. CGs are not a special case of embedded Bayesian networks: they involve UGs, and there are UG models which are not restrictions (I mean restrictions of dependency models) of DAG models—for example, the dependency model induced by an undirected cycle of length 4. On the other hand, there exist restrictions of DAG models which are not CG models. Perhaps it will be a topic of further research to study embedded CG models, which should involve both above-mentioned classes of models.

Another reviewer suggested discussing also the question whether essential arrows in a DAG which are lines in the corresponding largest CG carry a causal meaning or not (for example, the arrow $d \leftarrow e$ in Figure 7(ii)). The reviewer mentioned that some authors (for example [5]) have argued that under additional assumptions every essential arrow has a causal interpretation. My impression is that such a claim has only relative validity —when one is limited to the framework of DAG models. Perhaps if one allows the use of a wider class of graphs one will find that the conditional independence structure of a considered probability distribution can be described also by a CG where relationships between some variables are depicted by lines which have the interpretation of symmetrical associations.

ACKNOWLEDGMENTS

This work was supported by the grant of Grant Agency of the Czech Republic (201/94/0471) and MŠMT (VS96008). I would like to express my gratitude to the reviewers of the paper [19] and to Jan Koster for their useful suggestions. I am also grateful to the editor for grammatical corrections.

References

1. Andersson, S. A., Madigan, D., and Perlman, M. D., A characterization of Markov equivalence classes for acyclic digraphs, to appear in *Ann. Statist.*, 25, 1997.
2. Bouckaert, R. R., Bayesian belief networks: From construction to inference, Doctoral Thesis, Univ. of Utrecht, 1995.
3. Bouchaert, R. R., and Studený, M., Chain graphs: Semantics and expressiveness, in *Symbolic and Quantitative Approaches to Reasoning and Uncertainty* (Ch. Froidevaux and J. Kohlas, Eds.), Lectures Notes in Artificial Intelligence 946, Springer-Verlag, 67–76, 1995.
4. Buntine, W. L., Chain graphs for learning, in *Uncertainty in Artificial Intelligence 11* (P. Besnard and S. Hanks, Eds.), Morgan Kaufmann, 46–54, 1995.
5. Chickering, D. M., A transformational characterization of equivalent Bayesian network structures, in *Uncertainty in Artificial Intelligence 11* (P. Besnard and S. Hanks, Eds.), Morgan Kaufmann, 87–98, 1995.
6. Cox, D. R., and Wermuth, N., *Multivariate Dependencies—Models Analysis and Interpretation*, Chapman & Hall, London, 1996.
7. Frydenberg, M., The chain graph Markov property, *Scand. J. Statist.*, 17, 333–353, 1990.
8. Frydenberg, M., Marginalization and collapsability in graphical interaction models, *Ann. Statist.* 18, 790–805, 1990.
9. Geiger, D., and Pearl, J., On the logic of causal models, in *Uncertainty in Artificial Intelligence 4* (R. D. Shachter, T. S. Lewitt, L. N. Kanal, and J. F. Lemmer, Eds.), North-Holland, 3–14, 1990.
10. Geiger, D., and Pearl, J., Logical and algorithmic properties of conditional independence and graphical models, *Ann. Statist.*, 21, 2001–2021, 1993.

11. Geiger, D., Paz, A., and Pearl, J., On testing whether an embedded Bayesian network represents a probability model, in *Uncertainty in Artificial Intelligence 10* (R. L. de Mantaras and D. Poole, Eds.), Morgan Kaufmann, 244–252, 1994.

12. Lauritzen, S. L., and Wermuth, N., Mixed interaction models, Res. Report R-84-8, Inst. Elec. Sys., Univ. of Aalborg, 1984 (basis of [13]).

13. Lauritzen, S. L., and Wermuth, N., Graphical models for associations between variables, some of which are qualitative and some quantitative, *Ann. Statist.*, 17, 31–57, 1989.

14. Lauritzen, S. L., Mixed graphical association models, *Scand. J. Statist.* 16, 273–306, 1989.

15. Lauritzen, S. L., Dawid, A. P., Larsen, B. N., and Leimer, H.-G., Independence properties of directed Markov fields, *Networks* 20, 491–505, 1990.

16. Meek, Ch., Causal inference and causal explanation with background knowledge, in *Uncertainty in Artificial Intelligence 11* (P. Besnard and S. Hanks, Eds.), Morgan Kaufmann, 403–410, 1995.

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

18. Studený, M., and Bouckaert, R. R., On chain graph models for description of conditional independence structure, *Ann. Statist.*, accepted for publication.

19. Studený, M., On separation criterion and recovery algorithm for chain graphs, in *Uncertainty in Artificial Intelligence 12* (E. Horvitz and F. Jensen, Eds.), Morgan Kaufmann, 509–516, 1996.

20. Verma, T., and Pearl, J., Causal networks: Semantics and expressiveness, in *Uncertainty in Artificial Intelligence 4* (R. D. Shachter, T. S. Lewitt, L. N. Kanal, and J. F. Lemmer, Eds.), North-Holland, 69–76, 1990.

21. Verma, T., and Pearl, J., Equivalence and synthesis of causal models, in *Uncertainty in Artificial Intelligence 6* (P. P. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, Eds.), Elvevier, 220–227, 1991.

22. Verma, T., and Pearl, J., An algorithm for deciding if a set of observed independencies has a causal explanation, in *Uncertainty in Artificial Intelligence 8* (D. Dubois, M. P. Wellman, B. D'Ambrosio, and P. Smets, Eds.), Morgan Kaufmann, 323–330, 1992.

23. Whittaker, J., *Graphical Models in Applied Multivariate Statistics*, Wiley, 1990.