Stable mixed graphs

KAYVAN SADEGHI

Department of Statistics, University of Oxford, 1 South Parks Road, Oxford, OX1 3TG, United Kingdom.
E-mail: sadeghi@stats.ox.ac.uk

In this paper, we study classes of graphs with three types of edges that capture the modified independence structure of a directed acyclic graph (DAG) after marginalisation over unobserved variables and conditioning on selection variables using the *m*-separation criterion. These include MC, summary, and ancestral graphs. As a modification of MC graphs, we define the class of ribbonless graphs (RGs) that permits the use of the *m*-separation criterion. RGs contain summary and ancestral graphs as subclasses, and each RG can be generated by a DAG after marginalisation and conditioning. We derive simple algorithms to generate RGs, from given DAGs or RGs, and also to generate summary and ancestral graphs in a simple way by further extension of the RG-generating algorithm. This enables us to develop a parallel theory on these three classes and to study the relationships between them as well as the use of each class.

Keywords: ancestral graph; directed acyclic graph; independence model; *m*-separation criterion; marginalisation and conditioning; MC graph; summary graph

1. Introduction

Introduction and motivation. In graphical Markov models, graphs have been used to represent conditional independence statements of sets of random variables. Nodes of the graph correspond to random variables and edges typically capture dependencies. Different classes of graphs with different interpretation of independencies have been defined and studied in the literature.

One of the most important classes of graphs in graphical models is the class of directed acyclic graphs (DAGs) [4,6]. Their corresponding Markov models, often known under the name of Bayesian networks [8], have direct applications to a wide range of areas including econometrics, social sciences, and artificial intelligence. When, however, some variables are unobserved, that is also called latent or hidden, one can in general no longer capture the implied independence model among observed variables by a DAG. In this sense, the DAG models are not stable under marginalisation. A similar problem occurs because DAG models are not stable under conditioning [2,9,16].

This makes it necessary to identify and study a class of graphs that includes DAGs and is stable under marginalisation and conditioning in the sense that it is able to express the induced independence model after marginalisation and conditioning through an object of the same class. The methods that have been used to solve this problem employ three different types of edges instead of a single type.

Three known classes of graphs have previously been suggested for this purpose in the literature. We specifically call these stable mixed graphs (under marginalisation and conditioning) and they include MC graphs (MCGs) [5], summary graphs (SGs) [13,16], and ancestral graphs (AGs) [10].

MCGs do not use the same interpretation of independencies, called the *m*-separation criterion, as the other types of stable mixed graphs. In this paper, we use similar methods as in [5] to derive a modification of the class of MCGs to use *m*-separation, which we call ribbonless graphs (RGs). The class of RGs is exactly the class with three types of edges that is generated after marginalisation over and conditioning on the node sets of a DAG. More importantly, we extend the RG-generating algorithm to generate summary and ancestral graphs in a theoretically neat way. These algorithms are computationally polynomial, even though we shall not go through their computational complexity in this paper. Defining these algorithms leads to establishing a parallel theory for the different classes, and studying the similarities, differences, and relationships among them.

Structure of the paper. In the next section, we define some basic concepts of graph theory and independence models needed in this paper.

In Section 3, we define the class of RGs, give some basic graph-theoretical definitions for these, and define the *m*-separation criterion for interpretation of the independence structure on them.

In Section 4, we formally define marginalisation and conditioning for independence models in such a way that it conforms with marginalisation and conditioning for probability distributions. We also formally define stable classes of graphs.

Each of the next three sections of this paper deals with one type of stable mixed graphs. We discuss RGs in Section 5, SGs in Section 6, and AGs in Section 7. In each section, we introduce a straightforward algorithm to generate the stable mixed graph from DAGs or from graphs of the same type. For each type of stable mixed graph, we prove that the graphs and algorithms are well-defined in the sense that instead of marginalising over or conditioning on a set of nodes, by splitting the marginalisation or conditioning set into two subsets, one can marginalise over or condition on the first subset first, and then marginalise over or condition on the second subset and obtain the same graph. We also prove that the generated graphs induce the modified independence model after marginalisation and conditioning, meaning that the generated classes are stable under marginalisation and conditioning.

In Section 8, we scrutinise the relationships between the three types of stable mixed graphs. In Section 9, we provide a discussion on the use of the different classes of stable mixed graphs.

In the Appendix, we provide the proof of lemmas, propositions, and theorems given in the previous sections.

2. Basic definitions and concepts

Independence models and graphs. An *independence model* $\mathcal{J}$ over a set V is a set of triples $\langle X, Y | Z \rangle$ (called *independence statements*), where X, Y, and Z are disjoint subsets of V and Z can be empty, and $\langle \varnothing, Y | Z \rangle$ and $\langle X, \varnothing | Z \rangle$ are always included in $\mathcal{J}$. The independence statement $\langle X, Y | Z \rangle$ is interpreted as "X is independent of Y given Z". Notice that independence models contain probabilistic independence models as a special case. For further discussion on independence models, see [12].

A *graph* G is a triple consisting of a *node* set or *vertex* set V, an *edge* set E, and a relation that with each edge associates two nodes (not necessarily distinct), called its *endpoints*. When nodes

i and j are the endpoints of an edge, these are *adjacent* and we write $i \sim j$. We say the edge is *between* its two endpoints. We usually refer to a graph as an ordered pair $G = (V, E)$. Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called *equal* if $(V_1, E_1) = (V_2, E_2)$. In this case we write $G_1 = G_2$.

Notice that the graphs that we use in this paper (and in general in the context of graphical models) are so-called *labeled graphs*, that is, every node is considered a different object. Hence, for example, graph $i \longrightarrow j \longrightarrow k$ is not equal to $j \longrightarrow i \longrightarrow k$.

We use the notation $\mathcal{J}^G$ for an independence model defined over the node set of G. Among the independence models over the node set V of a graph G, those that are of interest to us *conform* with G, meaning that $i \sim j$ in G implies $\langle i, j|C \rangle \notin \mathcal{J}$ for any $C \subseteq V \setminus \{i, j\}$. Henceforth, we assume that independence models $\mathcal{J}^G$ conform with G, unless otherwise stated. Notice that henceforth we use the notation i instead of $\{i\}$ for a subset consisting of a single element i in an independence statement.

Basic graph theoretical definitions. Here we introduce some basic graph theoretical definitions. A *loop* is an edge with the same endpoints. *Multiple edges* are edges with the same pair of endpoints. A *simple graph* has neither loops nor multiple edges.

If a graph assigns an ordered pair of nodes to each edge, then the graph is a *directed graph*. We say that the edge is *from* the first node of the ordered pair *to* the second one. We use an arrow, $j \longrightarrow i$, to draw an edge in a directed graph. We also call node j a *parent* of i, node i a *child* of j and we use the notation pa(i) for the set of all parents of i in the graph.

A *walk* is a list $\langle v_0, e_1, v_1, \ldots, e_k, v_k \rangle$ of nodes and edges such that for $1 \le i \le k$, the edge e_i has endpoints v_{i-1} and v_i. A *path* is a walk with no repeated node or edge. A *cycle* is a walk with no repeated node or edge except $v_0 = v_k$. If the graph is simple, then a path or a cycle can be determined uniquely by an ordered sequence of node sets. Throughout this paper, however, we use node sequences for describing paths and cycles even in graphs with multiple edges, but we suppose that the edges of the path are all determined. Usually it is apparent from the context or the type of the path which edge belongs to the path in multiple edges. We say a path is *between* the first and the last nodes of the list in G. We call the first and the last nodes *endpoints* of the path and all other nodes *inner nodes*.

A path (or a cycle) in a directed graph is *direction preserving* if all its arrows point to one direction ($\circ \longrightarrow \circ \longrightarrow \cdots \longrightarrow \circ$). A directed graph is *acyclic* if it has no direction-preserving cycle.

If in a direction-preserving path an arrow starts at a node j and an arrow points to a node i, then j is an *ancestor* of i, and i a *descendant* of j. We use the notation an(i) for the set of all ancestors of i.

3. Independence model for ribbonless graphs

Loopless mixed graphs. Graphs that will be discussed in this paper are subclasses of loopless mixed graphs. A *mixed graph* is a graph containing three types of edges denoted by arrows, arcs (two-headed arrows), and lines (solid lines). Mixed graphs may have multiple edges of different types but do not have multiple edges of the same type. We do not distinguish between $i \longrightarrow j$ and $j \longrightarrow i$ or $i \longleftrightarrow j$ and $j \longleftrightarrow i$, but we do distinguish between $j \longrightarrow i$ and $i \longrightarrow j$. Thus

there are up to four edges as a multiple edge between any two nodes. A *loopless mixed graph* (LMG) is a mixed graph that does not contain any loops (a loop may be formed by a line, arrow, or arc).

Some definitions for mixed graphs. For a mixed graph H, we keep the same terminology introduced before for directed and undirected graphs. We say that i is a *neighbour* of j if these are endpoints of a line, and i is a parent of j if there is an arrow from i to j. We also define that i is a *spouse* of j if these are endpoints of an arc. We use the notations $\mathrm{ne}(j)$, $\mathrm{pa}(j)$, and $\mathrm{sp}(j)$ for the set of all neighbours, parents, and spouses of j, respectively.

In the cases of $i \longrightarrow j$ or $i \longleftrightarrow j$, we say that there is an arrowhead pointing to (at) j. A path $\langle j = i_0, i_1, \ldots, i_n = i \rangle$ is *from j to i* if ji_1 is either a line or an arrow from j to i_1, and $i_{n-1}i$ is either an arc or an arrow from i_{n-1} to i.

A *V-configuration* or simply *Vs* is a path with three nodes and two edges. In a mixed graph, the inner node of three Vs $i \longrightarrow t \longleftarrow j$, $i \longleftrightarrow t \longleftarrow j$ and $i \longleftrightarrow t \longleftrightarrow j$ is a *collider* and the inner node of all other Vs is a *non-collider* node in the V or more generally in a path on which the V lies. We also call a V with collider or non-collider inner node a *collider* or *non-collider V*, respectively. We may mention that a node is collider or non-collider without mentioning the V or path when this is apparent from the context. Notice that originally [4] and in most texts, the endpoints of a V are not adjacent whereas we do not use this restriction.

Two paths (including Vs or edges) are called *endpoint-identical* if presence or lack of arrowheads pointing to endpoints on the path are the same in both. For example, $i \longrightarrow j$, $i \longrightarrow k \longleftrightarrow j$, and $i \longrightarrow k \longleftarrow l \longleftrightarrow j$ are all endpoint-identical as there is an arrowhead pointing to j but there is no arrowhead pointing to i on the paths.

Ribbonless graphs and its subclasses. The largest subclass of LMGs studied in this paper is the class of ribbonless graphs.

A *ribbon* is a collider V $\langle h, i, j \rangle$ such that

1. there is no endpoint-identical edge between h and j, that is, there is no hj-arc in the case of $h \longleftrightarrow i \longleftrightarrow j$; there is no hj-line in the case of $h \longrightarrow i \longleftarrow j$; and there is no arrow from h to j in the case of $h \longrightarrow i \longleftrightarrow j$;
2. i or a descendant of i is an endpoint of a line or on a direction-preserving cycle.

A *ribbonless graph* (RG) is an LMG that does not contain ribbons as induced subgraphs.

Figure 1 illustrates ribbons $\langle h, i, j \rangle$. Figure 2(a) illustrates a graph containing a ribbon $\langle h, i, j \rangle$. Figure 2(b) illustrates a ribbonless graph. Notice that $\langle h, i, j \rangle$ is not here a ribbon since there is a line between h and j.
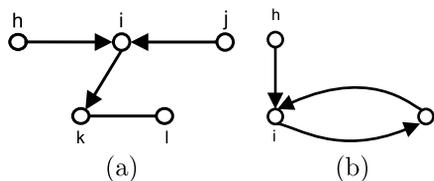
Figure 1. (a) A ribbon $\langle h, i, j \rangle$, where $\mathrm{ne}(i) = \varnothing$. (b) A ribbon $\langle h, i, j \rangle$.
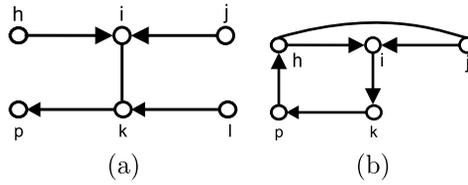
Figure 2. (a) A graph that is not ribbonless. (b) A ribbonless graph.

The three classes of undirected graphs (UGs) (used for concentration graph models), bidirected graphs (BGs) (used for covariance graph models), and DAGs are subclasses of RGs. SGs and AGs, which are studied in this paper, are also subclasses of RGs. We use the notations $\mathcal{RG}$, $\mathcal{SG}$, $\mathcal{AG}$, $\mathcal{UG}$, $\mathcal{BG}$ and $\mathcal{DAG}$ for the set of all RGs, SGs, AGs, UGs, BGs and DAGs, respectively. The common feature of all these graphs is that these all entail independence models using the same so-called separation criterion, which is called m-separation and will be shortly defined.

The m-separation criterion for RGs. The following definition was given in [10].

Let C be a subset of the node set V of an RG. A path is m-connecting given M and C if all its collider nodes are in $C \cup \mathrm{an}(C)$ and all its non-collider nodes are in M. For two other disjoint subsets of the node set A and B such that $M = V \setminus A \cup B \cup C$, we may just call the path m-connecting given C between A and B. We say $A \perp_m B | C$ if there is no m-connecting path between A and B given C.

Notice that the m-separation criterion induces an independence model $\mathcal{J}_m(G)$ on G by $A \perp_m B | C \iff \langle A, B | C \rangle \in \mathcal{J}_m(G)$.

4. Marginalisation, conditioning and stability

Marginal and conditional independence models. Consider an independence model $\mathcal{J}$ over a set V. For M a subset of V, the *independence model $\mathcal{J}$ after marginalisation over M*, denoted by $\alpha(\mathcal{J}; M, \varnothing)$, is the subset of $\mathcal{J}$ whose triples do not contain members of M, that is,

$$\alpha(\mathcal{J}; M, \varnothing) = \big\{ \langle A, B | D \rangle \in \mathcal{J} \colon (A \cup B \cup D) \cap M = \varnothing \big\}.$$

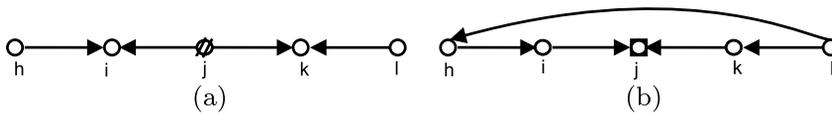One can observe that $\alpha(\mathcal{J}; M, \varnothing)$ is an independence model over $V \setminus M$.

For a subset C of V, the *independence model after conditioning on C*, denoted by $\alpha(\mathcal{J}; \varnothing, C)$, is

$$\alpha(\mathcal{J}; \varnothing, C) = \big\{ \langle A, B | D \rangle \colon \langle A, B | D \cup C \rangle \in \mathcal{J} \text{ and } (A \cup B \cup D) \cap C = \varnothing \big\}.$$

One can also observe that $\alpha(\mathcal{J}; \varnothing, C)$ is an independence model over $V \setminus C$.

Combining these definitions, for disjoint subsets M and C of V, the *independence model after marginalisation over M and conditioning on C* is

$$\alpha(\mathcal{J}; M, C) = \big\{ \langle A, B | D \rangle \colon \langle A, B | D \cup C \rangle \in \mathcal{J} \text{ and } (A \cup B \cup D) \cap (M \cup C) = \varnothing \big\},$$

Figure 3. (a) A directed acyclic graph G_1, by which it can be shown that the class of DAGs is not stable under marginalisation. ($\emptyset \in M$.) (b) A directed acyclic graph G_2, by which it can be shown that the class of DAGs is not stable under conditioning. ($\square \in C$.)

which is an independence model over $V \setminus (M \cup C)$.

Notice here that α is a function from the set of independence models and two of their subsets to the set of independence models. Notice also that operations for marginalisation and conditioning commute.

Marginalisation and conditioning in probability conform with marginalisation and conditioning for independence models. Consider a set $N = V \setminus (M \cup C)$ and a collection of random variables $(X_\alpha)_{\alpha \in N}$ with joint density $f_{(V \setminus M)|C}$. We associate an independence model to this density. It can be shown that if $\mathcal{J}$ is the associated independence model to the collection of random variables $(X_\alpha)_{\alpha \in V}$ with joint density f_V then the associated independence model to $(X_\alpha)_{\alpha \in N}$ with joint density $f_{(V \setminus M)|C}$ is $\alpha(\mathcal{J}, M, C)$.

Stability under marginalisation and conditioning for RGs and its subclasses. Consider a family of graphs $\mathcal{T}$. If, for every graph $G = (V, E) \in \mathcal{T}$ and every disjoint subsets M and C of V, there is a graph $H \in \mathcal{T}$ such that $\mathcal{J}_m(H) = \alpha(\mathcal{J}_m(G); \varnothing, C)$ then $\mathcal{T}$ is stable under conditioning, and if there is a graph $H \in \mathcal{T}$ such that $\mathcal{J}_m(H) = \alpha(\mathcal{J}_m(G); M, \varnothing)$ then $\mathcal{T}$ is stable under marginalisation. We call $\mathcal{T}$ *stable* (under marginalisation and conditioning) if there is a graph $H \in \mathcal{T}$ such that $\mathcal{J}_m(H) = \alpha(\mathcal{J}_m(G); M, C)$.

Notice that if the node set of such a graph H is N then $N = V \setminus (M \cup C)$.

We shall see that RGs, SGs, AGs, UGs, and BGs are stable. On the other hand, the class of DAGs is not stable. It can be shown that G_1 in Figure 3 is a DAG whose induced marginal independence model cannot be represented by a DAG and G_2 is a DAG whose induced conditional independence model cannot be represented by a DAG. We leave the details as an exercise to the readers.

Stable mixed graphs. As the class of DAGs is not stable, we look for stable classes of graphs that include the class of DAGs as a subclass. In this paper, we discuss three such types of graphs, namely RGs (as a modification of MCGs), SGs, and AGs, and specifically call these *stable mixed graphs*. We will see that in these graphs arcs are related to marginalisation and lines are related to conditioning.

For the graph $G_2 \in \mathcal{T}$ for which $\mathcal{J}^{G_2} = \alpha(\mathcal{J}_m(G_1); M, C)$, we use the notation $G_2 = \alpha_{\mathcal{T}}(G_1; M, C)$. For each type of stable mixed graphs, we later precisely define $\alpha_{\mathcal{T}}$ with specific algorithms. We call $\alpha_{\mathcal{T}}$ a *generating function* or more specifically a $\mathcal{T}$-*generating function*.

5. Ribbonless graphs

MC graphs and ribbonless graphs. MCGs only contain the three desired types of edges. However, these are not loopless and, in addition, in MCGs a different separation criterion is used for

inducing the independence model. However, from an MCG that can be generated by marginalisation and conditioning over DAGs and by a minor modification one can generate an RG that induces the same independence model. This modification includes adding edges between pairs of nodes connected by a ribbon such that the generated edges preserve the arrowheads at the endpoints of the ribbon, and removing all the loops. We shall not go through the details of this modification in this paper, but refer readers to [11].

5.1. Generating ribbonless graphs

A local algorithm to generate RGs from RGs. Here we present an algorithm to generate an RG from a given RG and two subsets of its node set that will be marginalised over and conditioned on. This algorithm is local in the sense that, after determining the ancestor set of the conditioning set, it looks solely for all Vs in the graph and not for longer paths. Later in this section, we will show that a graph generated by the algorithm is an RG and it induces the marginal and conditional independence model of the input graph by using m-separation.

Suppose that H is an RG and consider M and C two disjoint subsets of the node set. There are 10 possible non-isomorphic Vs in an RG, displayed in Table 1. Notice that this table generates endpoint-identical edges to the given Vs. We now define the following algorithm, derived from [16] and [5]. See also the appendix of [13].

Algorithm 1. $\alpha_{\mathrm{RG}}(H; M, C)$ (*Generating an RG from a ribbonless graph H*):
   *Start from H.*
   Generate an endpoint identical edge between the endpoints of collider Vs with inner node in $C \cup \mathrm{an}(C)$ and non-collider Vs with inner node in M, that is, generate an appropriate edge as in Table 1 between the endpoints of every V with inner node in M or $C \cup \mathrm{an}(C)$ if the edge of the same type does not already exist.
   Apply the previous step until no other edge can be generated. Then remove all nodes in $M \cup C$.

Table 1. Types of edge induced by Vs with inner node in $m \in M$ or $s \in C \cup \mathrm{an}(C)$

1	$i \longleftarrow m \longleftarrow j$	generates	$i \longleftarrow j$
2	$i \longleftarrow m \longrightarrow j$	generates	$i \longleftarrow j$
3	$i \longleftrightarrow m \longrightarrow j$	generates	$i \longleftarrow j$
4	$i \longleftarrow m \longrightarrow j$	generates	$i \longleftrightarrow j$
5	$i \longleftarrow m \longleftrightarrow j$	generates	$i \longleftrightarrow j$
6	$i \longrightarrow m \longleftarrow j$	generates	$i \longrightarrow j$
7	$i \longrightarrow m \longrightarrow j$	generates	$i \longrightarrow j$
8	$i \longleftrightarrow s \longleftarrow j$	generates	$i \longleftarrow j$
9	$i \longleftrightarrow s \longleftrightarrow j$	generates	$i \longleftrightarrow j$
10	$i \longrightarrow s \longleftarrow j$	generates	$i \longrightarrow j$
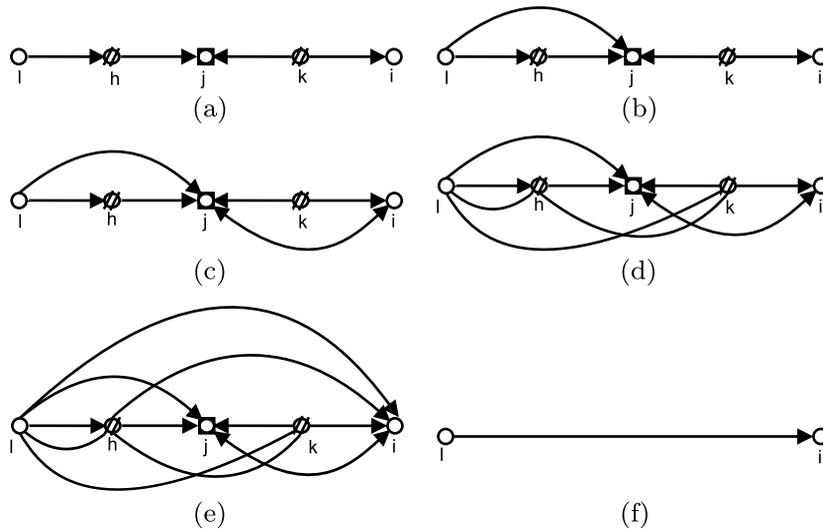
Figure 4. (a) A directed acyclic graph G, ⌀ $\in M$ and ◻ $\in C$. (b) The generated graph after applying step 1 of the table. (c) The generated graph after applying step 4. (d) The generated graph after applying step 10. (e) The generated graph after applying step 8. (f) The generated RG from G.

This method is a generalisation of the method used by [7], called *moralisation*, as a separation criterion on DAGs. Notice that the order of applying steps of Table 1 in Algorithm 1 is irrelevant since adding an edge does not destroy other Vs in the graph.

Figure 4 illustrates how to apply Algorithm 1 step by step to a DAG. We start from step 1 of Table 1 and proceed step by step. We return to step 1 at the end if there are any applicable steps left. Since $\mathcal{DAG} \subset \mathcal{RG}$, one can also use Algorithm 1 to generate an RG from a DAG. Notice that it is not enough to simply apply steps 1, 4, and 10 of Table 1 to a DAG.

Global interpretation of the algorithm. The following lemma explains the global characteristics of the process of marginalisation and conditioning.

Lemma 1. *Let H be a ribbonless graph. There exists an edge between i and j in the ribbonless graph $\alpha_{\mathrm{RG}}(H; M, C)$ if and only if there exists an endpoint-identical m-connecting path given M and C between i and j in H.*

Basic properties of α_{RG}. We show here that α_{RG} is an RG-generating function.

Proposition 1. *Graphs generated by Algorithm 1 are RGs.*

Notice that for every ribbonless graph H, it holds that $\alpha_{\mathrm{RG}}(H; \varnothing, \varnothing) = H$.

Surjectivity of α_{RG}. The following result shows that the class of RGs is the exact class of graph that is generated after marginalisation and conditioning for DAGs.

Proposition 2. *The map $\alpha_{\mathrm{RG}} : \mathcal{DAG} \to \mathcal{RG}$ is surjective.*

5.2. Two necessary properties of RG-generating functions

Here we establish the two important properties that α_{RG} (or every generating function) must have. In short, it must be well-defined and it must generate a stable class of graphs.

Well-definition of α_{RG}. The following theorem shows that α_{RG} is well-defined. This means that instead of directly generating an RG we can split the nodes that we marginalise over and condition on into two parts, first generate the RG related to the first part, then from the generated RG generate the desired RG related the second part.

Theorem 1. *For a ribbonless graph $H = (N, F)$ and disjoint subsets C, C_1, M, and M_1 of N,*

$$\alpha_{\mathrm{RG}}\big(\alpha_{\mathrm{RG}}(H; M, C); M_1, C_1\big) = \alpha_{\mathrm{RG}}(H; M \cup M_1, C \cup C_1).$$

Stability of the graphs generated by α_{RG}. Here we introduce the second important property that α_{RG} must have. This property is the core idea in defining RGs and in general stable mixed graphs. The modification applied by the function should generate a graph that induces the marginal and conditional independence model.

Theorem 2. *For a ribbonless graph $H = (N, F)$ and disjoint subsets A, B, C, C_1, and M of N,*

$$A \perp_m B | C_1 \qquad \text{in } \alpha_{\mathrm{RG}}(H; M, C) \qquad \Longleftrightarrow \qquad A \perp_m B | C \cup C_1 \qquad \text{in } H.$$

Corollary 1. *For a ribbonless graph $H = (N, F)$ and M and C disjoint subsets of N,*

$$\alpha\big(\mathcal{J}_m(H); M, C\big) = \mathcal{J}_m\big(\alpha_{\mathrm{RG}}(H; M, C)\big).$$

Corollary 2. *The class of RGs is stable.*

The following result has been implicitly discussed in the literature, for example, see [2].

Corollary 3. *The classes of UGs and BGs are stable.*

Proof. The result follows from the fact that, from UGs and BGs, Algorithm 1 generates UGs and BGs, respectively. □

Example. Figure 5 illustrates a DAG as well as two subsets M and C of its node set. Figure 6(a) illustrates the generated ribbonless graph H using Algorithm 1 as well as two subsets M_1 and C_1 of its node set, and Figure 6(b) illustrates the RG generated by the algorithm from H.

For example, consider the graph $\alpha_{\mathrm{RG}}(H, M_1, C_1)$ in Figure 6(b), and let $A = \{j\}$, $B = \{s\}$ and $C = \{i, u\}$. It is seen that $v \in \mathrm{an}(C)$. We have that A is not m-separated from B given C since $\langle j, i, v, t, s \rangle$ is an m-connecting path between A and B given C. By Theorem 2 we conclude that A is not m-separated from B given $C \cup C_1$. The same conclusion is made by observing m-connecting path $\langle j, i, v, w, t, r, s \rangle$ in H.
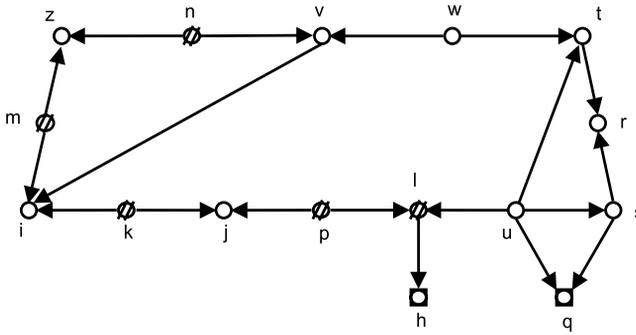
**Figure 5.** A directed acyclic graph G with sixteen nodes, ⊘ ∈ M and □ ∈ C.

6. Summary graphs

Definition of summary graphs. A *summary graph* is a loopless mixed graph $H = (N, F)$ which contains no ∘ —— ∘ ←— ∘ or ∘ —— ∘ ←→ ∘ (arrowhead pointing to line) and no direction-preserving cycle as subgraph. Notice that there are also no multiple edges in SGs except multiple edges consisting of an arrow and an arc.

Obviously the class of SGs is a subclass of RGs. Figure 7 illustrates an SG and an RG that is not an SG. (Because of two reasons: existence of arrowheads pointing to lines and existence of a double edge containing line and arrow.)

6.1. Generating summary graphs

A local algorithm to generate SGs. We now present a local algorithm (after determining the ancestor set of the conditioning set) to generate an SG from an SG.
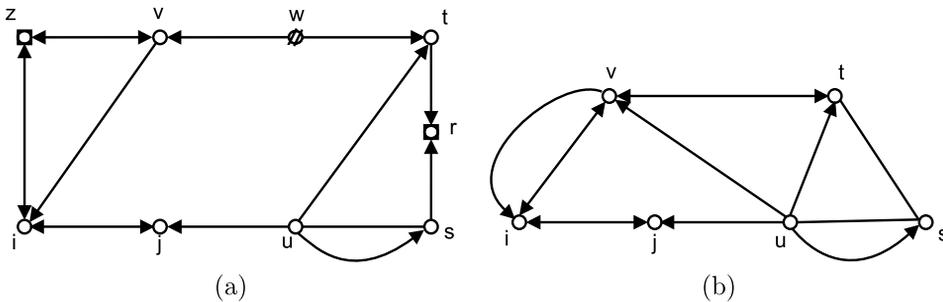
Figure 6. (a) The generated ribbonless graph $H = \alpha_{RG}(G, M, C)$ from the directed acyclic graph G in Figure 5, ⊘ ∈ M_1 and □ ∈ C_1. (b) The generated ribbonless graph $\alpha_{RG}(H, M_1, C_1)$ from H.
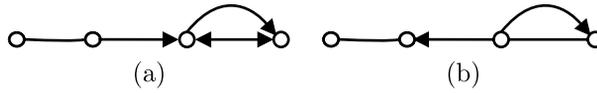
Figure 7. (a) An SG. (b) An RG that is not an SG.

Algorithm 2. $\alpha_{\text{SG}}(H; M, C)$: (*Generating an SG from a summary graph H*)
   Start from H. Label the nodes in an(C).

1. *Apply Algorithm* 1.
2. *Remove all edges (arrows or arcs) with arrowhead pointing to a node in* an(C), *and replace these by the edge with the arrowhead removed (line or arrow) if the edge does not already exist.*

Continually apply step 1 *until it is not possible to apply the given step further before moving to the second step.*

Figure 8 illustrates how to apply Algorithm 2 step by step to a DAG. Notice that as it is stated in the description of the algorithm the order of applying the steps does matter here.
The map α_{SG} and its basic properties. For SGs, we prove analogous results to those for RGs.

Proposition 3. *Graphs generated by Algorithm* 2 *are SGs.*

The map $\alpha_{\text{RG.SG}}$ and its properties. Notice that step 1 of Algorithm 2 generates an RG before removing the nodes in C. Hence, step 2 of the algorithm generates an SG from an RG and some extra nodes that are conditioned on. We denote these two steps by $\alpha_{\text{RG.SG}}$. This shows that for generating RGs from SGs, an(C) is needed.

Proposition 4. *Let $H = (N, E)$ be a ribbonless graph and M and C be subsets of N. It holds that $\alpha_{\text{SG}}(H; M, C) = \alpha_{\text{RG.SG}}(\alpha_{\text{RG}}(H; M, C); \text{an}(C))$.*

Surjectivity of α_{SG}. The following result shows that every member of $\mathcal{SG}$ can be generated by a DAG after marginalisation and conditioning.
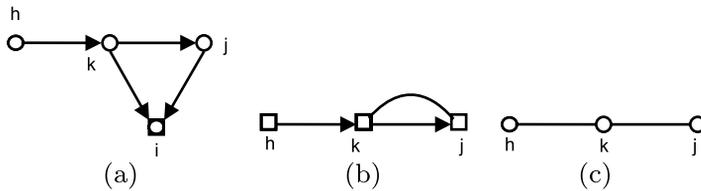
Figure 8. (a) A directed acyclic graph $G, \square \in C$. (b) The generated graph after applying step 1, $\square \in$ an(C). (c) The generated SG after applying step 2.

Proposition 5. *The map $\alpha_{SG} : \mathcal{DAG} \to \mathcal{SG}$ is surjective.*

6.2. Two necessary properties of SG-generating functions

Here, we express two important results that have been introduced for graphs generated by α_{RG} for graphs generated by α_{SG}.

Well-definition of α_{SG}. This property is analogous to the well-definition of α_{RG} as defined in the previous section. For a proof based on matrix representations of graphs and on properties of corresponding matrix operators, see [13].

Theorem 3. *For a summary graph $H = (N, F)$ and disjoint subsets C, C_1, M, and M_1 of N,*

$$\alpha_{SG}\big(\alpha_{SG}(H; M, C); M_1, C_1\big) = \alpha_{SG}(H; M \cup M_1, C \cup C_1).$$

Stability of the graphs generated by α_{SG}. We prove that analogous to RGs, graphs generated by α_{SG} induce the marginal and conditional independence model. This result can be implied from what was discussed in [13].

**Theorem 4.** *For a summary graph $H = (N, F)$ and disjoint subsets A, B, C, C_1, and M of N,*

$$A \perp_m B | C_1 \quad \text{in } \alpha_{SG}(H; M, C) \quad \Longleftrightarrow \quad A \perp_m B | C \cup C_1 \quad \text{in } H.$$

Corollary 4. *For a summary graph $H = (N, F)$ and M and C disjoint subsets of N,*

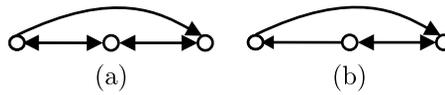$$\alpha\big(\mathcal{J}_m(H); M, C\big) = \mathcal{J}_m\big(\alpha_{SG}(H; M, C)\big).$$

Corollary 5. *The class of SGs is stable.*

Example. Figure 9(a) illustrates the generated SG from the DAG in Figure 5 using Algorithm 2 as well as the two subsets M_1 and C_1 of its node set. Figure 9(b) illustrates the SG generated by the algorithm from the SG in part (a).
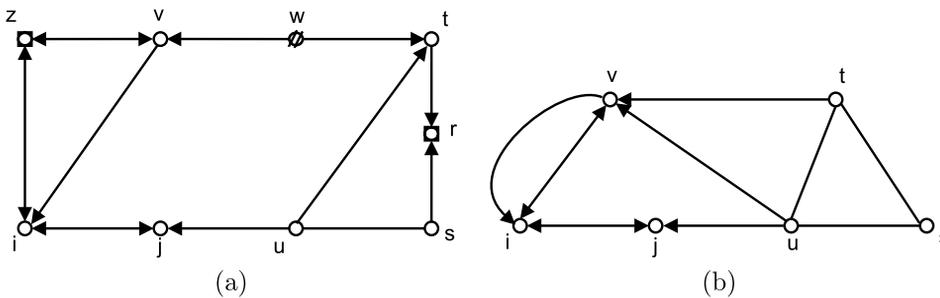
Figure 9. (a) The generated SG from the DAG in Figure 5, $\varnothing \in M_1$ and $\square \in C_1$. (b) The generated SG from the SG in (a).

Figure 10. (a) An AG. (b) An SG that is not ancestral.

7. Ancestral graphs

An *ancestral graph* (AG) is a simple mixed graph that has the following properties for every node i:

1. $i \notin \mathrm{an}(\mathrm{pa}(i) \cup \mathrm{sp}(i))$;
2. If $\mathrm{ne}(i) \neq \varnothing$, then $\mathrm{pa}(i) \cup \mathrm{sp}(i) = \varnothing$.

This means that there is no arrowhead pointing to a line and there is no direction-preserving cycle, and there is no arc with one endpoint that is an ancestor of the other endpoint in the graph.

AGs are obviously a subclass of SGs, and therefore RGs. Figure 10 illustrates an SG that is not ancestral. (Because of an arc with one endpoint that is an ancestor of the other.)

7.1. Generating ancestral graphs

A local algorithm to generate AGs. In [10], there is a method to generate AGs (in fact maximal AGs) globally by looking at the so-called inducing paths. Here we introduce an algorithm to generate AGs locally (after determining the ancestor set) by looking only for Vs after determining the ancestor set of the conditioning set.

Algorithm 3. $\alpha_{\mathrm{AG}}(H; M, C)$ (*Generating an AG from an ancestral graph H*):
  Start from H.

1. *Apply Algorithm 2.*
2. *Generate respectively an arrow from j to i or an arc between i and j for* V *$j \longrightarrow k \longleftrightarrow i$ or* V *$j \longleftrightarrow k \longleftrightarrow i$ when $k \in \mathrm{an}(i)$ if the arrow or the arc does not already exist.*
3. *Remove the arc between j and i in the case that $j \in \mathrm{an}(i)$, and replace it by an arrow from j to i if the arrow does not already exist.*

  Continually apply each step until it is not possible to apply the given step further before moving to the next step.

Figure 11 illustrates how to apply Algorithm 3 step by step to a DAG.
The map α_{AG} and its basic properties. Basic properties of Algorithm 3 and its corresponding function are analogous to the basic properties of RGs and SGs.

Proposition 6. *Graphs generated by Algorithm 3 are AGs.*
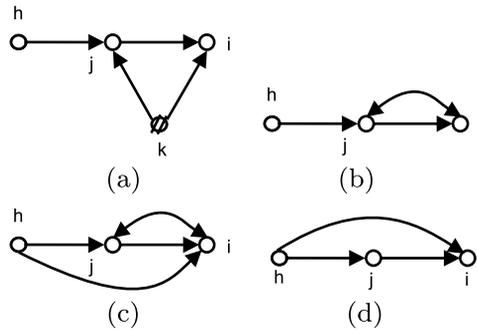
Figure 11. (a) A directed acyclic graph G, $\varnothing \in M$. (b) The generated graph after applying step 1. (c) The generated graph after applying step 2 for $\lor \langle h, j, i \rangle$. (d) The generated AG from G after applying step 3.

As before we consider α_{AG} as a function from the set of AGs and two subsets of their node set to the set of AGs.

Notice that by the extension of the generated AG to a maximal AG (as explained in [10]) the same maximal AG as that generated by the method explained in [10] is generated, and hence these two graphs induce the same independence model. This also explains the global interpretation of the algorithm. We will not give the details in this paper.

The map $\alpha_{SG.AG}$ and its properties. Notice that step 1 of Algorithm 3 generates an SG. Hence steps 2 and 3 of the algorithm generate an AG from an SG. We denote these two steps by $\alpha_{SG.AG}$, a function from $\mathcal{SG}$ to $\mathcal{AG}$.

Proposition 7. *It holds that $\alpha_{AG} = \alpha_{SG.AG} \circ \alpha_{SG}$.*

Surjectivity of α_{AG}. The following result shows that every member of $\mathcal{AG}$ can be generated by a DAG after marginalisation and conditioning.

Proposition 8. *The map $\alpha_{AG} : \mathcal{DAG} \to \mathcal{AG}$ is surjective.*

Proof. The result follows from Proposition 5, the fact that $\mathcal{AG} \subseteq \mathcal{SG}$, and if $H \in \mathcal{AG}$ then $\alpha_{SG.AG}(H) = H$. $\qquad\square$

7.2. Two necessary properties of AG-generating functions

Again we discuss the two important properties that we have proven for two other stable mixed graphs.

Well-definition of α_{AG}. Well-definition of α_{AG} is analogous to the well-definition of α_{RG} and α_{SG} as defined in the previous sections.

Theorem 5. *For an ancestral graph $H = (N, F)$ and disjoint subsets C, C_1, M, and M_1 of N,*

$$\alpha_{AG}\big(\alpha_{AG}(H; M, C); M_1, C_1\big) = \alpha_{AG}(H; M \cup M_1, C \cup C_1).$$
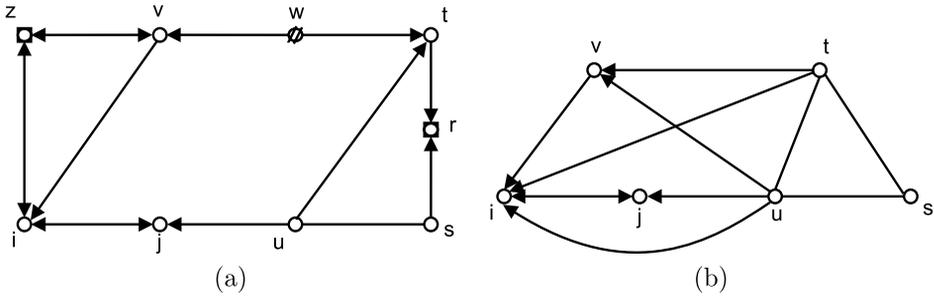
Figure 12. (a) The generated AG from the DAG in Figure 5, $\emptyset \in M_1$ and $\square \in C_1$. (b) The generated AG from the AG in (a).

Stability of the graphs generated by α_{AG}. Analogous to RGs and SGs, graphs generated by α_{AG} induce marginal and conditional independence models. An analogous result was proven in [10] for maximal AGs that were generated in that paper.

Theorem 6. *For an ancestral graph $H = (N, F)$ and disjoint subsets A, B, C, C_1, and M of N,*

$$A \perp_m B | C_1 \quad \text{in } \alpha_{\mathrm{AG}}(H; M, C) \quad \Longleftrightarrow \quad A \perp_m B | C \cup C_1 \quad \text{in } H.$$

Corollary 6 ([10]). *For an ancestral graph $H = (N, F)$ and M and C disjoint subsets of N,*

$$\alpha\big(\mathcal{J}_m(H); M, C\big) = \mathcal{J}_m\big(\alpha_{\mathrm{AG}}(H; M, C)\big).$$

Corollary 7. *The class of AGs is stable.*

Example. Figure 12(a) illustrates the AG generated from the DAG in Figure 5. Figure 12(b) illustrates the AG generated by the algorithm from the AG in part (a).

8. The relationship between different types of stable mixed graphs

Thus far, we have defined RGs (as a modification of MCGs), SGs, and AGs, and introduced algorithms to generate each of these from a graph of the same class or a DAG, and some algorithms that act between these classes. Despite the similarities of these definitions and generating algorithms of these different classes, as well as the parallel theory developed for these, it is of interest to investigate the exact relationship between these types of graphs.

Corresponding stable mixed graphs. When one starts from a DAG and generates different types of stable mixed graphs after marginalisation over and conditioning on two specific subsets of the node set of the DAG, the generated graphs must induce the same independence models. This leads us to the definition of corresponding stable mixed graphs. For a directed acyclic graph G

and two disjoint subsets of its node set M and C, graphs $\alpha_{\mathrm{RG}}(G; M, C)$, $\alpha_{\mathrm{SG}}(G; M, C)$, and $\alpha_{\mathrm{AG}}(G; M, C)$ are called, respectively, the *corresponding* RG, SG, and AG.

We observe that the corresponding RGs, SGs, and AGs of a DAG induce the same independence model. This fact, without being formulated in this way, was discussed in all three papers that define these graphs [5,10,13].

Proposition 9. *For a directed acyclic graph $G = (V, E)$ and disjoint subsets C and M of V,*

$$\mathcal{J}_m\big(\alpha_{\mathrm{RG}}(G; M, C)\big) = \mathcal{J}_m\big(\alpha_{\mathrm{SG}}(G; M, C)\big) = \mathcal{J}_m\big(\alpha_{\mathrm{AG}}(G; M, C)\big).$$

**Proof.** The result follows from Corollaries 1, 4 and 6. □

As it was shown, in SGs and AGs there are extra properties regarding the structure of the graph. We know that $\mathcal{AG} \subset \mathcal{SG} \subset \mathcal{RG}$. The corresponding AG to an SG can be generated by $\alpha_{\mathrm{SG.AG}}$ as outlined in Proposition 7. However, we cannot generate the corresponding SG to an RG by only knowing the RG and not the DAG (or the conditioning set of the DAG). For example, DAGs $\circ \longleftarrow \emptyset \longrightarrow \circ \longrightarrow \square \longleftarrow \emptyset$ and $\circ \longleftarrow \emptyset \longrightarrow \circ$, where $\emptyset \in M$ and $\square \in C$, give the same RG $\circ \longleftrightarrow \circ$ but different SGs $\circ \longleftarrow \circ$ and $\circ \longleftrightarrow \circ$ respectively. This is also true for AGs instead of SGs.

It is possible, however, to introduce an algorithm to generate SGs that induce the same independence model as the given RGs, by removing arrowheads pointing to a line or a node that is an ancestor of a node that is the endpoint of a line.

We have also seen that the image of generating functions is big enough to cover all graphs included in the set of the related type of stable mixed graphs, since the generating functions are surjective. On the other hand, it is easy to show that generating functions are not injective. Therefore, the relationship between the three types of stable mixed graphs is summarised by the diagram in Figure 13, in which one can only move towards the directions that arrows show.

9. Discussion on the use of different types of stable mixed graphs

By what we discussed, if G is a DAG with latent variables M and selection variables C then stable mixed graphs are a class of graphs that represent the independence model implied among
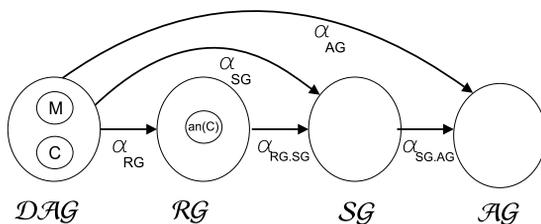
Figure 13. The relationship between $\mathcal{DAG}$, $\mathcal{RG}$, $\mathcal{SG}$ and $\mathcal{AG}$.

the remaining variables, conditional on the selection variables. However, each of the three types has been used in different contexts and for different purposes.

Why MCGs or RGs? MCGs have been introduced in order to straightforwardly deal with the problem of finding a class of graphs that is closed under marginalisation and conditioning by a simple process of deriving these from DAGs. In fact, the class of MCGs is much larger than what one really needs for representing independence models after marginalisation and conditioning. We have noted that only MCGs that are ribbonless can be generated this way.

Why SGs? The main goal of defining SGs is to trace the effects after marginalisation and conditioning, as will be explained shortly in this section. By using binary matrix representations of graphs, called edge matrices, and corresponding matrix operators [17], the edge matrix of a SG is obtained. It contains three types of edge matrices: those for solid lines, dashed lines (corresponding to arcs), and for arrows. In the family of joint Gaussian distributions, solid lines in concentration graphs correspond to concentration matrices, dashed lines in covariance graphs to covariance matrices and arrows to equation parameters in structural equation models.

SGs are used when the generating DAG is known. Despite knowledge on the structure of the generating DAG, SGs are still of interest in at least three situations: (1) For models with large number of unobserved and selection variables; and (2) for the comparison of models when one of them has unobserved or selection variables that are a subset of the unobserved or selection variables of the other; (3) for detecting some types of confounding as shown in [15] and as described briefly later.

Why AGs? The main goal of defining AGs is to represent and parametrise sets of distributions obeying Markov properties. Even though, we discussed the class of AGs in this paper to sustain a parallel theory to RGs and SGs, the class of maximal AGs possess some desired properties that AGs do not. These include the fact that under the Gaussian path diagram parametrisation the maximal AG only implies independence constraints, while a general AG implies other types of constraints. We will give a short discussion on maximality in this section. Maximal AGs are the simplest structures that capture the modified independence model, and are also of interest when the generating DAG is not known, but a set of conditional independencies is known. In the Gaussian case, maximal AGs are identified. In contrast to DAG models with hidden variable, the models are curved exponential families [10], and conditional fitting algorithm for maximum likelihood estimation exists [3].

Maximal stable mixed graph. A graph G is called *maximal* if by adding any edge to G the independence model induced by $\perp_m$ changes (gets smaller). Therefore, in maximal graphs, every missing edge corresponds to at least one independence statement in the induced independence model. This leads to validity of a so-called pairwise Markov property.

In [10], maximality of the subclass of AGs was studied. This result also holds for RGs and says that a ribbonless graph H is maximal if and only if H does not contain any *primitive inducing paths*, which are paths of form $\langle j, q_1, q_2, \ldots, q_p, i \rangle$, on which $i \nsim j$ and for every n, $1 \leq n \leq p$, q_n is a collider on the path and $q_n \in \mathrm{an}(\{i\} \cup \{j\})$. We shall not give the details in this paper.

Therefore, to generate a maximal stable mixed graph from a stable mixed graph one should repeatedly generate arrows from j to i for primitive inducing paths between non-adjacent i and j where there is no arrowhead pointing to j, and generate arcs between i and j for primitive inducing paths between non-adjacent i and j where there are arrowheads pointing to i and j. Notice that by applying this algorithm after the generating algorithms one can generate a maximal AG, SG, or RG.

As discussed, maximal AGs possess many desired properties that AGs do not. For SGs, it is conjectured that maximal SGs possess the same statistical properties that both maximal AGs and SGs do possess. To show this, further work is needed.

The structure of different types of stable mixed graphs. If we suppose that stable mixed graphs are only used to represent the independence model after marginalisation and conditioning, then we can consider all types as equally appropriate. The question then will be reduced to how simple or fast generating a type of graph is. We have seen that AGs have the simplest structure among the three types of stable mixed graphs, and RGs are the most complex. Therefore, as we have also seen, it is more complex to generate an AG than to generate an SG, and to generate an SG than to generate an RG. On the other hand, the simpler structure allows a faster way of checking independence statements. Hence, it is a tradeoff that depends on the relative size of the marginalisation and conditioning sets in graphs.

When generating stable mixed graphs from DAGs, one always loses some information in order to obtain a simpler structure in stable mixed graphs. RGs have lost the least information among the three types of stable mixed graphs, while AGs the most. Here we discuss the lost information in the context of regression analysis.

Multivariate regression and stable mixed graphs. The problem of constructing stable mixed graphs was originally posed by [16] in the context of multivariate statistics based on regression analysis. In such literature, the DAG model is defined by sequences of univariate recursive regressions, called a *linear triangular system* by [14], that is, for $i = 1, \ldots, d_N - 1$, each single response variable Y_i is regressed on $Y_{\mathrm{pa}(i)}$, where the parents of i are a subset of $\{i+1, \ldots, d_N\}$. Linear triangular systems can be written as $AY = \varepsilon$, where A is an upper-triangular matrix with unit diagonal elements, and ε is a vector of zero mean and uncorrelated random variables, called *residuals*. Here the nonzero regression coefficient of Y_i on Y_j can be attached the arrow from j to i in the DAG and is called the *direct effect* of Y_j on Y_i; see [1].

In particular, for linear triangular systems, RGs alert to distortions due to so-called over-conditioning via multiple edges consisting of a line and an arrow. Over-conditioning arises by conditioning on a variable that is a response of two variables, one of which itself is a response to the other one.

For example, in Figure 14, the generating process is given by three linear equations,

$$Y_1 = \beta Y_2 + \delta Y_3 + \varepsilon_1, \qquad Y_2 = \gamma Y_3 + \varepsilon_2, \qquad Y_3 = \varepsilon_3,$$

where each residual ε_i has mean zero and is uncorrelated with the explanatory variables on the right-hand side of the equation.

By conditioning on Y_1, the conditional dependence of Y_2 on only Y_3 is obtained, which consists of the direct effect γ and an indirect effect of Y_2 on Y_3 via Y_1. This may be seen by direct calculation, assuming that the residuals ε_i have a Gaussian distribution, which leads to

$$E(Y_2|Y_3) = \left(\gamma - \{(1 - \gamma^2)/(1 - \rho_{13}^2)\}\beta\rho_{13}\right)Y_2, \qquad \text{where } \rho_{13} = \delta + \beta\gamma.$$

Thus, the direct effect γ is distorted by $-\{(1 - \gamma^2)/(1 - \rho_{13}^2)\}\beta\rho_{13}$. The potential presence of this distortion is represented in (b) by the addition of an arrow.

In addition, the existence of multiple edges with an arrow and an arc, and arcs with one endpoint ancestor of the other, which are not permissible in AGs, respectively, alerts distortions due to so-called direct and indirect *confounding*.
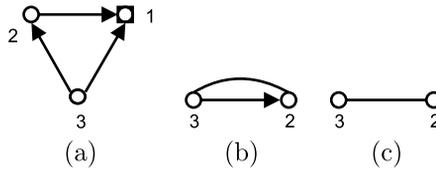
Figure 14. (a) A directed acyclic graph G with node 1 to be conditioned on. (b) The RG generated from G. (c) The SG or AG generated from G.

With the same generating process as explained for Figure 14, by integrating out Y_3 in Figure 15, the conditional dependence of Y_1 on only Y_2 is obtained, which consists of the direct effect β and an indirect effect of Y_1 on Y_2 via Y_3. This leads to

$$E(Y_1|Y_2) = (\beta + \delta\gamma)Y_2.$$

Thus, the direct effect β is distorted by $\delta\gamma$. The potential presence of this distortion is represented in (b) by the addition of an arc. This example indicates a distortion due to direct confounding; see [15]. Indirect confounding was also studied in [15] for marginalising only over a full set of background variables and also in [13] more generally relating SGs to corresponding maximal AGs.

Appendix: Proofs

Here we present the proof of lemmas, propositions, and theorems of this paper, but first we introduce some observations that are used in our proofs as the following lemmas.

Lemma 2. *If $i \in \mathrm{an}(j)$ in $\alpha_{\mathrm{RG}}(H; M, C)$, then in H one of the following holds: (1) $i \in \mathrm{an}(j)$; (2) i or a descendant of i is the endpoint of a line; (3) $i \in \mathrm{an}(C)$.*

Proof. We know that there is a direction-preserving path $\pi = \langle i = i_0, i_1, \ldots, i_p = j \rangle$ in $\alpha_{\mathrm{RG}}(H; M, C)$. Consider the $i_0 i_1$-edge. By Lemma 1 in H given M and C, there is an m-connecting path between i_0 and i_1, on which there is no arrowhead pointing to i_0. One can observe that if this path is not a direction-preserving path then one of the following holds: (1) i_0
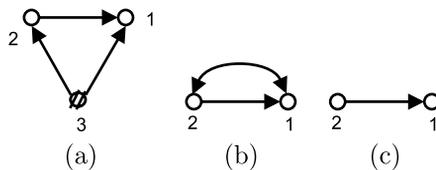
Figure 15. (a) A directed acyclic graph G with node 3 to be marginalised over. (b) The SG generated from G. (c) The AG generated from G.

is an ancestor of a collider node on the path, which is in $C \cup \mathrm{an}(C)$. Hence, $i \in \mathrm{an}(C)$; (2) i_0 is the endpoint of a line or an ancestor of a node that is the endpoint of a line on the path. If (1) or (2) holds, then we are done, hence assume that $i_0 \in \mathrm{an}(i_1)$. By the same argument and by induction along the nodes of π, we conclude the result. □

Lemma 3. *For i and j outside $M \cup C$, if $i \in \mathrm{an}(j)$ in H then one of the following holds*: (1) $i \in \mathrm{an}(j)$ in $\alpha_{\mathrm{RG}}(H; M, C)$; (2) $i \in \mathrm{an}(C)$ in H.

Proof. We know that there is a direction-preserving path $\pi = \langle i = i_0, i_1, \ldots, i_p = j \rangle$ in H. Consider the $i_0 i_1$-edge. We now have three cases: (1) If $i_1 \in C$, then $i \in \mathrm{an}(C)$ in H and we are done. (2) If $i_1 \in M$, then Algorithm 1 generates an arrow from i_0 to i_2. (3) If $i_1 \notin M \cup C$, then $i_0 \in \mathrm{an}(i_2)$ in $\alpha_{\mathrm{RG}}(H; M, C)$. By the same argument and by induction along the nodes of π, we conclude the result. □

The following lemma deals with the *concatenation* of m-connecting paths. We shall not give the details of the proof here; see [11].

Lemma 4. *In an RG, suppose that given M and C there are m-connecting paths $\langle i = i_0, i_1, \ldots, i_n, h \rangle$ between i and h and $\langle j = j_0, j_1, \ldots, j_m, h \rangle$ between h and j. In this case, there is an m-connecting path given M and C between i and j if one of the following holds*:

(a1) $\langle i_n, h, j_m \rangle$ *is collider and $h \in C \cup \mathrm{an}(C)$;*
(a2) $i_n = j_m$ *with arrowhead pointing to h on the $i_n h$-edge and $h \in C \cup \mathrm{an}(C)$;*
(b1) $\langle i_n, h, j_m \rangle$ *is non-collider and $h \in M$;*
(b2) $i_n = j_m$ *with no arrowhead pointing to h on the $i_n h$-edge and $h \in M$.*
(c1) $\langle i_n, h, j_m \rangle$ *is collider and h or a descendant of h is the endpoint of a line or a direction-preserving cycle;*
(c2) $i_n = j_m$ *with arrowhead pointing to h on the $i_n h$-edge and h or a descendant of h is the endpoint of a line or a direction-preserving cycle.*

Proof of Proposition 1. Graphs generated by Algorithm 1 have obviously three desired types of edges and are loopless.

Now suppose, for contradiction, that there is a ribbon $\langle i, h, j \rangle$ in a generated graph $\alpha_{\mathrm{RG}}(H; M, C)$. By Lemma 1 in H given M and C, there are m-connecting paths $\pi_1 = \langle i = i_0, i_1, \ldots, i_n, h \rangle$ between i and h and $\pi_2 = \langle j = j_0, j_1, \ldots, j_m, h \rangle$ between h and j such that there are arrowheads at h on both i_n- and $j_m h$-edges. (Notice that it is possible that $i_n = j_m$ and it is also possible that $i_n = i$ or $j_m = j$ in H.)

We also know that, in $\alpha_{\mathrm{RG}}(H; M, C)$, the node h is the endpoint of a line or on a direction-preserving cycle or there is a direction-preserving path $\pi = \langle h = h_0, h_1, \ldots, h_p = k \rangle$ from h to k such that k is the endpoint of a line or on a direction-preserving cycle. Now we consider two cases. In case I, we suppose that such a π does not exist and in case II we suppose that such a π exists.

Case I. In case I.1, we suppose that h is the endpoint of a line and in case I.2 we suppose that h is on a direction-preserving cycle.

Case I.1. Suppose that h is the endpoint of an hl-line in $\alpha_{\text{RG}}(H; M, C)$. By Lemma 1 in H given M and C, there is an m-connecting path between h and l, on which there is no arrowhead pointing to h or l. One can observe that h is an ancestor of either (1) a collider node on the path or (2) a node that is the endpoint of a line on the path. Thus, we have the two following cases:

(1) If h is an ancestor of a collider node t, then $h \in \text{an}(C)$ in H since $t \in C \cup \text{an}(C)$. Hence, by Lemma 4(a), there is an m-connecting path given M and C between i and j in H.

(2) If h is an ancestor of a node that is the endpoint of a line on the path, then by Lemma 4(c) there is an m-connecting path given M and C between i and j in H.

By Lemma 1, both cases imply that $i \sim j$ in $\alpha_{\text{RG}}(H; M, C)$ and the ij-edge is endpoint-identical to the m-connecting path. Therefore, $\langle i, h, j \rangle$ is not a ribbon, a contradiction.

Case I.2. Suppose that h is on a direction-preserving cycle in $\alpha_{\text{RG}}(H; M, C)$. By Lemma 2 in H one of the following holds: (1) $h \in \text{an}(h)$; (2) h or a descendant of h is the endpoint of a line; (3) $h \in \text{an}(C)$. Cases (2) and (3) lead to contradiction as explained in case I.1. Therefore, suppose that h is on a direction-preserving cycle in H. This by Lemma 4(c) implies that there is an m-connecting path given M and C between i and j in H, which implies that $\langle i, h, j \rangle$ is not a ribbon. This is a contradiction.

Case II. By Lemma 2 in H one of the following holds: (1) $h \in \text{an}(k)$; (2) h or a descendant of h is the endpoint of a line; (3) $h \in \text{an}(C)$. Cases (2) and (3) lead to contradiction as explained in case I.1. Hence, it holds that $h \in \text{an}(k)$ in H. This together with the same argument as that of case I (for k instead of h) leads to a contradiction. □

Proof of Lemma 1. ($\Rightarrow$) If an edge between i and j in $\alpha_{\text{RG}}(H; M, C)$ does not exist in H, then it has been generated by certain intermediate graphs that have each been generated by adding one edge to the previous graph by one of the steps of Table 1. We denote these graphs by the sequence $\langle H = H_0, H_1, \ldots, H_n, \alpha_{\text{RG}}(H; M, C) \rangle$, where H_n is the last step before removing M and C.

We prove by reverse induction on p that in all H_p, $0 \leq p \leq n$, between i and j there exists a path on which non-collider inner nodes are in M and collider inner nodes or their descendants are either in C or the endpoint of a line. For $p = n$, there is obviously an edge between i and j. We show that if there is such a path in H_r then we can find the same type of path between i and j in H_{r-1}.

If all edges along the path exist in H_{r-1}, then we should check that a collider node that is an ancestor of a member of C or an ancestor of a node that is the endpoint of a line in H_r is an ancestor of a member of C or an ancestor of a node that is the endpoint of a line in H_{r-1}. If an arrow has been generated along the direction-preserving path in H_r, then it has been generated by the Vs $\langle i', m, j' \rangle$ of the first three steps or the V $\langle i', s, j' \rangle$ of step 8 of Table 1. If it is step 1, then we can replace the $i'j'$-arrow by $\langle i', m, j' \rangle$ to obtain a direction-preserving path. If it is steps 2 or 3, then node j' is the endpoint of a line and we are done. If it is step 8 then, since $s \in C \cup \text{an}(C)$, the inner node of the V is in $\text{an}(C)$ and we are done.

Thus, suppose that an $i'j'$-edge along the m-connecting path is the edge that has been generated by this step. This has been generated by one of Vs of Table 1. Since in all cases the V is endpoint-identical to the $i'j'$-edge, and since all inner nodes of the non-collider Vs are in M and all inner nodes of the collider Vs are in $C \cup \text{an}(C)$, by placing the V instead of the $i'j'$-edge on the path, we still get a path whose non-collider inner nodes are in M and either whose collider

inner nodes are in $C \cup \text{an}(C)$ or whose collider nodes or a descendant of them are the endpoint of a line, as required.

Therefore, by reverse induction, there exists a path, as described above, in H. However, since H is ribbonless, the path cannot contain a collider $\mathsf{V} \langle i', h, j' \rangle$ such that h or a descendant of h is the endpoint of a line unless $i' \sim j'$ and the $i'j'$-edge is endpoint-identical to $\langle i', h, j' \rangle$. In this case, the $i'j'$-edge can be used instead of $\langle i', h, j' \rangle$ and, by induction, we obtain an m-connecting path given M and C between i and j.

The fact that in Table 1 the Vs are endpoint-identical to the generated $i'j'$-edges implies that all discussed paths in each H_p are endpoint-identical.

($\Leftarrow$) Suppose that there is an m-connecting path π given M and C between i and j in H_k, $0 \le k \le n-1$. We prove as long as $r > 0$ if there is an m-connecting path given M and C between i and j in H_k with r inner nodes then there is an m-connecting path given M and C between i and j in H_{k+1} with $r-1$ inner nodes. By induction we will finally obtain an m-connecting path between i and j without inner nodes, that is, an edge between j and i.

Consider an m-connecting path given M and C between i and j in H_k with $r > 0$ inner nodes. Consider an arbitrary inner node on the path. If this node is collider, then one of the Vs 8, 9, or 10 of Table 1 is employed to generate an edge between the endpoints of the V in H_{k+1}. Since the generated edge is endpoint-identical to the V, one can use the generated edge instead of the V to obtain an m-connecting path with $r-1$ inner nodes. If the arbitrary node is non-collider then one of the other Vs of Table 1 is used.

It is easy to check that the generated edges are endpoint-identical to the m-connecting paths in the final graph. This implies the result. $\qquad\square$

Proof of Proposition 2. Let $H = (N_1, F_1) \in \mathcal{RG}$. We generate a directed graph $G = (V, E)$ from H as follows: We leave arrows that are not on any direction-preserving cycle unchanged. For direction-preserving cycles, instead of one arbitrary arrow from j to i on the cycle we place $j \longrightarrow \square \longleftarrow \varnothing \longrightarrow i$, where $\varnothing \in M$ and $\square \in C$, and leave all other arrows unchanged. Instead of an arc between j and i we place a V between j and i with inner source node in M. Instead of a line between j and i we place a V between j and i with inner collider node in C. The graph G is obviously a directed graph. Furthermore, all newly generated nodes have degree 2 and the direction of arrows changes on them, hence these cannot be on any direction-preserving cycle. In addition, if i and j are in N_1 and $i \sim j$ in G then $i \in \text{pa}(j)$ or $j \in \text{pa}(i)$ in H. Therefore, the existence of a direction-preserving cycle in G implies the existence of the same direction-preserving cycle in H. But by the nature of the construction of G we know that direction-preserving cycles in H do not make direction-preserving cycles in G, hence G is acyclic.

We should prove that $\alpha_{\text{RG}}(G; M, C) = H$. Let $\alpha_{\text{RG}}(G; M, C) = (N_2, F_2)$. Obviously $N_1 = N_2$. Suppose that $i \sim j$ ($j \in \text{pa}(i)$, $j \in \text{sp}(i)$, or $j \in \text{ne}(i)$) in H. Therefore, we have the active alternating path or one of the Vs between i and j that by Algorithm 1 forms exactly the same type of edge in $\alpha_{\text{RG}}(G; M, C)$.

Conversely, suppose that $i \sim j$ ($j \in \text{pa}(i)$, $j \in \text{sp}(i)$, or $j \in \text{ne}(i)$) in $\alpha_{\text{RG}}(G; M, C)$. By Lemma 1 we know that there is an endpoint-identical m-connecting path given M and C in G. Consider a shortest endpoint-identical m-connecting path π. Since in G there is no transition node in M, π is active alternating with respect to M and $C \cup \text{an}(C)$. If π has no collider node in $\text{an}(C) \setminus C$, then by the nature of the construction of G we know that it has two edges (if both

endpoints are children or parents) or three (if it is from j to i on a direction-preserving cycle) and that it has been generated by an edge (arrow, arc, or line) in H. Suppose, for contradiction, that there is a collider node $i \in \mathrm{an}(C) \setminus C$ on π. We have that $i \in N_1$, and by the process of generating a DAG explained here, the only place that a node in C has been generated is by a line or an arrow on a direction-preserving cycle in H. Therefore, $i \in \mathrm{an}(k)$ for a node k that is the endpoint of a line or is on a direction-preserving cycle. Hence, H contains a ribbon, or the endpoints of the collider V with i as inner node are adjacent by an endpoint-identical edge. The former contradicts that H is ribbonless, and the latter contradicts that π is shortest. $\qquad\square$

Proof of Theorem 1. ($\Rightarrow$) If there is an ij-edge in $\alpha_{\mathrm{RG}}(\alpha_{\mathrm{RG}}(H; M, C); M_1, C_1)$, then by Lemma 1, there is an m-connecting path $\pi = \langle i = i_0, i_1, \ldots, i_{n-1}, i_n = j \rangle$ between i and j given M_1 and C_1 in $\alpha_{\mathrm{RG}}(H; M, C)$ that is endpoint-identical to the edge.

For the V $\langle i, i_1, i_2 \rangle$ on π, again by Lemma 1, given M and C there are m-connecting paths π_1 between i and i_1 and π_2 between i_1 and i_2 in H. These paths can be considered m-connecting given $M \cup M_1$ and $C \cup C_1$ and are endpoint-identical to the edges. This implies that if i_1 is collider (or non-collider) on π then on the concatenation of π_1 and π_2 it remains collider (or non-collider), or that there is an arrowhead pointing to it (or no arrowhead pointing to it) from a joint node on π_1 and π_2. We know that if i_1 is non-collider then it is in M_1 and if it is collider then it is in $C_1 \cup \mathrm{an}(C_1)$ in $\alpha_{\mathrm{RG}}(H; M, C)$. If $i_1 \in \mathrm{an}(C_1)$ in $\alpha_{\mathrm{RG}}(H; M, C)$ then, by Lemma 2, one of the following holds in H: (1) it is in $\mathrm{an}(C_1)$; (2) it is in $\mathrm{an}(C)$; (3) it is the endpoint of line or an ancestor of a node that is the endpoint of a line. Therefore, by Lemma 4, there is an m-connecting path between i and i_2 given $M \cup M_1$ and $C \cup C_1$ in H, which is endpoint-identical to the V. By induction along π there is an m-connecting path between i and j given $M \cup M_1$ and $C \cup C_1$ in H, which is endpoint-identical to the ij-edge. Therefore, by Lemma 1 there is the same type of ij-edge in $\alpha_{\mathrm{RG}}(H; M \cup M_1, C \cup C_1)$.

($\Leftarrow$) If there is an edge between i and j in $\alpha_{\mathrm{RG}}(H; M \cup M_1, C \cup C_1)$ then, by Lemma 1, there is an m-connecting path π given $M \cup M_1$ and $C \cup C_1$ in H that is endpoint-identical to the ij-edge. All inner nodes of π are either in $M \cup C \cup \mathrm{an}(C)$ or $M_1 \cup C_1 \cup \mathrm{an}(C_1)$. Therefore, π can be partitioned into m-connecting subpaths given M and C and single nodes, where the endpoints of subpaths and single nodes are in $M_1 \cup C_1 \cup \mathrm{an}(C_1)$. Therefore, by Lemma 1, in $\alpha_{\mathrm{RG}}(H; M, C)$ the endpoints of each of the discussed subpaths of π are connected by an edge that is endpoint-identical to the subpath.

In addition, for each collider V $\langle l, k, h \rangle$, where $k \in \mathrm{an}(C_1)$ in H and by Lemma 3, one of the following holds: (1) $k \in \mathrm{an}(C)$ in H; (2) $k \in \mathrm{an}(C_1)$ in $\alpha_{\mathrm{RG}}(H; M, C)$. Case (1) implies that there is an endpoint-identical lh-edge to the V in $\alpha_{\mathrm{RG}}(H; M, C)$, which can be used instead of $\langle l, k, h \rangle$ to generate an m-connecting path.

Hence in $\alpha_{\mathrm{RG}}(H; M, C)$, there is an m-connecting path given M_1 and C_1 between i and j, which is endpoint-identical to π. Therefore, again by Lemma 1, there is the same type of ij-edge in $\alpha_{\mathrm{RG}}(\alpha_{\mathrm{RG}}(H; M, C); M_1, C_1)$. $\qquad\square$

Proof of Theorem 2. To prove the result, it is enough to show that, between nodes i and j outside $C \cup C_1 \cup M$, there is an m-connecting path given $C \cup C_1$ in H if and only if there is an m-connecting path given C_1 in $\alpha_{\mathrm{RG}}(H; M, C)$.

($\Rightarrow$) Suppose that between i and j there is an m-connecting path given $C \cup C_1$ in H. This path can be partitioned into m-connecting subpaths given C and single nodes, where the endpoints of subpaths and single nodes are colliders in $C_1 \cup \mathrm{an}(C_1)$. In addition, for each collider $\vee \langle l, k, h \rangle$, where $k \in \mathrm{an}(C_1)$ in H and by Lemma 3, one of the following holds: (1) $k \in \mathrm{an}(C)$ in H; (2) $k \in \mathrm{an}(C_1)$ in $\alpha_{\mathrm{RG}}(H; \varnothing, C)$. Case (1) implies that there is an endpoint-identical lh-edge to the $\vee$ in $\alpha_{\mathrm{RG}}(H; \varnothing, C)$, which can be used instead of $\langle l, k, h \rangle$ to generate an m-connecting path.

Hence by Lemma 1, in $\alpha_{\mathrm{RG}}(H; \varnothing, C)$ there is an m-connecting path given C_1 between i and j. The inner non-collider nodes on this path are either in M or in $N \setminus (M \cup C \cup C_1)$. On this path there are subpaths with only non-collider inner nodes in M, that is, m-connecting subpaths given M and $\varnothing$. By Theorem 1 after marginalisation over M, $\alpha_{\mathrm{RG}}(H; M, C)$ is obtained, in which, by Lemma 1, the endpoints of each of such subpaths are connected by an edge that is endpoint-identical to the subpath. In addition, if a collider node is in $\mathrm{an}(C_1)$ in $\alpha_{\mathrm{RG}}(H; \varnothing, C)$, then by Lemma 3 (since the conditioning set is empty and case (2) of the lemma does not hold), the collider node remains in $\mathrm{an}(C_1)$ in $\alpha_{\mathrm{RG}}(H; M, C)$. Therefore, between i and j there is an m-connecting path given C_1 in $\alpha_{\mathrm{RG}}(H; M, C)$.

($\Leftarrow$) Suppose that between i and j there is an m-connecting path π given C_1 in $\alpha_{\mathrm{RG}}(H; M, C)$. By Lemma 1, for each edge of π, there is an endpoint-identical m-connecting path given M and C in H, which is obviously an m-connecting path given C. On the concatenation of these paths and for the endpoints of the paths we have the two following cases: (1) When the endpoints are non-collider or there is no arrowhead at them on the concatenation, they are non-collider on π and therefore outside $M \cup C \cup C_1$; (2) When the endpoints are collider or there is an arrowhead at them, they are collider on π and therefore in $C_1 \cup \mathrm{an}(C_1)$. Therefore, by Lemma 4, there is an m-connecting path given $C \cup C_1$ in H. $\qquad\square$

Proof of Proposition 3. Firstly, the graph generated by the algorithm has only the three desired types of edges and no multiple edge of the same type; therefore, it generates a mixed graph.

Let H be the input summary graph and $H_0 = \alpha_{\mathrm{RG}}(H; M, C)$ be the generated graph after applying step 1 of Algorithm 2. Here in part I we prove that there is no arrowhead pointing to lines, and in part II we prove that there is no direction-preserving cycle in the generated graph.

Part I. If, for contradiction, there is an arrowhead at i (on an ik-edge) and i is the endpoint of an ij-line in the generated graph then we have the following cases: (1) The ij-line is an arrow from i to j in H_0; (2) the ij-line is an arrow from j to i in H_0; (3) the ij-line is an arc in H_0; (4) the ik-edge is an arrow from k to i in the generated graph and an arc in H_0; (5) the ki- and ij-edges are the same in H_0.

(1) We have that $j \in \mathrm{an}(C)$ in H. Hence by Lemma 2 we have one of the two following cases in H: (a) i or a descendant of i is the endpoint of a line, which, since H is a summary graph, is a contradiction; (b) $i \in \mathrm{an}(C)$, which by the algorithm implies that in the generated graph there is no arrowhead at i on the ik-edge, again a contradiction.

(2), (3) We have that $i \in \mathrm{an}(C)$ in H, which by the algorithm implies that in the generated graph there is no arrowhead at i on the ik-edge, a contradiction.

(4), (5) We have that $\langle k, i, j \rangle$ still has an arrowhead pointing to a line in H_0. By Lemma 1 there is an arrowhead at i in H, hence the line has been generated by the algorithm. Again by Lemma 1 we observe that one of the following holds: (a) i or a descendant of i is the endpoint of a line, which, since H is a summary graph, is a contradiction; (b) $i \in \mathrm{an}(C)$, which by the

algorithm implies that in the generated graph there is no arrowhead pointing to i on the ik-edge, again a contradiction.

Part II. There is also no direction-preserving cycles in the generated graph: If, for contradiction, there is a direction-preserving cycle in the generated graph then at least one arrow, say from k to l, should be generated by the algorithm since there is no direction-preserving cycle in H. This arrow can be generated either by step 1, or by step 2 as an arc replaced by an arrow. If the kl-arrow is generated by step 2 then we have that $k \in \text{an}(C)$ in H. Therefore, there are no arrowheads pointing to k in the generated graph, which means that k cannot be on a direction-preserving cycle, a contradiction.

Therefore, we can assume that the direction-preserving cycle exists in H_0. Since there are no arrowheads pointing to lines in H and H does not contain a direction-preserving cycle, Lemma 2 implies that a node (and therefore all nodes) of the cycle are in $\text{an}(C)$. Hence, the arrows turn into lines in the generated graph, a contradiction. □

Proof of Proposition 5. We know that $\mathcal{SG} \subseteq \mathcal{RG}$. Notice that H is a summary graph. We know that, for the generated directed acyclic graph G, explained in (a), $\alpha_{\text{RG}}(G; M, C) = H$. Suppose, for contradiction, that step 2 of Algorithm 2 changes the graph. Thus a node with an arrowhead pointing to is in $\text{an}(C)$, which implies that it is an ancestor of a node that is the endpoint of a line or on a direction-preserving cycle in H, a contradiction. Therefore, $\alpha_{\text{SG}}(G; M, C) = H$. □

Proof of Theorem 3. We show that there is an edge between i and j in $\alpha_{\text{SG}}(H; M \cup M_1, C \cup C_1)$ if and only if there is the same type of edge in $\alpha_{\text{SG}}(\alpha_{\text{SG}}(H; M, C); M_1, C_1)$. For this purpose for summary graphs, it is enough to prove that there is an edge between i and j with arrowhead pointing to j in $\alpha_{\text{SG}}(H; M \cup M_1, C \cup C_1)$ if and only if there is an edge between i and j with arrowhead pointing to j in $\alpha_{\text{SG}}(\alpha_{\text{SG}}(H; M, C); M_1, C_1)$.

($\Rightarrow$) Suppose that in $\alpha_{\text{SG}}(H; M \cup M_1, C \cup C_1)$ there is an ij-edge with arrowhead pointing to j. We have that $j \notin \text{an}(C \cup C_1)$ in H and in $\alpha_{\text{RG}}(H; M \cup M_1, C \cup C_1)$ there is an ij-edge with arrowhead pointing to j. By Lemma 1, there is an m-connecting path between i and j given $M \cup M_1$ and $C \cup C_1$ in H with arrowhead pointing to j. By what we showed before in the proof of Theorem 1 in $\alpha_{\text{RG}}(H; M, C)$ there is an m-connecting path π between i and j given M_1 and C_1 with arrowhead pointing to j.

Now notice that by step 2 of Algorithm 2 non-collider nodes remain non-collider. In addition, if a collider $\vee \langle h, k, l \rangle$ on π turns into non-collider then $k \in \text{an}(C)$ and therefore by step 1 of the algorithm there is an endpoint-identical hl-edge that can be used instead of the $\vee$ to generate an m-connecting path. Moreover, if the collider node k is in $\text{an}(C_1)$, and on the direction-preserving path an arrow turns into a line then $k \in \text{an}(C)$ in H and once again there is an hl-edge to be used instead of the $\vee$ to establish an m-connecting path. Therefore, there is an m-connecting path between i and j given M_1 and C_1 in $\alpha_{\text{SG}}(H; M, C)$. We also have that since $j \notin \text{an}(C)$ in H, there is an arrowhead pointing j on the path.

Now by Lemma 1 in $\alpha_{\text{RG}}(\alpha_{\text{SG}}(H; M, C); M_1, C_1)$ there is an ij-edge with arrowhead pointing j. In addition, in $\alpha_{\text{SG}}(H; M, C)$, $j \notin \text{an}(C_1)$: This is because if, for contradiction, $j \in \text{an}(C_1)$ in $\alpha_{\text{SG}}(H; M, C)$ then, in $\alpha_{\text{RG}}(H; M, C)$, $j \in \text{an}(C \cup C_1)$. This by Lemma 2 and the fact that H is a summary graph implies that $j \in \text{an}(C \cup C_1)$ in H, a contradiction.

Therefore, since in $\alpha_{SG}(H; M, C)$, $j \notin \text{an}(C_1)$, there is an arrowhead pointing j on the ij-edge in $\alpha_{SG}(\alpha_{SG}(H; M, C); M_1, C_1)$.

($\Leftarrow$) Suppose that in $\alpha_{SG}(\alpha_{SG}(H; M, C); M_1, C_1)$ there is an ij-edge with arrowhead pointing to j. This implies that $j \notin \text{an}(C_1)$ in $\alpha_{SG}(H; M, C)$. In $\alpha_{RG}(\alpha_{SG}(H; M, C); M_1, C_1)$ there is also an ij-edge with arrowhead pointing to j. By Lemma 1, in $\alpha_{SG}(H; M, C)$ there is an m-connecting path π given M_1 and C_1 between i an j with arrowhead pointing to j on the path. This implies that $j \notin \text{an}(C)$ in H.

By step 2 of the algorithm, collider nodes on π in $\alpha_{SG}(H; M, C)$ are colliders in $\alpha_{RG}(H; M, C)$. In addition, if a non-collider V $\langle h, k, l \rangle$ on π is collider in $\alpha_{RG}(H; M, C)$ then $k \in \text{an}(C)$ in H and therefore by step 1 of the algorithm there is an endpoint-identical hl-edge that can be used instead of the V to generate an m-connecting path in $\alpha_{RG}(H; M, C)$. Moreover, if the collider node k on π is in $\text{an}(C_1)$, and on the direction-preserving path an arrow is an arc in $\alpha_{RG}(H; M, C)$ then the collider node is in $\text{an}(C)$ in H and once again there is an hl-edge to be used instead of the V to establish an m-connecting path. Therefore, there is an m-connecting path between i and j given M_1 and C_1 in $\alpha_{RG}(H; M, C)$ with arrowhead pointing to j on the path. By what we showed before in the proof of Theorem 1 in H there is an m-connecting path between i and j given $M \cup M_1$ and $C \cup C_1$ with arrowhead pointing to j on the path.

Lemma 1 implies that in $\alpha_{RG}(H; M \cup M_1, C \cup C_1)$ there is an ij-edge with arrowhead pointing to j. We showed before that $j \notin \text{an}(C)$ in H. In addition, in H, $j \notin \text{an}(C_1)$: Suppose, for contradiction, that $j \in \text{an}(C_1)$ in H. This direction-preserving path is also direction-preserving in $\alpha_{RG}(H; M, C)$ unless possibly a node t on the path is in M or in C. In the former case one can skip t and obtain a direction-preserving path. In the latter case $j \in \text{an}(C)$ in H, which is not permissible. Therefore, in $\alpha_{RG}(H; M, C)$, $j \in \text{an}(C_1)$. Hence, since $j \notin \text{an}(C)$ in H, $j \in \text{an}(C_1)$ in $\alpha_{SG}(H; M, C)$, a contradiction.

Therefore, $j \notin \text{an}(C \cup C_1)$ in H, and the ij-edge has arrowhead pointing to j in $\alpha_{SG}(H; M \cup M_1, C \cup C_1)$. □

Proof of Theorem 4. By what we proved in Theorem 2 it is enough to show between i and j there is an m-connecting path given C_1 in $\alpha_{RG}(H; M, C)$ if and only if there is an m-connecting path given C_1 in $\alpha_{SG}(H; M, C)$.

($\Rightarrow$) Consider an m-connecting path given C_1 in $\alpha_{RG}(H; M, C)$ between i and j. There obviously exists an m-connecting path π given C_1 in $\alpha_{RG}(H; M, C)$. Now by step 2 of the algorithm non-collider nodes remain non-collider. In addition, if a collider V $\langle h, k, l \rangle$ on π turns into non-collider then $k \in \text{an}(C)$ and therefore by step 1 of the algorithm there is an endpoint-identical hl-edge generated that can be used instead of the V to generate an m-connecting path. Moreover, if the collider node k is in $\text{an}(C_1)$, and on the direction-preserving path an arrow turns into a line then $k \in \text{an}(C)$ in H and once again there is an hl-edge to be used instead of the V to establish an m-connecting path. Therefore, there is an m-connecting path between i and j given C_1 in $\alpha_{SG}(H; M, C)$.

($\Leftarrow$) Consider an m-connecting path π given C_1 in $\alpha_{SG}(H; M, C)$ between i and j. By step 2 of the algorithm, collider nodes on π in $\alpha_{SG}(H; M, C)$ are colliders in $\alpha_{RG}(H; M, C)$. In addition, if a non-collider V $\langle h, k, l \rangle$ on π is collider in $\alpha_{RG}(H; M, C)$ then $k \in \text{an}(C)$ in H and therefore by step 1 of the algorithm there is an endpoint-identical hl-edge that can be used instead of the V to generate an m-connecting path in $\alpha_{RG}(H; M, C)$. Moreover, if the collider node k

on π is in an(C_1), and on the direction-preserving path an arrow is an arc in $\alpha_{\mathrm{RG}}(H; M, C)$ then $k \in$ an(C) in H and once again there is an hl-edge to be used instead of the V to establish an m-connecting path. Therefore, there is an m-connecting path between i and j given C_1 in $\alpha_{\mathrm{RG}}(H; M, C)$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Proof of Proposition 6. To prove that the graph generated by Algorithm 3 is an AG, first notice that graphs generated by step 1 of Algorithm 3 are summary graphs. Therefore, since steps 2 and 3 do not generate any lines, it is enough to prove that steps 2 and 3 of the algorithm remove all subgraphs where there is an arc with one endpoint that is an ancestor of the other endpoint in the generated summary graph, and that these do not generate any direction-preserving cycles.

Step 3 of the algorithm removes all such subgraphs. Step 2 does not generate any direction-preserving cycles by adding an arrow to the graph: Consider the first iteration of the algorithm, where, for contradiction, a direction-preserving cycle is generated. If it is generated by generating an arrow from i to j then we know that there is $i \longrightarrow k \longleftrightarrow j$, where $k \in$ an(j). Denote the direction-preserving path from k to j by π_1 and the direction-preserving path from j to i which, together with the generated ij-arrow, establishes a direction-preserving cycle by π_2. It is seen that in the previous iteration of the algorithm $\langle \pi_2, k, \pi_1 \rangle$ is a direction-preserving cycle, a contradiction.

Step 3 does not generate any direction-preserving cycles by replacing an arc by an arrow: Consider the first iteration of the algorithm, where, for contradiction, a direction-preserving cycle is generated. If it is generated by replacing an ij-arc by an arrow from i to j, then we know there is a direction-preserving path π_1 from i to j. Denote the direction-preserving path from j to i which, together with the generated ij-arrow, establishes a direction-preserving cycle by π_2. It is seen that in the previous iteration of the algorithm $\langle \pi_1, \pi_2 \rangle$ is a direction-preserving cycle, a contradiction. $\qquad\qquad\qquad\qquad\qquad\square$

We use the following lemma to prove Theorem 5. For more descriptive proofs for the following results, see [11].

Lemma 5. *Let H be a summary graph and M and C be two subsets of its node set. It holds that* $\alpha_{\mathrm{AG}}(\alpha_{\mathrm{SG.AG}}(H); M, C) = \alpha_{\mathrm{AG}}(H; M, C)$.

Proof. There are two differences between H and $\alpha_{\mathrm{SG.AG}}(H)$: (1) For an ij-arc such that $j \in$ an(i) in H, there is an arrow from j to i replaced in $\alpha_{\mathrm{SG.AG}}(H)$; (2) for a primitive inducing path π between i and j in H, there is an endpoint-identical ij-edge in $\alpha_{\mathrm{SG.AG}}(H)$.

Notice that an(C) is the same in both graphs. After applying step 1 of Algorithm 2 (a part of step 1 of Algorithm 3), for each difference, the following occurs:

(1) This step of the algorithm may generate further differences for (1) if there is a kj-edge with an arrowhead pointing to j and $j \in C \cup$ an(C). In this case, there is a ki-edge in H. However, such an edge already exists in $\alpha_{\mathrm{SG.AG}}(H)$ since $\langle k, j, i \rangle$ in H generates an edge by $\alpha_{\mathrm{SG.AG}}$;

(2) This step of the algorithm may generate further differences for (2) if π has more than three nodes and one of the two following cases occurs: (a) there is an arrowhead pointing to j on π, there is a kj-edge with an arrowhead pointing to j, and $j \in C \cup$ an(C); (b) there is a kj-edge with no arrowhead pointing to j, and $j \in M$. In both cases, by using the $\langle i, j, k \rangle$-V a ki-edge is

generated in $\alpha_{SG.AG}(H)$. In H, by using the $\langle h, j, k \rangle$-V, where h is the node adjacent to j on π, a kh-edge is generated, which establishes an endpoint-identical primitive inducing path between i and k in $\alpha_{SG.AG}(H)$;

After applying step 2 of Algorithm 2, the following occurs: (1) This step of the algorithm may generate further differences for (1) if $j \in C \cup an(C)$. In this case, ij-arc in H turns into an arrow from j to i, which, however, already exists in $\alpha_{SG.AG}(H)$; (2) This step of the algorithm may generate further differences for (2) if any of the nodes on π, say l, is in $C \cup an(C)$. However, in H, by the previous step of the algorithm, the nodes adjacent to l on π have become adjacent, and established a shorter primitive inducing path between i and k (or j).

Hence, thus far, the differences between the two generated graphs are the same as the differences between H and $\alpha_{SG.AG}(H)$. Therefore, by applying steps 2 and 3 of Algorithm 3, the same graphs will be generated. $\square$

Proof of Theorem 5. Using Theorem 3, Proposition 7, and Lemma 5, we have the following:

$$\alpha_{AG}\big(\alpha_{AG}(H; M, C); M_1, C_1\big)$$
$$= \alpha_{AG}\big(\alpha_{SG.AG}(\alpha_{SG}(H; M, C)); M_1, C_1\big) = \alpha_{AG}\big(\alpha_{SG}(H; M, C); M_1, C_1\big)$$
$$= \alpha_{SG.AG}\big(\alpha_{SG}(\alpha_{SG}(H; M, C); M_1, C_1)\big) = \alpha_{SG.AG}\big(\alpha_{SG}(H; M \cup M_1, C \cup C_1)\big)$$
$$= \alpha_{AG}(H; M \cup M_1, C \cup C_1). \qquad \square$$

Proof of Theorem 6. By what we proved in Theorem 4, it is enough to show between A and B there is an m-connecting path given C_1 in $\alpha_{SG}(H; M, C)$ if and only if there is an m-connecting path given C_1 in $\alpha_{AG}(H; M, C)$.

Let $\langle \alpha_{SG}(H; M, C) = H_0, H_1, \ldots, H_m \rangle$ be intermediate graphs that have each been generated by adding one edge to the previous graph by step 2 of Algorithm 3. In addition, let $\langle H_m = H'_0, H'_1, \ldots, H'_n = \alpha_{AG}(H; M, C) \rangle$ be intermediate graphs that have each been generated by replacing one edge in the previous graph by step 3 of Algorithm 3.

Suppose that in the step between H_p and H_{p+1} an arrow from j to i or an arc between i and j for V $j \longrightarrow k \longleftrightarrow i$ or V $j \longleftrightarrow k \longleftrightarrow i$, when $k \in an(i)$, is generated. It holds that there is an m-connecting path between A and B given C_1 in H_p if and only if there is an m-connecting path between A and B given C_1 in H_{p+1}. This is because if $k \in C_1$ or one of the descendants of k on the direction-preserving path from k to i is in C_1 then the ij-edge and the V $j \longleftrightarrow k \longleftrightarrow i$ can be interchanged on the m-connecting path. If these nodes are not in C_1 then the ij-edge and the path made up by the jk-edge and the direction-preserving path from k to i can be interchanged. By induction in one direction and reverse induction in the other direction, we conclude that there is an m-connecting path given C_1 in $\alpha_{SG}(H; M, C)$ if and only if there is an m-connecting path given C_1 in H_m.

Now suppose that in the step between $H'_{p'}$ and $H'_{p'+1}$ an arrow from j to i has been replaced by an arc between j and i, where $j \in an(i)$. The only interesting case here is when there is an edge between j and another node l with arrowhead pointing to j. In this case, an edge has been already generated between l and i by step 2 of the algorithm. Therefore, again by induction in one direction and reverse induction in the other direction the result follows. $\square$

Acknowledgements

References

[1] Cox, D.R. and Wermuth, N. (1993). Linear dependencies represented by chain graphs. *Statist. Sci.* **8** 204–218, 247–283. With comments and a rejoinder by the authors. MR1243593

[2] Cox, D.R. and Wermuth, N. (1996). *Multivariate Dependencies*: *Models*, *Analysis and Interpretation*. *Monographs on Statistics and Applied Probability* **67**. London: Chapman & Hall. MR1456990

[3] Drton, M. and Richardson, T. (2004). Iterative Conditional Fitting for Gaussian Ancestral Graph Models. In *Proceedings of the Proceedings of the Twentieth Conference Annual Conference on Uncertainty in Artificial Intelligence* (*UAI*-04) 130–137. Arlington, VA: AUAI Press.

[4] Kiiveri, H., Speed, T.P. and Carlin, J.B. (1984). Recursive causal models. *J. Austral. Math. Soc. Ser. A* **36** 30–52. MR0719999

[5] Koster, J.T.A. (2002). Marginalizing and conditioning in graphical models. *Bernoulli* **8** 817–840. MR1963663

[6] Lauritzen, S.L. (1996). *Graphical Models*. *Oxford Statistical Science Series* **17**. New York: Clarendon Press. MR1419991

[7] Lauritzen, S.L., Dawid, A.P., Larsen, B.N. and Leimer, H.G. (1990). Independence properties of directed Markov fields. *Networks* **20** 491–505. MR1064735

[8] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*: *Networks of Plausible Inference*. *The Morgan Kaufmann Series in Representation and Reasoning*. San Mateo, CA: Morgan Kaufmann. MR0965765

[9] Pearl, J. and Wermuth, N. (1994). When can association graphs admit a causal interpretation? *Models and Data*, *Artificial Intelligence and Statistics* **4** 205–214.

[10] Richardson, T. and Spirtes, P. (2002). Ancestral graph Markov models. *Ann. Statist.* **30** 962–1030. MR1926166

[11] Sadeghi, K. (2012). Graphical representation of independence structures. Ph.D. thesis, Univ. Oxford.

[12] Studeny, M. (2005). *Probabilistic Conditional Independence Structures*. London, United Kingdom: Springer.

[13] Wermuth, N. (2011). Probability distributions with summary graph structure. *Bernoulli* **17** 845–879. MR2817608

[14] Wermuth, N. and Cox, D.R. (2004). Joint response graphs and separation induced by triangular systems. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **66** 687–717. MR2088296

[15] Wermuth, N. and Cox, D.R. (2008). Distortion of effects caused by indirect confounding. *Biometrika* **95** 17–33. MR2409712

[16] Wermuth, N., Cox, D.R. and Pearl, J. (1994). Explanation for multivariate structures derived from univariate recursive regressions. Technical Report 94(1), Univ. Mainz, Germany.

[17] Wermuth, N., Wiedenbeck, M. and Cox, D.R. (2006). Partial inversion for linear systems and partial closure of independence graphs. *BIT* **46** 883–901. MR2285213