Max-Planck-Institut für Mathematik in den Naturwissenschaften Leipzig

Discovery of statistical equivalence classes using computer algebra

by

Christiane Görgen, Anna Bigatti, Eva Riccomagno, and Jim Q. Smith

Preprint no.: 33 2017

Discovery of statistical equivalence classes using computer algebra

Christiane Görgen, Anna Bigatti, Eva Riccomagno and Jim Q. Smith

May 26, 2017

Abstract

Discrete statistical models supported on labelled event trees can be specified using so-called interpolating polynomials which are generalizations of generating functions. These admit a nested representation. A new algorithm exploits the primary decomposition of monomial ideals associated with an interpolating polynomial to quickly compute all nested representations of that polynomial. It hereby determines an important subclass of all trees representing the same statistical model. To illustrate this method we analyze the full polynomial equivalence class of a staged tree representing the best fitting model inferred from a real-world dataset.

Keywords Graphical Models; Staged Tree Models; Computer Algebra; Ideal Decomposition; Algebraic Statistics.

1 Introduction

Families of finite and discrete multivariate models have been extensively studied, including many different classes of graphical models [2, 19]. Because these families of probability distributions can often be expressed as polynomials – or collections of vectors of polynomials – this has spawned a deep study of their algebraic properties [10, 21, 23]. These can then be further exploited using the discipline of computational commutative algebra and computer algebra software such as `CoCoA` [1] which has proved to be a powerful though somewhat neglected tool of analysis.

In this paper, we demonstrate how certain computer algebra techniques – especially the primary decomposition of ideals – can be routinely applied to the study of various finite discrete models. Throughout we pay particular attention to an important class of graphical models based on probability trees and called *staged trees* or *chain event graph* models [27]. These contain the familiar class of discrete (and context-specific) Bayesian networks as a special case. In particular, [16] gave a mathematical way of determining the statistical equivalence classes of staged tree models but did not give algorithms to actually find these. Here we use computer algebra in a novel way to systematically find a staged tree representation of a given family – if it indeed exists – and to uncover statistically equivalent staged trees in an elegant, systematic and useful way. This is an extensions of the techniques developed by [2] and others to determine Markov-equivalence classes of Bayesian networks where, instead of algebra, graph theory was used as a main tool.

So our methodology supports a new analysis of a very general but fairly recent statistical model class in a novel algebraic way and serves as an illustration of how more generally computer algebra can be a useful tool not only to the study of conventional classes of graphical model but other families of statistical model as well.

2 Staged trees and interpolating polynomials

2.1 Labeled event trees and staged trees

In this work we will exclusively consider graphs which are trees, so those which are connected and without cycles. We first review the theory of staged trees which represent interesting and very general discrete models in statistics [27].

Definition 1 (Labeled event trees)**.** Let $T = (V, E)$ be a finite directed rooted tree with vertex set V and edge set $E \subseteq V \times V$. We denote the root vertex of T by v_0.

The tree T is called an *event tree* if every vertex $v \in V$ has either no, two or more than two emanating edges. For $v \in V$, let $E_v = \{(v, w) \mid w \in V\} \cap E$ denote the set of the edges emanating from v. The pair (v, E_v) is called a *floret*.

Let Θ be a non-empty set of symbol/labels and let a function $\theta : E \longrightarrow \Theta$ be such that for any floret (v, E_v) the labels in $\theta(E_v)$ are all distinct. We call $\theta(E_v)$ the *floret labels* of v and denote this set by θ_v. The pair $\mathcal{T} = (T, \theta)$ of graph and function is called a *labeled event tree*. When θ takes values in $(0, 1)$ and $\sum_{e \in E_v} \theta(e) = 1$, $\mathcal{T}$ is called a *probability tree*[1].

For $v \in V$, the *labeled subtree rooted in* v is $\mathcal{T}_v = (T', \theta')$, where T' is the largest subtree of T rooted in v, and θ' is the restriction of θ to the edges in T'.

For any leaf $v \in V$, so for any vertex with no emanating edges, we trivially have that $E_v = \emptyset$, and hence $\theta_v = \emptyset$.

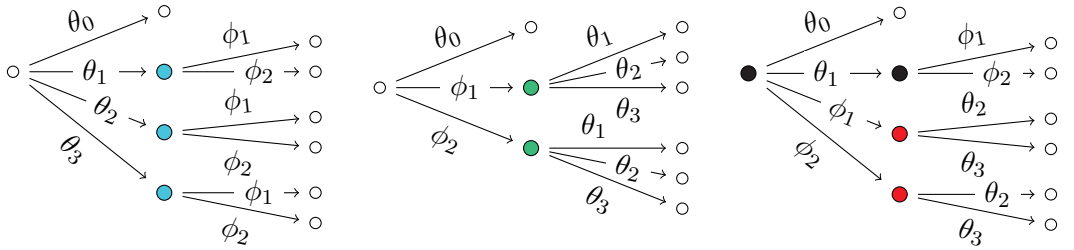
labeled event trees are well-known objects in probability theory and decision theory where they are used to depict discrete unfoldings of events. The labels on edges of a probability tree then correspond to transition probabilities from one vertex to the next and all edge probabilities belonging to the same floret sum to unity. See [25] for the use of probability trees in probability theory and causal inference, and see for instance [24] for how such a tree representation can be used in computational statistics.

In this paper, we generally do not require the labels on a labeled event tree to be probabilities.

Definition 2 (Staged trees)**.** A labeled event tree $\mathcal{T} = (T, \theta)$, with $T = (V, E)$, is called a *staged tree* if for every pair of vertices $v, w \in V$ their floret labels are either equal or disjoint, $\theta_v = \theta_w$ or $\theta_v \cap \theta_w = \emptyset$. A *stage* is a set of vertices with the same floret labels.

In illustrations of staged trees, all vertices in the same stage are usually assigned a common *color*: compare Fig. 1. Staged trees were first defined as an intermediate step to building *chain event graphs* as graphical representations for certain discrete statistical models [26]. Every chain event graph is uniquely associated to a staged tree and vice versa. In this way, the graphical redundancy of staged trees can be avoided, and elegant

[1] We should say more precisely: when the symbols $\theta(e)$ are evaluated in $(0, 1)$ for all $e \in E$.



(a) A labeled event tree which is staged.

(b) Another labeled event tree which is staged.

(c) A labeled event tree which is not staged.

Figure 1: Three illustrations of labeled event trees, analyzed in Examples 2, 8 and 9.

conjugate analyzes can be applied to staged tree models [3, 6, 12, 28]. In particular, every discrete and context-specific Bayesian network can alternatively be represented by a staged tree where stages indicate equalities of conditional probability vectors. We give examples of this later in the text.

For the development in this paper it is important to observe that staged trees with labels evaluated as probabilities are always also probability trees. This is however not the case for all labeled event trees because sum-to-1 conditions imposed on florets can be contradictory. See also Examples 2 and 9 below.

Example 1 (Saturated trees)**.** A *saturated* tree is a labeled event tree where all edges have distinct labels. So this is a staged tree where all floret labels are disjoint, or alternatively with every stage containing exactly one vertex. In the development below, saturated trees are graphical representations of saturated statistical models.

Example 2. Figure 1a shows a staged tree where all blue-coloured vertices are in the same stage. Figure 1b depicts a staged tree where the two green vertices are in the same stage. Figure 1c show a labeled event tree which is not staged because the floret labels of the two black vertices are neither equal nor disjoint.

2.2 Network polynomials and interpolating polynomials

We next define a polynomial associated to a labeled event tree which is the key tool used in this paper: see also [16].

Definition 3 (Network and interpolating polynomials)**.** Let $\mathcal{T} = (T, \theta)$ be a labeled event tree and let $\Lambda(\mathcal{T})$ denote the set of root-to-leaf paths in $\mathcal{T}$. For $\lambda \in \Lambda(\mathcal{T})$ let E_λ be the set of edges of λ. We call the products of the labels along a root-to-leaf path, $\pi_\theta(\lambda) = \prod_{e \in E_\lambda} \theta(e)$, *atomic monomials*.

Given a real-valued function $g : \Lambda(\mathcal{T}) \to \mathbb{R}$, we define the *network polynomial* of $\mathcal{T}$ and g, the linear combination of the atomic monomials with coefficients given by g, as:

$$c_{g,\mathcal{T}} = \sum_{\lambda \in \Lambda(\mathcal{T})} g(\lambda) \cdot \pi_\theta(\lambda) \tag{1}$$

with the particular case $c_{g,\mathcal{T}} = 1$ if $\mathcal{T}$ has no edges. The *interpolating polynomial* is the network polynomial with all $g(\lambda) = 1$ equal to one, and we write $c_{\mathcal{T}} = c_{1,\mathcal{T}}$.

Remark 1. A *network polynomial* $c_{g,\mathcal{T}}$ is a polynomial in the ring $\mathbb{R}[\Theta]$ of polynomials with real coefficients and whose indeterminates are the labels in Θ. An *interpolating polynomial* $c_{\mathcal{T}}$ is a polynomial with positive integer coefficients by construction. For these we write $c_{\mathcal{T}} \in \mathbb{Z}[\Theta]$.

Example 3. When $\mathcal{T} = (T, \theta)$ is a probability tree, every atomic monomial $\pi_\theta(\lambda)$ is the product of transition probabilities along a root-to-leaf path and thus the probability of an *atomic event* (or atom). Often the function g is an indicator function $g = \mathbb{1}_A$ of an event $A \subseteq \Lambda(\mathcal{T})$. In this case, (1) is a polynomial representation of the finite-additivity property of probabilities for A, so $c_{\mathbb{1}_A,\mathcal{T}} = \sum_{\lambda \in A} \pi_\theta(\lambda)$.

Interpolating polynomials have been used successfully to classify equivalence classes of staged trees which make the same distributional assumptions [16], as outlined in Section 3 below. They have further been used as a tool for calculating marginal and conditional probabilities in Bayesian networks and staged trees, using differentiation operations [8, 15].

In Theorem 1 and Proposition 1 for the purposes of this paper we now present two central results on interpolating polynomials. These results are given here in a reformulated, recursive form and very different from their original development [16, Proposition 1]. This refinement is necessary because the new proofs we give are constructive and, most importantly, transparently illustrate the mechanisms needed for our later algorithmic implementation.

Theorem 1. *Let $\mathcal{T} = (T, \theta)$ be an event tree and for $v \in V$ define*

$$\mathrm{poly}(\mathcal{T}_v) = \begin{cases} 1 & \text{if } E_v = \emptyset \\ \sum_{(v,w)\in E_v} \theta(v,w) \cdot \mathrm{poly}(\mathcal{T}_w) & \text{otherwise.} \end{cases}$$

Then the interpolating polynomial $c_{\mathcal{T}}$ of $\mathcal{T}$ is equal to $\mathrm{poly}(\mathcal{T}_{v_0})$ where v_0 is the root of $\mathcal{T}$.

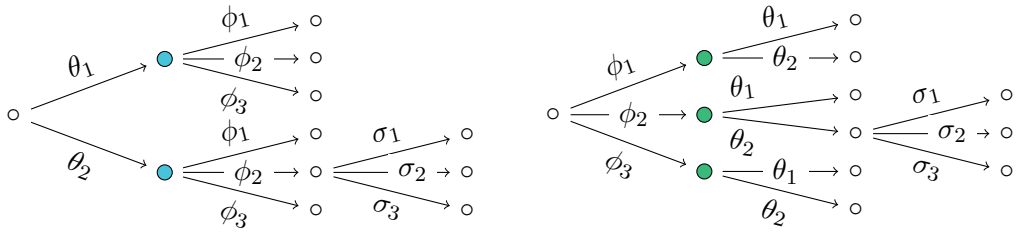
Proof. We prove the claim by induction on the *depth* of the tree, i.e. the number of edges in the longest root-to-leaf path. If $\mathcal{T}$ has depth $= 0$ then $E_{v_0} = \emptyset$ and $c_{\mathcal{T}} = 1 = \mathrm{poly}(\mathcal{T}_{v_0})$. If $\mathcal{T}$ has depth ≥ 1 then

$$\mathrm{poly}(\mathcal{T}_{v_0}) = \sum_{(v_0,w)\in E_{v_0}} \theta(v_0, w) \cdot \mathrm{poly}(\mathcal{T}_w).$$

Furthermore,

$$c_{\mathcal{T}} = \sum_{\lambda \in \Lambda(\mathcal{T})} \pi_\theta(\lambda) \;=\; \sum_{(v_0,w)\in E_{v_0}} \theta(v_0,w) \cdot \sum_{\lambda' \in \Lambda(\mathcal{T}_w)} \pi_\theta(\lambda') \;=\; \sum_{(v_0,w)\in E_{v_0}} \theta(v_0,w) \cdot c_{\mathcal{T}_w}$$

and $\mathrm{poly}(\mathcal{T}_w) = c_{\mathcal{T}_w}$ by the inductive hypothesis because the subtrees $\mathcal{T}_w$ all have lower depths than $\mathcal{T}$. □



(a) A labeled event tree (T, θ).

(b) A labeled event tree (S, θ').

Figure 2: Two staged trees with the same interpolating polynomial but different graphs. See Examples 4 and 5 and (6a) and (6b) in Example 6.

Example 4. The two staged trees $\mathcal{T} = (T, \theta)$ and $\mathcal{S} = (S, \theta')$ in Fig. 2 have the same interpolating polynomial, so the same sum of atomic monomials:

$$c_{\mathcal{T}} = c_{\mathcal{S}} = \theta_1\phi_1 + \theta_1\phi_2 + \theta_1\phi_3 + \theta_2\phi_1 + \theta_2\phi_2\sigma_1 + \theta_2\phi_2\sigma_2 + \theta_2\phi_2\sigma_3 + \theta_2\phi_3. \tag{2}$$

Here, the functions θ and θ' assign the same labels to different edges in the graphs T and S. Following the recursive construction in Theorem 1, we can then write this polynomial in terms of the interpolating polynomials of subtrees:

$$c_{\mathcal{T}} = \text{poly}(\mathcal{T}) = \theta_1 \cdot \text{poly}(\mathcal{T}_1) + \theta_2 \cdot \text{poly}(\mathcal{T}_2) \tag{3}$$

where $\text{poly}(\mathcal{T}_1) = \phi_1 + \phi_2$ and $\text{poly}(\mathcal{T}_2) = \phi_1 + \phi_2 \cdot (\sigma_1{+}\sigma_2{+}\sigma_3) + \phi_3$; or alternatively

$$c_{\mathcal{S}} = \text{poly}(\mathcal{S}) = \phi_1 \cdot \text{poly}(\mathcal{S}_1) + \phi_2 \cdot \text{poly}(\mathcal{S}_2) + \phi_3 \cdot \text{poly}(\mathcal{S}_3) \tag{4}$$

where $\text{poly}(\mathcal{S}_1) = \theta_1 + \theta_2$, $\text{poly}(\mathcal{S}_2) = \theta_1 + \theta_2 \cdot (\sigma_1{+}\sigma_2{+}\sigma_3)$ and $\text{poly}(\mathcal{S}_3) = \theta_1 + \theta_2$.

Example 4 shows that the distributive property of multiplication over addition is at the core of our work. The following corollary will be useful for studying staged trees with square-free atomic monomials: compare also Proposition 3 below.

Corollary 1. *Let $\mathcal{T} = (T, \theta)$ be a labeled event tree and let $c_{\mathcal{T}}$ be its interpolating polynomial. Then we can write*

$$c_{\mathcal{T}} = \sum_{(v_0, w) \in E_{v_0}} \theta(v_0, w) \cdot c_{\mathcal{T}_w}. \tag{5}$$

Moreover, if the root labels are not repeated, i.e. $\theta_{v_0} \cap \theta_v = \emptyset$ for all $v \in V \backslash \{v_0\}$, then no label in θ_{v_0} appears in any subtree-interpolating polynomial $c_{\mathcal{T}_w}$.

Proof. The proof is a trivial consequence of the construction of the polynomial $\text{poly}(\mathcal{T}_{v_0})$ in Theorem 1 above. □

Example 5. Consider again the two staged trees in Example 4. Their interpolating polynomial admits two different representations in terms of a linear combination as in Corollary 1, namely the ones in (3) and (4). We can see here explicitly how the polynomials above depend on the variables in subtrees of (T, θ) and (S, θ'). In particular, both sets $\{\theta_1, \theta_2\}$ and $\{\phi_1, \phi_2, \phi_3\}$ provide potential root-floret labels of a corresponding tree representation.

2.3 Polynomials with a nested representation

We know now that we can straightforwardly read an interpolating polynomial, and in particular a recursive representation of that polynomial, from a labeled event tree. In this section and in Section 5 we consider the inverse problem: given a polynomial in distributed form can we tell whether it is the interpolating polynomial of a labeled event tree? In order to answer this question first observe that the polynomials defined below admit a special structured representation and can be used as a surrogate for a labeled event tree as shown in Proposition 1.

Definition 4 (Nested representation)**.** Let $f \in \mathbb{Z}[\Theta]$ be a polynomial with positive integer coefficients. We say that f admits a *nested representation* if $f = 1$ or if it can be written as $f = \sum_{x \in A} x \cdot f_x$ where $A \subseteq \Theta$ is such that $\#A \geq 2$ and, for each $x \in A$, the polynomial f_x admits a nested representation.

Remark 2. The recursion in Definition 4 is finite because $\deg(f_x) = \deg(f) - 1$, for by construction polynomials with nested representations have positive coefficients.

The polynomial $\text{poly}(\mathcal{T}_v)$ in Theorem 1 is written in nested representation by construction. In this sense Proposition 1 below is the inverse result of Theorem 1, and a polynomial admits a nested representation if and only if it is the interpolating polynomial of a labeled event tree.

Proposition 1. *If $f \in \mathbb{Z}[\Theta]$ admits a nested representation then there exists a labeled event tree $\mathcal{T}$ such that $f = c_{\mathcal{T}}$.*

Proof. We prove the claim by induction on the degree of f. If $\deg(f) = 0$ then $f = 1$ and therefore $f = c_{\mathcal{T}}$ where $\mathcal{T}$ is formed by a single vertex with no edges and no labels.

If $\deg(f) > 0$ then $f = \sum_{x \in A} x \cdot f_x$ and therefore by Remark 2 and by induction $f_x = c_{\mathcal{T}_x}$ for some tree $\mathcal{T}_x$ labeled over Θ. For all $x \in A$ let v_x be the root of $\mathcal{T}_x$. Then a tree $\mathcal{T}$ with interpolating polynomial f can be constructed by taking a new vertex v_0 assigned as the root of $\mathcal{T}$ and defining the edges of the root floret E_{v_0} to be $\{(v_0, v_x) \mid x \in A\}$. Then $f = c_{\mathcal{T}}$. □

The result above implies in particular that if f is a polynomial with nested representation $f = \sum_{x \in A} x \cdot f_x$ then the root labels of a tree with interpolating polynomial f are given by A.

Example 6. The nested representations of the two event trees $\mathcal{T}$ and $\mathcal{S}$ in Fig. 2 are

$$c_{\mathcal{T}} = \theta_1(\phi_1 + \phi_2 + \phi_3) + \theta_2(\phi_1 + \phi_2(\sigma_1{+}\sigma_2{+}\sigma_3) + \phi_3), \tag{6a}$$

$$c_{\mathcal{S}} = \phi_1(\theta_1 + \theta_2) + \phi_2(\theta_1 + \theta_2(\sigma_1{+}\sigma_2{+}\sigma_3)) + \phi_3(\theta_1 + \theta_2) \tag{6b}$$

as in Examples 4 and 5. These nestings are in one-to-one correspondence with the depicted trees, just as stated in Proposition 1.

Example 7. Let $\Theta = \{\theta_1, \theta_2, \theta_3\}$ and consider the polynomial $f = \theta_1\theta_2 + \theta_2\theta_3 + 2\theta_1\theta_3 \in \mathbb{Z}[\Theta]$. Then f has nested representation $\theta_1 \cdot (\theta_2 + \theta_3) + \theta_3 \cdot (\theta_1 + \theta_2)$ corresponding to a labeled event tree which is not staged.

Example 8. Let $\Theta = \{\theta_0, \theta_1, \theta_2, \theta_3, \phi_1, \phi_2\}$ and consider the polynomial

$$f = \theta_0 + \theta_1\phi_1 + \theta_1\phi_2 + \theta_2\phi_1 + \theta_2\phi_2 + \theta_3\phi_1 + \theta_3\phi_2.$$

Then f admits three different nested representations:

$$\begin{aligned} f &= \theta_0{\cdot}(1) + \theta_1{\cdot}(\phi_1 + \phi_2) + \theta_2{\cdot}(\phi_1 + \phi_2) + \theta_3{\cdot}(\phi_1 + \phi_2), && \text{(7a)} \\ &= \theta_0{\cdot}(1) + \phi_1{\cdot}(\theta_1 + \theta_2 + \theta_3) + \phi_2{\cdot}(\theta_1 + \theta_2 + \theta_3), && \text{(7b)} \\ &= \theta_0{\cdot}(1) + \theta_1{\cdot}(\phi_1 + \phi_2) + \phi_1{\cdot}(\theta_2 + \theta_3) + \phi_2{\cdot}(\theta_2 + \theta_3). && \text{(7c)} \end{aligned}$$

In particular, (7a) corresponds to the staged tree in Fig. 1a and (7b) to the staged tree in Fig. 1b. In Section 4 we show that there are no other staged trees with interpolating polynomial f. The third nested representation (7c) corresponds to the labeled event tree in Fig. 1c which is not staged.

In the above examples, a given polynomial can admit several different nested representations. By the result below, this is not always the case.

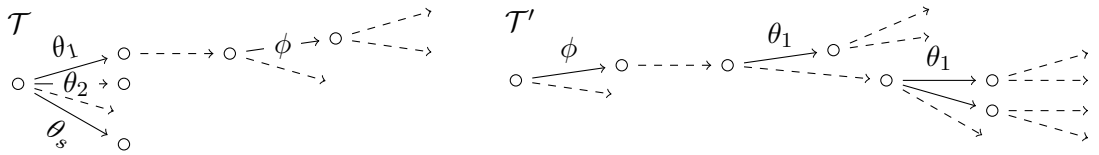
Proposition 2 (Saturated trees)**.** *For a saturated tree $\mathcal{T}$, the interpolating polynomial $c_\mathcal{T}$ has a unique nested representation.*

Proof. Let $\mathcal{T}'$ be a labeled event tree, not necessarily saturated nor staged, with interpolating polynomial $c_{\mathcal{T}'} = c_\mathcal{T}$. We prove that $\mathcal{T}' = \mathcal{T}$, i.e. $\mathcal{T}'$ is indeed the saturated tree $\mathcal{T}$.

Let $C = \text{support}(c_\mathcal{T})$ be the set of power-products (or *monomials*) in $c_\mathcal{T}$, and for a label x indicate the set of all multiples of that label with $C_x = \{t \in c \mid t \text{ multiple of } x\}$.

Let $F = \{\theta_1, \ldots, \theta_s\}$ and F', respectively, be the set of root-floret labels of $\mathcal{T}$ and $\mathcal{T}'$, so θ_{v_0} in Definition 1 w.r.t. $\mathcal{T}$ and $\mathcal{T}'$. We first prove that $F = F'$. For any $\theta_i \in F$ the power-products in c_{θ_i}, corresponding to the root-to-leaf paths originating from the root-edge in $\mathcal{T}$ which is labeled θ_i, are not multiples of any θ_j for $i \neq j$ because $\mathcal{T}$ is saturated. Thus, if $F' \subsetneq F$ and $\theta_i \notin F'$ then the power-products in c_{θ_i} could not correspond to root-to-leaf paths in $\mathcal{T}'$.

It follows that if $F \neq F'$ then there must be a label $\phi \in F'$ with $\phi \notin F$. Since $\mathcal{T}$ is saturated, ϕ is the label of only one edge in $\mathcal{T}$, and this edge is, say, in the subtree starting from the root edge labeled θ_1. In terms of the power-products, this implies that $C_\phi \subseteq C_{\theta_1}$. Hence, in $\mathcal{T}'$ all root-to-leaf paths originating from the root edge labeled by ϕ must have an edge labeled θ_1: see the figure below.



Now consider the root-to-leaf path in $\mathcal{T}'$ where θ_1 appears at greatest depth, i.e. with the longest path from the root vertex. The floret containing θ_1 must have at least another edge so the paths through this other edge have θ_1 at greater depth. But this is a contradiction. Hence $F = F'$.

The subtrees of $\mathcal{T}$ rooted in the s children of its root are again saturated trees, and their interpolating polynomials are $\sum_{t \in c_\tau} t$ for $\tau \in \{\theta_1, \ldots, \theta_s\}$ and have disjoint sets of labels because $\mathcal{T}$ is saturated. Therefore we can repeat the reasoning above on these subtrees and their interpolating polynomials. We conclude in a finite number of steps that $\mathcal{T} = \mathcal{T}'$. □

Thus when reading an interpolating polynomial from a tree, instead of summing atomic monomials as in Definition 3 we can directly use the tree graph to infer a bracketed, nested representation of that polynomial. This representation is in one-to-one correspondence with the labeled graph itself, so the original representation can be easily recovered. Similarly, once we are given *any* polynomial in distributed form and this polynomial admits such a nested bracketing then we can always find a corresponding tree representation. These insights open the door to replace graphical representations of statistical models by polynomial representations, and hence enable us to employ computer algebra in their study. We will show how this can be done in the next section.

3 Polynomial and statistical equivalence

Computer algebra is often used to study polynomials that arise naturally in statistical inference. For instance, context-specific Bayesian networks, staged trees and chain event graphs are all *parametric* statistical models whose probability mass function is of monomial form: $p_{\boldsymbol{\theta}}(x) = \boldsymbol{\theta}^{\alpha_x} = \theta_1^{\alpha_{x,1}} \cdots \theta_d^{\alpha_{x,d}}$ for every atom x in an underlying sample space where $\alpha_x = (\alpha_{x,1}, \ldots, \alpha_{x,d}) \in \mathbb{Z}^d_{\geq 0}$. This monomial $\boldsymbol{\theta}^{\alpha_x}$ can then be thought of as for instance a product of *potentials* [19] or simply a product of edge probabilities in a staged tree with root-to-leaf paths as atoms. So the network and interpolating polynomials as in Definition 3 can be defined for all parametric models admitting a general *monomial parametrization* as given above [20]. We can then apply the theory above to these models and employ computer algebra techniques in their study. In particular, often very different parametrizations can give rise to the same model and the interpolating polynomial can help to determine these.

Definition 5 (Polynomial and statistical equivalence)**.** Two staged trees $\mathcal{T} = (T, \theta)$ and $\mathcal{S} = (S, \theta')$ with the same label set Θ are called *polynomially equivalent* if their interpolating polynomials are equal.

Two staged trees $\mathcal{T} = (T, \theta)$ and $\mathcal{S} = (S, \theta')$ with possibly different label sets, say Θ and Ξ, are called *statistically equivalent* if there is a bijection $\Psi : \Lambda(\mathcal{T}) \to \Lambda(\mathcal{S})$ which identifies their root-to-leaf paths and for any evaluation function on Θ, namely $\mathrm{Val}_\Theta : \Theta \to (0, 1)$ extended to $\lambda \in \Lambda(\mathcal{T})$ as $\mathrm{Val}_\Theta(\lambda) = \prod_{e \in \lambda} \mathrm{Val}_\Theta(\theta(e))$, there exists an evaluation on Ξ, $\mathrm{Val}_\Xi : \Xi \to (0, 1)$, such that $\mathrm{Val}_\Theta(\lambda) = \mathrm{Val}_\Xi(\Psi(\lambda))$ for all $\lambda \in \Lambda(\mathcal{T})$.

By definition, two staged trees whose labels are evaluated as probabilities are statistically equivalent if and only if they represent the same statistical model.

Since the interpolating polynomials of polynomially equivalent trees are equal, they are the sum of the same atomic monomials. Therefore there is a bijection between the root-to-leaf paths of polynomially equivalent trees. This implies that polynomially equivalent trees are also statistically equivalent. For instance, the trees from Examples 5 and 6 are polynomially, and so statistically equivalent. In particular, the interpolating

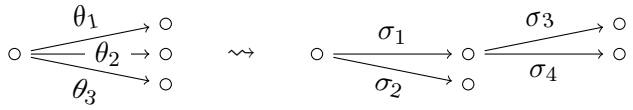


Figure 3: Maximal and minimal representations of a floret. See Examples 10 and 11.

polynomial is sufficient to determine a probability distribution up to a permutation of the values it takes across an underlying sample space.

From Proposition 1, the class of polynomially equivalent trees is fully described by *all* nested representations of the interpolating polynomial. Indeed, when reordering the terms of a nested representation as in Fig. 2, the atomic monomials of the underlying tree do not change. So if we are given the interpolating polynomial of a staged tree and we can find all its possible nested representations then we have automatically found all of its polynomially equivalent tree representations – and often a large subclass of the whole statistical equivalence class. For example, in the case of decomposable Bayesian networks the equivalence class of a polynomial given in clique parametrization contains the Markov-equivalence class [14].

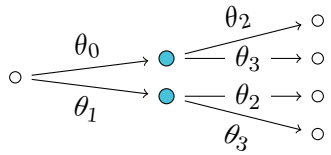
Polynomially equivalent trees can be thought of as those having the same parametrization. However this parametrization is often read in a different *non-commutative* way for different graphical representation in that class. For instance, the staged trees in Examples 5 and 6 have the same atomic monomials belonging to identified atoms but $\pi_\theta(\lambda) = \theta_1\phi_1$ in (T, θ) and $\pi_{\theta'}(\lambda') = \phi_1\theta_1$ in (S, θ') for identified atoms λ and λ'. Analogous instances of this phenomenon occur in the class of decomposable Bayesian networks where a model parametrization can be given by potentials on cliques which are renormalized across different graphical representations of the same model.

Statistically equivalent trees however can be thought of as reparametrizations of each other, very much like in Bayesian networks where a parametrization can either be based on parent relations between single nodes in a graph or alternatively on clique margins. See also Example 12.

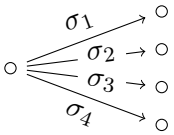
Example 9. Polynomially equivalent trees can often be described by a variety of different graphs. For instance, the polynomial $c = \theta_0 + (\theta_1 + \theta_2 + \theta_3)(\phi_1 + \phi_2)$ has at least three different labeled trees associated: see Fig. 1 and Example 8.

The two trees in Figs. 1a and 1b are polynomially equivalent representations of the same model on seven atoms. The tree in Fig. 1c is not because it is not a staged tree. In particular, this tree is not a probability tree because sum-to-1 conditions imposed on its florets would be contradictory.

Example 10 (Maximal representations)**.** For any labeled event tree there exists a statistical equivalent *binary* labeled event tree whose graph $T = (V, E)$ is such that $\#E_v \in \{0, 2\}$ for all $v \in V$. This can be thought of as a *maximal* representation within the class of statistically equivalent trees. We can easily obtain a binary tree by splitting up each floret with strictly more than two edges as shown in Fig. 3. In particular, for a floret in a probability tree labeled by $\theta_1, \theta_2, \theta_3$, we would obtain new labels $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ which are renormalizations of the original parameters such that sum-to-1 conditions hold, $\sigma_1 + \sigma_2 = 1$ and $\sigma_3 + \sigma_3 = 1$, while retaining the distribution over the three depicted atoms, so $\sigma_1 = \theta_1 + \theta_2$, $\sigma_1 = {}^{\theta_1}/_{\theta_1+\theta_2}$, $\sigma_2 = {}^{\theta_2}/_{\theta_1+\theta_2}$ and $\sigma_2 = 1 - \sigma_1$.



(a) A staged tree representing a binary independence model.



(b) Minimal representation of the saturated model on four atoms.

Figure 4: Trees analyzed in Example 12.

Example 11 (Minimal representations)**.** In the polynomial equivalence class of a saturated tree there is exactly one member, namely the tree itself. This is because, by Proposition 2, for saturated trees the nested representation of an interpolating polynomial is unique. The statistical equivalence class of a saturated tree however is much bigger. This is a consequence of Example 10 above. In particular, for every saturated tree there is a unique *minimal* graphical representation given by a single floret whose labels are the atomic monomials (or joint probabilities) and whose number of edges coincides with the number of root-to-leaf paths in any equivalent representation.

In the development in this paper we mainly focus on a parametric characterization of staged tree and other statistical models. This naturally links in with an alternative *implicit* characterization which is well known in algebraic statistics. For instance, a polynomial representation of a Bayesian network involving exclusively the joint probabilities – i.e. the values of the associated probability mass function $p(x)$ as x varies in the sample space – can be derived from the equalities $p(x) = \boldsymbol{\theta}^{\alpha_x}$ using ring operations. The algebraic theory behind this is called *elimination theory* [18] of which Gaussian elimination for solving systems of linear equations is a simple example. The representation of a Bayesian network as such a set of polynomials is an algebraic structure called a *toric ideal* and has great importance in algebraic statistics: see e.g. [10, 13, 23].

Notably, this alternative characterization can also be used to describe statistical equivalence – though in a less constructive way than the method we present here and without immediate links to a graphical representation of a model.

Example 12. The labeled event tree in Fig. 4a is a staged tree on four atoms with labels $\Theta = \{\theta_0, \theta_1, \theta_2, \theta_3\}$. The equalities holding for the four atomic monomials

$$p_1 = \theta_0\theta_2,\ p_2 = \theta_0\theta_3,\ p_3 = \theta_1\theta_2, p_4 = \theta_1\theta_3$$

imply the equality $p_1p_4 = p_2p_3$. This parametrization of the model in Fig. 4a is not to be confused with the minimal representation of the saturated model on four atoms in Fig. 4b.

An interpretation of this equation is as follows. Assume two binary random variables $X, Y \in \{0, 1\}$ are such that

$$\begin{aligned} \Pr(Y = 1,\, X = 1) &= p_1, \qquad \Pr(Y = 0,\, X = 1) = p_2, \\ \Pr(Y = 1,\, X = 0) &= p_3, \qquad \Pr(Y = 0,\, X = 0) = p_4. \end{aligned}$$

Then $p_1p_4 = p_2p_3$ is an instance of a fundamental relationship in algebraic statistics for representing conditional independence of discrete random variables: see e.g. [23, Section

6.10] and [10, Proposition 3.1.4]. In this specific case the equality implies that X and Y are independent.

4 From polynomials to trees: finding the nested representations

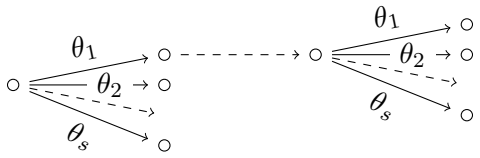
4.1 Potential root-floret labels and square-free monomials

Building on the results above we can now use methods from commutative algebra to compute all the staged trees with a given interpolating polynomial and so to compute a complete polynomial equivalence class. The two key notions we use to build an algorithm which determines these classes are those of a monomial ideal and of its primary decomposition which, for square-free monomials, coincides with the prime decomposition. These notions are recalled in the appendix.

The key of the proposed algorithm is Theorem 2 below. This states in algebraic terms that for any tree $\mathcal{T} = (T, \theta)$ each monomial in $c_{\mathcal{T}}$ is divisible by some label in the set $F = \theta_{v_0}$ of the floret labels belonging to the root of $\mathcal{T}$, and that F is minimal (with respect to inclusion) with this property.

Theorem 2. *Let $\mathcal{T}$ be a staged tree. The monomial ideal $\langle \theta_{v_0} \rangle$ generated by the root-floret labels is a minimal prime of the ideal $\langle \text{support}(c_{\mathcal{T}}) \rangle$ generated by the support of $c_{\mathcal{T}}$.*

Proof. Let $F = \theta_{v_0} = \{\theta_1, ..., \theta_s\}$ be the set of root-floret labels. Then each power-product in $c_{\mathcal{T}}$ is a multiple of some label in F. Because it is generated by indeterminates, $\langle F \rangle$ is a prime ideal containing all power-products in $c_{\mathcal{T}}$. Suppose, by contradiction, that F is not minimal. Then there exists $\tilde{F} \subsetneq F$ with $\langle \tilde{F} \rangle$ containing all power-products in $c_{\mathcal{T}}$. Without loss of generality let $\tilde{F} = \{\theta_2, ..., \theta_s\}$. Now, each root-to-leaf path starting with the root edge labeled θ_1 has an associated atomic monomial $\theta_1 t \in \text{support}(c_{\mathcal{T}}) \subseteq \tilde{F}$, $j \geq 2$. Therefore $\theta_1 t = \theta_1 \theta_j t'$ for some $\theta_j \in \tilde{F}$. As $\mathcal{T}$ is staged, this implies that the whole root floret F must appear again in the subtree: see the illustration below.



Next consider the subtree containing the repeated root-floret labels at a minimum depth and repeat the reasoning above: each root-to-leaf path containing the two edges labeled θ_1 corresponds to an atom $\theta_1^2 t \in \text{support}(c_{\mathcal{T}})$ and is therefore a multiple of some label in $\tilde{F}$. Then the whole root floret is repeated again deeper in the subtree, producing some atom divisible by θ_1^3. Since this reasoning can be repeated a finite number of times, we have the contradiction that there is an atomic monomial divisible by a power of θ_1 and by no label in $\tilde{F}$. Therefore $F = \{\theta_1, ..., \theta_s\}$ is minimal. □

Example 13. The interpolating polynomial $c_{\mathcal{T}}$ in Example 4 has support

$$\text{support}(c_{\mathcal{T}}) = \{\theta_1\phi_1,\ \theta_1\phi_2,\ \theta_1\phi_3,\ \theta_2\phi_1,\ \theta_2\phi_2\sigma_1,\ \theta_2\phi_2\sigma_2,\ \theta_2\phi_2\sigma_3,\ \theta_2\phi_3\}.$$

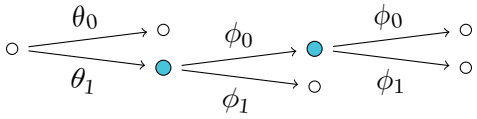


Figure 5: A tree whose interpolating polynomial has a non-square-free atomic monomial. See Example 14.

The primary decomposition of the corresponding square-free monomial ideal is

$$\langle \text{support}(c_{\mathcal{T}}) \rangle = \langle \phi_1, \phi_2, \phi_3 \rangle \cap \langle \theta_1, \theta_2 \rangle \cap \langle \phi_1, \phi_3, \theta_1, \sigma_1, \sigma_2, \sigma_3 \rangle.$$

Therefore, by Theorem 2, there are three different sets of possible root labels for a staged tree with interpolating polynomial $c_{\mathcal{T}}$. We show in Example 15 below that the polynomial equivalence class of $c_{\mathcal{T}}$ is given by just two trees.

Example 14. Consider the interpolating polynomial $c_{\mathcal{T}} = \theta_0 + \theta_1\phi_1 + \theta_1\phi_0\phi_1 + \theta_1\phi_0^2$. The minimal prime decomposition of $\langle \text{support}(c_{\mathcal{T}}) \rangle$ is given by two sets, namely $\langle \theta_0, \theta_1 \rangle$ and $\langle \phi_0, \theta_0, \phi_1 \rangle$. The first one leads to the tree in Fig. 5. It can be shown by exhaustive search that the second does not give the labels of a root floret in a labeled event tree.

The key assumption in Theorem 2 is that the input tree $\mathcal{T}$ is staged, otherwise the result need not be true.

This theorem is central to the algorithm we present in the following section because it shows that instead of searching for root-floret labels among all subsets of labels Θ, the search can be limited to those subsets which are the generators of the minimal primes of $\langle \text{support}(c_{\mathcal{T}}) \rangle$. If Θ has d elements, their number is bounded above by $\binom{d}{\lceil d/2 \rceil}$ whereas the number of the subsets of Θ is 2^d. So considering all possible subsets of Θ, and having to repeat this recursively, may lead to a combinatorial explosion of cases to analyze. As a consequence, Theorem 2 gives a drastic reduction of the set of candidate root-floret labels.

Staged trees whose interpolating polynomials are sums of square-free power-products are interesting cases both from an algebraic viewpoint and for their interpretation in statistical inference. For instance, if all power-products in $c_{\mathcal{T}}$ are square-free then the proof of Theorem 2 can be shortened obtaining the contradiction by Proposition 3 directly. In terms of staged tree models, this condition implies that if a unit passes through a vertex in a given stage it cannot subsequently pass through another vertex in the same stage. By making this requirement we can avoid various complex ambiguities associated with exactly how we relate a sample distribution to a polynomial family. Although less useful in modeling time series, in most cross-sectional statistical models this constraint will almost always apply.

The restriction to polynomials with square-free support enables us to prove the second and third central result for our algorithmic implementation.

Proposition 3 (Root-floret labels)**.** *Let $\mathcal{T}$ be a staged tree whose interpolating polynomial $c_{\mathcal{T}} = \sum_{(v_0,w)\in E_{v_0}} \theta(v_0, w) \cdot c_{\mathcal{T}_w}$ is a sum of square-free power-products. Then no label in θ_{v_0} appears in any subtree-polynomial $c_{\mathcal{T}_w}$.*

Proof. Because $\mathcal{T}$ is a staged tree we have $\theta_{v_0} \cap \theta_v = \emptyset$ or $\theta_{v_0} = \theta_v$ for all $v \in V \backslash \{v_0\}$ by Definition 3. By contradiction, suppose there is a subtree $\mathcal{T}_w$ containing a floret with labels θ_{v_0}. Let θ_1 be the label of the edge (v_0, w) for some $w \in V$. Then there is a root-to-leaf path with at least two edges labeled θ_1: see also the illustration in the proof of Theorem 2. Hence there is a multiple of θ_1^2 in $c_\mathcal{T}$. This is a contradiction because $c_\mathcal{T}$ is a sum of square-free power-products. So there is no subtree $\mathcal{T}_w$ containing a floret with labels θ_{v_0}. The claim follows from Corollary 1. □

Corollary 2. *Let $\mathcal{T}$ be a staged tree whose interpolating polynomial $c_\mathcal{T}$ is a sum of square-free power-products. Then all coefficients in $c_\mathcal{T}$ are equal to 1.*

Proof. The claim follows from Proposition 3 and its recursive application to subtrees of $\mathcal{T}$. □

So when searching for staged trees using square-free interpolating polynomials, coefficients might be ignored. This is not true for labeled event trees by Example 7. In Section 4.3 we will see that this result will allow the application of the algorithm in Section 4.2 to network polynomials of staged trees.

4.2 The algorithm `StagedTrees`

Given a polynomial f whose power-products are square-free and with coefficients all equal to one, there is an obvious algorithm which determines all its nested representations, and in particular all staged trees for which f is the interpolating polynomial. This algorithm is here called `StagedTrees` and is given in pseudo-code in Alg. 1. Following the notation in Definition 4, the proposed algorithm searches over subsets $A \subseteq \Theta$ of the indeterminates appearing in f and recursively checks whether it is possible to construct the polynomials f_x for $x \in A$. The choices of A are hereby constrained to the minimal primes of the monomial ideal associated to f as determined by Theorem 2. This algorithm works even when it is not known *a priori* whether or not f is the interpolating polynomial of a staged tree. Since the support of f is finite it is clear that the recursion terminates. The function `StagedTrees` is part of the `CoCoA` distribution from version 5.1.6 (`http://cocoa.dima.unige.it/download/CoCoAManual/html/cmdStagedTrees.html`).

The base steps of the recursion in Alg. 1 are given by the simplest trees: a single vertex tree for $C = 1$ (Step 2), or a floret without subtrees for $C \subseteq \Theta$ (Step 4) with at least two edges (Step 3). Compare also the recursive description in Theorem 1. In Step 5, Theorem 2 is applied to determine the candidate root-florets $F_1, \ldots, F_k$. The main loop in Step 6 considers each F_i one at a time, and determines all the staged trees having root floret F_i, $i = 1, \ldots, k$.

In the main loop, Step 6.2 checks if the subsets defined in Step 6.1 give a partition for C which is a necessary condition from Proposition 3: since F_i is a minimal prime for $\langle C \rangle$ it follows that $C = \cup_{x \in F_i} C_x$. Therefore only disjointness needs to be verified. Then the inner loop in Step 6.3, with its sub-steps, considers one at a time each $x \in F_i$, and determines (if possible) all the subtrees emanating from the second vertex of the edge labeled x. In particular, Step 6.3.1 stops the search for F_i if there is a single emanating edge and therefore by definition not an event tree. Step 6.3.3 makes the recursive call

Algorithm 1: `StagedTrees`: Inferring all nested representations of a given polynomial.

Input : $C = \text{support}(f)$ a set of square-free power-products over a set of indeterminates Θ for a polynomial $f = \sum_{t \in C} t \in \mathbb{Z}[\Theta]$ with all coefficients zero or one.

Output: The set W of all staged-trees with interpolating polynomial f.

1 Let $W = \emptyset$ (initialise the output set of trees)

2 **if** $C = \{1\}$ **then**
  return a single-vertex tree

3 **if** $\#C = 1$ *has only one element* **then**
  return the emptyset $\emptyset$

4 **if** $C \subseteq \Theta$ *is a subset of indeterminates and* $\#C \geq 2$ *has at least two elements* **then**
  return the staged tree made of the single floret labeled by Θ

else

5 compute the prime decomposition $\{F_1, \ldots, F_k\}$ of the square-free monomial ideal $\langle C \rangle$

6 **for** *each* $i = 1, \ldots, k$ **do**

consider F_i and proceed as follows:

6.1 **for** *each indeterminate* $x \in F_i$ **do**
  define $C_x = \{t \in C \mid t \text{ is a multiple of } x\}$

6.2 **if** *there exist* $y \neq x$ *such that* $C_x \cap C_y \neq \emptyset$ **then**
  discard F_i and go to next minimal prime in Step 6

6.3 **for** *each indeterminate* $x \in F_i$ **do**

6.3.1 **if** $\#C_x = 1$ *has only one element and this is is not equal to* x*, so* $C_x \neq \{x\}$ **then**
  discard F_i and go to next minimal prime in Step 6

6.3.2 define a set $C'_x = \{\frac{t}{x} \mid t \in C_x\}$ of square-free power-products over $\Theta \backslash \{F_i\}$;

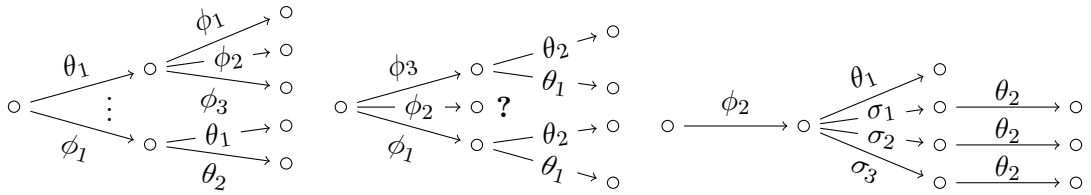
6.3.3 call `StagedTrees` recursively with input C'_x and obtain the set W_x of all staged trees with interpolating polynomial $\sum_{t \in C'_x} t$;

6.3.4 **if** $W_x = \emptyset$ *is the emptyset* **then**
  discard F_i and go to next minimal prime in Step 6

6.4 construct the set W' of all trees with root-floret labels $F_i = \{x_1, \ldots, x_{r_i}\}$ and
construct the subtrees $(T_{x_1}, \ldots, T_{x_{r_i}}) \in W_{x_1} \times \cdots \times W_{x_{r_i}}$ where each T_{x_j} is rooted at the second vertex of the edge labeled x_j;

6.5 discard from W' all trees which are not staged;

6.6 redefine W as $W \cup W'$;



(a) F_1 leads to a non-staged tree.

(b) F_2: subtrees from θ_0 and ϕ_2.

(c) F_2: subtree from ϕ_1 is not an event tree.

Figure 6: The working of the `StagedTrees` algorithm. See Example 15.

on C'_x (defined in Step 6.3.2) to determine the set W_x of all possible subtrees from x. If W_x is empty then Step 6.3.4 stops the search for F_i.

Concluding the main loop, Step 6.4 is reached if for each edge having a label in F_i there is at least one subtree. Then the floret labeled by F_i together with all combinations of its subtrees make a set W' of event trees, with root-floret labels F_i, whose interpolating polynomial is the sum of the monomials in C. At this point Step 6.5 discards those which are not staged. In particular the subtrees are staged, and compatibility of stages across the subtrees is checked here, in the obvious way. Finally, Step 6.6 stores them in W.

Example 15. We illustrate the working of the `StagedTrees` algorithm on Example 4. From Example 13 we can consider only three sets of potential root-floret labels of staged trees with interpolating polynomial $c_{\mathcal{T}}$ given in (2). These are:

$$\begin{aligned} F_1 &= \{\phi_1, \phi_3, \theta_1, \sigma_1, \sigma_2, \sigma_3\} \\ F_2 &= \{\phi_1, \phi_2, \phi_3\} \\ F_3 &= \{\theta_1, \theta_2\}. \end{aligned}$$

The first set F_1 cannot be a floret-label set because $C_{\phi_1} \cap C_{\theta_1} \neq \emptyset$, see Step 6.2 in the algorithm. Indeed the two sets

$$\begin{aligned} C_{\phi_1} &= \{\theta_1\phi_1, \theta_2\phi_1\} &&= \phi_1\{\theta_1, \theta_2\} \\ C_{\theta_1} &= \{\theta_1\phi_1, \theta_1\phi_2, \theta_1\phi_3\} &&= \theta_1\{\phi_1, \phi_2, \phi_3\} \end{aligned}$$

show that, if F_1 were a floret-label set, then the tree would include a structure such as in Fig. 6a which cannot be part of a staged tree: see also Corollary 1 and Proposition 3. Above we have used the convention that the product of a single label with a set of labels is defined as the set of all elementwise products.

With F_2 in the first step of the algorithm we have

$$\begin{aligned} C_{\phi_3} &= \{\phi_3\theta_1, \phi_3\theta_2\} &&= \phi_3\{\theta_1, \theta_2\} \\ C_{\phi_2} &= \{\theta_1\phi_2, \theta_2\phi_2\sigma_1, \theta_2\phi_2\sigma_2, \theta_2\phi_2\sigma_3\} &&= \phi_2\{\theta_1, \theta_2\sigma_1, \theta_2\sigma_2, \theta_2\sigma_3\} \\ C_{\phi_1} &= \{\theta_1\phi_1, \theta_2\phi_1\} &&= \phi_1\{\theta_1, \theta_2\} \end{aligned}$$

The algorithm calls recursively on the sets C'_{ϕ_3} and C'_{ϕ_1} but stops immediately (Step 4 in the algorithm) as summarized in Fig. 6b. For the middle branch we need to continue the recursion by working on C'_{ϕ_2}. The monomial ideal generated by C'_{ϕ_2} has the following primary decomposition

$$\langle C'_{\phi_2} \rangle = \langle \theta_1, \theta_2 \rangle \cap \langle \theta_1, \sigma_1, \sigma_2, \sigma_3 \rangle.$$

Taking $F = \{\theta_1, \theta_2\}$ gives the tree in Fig. 2b while $F = \{\theta_1, \sigma_1, \sigma_2, \sigma_3\}$ leads to the situation in Fig. 6c which does not correspond to an event tree. In conclusion, F_2 gives the tree in Fig. 2b only. The result of the algorithm starting from F_3 is analogous and leads to the tree in Fig. 2a.

4.3 Discussion of the algorithm

It was shown in [16] that the application of two graphical operators called the "swap" and "resize" on a staged tree could be used to traverse a statistical equivalence class. However these authors did not provide an implementation of their graphical methods in algebraic or computational terms. So Alg. 1 fills that gap and enables us to determine the full polynomial equivalence class of a given staged tree. We hereby focus on staged as opposed to labeled event trees because these can always be interpreted as representations of statistical models as in Sections 2 and 3. Of course our new algorithm can be easily adapted to discover more general representations. We will now discuss some of the properties of this algorithm.

First, the `StagedTrees` algorithm can be modified to work on non-square-free power-products. For this purpose Step 6.2 must be disabled and all the possible partitions of C need to be checked, making the algorithm more expensive. For example, the only minimal prime for the ideal $\langle \text{support}(\theta_1 + \theta_2 \cdot (\theta_1 + \theta_2)) \rangle$ is $\langle \theta_1, \theta_2 \rangle$ which leads to two partitions $\{\theta_1, \theta_1\theta_2\}, \{\theta_2^2\}$, and $\{\theta_1\}, \{\theta_1\theta_2, \theta_2^2\}$. Calling the algorithm on the first partition gives no answer because it leads to a tree which is not an event tree, whereas the second gives the original nested representation. Moreover, in this partitioning one also needs to keep track of the coefficients: as illustrated by the nested representation $\theta_1 \cdot (\theta_1 + \theta_2) + \theta_2 \cdot (\theta_1 + \theta_2) = \theta_1^2 + \theta_1\theta_2 + \theta_2^2$.

Second, so far we often emphasized the use of the *interpolating polynomial* as opposed to the *network polynomial* in Definition 3. This was to highlight the *structure* of the tree, as opposed to the real values associated its root-to-leaf paths: compare also Definition 5. However, if $c_{g,\mathcal{T}}$ is the network polynomial associated to a staged tree $\mathcal{T}$ and its *power-products are square-free*, from Proposition 3 it follows that the root-to-leaf paths $\lambda \in \Lambda(\mathcal{T})$ are labeled by distinct monomials. This means that in the network polynomial the coefficients $g(\lambda)$ are kept distinct. In conclusion, all staged trees with a given network polynomial $c_{g,\mathcal{T}}$ are found by the algorithm `StagedTrees` applied to $C = \text{support}(c_{g,\mathcal{T}})$. Afterwards the coefficients $g(\lambda)$ can be associated to the corresponding root-to-leaf paths.

Third, thanks to the reduction to minimal primes, the algorithm is very fast also for real-world settings. In Section 7 we will apply `StagedTrees` to discover the polynomial equivalence class of a staged tree describing a real problem with 24 atomic events. This computation takes much less than a second on a laptop with a 2.4 GHz Intel Core 2 Duo processor. Similarly, it takes 2.3 seconds to compute the 576 staged trees sharing the interpolating polynomial $(\theta_0 + \theta_1)(\phi_1 + \phi_2)(\tau_0 + \tau_1)(\sigma_0 + \sigma_1)$ representing four independent binary random variables: compare Fig. 4a. Computing the polynomial equivalence class of four independent random variables taking three levels each takes significantly longer at 12:23min but produces 55,296 different staged trees, each having 81 atoms. Naturally, the more stage structure there is present the more different polynomially equivalent representations are possible, so the latter two are somewhat extreme cases. On medium-sized real-world applications like the one presented below our computations

are very fast. So this algorithm allows us to systematically enumerate and analyze staged trees of the same order or even bigger than the study we will consider.

Fourth, every Bayesian network, context-specific Bayesian network [4] and object-oriented Bayesian network [17] can be represented by a staged tree where inner vertices correspond to conditional random variables and the emanating edges correspond to the different states of these variables. Then two vertices are in the same stage if and only if the corresponding rows of conditional probability tables are identified. For instance, the independence model of two binary random variables can be represented by the staged tree depicted in Fig. 4a. The complete Bayesian network on two binary random variables can be represented by the staged tree in Fig. 4b. However, staged trees allow for much less symmetric – and hence more general – modeling assumptions. In particular, they do not rely on an underlying product-space structure but can express relationships directly in terms of events. So this class of models is much larger than the Bayesian network class and as a consequence the `StagedTrees` algorithm can be optimized to traverse this wider class as well as the class of Bayesian networks.

So the methodology we developed for the `StagedTrees` algorithm will serve as a springboard for really fast algorithms to analyze equivalence classes of staged trees and in the future causal discovery algorithms over this class: see also Section 7. We illustrate below that these computer algebra analysis enable us to obtain further insights about the properties of the underlying class of statistical models.

5 Additional properties of interpolating polynomials

A natural question to ask is whether or not a given polynomial can be seen to be the interpolating polynomial of an event tree *without* having to construct a nested representation first. The following proposition gives some necessary conditions for a polynomial to be an interpolating polynomial of a labeled event tree.

Recall that for a power-product $\boldsymbol{\theta}^a = \theta_1^{a_1}, \ldots, \theta_d^{a_d}$, the degree is the sum of the exponents, $\deg(\boldsymbol{\theta}^a) = \sum_{j=1}^d a_j$, and for a polynomial $c = \sum_{i=1}^d \boldsymbol{\theta}^{\alpha_i}$ the degree is $\deg(c) = \max\{\deg(\boldsymbol{\theta}^{\alpha_i})\}$.

Proposition 4. *Let $c(\boldsymbol{\theta}) = \sum_{i=1}^n \boldsymbol{\theta}^{\alpha_i}$ be a polynomial with square-free support, i.e. $\alpha_i = (a_{i1}, \ldots, a_{id}) \in \{0,1\}^d$ for all $i = 1, \ldots, n$ and some $d \geq 1$. If there exists a labeled event tree such that c is its interpolating polynomial then the following conditions hold:*

1. *If $c \neq 1$ then $d, n \geq 2$ and $d \leq 2n-2$, and $d > \deg(c)$.*
2. *The frequency with which each root label appears in the monomials $\boldsymbol{\theta}^{\alpha_i}$, $i = 1, \ldots, n$, is greater than the degree of the monomials in which they appear.*
3. *If the degree of $\boldsymbol{\theta}^{\alpha_i}$ is equal to the degree of c, then there exists $\boldsymbol{\theta}_j$ with $i \neq j$ with the same degree as $\boldsymbol{\theta}^{\alpha_i}$ and the degree of the greatest common divisor of $\boldsymbol{\theta}_j$ and $\boldsymbol{\theta}_i$ is equal to the degree of c minus one.*
4. *No power-product in the support of c can be a proper multiple of another.*

Proof. 1. The root floret of a labeled event tree with at least one edge has at least two edges with distinct labels, thus $d, n \geq 2$. We prove the claim by induction on the

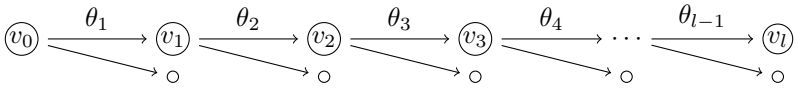


Figure 7: A root-to-leaf path $\lambda = (e_1, \ldots, e_l)$ in an event tree.

number of florets in a labeled event tree. Let E be the set of edges and L the set of leaves of the tree. If a tree is formed by a single vertex then $\#E = 0$ and $\#L = 1$. Therefore $\#E = 0 = 2\#L-2$. By induction suppose that $\#E \leq 2\#L - 2$ for the tree $\mathcal{T}$. Consider the tree $\mathcal{T}'$ obtained by adding to a leaf in $\mathcal{T}$ a floret with s edges. Because $s \geq 2$, thus $s \leq 2s-2$ and hence $\#E' = \#E+s$ and $\#L' = \#L+s-1$. As a result, $\#E' = (\#E) + (s) \leq (2\#L-2) + (2s-2) = 2(\#L + s-1) - 2 = 2\#L'-2$. We conclude by noticing that $d \leq \#E$ and $n = \#L$.

2. Consider Fig. 7. In labeled event trees, an atomic monomial of degree $l \in \mathbb{N}$ is associated to a root-to-leaf path of length l. This path has one bifurcation at every vertex, so is embedded in a graph with at least $l+1$ distinct root-to-leaf paths. So every root-label θ_1 occurs in monomials of maximal degree l and there are at least $l+1$ of those.

3. Because $\#E_v \geq 2$ for all $v \in V$, every leaf-floret has two edges. There are hence at least two monomials of the same maximal degree, namely those belonging to the longest paths in the tree: these are equal until they split at a leaf-floret.

4. Let t_1 and t_2 in c be multiples of each other, written as $t_1|t_2$. They are atomic monomials of two root-to-leaf paths, λ_1 and λ_2, which are not empty if $\mathcal{T}$ is not trivial. Let e be the root edge labeled θ_1, the first edge in λ_1. Then λ_2 starts with the same edge: otherwise $\theta_1|t_1$, and $\theta_1 \not| t_2$ for Proposition 3. Therefore we can repeat the reasoning on $\lambda_1\backslash\{e\}$ and $\lambda_2\backslash\{e\}$ in the subtree $\mathcal{T}(w)$. After a finite number of steps we can then conclude $\lambda_1 = \lambda_2$ and thus $t_1 = t_2$.

□

The conditions in Proposition 4 are necessary but not sufficient.

Example 16. The polynomial $\theta_1\phi_1 + \theta_1\phi_2 + \theta_2\theta_3\theta_4 + \theta_2\theta_3\phi_1 + \theta_2\theta_4\phi_2$ satisfies all points in Proposition 4. However, it cannot be written in the form of a nested representation. It is thus not the interpolating polynomial of a labeled event tree.

6 Two other representations of labeled event trees

From the previous section we see that if there is a labeled event tree for a square-free polynomial c with n terms then that tree has n root-to-leaf paths. Every such path is labeled by a monomial $\boldsymbol{\theta}^{\alpha}$ which is a power-product in support(c). We next present two well-known alternative representations of these atomic monomials of a staged tree.

The first representation is based on the notion of an abstract simplicial complex, i.e. a family $\mathcal{G}$ of subsets of a finite set (the nodes of the simplicial complex) such that if $A \in \mathcal{G}$ and $B \subseteq A$ then $B \in \mathcal{G}$. In our case the nodes of the simplicial complex are the labels Θ of a labeled event tree $\mathcal{T} = (T, \theta)$ and the family is given by the monomials

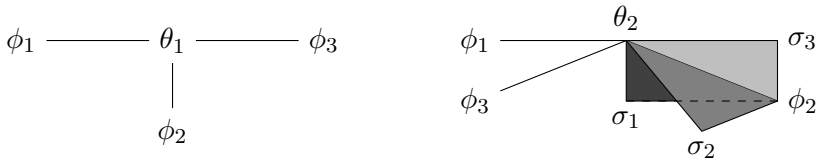


Figure 8: The simplicial complex for the staged tree with nested representation $c_{\mathcal{T}} = \theta_1(\phi_1 + \phi_2 + \phi_3) + \theta_2(\phi_1 + \phi_2(\sigma_1+\sigma_2+\sigma_3) + \phi_3)$ given in (6a) is the direct sum of the two simplicial complexes above. The three triangles in the right hand complex with vertices $\theta_2\phi_2\sigma_i$, $i = 1, 2, 3$, correspond to the root-to-leaf paths of length three.

$\pi_\theta(\lambda_i) = \boldsymbol{\theta}^{\alpha_i}$, $i = 1, \ldots, n$, and all of their divisors. For an illustration see Fig. 8. This graphical representation for a set of monomials has been successfully used in the data analysis of complex systems [5, 9, 22].

Proposition 5. *A labeled event tree $\mathcal{T}$ is saturated with root labels $\theta_1, \ldots, \theta_k$ if and only if its associated simplicial complex $\mathcal{G} = \mathcal{G}_1 \oplus \mathcal{G}_2 \oplus \ldots \oplus \mathcal{G}_k$ is the disjoint union of k connected simplicial complicies and the vertex of maximal degree within each complex is a root-label.*

Proof. Let $\mathcal{T}$ be a saturated tree. If no edge labels are identified, then writing (5) as $c_{\mathcal{T}} = \sum_{i=1}^{k} \theta_i c_i$ we find that no two c_i and c_j, $i \neq j$, have any indeterminates in common, $i, j = 1, \ldots, k$. Thus, we can split the set of atomic monomials $\boldsymbol{\theta}^{\alpha_i}$, $i = 1, \ldots, n$, into k disjoint sets, each given by the monomial terms in one $\theta_i c_i$. This gives us the disjoint union of $\mathcal{G} = \mathcal{G}_1 \oplus \mathcal{G}_2 \oplus \ldots \oplus \mathcal{G}_k$. By the linear expansion of the interpolation polynomial, the vertex θ_i is connected to every other monomial in $\mathcal{G}_i$. It is thus of highest degree in the sense that it has the highest number of emanating edges. For if in $\mathcal{G}_i$ there was a second vertex θ_j, $i \neq j$, of equally high degree then both θ_i and θ_j would divide every monomial in that subset. But by definition a sequence of single edges, here labeled θ_i and θ_j, is not possible.

Conversely, assume we have a set of monomials belonging to an event tree. Then the associated simplicial complex is the disjoint union of simplicial complicies $\mathcal{G} = \mathcal{G}_1 \oplus \mathcal{G}_2 \oplus \ldots \oplus \mathcal{G}_k$ where each $\mathcal{G}_i$ has a vertex θ_i of highest degree, $i = 1, \ldots, k$. Thus, we can write the corresponding interpolating polynomial in the form (5). Because no $\mathcal{G}_i$ is connected to any $\mathcal{G}_j$ for $i \neq j$, the terms belonging to one sub-simplicial complex have no indeterminates in common with those belonging to the other. Thus the subtrees rooted after the root do not have any labels in common. Therefore the original tree is saturated. □

The proposition enables us to use this simplicial complex representation of an interpolating polynomial to quickly decide whether or not the corresponding labeled event tree is saturated. Thus, by Proposition 2, we will know whether or not we need to check for different nested representations of its interpolating polynomial, or whether or not any representation that is discovered is unique . If a tree is saturated, we can then resize it to a simpler graphical representation as in Example 11.

The other natural representation of these monomials is via an incidence matrix. Let $\mathcal{T} = (T, \theta)$ be a labeled event tree with monomials $\theta_1^{\alpha_{1,j}}\theta_2^{\alpha_{2,j}} \cdots \theta_d^{\alpha_{d,j}} = \boldsymbol{\theta}^{\alpha_j}$, for $\alpha_j \in \mathbb{Z}^d_{\geq 0}$

and $j = 1, \ldots, n$. The interpolating polynomial of $\mathcal{T}$ can be visualized by a $d \times n$ matrix $A_{\mathcal{T}} = (a_{ij})_{ij}$ with integer non-negative entries such that

$$a_{ij} = \begin{cases} m & \text{if } \theta_i^m \text{ divides } \boldsymbol{\theta}^{\alpha_j} \text{ and } m \in \mathbb{N} \text{ is maximal} \\ 0 & \text{otherwise.} \end{cases}$$

If the atomic monomials in $\mathcal{T}$ are square-free then $A_{\mathcal{T}}$ is a matrix with entries 0 or 1. The matrix $A_{\mathcal{T}}$ codes a number of properties of the atomic monomials of $\mathcal{T}$. In particular, every column encodes those indeterminates which divide the associated monomial, so column sums are the degree of the monomial indexing the column. Every row sum codes the number of monomials which are divided by a certain indeterminate. In order for a set of monomials to be associated to a tree, we need that

$$\sum_{i=1,\ldots,d} a_{il} \;<\; \sum_{j=1,\ldots,n} a_{kj}$$

for all pairs of k, l. This follows from Proposition 4.2. Submatrices of $A_{\mathcal{T}}$ can easily be associated to subtrees of $\mathcal{T}$. For instance for a subtree $\mathcal{T}_v \subseteq \mathcal{T}$ rooted after an edge $(\cdot, v)$ labeled θ_i, we cancel all rows $a_{i\cdot}$ and all columns $a_{\cdot j}$ from the matrix which include an entry $a_{ij} = 0$. The remaining matrix $A_{\mathcal{T},i} = A_{\mathcal{T}_v}$ is then the incidence matrix of $\mathcal{T}_v$.

For example, the incidence matrix $A_{\mathcal{T}}$ for the interpolating polynomial $c_{\mathcal{T}}$ in (2) of the trees in Fig. 2 is

$$\begin{array}{c} \\ \theta_1 \\ \theta_2 \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{array} \begin{array}{c} \begin{array}{cccccccc} \theta_1\phi_1 & \theta_1\phi_2 & \theta_1\phi_3 & \theta_2\phi_1 & \theta_2\phi_3 & \theta_2\phi_2\sigma_1 & \theta_2\phi_2\sigma_2 & \theta_2\phi_2\sigma_3 \end{array} \\ \left(\begin{array}{cccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right) \end{array}$$

The sum of the first two rows in this matrix is a vector with all entries equal to one and the labels indexing these first two rows are root-floret labels. This is not by chance. In fact, the full tree can be retrieved by splitting the set of columns into those which have one in the first row or in the second row and proceeding recursively. This procedure can be turned into a matrix version of the `StagedTree` algorithm.

This matrix representation enables us to link model representations given by labeled or staged trees to log-linear models and well-known results in algebraic stiatistics [13].

7 An application

In this section we will apply the algorithm presented in Section 4 to determine the full polynomial equivalence class of a staged tree representing the best fitting model inferred from a real-world dataset. The work of [11] provides an early analysis of what we will refer to as "the Christchurch dataset". These data have been collected on a cohort of

stage colour	label	interpretation
	$(a_1, a_2, a_3, a_4, a_5)$	access to credit: $++, \ldots, --$
	(h_1, h_2)	hospital admission: yes or no
	(l_1, l_2, l_3)	number of life events: high, average or low
	(l_3, l_4, l_5)	number of life events: high, average or low
	(a_1, a_2, a_3)	access to credit: $++, +-, -+$

Table 1: The labels of the staged trees in Fig. 9, used in the interpolating polynomial (8).

nearly one thousand children over the course of thirty years and include measurements of a number of possibly relevant factors to determine the likelihood of child illness. These measurements can be grouped into the very broad categories of socio-economic background and number of life events – like divorce of its parents or death in the family – of a child, with respective states "high", "average" and "low". The state of health of a child is then assessed as hospital admission "yes" or "no" [3].

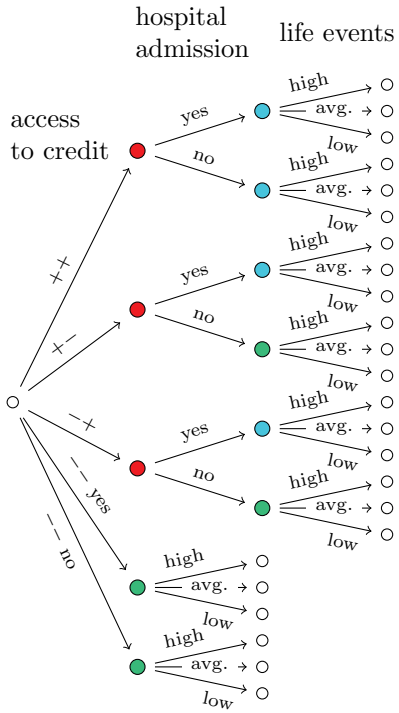
An MAP algorithm running on the Christchurch dataset determined the highest scoring staged tree representation among those which had all vertices that are in the same stage also at the same depth [7]. Later, [16] found a statistically equivalent but graphically simpler representation with no saturated subtrees. This staged tree (T, θ) is shown in Fig. 9a. Here, socio-economic background of a child has been modified to a measure of the access to credit which can be high $(++)$, moderately high $(+-$ or $-+)$ or low $(--)$. The colouring of the staged tree then indicates a number of interesting conditional independence statements. For instance, the red stages on the first level of the tree state that the likelihood of hospital admission was inferred to be the same for all children from a family with high or moderately high access to credit. The blue stages on the subsequent level add that the number of life events of a child is independent of it being admitted to hospital given that its family's access to credit was high, but different given that its access to credit was low. From the green stages we can see that for children with moderate access to credit the likelihood of a certain quantity of life events is not independent of admission to hospital.

The order of events depicted by the staged tree in Fig. 9a suggests that the number of life events of a child might be a putative cause of its admission to hospital. The analysis of [7, 16] then showed that in fact when keeping the original problem variables intact across the class of staged trees which are statistically equivalent to (T, θ), this order is preserved. This interpretation of the tree's directionality thus seems to be supported by the Christchurch data.

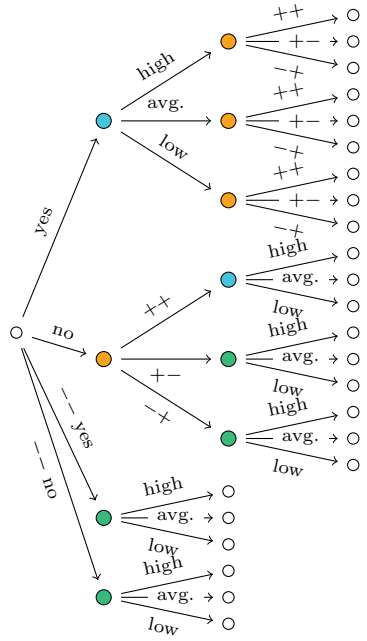
We will now use the algorithm `StagedTrees` in Section 4.2 to automatically determine the polynomial equivalence class of $\mathcal{T} = (T, \theta)$. To this end we first specify the interpolating polynomial for the tree in Fig. 9a, using labels as specified in Table 1:

$$\begin{aligned} c_{\mathcal{T}}(\boldsymbol{a}, \boldsymbol{h}, \boldsymbol{l}) \;=\; & a_1h_1l_1 + a_1h_1l_2 + a_1h_1l_3 + a_1h_2l_1 + a_1h_2l_2 + a_1h_2l_3 \\ & + a_2h_1l_1 + a_2h_1l_2 + a_2h_1l_3 + a_2h_2l_4 + a_2h_2l_5 + a_2h_2l_6 \\ & + a_3h_1l_1 + a_3h_1l_2 + a_3h_1l_3 + a_3h_2l_4 + a_3h_2l_5 + a_3h_2l_6 \\ & + a_4l_4 + a_4l_5 + a_4l_6 + a_5l_4 + a_5l_5 + a_5l_6 \end{aligned} \tag{8}$$

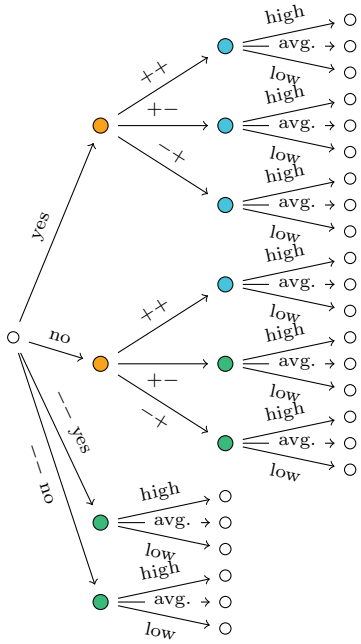
where $\boldsymbol{a} = (a_1, a_2, a_3, a_4, a_5)$, $\boldsymbol{h} = (h_1, h_2)$ and $\boldsymbol{l} = (l_1, l_2, l_3, l_4, l_5, l_6)$ are the respective



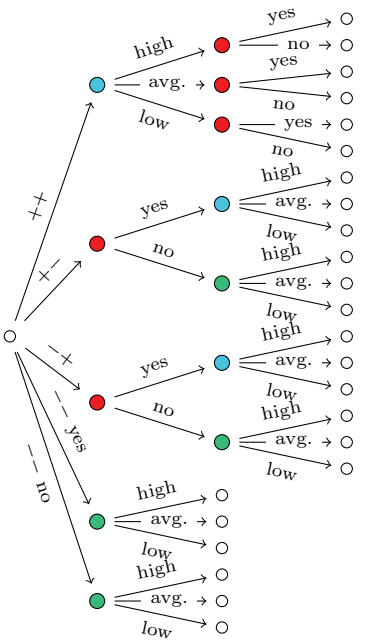
(a) The original staged tree (T, θ) from Fig. 4(b) in [16]. See (9).



(b) A staged tree $(T, \theta)_1$ with nested representation (10).



(c) A staged tree $(T, \theta)_2$ with nested representation (11).



(d) A staged tree $(T, \theta)_3$ with nested representation (12).

Figure 9: All four elements of the polynomial equivalence class of $c_{\mathcal{T}}$ in (8).

(conditional) probabilities of different degress of **a**ccess to credit, **h**ospital admission and numbers of **l**ife events, read from left to right and from top to bottom along the root-to-leaf paths of $\mathcal{T}$.

Running `StagedTrees`, we find precisely four different nested representations of $c_{\mathcal{T}}$. These are:

$$\begin{aligned}
r_0(c_{\mathcal{T}}) &= a_1(h_1(l_1+l_2+l_3)+h_2(l_1+l_2+l_3)) \\
&\quad + a_2(h_1(l_1+l_2+l_3)+h_2(l_4+l_5+l_6)) \\
&\quad + a_3(h_1(l_1+l_2+l_3)+h_2(l_4+l_5+l_6)) \\
&\quad + a_4(l_4+l_5+l_6)+a_5(l_4+l_5+l_6) \qquad (9)\\
r_1(c_{\mathcal{T}}) &= h_1(l_1(a_1+a_2+a_3)+l_2(a_1+a_2+a_3)+l_3(a_1+a_2+a_3)) \\
&\quad + h_2(a_1(l_1+l_2+l_3)+a_2(l_3+l_4+l_5)+a_3(l_3+l_4+l_5)) \\
&\quad + a_4(l_4+l_5+l_6)+a_5(l_4+l_5+l_6) \qquad (10)\\
r_2(c_{\mathcal{T}}) &= h_1(a_1(l_1+l_2+l_3)+a_2(l_1+l_2+l_3)+a_3(l_1+l_2+l_3)) \\
&\quad + h_2(a_1(l_1+l_2+l_3)+a_2(l_3+l_4+l_5)+a_3(l_3+l_4+l_5)) \\
&\quad + a_4(l_4+l_5+l_6)+a_5(l_4+l_5+l_6) \qquad (11)\\
r_3(c_{\mathcal{T}}) &= a_1(l_1(h_1+h_2)+l_2(h_1+h_2)+l_3(h_1+h_2)) \\
&\quad + a_2(h_1(l_1+l_2+l_3)+h_2(l_4+l_5+l_6)) \\
&\quad + a_3(h_1(l_1+l_2+l_3)+h_2(l_4+l_5+l_6)) \\
&\quad + a_4(l_4+l_5+l_6)+a_5(l_4+l_5+l_6) \qquad (12)
\end{aligned}$$

where for now r_i denotes one fixed order of summation in a nested representation, $i = 0, 1, 2, 3$.

By Proposition 1, $r_0(c_{\mathcal{T}})$ is the nested factorisation of (T, θ). In Fig. 9b we have drawn the staged tree $(T, \theta)_1$ corresponding to the representation $r_1(c_{\mathcal{T}})$, in Fig. 9c the staged tree $(T, \theta)_2$ corresponding to $r_2(c_{\mathcal{T}})$ and in Fig. 9d the staged tree $(T, \theta)_3$ corresponding to $r_3(c_{\mathcal{T}})$. These staged trees are the only labeled event trees with the above interpolating polynomial on which sum-to-1 conditions imposed on florets induce a probability distribution over the depicted atoms. So in Fig. 9 we see all four elements of the polynomial equivalence class of (T, θ). By Definition 5, these staged trees all represent the same underlying model. So we can now analyse the orders in which the same events are depicted across different graphs.

Because in Fig. 9a and 9c all vertices in the same stage are also at the same distance from the leaves, we can in this case assign an interpretation to each such level of the tree. So in Fig. 9a the first level of (T, θ) depicts all states of the random variables access to credit, the second level depicts all states of the random variable hospital admission and the third and last level depicts all states of the random variable life events. Now this interpretation has been reversed in Fig. 9c. In $(T, \theta)_2$, the third level still depicts life events but the first two levels have been interchanged. The first level now represents the states of a joint random variable "hospital admission" and "hospital admission having low access to credit". The second level then depicts access to credit with states "high" and "moderately high". So because both (T, θ) and $(T, \theta)_2$ represent the same model with (T, θ) showing access to credit before hospital admission and $(T, \theta)_2$ reversing that order,

we cannot hypothesize a putative causal relationship on these (conditionally independent) variables: see [27] for a more thorough presentation of this very subtle point.

It is less straightforward to assign a meaning in terms of problem variables to the staged trees in Fig. 9b and 9d. However, we can still see when comparing $(T,\theta)_1$ with $(T,\theta)_2$ or (T,θ) with $(T,\theta)_3$ that only for children from a family with high access to credit is the order of hospital admission and life events reversible. In all other circumstances the model depicts hospital admission before life events. As in [7, 16], we therefore might want to assign this a putative causal interpretation.

Acknowledgments

Christiane Görgen was supported by the EPSRC grant EP/L505110/1. Part of this research was supported through the programme "Oberwolfach Leibniz Fellows" by the Mathematisches Forschungsinstitut Oberwolfach in 2017. During some of this development Jim Q. Smith was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

Appendix

Square-free monomial ideals

We summarize here the notions from commutative algebra which have been mentioned in this paper.

Given a non-zero polynomial $f \in \mathbb{R}[x_1,\dots,x_d]$, with **coefficients** in $\mathbb{R}$ and **indeterminates** (or variables) $x_1,\dots,x_d$, f is uniquely written as $f = \sum_{i=1}^{s} \beta_i t_i$, with coefficients $b_i \neq 0$, and **power-products** (or terms, or monomials) $t_i = x_1^{\alpha_{i,1}} \cdots x_d^{\alpha_{i,d}}$ all distinct, for every $i = 1,\dots,s$.

The **support** of a polynomial f is the set of the power-products actually occurring in f. With the notation above, $\text{support}(f) = \{t_i \mid i = 1,\dots,s\}$.

An **ideal** generated by a set of polynomials, say $I = \langle f_1,\dots,f_k\rangle$, is the set of all linear combinations with polynomial coefficients, i.e. $I = \{g_1 f_1 + \cdots + g_k f_k \mid g_i \in \mathbb{R}[x_1,\dots,x_d] \text{ for } i = 1,\dots,k\}$. In particular, if all f_i's are power-products, I is called a **monomial ideal**. If a power-product has all exponents in $\{0,1\}$, it is said **square-free**, and an ideal generated by square-free power-products is called **square-free monomial ideal**.

Given a monomial ideal I, a **minimal prime** of I is an ideal $\mathcal{P}$ generated by a subset of the indeterminates $\{x_1,\dots,x_d\}$ such that I is contained in $\mathcal{P}$, but is not contained in any ideal generated by a subset of the generators of $\mathcal{P}$ (used in Theorem 2).

An ideal is **primary** if $fg \in I$ implies either $f \in I$ or some power $g^m \in I$ (for some integer $m > 0$). All ideals in $\mathbb{R}[x_1,\dots,x_d]$ admit a **primary decomposition**, i.e. may be written as an intersection of primary ideals. In the particular case of interest in this paper, a square-free monomial ideal has primary decomposition $I = \mathcal{P}_1 \cap \dots \mathcal{P}_\ell$, where the primary ideals $\mathcal{P}_i$ are indeed the minimal primes of I. In general, the **prime decomposition** of an ideal is given by the minimal primes of the ideal (used in Example 14), and is the primary decomposition of the radical of the ideal.

In general, computing the primary decomposition of a polynomial ideal is quite difficult, but for monomial ideals the operations are a lot easier. In particular, for square-free monomial ideals there is a very simple and efficient algorithm called **Alexander Dual**.

References

[1] J. Abbott, A.M. Bigatti, and G. Lagorio. CoCoA-5: a system for doing Computations in Commutative Algebra. Available at `http://cocoa.dima.unige.it`, 2016.

[2] Steen A. Andersson, David Madigan, and Michael D. Perlman. A characterization of Markov Equivalence Classes for Acyclic Digraphs. *Ann. Statist.*, 25(2):505–541, 1997.

[3] Lorna M. Barclay, Jane L. Hutton, and Jim Q. Smith. Refining a Bayesian network using a Chain Event Graph. *Internat. J. Approx. Reason.*, 54(9):1300–1309, 2013.

[4] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *Uncertainty in Artificial Intelligence (Portland, OR, 1996)*, pages 115–123. Morgan Kaufmann, San Francisco, CA, 1996.

[5] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[6] Rodrigo A. Collazo and Jim Q. Smith. A New Family of Non-Local Priors for Chain Event Graph Model Selection. *Bayesian Anal.*, 11(4):1165–1201, 2015.

[7] Robert G. Cowell and Jim Q. Smith. Causal discovery through MAP selection of stratified Chain Event Graphs. *Electron. J. Stat.*, 8(1):965–997, 2014.

[8] Adnan Darwiche. A differential approach to inference in Bayesian networks. *J. ACM*, 50(3):280–305 (electronic), 2003.

[9] I Donato, M Gori, M Petini, G Petri, S De Nigris, R Franzosi, and F Vaccarino. Persistent homology analysis of phase transitions. *Physical Review E*, 93(5):052138, 2016.

[10] Mathias Drton, Bernd Sturmfels, and Seth Sullivant. *Lectures on algebraic statistics*, volume 39 of *Oberwolfach Seminars*. Birkhäuser Verlag, Basel, 2009.

[11] D. M. Fergusson, L. J. Horwood, and F. T. Shannon. Social and family factors in childhood hospital admission. *Journal of Epidemiology and Community Health*, 40:50–58, 1986.

[12] Guy Freeman and Jim Q. Smith. Bayesian MAP model selection of Chain Event Graphs. *J. Multivariate Anal.*, 102(7):1152–1165, 2011.

[13] Dan Geiger, Christopher Meek, and Bernd Sturmfels. On the toric algebra of graphical models. *Ann. Statist.*, 34(3):1463–1492, 2006.

[14] Christiane Görgen. *An algebraic characterisation of staged trees: their geometry and causal implications.* PhD thesis, University of Warwick, Department of Statistics, 2017.

[15] Christiane Görgen, Manuele Leonelli, and Jim Q. Smith. A Differential Approach for Staged Trees. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Proceedings*, Lecture Notes in Artificial Intelligence, pages 346–355. Springer, 2015.

[16] Christiane Görgen and Jim Q. Smith. Equivalence Classes of Staged Trees. *Bernoulli (forthcoming).* Preprint available from `arXiv:1512.00209 [math.ST]`, 2017.

[17] Daphne Koller and Avi Pfeffer. Object-Oriented Bayesian Networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 302–313, 1997.

[18] Martin Kreuzer and Lorenzo Robbiano. *Computational commutative algebra. 1.* Springer-Verlag, Berlin, 2000.

[19] Steffen L. Lauritzen. *Graphical models*, volume 17 of *Oxford Statistical Science Series*. The Clarendon Press, Oxford University Press, New York, 1996. Oxford Science Publications.

[20] Manuele Leonelli, Christiane Görgen, and Jim Q. Smith. Sensitivity analysis, multilinearity and beyond. Preprint available from *arXiv:1512.02266 [cs.AI]*, 2015.

[21] Lior Pachter and Bernd Sturmfels, editors. *Algebraic statistics for computational biology*. Cambridge University Press, New York, 2005.

[22] Virginia Pirino, Eva Riccomagno, Sergio Martinoia, and Paolo Massobrio. A topological study of repetitive co-activation networks in in vitro cortical assemplies. *Physical Biology*, 12(1):016007, February 2015.

[23] Giovanni Pistone, Eva Riccomagno, and Henry P. Wynn. *Algebraic Statistics*, volume 89 of *Monographs on Statistics and Applied Probability*. Chapman & Hall/CRC, Boca Raton, FL, 2001. Computational commutative algebra in statistics.

[24] Antonio Salmerón, Andrés Cano, and Serafín Moral. Importance sampling in Bayesian networks using probability trees. *Comput. Statist. Data Anal.*, 34(4):387–413, 2000.

[25] Glenn Shafer. *The Art of causal Conjecture*. MIT Press, Cambridge, 1996.

[26] Jim Q. Smith and Paul E. Anderson. Conditional independence and Chain Event Graphs. *Artificial Intelligence*, 172(1):42–68, 2008.

[27] Jim Q. Smith, Christiane Görgen, and Rodrigo A. Collazo. *Chain Event Graphs*. Chapman & Hall (forthcoming), 2017.

[28] Peter A. Thwaites, Jim Q. Smith, and Robert G. Cowell. Propagation using Chain Event Graphs. In *Proceedings of the 24th Conference on Uncertainty in Artifical Intelligence*, pages 546–553, Helsinki, 2008.

Authors' addresses

Christiane Görgen, `goergen@mis.mpg.de`, Max-Planck-Institute for Mathematics in the Sciences, Leipzig, Germany.

Anna Bigatti, `bigatti@dima.unige.it`, Dipartimento di Matematica, Università degli Studi di Genova, 16146 Genova, Italy.

Eva Riccomagno, `riccomagno@dima.unige.it`, Institute of Intelligent Systems for Automation, National Research Council, Italy; and Dipartimento di Matematica, Università degli Studi di Genova, 16146 Genova, Italy.

Jim Q. Smith, `j.q.smith@warwick.ac.uk`, Department of Statistics, University of Warwick, Coventry CV5 7AL, U.K.; and The Alan Turing Institute, British Library, 96 Euston Road, NW1 2DB London, U.K..