Available online at www.sciencedirect.com
ScienceDirect
International Journal of Approximate Reasoning
45 (2007) 386–401
INTERNATIONAL JOURNAL OF APPROXIMATE REASONING
www.elsevier.com/locate/ijar

Racing algorithms for conditional independence inference

Remco R. Bouckaert [a,*], Milan Studený [b,1]

[a] *Computer Science Department, University of Waikato & Xtal Mountain Information Technology, 1st Floor, G-Block, Gate 8, Hillcrest Road, Hamilton 3240, New Zealand*
[b] *Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, Prague, Czech Republic*

Received 30 December 2005; received in revised form 15 June 2006; accepted 30 June 2006
Available online 22 August 2006

Abstract

In this article, we consider the computational aspects of deciding whether a conditional independence statement t is implied by a list of conditional independence statements L using the independence implication provided by the method of structural imsets. We present two algorithmic methods which have the interesting complementary properties that one method performs well to prove that t is implied by L, while the other performs well to prove that t is not implied by L. However, both methods do not well perform the opposite. This gives rise to a parallel algorithm in which both methods race against each other in order to determine effectively whether t is or is not implied.

Some empirical evidence is provided that suggests this racing algorithms method performs considerably better than an existing method based on so-called skeletal characterization of the respective implication. Furthermore, unlike previous methods, the method is able to handle more than five variables.

Keywords: Conditional independence; Inference; Imset; Algorithm

* Corresponding author.
E-mail addresses: remco@cs.waikato.ac.nz, rrb@xm.co.nz (R.R. Bouckaert), studeny@utia.cas.cz (M. Studený).
[1] The work of the second author has been supported by the grant GAČR no. 201/04/0393.

0888-613X/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2006.06.018

1. Introduction

Conditional independence (CI) is a crucial notion in statistics [6,4] and in many calculi for dealing with knowledge and uncertainty in artificial intelligence [11,12]. A powerful formalism for describing probabilistic CI structures is provided by the method of structural imsets [17]. In this algebraic approach, CI structures are described by certain vectors whose components are integers, called *structural imsets*. An important question is to decide whether a CI statement is implied by a set of CI statements. The method of structural imsets offers a sufficient condition for the probabilistic implication of CI statements. The offered inference mechanism is based on linear algebraic operations with (structural) imsets. The basic idea is that every CI statement can be translated into a simple imset and the respective algebraic relation between imsets, called *independence implication*, forces the probabilistic implication of the respective CI statements. Techniques were developed in [15] to test the independence implication through systematic calculation of certain inner products. However, these techniques are for some computational reasons applicable only when there are up to five variables involved.

For reasoning with CI statements involving more than five variables one may resort to making strict assumptions. For example, one can assume that the considered CI structure is graph isomorphic for a class of graphs such as directed acyclic graphs (DAG) [11,18], undirected graphs (UG) [4,8], chain graphs (CG) [3,19], etc. Then CI inference from a set of CI statements, a so-called *input list*, of a special form can be made as follows. The list is used to construct a graph and CI statements are read from the graph through the respective graphical separation criterion. However, the assumption that a CI structure is graph isomorphic may be too strong in many cases and only special input lists can be processed anyway. Using the method of structural imsets, many more CI structures can be described than with DAGs, UGs or CGs.

Many other graphical formalisms representing CI structures have been proposed for which it is not quite clear whether a particular efficient input list can be designed such that exactly all statements represented in the graph can be derived from the list. For example, generalized directed graphs [13], IDAGs [2], reciprocal graphs [10], joint response chain graphs [5], covariance graphs [9], alternative chain graphs [1] and others. What these methods have in common with DAGs, UGs and CGs is that their representative power (with an increasing number of variables) tends to be lower than those of structural imsets – see Section 3.6 in [17].

However, the computational effort required for inference using structural imsets when more than five variables are involved is not clear at present. Fortunately, they have some properties that we can exploit. First, a relatively easy sufficient condition for independence implication is that a certain corresponding linear combination of imsets can be decomposed into so-called *elementary imsets*. The existence of this decomposition can be found relatively quickly. On the other hand, to prove that the decomposition does not exist requires trying all potential decompositions, which often takes a long time. Second, there exists a method to show that the independence implication does not hold. It suffices to find a certain vector, called a *supermodular function*, such that its inner product with the respective combination of structural imsets is negative. These supermodular functions can be generated randomly. This only allows us to disprove independence implication of imsets, not to disprove probabilistic implication of the respective CI statements. However, if the obtained supermodular function is a multiple of a multiinformation function of a

probability distribution [17] then it also allows us to disprove probabilistic implication of the respective CI statements. Thus, we have one method that allows us to find a proof that a statement is implied, and one method to find a proof that a statement is not implied. However, both methods perform poorly in proving their opposite outcome. This gives rise to a race: both methods are started at the same time and the method that returns its outcome first also returns a proof whether the statement of interest is implied or not.

The following section introduces formal terminology and the fundamentals for CI inference using structural imsets. The racing algorithms are described in Section 3 where many more smaller optimizations are mentioned as well. Section 4 presents experiments that were performed to get an impression of the run-times of various variants of inference algorithms. We conclude with some final comments and directions for further research.

2. Terminology

Let N be a set of variables $\{x_1, \ldots, x_n\}$, $n \geqslant 1$, as will be assumed throughout the paper. Let X and Y be subsets of N. We will use XY to denote the union of X and Y and $X \setminus Y$ to denote the set of variables that are in X but not in Y. Further, if x is a variable in N then x will also denote the singleton $\{x\}$.

2.1. Conditional independence

Let P be a discrete probability distribution over N and X, Y, Z pairwise disjoint subsets of N. We say that X is *conditionally independent* of Y given Z if $P(\mathbf{x}|\mathbf{yz}) = P(\mathbf{x}|\mathbf{z})$ for all configurations $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ of values for X, Y, Z with $P(\mathbf{yz}) > 0$. We write then $X \perp\!\!\!\perp Y|Z$ $[P]$ or just $X \perp\!\!\!\perp Y|Z$, and call it a *CI statement*. It is well-known that CI follows some simple rules, known as the *semi-graphoid axioms* [11] defined as follows ($X, Y, Z, W \subseteq N$ are pairwise disjoint):

Symmetry	$X \perp\!\!\!\perp Y\|Z \;\Rightarrow\; Y \perp\!\!\!\perp X\|Z,$
Decomposition	$X \perp\!\!\!\perp WY\|Z \;\Rightarrow\; X \perp\!\!\!\perp Y\|Z,$
Weakunion	$X \perp\!\!\!\perp WY\|Z \;\Rightarrow\; X \perp\!\!\!\perp W\|YZ,$
Contraction	$X \perp\!\!\!\perp W\|YZ \;\&\; X \perp\!\!\!\perp Y\|Z \;\Rightarrow\; X \perp\!\!\!\perp WY\|Z.$

The problem we address in this paper is the following *inference problem*. Let L be a set of CI statements (over N), called an *input list* and t is a CI statement $X \perp\!\!\!\perp Y|Z$ outside L. Does L imply t? More formally, is it true that for any discrete distribution P for which all statements in L are valid necessarily t is valid as well? This is a *probabilistic implication* of those CI statements, sometimes denoted by $L \models t$. The semi-graphoid axioms do not cover this implication. For example,

$$X \perp\!\!\!\perp Y|WZ \;\&\; W \perp\!\!\!\perp Z|X \;\&\; W \perp\!\!\!\perp Z|Y \;\&\; X \perp\!\!\!\perp Y|\emptyset \iff \tag{1}$$

$$\iff W \perp\!\!\!\perp Z|XY \;\&\; X \perp\!\!\!\perp Y|Z \;\&\; X \perp\!\!\!\perp Y|W \;\&\; W \perp\!\!\!\perp Z|\emptyset \tag{2}$$

is also a valid rule – see p. 16 in [17]. In fact, there is no complete set of rules of this kind describing relationships between probabilistic CI statements [14]. A more powerful formalism to describe the properties of CI is provided by the method of structural imsets.

2.2. Imsets

An *imset* over N (abbreviation for integer-valued multiset is an integer-valued function on the power set of N. It can be viewed as a vector whose components, indexed by subsets of N, are integers. Given $X \subseteq N$, we use δ_X to denote the identifier imset, that is, $\delta_X(X) = 1$ and $\delta_X(Y) = 0$ for all $Y \subseteq N$, $Y \neq X$. An imset associated with a CI statement $X \perp\!\!\!\perp Y|Z$ is $u_{\langle X,Y|Z\rangle} = \delta_{XYZ} + \delta_Z - \delta_{XZ} - \delta_{YZ}$. The imset associated with an input list L is then $u_L = \sum_{t\in L} u_t$.

The basic technique for inference of a statement t from an input list L using the method of structural imsets is based on the following property. If $k \cdot u_L$ (for some natural number $k \in \mathbb{N}$) can be written as u_t plus the sum of some imsets associated with CI statements then t is implied by L. This conclusion can be derived from results of [17]; however, in this paper, we intentionally omit technical details. For example, if the list L consists of a single statement $X \perp\!\!\!\perp WY|Z$ and t is $X \perp\!\!\!\perp Y|Z$, we have (with $k = 1$)

$$\begin{aligned} k \cdot u_L &= \delta_{WXYZ} + \delta_Z - \delta_{XZ} - \delta_{WYZ} \\ &= (\delta_{XYZ} + \delta_Z - \delta_{XZ} - \delta_{YZ}) + (\delta_{WXYZ} + \delta_{YZ} - \delta_{XYZ} - \delta_{WYZ}) \\ &= u_t + u_{\langle X,W|YZ\rangle}. \end{aligned}$$

Thus, $X \perp\!\!\!\perp WY|Z$ implies t and we have derived the decomposition rule of the semi-graphoid axioms. Note that any statement in the decomposition on the right-hand side can be swapped for t, so those statements are implied too. This means that above we have derived weak union as well.

An *elementary imset* is an imset associated with an elementary CI statement $x \perp\!\!\!\perp y|Z$, where x, y are singletons; namely $u_{\langle x,y|Z\rangle} = \delta_{xyZ} + \delta_Z - \delta_{xZ} - \delta_{yZ}$. It is convenient to denote the set of elementary imsets over N by $\mathscr{E}(N)$ or simply $\mathscr{E}$. A *structural imset* is an imset u that can be decomposed into elementary imsets when multiplied by a positive natural number, that is,

$$n \cdot u = \sum_{v\in\mathscr{E}} k_v \cdot v$$

for some $n \in \mathbb{N}$ and $k_v \in \mathbb{Z}^+$. Note that every structural imset induces a whole CI structure through an algebraic criterion, which is omitted here. The attraction of the method of structural imsets is that every discrete probabilistic CI structure can be described in this way – see Theorem 5.2 in [17].

A function m on the power set of N will be called *supermodular* if $m(XY) + m(X \cap Y) \geqslant m(X) + m(Y)$ for every pair of sets $X, Y \subseteq N$. An equivalent definition is that the inner product of m with any elementary imset is non-negative

$$\langle m, v\rangle \equiv \sum_{Z\subseteq N} m(Z) \cdot v(Z) \geqslant 0 \quad \text{for every } v \in \mathscr{E}.$$

Observe that a necessary condition for an imset u to be structural is $\langle m, u\rangle \geqslant 0$ for every supermodular function m.

Let u, v be structural imsets over N. We say that u *independence implies* v and write $u \rightharpoonup v$ if there exists $k \in \mathbb{N}$ such that $k \cdot u - v$ is a structural imset. This terminology is motivated by the fact that $u \rightharpoonup v$ actually means that u encodes more CI statements than v – see Lemma 6.1 in [17]. If $v \in \mathscr{E}$ then the constant $k \in \mathbb{N}$ can be supposed lower than a limit $k_{\max}$ depending on the number of variables $|N|$ – see Lemma 4 in [16]. However, the

value of the exact limit $k_{\max}$ for $|N| \geqslant 6$ is not known. It follows from results of [15] that $k_{\max} = 1$ if $|N| \leqslant 4$ and $k_{\max} = 7$ if $|N| = 5$.

Now, we can reformulate our inference problem. Given an elementary CI statement t and an input list (of elementary CI statements) L we are going to test whether $u_L \rightharpoonup u_t$. We have already mentioned that this is a sufficient condition for probabilistic implication of t by L. However, in general, $u_L \rightharpoonup u_t$ is not a necessary condition for $L \models t$.

Example. Assume $\{a,b,c,d,e\} \subseteq N$ and put

$$L = \{a \perp\!\!\!\perp b|cd, a \perp\!\!\!\perp c|de, a \perp\!\!\!\perp d|be, a \perp\!\!\!\perp e|bc\}.$$

We are interested in the question whether these statements are implied by L;

1. $a \perp\!\!\!\perp b|de$,
2. $a \perp\!\!\!\perp c|be$,
3. $a \perp\!\!\!\perp b|cde$, and
4. $a \perp\!\!\!\perp b|ce$.

The semi-graphoid axioms do not give any new statements apart from the symmetric versions. Thus, they are not of any help. However, using structural imsets, we can write (with $k = 1$)

$$\begin{aligned} k \cdot u_L = u_L &= u_{\langle a,b|cd\rangle} + u_{\langle a,c|de\rangle} + u_{\langle a,d|be\rangle} + u_{\langle a,e|bc\rangle} \\ &= (\delta_{abcd} + \delta_{cd} - \delta_{acd} - \delta_{bcd}) + (\delta_{acde} + \delta_{de} - \delta_{ade} - \delta_{cde}) \\ &\quad + (\delta_{abde} + \delta_{be} - \delta_{abe} - \delta_{bde}) + (\delta_{abce} + \delta_{bc} - \delta_{abc} - \delta_{bce}) \\ &= (\delta_{abde} + \delta_{de} - \delta_{ade} - \delta_{bde}) + (\delta_{abce} + \delta_{be} - \delta_{abe} - \delta_{bce}) \\ &\quad + (\delta_{abcd} + \delta_{bc} - \delta_{abc} - \delta_{bcd}) + (\delta_{acde} + \delta_{cd} - \delta_{acd} - \delta_{cde}) \\ &= u_{\langle a,b|de\rangle} + u_{\langle a,c|be\rangle} + u_{\langle a,d|bc\rangle} + u_{\langle a,e|cd\rangle}. \end{aligned}$$

Thus, the first and second statement ($a \perp\!\!\!\perp b|de$ and $a \perp\!\!\!\perp c|be$) are indeed implied by L.

However, the third statement $a \perp\!\!\!\perp b|cde$ is not independence implied by L since $u_{\langle a,b|cde\rangle} = \delta_{abcde} + \delta_{cde} - \delta_{acde} - \delta_{bcde}$ and no term δ_X with $abcde \subseteq X$ occurs anywhere in u_L. So, there is no way to decompose $k \cdot u_L$ into a sum containing δ_{abcde}. In more details, consider a supermodular function $m^{abcde\uparrow} \equiv \sum_{Z,abcde\subseteq Z} \delta_Z$. The inner product of $m^{abcde\uparrow}$ with u_L is zero, while its inner product with $u_t \equiv u_{\langle a,b|cde\rangle}$ is 1. In particular,

$$\langle m^{abcde\uparrow}, k \cdot u_L - u_t\rangle = k \cdot \langle m^{abcde\uparrow}, u_L\rangle - \langle m^{abcde\uparrow}, u_t\rangle = k \cdot 0 - 1 = -1,$$

and $k \cdot u_L - u_t$ is not a structural imset for any $k \in \mathbb{N}$. Therefore, it cannot be written as a sum of elementary imsets.

Likewise, the fourth statement $a \perp\!\!\!\perp b|ce$ is represented by $u_{\langle a,b|ce\rangle} = \delta_{abce} + \delta_{ce} - \delta_{ace} - \delta_{bce}$ and no term δ_Z with $Z \subseteq ce$ occurs in u_L. Thus, the fact that $m^{ce\downarrow} \equiv \sum_{Z,Z\subseteq ce} \delta_Z$ is a supermodular function allows us to show that $a \perp\!\!\!\perp b|ce$ is not independence implied by L.

Note that the fact that both $m^{abcde\uparrow}$ and $m^{ce\downarrow}$ are multiples of multiinformation functions for discrete distributions over N implies that neither $a \perp\!\!\!\perp b\,|\,cde$ nor $a \perp\!\!\!\perp b\,|\,ce$ is probabistically implied by L. We again omit details why it is so.

In fact, the preceding example can be generalized, as in the following lemma (see the Appendix for a proof).

Lemma 2.1. *Let* $\{v\} \cup UW \subseteq N$, $U \cap W = \emptyset$, $v \notin UW$, $|W| \geqslant 1$ *and* $\pi : W \mapsto W$ *be a permutation of* W. *Then*

$$\forall w \in W, \quad v \perp\!\!\!\perp w | UW \setminus \pi(w)w \iff \forall w \in W, \quad v \perp\!\!\!\perp \pi(w) | UW \setminus \pi(w)w.$$

Indeed, we put $v = a$, $U = \emptyset$, $W = \{b, c, d, e\}$ and consider the following permutation $\pi : b \to e \to d \to c \to b$ to get the above conclusion.

3. Algorithms

This section introduces algorithms for testing the implication $u_L \rightharpoonup u_t$. In Section 3.1, we revisit a method based on skeletal characterization of structural imsets from [17] and optimize the method. In Section 3.2, an algorithm for verification of $u_L \rightharpoonup u_t$ is presented based on searching a decomposition of $k \cdot u_L - u_t$ into elementary imsets. Section 3.3 concentrates on a method of disproving $u_L \rightharpoonup u_t$ by exploiting properties of supermodular functions. Section 3.4 combines the two previous methods by letting them race against each other and the one that returns its outcome first has a proof whether $u_L \rightharpoonup u_t$ or not.

3.1. Skeletal characterization of independence implication

We will only consider the implementation details here. Technical details and motivation of this approach can be found in Section 6.2.2 of [17]. This skeletal characterization is based on a particular set of imsets called the *ℓ-skeleton*, denoted as $\mathscr{K}_\ell^\diamondsuit(N)$. It follows from Lemma 6.2 in [17] that, for this particular set of imsets, we have $u_L \rightharpoonup u_t$ iff

$$\text{for all } m \in \mathscr{K}_\ell^\diamondsuit(N) \text{ if } \langle m, u_t \rangle > 0 \text{ then } \langle m, u_L \rangle > 0. \tag{3}$$

Recall that the inner product $\langle m, u \rangle$ of a function $m : \mathscr{P}(N) \to \mathbb{R}$ and an imset u is defined by $\sum_{S \subseteq N} m(S) \cdot u(S)$. Thus, to conclude $u_L \rightharpoonup u_t$, we just need to check the conditions in (3) for all imsets in the ℓ-skeleton.[2] It can be used to check which elementary imsets over five variables are implied in this sense by a user defining the input list.

The ℓ-skeleton for five variables consists of 117,978 imsets, which break into 1319 permutational types with each involving at most 120 imsets. Thus, checking whether $u_L \rightharpoonup u_t$ requires at most 117,978 operations [15]. However, if t is not implied by L, we might find out far earlier that (3) does not hold for a particular imset in $\mathscr{K}_\ell^\diamondsuit(N)$. By ordering skeletal imsets such that imsets that are more likely to cause violation in (3) are tried earlier, the required time can be minimized. These are the imsets $m \in \mathscr{K}_\ell^\diamondsuit(N)$ with many zeros in $\{\langle m, v \rangle;\ v \in \mathscr{E}\}$. Thus, sorting skeletal imsets on basis of this criterion helps to speed up the inference. The second auxiliary criterion is the number of sets $S \subseteq N$ with $u(S) = 0$.

Unfortunately, the skeletal characterization approach is hard to extend to more than five variables. First, because finding all elements of the ℓ-skeleton for more than five variables is computationally infeasible. Second, because it appears that the size of the

[2] An applet at http://www.utia.cas.cz/user_data/studeny/VerifyView.html uses this method.

ℓ-skeleton grows extremely fast with a growing number of variables. Therefore, we will consider different approaches to perform the inference in the rest of the paper.

3.2. Verification algorithm

If an imset u is a combination of elementary imsets $u = \sum_{v \in \mathscr{E}} k_v \cdot v$, $k_v \in \mathbb{Z}^+$ then we say that it is a *combinatorial imset*. This is a sufficient condition for an imset to be structural and it is an open question if it is also a necessary condition [17]. The method to verify $u_L \rightharpoonup u_t$ presented in this section is based on testing whether $u \equiv k \cdot u_L - u_t$ is a combinatorial imset for some $k \in \mathbb{N}$.

Testing whether u is combinatorial can be done recursively, by checking, for each $v \in \mathscr{E}$, whether $u - v$ is combinatorial. Obviously, this naive approach is computationally demanding and it requires some guidance and extra tests in order to reduce the search space.

There are a number of sanity checks we can apply, before starting the search. First of all, let t be $X \perp\!\!\!\perp Y | Z$, then $u_L \rightharpoonup u_t$ implies the existence of a set $W \subseteq XYZ$ with $u_L(W) > 0$. This can be shown by Proposition 4.4 from [17] where we use $m^{A\uparrow}$ with $A = XYZ$. Another sanity check is as follows. Whenever u is a structural imset and $S \subseteq N$ a maximal set with respect to inclusion satisfying $u(S) \neq 0$ then $u(S) > 0$. Likewise, $u(S) > 0$ for any minimal set satisfying $u(S) \neq 0$ – see Lemma 6.5 in [17].

To guide the search, for each elementary imset $v \in \mathscr{E}$, we define the *deviance* of v from a non-zero imset u as follows. Let maxcard(u) be the cardinality of the largest set $S \subseteq N$ for which $u(S) \neq 0$. It follows from the notes above that if u is structural then $u(S) \geqslant 0$ whenever $|S| = \text{maxcard}(u)$. Then, with $v = u_{\langle x,y|Z\rangle}$,

$$\text{dev}(v|u) = \begin{cases} \infty, & |xyZ| < \text{maxcard}(u) \text{ or } u(xyZ) \leqslant 0, \\ \sum\limits_{S \subseteq N} |v(S) - u(S)|, & \text{otherwise.} \end{cases}$$

Thus, the deviance of v from a combinatorial imset u is finite only if δ_{xyZ} has a positive coefficient in u and no set larger than $|xyZ|$ has a positive coefficient in u. We pick the elementary imset with the lowest deviance first. Observe that if u is a non-zero combinatorial imset then $v \in \mathscr{E}$ with finite dev($v|u$) exists.

The deviance is defined in such a way that the elementary imsets that cancel as many of the non-zero values in u as possible are tried before the imsets that cancel out fewer of the non-zero values. For example, let $u = u_{\langle a,bc|d\rangle} + u_{\langle a,b|d\rangle} = \delta_{abcd} + 2\delta_d - 2\delta_{ad} - \delta_{bcd} + \delta_{abd} - \delta_{bd}$ and $v_1 = u_{\langle a,c|bd\rangle} = \delta_{abcd} + \delta_{bd} - \delta_{abd} - \delta_{bcd}$ then $\text{dev}(v_1|u) = 8$ while $v_2 = u_{\langle c,d|ab\rangle} = \delta_{abcd} + \delta_{ab} - \delta_{abc} - \delta_{abd}$ has the deviance $\text{dev}(v_2|u) = 10$. Furthermore $v_3 = u_{\langle a,b|d\rangle}$ has infinite deviance since $|abd| = 3$ while maxcard(u) = 4. Finally, $v_4 = u_{\langle b,c|de\rangle}$ has infinite deviance as $u(bcde) = 0$. Therefore, v_1 will be tried before v_2, while v_3 and v_4 will not be tried at all in this round.

Thus, the deviance leads our search in a direction where we can hope to find a proper decomposition. Obviously, if t is not implied by L, the verification algorithm can spend a long time searching through the complete space of all possible partial decompositions.

3.3. Falsification algorithm

Falsification is based on supermodular imsets. By a *supermodular imset* we understand an imset which is a supermodular function.

Theorem 3.1. *An imset u is structural iff $\langle m,u\rangle \geqslant 0$ for any supermodular function m and $\sum_{S,S\supseteq T} u(S) = 0$ for any $T \subseteq N$ with $|T| \leqslant 1$.*

Proof. The necessity of the conditions is easy since they both hold for elementary imsets and can be extended to structural imsets. The sufficiency follows from Theorem 5.1 in [17] which claims that the same holds for a finite subset of the class of supermodular functions, namely the ℓ-skeleton $\mathscr{K}_\ell^\diamondsuit(N)$. □

So, we can exploit Theorem 3.1 to disprove $u_L \rightharpoonup u_t$ by constructing non-negative supermodular imsets randomly and taking their inner products with u_L and u_t. If we find a supermodular imset m such that $\langle m,u_L\rangle = 0$ and $\langle m,u_t\rangle > 0$ then we can observe $\langle m,k\cdot u_L - u_t\rangle < 0$ for any $k \in \mathbb{N}$ and conclude that $\neg(u_L \rightharpoonup u_t)$. A random supermodular imset m can be generated by first generating a 'base' imset m_{base} and then by modifying it to ensure the resulting imset is supermodular. We randomly select the size n of the base, then randomly select n different subsets $S_1,\ldots,S_n$ of N and assign $m_{base} = \sum_{S\in\{S_1,\ldots,S_n\}} k_S \cdot \delta_S$ where k_S are randomly selected integers in the range from 1 to $2^{|N|}$. Selecting larger values of the coefficients k_S would not make difference. On the other hand, they also would not help.

Now, m_{base} needs to be modified to ensure that the obtained function m is supermodular. We perform the following operation on m_{base}. Let $S_1,\ldots,S_{2^{|N|}}$ be an ordering of the subsets of N with $S_j \subseteq S_i \Rightarrow j \leqslant i$. For $i = 1,\ldots,2^{|N|}$ define $m(S_i)$ to be the maximum of $m_{\text{base}}(S_i)$ and $m(S_i\backslash x) + m(S_i\backslash y) - m(S_i\backslash xy)$ for all $x,y \in S_i$. This ensures that $\langle m,v\rangle \geqslant 0$ for all $v \in \mathscr{E}$ and we have constructed an imset m which is supermodular.

Note that this technique can be used to disprove $u_L \rightharpoonup u_t$ but it cannot be used to prove it. At best, an impression could be given about the chance that not $u_L \rightharpoonup u_t$. However, we have not explored this venue, but instead proceeded by combining this algorithm with the one in the previous section.

3.4. Racing algorithms for a proof

Typically, the verification algorithm from Section 3.2 can quickly find a decomposition of $k\cdot u_L - u_t$ into $\sum_{v\in\mathscr{E}} k_v \cdot v$, which proves that t is implied by L. Nevertheless, if $\neg(u_L \rightharpoonup u_t)$, the verification algorithm may spend a long time before it exhausts the whole space of possible decompositions of $k\cdot u_L - u_t$. However, the falsification algorithm from Section 3.3 can find a supermodular imset m with $\langle m,u_t\rangle > 0 = \langle m,u_L\rangle$, which proves u_t is not implied by u_L. On the other hand, it will not be able to prove that $u_L \rightharpoonup u_t$.

We can combine the two algorithms by starting two threads, one with the verification algorithm and one with the falsification algorithm. The one that finds a proof first, returns its outcome and stops the other thread. Fig. 1 illustrates the algorithm.

4. Experiments

We would like to judge the algorithms above on computational speed. However, it is hard to get a general impression of the performance of the algorithms, because it depends on a distribution of inference problems, which is unknown.

Still, we think we can get a representative impression of the relative performance of the algorithms by generating inference problems randomly and measuring the computation

```
Algorithm: Racing for inference with structural imsets
Input: Input list L, CI statement t
1: thread1 = new RaceThread(Verify(L, t, proof))
2: thread2 = new RaceThread(Falsify(L, t, proof), thread1)
4: thread1.start(); thread2.start()
5: thread1.join() // wait for thread1 to stop
// if thread2 finished first, it will stop thread1
6: thread2.stop()
return proof
```

Fig. 1. Racing algorithms.

speed. We generated inference problems over five variables so that we can compare the performance of the skeleton-based algorithm from Section 3.1 with the others. All experiments were performed on a PC with 2.6 GHz Celeron processor and 186 MB memory running Linux. A thousand input lists each were generated by randomly selecting 3, 4 up to 10 elementary CI statements, giving a total of 8000 input lists. The algorithms described in Section 3 were applied to this class of lists with each of the elementary CI statements that were not in the list. This gave 1000×77 inference problems for input lists with three statements, 1000×76 inference problems for input lists with four statements, etc. In total, this created $1000 \times ([80-3]+[80-4]+\cdots+[80-10]) = 588.000$ inference problems over five variables.

4.1. Results

Fig. 2 shows the total number of elementary CI statements that are implied (labeled by Accept) and not implied (labeled by Reject) grouped by the number of elementary CI statements (3, 4 up to 10) in the input list. Naturally, the number of implied statements increases with increased input list size. The total number decreases since the number of imsets that is not in the input list decreases with growing input lists.

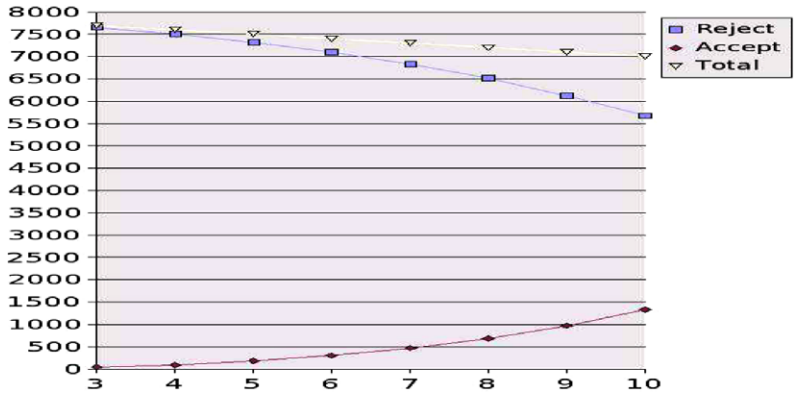

Fig. 2. Total number of rejects and accepts per experiment over five variables for various input list sizes. The size of the input list is shown on the x-axis. The number of rejects, accepts and total of unknown elementary statements is shown on the y-axis.

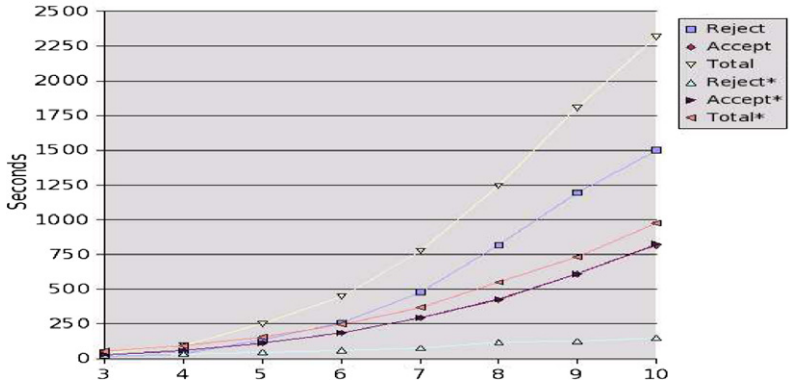

Fig. 3. Original skeleton-based testing compared with sorted skeleton-based testing. The x-axis shows the size of the input list, and the y-axis the time. Sequences marked with an asterisk are results for the sorted testing.

Fig. 3 shows the total run-times for running the experiments comparing skeleton-based testing with sorted skeleton-based testing. We distinguish between run-time for accepts, rejects and total because the run-time for accepts is not influenced by the order of skeletal imsets as all of them need to be inspected. Indeed, run-times for accepts hardly differed (run-times only slightly differ due to the fact that at random intervals garbage collection and other processes were performed). Run-times for rejects are reduced by about one order of magnitude so that total run-times are about halved. Thus, sorting the skeleton indeed helps significantly.

Fig. 4 shows the striking difference in reject times for the racing algorithms method from Section 3.4 and the skeleton-based method from Section 3.1, which clearly favors the new method. Only input lists of size 10 are shown, but the shapes for input lists of other size are the same.

The distribution of accept times shows a different picture, as illustrated in Fig. 5. The graph for skeleton-based method shows just one peak around 6 s per elementary CI statement, because that is how long it approximately takes to visit all skeletal imsets. The graph

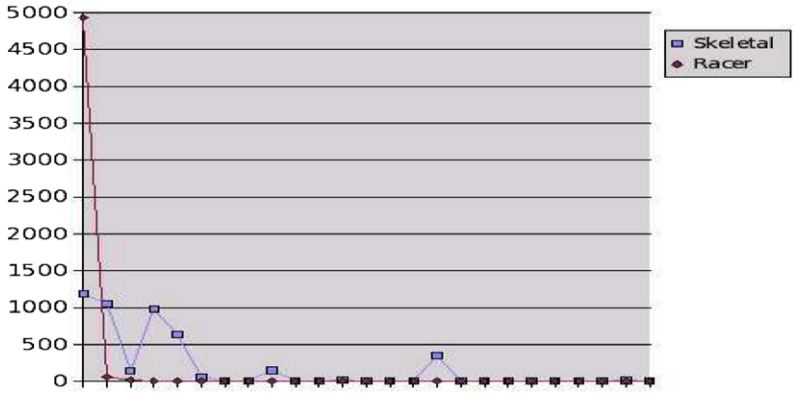

Fig. 4. Distribution of reject times of sorted skeleton-based method and racing algorithms method for input lists of size 10. The x-axis shows time, and the y-axis the number of elementary statements rejected in that time.

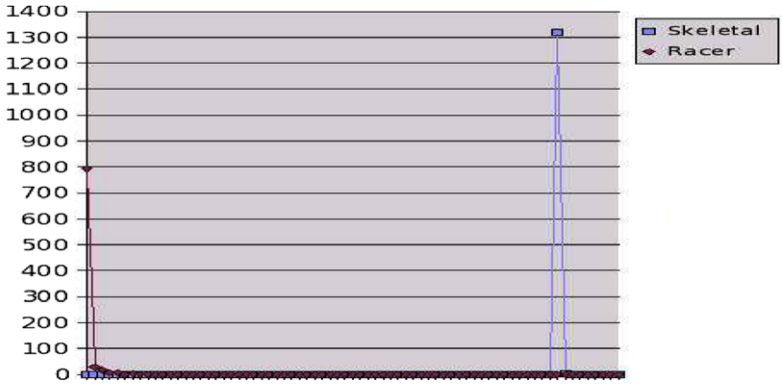

Fig. 5. Distribution of accept times of the sorted skeleton-based method and the racing algorithms method for input lists of size 10. The x-axis shows time, and the y-axis the number of elementary statements accepted in that time.

Table 1
Number of fails of the falsification algorithm with two different methods of generating random base imsets and various input list sizes (times $1000 \times k_{\max}$)

$\|L\|$	Rnd 1	Rnd 2					
	1	1	2	3	4	5	20
3	1	0	0	0	0	0	0
4	19	2	0	0	0	0	0
5	57	18	3	6	2	3	1
6	147	50	37	24	18	16	5
7	243	92	61	39	46	42	21
8	429	189	144	124	109	95	48
9	423	195	138	112	97	92	46
10	547	299	239	201	192	193	110

for the racing algorithms[3] shows a peak close to 10 ms, that drops off pretty quickly. Shapes for input lists of other size look very similar, though the tail gets thinner with decreasing size of input lists.

An alternative approach is to only run the falsification algorithm and run it long enough that the complete space of elementary statements is covered. Table 1 shows the number of fails of the pure falsification algorithm. These are those elementary CI statements that are not implied by the input list but the algorithm did not succeed to identify them within a fixed time limit.

Two methods of generating random 'base' imsets were compared. The first method draws weights from the interval 1–32 for randomly selected subsets, while the second always selects 1. The second method appears far more effective in identifying rejections as one can judge from the number of fails in the columns labeled 1 in Table 1. We also looked at the impact of the number of randomly selected supermodular imsets on the

[3] It is actually an enlargement of the graph for the verification algorithm since the falsification thread cannot return acceptance.

number of fails. Increasing this number decreases the failure rate, but the rate only drops very slowly. Even when generating the same number of supermodular functions as the number of skeletal imsets in the skeleton-based method, not all statements are correctly classified.

Fig. 6 shows run-times of the racing algorithms method compared with pure falsification algorithm (that is, without the verification part). While reject times are about a third on average for pure falsification, non-reject times are about four times larger than the accept times of the combined algorithm.

The same experiments as for five variables were performed with six variables, but obviously the skeleton-based algorithm was not applied on these problems. Apart from longer run-times of the algorithms, all observation as for five variables were confirmed.

4.2. Available software

An applet shown in Fig. 7, available at http://www.cs.waikato.ac.nz/~remco/ci/Vefify-View5.html, allows for efficiently posing queries for up to five variables. The user can enter a set of elementary CI statements by clicking the statements in the applet which then turn white. The elementary CI statements are organized by size such that elementary statement of the form $x \perp\!\!\!\perp y|Z$ can be found in the rectangle with other statements over $|xyZ|$ variables. Rows and columns are labeled with sets that can be decomposed as xZ and yZ respectively. The cross point of row xZ and column yZ contains statement $x \perp\!\!\!\perp y|Z$. The inference process is started by pressing the Go!-button and all statements not selected are verified and turn red if they are not implied or green if they are. Fig. 7 shows the inference from the example in Section 2.2.

Variants of the racing algorithm for three, four and six variables are available via http://www.cs.waikato.ac.nz/~remco/ci/index.html. A minor optimization is applied if a statement t is implied by a list L and we have a decomposition $\sum_{l\in L} k_l.u_l - u_t = \sum_{f\in F} u_f$ for a set of elementary statements F (and some positive integer constants k_l). This implies that all statements in F are implied by L as well, so they need not be verified individually but the applet can be updated directly. The skeleton algorithm and sorted skeleton for five

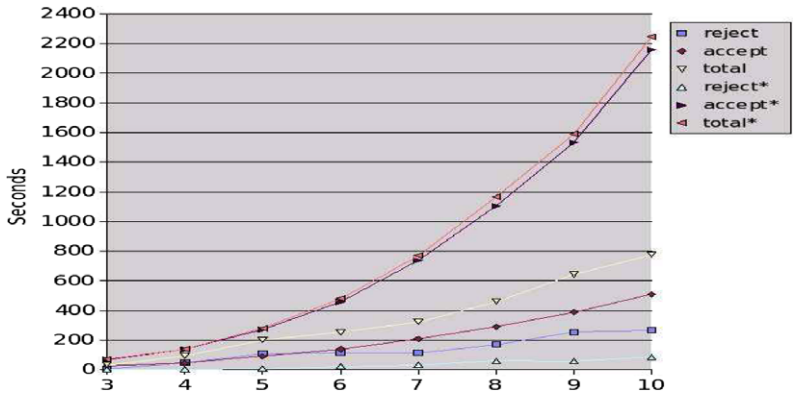

Fig. 6. Racing algorithms vs. pure falsification algorithm. The size of the input list is shown on the x-axis; the y-axis shows the time. Sequences marked with asterisk are results for the falsification.

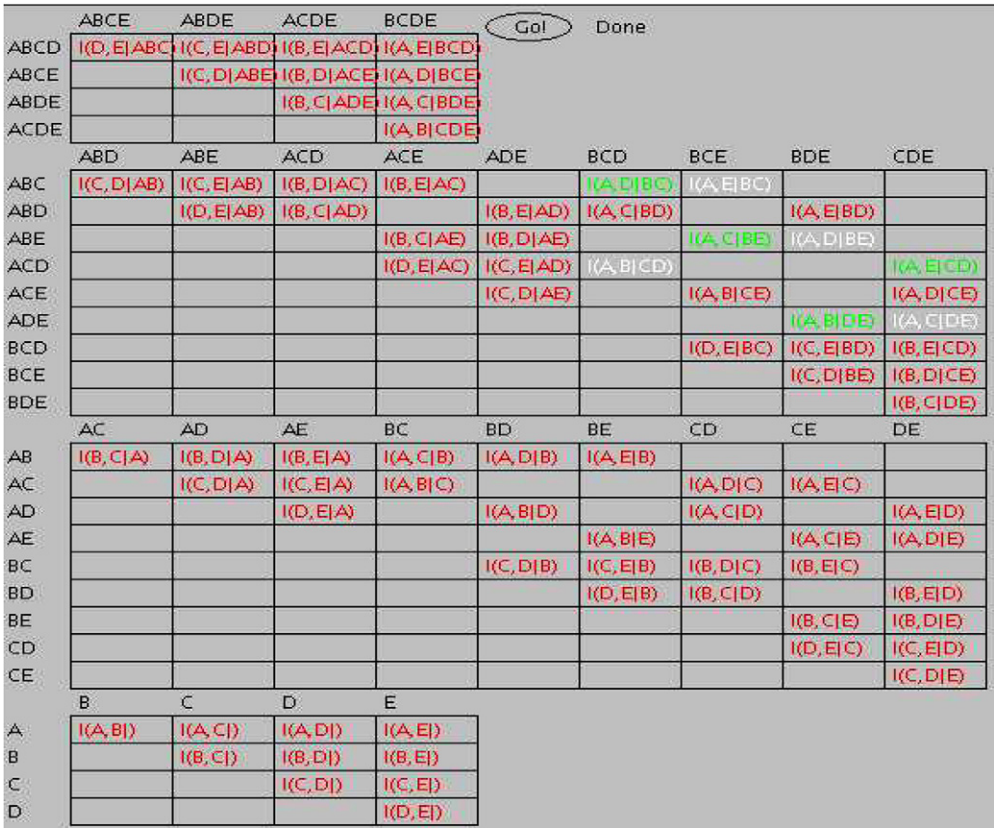

Fig. 7. An applet for racing inference over five variables.

variables and falsification algorithms for three up to six variables can also be accessed via that page.

5. Conclusions

We considered the computational aspects of performing CI inference using the method of structural imset, that is, deciding whether a CI statement t follows from an input list L of CI statements in that sense. The existing skeleton-based algorithm [15] that allows inference with up to five variables was improved. We presented an algorithm for creating a constructive proof that t follows from L. Unfortunately, this method does not perform well if t is not implied by L. Fortunately, we can prove t is not implied by L by randomly generating supermodular functions and testing whether the difference of inner products based on L and t is negative. But this method cannot be used to give a conclusive proof that t is implied by L. Together, these methods can race against each other on the same problem.

Empirical evidence suggests the mode of the run-time of the racing algorithms method is an order of magnitude less than the skeleton-based method. Furthermore, the new method also works well for problems with six variables, unlike the old one. Though we

have not verified this empirically yet, we expect our method to perform reasonably well with more than six variables, given that the majority of problems over five and six variables in our experiments were solved in just a few miliseconds. However, the number of statements in the input list needs to increase in order to have meaningful inferences, thereby increasing the complexity of the problem and increasing run-times. An analysis of accept times of the new method indicates that the verification algorithm sometimes cannot find the decomposition efficiently. This suggests that it can benefit from further guidance.

Some questions remain open, in particular finding an upper estimate on $k_{\max}$ (see Section 2.2) for six and more variables. A good upper estimate can decrease the computational effort in proving t is not implied by L. The similarity of this inference problem to other inference problems, which are known to be NP hard [7], suggests that the inference problem is NP hard. A formal proof of this property would provide further justification of using the heuristic approaches presented here.

Though the falsification algorithm cannot give a conclusive proof that an statement t is implied by L, we found that it was often very good at finding all elementary CI statements that are not implied by L in our experiments. This suggests that one can have some confidence that the falsification algorithm can indicate statements that are possibly implied by L. Deriving theoretical bounds on the probability that the falsification algorithm actually correctly identifies such statements would be interesting, since this would allow us to quantify our confidence.

Appendix. Proof of Lemma 2.1

We base our proof on the properties of the multiinformation function $m \equiv m_P$ corresponding to a (discrete) probability distribution P over N – see Section 2.3.4 in [17]. Alternatively, we can use the entropy function $\hat{h}_P$, defined by $\hat{h}_P(A) = H(P_A)$, $A \subseteq N$, where P_A is the marginal of P for A and H is the symbol for Shanonn entropy. This function has analogous properties – see Remark 4.4. in [17].

It follows from Corollary 2.2. in [17] that the assumption of Lemma 2.1 $v \perp\!\!\!\perp w | UW \setminus \pi(w) w$ for any $w \in W$ can be rewritten as the requirement $\langle m, u_{\langle v,w|UW\setminus\pi(w)w\rangle}\rangle = 0$ for any $w \in W$. More specifically, we have

$$\forall w \in W \quad 0 = m(vw[UW \setminus \pi(w)w]) + m(UW \setminus \pi(w)w) - m(v[UW \setminus \pi(w)w]) - m(w[UW \setminus \pi(w)w]).$$

We sum this over W and observe that the following sum vanishes

$$\sum_{w\in W}\{m(vUW \setminus \pi(w)) + m(UW \setminus \pi(w)w) - m(vUW \setminus \pi(w)w) - m(UW \setminus \pi(w))\}.$$

In this expression, we change the order of summation and get

$$0 = \sum_{w\in W} m(vUW \setminus \pi(w)) + \sum_{w\in W} m(UW \setminus \pi(w)w) - \sum_{w\in W} m(vUW \setminus \pi(w)w) - \sum_{w\in W} m(UW \setminus \pi(w)).$$

However, since π is a permutation one has $\sum_{w\in W} m(S \setminus \pi(w)) = \sum_{w\in W} m(S \setminus w)$ for any S with $W \subseteq S \subseteq N$, in particular, for $S = vUW$ and $S = UW$. Hence, we get

$$0 = \sum_{w\in W} m(vUW \setminus w) + \sum_{w\in W} m(UW \setminus \pi(w)w) - \sum_{w\in W} m(vUW \setminus \pi(w)w) - \sum_{w\in W} m(UW \setminus w).$$

We again change the order of summation and observe that the following sum vanishes

$$\sum_{w\in W} \{m(vUW \setminus w) + m(UW \setminus \pi(w)w) - m(vUW \setminus \pi(w)w) - m(UW \setminus w)\}.$$

This means $0 = \sum_{w\in W} \langle m, u_{\langle v,\pi(w)|UW\setminus\pi(w)w\rangle}\rangle$. As $\langle m, u_{\langle v,\pi(w)|UW\setminus\pi(w)w\rangle}\rangle \geqslant 0$ for any $w \in W$, by Corollary 2.2. in [17], the above equality means

$$\forall w \in W \langle m, u_{\langle v,\pi(w)|UW\setminus\pi(w)w\rangle}\rangle = 0,$$

and, again by Corollary 2.2. in [17], get $v \perp\!\!\!\perp \pi(w) | UW \setminus \pi(w)w$ for any $w \in W$.

References

[1] S.A. Anderson, D. Madigan, M.D. Perlman, Alternative Markov properties for chain graphs, Scandinavian Journal of Statistics 28 (1) (2001) 33–85.

[2] R.R. Bouckaert, IDAGs: a perfect map for any distribution, in: M. Clarke, R. Kruse, S. Moral (Eds.), Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Lecture Notes in Computer Science, vol. 747, Springer-Verlag, 1993, pp. 49–56.

[3] R.R. Bouckaert, M. Studený, Chain graphs: semantics and expressiveness, in: C. Froidevaux, J. Kohlas (Eds.), Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Lecture Notes in AI, vol. 946, Springer-Verlag, 1995, pp. 67–76.

[4] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, D.J. Spiegelhalter, Probabilistic Networks and Expert Systems, Springer-Verlag, New York, 1999.

[5] D.R. Cox, N. Wermuth, Multivariate Dependencies – Models, Analysis and Interpretation, Chapman & Hall, 1996.

[6] A.P. Dawid, Conditional independence in statistical theory, Journal of the Royal Statistical Society B 41 (1979) 1–31.

[7] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to The Theory of NP-completeness, W.H. Freeman and Company, San Francisco, 1979.

[8] D. Geiger, J. Pearl, Logical and algorithmic properties of conditional independence and graphical models, Annals of Statistics 21 (4) (1993) 2001–2021.

[9] G. Kauermann, On a dualization of graphical Gaussian models, Scandinavian Journal of Statistics 23 (1) (1996) 105–116.

[10] J.T.A. Koster, Markov properties of nonrecursive causal models, Annals of Statistics 24 (5) (1996) 2148–2177.

[11] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufman, San Mateo, 1988.

[12] P.P. Shenoy, Conditional independence in valuation-based systems, International Journal of Approximate Reasoning 10 (3) (1994) 203–234.

[13] P. Spirtes, C. Glymour, R. Scheines, Causation, prediction and search, Lecture Notes in Statistics, vol. 81, Springer-Verlag, 1993.

[14] M. Studený, Conditional independence relations have no finite complete characterization, in: S. Kubík, J.Á. Víšek (Eds.), Information Theory, Statistical Decision Functions and Random Processes, vol. B, Kluwer, Dordrecht, 1992, pp. 377–396.

[15] M. Studený, R.R. Bouckaert, T. Kočka, Extreme supermodular set functions over five variables, Research Report No. 1977, Institute of Information Theory and Automation, Prague, January 2000.

[16] M. Studený, Structural imsets: an algebraic method for describing conditional independence structures, in: B. Bouchon-Meunier, G. Coletti, R.R. Yager (Eds.), Proceedings of IPMU 2004, pp. 1323–1330.

[17] M. Studený, Probabilistic Conditional Independence Structures, Springer-Verlag, London, 2005.

[18] T. Verma, J. Pearl, Causal networks: semantics and expressiveness, in: R.D. Shachter, T.S. Lewitt, L.N. Kanal, J.F. Lemmer (Eds.), Uncertainty in Artificial Intelligence, vol. 4, North-Holland, Amsterdam, 1990, pp. 69–76.
[19] N. Wermuth, S.L. Lauritzen, On substantive research hypotheses, conditional independence graphs and graphical chain models (with discussion), Journal of the Royal Statistical Society B 52 (1990) 21–72.