Artificial Intelligence 58 (1992) 237–270
Elsevier

ARTINT 954

Structure identification in relational data *

Rina Dechter

Information and Computer Science, University of California, Irvine, CA 92717, USA

Judea Pearl

Cognitive Systems Laboratory, Computer Science Department, University of California, Los Angeles, CA 90024, USA

Abstract

Dechter, R. and J. Pearl, Structure identification in relational data, Artificial Intelligence 58 (1992) 237–270.

This paper presents several investigations into the prospects for identifying meaningful structures in empirical data, namely, structures permitting effective organization of the data to meet requirements of future queries. We propose a general framework whereby the notion of identifiability is given a precise formal definition similar to that of learnability. Using this framework, we then explore if a tractable procedure exists for deciding whether a given relation is decomposable into a constraint network or a CNF theory with desirable topology and, if the answer is positive, identifying the desired decomposition. Finally, we address the problem of expressing a given relation as a Horn theory and, if this is impossible, finding the best k-Horn approximation to the given relation. We show that both problems can be solved in time polynomial in the length of the data.

1. Introduction

Discovering meaningful structures in empirical data has long been regarded as the hallmark of scientific activity. Yet, despite the mystical aura surrounding such discoveries, we often find that computational considerations of efficiency and economy play a major role in determining what

Correspondence to: R. Dechter, Information and Computer Science, University of California, Irvine, CA 92717, USA. E-mail: dechter@ics.uci.edu.

*This work was supported in part by the Air Force Office of Scientific Research grant AFOSR 900136, NSF grant IRI-9157636, GE Corporate R&D and Micro grant 91-125.

structures are considered meaningful by scientists. In this paper we address the task of finding a computationally attractive description of the data, a description that both is economical in storage and permits future queries to be answered in a tractable way.

Invariably, the existence of such a desirable description rests on whether the dependencies among the data items are decomposable into local, more basic dependencies, possessing some desirable features. A classic example would be finding a finite state machine (with the least number of states) that accounts for observed dependencies among successive symbols in a very long string. In more elaborate settings, the dependencies can form a graph (as in the analysis of Markov fields [24]) or a hypergraph (as in relational databases [19]), and the task is to find the topology of these structures. Structure identification includes tasks such as finding effective representations for probability distributions [7,16,30], devising economical decompositions of database schemas, synthesizing simple Boolean expressions for truth tables [6], and casting logical theories that render subsequent processing tractable.

Despite the generality of the task, very few formal results have been established, and those that exist were confined primarily to probabilistic analysis of statistical data [7,18,24,25]. In this paper we focus on relational (non-statistical) data and deterministic descriptions of the data. Given a relation ρ in the form of an explicit listing of the tuples of ρ, we ask whether we can find a more desirable description of ρ, say a constraint network possessing desirable topological features or a logical theory possessing desirable syntactic features (e.g., Horn theories). In both cases the desirable features would be those that facilitate efficient query processing routines.

We view this task as an exercise in automatic *identification*, because our main concerns are to recognize cases for which desirable descriptions exist and to identify the parameters of at least one such description. Thus, we explore the existence of a tractable identification procedure that takes data as input, returns a structure, and works in time polynomial in the size of the input and output. Given that the data were generated from a model that has a desirable structure, our procedure should identify either the underlying structure when it is unique, or an equivalent structure when it is not unique. Conversely, if the data does not lend itself to effective organization, we wish our procedure to acknowledge this fact, so as to save further exploration. Sometimes an additional requirement is imposed on the procedure, namely, to identify a "best" approximated theory, when an exact desirable theory does not exist. We call this latter requirement "strong identifiability".

Our analysis bears a close relation to that of Selman and Kautz [27], where theory formation is treated as a task of "knowledge compilation". The main difference between the two approaches is that Selman and Kautz

begin with a preformed *theory* in the form of a (reasonably sized) set of clauses, while we start with the bare *observations*, namely, a (reasonably sized) set of tuples that represent the models of the desired theory. This enables us to easily project the tuples onto subsets of variables and to solve subtasks that would be intractable had we started with a causal theory. Another difference is that we require definite determination of whether the theory approximates or describes the data.

This paper is organized as follows: Section 2 introduces a general framework of the identification task. We define weak and strong notions of identifiability and, using familiar examples, compare them to Valiant's [30] notion of learnability. Section 3 investigates the identifiability of structure-based tractable classes of constraint networks and propositional theories. We show that stars and trees are identifiable, while chains and k-trees are not. Section 4 focuses on identifying theories whose tractability stems from syntactic rather than structural features. In particular, we show that relations describable by Horn theories can readily be recognized (Theorem 4.9), and that corresponding Horn theories can be found in time polynomial in the length of the data (Theorem 4.10). Additionally, we show that a best approximation in k-Horn theory (in which every clause contains at most k literals) can be identified in $O(|\rho| n^{k+1})$ time, where n is the number of variables (Corollary 4.11).

2. Preliminaries and basic definitions

2.1. Theories: networks and formulas

We denote propositional symbols, also called *variables*, by uppercase letters $P, Q, R, X, Y, Z, \ldots$, propositional literals (i.e., $P, \neg P$) by lowercase letters $p, q, r, x, y, z, \ldots$, and disjunctions of literals, or *clauses*, by $\alpha, \beta, \ldots$. The complement operator $\sim$ over literals is defined as usual: If $p = \neg Q$, then $\sim p = Q$; if $p = Q$ then $\sim p = \neg Q$. A *formula* in conjunctive normal form (CNF) is a set of clauses $\varphi = \{\alpha_1, \ldots, \alpha_t\}$, implying their conjunction. The *models* of a formula φ, $M(\varphi)$, is the set of all satisfying truth assignments to all of the formula's symbols. A clause α is *entailed* by φ, written $\varphi \models \alpha$, iff α is true in all models of φ. A clause α is a *prime implicate* of φ iff $\varphi \models \alpha$ and $\nexists \beta \subseteq \alpha$ such that $\varphi \models \beta$. A Horn formula is a CNF formula whose clauses all have at most one positive literal. A k-CNF formula is one in which clauses are all of length k or less, and a k-Horn formula is defined accordingly.

To characterize the structure of a formula φ we define its *scheme* as the set of variable sets on which clauses are defined. Formally:

Definition 2.1 (*Scheme*). Let $\varphi = \{\alpha_1, \ldots, \alpha_r\}$, and let $base(\alpha)$ be the set of all propositional symbols on which clause α is defined, then

$$scheme(\varphi) = \{base(\alpha_j) \mid 1 \leqslant j \leqslant r\}. \tag{1}$$

Example 2.2. Consider the formula

$$\varphi = \{(\neg P \vee Q \vee R), (P \vee S), (\neg P \vee \neg S), (\neg P \vee R)\}. \tag{2}$$

In this case,

$$scheme(\varphi) = \{\{P, Q, R\}, \{P, S\}, \{P, R\}\}. \tag{3}$$

We next define the notions of *constraint networks* and *relations*, which parallel the notions of formulas and their satisfying models, for multivalued variables. A *relation* associates a set of multivalued variables with a set of tuples specifying their allowed combinations of values. A *constraint network* is a set of such relations, each defined on a subset of the variables. Taken together, this set represents conjunction of constraints, namely, it restricts value assignments to comply with each and every constituent relation. The theory of relations has been studied extensively in the database literature [19].

Definition 2.3 (*Relation and network*). Given a set of multivalued variables $X = \{X_1, \ldots, X_n\}$, each associated with a domain of discrete values $D_1, \ldots, D_n$, respectively, a *relation* (or, alternatively, a *constraint*) $\rho = \rho(X_1, \ldots, X_n)$ is any subset

$$\rho \subseteq D_1 \times D_2 \times \cdots \times D_n. \tag{4}$$

A *constraint network* N over X is a set $\rho_1, \ldots, \rho_t$ of such relations. Each relation ρ_i is defined on a subset of variables $S_i \subseteq X$. The set of subsets $S = \{S_1, .., S_t\}$ is called the *scheme* of N (also denoted $scheme(N)$). The *dimension* of scheme S, denoted $dim(S)$, is defined as the cardinality of the largest component in S. The network N represents a unique relation $rel(N)$ defined over X, which stands for all consistent assignments (or all solutions), namely,

$$rel(N) = \{x = (x_1, \ldots, x_n) \mid \forall S_i \in S, \Pi_{S_i}(x) \in \rho_i\}, \tag{5}$$

where $\Pi_{S_i}(x)$ is the projection of x onto S_i. The *projection* of a relation ρ onto a subset of variables R, denoted $\Pi_R(\rho)$, is the set of tuples defined on the variables in R that can be extended to a tuple in ρ. If $rel(N) = \rho$ we say that N *describes* or *represents* ρ.

Clearly, any CNF formula can be viewed as a special kind of constraint network, where the domains are bivalued and where each clause specifies a constraint on its propositional symbols. We say that a bivalued relation $\rho = \rho(x_1, \ldots, x_n)$ is *described* (or *represented*) by a formula $\varphi = \varphi(x_1, \ldots, x_n)$ iff $M(\varphi) = \rho$. We will use the term *theory* to denote either a network or a formula and, correspondingly, will use $M(T)$ and $rel(T)$ interchangeably.

Example 2.4. The relation

$$\begin{aligned}\rho(P,Q,R,S) = \{&(1010),\\ &(1110),\\ &(0101),\\ &(0011),\\ &(0111),\\ &(0001)\}\end{aligned}$$

can be described by the network

$$\begin{aligned}N = \{&(\rho(P,Q,R) = \{(101),(111),(010),(001),(011),(000)\}),\\ &(\rho(P,S) = \{(01),(10)\}),\\ &(\rho(P,R) = \{(00),(01),(11)\})\},\end{aligned}$$

since the consistent assignments of N coincide with ρ. Being bivalued, ρ can also be described by the formula

$$\varphi = \{(\neg P \vee Q \vee R), (P \vee S), (\neg P \vee \neg S), (\neg P \vee R)\}, \tag{6}$$

since $M(\varphi) = \rho$.

When considering ways of approximating a relation ρ by a theory T, we will examine primarily upper bound approximations, namely, theories T such that $\rho \subseteq M(T)$.

Definition 2.5 (*Tightest approximation*). A theory $T \in C$ is said to be a *tightest approximation* of ρ relative to a class C of theories if $\rho \subseteq M(T)$ and there is no $T' \in C$ such that $\rho \subseteq M(T') \subset M(T)$.

2.2. Identifiability

We are now ready to give a formal definition of identifiability: A property that is intrinsic to any class of theories and that governs our ability to decide whether a given relation ρ has a description within the class. As the preliminary example, we will use the class of k-CNF formulas. We will show that, while this class is identifiable for $k = 2$, it may not be identifiable for

any $k > 2$. In other words, there may not be any tractable way of deciding whether an arbitrary relation ρ has a description as a k-CNF formula for $k > 2$. The class of 2-CNF theories will turn out to be *strongly* identifiable, namely, not only can we decide the existence of a 2-CNF description, and produce such a description if it exists, but we can also produce a tightest 2-CNF formula (as well as a k-CNF, $k > 2$) if a precise description does not exist (hence the term "strong").

Not surprisingly, the decisions above would depend on our prior knowledge about the observed relation ρ. For example, if we were given assurance that ρ has a description in k-CNF, it would be easy to produce such a description. Thus, it is necessary to define the notion of identifiability relative to a background class C' of theories from which ρ is chosen. We will adopt the convention that, unless stated otherwise, C' is presumed to be the class of all theories, namely, ρ is an arbitrary relation. We will denote by $|\rho|$ the number of tuples in ρ.

Definition 2.6 (*Identifiability*). A class of theories C is said to be *identifiable* relative to a background class C' if there is an algorithm A, polynomial in the size of its input and output, such that:

(1) *Recognition*: For every relation ρ that is describable by some theory T in C', A determines whether ρ has a description in C.

(2) *Description*: If the answer to (1) is positive, algorithm A finds one theory $T \in C$ that describes ρ (i.e., $\rho = M(T)$).

(3) *Tightness*: C is said to be *strongly* identifiable if, in addition to (1) and (2) above, A always finds a theory T_0 in C that is a tightest approximation of ρ.

By convention, a class in which the problems associated with the recognition or description tasks are NP-hard will be defined as nonidentifiable. Note that in conditions (2) and (3) the complexity of A is measured relative to the size of ρ as well as to the size of the theories that describe ρ. In the analysis of structure-based constraint networks and CNFs (Section 3), the description length will usually be insignificant, so $|\rho|$ will be the dominant factor in the complexity of A. In Section 4, however, where we analyze Horn approximations, the two factors will play equal roles. Practically speaking, taking the size of ρ as the basis for measuring complexity amounts to focusing the identification task on highly constrained data, where the number of distinct observations grows polynomially with the number of variables.

Coming back to our k-CNF example, we consider again the question of whether the class C_k of all relations expressible by k-CNF theories is identifiable relative to $C' = C_n$, the class of all CNFs. As we will show in Section 3, although algorithms for condition (3) (and hence condition (2)), namely, of constructing a tightest k-CNF approximation, do exist for

any ρ, we do not have an effective way of fulfilling condition (1), namely, of testing whether this approximation represents the relation ρ exactly or a superset thereof. (Even generating a single model of a tightest k-CNF theory may be NP-hard for $k > 2$.) We will thus conclude that C_k is not identifiable.[1] On the other hand, C_k is strongly identifiable relative to itself, since recognition is trivially satisfied and any tightest approximation must be exact.

2.3. Identifiability versus learnability

There is a strong resemblance between the notion of identifiability and of learnability [30]. If we associate theories with concepts (or functions) and the models of a theory with the learning examples, we see that in both cases we are seeking a polynomial algorithm that will take in a polynomial number of examples and will produce a concept (or a function) which is consistent with those examples, from some family of concepts C. Moreover, for a family C to be learnable from positive examples (with one-sided errors), we know that it must be closed under intersection and that the algorithm must produce the tightest concept in C consistent with the observations [23]. This is identical to condition (3) (strong identifiability).

The main difference between the problems described in this paper and those addressed by learning models is that in learnability we are given the concept class C and our task is to identify an individual member of C that is (either surely or probably) responsible for the observed instances. By contrast, in structure identification we are not given the concept class C. Rather, one of our primary objectives is to decide whether a fully observed concept ρ, taken from some broad class C' (e.g., all relations), is also a member of a narrower class C of concepts, one that possesses desirable syntactical features (e.g., a 2-CNF, a constraint tree, or a Horn theory). Thus, the task is not to infer the extension of a concept from a subset of its examples (the entire extension is assumed to be directly observed), but to decide whether the concept admits a given syntactical description.

It turns out that deciding whether the tightest approximation exactly describes a given concept, even when the concept is of small size, might be computationally expensive—a problem not normally addressed in the learning literature.

The differences between learnability and identifiability can be well demonstrated using our previous example of the class C_k of k-CNF theories. We have argued earlier that while C_k may not be identifiable relative to the class C' of all relations, it is nevertheless strongly identifiable relative to

[1] The nonexistence of a tractable procedure for testing exact match with ρ is subject to Conjecture 3.10 (see Section 3.1).

$C' = C_k$. By comparison the class C_k is known to be polynomially learnable [30] since, given a collection of instances I of $M(\varphi)$, one can find in polynomial time the tightest k-CNF expression that contains I (see Section 3). The fact that C_k is not identifiable is not too disturbing in learning tasks, because there we assume that the examples must be drawn from some k-CNF theory φ, so in the long run the tightest k-CNF approximation to the data will coincide with φ. However, nonidentifiability can be very disturbing if the examples are taken from a theory outside C_k. In this case, the tightest k-CNF theory consistent with the examples might lead to substantial (one-sided) errors.

In general, if we set $C' = C$, then, if C is learnable from positive and negative examples, it must also be identifiable, because identification is an easier task under this condition. Assuming that ρ contains *all* instances of a concept amounts to observing both positive and negative examples and, compared to PAC-learning, we are effectively provided with an answer to every membership query without having to wait for examples to be generated by a random process. Likewise, if C is one-sided learnable, it must be strongly identifiable, because condition (1) is satisfied automatically, and the one-sided learnability requirement of zero error on negative examples is equivalent to condition (3). There are, of course, concept classes that are identifiable but not one-sided learnable under the condition $C' = C$ (a trivial example of which is the class of relations ρ having size $|\rho| = k$), again because, in identification, negative examples are implicit through their absence from the data.

3. Topology-based identification

A given relation ρ can be represented by many networks or formulas (if ρ is bivalued), each having a different scheme. All such representations will be called *equivalent* (denoted by $\approx$). In this section we will focus on networks and CNF formulas parameterized by their corresponding schemes. We will first analyze the identifiability of several classes of constraint networks and then show, using a simple translation, that most results are extensible to the scheme-based identifiability of CNF formulas.

3.1. Identifying constraint networks

We will first make some observations that generalize Montanari's [21] notion of *minimal networks* (originally defined on binary networks) to constraint networks having arbitrary schemes. Some of these observations were also made in [22], albeit under a different notation.

We denote by N_S the class of all constraint networks having a common scheme S, assuming all networks are defined on n variables with the same set of domain values.

Definition 3.1 (*Projection network*). Given a relation ρ and a scheme $S = \{S_1, \ldots, S_r\}$, the projection of ρ on S, $\Pi_S(\rho)$, is defined as the network obtained by projecting ρ onto each component of S:

$$\Pi_S(\rho) = \{\Pi_{S_i}(\rho) | \ \forall S_i \in S\}. \tag{7}$$

Clearly, generating $\Pi_S(\rho)$ from ρ is polynomial:

Lemma 3.2. *The network $\Pi_S(\rho)$ can be constructed in $O(|\rho||S|)$ steps.*

Theorem 3.3 (Montanari and Rossi [22]). *The network $\Pi_S(\rho)$ is a tightest approximation of ρ relative to the class N_S.*

As in [21,22], we next observe that among all networks $R \in N_S$ that are equivalent to $\Pi_S(\rho)$, $\Pi_S(\rho)$ has a unique syntactic property called *minimality* with respect to the partial order $\subseteq$ defined as follows:

Definition 3.4 (*Network ordering*).

$$R \subseteq Q \quad \text{iff} \quad \forall S_i \in S, R_{S_i} \subseteq Q_{S_i}, \tag{8}$$

where R_{S_i} is the relation associated with S_i in network R.

Similarly, we define the *intersection of networks* R and Q in N_S as the network created by the intersection of the corresponding constraints.

Definition 3.5 (*Network intersection*). Let $R, Q \in N_S$. The intersection $\cap$ of R and Q is given by:

$$R \cap Q = \{R_{S_i} \cap Q_{S_i} | \ \forall S_i \in S\}. \tag{9}$$

Note that $R \in N_S$ and $Q \in N_S$ implies that $R \cap Q \in N_S$.

The next theorem states the existence of a unique (with respect to $\subseteq$) *minimal* network M_S of ρ having scheme S.

Theorem 3.6 (Montanari and Rossi [22]). . *Let $R, Q \in N_S$ and let $\rho = rel(R)$, then*

(1) $R \approx Q \implies R \cap Q \approx R$,

(2) *there exists a unique minimal (with respect to $\subseteq$) network M_S representing ρ, and it is given by $M_S = \Pi_S(\rho)$.*

Proof. The first part is clear and implies the existence of a unique minimal network M_S, which equals the intersection of all equivalent networks. We will show that $M_S = \Pi_S(\rho)$. Clearly, $M_S \approx \Pi_S(\rho)$. By definition, $M_S \subseteq \Pi_S(\rho)$. However, if we eliminate even a single tuple from any constraint in $\Pi_S(\rho)$, the resulting network will not allow a legal tuple of ρ, contradicting the fact that M_S describes ρ. Consequently, $M_S = \Pi_S(\rho)$. □

Corollary 3.7. *Among all tightest approximations to ρ from N_S, $\Pi_S(\rho)$ is the minimal one.*

We are now ready to discuss identifiability relative to specific schemes. The negative result that follows hinges on Conjecture 3.10 and Lemma 3.8.

Lemma 3.8 (Ullman[29]). *Given a relation ρ and an arbitrary scheme S, deciding whether $rel(\Pi_S(\rho)) = \rho$ is NP-hard.*

Theorem 3.9. *Given an arbitrary scheme S, the class N_S is not identifiable relative to all networks N, but is strongly identifiable relative to N_S.*

Proof. We will show a polynomial reduction from the decision problem of whether $rel(\Pi_S(\rho)) = \rho)$ (which is NP-complete) to the problem of deciding the identifiability of N_S (for any S). Given a relation ρ and a scheme S, if we can identify in polynomial time whether ρ is representable by scheme S, then, due to condition (1), we can also determine whether $rel(\Pi_S(\rho)) = \rho)$: If ρ is identifiable, $rel(\Pi_S(\rho)) = \rho$; if not, $rel(\Pi_S(\rho)) \supset \rho$. Since the projection formula can be computed in time proportional to $|\rho||S|$, the reduction is polynomial. Consequently, based on Lemma 3.8, unless P = NP, the identifiability of N_S relative to all networks is NP-hard. When the background is N_S, the exactness decision is trivially satisfied (the projection $\Pi_S(\rho)$ is guaranteed then to represent ρ) and hence N_S is strongly identifiable. □

Let S_k be the set of all subsets of size k or less of $X = \{X_1, \ldots, X_n\}$. Although we believe that the NP-completeness result of Lemma 3.8 extends to these special schemes, we are unable to prove it as yet. For convenience we will use the shorthand $N_k = N_{S_k}$.

Conjecture 3.10. *Given a relation ρ and an integer k, deciding whether $rel(\Pi_{S_k}(\rho)) = \rho$ is NP-hard.*

Corollary 3.11. *The class N_k is not identifiable relative to N.*

Example 3.12. The class N_2 of all binary multivalued constraint networks is not identifiable (unless the number of values is 2). This class is characterized by a scheme consisting of all variable pairs (i.e., the complete graph) for which, subject to Conjecture 3.10, we cannot establish whether $rel(\Pi_{S_2}(\rho)) = \rho$ in polynomial time.

When the scheme S has topological properties that permit solution in polynomial time, N_S is identifiable. We say that a scheme S is tractable when there exists a polynomial algorithm for deciding the consistency of every constraint network in the class N_S. For instance, any tree or acyclic hypergraph [10] is a tractable scheme.

Theorem 3.13. *Let S be a tractable scheme. Then the class of networks N_S is strongly identifiable.*

Proof. The projection $\Pi_S(\rho)$ provides a tightest N_S approximation to ρ and can be computed in polynomial time. The tractability of S assures that the equality $|rel(\Pi_S(\rho))| = |\rho|$ can also be tested in polynomial time. It was recently shown [11] that for every theory T satisfiable in time t, deciding whether $|M(T)| > c$ takes time $O(ct)$. Now, since S is tractable, it is satisfiable in polynomial time; therefore, deciding $rel(\Pi_S(\rho)) \supset \rho$ can be accomplished in polynomial time. □

3.2. Identifying CNF formulas

In this section we shift our attention to bivalued relations and to the task of identifying tractable classes of CNF formulas. We will show that there is great similarity between the identifiability of scheme-based constraint networks and that of scheme-based CNF having the same scheme. This will become clear through a simple translation from bivalued relations into CNFs. As in the multivalued case, we will first extend the auxiliary notions of *projection network* and *minimal network* to *projection formula* and *maximal formula.*

Definition 3.14 (*Canonical representation*).

(1) Let ρ be a bivalued relation over $X = X_1, \ldots, X_n$. We define

$$\begin{aligned} &canonical(\rho) \\ &\quad = \{(\sim x_1 \vee \sim x_2 \vee \cdots \vee \sim x_n) | (x_1, x_2, \ldots, x_n) \notin \rho\}. \end{aligned} \tag{10}$$

(2) Given a constraint network $N = \{\rho_1, \ldots, \rho_t\}$, we define $canonical(N)$ as the formula generated by collecting the canonical formulas of every

constituent relation in N. Namely,

$$canonical(N) = \bigcup\{canonical(\rho_i) | \rho_i \in N\}. \tag{11}$$

The equivalence of the relational and propositional representations of this translation, as stated in the following lemma, is quite immediate:

Lemma 3.15. *The models of the canonical formula of network N coincide with the solutions of N. Namely, $M(canonical(N)) = rel(N)$.*

We now extend the notion of projection network to projection formula.

Definition 3.16 (*Projection formula*). Given a relation ρ and a scheme S, the *projection formula* of ρ with respect to S, denoted $\Gamma_S(\rho)$, is given by

$$\Gamma_S(\rho) = canonical(\Pi_S(\rho)). \tag{12}$$

Example 3.17. Let

$$\rho(P, Q, R) = \{(100), (010), (001)\}$$

and

$$S = \{\{P, Q\}\{P, R\}\{Q, R\}\}.$$

Then,

$$\begin{aligned}\Pi_S(\rho) = \{&(\rho(P, Q) = \{(10), (01), (00)\}),\\ &(\rho(P, R) = \{(10), (01), (00)\}),\\ &(\rho(Q, R) = \{(10), (01), (00)\})\},\end{aligned}$$

and

$$\Gamma_S(\rho) = \{(\neg P \vee \neg Q), (\neg P \vee \neg R), (\neg R \vee \neg Q)\}.$$

Generating $\Gamma_S(\rho)$ from $\Pi_S(\rho)$ may be exponential in the size of the largest component in S, as it requires enumerating all tuples defined on each component. Consequently, for schemes of bounded dimension, this construction is polynomial.

Lemma 3.18. *Let $dim(S) \leqslant k$. The complexity of generating $\Gamma_S(\rho)$ is* $O(|\rho||S| + |S|2^k)$.

Let C_S be the class of CNF formulas having scheme S. Parallel to Theorem 3.3, we have:

Theorem 3.19. *The formula $\Gamma_S(\rho)$ is a tightest approximation of ρ relative to C_S.*

Proof. Follows immediately from the facts that $\Pi_S(\rho)$ is a tightest approximation to ρ and $M(\Gamma_S(\rho)) = rel(\Pi_S(\rho))$. □

Parallel to the notion of minimal networks in multivalued relations, we will now show that among all formulas φ in C_S that are equivalent to $\Gamma_S(\rho)$, $\Gamma_S(\rho)$ is *maximal* with respect to the partial order $\subseteq$ defined by set inclusion (of clauses). Clearly the class C_S is closed under union. The next lemma (parallel to Theorem 3.6) proves that among all equivalent formulas in C_S, $\Gamma_S(\rho)$ is the unique maximal formula.

Lemma 3.20. *Let $\varphi, \tau \in C_S$ and let $\rho = M(\varphi)$. Then*

(1) $\varphi \approx \tau \implies \varphi \cup \tau \approx \varphi$,

(2) *there exists a unique maximal (with respect to $\subseteq$) formula μ_S representing ρ, and it is given by $\mu_S = \Gamma_S(\rho)$.*

Proof. The first part is immediate, since the union of formulas stands for their conjunction. The second part is similar to the proof of Theorem 3.6. □

When the scheme S has components containing each other, the corresponding clauses may subsume each other and thus present redundancy. We clearly prefer to consider formulas in *reduced* form, without subsumption.

Definition 3.21 (*Reduced formula*). A formula φ is *reduced* if none of its clauses contain another clause. The formula obtained after eliminating clause subsumption from φ is denoted *reduced*(φ).

Lemma 3.22. *Reduced($\Gamma_S(\rho)$) contains all (but not only) the prime implicates of $\Gamma_S(\rho)$ that are restricted to the subsets in S.*

Proof. Being maximal, $\Gamma_S(\rho)$ must contain all prime implicates restricted to S. □

We now state a scheme-based correspondence between scheme-based identification of constraint networks and CNFs.

Theorem 3.23. *If N_S is identifiable relative to all constraint networks and if scheme S has a bounded dimension, then the class C_S is also identifiable relative to all CNFs.*

Proof. Given a bivalued relation ρ and a scheme S, since N_S is identifiable we can decide in polynomial time whether the projection network $\Pi_S(\rho)$ represents ρ. moreover, since $dim(S)$ is bounded the translation from $\Pi_S(\rho)$ to $\Gamma_S(\rho)$ is polynomial. Hence, the conclusion follows because $\Pi_S(\rho)$ describes ρ iff $\Gamma_S(\rho)$ describes ρ. □

The following theorem shows that the converse is not true. Although N_2 is not identifiable, C_2 is, because the satisfiability of 2-CNF is tractable. Consequently, testing of whether $|\Gamma_{S_2}(\rho)| > |\rho|$ can be done in polynomial time [11].

Theorem 3.24. *The class of 2-CNFs is strongly identifiable relative to all CNFs.*

The negative results ahead are based on an extended version of Lemma 3.8 and on the bivalued version of Conjecture 3.10.

Lemma 3.25 (Parallel to Lemma 3.8). *Given a bivalued relation ρ and an arbitrary scheme S, deciding whether $M(\Gamma_S(\rho)) = \rho$ is NP-complete.*

Proof. The proof follows from a simple polynomial translation of any multivalued relation to a bivalued relation and from Lemma 3.8. □

Theorem 3.26 (Parallel to Theorem 3.9). *Given a parameterized scheme class SC_n of bounded dimension, the class C_{SC_n} is not identifiable relative to the class of all CNFs, but is strongly identifiable relative to itself.*

Proof. The proof of the first part is identical to that of Theorem 3.9. The reason that C_{SC_n} is identifiable relative to itself is that the translation from $\Pi_{SC_n}(\rho)$ to $\Gamma_{SC_n}(\rho)$ is polynomial for bounded schemes. □

Conjecture 3.27 (Parallel to Conjecture 3.10). *Given a bivalued relation ρ and an integer k, deciding whether $M(\Gamma_{S_k}(\rho)) = \rho$ is NP-hard.*

Consequently,

Corollary 3.28. *The class of k-CNF is not identifiable relative to all CNFs.*

3.3. Identifying constraint trees

We saw in the previous section that the class N_2 of binary constraint networks is not identifiable, while the class C_S of constraint networks with some specific scheme S is identifiable if S is tractable. We now study a unique class of tractable schemes, those structured as constraint trees, which is identifiable not only when we present one specific scheme (i.e., a particular tree), but also when we have no knowledge of the underlying scheme except that it is a tree.

Let N_T be the class of all constraint networks that have a tree-structured scheme. The following theorem [9] shows that N_T is identifiable.

Theorem 3.29. *Given an arbitrary relation ρ, let $n(x_i)$ be the number of n-tuples in ρ for which $X_i = x_i$, and let $n(x_i, x_j)$ be the number of n-tuples in ρ for which $X_i = x_i$ and $X_j = x_j$. Define the arc-weights $m(X_i, X_j)$ as*

$$m(X_i, X_j) = \frac{1}{|\rho|} \sum_{(x_i,x_j)\in \Pi_{X_i X_j}(\rho)} n(x_i, x_j) \log \frac{n(x_i, x_j)}{n(x_i)n(x_j)}. \qquad (13)$$

If ρ has a constraint-tree representation, then any maximum-weight spanning tree (MWST) formed with the arc-weights defined above constitutes a scheme of such a representation.

Example 3.30. Consider the relation ρ over the binary variables A, B, C, D and E given in Fig. 1. The first step in the algorithm computes the quantities $n(X_i = x_i)$ and $\{n(X_i = x_i, X_j = x_j)\}$ for all variables and their values, obtaining

$$n(A = 0) = 8, \quad (B = 1) = 6, \quad n(B = 0) = 2,$$
$$n(B = 0, C = 1) = 2, \quad n(B = 1, C = 1) = 3, \quad \text{etc.}$$

Next, for each pair of variables (X_i, X_j), we compute the weights $m(X_i, X_j)$, according to equation (13).

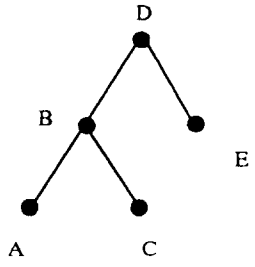
$$m(A,B) = m(A,C) = m(A,D) = m(A,E) = -16.63,$$

$$m(B,C) = -13.97, \quad m(B,D) = -15.95, \quad m(B,E) = -16.55,$$
$$m(C,D) = -16.55, \quad m(C,E) = -17.13, \quad m(D,E) = -15.50.$$

Finally, using the MWST algorithm on these weighted arcs, the tree shown in Fig. 2 is produced. The relations associated with the arcs of the tree are the projections of ρ on pairs of connected variables. For instance, the relation associated with variables D and B is given in Fig. 2(a). The tree generated in this example, together with its associated database (see Fig. 2), represents the original relation, in the sense that it provides a lossless decomposition.

A	B	C	D	E
0	0	1	1	1
0	0	1	1	0
0	1	1	1	1
0	1	1	1	0
0	1	1	0	1
0	1	0	1	1
0	1	0	1	0
0	1	0	0	1

Fig. 1. The input relation ρ for Example 3.30.



D	B
1	0
1	1
0	1

(a)

D	E
1	1
1	0
0	1

(b)

A	B
0	0
0	1

(c)

B	C
0	1
1	1
1	0

(d)

Fig. 2. The tree decomposition of ρ.

In other words, the set of all solutions to this network coincide with relation ρ and they can be recovered efficiently.

Corollary 3.31. *N_T is identifiable in time* $O((|\rho| + \log n)n^2)$.

Proof. The MWST can be constructed in $O((|\rho| + \log n)n^2)$ steps. To verify that the generated tree t represents the input relation ρ, we project ρ onto the arcs of t, compute (in linear time) the number of n-tuples represented by the resulting constraint tree, and compare it to the size of ρ. If the two numbers are equal, the constraint tree represents ρ precisely. Otherwise, we know that no tree representation exists for ρ. □

An alternative method of identifying and constructing tree representations, avoiding the numerical precision required for computing equation (13), is described in [20]. We first project ρ onto all triplets of variables, then examine each triplet for possible redundancy, namely, whether the constraint on one of the pairs is implied by the other two. Next we assign integer

weights to the edges (X_i, X_j) in such a way that any redundant edge should always receive a lower weight than that of a nonredundant edge in the same triangle. Finally, we construct an MWST according to the integer weights thus assigned. It can be shown that the resulting tree has all the properties of the tree constructed by the weights of equation (13).

Although the class of constraint trees is identifiable, it is still open to question whether this class is strongly identifiable; we were not able to prove (or disprove) that the MWST method returns a tightest tree containing ρ when the input relation ρ does not have a tree representation. In all examples examined so far, the returned tree was a tightest one.

Note that although the class of constraint trees is identifiable, it is not learnable with one-sided error, because it is not closed under intersection—the intersection of two trees may form a graph with cycles. Hence, there is no unique tightest tree that contains every subset of positive examples, and this implies that N_T is not learnable with one-sided error.

3.4. Chains, stars, and k-trees

The previous analysis might leave the impression that any class of networks is identifiable as long as the scheme of each network in the class is tractable. To see that this may not be so, consider the class N_c of constraint chains. Being a special kind of a tree, each individual member of N_c is clearly tractable. Yet deciding whether an arbitrary relation ρ can be described by some chain seems an insurmountable task. We do not know of any method of solving this decision problem, save for exhaustive enumeration of all $n!$ chains or all spanning trees that tie for the minimal weight. The difficulty is that since chains are not matroids we do not have a greedy algorithm similar to the MWST for identifying the correct ordering of the variables.

Another class of trees that are not matroids are stars, namely, trees in which one node is adjacent to all the others. This class is (strongly) identifiable, however, primarily due to its low cardinality; there are exactly n stars on n variables. To identify a star, we simply test whether any of the n possible stars represents the input relation ρ. This test, as usual, involves projecting ρ on the edges of the tested star, then counting (in linear time) the number of solutions of the resulting constraint network and testing whether it is equal to $|\rho|$.

An important class of networks is k-trees (or chordal graphs), a powerful generalization of trees (where $k = 1$) investigated in [3,4], and, more recently, [13]. k-Trees can be viewed as trees of clusters, where each cluster consists of a clique of variables with cardinality not exceeding k. Like trees, they are tractably satisfiable. However, not being matroids, k-trees are probably not identifiable. We know of no method (save for exhaustive enumeration) of testing whether an arbitrary relation is representable as a

k-tree for any $k > 1$. In [9] we provide a heuristic algorithm for identifying k-trees which is a generalization of the one for 2-trees.

4. Identifying Horn theories

In this section we shift our attention to bivalued relations and to the task of identifying tractable classes of CNF formulas, such as Horn theories.

In general, determining whether a given query formula follows from a given CNF formula is intractable [8]. However, when the latter contains only Horn clauses, the problem can be solved in linear time [14]. The tractability of Horn theories stems not from the topology of the interactions among their clauses, but rather from a syntactic restriction imposed on each individual clause. This restriction is, in general, less constraining than those imposed by topological considerations; experience with logic programming and databases suggests that humans find it natural to communicate knowledge in terms of Horn expressions. Additionally, the tractability of Horn theories covers a wide range of queries, including, for example, membership, equivalence, disjointness, and entailment. In contrast, the data compression techniques used in classification learning, such as decision trees [26], are effective only for certain membership queries.[2] Thus, it would be useful to determine whether a given set of observations (the data ρ) can be described as a Horn theory.

We shall show that Horn theories are polynomially identifiable; the recognition test of whether ρ has a Horn description can be decided in time proportional to $|\rho|^2 \log |\rho|$, while the time needed to find such a description is proportional to $|\rho|^2 n^2$. However, there are several impediments to using Horn theories as effective approximations to relational data. Selman and Kautz [27], have shown that finding a tightest Horn approximation to a given CNF formula is NP-hard and that any tightest approximation might sometimes require exponentially many clauses (in the size of the source theory). All indications are that similar problems would surface in using Horn expressions to approximate empirical data. First, we have no guarantee that the length of the best approximation would not be exponential in the input $|\rho|$ and, second, we still have no polynomial method of finding the best approximation, even if we take the length of the minimal output theory as the basis for analysis.

In such cases it might be futile to use Horn approximations instead of the observations themselves. A more practical question to ask, then, is whether

[2]For example, decision trees are efficient for determining class membership from a set of properties but do not permit us to effectively infer properties from class membership (and other properties) or to decide whether two decision trees are equivalent or disjoint.

a given relation can be approximated by a Horn theory of a reasonable size. To that end, we analyze the identifiability of k-Horn formulas, namely, Horn formulas in which every clause contains at most k literals. We show that this class of formulas is strongly identifiable.

4.1. Preliminaries

We will assume that all Horn clauses are represented as implications $v_1 \wedge v_2 \wedge \cdots \wedge v_l \longrightarrow z$, where $v_1, v_2, \ldots, v_l$ are positive literals and z may be either a positive literal or 0 (0 stands for "false" and 1 for "true").

Definition 4.1. Let $x = (x_1, x_2, \ldots, x_n)$ be a tuple where $x_i \in \{1, 0\}$. Then *true*(x) is the set of variables assigned 1 by x and *false*(x) is the set of variables assigned the value 0 by x.

Definition 4.2 (*Intersection and closure*). Let x and y be two tuples. Then $x \cap y$ is defined to be the tuple z such that *true*(z) = *true*$(x) \cap$ *true*(y). A set of tuples X is said to be *closed under intersection* if $x \cap y \in X$ whenever $x \in X$ and $y \in X$. A set X^* is said to be the *intersection closure* of X if it is the smallest set containing X that is closed under intersection.

We shall show that a relation ρ has a precise Horn representation iff it is closed under intersection, namely, $\rho = \rho^*$. The proof is based on the notion of *extreme tuples*[3], and on Lemma 4.4 below.

Definition 4.3 (*Extreme tuples*). Given a set X of tuples, $x \in X$ is said to be *extreme* (relative to X) if it is not in the intersection closure of $\{X - x\}$. A tuple that is not extreme is called *interior*, and the set of extreme tuples will be called the *basis* of X, denoted $B(X)$.

Remarks. It is easy to show that every set X has a unique basis $B(X)$ and, moreover, that $B(X)$ is the minimal set having the same closure as X, namely, $[B(X)]^* = X^*$. $B(X)$ is also the maximal subset of X that is the basis of itself, that is, $B(X)$ is the maximal $Y \subseteq X$ satisfying $B(Y) = Y$. Finally, the basis of any set X can be found in time quadratic in $|X|$; we simply check for every $x \in X$ whether it is extreme (relative to X) by intersecting all tuples $y \in X$ such that *true*$(y) \supset$ *true*(x) and then testing whether the intersection differs from x.

[3]This notion, and its connection to Horn approximation (Corollary 4) was brought to our attention by H. Kautz. Lemma 4.5 which makes this connection possible, appears to be a general folklore among many researchers, although we could not trace its precise origin. An explicit proof for the propositional case is presented below; alternative treatment is given in [28].

Lemma 4.4. *Given a set X of tuples and another tuple t such that t is not in the intersection closure of X, there exists a Horn theory H that contains X and excludes t.*

Proof. Construct H as follows: Start with the set of all Horn clauses and remove every clause that conflicts with any tuple $x \in X$. We will show that among the remaining clauses, there must be at least one clause c' that excludes t. Consider the set of variables $T(t)$ such that:

$$T(t) = \bigcap_{x: true(x) \supset true(t)} true(x),$$

and assume $T(t)$ is nonempty. Since t is not in the intersection closure of X, $true(t)$ is a proper subset of $T(t)$, and we can choose an element z from $\{T(t) - true(t)\}$ and form the clause $c' : \bigwedge_i v_i \longrightarrow z$ where $v_i \in true(t)$. This clause will not be removed during the construction of H, because it is not violated by any $x \in X$. For c' to be violated by a tuple x, $true(t)$ must be a subset of $true(x)$ and z must be in $false(x)$, but c' was chosen such that each set $true(x)$ that contains $true(t)$ also contains z. Thus, since t violates c' and H contains c', t will not be a model of H. The argument remains valid in case $T(t)$ is empty, because in that case we have $z = 0$, thus, c' will eliminate t but nothing else. □

Lemma 4.5. *Let ρ be a set of tuples. Then ρ is the set of models of some Horn theory H if and only if it is closed under intersection.*

Proof. Let ρ be closed under intersection, and let H be the tightest Horn theory containing ρ. Suppose H has a model t that is not in ρ. Since ρ is closed under intersection, t cannot be in the intersection closure of ρ and, according to Lemma 4.4, it is possible to form a Horn theory, H_1, that contains ρ and excludes t. The Horn theory $H \cap H_1$ is clearly a tighter approximation of ρ, contrary to our assumption. This establishes the "if" part of the lemma. The "only if" part follows from showing that any clause that satisfies x and y also satisfies $x \cap y$ (see [31] and [2, Lemma 3]). □

Corollary 4.6. [4] *Let $h(\rho)$ stands for any tightest Horn approximation of an arbitrary relation ρ, and let $B(\rho)$ stand for the basis of ρ. Then*

$$h[B(\rho)] \approx h(\rho) \approx h(\rho^*).$$

[4] Independently proved in [28].

Proof. For any ρ, we have $h(\rho) \approx h(\rho^*)$, because Lemma 4.5 dictates $M[h(\rho^*)] = \rho^*$, and so, if $M[h(\rho)]$ were a proper subset of $M[h(\rho^*)] = \rho^*$, it would constitute a set closed under intersection containing ρ that is properly contained in ρ^*, thus violating the status of ρ^* as the closure of ρ (see Definition 4.2). Therefore, substituting $B(\rho)$ for ρ, we also have $h[B(\rho)] \approx h[B(\rho)^*]$. Moreover, since $[B(\rho)]^* = \rho^*$, we get $h[B(\rho)] \approx h(\rho^*) \approx h(\rho)$, which proves the corollary. □

Corollary 4.7. *Let X be a set of extreme tuples. Then, for every subset Y of X, there exists a Horn theory H that contains $X - Y$ and excludes Y. Conversely, if such a theory can be found for every subset Y of X, then X must be a set of extreme tuples.*

Proof. Let $Y = \{y_1, y_2, \ldots\}$. From Lemma 4.4, there exists a set of Horn theories $\{H_1, H_2, \ldots\}$ such that H_i contains $\{X - y_i\}$ and excludes y_i. Clearly, the union of clauses in $\{H_1, H_2, \ldots\}$ is a Horn theory that satisfies the condition of the corollary. The converse follows from the fact that if some member of X is given by the intersection of several other elements of X then every Horn theory that contains the latter must also contain the former. □

Corollary 4.8. *The VC-dimension of (the class of) Horn theories is* $O[\exp(n/2)]$.

Proof. The VC-dimension is defined as the maximum number of points that can be "shattered" by Horn theories, in the sense of Corollary 4.7 [5]. Accordingly, this number is equal to the maximum number of n-tuples such that none is in the intersection closure of the rest, namely,

$$\binom{n}{n/2} = O[\exp(n/2)]. \qquad \square$$

The VC-dimension plays an important role in the framework of PAC-learning, where it is used to assess the number of random samples needed before the error associated with learning an incorrect theory can be bounded [5]. Roughly speaking, Corollary 4.8 states that approximately as many samples as the square root of the 2^n possible tuples are needed before one can be fairly confident that the Horn theory learned does not deviate substantially from the one generating the data. Since the VC-dimension grows exponentially in n, we conclude that Horn theories are not polynomially learnable (in the PAC sense) from random examples [5]. This negative result has no significant impact on identification tasks, where we assume that all models of the learned theory are available explicitly. It does mean, however, that every Horn theory H can be completely characterized by at most $\exp(n/2)$

of its models. Conversely, at most $\exp(n/2)$ tuples would ever be needed to characterize ρ if we are determined to approximate ρ by a Horn theory.

4.2. Recognition, description, and approximation

This subsection analyzes the three conditions required for the identifiability of Horn theories. We first show that Horn theories are identifiable by analyzing the recognition and description conditions. We later address difficulties in finding tightest Horn approximations—the condition needed for strong identifiability.

Theorem 4.9. *Deciding whether an arbitrary set ρ of tuples can be represented by a Horn theory can be done in $O(|\rho|^2 \log|\rho|)$ time.*

Proof. According to Lemma 4.5, it is enough to test whether ρ is closed under intersection. This can be done by simply checking whether the intersection of any two elements in ρ is in ρ [17]. This method requires $O(|\rho|^2 \log|\rho|)$ steps (the number of pairs times the time to check whether the intersection of the pair is in ρ), thus proving the theorem. □

Theorem 4.9 establishes the recognition part of the identification task. We remark that of all the classes considered in this paper, the class of Horn theories is unique in that we can decide the existence of a description in the class without actually producing one. Consequently, the time required for reaching this decision is independent of the length of the final theory (if such exists).

To fully establish the identifiability of Horn theories, there remains to show that whenever ρ is describable by a Horn theory, it is possible to find one such theory in polynomial time. This task is facilitated by a learning algorithm called HORN, recently devised by Angluin, Frazer, and Pitt [2].[5] The learning algorithm of Angluin et al. assumes that an oracle possesses a target Horn theory H^* having m clauses, and tries to find a Horn theory equivalent to H^* by asking the oracle two types of queries: *equivalence* and *membership*. An equivalence query asks whether some conjectured Horn theory H is equivalent to H^*, and its answer is either a confirmation or a counterexample (i.e. an assignment that satisfies H^* but not H, or vice versa). When there are no counterexamples, the learning algorithm has clearly succeeded in identifying a correct theory. Membership queries allow the algorithm to ascertain whether a given tuple satisfies the target theory

[5]This algorithm and the possibility of simulating it on relational data was brought to our attention by an anonymous reviewer. These results were independently recognized by Kautz, Kearns, and Selman [17]. A direct and more efficient algorithm is described in the appendix.

H^*; they are answered "yes" or "no" by the oracle. Angluin et al. have shown that HORN finds a theory equivalent to H^* in polynomial time, making $O(mn)$ equivalence queries and $O(m^2n)$ membership queries. Moreover, every theory H that HORN presents as a conjecture (including, of course, the final output theory) has at most $m(n+1)$ clauses.

To find a Horn description of a given relation ρ, we simply simulate HORN by addressing its queries to the data ρ rather than to the oracle's theory H^*. A membership query is answered by simply checking whether the tuple presented is in ρ. Equivalence queries can be answered as follows: Given a conjectured Horn theory H, we first check that every tuple of ρ satisfies H. If not, we return the unsatisfying tuple as a counterexample. Otherwise, $M(H)$ contains ρ, and we then determine whether $M(H) = \rho$ by the polynomial enumeration method of [11].

Thus, since we can correctly answer the two basic queries of HORN, the simulation algorithm must output an exact Horn representation of ρ if one exists. Moreover, since every membership query takes $O(n \log |\rho|)$ time, and every equivalence theory takes $O(|\rho||H|)$ time, we conclude that a Horn description of ρ can be found in

$$\begin{aligned} O(mn|\rho||H| + m^2n^2 \log|\rho|) &= O(m^2n^2(|\rho| + \log|\rho|)) \\ &= O(m^2n^2|\rho|) \end{aligned}$$

time, where m is the minimum number of clauses in any Horn theory describing ρ.

This essentially establishes the identifiability of Horn theories as prescribed in Definition 2.6, according to which a description must be found in time polynomial in both the input ($|\rho|$) and the shortest possible output ($m(n+1)$). However, we can establish an even stronger result by showing that m is polynomial in $|\rho|$, namely, the size of the shortest output theory cannot be substantially larger than the input. Indeed, it is possible to show (see appendix) that every Horn theory with K models can be expressed by a Horn formula that employs at most Kn^2 clauses. Translated to our setting, this means that m cannot exceed $|\rho|n^2$ and hence that the simulation algorithm will find a Horn description for ρ of length $O(|\rho|n^3)$ clauses. Moreover, while the time it takes the HORN algorithm to find this description is $O(|\rho|^3n^6)$, a simpler algorithm can be found (see appendix), which works directly on ρ, finds a description of length $O(|\rho|n^2)$ clauses, and runs in only $O(|\rho|^2n^2)$ steps.

We summarize this analysis by stating:

Theorem 4.10. *If a relation ρ has a Horn description, then one such description, having at most $|\rho|n^2$ clauses, can be found in $O(|\rho|^2n^2)$ time.*

Combining Theorems 4.9 and 4.10 we now have:

Corollary 4.11. *Horn theories are identifiable in input-polynomial time.*

We remark that the tight connection between the number of models and the number of clauses renders Horn theories a useful tool in data compression. Short relations are guaranteed to have short theories, whereas the converse does not hold; some extremely long relations may have short Horn descriptions. For example, the theory containing a single clause, say $a \longrightarrow b$, has exponentially many models. Recent study reveals that such a tight connection does not exist between the length of a Horn theory and the number of extreme models it may have [17]. In other words, the number of extreme models of some Horn theory with m clauses is exponential in m and, conversely, the length of the shortest Horn theory having K extreme models may be exponential in K and n. This implies that the tightest Horn approximation of some long relations may be much shorter than the relation, even if the input data consists of only extreme models. Conversely, it raises the interesting possibility that the basis of ρ, which can be computed in $O(|\rho|^2)$ steps, could serve as a more economical description of ρ than any Horn approximation of ρ.

This brings us to the complexity of finding tightest Horn approximations to relations ρ that do not have precise Horn descriptions. In principle, we can find such approximations by simulating the HORN algorithm again, this time referring all its queries to the closure ρ^* of ρ, which we can keep implicit. We know from Corollary 4 that $B(\rho)$ contains all the information about ρ^*, and that ρ^* has a Horn representation that is the tightest Horn approximation of ρ. This simulation would have a difficulty answering equivalence queries, however. Whereas previously we were able to answer equivalence queries in time polynomial in $|\rho|$, the size of ρ^* may be exponential in ρ, and we do not have a way of testing whether $M(H) \subseteq \rho^*$ except by enumerating $M(H)$ and ρ^*. Thus, the strong identifiability of Horn theory remains an open problem.

We can still assert a weaker result:

Corollary 4.12. *Horn theories are strongly identifiable for every dataset whose intersection closure is of a polynomial size.*

Corollary 4.12 might seem weak in view of the fact that there is no simple method of estimating the size of ρ^*, short of actually enumerating ρ^*. However, if the size of ρ^* is substantially larger than that of ρ, we know that any Horn approximation is bound to be very poor. It is only when $|\rho^* - \rho|$ is a fraction of $|\rho|$ that Horn theories can offer a reasonable approximation to ρ, and it is precisely in those cases that we can find a tightest Horn

approximation in a reasonable time. This suggests a strategy of focusing the development of Horn approximations on those cases only that can benefit from such approximations. Given a relation ρ and a tolerance level τ, we begin to generate the closure of ρ and test whether its size exceeds $(1+\tau)|\rho|$. If it does, we know that no acceptable Horn approximation is feasible. If $|\rho^*| < (1+\tau)|\rho|$, we proceed to find a tightest Horn approximation using either the HORN simulation or the envelope-based algorithm described in the appendix.

4.3. Identifying k-Horn formulas

We now restrict our attention to the identifiability of k-Horn formulas.

As before, S_k denotes the set of all subsets of X of size k or less. A tightest k-Horn approximation can be generated by first constructing the tightest CNF approximation over the scheme S_k and then eliminating all non-Horn clauses from that approximation. In other words: given a relation ρ on n variables and a constant k, we generate the formula $\Gamma_{S_k}(\rho)$ and throw away all non-Horn clauses. We claim that the resulting Horn theory is the tightest k-Horn approximation of ρ (which may have, of course, many equivalent syntactic representations). Since, as we will show, this is also the longest form of the tightest approximation, we then generate an equivalent reduced version by eliminating subsumptions. To test if the resulting Horn theory represents ρ exactly, we enumerate its models. Note, however, that while there are 2^k clauses over a set of k symbols, there are only $k+1$ Horn clauses over the same set. Thus, it makes more sense to go in the opposite direction: first enumerate all possible Horn formulas over scheme S_k, then eliminate those clauses that conflict with any tuple of ρ. It can be shown that these two methods yield the same expression. Given a CNF formula φ, we denote by $Horn(\varphi)$ the formula resulting from eliminating all non-Horn clauses from φ. Given a relation ρ, let $\Omega_k(\rho)$ be the set of all possible Horn formulas over scheme S_k that are consistent with all tuples of ρ.

Theorem 4.13. *Let ρ be an n-ary bivalued relation, k a constant, $\pi = \Gamma_{S_k}(\rho)$, and $\eta = Horn(\pi)$. Let H_k be the family of k-Horn formulas, then*

(1) *η is a tightest k-Horn approximation of π,*
(2) *η is maximal with respect to H_k,*
(3) *if $M(\eta) \supset \rho$, no k-Horn formula describes ρ,*
(4) *reduced(η) equals the set of all k-Horn prime implicates of η,*
(5) *$\eta = \Omega_k(\rho)$.*

Proof. (1) and (2) follow from the fact that π already contains all k-Horn clauses consistent with ρ. (3) follows immediately from the tightness of η. Since the scheme S_k contains all subsets of size k or less, it follows

Algorithm Horn-generation(ρ, k)

Input: A relation ρ on n variables and an integer k.
Output: A k-Horn formula describing ρ or a k-Horn tightest approximation of ρ.

(1) Enumerate Ω_k, the set of all Horns over S_k.
(2) Eliminate any Horn clause that violates ρ, resulting in $\Omega_k(\rho)$.
(3) $\eta \Leftarrow= reduced(\Omega_k(\rho))$ (by eliminating subsumptions).
(4) Enumerate the models of η, $\{m_1, m_2, \ldots\}$, using the method in [11], and, if for some $i \leqslant |\rho|$, $m_i \notin \rho$, or if $M(\eta)$ contains more than $|\rho|$ elements, then return: "η is a tightest k-Horn approximation"; else, return "η describes ρ".

Fig. 3. Algorithm Horn-generation.

from Lemma 3.22 that $reduced(\eta)$ contains all and only the k-Horn prime implicates of η, thus proving (4). Finally, (5) follows from (2) and from the observation that by definition, $\Omega_k(\rho)$ is the tightest maximal k-Horn of ρ. □

Theorem 4.13 implies that algorithm *Horn-generation* in Fig. 3 which outputs the formula $reduced(\Omega_k(\rho))$, is guaranteed to return the tightest k-Horn approximation of ρ relative to H_k. The algorithm also returns a statement as to whether the formula found is an exact representation of ρ.

To summarize:

Corollary 4.14. *Algorithm Horn-generation provides a tightest k-Horn approximation of an arbitrary relation ρ. Moreover, this approximation equals the k-Horn prime implicates of ρ.*

Example 4.15. Consider again the relation

$$\rho(PQR) = \{(100), (010), (001)\}$$

and let $k = 2$. For this example, it is easier to first list the tightest k-CNF approximation and then eliminate non-Horn clauses. We have

$$\begin{aligned} \Pi_{S_2}(\rho) = \{&(\rho(P,Q) = \{(10), (01), (00)\}), \\ &(\rho(P,R) = \{(10), (01), (00)\}), \\ &(\rho(Q,R) = \{(10), (01), (00)\})\}, \end{aligned}$$

and $P = \{0,1\}$, $Q = \{0,1\}$, $R = \{0,1\}$. When applying the canonical transformation to each of these relations, we get the (already reduced) formula

$$\Gamma_{S_2}(\rho) = \{(\neg P \vee \neg Q), (\neg P \vee \neg R), (\neg R \vee \neg Q)\}.$$

Since this is a Horn formula, we need not throw any clauses away. Computing the number of models of this theory yields four models (there is an additional (0,0,0) tuple), so we conclude that the formula is a tightest 2-Horn approximation of ρ and that ρ is not 2-Horn identifiable. If we generate the 3-Horn approximation for ρ, we get the same formula (because, in this case, the 2-Horn approximation already contains all its Horn prime implicates). Going through algorithm *Horn-generation*, step (2) yields:

$$\begin{aligned}\Omega_k(\rho) = \{&(\neg P \vee \neg Q \vee R), (P \vee \neg Q \vee \neg R), (\neg P \vee Q \vee \neg R),\\ &(\neg P \vee \neg Q \vee \neg R), (\neg P \vee \neg Q), (\neg P \vee \neg R), \neg R \vee \neg Q)\}.\end{aligned}$$

The result of further eliminating subsumptions yields the same formula:

$$\textit{reduced}(\Omega_k(\rho))) = \{(\neg P \vee \neg Q), (\neg P \vee \neg R), (\neg R \vee \neg Q)\}. \quad (14)$$

Example 4.15 suggests an *anytime* variation of the algorithm described in Fig. 3. Instead of applying the algorithm to all subsets of size k, we first apply the algorithm to subsets of size 2, then add the result of processing subsets of size 3, and so on, until we get a satisfying approximation. The algorithm is given in Fig. 4. Let us denote by $S_{(k)}$ all subsets of size exactly k and by $\Omega_{(k)}$ the set of all Horn clauses of length k that are consistent with ρ.

Note that the algorithm always retains the unreduced formula generated in the previous iteration.

We next assess the complexity of our approximation and the size of its resulting Horn theory.

Theorem 4.16 (Complexity).

(1) *The length (number of clauses) of reduced*$(\Omega_k(\rho))$ *is* $O(kn^{k+1})$.

(2) *The complexity of Horn-generation*(ρ, k) *is* $O(|\rho| kn^{k+1})$.

Proof. Since worst-case analysis is unable to distinguish between a maximal formula and its reduced form, we assume that the algorithm generates the former.

(1) Since there are $i + 1$ distinct Horn clauses on a subset of size i and since there are n^{k+1} subsets in scheme S_k, the overall number of Horn clauses is $O(kn^{k+1})$.

Algorithm Anytime-Horn-generation(ρ, k)

Input: A relation ρ and a constant k.
Output: A Horn formula describing ρ or a k-Horn tightest approximation to ρ.

(1) $\pi \Leftarrow= \Omega_{(1)}(\rho)$
(2) For $i = 2$, while $i \leqslant k$, do
- $\pi \Leftarrow= \pi \cup \Omega_{(i)}(\rho)$
- $\eta \Leftarrow= reduced(\pi)$
- if $|M(\eta)|$ equals $|\rho|$, then return "η describes ρ".

(3) endwhile.
(4) Return η and a statement "η is a tightest k-Horn approximation of ρ".

Fig. 4. Algorithm Anytime-Horn-generation.

(2) Generating all possible Horn clauses over S_k not in conflict with ρ takes $O(|\rho|kn^{k+1})$. Eliminating subsumption may take additional $O((2n)^k)$, resulting in an overall time complexity of $O((|\rho| + kn + 2^k)n^k)$. Finally, computing the number of models of a Horn theory is linear in the theory size and the number of models [11]. Therefore, testing whether this number exceeds $|\rho|$ takes $O(kn^{k+1}|\rho|))$ steps. □

Corollary 4.17. *The class of k-Horn theories is strongly identifiable in* $O(|\rho|n^{k+1})$ *time.*

Interestingly, algorithm Horn-generation can easily be converted into an on-line version, which is useful for stream processing. Assume the tuples of ρ are not available all at once, but are obtained sequentially as a stream of observations, normally containing many repetitions. In this case it might be advantageous to store a parsimonious theory of past data, rather than the data itself, and to update the theory incrementally whenever an observation arrives that contradicts the theory. If storage space permits, the update can be made particularly easy if in addition to the reduced approximation η we also keep the maximal tightest Horn approximation π. Then, whenever a new tuple arrives, all clauses in π that conflict with it are eliminated, and the resulting theory can now be reduced to form a new η so as to facilitate query answering. When the size of π is much larger than that of η, it might be advantageous to store only η and compute the maximal π on

the fly, update π to conform with the new tuple and reduce it back to more economical form. The time it takes for this operation is $O(n^k)$ per update.

5. Conclusions

This paper summarizes several investigations into the prospects for identifying meaningful structures in empirical data. The central aim is to identify a computationally attractive description, in cases where the observed data possess such a description and a best approximate description otherwise. The feasibility of performing this task in reasonable time has been given a formal definition through the notion of identifiability, which is normally weaker (if $C' = C$) than that of learnability.

In exploring the decomposition of data into a given scheme of smaller relations, it was shown that, whereas a best approximation can always be found, it is only in cases where the scheme is tractable that we can (tractably) decide whether the resulting approximation constitutes an exact representation of the data. It is worth noting that the difficulty associated with this decision can be mitigated by allowing approximation through sampling. It is a known result by Angluin [1] that polynomial-time algorithms for exact identification of concepts using equivalence and membership queries can be transformed into polynomial-time PAC learning algorithms using membership queries only. In our case, the difficulty associated with confirming the exactness of the tightest theory amounts to that of answering an equivalence query, and hence, it can be transformed to answering a sequence of (randomly sampled) membership queries, yielding an approximately correct confirmation of the exactness of the output theory.[6]

The decomposition of data into a structure taken from a *class* of schemes turned out to be a harder task, one that is intractable even in cases where each individual member of the class is tractable. The class of tree structured schemes is an exception. Here it was shown that an effective procedure exists for determining whether a given relation is decomposable into a tree of binary relations and, if the answer is positive, identifying the topology of such a tree. The procedure runs in time proportional to the size of the relation, but whether it provides a tightest tree-structured approximation in cases where the answer is negative is still an open question.

Focusing on bivalued data, we then explored the identification of descriptions whose tractability stems from syntactic rather than structural features. In particular, we showed that Horn theories can be identified in input-polynomial time, that is, one can decide whether the input data possesses an exact Horn description and find such a description (whenever

[6]For further detail see [17].

possible) in time polynomial with the length of the input. The strong identifiability of Horn theories, that is, the problem of finding a tightest Horn approximation, remains open. Since there are small sets of models with exponentially long tightest Horn approximations [17], the best one can hope for is an output-polynomial algorithm for generating such approximations. So far, only sampling algorithms are known for this task, namely, algorithms which guarantee that the output theory is "probably almost tightest", thus rendering Horn theories "strongly PAC-identifiable". Whether there is an output-polynomial algorithm that returns the tightest Horn approximation is still an open question.

By contrast, k-Horn theories were shown to be strongly identifiable in polynomial time, when k is bounded. Both anytime and on-line algorithms where discussed for identifying these theories.

An important issue not dealt with in this paper is assessment of the goodness of the approximations provided by Horn theories. Another issue is the feasibility of constructing both an upper bound and a lower bound approximations of ρ, in the manner discussed in [27] and also in [9]. Finally, we should mention that the methods presented in this paper will also handle partial observations, namely, observations of truncated tuples of ρ.

Appendix A. Proof of Theorem 4.10

In this appendix we prove the two assertions stated in Theorem 4.10, Section 4.2:[7]

(1) Every Horn formula with K models has an equivalent Horn formula that employs at most Kn^2 clauses.
(2) Given a relation ρ, closed under intersection, it is possible to find a Horn description of ρ in time $O(|\rho|^2 n^2)$.

Let x and y be two arbitrary tuples. We say that x is an ancestor of y (equivalently, y is a descendant of x) if $true(x) \supset true(y)$; we say that x is a parent of y (equivalently, y is a child of x) if x is an ancestor of y and $|true(x)| = |true(y) + 1|$. Let ρ be a set of tuples closed under intersection, we say that x is a least ancestor of y (relative to ρ) if x is in ρ and y has no other ancestor z in ρ such that $true(z) \subset true(x)$. Note that if y is not in ρ, then it either has a unique least ancestor (since the intersection of any two ancestors in ρ yields another ancestor in ρ), or it has no ancestor in ρ, in which case we say that the least ancestor of y is $\emptyset$.

[7] We are indebted to Dana Angluin for outlining the method used in this proof.

Define the *envelope* $E(\rho)$ of ρ as the set of tuples *not* in ρ that either have a child in ρ or have no child at all (the latter corresponds to the tuple containing all zeros). Clearly, there are at most $n|\rho|$ elements in $E(\rho)$. Also, every tuple that is not in $E(\rho)$ must either be in ρ or have a descendant in $E(\rho)$.

Let e be an element in $E(\rho)$ and let e' be the (unique) least ancestor of e in ρ (possibly $\emptyset$). Attach to every pair (e, e') a Horn theory $H(e, e')$ containing one clause, $c_i = \textit{antecedent} \rightarrow v_i$ for every variable v_i in $\textit{true}(e') - \textit{true}(e)$, where *antecedent* stands for the conjunction of all positive literals of $\textit{true}(e)$.[8] If $\textit{true}(e) = \{\emptyset\}$, then $c_i = v_i$ and if $e' = \emptyset$, then $v_i = 0$. Note that $H(e, e')$ excludes those and only those tuples that are ancestors of e and not of e'.

Lemma A.1. *Let ρ be a relation closed under intersection and let $H(\rho)$ be the Horn formula formed by collecting the clauses from all the subtheories $H(e, e')$, where e ranges over all elements of $E(\rho)$. Then $H(\rho)$ constitutes a precise description of ρ, and contains at most $|\rho|n^2$ clauses.*

Proof. It is easy to show that every tuple in ρ is a model of $H(\rho)$. For if a tuple x in ρ conflicts with any $H(e, e')$ then x must be an ancestor of e and not of e', and then the intersection $x \cap e'$ which is in ρ would also be an ancestor of e with a smaller number of ones than e'. Hence e' cannot be the least ancestor of e in ρ.

To prove that every model of $H(\rho)$ is in ρ we show that the opposite alternative leads to a contradiction. Suppose there is a model y of $H(\rho)$ that is not in ρ. Since $H(\rho)$ excludes all tuples in $E(\rho)$, it is clear that y cannot be in $E(\rho)$. Since y itself is not in ρ or in $E(\rho)$, there must be at least one descendant of y that is in $E(\rho)$; let z be a maximal such descendant (i.e., there is no $x \in E(\rho)$ that is an ancestor of z and a descendant of y. Being in $E(\rho)$, z must contribute a set of clauses $H(z, z')$ to H, where z' is the least ancestor of z in ρ and $H(z, z')$ excludes all ancestors of z unless they are also ancestors of z'. Thus, if y is a model of H, it must be that y is also an ancestor of z'. Now consider any descending path P from y to z (i.e., every pair of successive elements along P consists of a parent followed by its child). Since y is not in $E(\rho)$ and z' is in ρ, P must contain an element $z'' \in E(\rho)$ such that z'' is an ancestor of z' and a descendant of y. But this contradicts our assumption that z is the maximal descendant of y in $E(\rho)$. □

[8] For example, for variables a, b, c, d, and $e = (1,0,1,0)$, $e' = (1,1,1,1)$, we have $\textit{true}(e) = \{a, c\}$, $\textit{true}(e') = \{a, b, c, d\}$, and $H(e, e') = \{a \wedge c \rightarrow b, a \wedge c \rightarrow d\}$.

Theorem A.2. *Every Horn formula with K models has an equivalent Horn formula that employs at most Kn^2 clauses.*

Proof. The proof follows immediately from Lemma A.1. If ρ stands for the models of a Horn formula H', then ρ must be closed under intersection and contain precisely K models. From Lemma A.1, an equivalent Horn formula H can be constructed from the elements of $E(\rho)$ that describes ρ precisely and employs at most $n|E(\rho)|$ clauses. Since, each of the K elements in ρ can contribute at most n elements to $E(\rho)$, we conclude that the number of clauses in H is at most Kn^2. □

We will now prove the second claim by analyzing the complexity of constructing H.

Theorem A.3. *Given a relation ρ, closed under intersection, it is possible to find a Horn description of ρ in time $O(|\rho|^2 n^2)$.*

Proof. Assume $|\rho| = K$. The construction of H consists of three parts:

(a) identifying the elements of the envelope $E(\rho)$,
(b) identifying the pair (e, e') for every element in $E(\rho)$, and
(c) constructing the formulas $H(e, e')$ for every pair found in (b).

Part (a) can be done in $O(nK \log K)$ time, simply testing which of the n parents of each member of ρ is not in ρ.

Part (b) requires the identification of the least ancestor $e' \in \rho$ for each member $e \in E(\rho)$. Clearly, there are at most nK elements e in $E(\rho)$, and identifying e' requires at most $2nK$ steps (i.e., taking each element of ρ and testing whether it is an ancestor of e, then taking its intersection with that of previously found ancestors of e). This takes a total of at most $2n^2K^2$ operations.

Part (c) requires at most n operations for each of the (e, e') pairs, of which there are at most nK. This gives a total of n^2K operations.

The dominant effort is clearly part (b) , yielding a total of $O(K^2n^2)$ steps, thus confirming the theorem. □

We remark that while the envelope-based algorithm described in the proof of Theorem A.3 yields a theory of size $O(|\rho|n^2)$ and the HORN simulation algorithm produces a theory of size $O(|\rho|n^3)$, the latter has the advantage of always producing theories that lie within a factor $(n + 1)$ of the shortest possible theory representing ρ. Thus, in cases where a long ρ is suspected of having a short Horn description, it is worth running HORN instead of the envelope-based algorithm. Alternatively, it is possible in such cases to run the HORN algorithm directly on the theory H found by the envelope-based

algorithm, so as to reduce its length. Given any Horn theory H, if we use H to answer the queries of HORN, then HORN is guaranteed to yield a theory equivalent to H, whose length is within a factor $n + 1$ of $H_{\min}$, the shortest Horn equivalent of H. This simplification procedure runs in time proportional to $n|H|^2|H_{\min}|$.

Acknowledgement

We are indebted to many colleagues for most generous assistance. We deeply appreciate the insightful comments of three anonymous referees, who were responsible for many improvements and for illustrating how we could use the HORN algorithm. Conversations with Henry Kautz, Michael Kearns, and Bart Selman have pushed us toward many of the results in Sections 4.1 and 4.2, and the ideas of Dana Angluin were responsible for the claims and proof of Theorem 4.10. Jeff Ullman has been most helpful in generating a proof of Lemma 3.8, and Itay Meiri and Amir Weinstein contributed several ideas in the early stages of this paper.

References

[1] D. Angluin, Queries and concept learning, *Mach. Learn.* **2** (1988) 319–342.
[2] D. Angluin, M. Frazier and L. Pitt, Learning conjunctions of Horn clauses, in: *Proceedings 31st Annual Symposium on Foundations of Computer Science, Vol. I* (IEEE Computer Society Press, St. Louis, MO, 1990).
[3] S. Arnborg, Efficient algorithms for combinatorial graphs with bounded decomposability—a survey, *BIT* **25** (1985) 2–23.
[4] C. Beeri, R. Fagin, D. Maier and M. Yannakakis, On the desirability of acyclic database schemes, *J. ACM* **30** (1983) 479–513.
[5] A. Blumer, A. Ehrenfeucht, D. Haussler and M. K. Warmuth, Learnability and the Vapnik–Chervonenkis dimension, *J. ACM* **36** (1989) 929–965.
[6] R.K. Brayton, G.D. Hachtel and A.L. Sangiovanni-Vincentelli, Multilevel logic synthesis, *Proc. IEEE* **78** (2) (1990).
[7] C.K. Chow and C. N. Liu, Approximating discrete probability distributions with dependence trees, *IEEE Trans. Inf. Theor.* **14** (1968) 462–467.
[8] S.A. Cook, The complexity of theorem-proving procedures, in: *Proceedings 3rd Annual ACM Symposium on the Theory of Computing*, New York (1971) 151–158.
[9] R. Dechter, Decomposing a relation into a tree of binary relations, *J. Comput. Syst. Sci.* **41** (1990) 2–24 (Special Issue on the Theory of Relational Databases).
[10] R. Dechter, Constraint networks, in: *Encyclopedia of Artificial Intelligence* (Wiley, New York, 2nd ed., 1992) 276–285.
[11] R. Dechter and A. Itai, The complexity of finding all solutions, UCI Rept., University of California, Irvine, CA (1991).
[12] R. Dechter and J. Pearl, Network-based heuristics for constraint-satisfaction problems, *Artif. Intell.* **34** (1) (1987) 1–38.
[13] R. Dechter and J. Pearl, Tree clustering for constraint networks, *Artif. Intell.* **38** (3) (1989) 353–366.
[14] W.F. Dowling and J.H. Gallier, Linear time algorithms for testing the satisfiability of propositional Horn formula, *J. Logic Program.* **3** (1984) 267–284.

[15] E.C. Freuder, Complexity of k-structured constraint satisfaction problems, in: *Proceedings AAAI-90*, Boston, MA (1990) 4–9.
[16] C. Glymour, R. Scheines, P. Spirtes and K. Kelly, *Discovering Causal Structure* (Academic Press, Orlando, FL 1987).
[17] H.A. Kautz, M. Kearns and B. Selman, *Horn Approximations of Empirical Data*, AT&T Bell Laboratories (1992).
[18] P.F. Lazarsfeld, Latent structure analysis, in: S.A. Stouffer, L. Guttman, E.A. Suchman, P.F. Lazarsfeld, S.A. Star and J.A. Claussen, eds., *Measurement and Prediction* (Wiley, New York, 1966).
[19] D. Maier, *The Theory of Relational Databases* (Computer Science Press, Rockville, MD, 1983).
[20] I. Meiri, R. Dechter and J. Pearl, Tree decomposition with applications to constraint processing, in: *Proceedings AAAI-90*, Boston, MA (1990) 10–16.
[21] U. Montanari, Networks of constraints, fundamental properties and applications to picture processing, *Inf. Sci.* **7** (1974) 95–132.
[22] U. Montanari and F. Rossi, Fundamental properties of networks of constraints: a new formulation, in: L. Kanal and V. Kumar, eds., *Search in Artificial Intelligence* (Springer, New York, 1988) 426–449.
[23] B.K. Natarajan, On learning Boolean functions, in: *Proceedings 19th Annual ACM Symposium on Theory of Computation*, New York (1987) 296–304.
[24] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, San Mateo, CA, 1988).
[25] J. Pearl and T. Verma, A theory of inferred causation, in: J.A. Allen, R. Fikes and E. Sandewall, eds., *Principles of Knowledge Representation and Reasoning: Proceedings Second International Conference* (Morgan Kaufmann, San Mateo, CA, 1991) 441–452.
[26] J.R. Quinlan, Induction of decision trees, *Mach. Learn.* **1** (1986) 81–106.
[27] B. Selman and H.A. Kautz, Knowledge compilation using Horn approximation, in: *Proceedings AAAI-91*, Anaheim, CA (1991).
[28] B. Selman and H.A. Kautz, Tractability through theory approximation, AI Tech. Rept., AT&T Bell Laboratories, Murray Hill, NJ (1992).
[29] J. Ullman, Personal communication (1991).
[30] L.G. Valiant, A theory of the learnable, *Commun. ACM* **27** (11) (1984) 1134–1142.
[31] M.A. van Emden and R.A. Kowalski, The semantics of the predicate logic as a programming language, *J. ACM* **23** (1976) 733–742.