Identifying the Minimal Transversals of a Hypergraph and Related Problems

Thomas Eiter, Georg Gottlob
Christian Doppler Labor für Expertensyteme
Technische Universität Wien
Paniglgasse 16, 1040 Wien

{eiter|gottlob}@vexpert.dbai.tuwien.ac.at

January 14, 1991

CD-TR 91/16

Identifying the Minimal Transversals of a Hypergraph and Related Problems[*]

Thomas Eiter and Georg Gottlob

Christian Doppler Laboratory for Expert Systems
Department of Applied Computer Science[†]
Vienna University of Technology, Austria

CD-TR 91/16, 1991.

Abstract

The paper considers two decision problems on hypergraphs, hypergraph saturation and recognition of the transversal hypergraph, and discusses their significance for several search problems in applied computer science. Hypergraph saturation, i.e., given a hypergraph $\mathcal{H}$, decide if every subset of vertices is contained in or contains some edge of $\mathcal{H}$, is shown to be co-**NP**-complete. A certain subproblem of hypergraph saturation, the saturation of simple hypergraphs, is shown to be computationally equivalent to transversal hypergraph recognition, i.e., given two hypergraphs $\mathcal{H}_1, \mathcal{H}_2$, decide if the sets in $\mathcal{H}_2$ are all the minimal transversals of $\mathcal{H}_1$. The complexity of the search problem related to the recognition of the transversal hypergraph, the computation of the transversal hypergraph, is an open problem. This task needs time exponential in the input size, but it is unknown whether an output-polynomial algorithm exists for this problem. For several important subcases, for instance if an upper or lower bound is imposed on the edge size or for acyclic hypergraphs, we present output-polynomial algorithms. Computing or recognizing the minimal transversals of a hypergraph is a frequent problem in practice, which is pointed out by identifying important applications in database theory, Boolean switching theory, logic, and AI, particularly in model-based diagnosis.

[*]Reformatted and spell corrected update of March 5, 1991.
A revised short version of this reports will appear in *SIAM Journal on Computing.*
[†]Mailing address: Paniglgasse16, 1040 Wien, Austria.
Internet e-mail: {`eiter,gottlob`}`@vexpert.dbai.tuwien.ac.at`

1 Introduction

Hypergraph theory [Ber89] is an important subfield of discrete mathematics with many relevant applications in both theoretical and applied computer science. In this paper, we study complexity issues of relevant computational problems on hypergraphs.

A hypergraph $\mathcal{H}$ is a family of subsets (edges) of a finite set of vertices. A hypergraph is simple if none of its edges is contained in any other of its edges. We say that a hypergraph is saturated if every subset of the vertex set is contained in an edge or contains an edge of the hypergraph.

The first problem we consider is the test whether a given hypergraph $\mathcal{H}$ is saturated. This decision problem will be referred to as HYPERGRAPH SATURATION. We will also deal with a restricted version of this problem where $\mathcal{H}$ is supposed to be a simple hypergraph. This subproblem is called SIMPLE HYPERGRAPH SATURATION.

The second main problem – and probably the more important one from the application viewpoint – concerns hypergraph transversals.

A transversal of a hypergraph $\mathcal{H}$ is a subset of the vertex set of $\mathcal{H}$ which intersects each edge of $\mathcal{H}$. Outside hypergraph theory, transversals are often called hitting sets. A transversal is minimal if it does not contain any transversal as proper subset. The set $Tr(\mathcal{H})$ of all minimal transversals of a hypergraph $\mathcal{H}$ is itself a hypergraph called the transversal hypergraph of $\mathcal{H}$.

Our second problem can thus be formulated as follows: given two hypergraphs $\mathcal{G}$ and $\mathcal{H}$, decide whether $\mathcal{G}$ is the transversal hypergraph of $\mathcal{H}$. This decision problem – referred to as TRANSVERSAL HYPERGRAPH – is closely related to the search problem of computing $Tr(\mathcal{H})$ for a given hypergraph $\mathcal{H}$.

The complexity of TRANSVERSAL HYPERGRAPH is to date an open issue. The problem is clearly in co-**NP** but there is no proof of co-**NP**-completeness. On the other hand, while the problem is polynomially solvable if $\mathcal{G}$ or $\mathcal{H}$ is a graph, no polynomial algorithm for the solution of the general version of TRANSVERSAL HYPERGRAPH is currently known. Similarly, it is an open problem, whether $Tr(\mathcal{H})$ can be computed in output-polynomial total time (i.e., in time polynomial in the combined sizes of the input and the output). This complexity problem was posed independently by several researchers [MR87, DT87, JYP88]. Note that the existence of an output-polynomial algorithm for computing $Tr(\mathcal{H})$ would imply the polynomial solvability of TRANSVERSAL HYPERGRAPH; vice-versa, if TRANSVERSAL HYPERGRAPH is co-**NP**-complete, then no output-polynomial algorithm for the computation of $Tr(\mathcal{H})$ is likely to exist.

This paper presents an extensive study of the above-mentioned complexity and computation problems. We derive some new complexity results and exhibit relationships to other relevant problems such as satisfiability and hypergraph two-colorability. Particular attention is paid to restricted versions of the general problem classes which are either polynomially solvable or complete for TRANSVERSAL HYPERGRAPH. We also show that the considered problems have important applications in database theory, switching theory, and artificial intelligence, from where it will become clear that computing the minimal transversals of a hypergraph is a central problem of computer science. In the rest of this section we outline the structure of the paper and highlight the most important results.

Our study of hypergraph problems starts in Section 2 with the definitions of the necessary basic concepts of hypergraph theory and a brief overview of results known so far.

In Section 3 we show that HYPERGRAPH SATURATION is co-**NP**-complete and relate this problem to the well-known problem of hypergraph two-colorability. SIMPLE HYPERGRAPH SATURATION turns out to be polynomially equivalent to the problem of two-coloring a hypergraph whose edges are mutually intersecting.

In Section 4 it is shown that SIMPLE HYPERGRAPH SATURATION is polynomially equivalent to TRANSVERSAL HYPERGRAPH. A restricted version of the latter problem is the question whether a given hypergraph is equal to its own transversal hypergraph, i.e., $Tr(\mathcal{H}) = \mathcal{H}$. This problem, which we call SELFTRANSVERSALITY, has attracted much interest by mathematicians [Lov73, Ben80, Ber89] but no complexity results have been derived so far. We show that SELFTRANSVERSALITY is complete for TRANSVERSAL HYPERGRAPH, i.e., it has the same complexity as the general problem.

In Section 5 we identify subclasses of the well-known SATISFIABILITY problem which are polynomially equivalent to TRANSVERSAL HYPERGRAPH. As it will turn out later, these classes are of particular interest, since, unlike other restrictions of SATISFIABILITY, they become polynomially decidable as soon as the cardinality of the clauses in the problem instances is bounded by a constant.

Section 6 is dedicated to the investigation of polynomially solvable subcases of the main problems. The most interesting results are briefly summarized as follows:

- HYPERGRAPH SATURATION becomes solvable in polynomial time if the cardinalities of the edges of the input hypergraph $\mathcal{H}$ differ by at most k, where k is a constant. In other words, if all edges of $\mathcal{H}$ have approximately the same size.
- TRANSVERSAL HYPERGRAPH is decidable in polynomial time if the cardinalities of the edges of one of the input hypergraphs, say $\mathcal{H}$, are bounded by a constant. For these cases we present an output-polynomial algorithm to compute $Tr(\mathcal{H})$. Note that this result immediately leads to a relevant generalization of well-known output-polynomial methods for computing the maximal independent sets of graphs [TIAS77, LLK80, JYP88]. (Indeed, the maximal independent sets of a graph or hypergraph are exactly the complements of its minimal transversals.)
- TRANSVERSAL HYPERGRAPH is decidable in polynomial time if one of the input

hypergraphs is acyclic. We present an output-polynomial algorithm for computing $Tr(\mathcal{H})$ for acyclic hypergraphs $\mathcal{H}$. The type of hypergraph acyclicity we consider is β-acyclicity as defined by Fagin in [Fag83]. Note that β-acyclicity is among the weakest types of acyclicity that have been defined for hypergraphs. Of course, our results and methods also hold if we replace β-acyclicity by any of the several stronger types of acyclicity which have been defined in the literature [Fag83, Ber89].

Several applications where the computation of hypergraph transversals plays a fundamental role are described in Sections 7 and 8.

The first and main application area we consider is the one of databases. It was recently advocated that Armstrong Relations can be used as a very profitable tool in database design [MR89, MR86]. In this context it is important to compute an Armstrong relation from a given set of functional dependencies and vice-versa. Both problems can be reduced to the problem of computing $Tr(\mathcal{H})$ for an appropriate hypergraph $\mathcal{H}$ if the relation scheme (resp. relation instance) is in Boyce-Codd Normal Form (BCNF). In this case, the problem of deciding whether a set of functional dependencies is represented by an Armstrong relation is complete for TRANSVERSAL HYPERGRAPH. We also show that the problem of deciding whether to a given Armstrong relation r (not necessarily in BCNF) and a given set K of keys for r there exists an additional key not already contained in K is polynomially equivalent to the complement of TRANSVERSAL HYPERGRAPH. Note that the additional key problem for sets of functional dependencies (instead of Armstrong relations) is solvable in polynomial time [LO78].

Another, completely different application in the field of databases concerns data access in distributed databases. In such systems, mutual exclusion of groups of sites, which is necessary for executing critical operations, can be realized by defining a priori a set of groups (quorums) that intersect each other [Lam78]. A group of sites can perform the critical operation only if it contains a quorum of this set, which is termed a coterie in [GMB85]. With respect to system reliability, the coterie should have a certain property called nondomination. The test for this property turns out as testing whether a hypergraph equals its transversal hypergraph, i.e., is an instance of SELFTRANSVERSALITY. We show that for a recent generalization of this approach to bi- and semicoteries, which model read and write operations by read quorums and write quorums [Fu90, IK90], the complexity of testing nondomination is not increased and remains complete for TRANSVERSAL HYPERGRAPH.

We also outline some important applications in Boolean switching theory and in Artificial Intelligence. As switching theory is concerned, it is easy to see that computing the prime implicants of a positive formula in CNF is equivalent to computing the minimal transversals of a hypergraph $\mathcal{H}$. Each edge E of $\mathcal{H}$ corresponds to the set of propositional variables occurring in some conjunct D of the formula and vice-versa. In AI, we focus on the new area of model-based diagnosis, where the efficient computation of hitting sets (= minimal transversals) has already been acknowledged as a fundamental problem [Rei87, dKW87].

Our results can be profitably used in all these application areas. In particular, our new algorithms for the output-polynomial computation of the minimal transversals in several relevant subcases of the general problem can be applied in these fields.

Section 9 concludes our paper. The different complexity results and their corresponding versions for several applications are succinctly presented in form of tables. We also state some open problems and give directions for further research.

2 Preliminaries and Previous Results

We start in this section with some basic definitions on hypergraphs.

Definition 2.1 *A hypergraph is a pair $(V, \mathcal{E})$ of a finite set $V = \{v_1, \ldots, v_n\}$ and a family $\mathcal{E}$ of subsets of V. The elements of V are called vertices, the elements of $\mathcal{E}$ edges. For every hypergraph $\mathcal{H}$ let $V(\mathcal{H})$ denote its vertex set and $\mathcal{E}(\mathcal{H})$ its edge set.*

We remark that in some definitions, e.g. [Ber89], the empty set $\emptyset$ is excluded as edge-set and is, moreover, not allowed to occur in the edge-set $\mathcal{E}$; furthermore, the edges have to establish $\bigcup_{E \in \mathcal{E}} E = V$. The above definition is adopted for convenience and practicality with respect to the problems considered in the later sections. Of course, if we remove those vertices from V which do not appear in any edge of $\mathcal{E}$, then $\mathcal{H} = (V, \mathcal{E})$ induces a unique hypergraph $(V_e(\mathcal{H}), \mathcal{E})$ in terms of [Ber89], where $V_e(\mathcal{H}) = \bigcup_{E \in \mathcal{E}} E$ denotes the *essential nodes* of $\mathcal{H}$. For notational convenience, we will identify a hypergraph with its edge-set and vice versa if there is no danger of ambiguity. Thus we write $E \in \mathcal{H}$ for $E \in E(\mathcal{H})$, $\mathcal{H}_1 \cup \mathcal{H}_2$ for $(V(\mathcal{H}_1) \cup V(\mathcal{H}_2), E(\mathcal{H}_1) \cup E(\mathcal{H}_2))$, $\min(\mathcal{H})$ for $(V, \min(E(\mathcal{H})))$ etc. Moreover, if not stated otherwise, it is supposed that hypergraphs have the same set of vertices.

A hypergraph $(V, \mathcal{E})$ is called *simple* if it satisfies $\forall X, Y \in \mathcal{E} : X \subseteq Y \Rightarrow X = Y$. Simple hypergraphs are also called *Sperner families* according to [Spe28], where it is proved that the cardinality of a simple hypergraph with n vertices is bounded by $\begin{pmatrix} n \\ [n/2] \end{pmatrix}$, which is asymptotic to $\left(\frac{2}{\pi}\right)^{1/2} 2^n n^{-1/2}$ [BDFS84]. For example, the set of minimal models for a finite set of propositional clauses or the minimal keys of a database relation form a Sperner system.

The *rank* $r(\mathcal{H})$ of hypergraph $\mathcal{H}$ is defined by $r(\mathcal{H}) = \max\{|E| \mid E \in \mathcal{H}\}$, and its *anti-rank* $ar(\mathcal{H})$ by $ar(\mathcal{H}) = \min\{|E| \mid E \in \mathcal{H}\}$. Trivially, $ar(\mathcal{H}) \leq r(\mathcal{H})$. If $ar(\mathcal{H}) = r(\mathcal{H})$ holds, $\mathcal{H}$ is called *uniform*. Clearly, every uniform hypergraph is simple. For example, every graph is a uniform hypergraph of rank 2. A hypergraph is called *intersecting* if no pair of its edges is disjoint, i.e., $\forall E_1, E_2 \in \mathcal{H} : E_1 \cap E_2 \neq \emptyset$. Singleton edges in a hypergraph are usually referred to as *loops*.

Example: Let $V = \{1,2,3,4\}$. Then $\mathcal{H} = \{\{1\},\{2,3\},\{1,3,4\}\}$ is a hypergraph on V. We note that $\mathcal{H}$ is not simple, and $ar(\mathcal{H}) = 1$, $r(\mathcal{H}) = 3$. $\mathcal{H}$ has no inessential nodes, and $\mathcal{H}$ is not intersecting.

Definition 2.2 *Let $\mathcal{H} = (V,\mathcal{E})$ be a hypergraph. Then* $\min(\mathcal{H})$ *denotes the set of minimal edges of $\mathcal{H}$ with respect to set inclusion, i.e.,* $\min(\mathcal{H}) = \{E \in \mathcal{E} \mid \nexists E' \in \mathcal{E} : E' \subset E\}$, *and* $\max(\mathcal{H})$ *denotes the set of maximal edges of $\mathcal{H}$ with respect to set inclusion, i.e.,* $\max(\mathcal{H}) = \{E \in \mathcal{E} \mid \nexists E' \in \mathcal{E} : E' \supset E\}$.

Clearly, for any hypergraph $\mathcal{H}$ on V, $\min(\mathcal{H})$ and $\max(\mathcal{H})$ are simple hypergraphs on V. For the hypergraph $\mathcal{H} = \{\{1\},\{2,3\},\{1,3,4\}\}$ of the last example, we have $\min(\mathcal{H}) = \{\{1\},\{2,3\}\}$ and $\max(\mathcal{H}) = \{\{2,3\},\{1,3,4\}\}$.

Definition 2.3 (Transversal) *Let $\mathcal{H} = (V,\mathcal{E})$ be a hypergraph. A set $T \subseteq V$ is called a transversal of $\mathcal{H}$ if it meets all edges of $\mathcal{H}$, i.e., $\forall E \in \mathcal{H} : T \cap E \neq \emptyset$. A transversal T is called minimal if no proper subset T' of T is a transversal.*

Example: We consider the hypergraph $\mathcal{H} = \{\{1\},\{2,3\},\{1,3,4\}\}$ on $V = \{1,2,3,4\}$ again. There are two minimal transversals of $\mathcal{H}$: $\{1,2\}$ and $\{1,3\}$.

Note that outside hypergraph theory a transversal is usually called a *hitting set*. For graphs, a transversal is usually called a *vertex cover*. The minimal vertex covers of a graph as well as the maximal independent sets are output-efficiently computable. An *independent set* of a graph $\mathcal{G}$ is a subset S of vertices that contains no edge of $\mathcal{G}$, and it is maximal if no proper superset of it has this property. Clearly, S is a maximal independent set of $\mathcal{G}$ iff $\overline{S}$ is a minimal vertex cover of $\mathcal{G}$. Finding the lexicographically first maximal independent set is LOG-SPACE-hard [Coo85] and hence a "hard" problem inside **P**; finding the lexicographically last maximal independent set, however, is **NP**-hard [JYP88]. The maximal independent sets of a graph are incrementally computable with polynomial delay between the outputs [TIAS77, LLK80], and a bit surprisingly, even if they have to be in lexicographic order [JYP88]. Moreover, fast parallel algorithms for maximal independent sets are known, cf. [KW86].

We remark that transversals are closely related to *edge covers*. An edge cover of a hypergraph is a subset of its edges such that their union yields the whole vertex set, and it is minimal if no proper subset is an edge cover. It is well-known that every minimal transversal of a hypergraph $\mathcal{H}$ is a minimal edge cover in the dual hypergraph of $\mathcal{H}$ and vice versa, where the dual hypergraph of $\mathcal{H}$ has for vertex set $E(\mathcal{H})$ and its edges are for every vertex $v \in V(\mathcal{H})$ the set of edges $E \in \mathcal{H}$ that contain v. The computation of minimal transversals and minimal edge covers is easily transformable into each other, thus results on one of these problems apply to the other (e.g., [Law66]).

Finding a minimal transversal of a hypergraph $\mathcal{H} = (V, E)$ is efficiently possible: If $\mathcal{E} = \emptyset$, then every subset $V' \subseteq V$ is a transversal of $\mathcal{E}$, hence $\emptyset$ is the one and only minimal transversal of $\mathcal{E}$. If $\mathcal{E} \neq \emptyset$, then $V_e(\mathcal{E}) = \{v_1, \dots, v_n\}$ is a transversal of $\mathcal{H}$. If we define

$$V_0 = V_e(\mathcal{H}), \quad V_{i+1} = \left\{ \begin{array}{ll} V_i & V_i - \{v_i\} \text{ is not a transversal} \\ V_i - \{v_i\} & V_i - \{v_i\} \text{ is a transversal} \end{array} \right. , \quad 0 \leq i < n,$$

then V_n is a minimal transversal of $\mathcal{H}$. Thus a minimal transversal of $\mathcal{H}$ can be found in time $O(|\mathcal{H}||V_e(\mathcal{H})|)$. To find a minimum cardinality transversal, however, is **NP**-hard, which is an immediate consequence of the well-known **NP**-complete MINIMUM HITTING SET problem [GJ79].

The family of all minimal transversals of $\mathcal{H}$ constitutes a simple hypergraph on V called the *transversal hypergraph* of $\mathcal{H}$, which we denote by $Tr(\mathcal{H})$. The following propositions capture important relations between a hypergraph and its transversal hypergraph.

Proposition 2.1 *Let* $\mathcal{H} = (V, \mathcal{E})$ *be a hypergraph. Then* $Tr(\mathcal{H})$ *is a simple hypergraph, and* $Tr(\mathcal{H}) = Tr(\min(\mathcal{H}))$.

Proposition 2.2 ([Ber89]) *Let* $\mathcal{G}$ *and* $\mathcal{H}$ *be two simple hypergraphs on vertices* V. *Then* $\mathcal{G} = Tr(\mathcal{H})$ *if and only if* $\mathcal{H} = Tr(\mathcal{G})$.

Corollary 2.3 *If* $\mathcal{G}$ *and* $\mathcal{H}$ *are simple hypergraphs, then* $Tr(\mathcal{G}) = Tr(\mathcal{H})$ *iff* $\mathcal{G} = \mathcal{H}$, *and* $Tr(Tr(\mathcal{H})) = \mathcal{H}$.

A simple algorithm to determine $Tr(H)$ is easily derived from the next proposition, for which we need some further notation.

Definition 2.4 *Let* $\mathcal{H}_1$, $\mathcal{H}_2$ *be two hypergraphs on vertices* V *that contain each at least one nonempty edge, i.e.,* $\mathcal{H}_1 \cup \mathcal{H}_2 \neq \emptyset$, *and* $\emptyset \notin \mathcal{H}_1 \cup \mathcal{H}_2$. *Then* $H_1 \vee H_2 = min\,\{H_1 \cup H_2 \mid H_1 \in \mathcal{H}_1, H_2 \in \mathcal{H}_2\}$.

Proposition 2.4 (cf. **[Ber89]**) *Let* $\mathcal{H}_1$, $\mathcal{H}_2$ *be two hypergraphs such that* $\mathcal{H}_1 \vee \mathcal{H}_2$ *is defined. Then* $Tr(\mathcal{H}_1 \cup \mathcal{H}_2) = Tr(\mathcal{H}_1) \vee Tr(\mathcal{H}_2)$.

Thus if $\mathcal{H} = \{E_1, \dots, E_m\}$, $m \geq 0$, defining a sequence $\mathcal{T}_0, \dots, \mathcal{T}_m$ by

$$\mathcal{T}_0 = \{\emptyset\}, \quad \mathcal{T}_{i+1} = \min\{T \cup \{e\} \mid T \in \mathcal{T}_i \wedge e \in E_i\}, \; 0 \leq i < m,$$

we have that $\mathcal{T}_j = Tr(\{E_1, \ldots, E_j\})$, $0 \leq j \leq m$, and thus $\mathcal{T}_m = Tr(H)$ (cf. [Mag59, Law66]).

It is easy to see that the computation of the transversal hypergraph is inherently exponential. For example, let $\mathcal{H}_n = (V_n, \mathcal{E}_n)$, $V_n = \{v_1, \ldots, v_{2n}\}$, $\mathcal{E}_n = \{\{v_{2i-1}, v_{2i}\} \mid 1 \leq i \leq n\}$. The transversal hypergraph for $\mathcal{H}_n$ is the uniform hypergraph $Tr(\mathcal{H}_n) = \{\{x_1, \ldots, x_n\} \mid x_i \in \{v_{2i}, v_{2i-1}\}, 1 \leq i \leq n\}$. The size of the transversal hypergraph $|Tr(\mathcal{H}_n)| = 2^n$ is exactly exponential in both $|V_n|$ and $|\mathcal{H}_n|$, thus for its computation space (hence also time) exponential in the input size is needed.

Since an efficient (polynomial time) algorithm for transversal hypergraph computation in the input size is impossible, an algorithm which works if not polynomial in the input size IS, then at least in polynomial time if the number of minimal transversals resp. the output size OS is taken into account is desirable. There are various possibilities to define appealing notions of output-polynomiality, e.g. total output-polynomiality, incrementally polynomiality, polynomial time delay computability [JYP88], or *P-enumerability* [Val79].

The most general of this concepts is *total output-polynomiality*, which requires that an algorithm works in time bounded by a polynomial in the input *and* the output size. Note that an output-polynomial total time algorithm may run for exponentially many steps in the input size without producing any output. The same holds for the notion of *P-enumerability*, where a search algorithm computes all solutions of a problem in time $p(IS)N$, where p is a polynomial in the input time and N the number of solutions [Val79]. To consider incremental computations, an algorithm meets the *incremental polynomial time* criterion if it outputs all solutions $s_1, \ldots, s_N$ to the respective problem one by one in such a manner that the time until the first output and the time between output of solutions s_i, s_{i+1}, $1 \leq i < N$, is bounded by a polynomial in the combined sizes of the input and all solutions $s_1, \ldots, s_i$, and stops in polynomial time after the last solution is output. A more restrictive criterion for incremental computation, related to P-enumerability, is output with *polynomial delay*. An algorithm with this property generates the solutions in a way such that the time until the first solution is output and between the output of consecutive solutions is bounded by a polynomial in the input size. Clearly, an algorithm with the polynomial delay output property P-enumerates the solutions.

For the above algorithm one can easily show that it stops not in output-polynomial total time. For example, consider the hypergraph $\mathcal{K}_n = \{\{v_i, v_j\} \mid 1 \leq i < j \leq n\}$ on vertices $\{v_1, \ldots, v_n\}$, i.e. the complete graph on n vertices, where $n \geq 4$ is even. Note that $\mathcal{K}_n$ is simple, hence $\mathcal{K}_n = \min(\mathcal{K}_n)$. Let $k = k(n) = |\mathcal{K}_n| = \frac{n(n-1)}{2}$, and assume that an ordering $E_1, \ldots, E_k$ of the edges of $\mathcal{K}_n$ satisfies $E_1 = \{1,2\}, E_2 = \{3,4\}, \ldots, E_{n/2} = \{n-1, n\}$. Computing the transversal hypergraph of $Tr(\mathcal{K}_n)$ by the computational sequence $\mathcal{T}_0, \ldots \mathcal{T}_m$, we find that $|\mathcal{T}_{n/2}| = 2^{n/2}$, because $\mathcal{T}_{n/2} = Tr(\{E_1, \ldots, E_{n/2}\})$ and $\{E_1, \ldots, E_{n/2}\}$ is just the hypergraph $\mathcal{H}_{n/2}$ from above. The transversal hypergraph of $\mathcal{K}_n$ is given by $\mathcal{T}_{k(n)} = \{\{1, \ldots, n\} - \{i\} \mid 1 \leq i \leq n\}$, thus $n = |Tr(\mathcal{K}_n)| = \Theta(|\mathcal{K}_n|^{1/2})$. Relating the number of edges of $\mathcal{T}_{n/2}$ to the m-th power, m arbitrarily fixed, of $\max\{|\mathcal{H}|, |Tr(\mathcal{H})|\} = |\mathcal{H}|$ we get

$$|\mathcal{T}_{n/2}| = 2^{\Theta(|\mathcal{K}_n|^{1/2})}, \text{ thus } |\mathcal{T}_{n/2}| > |\mathcal{K}_n|^m$$

for sufficiently large n. This implies that the size of $\mathcal{T}_{n/2}$ is not bounded by any polynomial in the input and output size. Due to its superpolynomial space requirement for intermediate results, the above algorithm is not efficient with respect to output-polynomiality.

Though there are several algorithms which involve, in a more or less obvious way, the computation of transversal hypergraphs, (e.g., [Mag59, Law66, Roy70, DT87, MR86, MR88a, Rei87]), unfortunately no output polynomial transversal hypergraph algorithm is known today; moreover, it is even an open question whether such an algorithm is likely to exist at all (cf. [DT87, JYP88, MR88b]). We will show in Section 4 that the complexity of the closely related decision problem TRANSVERSAL HYPERGRAPH, which is, given two hypergraphs $\mathcal{G}$ and $\mathcal{H}$, to decide if $\mathcal{G} = Tr(\mathcal{H})$, is of crucial importance to answer this question.

The second kind of hypergraph problem in our study is the complexity of hypergraph saturation, a covering problem for the powerset $\mathcal{P}(S)$ of a finite set S considered in [Thi86, BDK87]. We introduce some notation first:

Definition 2.5 *Let S be a finite set, and $X \subseteq S$. Then $Cov(X) = \{Y \subseteq S | Y \subseteq X \vee Y \supseteq X\}$. For any hypergraph $\mathcal{H} = (S, \mathcal{E})$, $Cov(\mathcal{H}) = \bigcup_{E \in \mathcal{E}} Cov(E)$.*

In words, $Cov(\mathcal{H})$ is the family of vertex sets $X \subseteq V(\mathcal{H})$ that are covered by an edge E of $\mathcal{H}$ as subset $X \subseteq E$) or superset $(X \supseteq E)$.

Definition 2.6 *Let $\mathcal{P}(V)$ denote the powerset 2^V of V. A hypergraph $\mathcal{H} = (V, \mathcal{E})$ is called saturated if and only if $Cov(\mathcal{H}) = \mathcal{P}(V)$.*

According to the definition, a hypergraph $\mathcal{H} = (V, \mathcal{E})$ is saturated iff every set of vertices $V' \subseteq V$ is contained in at least one of its edges or contains at least one of the edges of $\mathcal{H}$.

Examples: Consider the hypergraph $\mathcal{H} = \{\{1\}, \{2,3\}, \{1,3,4\}\}$ on $V = \{1,2,3,4\}$ again. Then, $Cov(\mathcal{H}) = \mathcal{P}(V) - \{2,4\}$, i.e., $\mathcal{H}$ covers all subsets of V but $\{2,4\}$, and hence $\mathcal{H}$ is not saturated. If we add $\{2,4\}$ to $\mathcal{H}$, the resulting hypergraph $\mathcal{H}' = \{\{1\}, \{2,3\}, \{1,3,4\}, \{2,4\}\}$ is saturated. Note that saturation of hypergraph $\mathcal{G}$ does not imply saturation of either $\min(\mathcal{G})$ or $\max(\mathcal{G})$. As an example consider $\mathcal{H}'$. Neither $\min(\mathcal{H}') = \{\{1\}, \{2,3\}, \{3,4\}\}$ nor $\max(\{\{2,3\}, \{1,3,4\}, \{2,4\}\})$ is saturated, because $\{2,4\}$ and $\{1,2\}$ respectively are not covered.

The complexity of checking if a hypergraph is saturated has, to our best knowledge, not been studied yet in the literature.

We conclude this section with some notational conventions on decision problems. We assume the reader familiar with the basic concepts of the theory of **NP**-completeness (cf. [GJ79]). If Π is a decision problem, the problem complementary to Π is denoted by co-Π. A problem Π' is called Π-hard if the polynomial time solvability to decision problem Π implies an polynomial time algorithm for Π'. A decision problem Π' is called Π-complete if Π' is polynomial time transformable into decision problem Π and vice versa. According to this, every **NP**-complete problem is SATISFIABILITY-complete. Problems that are solvable by some polynomial time algorithm are referred to as *tractable*, and those which are most likely not solvable in polynomial time, amongst them all **NP**-hard problems, as *intractable*.

3 Hypergraph saturation

3.1 Hypergraph saturation is co-NP-complete

Imagine you are a friend of Toni who manages a pizza restaurant that is famous for its rich variety of pizze.[1] The pizze differ in their topping, and accordingly every pizza has its special name. For example, on a *Pizza Margherita* there are tomatoes, onions, and some cheese, on a *Pizza al Tonno* tunny and onions, and on a *Pizza Provinciale* tomatoes, ham, corn, onions, mushrooms, salami, pepperonies, and cheese. Now Toni wants to enlarge his pizza offer by a new pizza, but the new pizza should be neither the "Grande" version of another pizza, i.e., have all the food of another pizza on it and some additional, nor the "Mini" version, i.e., all the food on it is also on some other pizza. Toni has been testing quite a number of food collections for the new pizza yet, but each of his compositions turned out to be a Grande or a Mini version of one of the many pizze. This makes Toni wondering if there is any food collection for a new pizza at all. Since you are an expert in computer science, he asks you to write a computer program for answering this question. Under the principle of *quot capita tot gustus*, which means that every collection of food is appropriate for a pizza, you find that Toni's problem is just the hypergraph saturation problem if the food available is considered as set V of vertices and the pizze already offered, described by food collections, are considered as edges $\mathcal{E}$ of a hypergraph. In a more precise formulation,

Problem:	HYPERGRAPH SATURATION (H-SAT)
Instance:	A hypergraph $\mathcal{H} = (V, \mathcal{E})$ on vertices $V = \{v_1, \ldots, v_n\}$
Question:	Is $\mathcal{H}$ saturated ?

A simple brute force algorithm would be to test all possible 2^n vertex sets subsequently until some collection is found that is not covered by $\mathcal{H}$; this collection would be a new pizza which satisfies the requirements. However, if $\mathcal{H}$ is saturated, the algorithm recognizes this only

[1]Problems on pizzas are studied elsewhere, cf. [GKP89], p. 4, p. 409.

after 2^n covering checks – too many if the number of food items on Toni's pizze is taken into account, as then the algorithm would take weeks to run. Thus a more subtle one than the naive exponential algorithm is needed to solve the problem efficiently. However, chances are very low that there is some algorithm substantially faster than the brute force algorithm, because the problem is co-**NP**-complete.

To show co-**NP**-completeness of H-SAT we proceed in two steps as usually done: We prove that H-SAT is in co-**NP** and present a polynomial time transformation of a co-**NP**-complete problem into H-SAT. In particular, we use the problem complementary to a subproblem of HYPERGRAPH 2-COLORABILITY [GJ79] for this purpose. Prior to the problem statement, we need some additional definitions.

Definition 3.1 (Complemented hypergraph) *Let V be a set and $V' \subseteq V$ a subset of it. Then $\overline{V'}$ denotes $V - V'$. Let $\mathcal{F}$ be a family of subsets of V. Then $\overline{\mathcal{F}} = \{\overline{F} \mid F \in \mathcal{F}\}$ is called the complemented family of $\mathcal{F}$. For every hypergraph $\mathcal{H}(V, \mathcal{E})$ we refer to $\overline{\mathcal{H}} = (V, \overline{\mathcal{E}})$ as the complemented hypergraph of $\mathcal{H}$.*

The following elementary relationships between a hypergraph and its complemented hypergraph are not difficult to see but nontheless important:

Proposition 3.1 *Let $\mathcal{H}$ be a hypergraph. Then $\overline{\overline{\mathcal{H}}} = \mathcal{H}$, and $\mathcal{H}$ is simple if and only if $\overline{\mathcal{H}}$ is simple.*

Proof: The first claim is obviously true. Assume $\mathcal{H}$ is not simple, i.e., there exists $X, Y \in E(\mathcal{H}), X \neq Y$, such that $X \subseteq Y$. But then, $\overline{Y} \subseteq \overline{X}$, hence $\overline{\mathcal{H}}$ is not simple. The *if* direction is trivial from the first claim. □

If $\mathcal{H} = \overline{\mathcal{H}}$ holds, $\mathcal{H}$ is called a *self complemented* hypergraph. 2-colorability of a hypergraph, also known as set splitting, is defined as follows:

Definition 3.2 *Let $\mathcal{H}$ be a hypergraph. A partitioning $A \cup B = V, A \cap B = \emptyset$ of V is a 2-coloring of $\mathcal{H}$ iff for all $E \in \mathcal{H}$ it holds that $E \not\subseteq A \wedge E \not\subseteq B$. $\mathcal{H}$ is 2-colorable iff there exists a 2-coloring for $\mathcal{H}$.*

In words, a hypergraph is 2-colorable if and only if there is an assignment of one of two "colors" to each vertex such that each edge has at least two colors. Formulated as a decision problem, we have

Problem:	HYPERGRAPH 2-COLORABILITY (HP2C)
Instance:	A hypergraph $\mathcal{H} = (V, \mathcal{E})$.
Question:	Is $\mathcal{H}$ 2-colorable ?

We prove that HP2C remains **NP**-complete for the following restriction:

Theorem 3.2 HYPERGRAPH 2-COLORABILITY *remains* **NP**-*complete even if the hypergraph* $\mathcal{H}$ *is self complemented, i.e.,* $\mathcal{H} = \overline{\mathcal{H}}$.

Proof: Consider the hypergraph $\mathcal{H}' = (V', \mathcal{E}')$, where $V' = V \cup \{e, f\}$ for some $e, f \notin V$, and $\mathcal{E}' = \mathcal{H} \cup \{\overline{E} \cup \{e, f\} \mid E \in \mathcal{H}\}$. It is clear that $\mathcal{H}'$ can be constructed from $\mathcal{H}$ in polynomial time; moreover, $\mathcal{H}'$ is self complemented. We claim that $\mathcal{H}$ is 2-colorable if and only if $\mathcal{H}'$ is 2-colorable. To prove the *only if* direction, let $A \cup \overline{A}$ be any 2-coloring of $\mathcal{H}$. If e,f are colored differently, only an edge of $\mathcal{H}'$ that is also an edge of $\mathcal{H}$ can prevent a 2-coloring of $\mathcal{H}'$. Hence $(A \cup \{e\}) \cup (\{f\} \cup V - A)$ is a 2-coloring of $\mathcal{H}'$. For the *if* direction, if $B \cup \overline{B}$ is any 2-coloring of $\mathcal{H}'$, it does not color any edge from $\mathcal{H}$ monochromatically because $\mathcal{H} \subseteq \mathcal{H}'$, and since no edge of $\mathcal{H}$ contains e or f, clearly $B - \{e, f\}, \overline{B} - \{e, f\}$ is a 2-coloring of $\mathcal{H}$. □

Now we are prepared to prove the following result:

Theorem 3.3 *Deciding if a hypergraph is saturated is co-***NP***-complete.*

Proof: Let $I = [\mathcal{H}]$ be an instance of H-SAT. Membership in co-**NP** is easy to show: Nondeterministically guess a subset $V' \subseteq V$ first and check if $V' \in Cov(\mathcal{H})$ holds thereafter, i.e., if V' is contained in an edge of $\mathcal{H}$ or contains an edge of $\mathcal{H}$. This can clearly be done in time polynomial in $|V|, |\mathcal{E}|$.

For the second part of the proof, let $J = [\mathcal{H}]$ be an instance of HP2C, $\mathcal{H} = (V, \mathcal{F})$. By Theorem 3.2 we may assume without loss of **NP**-completeness that $\mathcal{H}$ is self complemented. We claim that $\mathcal{H}$ is not 2-colorable if and only if $\mathcal{H}$ is saturated. If $\mathcal{H}$ is not 2-colorable, for every partitioning $A \cup \overline{A}$ of V there exists $F \in \mathcal{F}$ such that $F \subseteq A \vee F \subseteq \overline{A}$ holds. Since the self complementarity of $\mathcal{H}$ implies that $A \in Cov(\mathcal{H})$ iff $\overline{A} \in Cov(\mathcal{H})$, it follows that $A, \overline{A} \in Cov(\mathcal{H})$, thus $\mathcal{H}$ is saturated, what proves the *only if* direction. For the *if* direction, assume $\mathcal{H}$ is 2-colored by $A \cup \overline{A}$. Consider $F \in \mathcal{F}$. Since $A, \overline{A}$ is a 2-coloring, we have $A \not\supseteq F$. If $A \subseteq F$ holds, then also $\overline{A} \supseteq \overline{F}$, which by self complementarity of $\mathcal{H}$ contradicts to the assumption that $A, \overline{A}$ is a 2-coloring of $\mathcal{H}$. Thus $A \not\subseteq F$. We infer $A \notin Cov(\mathcal{H})$, which means that $\mathcal{H}$ is not saturated, and the claim is proved. □

Note that by the proof of Theorem 3.3 we immediately get the following Corollary:

Corollary 3.4 H-SAT *remains co-***NP***-complete for self complemented hypergraphs.*

This result can be strengthened to the following subcase:

Proposition 3.5 H-SAT *remains co-***NP***-complete even for self complemented hypergraphs where all edges have size* 3 *or* $n - 3$*, where* n *is the number of vertices of the hypergraph.*

Proof: HYPERGRAPH-2-COLORABILITY remains **NP**-complete even if no edge of $\mathcal{H}$ contains more than three vertices [GJ79]. Applying the transformation in the proof of Theorem 3.2 on min($\mathcal{H}$), we obtain a self complemented hypergraph $\mathcal{H}'$ which is saturated iff $\mathcal{H}$ is not 2-colorable, and every edge of $\mathcal{H}'$ contains either 3 vertices at most or $n-3$ vertices at least, $n = |V(\mathcal{H}')|$. Since $\binom{n}{3} < n^3, n \geq 1$, we can test in polynomial time if all vertex sets which contain at most three vertices are covered by $\mathcal{H}'$. If this holds, then also all vertex sets of size n-3 at least are covered by $\mathcal{H}'$, because a self complemented hypergraph $\mathcal{G}$ satisfies $\forall X \subseteq V(\mathcal{G}) : X \in Cov(\mathcal{G})$ *iff* $\overline{X} \in Cov(\mathcal{G})$. Now we may replace in $\mathcal{H}'$ every edge of size less than 3 by all its supersets of size 3 and every edge of size greater than n-3 by all its subsets of size n-3 without altering $Cov(\mathcal{H}')$. The hypergraph $\mathcal{H}''$ which results from this modification contains only edges of size 3 and n-3, and, moreover, $\mathcal{H}''$ is self complemented. Of course, this transformation can be done in polynomial time. □

Corollary 3.6 HP2C *remains* **NP**-*complete even for self complemented hypergraphs where all edges have size k or n-k, where n is the number of vertices of the hypergraph, for fixed* $k \geq 3$, *but is polynomial for* $k = 2$.

Proof: For $k \geq 3$ the claim follows immediately from Proposition 3.5 and the proof of Theorem 3.3. For $k = 2$, consider the family $\mathcal{G}$ of all edges with 2 vertices in hypergraph $\mathcal{H}$. $\mathcal{G}$ is a graph, and since $\mathcal{G} \subseteq \mathcal{H}$, $\mathcal{H}$ is 2-colorable only if $\mathcal{G}$ is 2-colorable. To prove that the converse also holds, consider any 2-coloring $A, \overline{A}$ of $\mathcal{G}$. If we extend this 2-coloring to an arbitrary partitioning $B, \overline{B}$ of $V(\mathcal{H})$ such that $A \subseteq B, \overline{A} \subseteq \overline{B}$, then neither B nor $\overline{B}$ is covered by $\mathcal{H}$. For if B is covered (as subset) by $\mathcal{H}$, then $\overline{B}$ is covered by an edge of size 2 in $\mathcal{H}$ as superset, contradicting that $A, \overline{A}$ is a 2-coloring of the graph of edges of size 2. Because $B, \overline{B} \notin Cov(\mathcal{H})$, $B, \overline{B}$ is a 2-coloring of $\mathcal{H}$. Thus 2-colorability of $\mathcal{G}$ implies 2-colorability of $\mathcal{H}$. Since 2-colorability for a graph can be decided in polynomial time [GJS76], 2-colorability of $\mathcal{H}$ is also polynomial. □

We note that H-SAT and HP2C remain intractable for a certain restriction on the chains on self complemented hypergraphs. If $\mathcal{H}$ is a hypergraph, a family $\mathcal{C} = \{C_1, \ldots, C_i\} \subseteq \mathcal{H}$ with property $C_j \subset C_{j+1}, 1 \leq j < i$, is called a *chain of length* i on $\mathcal{H}$. Then,

Proposition 3.7 H-SAT *remains co-***NP**-*complete if there is no chain on* $\mathcal{H}$ *of length* ≥ 3, *even if* $\mathcal{H}$ *is self complemented and all edges have size 3 or n-3.*

Proof: Follows immediately from Proposition 3.5 and Corollary 3.6. □

3.2 Simple hypergraph saturation

Toni has a brother Luigi, who also manages a big pizzeria. Luigi is proud that each of his pizze is "originale" which means that in his restaurant there are no Grande or Mini versions of pizzas. As he hears from the plans of his brother, Luigi also intends to enlarge his pizza offer by a new pizza, which has, of course, to be a pizza originale. It is clear that Luigi's problem is a restricted version of Toni's problem. Since all pizze are 'originali', in the corresponding hypergraph saturation problem the hypergraph of pizze is simple. Thus in terms of hypergraphs, we have

Problem:	SIMPLE HYPERGRAPH SATURATION (SIMPLE-H-SAT)
Instance:	A simple hypergraph $\mathcal{H} = (V, \mathcal{E})$ on vertices $V = \{v_1, \ldots, v_n\}$, i.e. $\forall H, H' \in \mathcal{E} : H \neq H' \Rightarrow H \not\subseteq H'$
Question:	Is $\mathcal{H}$ saturated ?

The question is if this restriction on the input makes the general saturation problem become tractable. Though SIMPLE-H-SAT is clearly in co-**NP**, there is no obvious evidence which suggests that the problem is also in **NP** or in **P**.

Note that a hypergraph $\mathcal{H}$ is saturated only if it has no inessential vertices, i.e., $V(\mathcal{H}) = V_e(\mathcal{H})$. Thus the problem reduces trivially to hypergraphs in terms of [Ber89]. A saturation test for two simple hypergraphs is easily reducible to an equivalent test for one hypergraph:

Definition 3.3 *Let $\mathcal{G}$, $\mathcal{H}$ be two simple hypergraphs on vertex set V, and let e, f be two new vertices not contained in V. Then $\mathcal{C}_{e,f}(\mathcal{G}, \mathcal{H}) = (V \cup \{e, f\}, \mathcal{E})$, where $\mathcal{E} = \{G \cup \{e\} \mid G \in \mathcal{G}\} \cup \{H \cup \{f\} \mid H \in \mathcal{H}\} \cup \{V, \{e, f\}\}$.*

Proposition 3.8 *Let $\mathcal{G}, \mathcal{H} \notin \{\{\emptyset\}, \{V\}\}$ be two simple hypergraphs on V, and let $e, f \notin V$. Then $\mathcal{C}_{e,f}(\mathcal{G}, \mathcal{H})$ is simple and it is saturated if and only if $\mathcal{G}$ and $\mathcal{H}$ are saturated.*

Proof: $\mathcal{C}_{e,f}$ is obviously simple. Assume that $\mathcal{H}$ is not saturated, i.e., there exists $X \subseteq V \cup \{e, f\}, X \notin Cov(\mathcal{C}_{e,f})$. X must contain exactly one of e or f. If it contains without loss of generality e, then X is not covered by the edges which contain e, hence $X - \{e\}$ is not covered by any edge of $\mathcal{G}$, and $\mathcal{G}$ is not saturated. Conversely, if without loss of generality $\mathcal{G}$ is not saturated, then for $X \subseteq V, X \notin Cov(\mathcal{G})$ we have that $X \cup \{e\}$ is not covered by $\mathcal{C}_{e,f}(\mathcal{G}, \mathcal{H})$, thus $\mathcal{C}_{e,f}$ is not saturated. □

We conclude this section with some observations on saturated hypergraphs.

Proposition 3.9 *Let $\mathcal{H}$ be a hypergraph. Then $\mathcal{H}$ is saturated if and only if $\overline{\mathcal{H}}$ is saturated.*

Proof: Indeed, if $X \subseteq V(\mathcal{H}), X \in Cov(\mathcal{H})$ then $\overline{X} \in Cov(\overline{\mathcal{H}})$. $\mathcal{H}$ is saturated iff $Cov(\mathcal{H}) = \mathcal{P}(V(\mathcal{H}))$, thus $Cov(\overline{\mathcal{H}}) = \overline{\mathcal{P}(V(\mathcal{H}))} = \mathcal{P}(V(\mathcal{H}))$, hence $\overline{\mathcal{H}}$ is saturated. The converse follows immediately from Proposition 3.1. □

Corollary 3.10 SIMPLE-H-SAT *is with respect to polynomiality equivalent to the subproblem where the instances are self complemented simple hypergraphs.*

Proof: It is easy to see that $\mathcal{C}_{e,f}(\mathcal{G},\overline{\mathcal{G}})$ is a simple self complemented hypergraph. Thus by Propositions 3.8 and 3.9 the Corollary follows immediately. $\square$

Corollary 3.11 HYPERGRAPH-2-COLORABILITY *for simple, self complemented hypergraphs is co-*SIMPLE-H-SAT*-complete.*

Proof: By Corollary 3.10 SIMPLE-H-SAT reduces to the saturation problem for simple self complemented hypergraphs. However, such a hypergraph $\mathcal{H}$ is not saturated if and only if there exists $X \subseteq V(\mathcal{H})$ such that $X \notin Cov(\mathcal{H}), \overline{X} \notin Cov(\mathcal{H})$, which means that $(X, \overline{X})$ is a 2-coloring of $\mathcal{H}$. $\square$

Note that Corollary 3.11 establishes that SIMPLE-H-SAT is a natural subcase of the **NP**-complete problem to decide if a self complemented hypergraph is 2-colorable, which results if we restrict the hypergraph to be simple.

4 Relating Problems on Hypergraph Transversals and Saturation

Problems involving hypergraph transversals appear in many practical applications (s. Sections 7 and 8), so their computational complexity is of major interest. Moreover, several computational problems on hypergraphs can be reformulated in terms of transversals; for example, finding a 2-coloring for a hypergraph $\mathcal{H}$ is equivalent to the task of searching for a transversal T of $\mathcal{H}$ such that $\overline{T}$ is also a transversal.

In this section we give first a characterization of saturated hypergraphs in terms of a transversal relation and then we will consider the problem of transversal recognition, where we will show that this problem is equivalent to SIMPLE-H-SAT with respect to polynomial time transformability . Moreover, we will investigate into necessary and sufficient criterions for simple hypergraph saturation and transversal hypergraph recognition, and we will show how the latter problem reduces to a very specific subcase of it.

We need some additional definitions first.

Definition 4.1 (Edge containment [Ber89]) *Let* $\mathcal{H}$, $\mathcal{G}$ *be two hypergraphs on vertices* V. *Then* $\mathcal{H} \geq \mathcal{G}$ *if and only if* $\forall H \in \mathcal{H} : \exists G \in \mathcal{G} : H \supseteq G$.[2] *Dual to this relation, define* $\mathcal{H} \leq \mathcal{G}$ *if and only if* $\forall H \in \mathcal{H} : \exists G \in \mathcal{G} : H \subseteq G$.

[2]In [Ber89] $\geq$ is denoted by $\prec$.

Note that $\mathcal{H} = \mathcal{G}$ iff $\mathcal{H} \geq \mathcal{G}$ and $\mathcal{G} \geq \mathcal{H}$ ($\mathcal{H} \leq \mathcal{G}$ and $\mathcal{G} \leq \mathcal{H}$, respectively), and $\mathcal{H} \geq \mathcal{G}$ does not imply $\mathcal{G} \leq \mathcal{H}$ and vice versa. We get the following characterization of saturated hypergraphs:

Theorem 4.1 *Let $\mathcal{H} \neq \emptyset$ be a hypergraph. Then $\mathcal{H}$ is saturated if and only if $Tr(\overline{\max(\mathcal{H})}) \geq \min(\mathcal{H})$.*

Proof: Assume $\mathcal{H}$ is saturated, and consider an edge T of $Tr(\overline{\max(\mathcal{H})})$. We have $T \in Cov(\mathcal{H})$, but since T is a transversal of $\overline{\max(\mathcal{H})}$, for all $E \in \mathcal{H}$ it holds that $T \not\subseteq E$. But then $E' \subseteq T$ for some $E' \in \mathcal{H}$ which implies $T \supseteq E''$ for some $E'' \in \min(\mathcal{H})$. Thus it follows that $Tr(\overline{\max(\mathcal{H})}) \geq \min(\mathcal{H})$. Assume $\mathcal{H}$ is not saturated, i.e., $X \subseteq V, X \notin Cov(\mathcal{H})$ exists. Since X is not covered by $\mathcal{H}$ as a subset, it is a transversal of $\overline{\max(\mathcal{H})}$, hence it contains at least one minimal transversal X' of $\overline{\max(\mathcal{H})}$. Since for every edge E of $\min(\mathcal{H})$ it holds that $E \not\subseteq X$, it follows that $E \not\subseteq X'$. But this means $Tr(\overline{\max(\mathcal{H})}) \not\geq \min(\mathcal{H})$, completing the proof. □

Corollary 4.2 *A hypergraph $\mathcal{H} \neq \emptyset$ is saturated if and only if $Tr(\overline{\mathcal{H}}) \geq \mathcal{H}$.*

Proof: As it is easily proved that $\overline{\max(\mathcal{H})} = \min(\overline{\mathcal{H}})$, this is immediate from Theorem 4.1. □

Corollary 4.3 *To test if $Tr(\mathcal{H}) \geq \mathcal{H}$ holds for a hypergraph $\mathcal{H}$ is co-**NP**-complete, even if $\mathcal{H}$ is self complemented and contains only edges of size 3 and n-3, where $n = |V(\mathcal{H})|$.*

Proof: If $\mathcal{H}$ is of the indicated form, then $\overline{\max(\mathcal{H})} = \min(\mathcal{H})$. Thus by Theorem 4.1 and Proposition 3.5 the proof is obvious. □

The main problem

Besides SIMPLE-H-SAT, which we introduced in the previous section, the recognition of the transversal hypergraph is the second issue of central concern in our study. We will show now that the complexity of this decision problem is of importance for the related search problem of computing the transversal hypergraph, and then we will demonstrate the equivalence of SIMPLE-H-SAT and transversal recognition by polynomial time transformations. Let us start with a precise problem statement:

Problem:	TRANSVERSAL HYPERGRAPH (TRANS-HYP)
Instance:	Two hypergraphs $\mathcal{G} = (V, \mathcal{E}_1)$, $\mathcal{H} = (V, \mathcal{E}_2)$ on a finite set V.
Question:	Does $\mathcal{H} = Tr(G)$ hold ?

Clearly the problem is in co-**NP**, but there is no trivial evidence that the problem is in **NP**, even less that it is in **P**. Obviously, the problem is computationally equivalent to the following subproblem:

Proposition 4.4 TRANSVERSAL HYPERGRAPH *is with respect to polynomial time transformation equivalent to the subproblem where both* $\mathcal{H}$ *and* $\mathcal{G}$ *are simple hypergraphs.*

Proof: $\mathcal{H} = Tr(\mathcal{G})$ if and only if $\mathcal{H} = Tr(\min(\mathcal{G}))$. Clearly $\mathcal{H}$, $\min(\mathcal{G})$ are simple hypergraphs, thus every TRANS-HYP instance is reducible to a TRANS-HYP instance on simple hypergraphs in polynomial time. □

The significance of TRANSVERSAL HYPERGRAPH on the related search problem, namely the computation of the transversal hypergraph, is captured by the following theorem :

Theorem 4.5 *If* TRANSVERSAL HYPERGRAPH *is not in* **P**, *then no output polynomial total time algorithm for the computation of the transversal hypergraph exists.*

Proof: We give an indirect proof. Assume that an output polynomial algorithm A for the computation of the transversal hypergraph exists. Let its runtime be bounded by a polynomial $p(I, N)$, where I denotes the input size and N is the number of edges of the transversal hypergraph. To solve TRANSVERSAL HYPERGRAPH for instance $I = [\mathcal{G}, \mathcal{H}]$ we can construct an algorithm A' which works as follows: The algorithm duplicates the steps of algorithm A to compute the transversal hypergraph of $\mathcal{G}$, but does this at most for $p(I, |\mathcal{H}|)$ steps. (The book-keeping between each simulation step can be done in polynomial time .) If algorithm A did not terminate within $p(I, |\mathcal{H}|)$ simulation steps, this implies $\mathcal{H} \neq Tr(\mathcal{G})$. Otherwise A' compares the hypergraph output by A with $\mathcal{H}$; if and only if these hypergraphs are identical then $\mathcal{H} = Tr(\mathcal{G})$ holds. It is not difficult to verify that the runtime of A' can be bounded by a polynomial in $|\mathcal{G}|$ and $|\mathcal{H}|$. But this means that A' is polynomial with respect to the input size, and the claim is proved. □

Thus the importance of the complexity of TRANSVERSAL HYPERGRAPH for applications in which transversal hypergraphs are computed is clear.

We will show now that TRANS-HYP reduces to a very specific subproblem which is closely related to the issue where the 'boundary' lies between the hypergraphs that are 2-colorable and those that are not. This issue is the more of interest as graph k-colorability (**NP**-complete for $k \geq 3$ reduces to hypergraph 2-colorability [Lov73]. Hypergraphs that are 2-uncolorable but become 2-colorable if any edge is removed (edge-*critical hypergraphs*) have been studied extensively in the literature, cf. [Lov68, Lov73, EL73, AL86, Tof74]. Lovász observed that such a critical hypergraph must have strict properties if it is intersecting ("strange hypergraphs", [Lov73]). The criticality criterion for hypergraphs in [Ben80], where a hypergraph $\mathcal{H}$ is called critical if it is not 2-colorable but every hypergraph $\mathcal{H}' \neq \mathcal{H}$

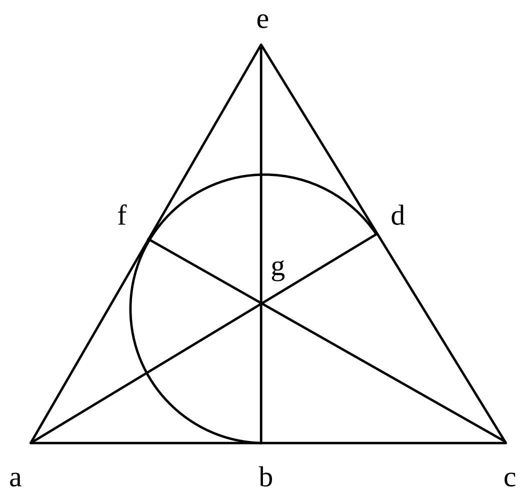

Figure 1: The finite projective plane $\mathcal{P}_7$

with $\mathcal{H}' \geq \mathcal{H}$ is 2-colorable, strengthens the above edge-based criticality to a vertex-based criticality, which is closely related to self-transversality.

Definition 4.2 *Let $\mathcal{H}$ be a hypergraph on vertices V, and let x be a new vertex not in V. $\mathcal{H}$ is* strongly minimally not 2-colorable *if it is not 2-colorable, but for every edge $E \in \mathcal{H}$ and every $v \in (V - E) \cup \{x\}$, the hypergraph $(\mathcal{H} - E) \cup (E \cup \{v\})$ is 2-colorable.*

In words, a hypergraph is strongly minimally not 2-colorable if it is not 2-colorable, but by adding any (possibly fresh) vertex to any edge it becomes 2-colorable. Note that adding a new vertex is equivalent to deleting an edge.

For example, the hypergraph $\mathcal{H} = \{\{1,2\},\{1,3\},\{2,3\}\}$ on vertex set $\{1,2,3\}$ is strongly minimally not 2-colorable, as well as the hypergraph $\mathcal{P}_7 = \{\{a,b,c\}, \{c,d,e\}, \{b,d,f\}, \{a,e,f\}, \{a,d,g\}, \{b,e,g\}, \{c,f,g\}\}$, on vertex set $V = \{a,\ldots,g\}$. The edges of $\mathcal{P}_7$ are the lines of the projective plane on seven points (see Figure 1). We have the following characterization of strongly minimally not 2-colorable hypergraphs:

Theorem 4.6 *A hypergraph $\mathcal{H}$ is strongly minimally not 2-colorable if and only if $\mathcal{H} = Tr(\mathcal{H})$.*

Proof: The claim holds if $V_e(\mathcal{H}) \leq 1$, i.e., $\mathcal{H}$ has at most one essential vertex. Otherwise, by a theorem in [Ben80], a hypergraph $\mathcal{H}$ with $V_e(\mathcal{H}) > 1$ satisfies $\mathcal{H} = Tr(\mathcal{H})$ if and only if it is not 2-colorable, but 3-colorable (i.e., appropriately colorable with 3 colors), and critical, i.e., every hypergraph $\mathcal{H}' \neq \mathcal{H}$, $\mathcal{H}' \geq \mathcal{H}$ is 2-colorable. It is easy to see that every strongly minimally not 2-colorable hypergraph $\mathcal{H}$ with more than one essential vertex is 3-colorable ($\mathcal{H}$ - E has a 2-coloring, which becomes a 3-coloring of $\mathcal{H}$ by assigning any vertex in E the third color), and $\mathcal{H}$ is by the definition of "$\geq$" ($\mathcal{F} \geq \mathcal{G} \iff \forall E \in \mathcal{F} : \exists E' \in \mathcal{G} : E \supseteq E'$) also critical. Thus the theorem is obvious. □

As promised above, we show that recognizing the transversal hypergraph reduces equivalently to the SELFTRANSVERSALITY problem, which is to decide if a hypergraph is self-transversal, i.e., it is its own transversal hypergraph.

Theorem 4.7 *To decide if a hypergraph $\mathcal{H}$ satisfies $Tr(\mathcal{H}) = \mathcal{H}$ is co-*SIMPLE-H-SAT-*complete.*

Proof: We show how to transform TRANS-HYP, which is co-SIMPLE-H-SAT-complete, in polynomial time into an equivalent instance of SELFTRANSVERSALITY. Let $\mathcal{G}$, $\mathcal{H}$ be two simple hypergraphs on vertices V, and let e, f be two new distinct vertices with $e, f \notin V$. Define a hypergraph $\mathcal{D} = \mathcal{D}_{e,f}(\mathcal{G}, \mathcal{H})$ on the vertex set $V' = V \cup \{e, f\}$ by $\mathcal{D} = \mathcal{G}^e \cup \mathcal{H}^f \cup \{\{e, f\}\}$, where $\mathcal{G}^e = \{G \cup \{e\} \mid G \in \mathcal{G}\}$ and $\mathcal{H}^f = \{H \cup \{f\} \mid H \in \mathcal{H}\}$. We claim that $\mathcal{G} = Tr(\mathcal{H})$ if and only if $\mathcal{D} = Tr(\mathcal{D})$. To prove this, consider $Tr(D)$. We have that $\{e, f\} \in Tr(\mathcal{D})$ and also $\{e, f\} \in \mathcal{D}$. Every other minimal transversal $T \in Tr(\mathcal{D}) - \{\{e, f\}\}$ contains either e or f, bot not both. Assume that T contains e, but not f. Since no edge in $\mathcal{H}^f$ contains e and since all other edges in $\mathcal{D} - \mathcal{H}^f$ contain e, we infer that $T - \{e\}$ must be a transversal of $\mathcal{H}$. Conversely, if T' is a transversal of $\mathcal{H}$, then $T' \cup \{e\}$ is easily shown to be a transversal of $\mathcal{D}$. This implies that the minimal transversals of $\mathcal{D}$ which contain e but not f are given by $Tr(\mathcal{D}_e) = \{E \cup e \mid E \in Tr(H)\}$. From this it is immediately verified that $\mathcal{G} = Tr(\mathcal{H})$ if and only if $\mathcal{G}^e = Tr(\mathcal{D})_e$. In an analogous way one can show that the family $\mathcal{D}_f$ of minimal transversals of $\mathcal{D}$ containing f but not e satisfies $\mathcal{D}_f = \mathcal{H}$ if and only if $\mathcal{H} = Tr(\mathcal{G})$. However, since by Proposition 2.2 we have for simple hypergraphs that $\mathcal{G} = Tr(\mathcal{H})$ if and only if $\mathcal{H} = Tr(\mathcal{G})$, and since $Tr(\mathcal{D}) = Tr(\mathcal{D})_e \cup Tr(\mathcal{D})_f \cup \{\{e, f\}\}$, we have that $\mathcal{G} = Tr(\mathcal{H})$ if and only if $\mathcal{D} = Tr(\mathcal{D})$, as was the claim. Since we may by Proposition 4.4 assume that the instances of TRANS-HYP are simple hypergraphs, the proof is complete. □

Corollary 4.8 *The recognition of strongly minimally not 2-colorable hypergraphs is co-*SIMPLE-H-SAT-*complete.*

Proof: Immediately from Theorem 4.6. □

Corollary 4.9 HYPERGRAPH-2-COLORABILITY *for intersecting hypergraphs is co-*SIMPLE-H-SAT-*complete.*

Proof: If $\mathcal{H}$ is intersecting, then also $\min(\mathcal{H})$. Every $E \in \mathcal{H}$ is a transversal of $\min(\mathcal{H})$. Thus without of loss of generality we may assume that $\mathcal{H}$ is simple, and clearly, that $\mathcal{H}$ contains no loops. A simple intersecting hypergraph $\mathcal{H}$ without loops satisfies $\mathcal{H} = Tr(\mathcal{H})$ if and only if it is not 2-colorable [Ber89]. Thus if $\mathcal{H}$ is intersecting, 2-colorability is reducible in polynomial time to self-transversality, which is co-SIMPLE-H-SAT-complete. □

We will see in Section 5 that SELFTRANSVERSALITY corresponds to very strict characterized sets of propositional clauses, and in Section 7.2 we will find an important application of self-transversal hypergraphs in distributed database systems.

Characterizations of the Transversal Hypergraph

Since one can test in polynomial time for every $X \subseteq V(\mathcal{H})$ whether X is a minimal transversal for a given hypergraph $\mathcal{H}$, we wonder if we can infer that $\mathcal{G} = Tr(\mathcal{H})$ if all edges of $\mathcal{H}$ are minimal transversals of $\mathcal{G}$ and all edges of $\mathcal{H}$ are minimal transversals of $\mathcal{G}$, since this can be checked in polynomial time. The answer is no:

Proposition 4.10 *Let $\mathcal{G}$, $\mathcal{H}$ be simple hypergraphs. Then $\mathcal{H} \subseteq Tr(\mathcal{G})$, $\mathcal{G} \subseteq Tr(\mathcal{H})$ does not imply $\mathcal{H} = Tr(\mathcal{G})$.*

Proof: We give an example. Let $V = \{a, \ldots, g\}$. Consider the hypergraphs $\mathcal{H} = \mathcal{P}_7 \cup \overline{\mathcal{P}_7} - \{\{a, b, d, e\}\}$ and $\mathcal{G} = \{E \cup \{e\} \mid E \in \mathcal{P}_7 - \{\{c, f, g\}\}, e \in \overline{E}\}$, where $\mathcal{P}_7$ is the 7-point plane from Figure 1 again. Then $\mathcal{H} \subseteq Tr(\mathcal{G})$, $\mathcal{G} \subseteq Tr(\mathcal{H})$ holds, but $\mathcal{G} \neq Tr(\mathcal{H})$ as the minimal transversal $\{c, f, g\}$ of $\mathcal{H}$ is not contained in $\mathcal{G}$. □

Note that if the criterion in Proposition 4.10 holds, the problem to decide if $\mathcal{H} = Tr(\mathcal{G})$ generalizes from deciding if there is an additional transversal as follows:

Proposition 4.11 *Let $\mathcal{G}$, $\mathcal{H}$ be simple hypergraphs on V satisfying $\mathcal{H} \subseteq Tr(\mathcal{G})$. Then $\mathcal{H} = Tr(\mathcal{G})$ if and only if there exists no (not necessarily minimal) transversal T of $\mathcal{G}$ such that $T \notin Cov(\mathcal{H})$.*

Proof: If a transversal T of $\mathcal{G}$ exists that is not covered by $\mathcal{H}$, it contains necessarily a minimal transversal of $\mathcal{G}$ not covered by $\mathcal{H}$. □

Below we will present an additional condition that makes the necessary condition from proposition 4.10 sufficient as well, but as we need Theorem 4.13 for a proof, we place this proposition at the end of this section.

For further characterizations of the transversal hypergraph of a simple hypergraph we introduce a special delete operator which removes from every edge of a hypergraph a vertex in all possible ways:

Definition 4.3 *Let V be a finite set, and let $\mathcal{HYP}(V) = 2^{2^V}$ denote the set of all hypergraphs on V. The mapping $\delta : \mathcal{HYP}(V) \to \mathcal{HYP}(V)$ is defined by*

$$\delta(\mathcal{H}) = (V, \{H - \{v\} \mid H \in \mathcal{H}, v \in H\}).$$

Example: Let $\mathcal{H}$ be the hypergraph $\mathcal{H} = \{\{1\},\{2,3\},\{1,3,4\}\}$ on vertices $V = \{1,2,3,4\}$. Then $\delta(\mathcal{H}) = \{\emptyset, \{2\}, \{3\}, \{1,3\}, \{1,4\}, \{3,4\}\}$. Now we note the following relationships:

Proposition 4.12 *Let $\mathcal{G}$, $\mathcal{H}$ be simple hypergraphs which contain both at least one non-empty edge. Then*

> **(1)** *$\mathcal{H} \subseteq Tr(\mathcal{G})$ implies $\mathcal{H} \leq \overline{\delta(\mathcal{G})}$, but the converse does not hold.*
>
> **(2)** *$\overline{\delta(\mathcal{H})} \geq Tr(\mathcal{H})$, i.e., every edge of $\overline{\delta(\mathcal{H})}$ is a transversal of $\mathcal{H}$.*
>
> **(3)** *$Tr(\mathcal{H}) \leq \overline{\delta(\mathcal{H})}$, i.e., every minimal transversal of $\mathcal{H}$ is contained in some edge of $\overline{\delta(\mathcal{H})}$.*

Proof:

(1) Consider $T \in \mathcal{H}$. Since T is a minimal transversal of $\mathcal{G}$, it follows that for every $v \in T$ there exists $G \in E(\mathcal{G})$ such that $(T - \{v\}) \cap G = \emptyset$, hence $T - \{v\} \subseteq \overline{G}$, but this means that $T \subseteq \overline{G} \cup \{v\} = \overline{G - \{v\}}$. From this it follows immediately that $\mathcal{H} \leq \overline{\delta(\mathcal{G})}$. That the converse does not hold is trivial, e.g., consider $\mathcal{H} = \{\emptyset\}$ and $\mathcal{G} = \{\{1\},\{2\}\}$ on vertex set $V = \{1,2\}$. Then $\mathcal{H} \leq \overline{\delta(\mathcal{G})} = \{\{1,2\}\}$, but $\mathcal{H} \neq Tr(\mathcal{G}) = \{\{1,2\}\}$.

(2) Consider $\overline{H}$, $H \in \mathcal{H}$. $\overline{H}$ is not a transversal of $\mathcal{H}$, but since $\mathcal{H}$ is simple, it is a transversal of the hypergraph with edges $\mathcal{H} - \{H\}$. Thus for every $v \in H$, $\overline{H} \cup \{v\}$ is a transversal of $\mathcal{H}$, hence $\overline{H} \cup \{v\} = \overline{H - \{v\}}$ contains a minimal transversal of $\mathcal{H}$. But then the claim follows immediately from the definition of the δ operator.

(3) For every minimal transversal T of $\mathcal{H}$ and every vertex $v \in T$ there exists an edge H of $\mathcal{H}$ such that $T \cap H = \{v\}$, hence $T \subseteq \overline{H} \cup \{v\}$, but $\overline{H} \cup \{v\} = \overline{H - \{v\}} \in \overline{\delta(\mathcal{H})}$.

□

Note that in the above proposition (1) states a necessary, but not sufficient condition for TRANSVERSAL HYPERGRAPH. With the δ-operator we have the following important characterization of saturated simple hypergraphs:

Theorem 4.13 *Let $\mathcal{H} \neq \emptyset$ be a simple hypergraph. Then $\mathcal{H}$ is saturated if and only if $Tr(\mathcal{H}) = \min(\overline{\delta(\mathcal{H})})$.*

Proof: We prove the *only if* direction first. Assume $\mathcal{H}$ is saturated, but $Tr(H) \neq \min(\overline{Comp\delta(\mathcal{H})})$, i.e., there exists a minimal transversal T of $\mathcal{H}$ such that $T \in Tr(H) - \min(\overline{\delta(\mathcal{H})})$. From Proposition 4.12(2) we infer that T is not a superset of any edge of $\min(\delta(\mathcal{H}))$, hence T is not a superset of any edge from $\overline{\delta(\mathcal{H})}$, which implies that $\overline{T}$ is a transversal of $\overline{\delta(\mathcal{H})}$. Now consider $E \in \overline{\delta(\mathcal{H})}$. Then $E = \overline{H} \cup \{e\}$ for some $H \in \mathcal{H}, e \in H$. Assume that $\overline{H} \subseteq T$ holds. Since for all $v \in H$ it holds that $\overline{H} \cup \{v\}$ is a transversal of $\mathcal{H}$ and T is a minimal transversal of $\mathcal{H}$, this implies that $T = \overline{H'} \cup v'$, for some $H' \in \mathcal{H}, v' \in H'$. It follows that $T = \overline{H} \cup \{v''\}, v'' \in H$, hence $T \in \overline{\delta(\mathcal{H})}$ and trivially T is superset of some edge in $\overline{\delta(\mathcal{H})}$, a contradiction. Thus $\overline{H} \not\subseteq T$, and we infer that $\overline{T}$ is a transversal of $\overline{\mathcal{H}}$. Since $\mathcal{H}$ is saturated, we have by Theorem 4.1 ($\min(\mathcal{H}) = \max(\mathcal{H}) = \mathcal{H}$) that $Tr(\overline{\mathcal{H}}) \geq \mathcal{H}$. Thus there exists $H \in \mathcal{H}$ such that $\overline{T} \supseteq H$. However, this implies that $T \subseteq \overline{H}$. But this means that T is not a transversal of $\mathcal{H}$, a contradiction.

To prove the *if* direction, assume $Tr(H) = \min \overline{\delta(\mathcal{H})}$, but $\mathcal{H}$ is not saturated. Consider $T \subseteq V$, $T \notin Cov(\mathcal{H})$. $\overline{T}$ is a transversal of $\mathcal{H}$, thus $\overline{T} \supseteq E$ for some $E \in \min \overline{\delta(\mathcal{H})}$. Since $E = \overline{H} \cup \{v\}$ for some $H \in \mathcal{H}, v \in H$ it follows that $\overline{T} \supseteq \overline{H}$, which implies $T \subseteq H$. But this is in contradiction to our assumption that $T \in Cov(\mathcal{H})$, and the claim is proved. $\Box$

We know of two other characterizations of saturated simple hypergraphs in [Thi86], which we state in a transcription from the context of relational database theory. These criterions are, however, in a certain sense less stringent than Theorem 4.13 because they do not describe the transversal hypergraph *explicitly*. Note that condition (2) is an immediate corollary to Theorem 4.1.

Proposition 4.14 *Let $\mathcal{H} \neq \emptyset$ be a simple hypergraph. The following propositions are equivalent:*

(1) *$\mathcal{H}$ is saturated*

(2) *$Tr(\overline{\mathcal{H}}) \geq \mathcal{H}$ [Thi86]*

(3) *$\overline{\mathcal{H}} \leq Tr(\mathcal{H})$ [Thi86]*

(4) *$Tr(\mathcal{H}) = \min(\overline{\delta(\mathcal{H})})$ (Theorem 4.13)*

Since given a hypergraph $\mathcal{H}$, the hypergraph $\mathcal{H}' = \min(\overline{\delta(\mathcal{H})})$ is computable in polynomial time, and it is also checkable in polynomial time whether all edges from $\mathcal{H}$ are minimal transversals of $\mathcal{H}'$ and vice versa, we wonder if this necessary condition is also sufficient to infer that $\mathcal{H}$ is saturated. Unfortunately, this does not hold:

Proposition 4.15 *Let $\mathcal{H} = (V, \mathcal{E})$ be a simple hypergraph, and let $\mathcal{H}' = \min(\overline{\delta(\mathcal{H})})$. Then $\mathcal{H} \subseteq Tr(\mathcal{H}')$, $\mathcal{H}' \subseteq Tr(\mathcal{H})$ does not imply that $\mathcal{H}$ is saturated. The same holds if $\mathcal{H}$ is self complemented.*

Proof: Consider the example in the proof of Proposition 4.10 again. It holds that $\mathcal{G} = \min(\overline{\delta(\mathcal{H})})$, $\mathcal{G} \subseteq Tr(\mathcal{H})$ and $\mathcal{H} \subseteq Tr(\mathcal{G})$, but $\mathcal{H}$ is not saturated because $\{a, b, d, e\} \notin Cov(\mathcal{H})$. However, if we transform $\mathcal{H}$ into the equivalent hypergraph $\mathcal{H}' = \mathcal{C}_{e,f}(\mathcal{H}, \overline{\mathcal{H}})$ (Corollary 3.10), then $\mathcal{H}'$ does not satisfy $\min \overline{\delta(\mathcal{H}')} \subseteq Tr(\mathcal{H}')$, leaving hope that the criterion is sufficient for simple self complemented hypergraphs. Unfortunately, it is not. Though the condition is established in many cases, the authors know of a self complemented simple hypergraph on 22 vertices with 308 edges which is a counter-example to this hypothesis. □

From Theorem 4.13 it is easy to see that every instance of SIMPLE-H-SAT is transformable into an instance of TRANSVERSAL HYPERGRAPH in polynomial time. We will show now that conversely every instance of TRANSVERSAL HYPERGRAPH can be transformed into an instance of SIMPLE-H-SAT, which establishes that SIMPLE-H-SAT and TRANSVERSAL HYPERGRAPH are with respect to polynomial time transformability computationally equivalent. In the proof we use the following simple, but helpful lemma:

Lemma 4.16 *Let $\mathcal{H}$ be a simple hypergraph. Then for all $E \in Tr(\mathcal{H})$ it holds that there is no $E' \in \mathcal{H}$ such that $E' \subseteq \overline{H}$.*

Proof: Indeed, if E is a minimal transversal of $\mathcal{H}$, then $E \cap E' \neq \emptyset$ for all $E' \in \mathcal{H}$, hence $E' - \overline{E} \neq \emptyset$, from which the claim follows immediately. □

With the lemma we prove the following theorem:

Theorem 4.17 *Let $\mathcal{G}_1$, $\mathcal{G}_2$ be two simple hypergraphs on vertices V. Define the hypergraph $\mathcal{H}$ by $\mathcal{H} = (V', \mathcal{E})$, where $V' = V \cup \{e, f\}$ for some $e, f \notin V$, and $\mathcal{E} = \mathcal{F}_1 \cup \mathcal{F}_2$, $\mathcal{F}_1 = \mathcal{G}_1$, $\mathcal{F}_2 = \{\overline{E} \cup \{e, f\} \mid E \in \mathcal{G}_2\}$. Then $\mathcal{G}_1 = Tr(\mathcal{G}_2)$ if and only if $\mathcal{H}$ is simple and saturated.*

Proof: For the *only if* direction, let $\mathcal{G}_1 = Tr(\mathcal{G}_2)$ hold. We first show that $\mathcal{H}$ is simple. Suppose $\mathcal{H}$ is not simple. Since $\mathcal{F}_1, \mathcal{F}_2$ constitute simple hypergraphs, there must exist $F_1 \in \mathcal{F}_1, F_2 \in \mathcal{F}_2$ such that $F_1 \subseteq F_2$ or $F_2 \subseteq F_1$. By Lemma 4.16 and the construction of $\mathcal{F}_1, \mathcal{F}_2$ it follows that $F_1 \subseteq F_2$ is impossible to hold. However, $F_2 \subseteq F_1$ is also impossible because $e \in F_2$ but $e \notin F_1$. Thus we have reached a contradiction, and $\mathcal{H}$ is simple. Now we show that $\mathcal{H}$ must also be saturated. Assume this is not the case, i.e., there exists $X \subseteq V', X \notin Cov(\mathcal{H})$. Since for all $F \in \mathcal{F}_2$ we have $X \not\subseteq F$, it follows that X is a transversal of $\overline{\mathcal{F}_2} = \overline{\overline{\mathcal{G}_2}} = \mathcal{G}_2$. Since $\mathcal{F}_1$ constitutes $Tr(\mathcal{G}_2)$, this implies that $E \subseteq X$ for some $E \in \mathcal{F}_1$, which means that $X \in Cov(\mathcal{H})$, a contradiction. Thus $\mathcal{H}$ is saturated, and the *only if* direction is proved.

To prove the *if* claim, assume that $\mathcal{H}$ is simple and saturated, but $\mathcal{G}_1 \neq Tr(\mathcal{G}_2)$. From the already proved only if direction, we can infer that $\mathcal{G}_1 \not\supset Tr(\mathcal{G}_2)$: If $\mathcal{G}_1 \supset Tr(\mathcal{G}_2)$ would hold, for a hypergraph $\mathcal{H}' \subset \mathcal{H}$ it would hold that $\mathcal{H}'$ is simple and saturated, which immediately would imply that $\mathcal{H}$ is not simple, a contradiction. Now since $\mathcal{G}_1 \neq Tr(\mathcal{G}_2)$ and $\mathcal{G}_1 \not\supset Tr(\mathcal{G}_2)$, there exists a $T \in Tr(\mathcal{G}_2) - \mathcal{G}_1$. Consider $T \cup \{e\}$. Because of T is a transversal of $\mathcal{G}_2 = \overline{\mathcal{F}_2}$, it is clear that there is no $F \in \mathcal{F}_2$ such that $T \cup \{e\} \subseteq F$. There is no $F \in \mathcal{F}_2$ such that $T \cup \{e\} \supseteq F$ since $f \in F, f \notin T$, and there is no $F \in \mathcal{F}_1$ with property $T \cup \{e\} \subseteq F$ because $e \notin F$. Since $\mathcal{H}$ is saturated, this implies that there is some $F \in \mathcal{F}_1$ such that $F \subseteq T \cup \{e\}$, from which clearly $F \subseteq T$ follows. Because $\mathcal{H}$ is simple, one can easily show that F must be a transversal of $\overline{\mathcal{F}}_2 = \mathcal{G}_2$. But because of T is a minimal transversal, it follows that $F = T$, hence $T \in \mathcal{G}_1$, in contradiction to the assumption. $\Box$

Now it is easy to prove the above stated computational relationship between SIMPLE-H-SAT and TRANSVERSAL HYPERGRAPH:

Theorem 4.18 SIMPLE-H-SAT *and* TRANSVERSAL HYPERGRAPH *are with respect to polynomial time transformation computationally equivalent, i.e.,* TRANSVERSAL HYPERGRAPH *is* SIMPLE-H-SAT-*complete.*

Proof: Theorem 4.13 enables immediately a transformation of any instance of SIMPLE-H-SAT into an equivalent instance of TRANSVERSAL HYPERGRAPH in polynomial time. Conversely, every instance of TRANSVERSAL HYPERGRAPH is polytime transformable according to Theorem 4.17 into an appropriate instance of SIMPLE-H-SAT. $\Box$

We conclude this section with a strengthening of the necessary condition for TRANSVERSAL HYPERGRAPH in Proposition 4.10 such that the criterion becomes sufficient. We make use of the following simple lemma:

Lemma 4.19 *Let $\mathcal{H}_1, \mathcal{H}_2$, and $\mathcal{H}_3$ be three simple hypergraphs on V which contain each at least one nonempty edge with property $\mathcal{H}_1 \cap \mathcal{H}_3 = \emptyset$, $\mathcal{H}_2 \cap \mathcal{H}_3 = \emptyset$, and $\mathcal{H}_1 \cup \mathcal{H}_3$, $\mathcal{H}_2 \cup \mathcal{H}_3$ constitute simple hypergraphs on V. Then $\mathcal{H}_1 \vee \mathcal{H}_3 = \mathcal{H}_2 \vee \mathcal{H}_3$ implies that $\mathcal{H}_1 = \mathcal{H}_2$.*

Proof: Easily proved by Corollary 2.3 and Proposition 2.4 $\Box$

Theorem 4.20 *Let $\mathcal{D}$, $\mathcal{G}$ be simple hypergraphs, $\mathcal{D} \neq \emptyset$, and let $\mathcal{D} \subseteq Tr(\mathcal{G})$ hold. Then $\mathcal{D} = Tr(\mathcal{G})$ if and only if for all saturated simple hypergraphs $\mathcal{H}$ with property $\mathcal{G} \subseteq \mathcal{H}$ it holds that $\overline{\delta(\mathcal{H})} \geq \mathcal{D}$. Moreover, if $\overline{\delta(\mathcal{H})} \not\geq \mathcal{D}$ for some $\mathcal{H}$ with this property, then $\overline{\delta(\mathcal{H}')} \not\geq \mathcal{D}$ for all $\mathcal{H}'$ with that property.*

Proof: If $\mathcal{D} = \{\emptyset\}$, then $\mathcal{G} = \emptyset$ and vice versa. In these cases the claim holds. We assume in the following that both $\mathcal{D}$ and $\mathcal{G}$ contain at least one nonempty edge. Consider any saturated simple hypergraph $\mathcal{H}$ that contains $\mathcal{G}$ as a proper subset. Then there is a simple hypergraph $\mathcal{G}'$ with at least one nonempty edge such that $\mathcal{G} \cap \mathcal{G}' = \emptyset$ and $\mathcal{H} = \mathcal{G} \cup \mathcal{G}'$. By Proposition 2.4 we have that $Tr(H) = Tr(\mathcal{G}) \vee Tr(\mathcal{G}')$. As $\mathcal{D} \subseteq Tr(\mathcal{G})$, there exists a simple hypergraph $\mathcal{F}$, $\mathcal{F} \cap \mathcal{D} = \emptyset$, such that $Tr(\mathcal{G}) = \mathcal{D} \cup \mathcal{F}$. Since $\mathcal{H}$ is saturated, by Theorem 4.13 it holds that $Tr(\mathcal{H}) = \min(\overline{\delta(\mathcal{H})})$; hence the following equations hold (all hypergraphs except possibly $\mathcal{F}$ contain at least one nonempty edge, thus $\vee$ is defined in all occurrences):

$$Tr(\mathcal{H}) = Tr(\mathcal{G}) \vee Tr(\mathcal{G}') = (\mathcal{D} \cup \mathcal{F}) \vee Tr(\mathcal{G}') = \min(\overline{\delta(\mathcal{H})}) \tag{1}$$

We show first that the first part of the claim holds if $\mathcal{H} = \mathcal{G}$. In this case we have $Tr(\mathcal{H}) = Tr(\mathcal{G}) \supseteq \mathcal{D}$, and since $\mathcal{H}$ is saturated, we get $\mathcal{D} \subseteq \min(\overline{\delta(\mathcal{H})})$, thus clearly $\min(\overline{\delta(\mathcal{H})}) \geq \mathcal{D}$. In the following assume that $\mathcal{H} \neq \mathcal{G}$. We now prove the *only if* claim. Assume $\mathcal{D} = Tr(\mathcal{G})$, i.e., $\mathcal{F}$ has empty edge-set. Then from Equation 1 we infer that $(\mathcal{D} \cup \mathcal{F}) \vee Tr(\mathcal{G}') = \mathcal{D} \vee Tr(\mathcal{G}') = \min(\overline{\delta(\mathcal{H})})$, hence clearly $\overline{\delta(\mathcal{H})} \geq \mathcal{D}$. For the *if* direction, assume $\mathcal{D} \neq Tr(\mathcal{G})$, i.e., $\mathcal{F}$ has a nonempty edge-set. Now suppose that $\overline{\delta(\mathcal{H})} \geq \mathcal{D}$ holds. This implies that every minimal transversal of $\mathcal{H}$ contains some edge of $\mathcal{D}$. Thus from Equation 1 we get

$$Tr(\mathcal{H}) = (\mathcal{D} \cup \mathcal{F}) \vee \mathcal{G}' = \mathcal{D} \vee \mathcal{G}' = Tr(\mathcal{G}) \vee \mathcal{G}'.$$

By Lemma 4.19 we can infer from $\mathcal{D} \vee \mathcal{G}' = Tr(\mathcal{G}) \vee \mathcal{G}'$ that $\mathcal{D} = Tr(\mathcal{G})$. By this contradiction the first part of the claim is proved. To prove the remainder of the claim, if some saturated simple hypergraph $\mathcal{H} \supseteq \mathcal{G}$ satisfies $\overline{\delta(\mathcal{H})} \not\geq \mathcal{D}$, then $\mathcal{D} \neq Tr(\mathcal{G})$, and since we made no special assumption on $\mathcal{H}$ in the proof of the *if* direction above, this property holds by generalization for any simple saturated hypergraph $\mathcal{H}'$ which contains all edges of $\mathcal{G}$. This completes the proof. □

Corollary 4.21 *Let $\mathcal{G}$, $\mathcal{H}$ be two simple hypergraphs on vertices V satisfying $\mathcal{H} \neq \emptyset$, $\mathcal{H} \subseteq Tr(\mathcal{G})$, and let $\mathcal{F}$ denote the hypergraph on V with edge-set $\mathcal{G} \cup (2^V - Cov(\mathcal{G}))$. Then $\mathcal{H} = Tr(\mathcal{G})$ if and only if $\overline{\delta(\mathcal{F})} \geq \mathcal{H}$.*

Proof: By Lemma 4.20 $\mathcal{H} = Tr(\mathcal{G})$ if and only if every saturated simple hypergraph $\mathcal{H}$ that contains all the edges of $\mathcal{G}$ satisfies $\overline{\delta(\mathcal{F})} \geq \mathcal{H}$. Since for every $X \subseteq V, X \notin Cov(\mathcal{G})$ clearly some saturated simple hypergraph exists that contains all edges of $\mathcal{G}$ and X, the *only if* direction must hold. Conversely, $\mathcal{F}$ contains at least one simple saturated hypergraph $\mathcal{H}'$ that contains all edges of $\mathcal{G}$. Thus if $\overline{\delta(\mathcal{F})} \geq \mathcal{H}$ holds, then clearly $\overline{\delta(\mathcal{H}')} \geq \mathcal{H}$, hence by Lemma 4.20 $\mathcal{H} = Tr(\mathcal{G})$. □

Theorem 4.22 *Let $\mathcal{G} \neq \emptyset, \mathcal{H} \neq \emptyset$ be two simple hypergraphs satisfying $\mathcal{G} \subseteq Tr(\mathcal{H})$ and $\mathcal{H} \subseteq Tr(\mathcal{G})$. Then $\mathcal{G} = Tr(\mathcal{H})$ if and only if there exist simple saturated hypergraphs $\mathcal{G}_1, \mathcal{H}_1$ such that $\mathcal{G} \subseteq \mathcal{G}_1, \mathcal{H} \subseteq \mathcal{H}_1$ and $\overline{\delta(\mathcal{H}_1)} \geq \mathcal{G}$ and $\overline{\delta(\mathcal{G}_1)} \geq \mathcal{H}$.*

Proof: Follows immediately from Theorem 4.20 and Corollary 2.3. □

5 Relating Satisfiability and Hypergraph Saturation

The satisfiability of logical formulae, especially sets of clauses, has special interest in complexity theory. There is a strong interconnection between satisfiability of a set of clauses and 2-colorability of hypergraphs, and these problems are decidable by similar resolution methods, cf. [LT85]. We have shown in Sections 3 and 4 that SIMPLE-H-SAT is equivalent to SIMPLE-H-SAT for self complemented hypergraphs and to SELFTRANSVERSALITY, which are both essentially 2-coloring problems for simple hypergraphs. Utilizing this relationship we point out in this section three very simply described subclasses of MONOTONE SAT (MSAT), a subclass of the well known SATISFIABILITY problem, which correspond to SIMPLE-H-SAT and TRANSVERSAL HYPERGRAPH.

The SATISFIABILITY problem (SAT) is to decide if a given set $\mathcal{C} = \{C_1, \ldots, C_m\}$ of clauses containing (possibly negated) logical variables $X = \{x_1, \ldots, x_n\}$ is satisfied by any truth value assignment. In MSAT, which is still **NP**-complete, all clauses of $\mathcal{C}$ are restricted to be either positive or negative, i.e., must not contain any negative or positive literal, respectively.

Definition 5.1 *Let $Y \subseteq X$ be a subset of a set X of propositional variables. Then $Y^{\neg} = \{\neg y \mid y \in Y\}$.*

Definition 5.2 *Let $\mathcal{C}$ be an instance of* MSAT *with variables X. Then $\mathcal{C}^+$ and $\mathcal{C}^-$ denote the hypergraphs of the families of variable sets in the positive and the negative clauses of $\mathcal{C}$ respectively, i.e., $\mathcal{C}^+ = \{Y \subseteq X \mid Y \in \mathcal{C}\}$ and $\mathcal{C}^- = \{Y \subseteq X \mid Y^{\neg} \in \mathcal{C}\}$.*

Example: Let $X = \{x_1, \ldots, x_4\}$, and $\mathcal{C} = \{\{x_1\}, \{x_2, x_4\}, \{\neg x_2, \neg x_4\}, \{\neg x_3\}\}$. Then $\mathcal{C}^+ = \{\{x_1\}, \{x_2, x_4\}\}$, $\mathcal{C}^- = \{\{x_2, x_4\}, \{x_3\}\}$.

We note the following lemma (Recall that for hypergraphs $\mathcal{G}$, $\mathcal{H}$ the relation $\mathcal{G} \geq \mathcal{H}$ holds iff every edge of $\mathcal{G}$ contains some edge of $\mathcal{H}$):

Lemma 5.1 *Let $\mathcal{C}$ be an instance of* MSAT. *Then $\mathcal{C}$ is unsatisfiable iff $Tr(\mathcal{C}^+) \geq \mathcal{C}^-$.*

Proof: We may identify every truth value assignment with the set $X_t \subseteq X$ of variables which are assigned the value true. Assume $Tr(\mathcal{C}^+) \geq \mathcal{C}^-$. Every candidate X_t for a satisfying truth value assignment must be a transversal of $\mathcal{C}^+$. Since every minimal transversal of $\mathcal{C}^+$ contains some edge from $\mathcal{C}^-$, it follows that there exists $Y \in \mathcal{C}^-$ such that $Y \subseteq X_t$. Since all variables in X_t are set true, no literal in Y is true, and X_t does not satisfy $\mathcal{C}$. It follows that $\mathcal{C}$ is unsatisfiable. Conversely, if $\mathcal{C}$ is unsatisfiable then every truth value assignment that satisfies the positive clauses must falsify at least one negative clause. Thus it is clear that every minimal transversal of $\mathcal{C}^+$ must contain all variables in a negative clause, i.e., $Tr(\mathcal{C}^+) \geq \mathcal{C}^-$. $\square$

The first subclass of MSAT we consider are the MSAT instances with the property that every pair of a positive clause and a negative clause resolves, i.e., there is a variable that occurs unnegated in the positive and negated in the negative clause. Satisfiability of this subclass of MSAT turns out to be computationally equivalent to SIMPLE-H-SAT:

Theorem 5.2 *Let $\mathcal{C}$ be an instance of* MSAT *such that every pair of a positive and a negative clause resolves, i.e., $\forall C \in \mathcal{C}^+, \forall C' \in \mathcal{C}^- : C \cap C' \neq \emptyset$. To decide if $\mathcal{C}$ is satisfiable is co-*SIMPLE-H-SAT*-complete.*[3]

Proof: By Lemma 5.1 $\mathcal{C}$ is satisfiable only if $Tr(\mathcal{C}^+) \not\geq \mathcal{C}^-$ which is equivalent to $Tr(\mathcal{C}^+) \not\geq \min(\mathcal{C}^-)$. Since every pair of a negative and a positive clause resolves, every $C \in \min(\mathcal{C}^-)$ is a transversal of $\mathcal{C}^+$. If C is not a minimal transversal of $\mathcal{C}^+$, then $\mathcal{C}$ is certainly satisfiable. However, if every C in $\min(\mathcal{C}^-)$ is a minimal transversal of $\mathcal{C}^+$ (i.e., $\min(\mathcal{C}^-) \geq \mathcal{C}^+$, checkable in polynomial time), then $Tr(\mathcal{C}^+) \not\geq \min(\mathcal{C})$ iff $Tr(\mathcal{C}^+) \neq \min(\mathcal{C}^-)$, because for hypergraphs $\mathcal{A}, \mathcal{B}$ we have that $\mathcal{A} \neq \mathcal{B}$ iff $\mathcal{A} \not\geq \mathcal{B} \vee \mathcal{B} \not\geq \mathcal{A}$. Thus the problem is polynomial time transformable into co-TRANS-HYP, which is co-SIMPLE-H-SAT-complete. Conversely, without loss of generality we may assume that in instance $I = [\mathcal{G}, \mathcal{H}]$ of co-TRANS-HYP every edge in $\mathcal{G}$ is a transversal of $\mathcal{H}$ et vice versa. For if not, then certainly $\mathcal{H} \neq Tr(\mathcal{G})$ holds. Under this assumption, $\mathcal{G} \neq Tr(\mathcal{D})$ holds if and only if the MSAT instance $\mathcal{C} = \{G \mid G \in \mathcal{G}\} \cup \{D^{\neg} \mid D \in \mathcal{D}\}$, which is of the indicated form, is satisfiable. Clearly, the necessary transformation can be done in polynomial time. $\Box$

From co-SIMPLE-H-SAT-completeness of hypergraph saturation for simple self complemented hypergraphs, we get that satisfiability of the following subclass of MSAT, which is a subset of the class with pairwise resolving positive and negative clauses, is also as hard to decide as the saturation of simple hypergraphs:

Corollary 5.3 *Let $\mathcal{C}$ be an instance of* MSAT *such that $\mathcal{C}^+$ is a simple, self-complemented hypergraph and that $\mathcal{C}^- = \{C^{\neg} \cup \{\neg x\} \mid C \in \mathcal{C}^+ \wedge x \in \overline{C}\}$. To decide satisfiability of $\mathcal{C}$ is co-*SIMPLE-H-SAT*-complete.*

Proof: Every MSAT instance $\mathcal{C}$ of the indicated form is in the subclass of MSAT described in Theorem 5.2. Since to decide saturation of a simple self-complemented hypergraph is co-SIMPLE-H-SAT-complete, it is possible to transform $\mathcal{C}$ in polynomial time into an equivalent instance of hypergraph saturation for simple self complemented hypergraphs. Applying Theorem 4.13, a simple self-complemented hypergraph $\mathcal{H}$ is saturated if and only if $Tr(\mathcal{H}) = \mathcal{H}'$, where $\mathcal{H}' = \min(\{E \cup \{e\} | E \in \mathcal{H}, e \in \overline{E}\})$. Every edge of $\mathcal{H}'$ is a transversal of $\mathcal{H}$ and vice versa, thus if we apply the transformation of co-TRANS-HYP into MSAT of Theorem 5.2,

[3]D.A. Plaisted observed (personal communication) that if every every pair of distinct clauses resolves then $\mathcal{C} = \{C_1, \ldots, C_m\}$ is satisfiable if and only if $\sum_{i=1}^{m} 2^{-|C_i|} \neq 1$ (provided that no clause is subsumed by some other clause and that no variable occurs both negated and unnegated in the same clause).

a clause set of the indicated form results that is satisfiable if and only if the simple self complemented hypergraph is not saturated. □

By Theorem 4.7 we are able to show that a second simply described subclass of MSAT is co-SIMPLE-H-SAT-complete. The class is another subclass of MSAT restricted to instances with pairwise resolving positive and negative clauses, which results if additionally positive clauses and negative clauses respectively are pairwise interconnected.

Corollary 5.4 *Let $\mathcal{C}$ be any instance of* MSAT *on Variables $X = \{x_1, \ldots, x_n\}$ with properties $\mathcal{C}^+ = \mathcal{C}^-$ and $\mathcal{C}^+$ is intersecting. To decide satisfiability of the class of clause sets similar to $\mathcal{C}$ is co-*SIMPLE-H-SAT*-complete.*

Proof: It is easy to see that $\mathcal{C}$ is in the subclass of MSAT where every pair of a positive and a negative clause resolves, thus by Theorem 5.2 $\mathcal{C}$ can be polynomially transformed into an equivalent co-SIMPLE-H-SAT instance. Conversely, co-SELFTRANSVERSALITY is co-SIMPLE-H-SAT-complete, and since it is easily checked that this problem remains co-SIMPLE-H-SAT-complete if every edge of the hypergraph is a transversal, we can by the transformation in the proof of Theorem 5.2 transform such an hypergraph into an equivalent instance of MSAT which is of the indicated form. □

Consequently, the restriction of MSAT to clauses with at most three literals (M3SAT, which is **NP**-complete, as well-known [GJ79]), is polynomial time decidable if positive and negative clauses pairwise resolve. We may select any clauses $C_1 \in \mathcal{C}^+$, $C_2 \in \mathcal{C}^-$ from clause set $\mathcal{C}$, and for every of the up to $2^3 2^3 = 64$ possible truth value assignments to the literals in C_1 and C_2 the clause set is reducible to an instance of 2SAT, since in every clause the truth value of at most two literals remains open thereafter. However, 2SAT is solvable in polynomial time (cf. [GJ79]). In Section 6 it appears that this result generalizes from M3SAT to MkSAT, which is MSAT restricted to instances where the size of the clauses is bounded by some constant k: If the positive and negative clauses pairwise resolve, then satisfiability is decidable in polynomial time (rf. Corollary 6.6).

6 Hypergraph Saturation and Transversal Recognition resp. Computation: Polynomial Subcases

Although we do not know whether SIMPLE-H-SAT and TRANSVERSAL HYPERGRAPH are intractable in their general problem statement, in practical occurrences there are (natural) restrictions on the instances of a problem such that the problem reduces to a subproblem of the general problem. Since a subproblem might be computationally easier, there is hope to get efficient algorithms for it even if the more general problem is is intractable. For example, in the pizza baker's problem it seems quite reasonable to assume that all pizze have approximately the same number of food items; in this case, however, the problem becomes polynomial time solvable for both Toni and Luigi. Narrowing the open problems-"frontier" between the intractable general problem and tractable subcases [GJ79], we identify some important subcases of SIMPLE-H-SAT and TRANSVERSAL HYPERGRAPH which are in **P**, among them the recognition of the minimal transversals of a hypergraph with bounded edge size and hypergraph saturation if the difference between rank and antirank is bounded. In particular, we present algorithms for output-efficient computation of the transversal hypergraph, if the edge-size is small (probably the most important of the restrictions treated for practice) and if its large, or if the hypergraph is acyclic.

6.1 Restrictions on the number of edges

Let us first turn to the restriction of the number of edges of a hypergraph. Clearly,

Lemma 6.1 *Let $\mathcal{H}$ be a hypergraph. If $|\mathcal{H}| \leq k$ for some constant k, then $Tr(\mathcal{H})$ is computable in polynomial time.*

Proof: Let $\mathcal{H} = \{E_1, \ldots, E_k\}$, $k \geq 1$. Then $\mathcal{F} = \{\{e_1\} \cup \cdots \cup \{e_k\} \mid e_i \in E_i, 1 \leq i \leq k\}$ contains all minimal transversals of $\mathcal{H}$, moreover $Tr(\mathcal{H}) = \min(\mathcal{F})$. If k is a constant, then $|\mathcal{F}| \leq |V|^k$. Thus the computation of $\min(\mathcal{F})$ can be done in polynomial time in this case, from which the claim follows immediately. $\Box$

Proposition 6.2 *If the instances $I = [\mathcal{G}, \mathcal{D}]$ of* TRANSVERSAL HYPERGRAPH *satisfy that either $|\mathcal{G}|$ or $|\mathcal{D}|$ is bounded by some constant k, then the problem is solvable in polynomial time.*

Proof: Follows immediately from Lemma 6.1. $\Box$

H-SAT is also efficiently decidable if the number of edges is limited:

Proposition 6.3 H-SAT *is solvable in polynomial time if the instances $I = [\mathcal{H}]$ satisfy $|\mathcal{H}| \leq k$ for some constant k.*

Proof: Follows immediately from Theorem 4.1 and Lemma 6.1. $\Box$

6.2 Restrictions on the edge size

Let us consider restrictions on the size of the edges of a hypergraph. A straightforward restriction of this kind which is important in practice is a constant upper bound. Fortunately, the transversal hypergraph is output-efficiently computable in this case, which implies that the minimal transversals of such a hypergraph can be recognized in polynomial time and that also saturation of a simple hypergraph of this form is efficiently decidable.

Definition 6.1 *Let $\mathcal{H}$ be a hypergraph on vertices V, and let $x \in V$. The degree $d(x,\mathcal{H})$ of vertex x in hypergraph $\mathcal{H}$ is defined as $d(x,\mathcal{H}) = |\{E \in \mathcal{H} \mid x \in E\}|$.*

In the proof we apply a result on k-conformal hypergraphs by Berge and Duchet:

Lemma 6.4 ([Ber89], p. 58, Corollary 1) *Let $\mathcal{G}$ be a simple hypergraph, and let $k \geq 2$ be an integer. Then $r(Tr(\mathcal{G})) \leq k$ if and only if for all subhypergraphs $\mathcal{G}' \subseteq \mathcal{G}$ with $k+1$ edges there exists an edge $E \in \mathcal{G}$ contained in the set $\{v \in V(\mathcal{G}) \mid d(v,\mathcal{G}') > 1\}$.*

Theorem 6.5 *The transversal hypergraph $Tr(\mathcal{H})$ of a hypergraph $\mathcal{H}$ is computable in output-polynomial time (more precise, incremental polynomial time computable) if the size of the edges of $\mathcal{H}$ is bounded by some constant k, i.e., $r(\mathcal{H}) \leq k$.*

Proof: We sketch an algorithm $Trans(\mathcal{H})$ for transversal computation which has the desired property. By Lemma 6.4, the following characterization of the transversal hypergraph of a simple hypergraph $\mathcal{H}$ on vertices V is easily verified, if $k = r(\mathcal{H}) \geq 2$:

$$\mathcal{G} = Tr(\mathcal{H}) \iff \mathcal{G} \subseteq Tr(\mathcal{H}) \wedge P \wedge Q , \tag{2}$$

where

$$\begin{aligned} P &\equiv \nexists T \subseteq V, |T| \leq k : T \in Tr(\mathcal{G}) \wedge T \notin \mathcal{H} \\ Q &\equiv \nexists \mathcal{G}' \subseteq \mathcal{G}, |\mathcal{G}'| = k+1 : \forall E \in \mathcal{G} : E \not\subseteq \{x \in V \mid d(x,\mathcal{G}') > 1\}. \end{aligned}$$

To prove Equivalence 2, if $\mathcal{G} = Tr(\mathcal{H})$ and $\mathcal{H}$ is simple, then $\mathcal{H} = Tr(\mathcal{G})$, and the *only if* direction is obvious by Lemma 6.4. For the *if* direction, if $\mathcal{H}$ is simple, $\mathcal{G} \subseteq Tr(\mathcal{H})$, and P holds, then $\mathcal{H} \subseteq Tr(\mathcal{G})$ is true. Since by validity of Q and Lemma 6.4 again all minimal transversals of $\mathcal{G}$ have no more than k vertices, it follows $\mathcal{H} = Tr(\mathcal{G})$.

Equivalence 2 also holds for simple hypergraphs $\mathcal{H} \neq \emptyset$ with $r(\mathcal{H}) \leq 1$. We omit the simple proof of this for brevity.

We will now derive a method for incremental transversal computation from Equivalence 2. If $\mathcal{G} \subseteq Tr(\mathcal{H})$ and $\mathcal{G} \neq Tr(\mathcal{H})$, then P or Q is false (provided $\mathcal{H}$ is simple an nonempty, what is assumed). In this case, an additional transveral T_1 of $\mathcal{H}$ not in $\mathcal{G}$ can be found as follows.

First, assume that P is false, i.e. there exists $T \in V, |T| \leq k$ such that $T \in Tr(\mathcal{G}) \wedge T \notin \mathcal{H}$. Because $\mathcal{G} \subseteq Tr(\mathcal{H})$, every edge $E \in \mathcal{H}$ is a minimal transversal of $\mathcal{G}$, thus E can not contain T as subset; consequently, $\overline{T}$ is a transversal of $\mathcal{H}$ which is not covered by $\mathcal{G}$: If for some $G \in \mathcal{G}$, $G \subseteq \overline{T}$, then T would not be a transversal of $\mathcal{G}$, and if $G \supset \overline{T}$, then $\overline{T}$ would be a smaller transversal of $\mathcal{H}$ than G, contradicting $\mathcal{G} \subseteq Tr(\mathcal{H})$. Thus $\overline{T}$ is not covered by $\mathcal{G}$, and every transversal of $\mathcal{H}$ contained in $\overline{T}$ is not covered by $\mathcal{G}$. It follows that $\overline{T}$ contains an additional minimal transversal T_1 of $\mathcal{H}$ that is not in $\mathcal{G}$.

Now let Q be false, i.e. there exists $\mathcal{G}' \subseteq \mathcal{G}, |\mathcal{G}'| = k+1$ such that for all $E \in \mathcal{G}$ we have $E \not\subseteq \{x \in V \mid d(x, \mathcal{G}') > 1\}$. Note that this is only possible if $|\mathcal{G}| > r(\mathcal{H})$, which implies that $r(\mathcal{H}) \geq 2$. By Lemma 6.4, it follows that $r(Tr(\mathcal{G})) > k$, thus $\mathcal{H} \neq Tr(\mathcal{G})$. We claim that $T = \{x \in V \mid d(x, \mathcal{G}') > 1\}$ is a transversal of $\mathcal{H}$ not covered by $\mathcal{G}$, i.e., $T \notin Cov(\mathcal{G})$. Note that if T is a transversal of $\mathcal{H}$, then it cannot be covered by $\mathcal{G}$, as no edge of $\mathcal{G}$ is contained in T, and, on the other hand, T cannot be properly contained in any edge $E \in \mathcal{G}$ as E is a minimal transversal of $\mathcal{H}$. Thus to prove our claim, we only have to show that T is a transveral of $\mathcal{H}$. Assuming that T is not a transversal of $\mathcal{H}$, there is some $E \in \mathcal{H}$ such that $E \cap T = \emptyset$, i.e., $E \subseteq \overline{T}$. However, this is impossible to hold. Since T contains no edge of $\mathcal{G}$, $\overline{T}$ is a transversal of $\mathcal{G}$, and thus also a transversal of $\mathcal{G}'$. From the definition of T, each vertex of $\overline{T}$ appears in one edge of $\mathcal{G}'$ at most. Since $\mathcal{G}'$ has $k+1$ edges, it follows that no transversal T' of $\mathcal{G}'$ with $T' \subseteq \overline{T}$ can have less than $k+1$ vertices. Because every $E \in \mathcal{H}$ is a transversal of $\mathcal{G}$' (this is immediate from $\mathcal{G}' \subseteq \mathcal{G} \subseteq Tr(\mathcal{H})$) and $|E| \leq k$ (recall that $k = r(\mathcal{H})$), it follows that $E \not\subseteq \overline{T}$, and the desired contradiction is reached. Thus T is a transversal of $\mathcal{H}$, and our claim is proved. Clearly, T contains a minimal transversal T_1 of $\mathcal{H}$ not covered by $\mathcal{G}$.

Utilizing this result, the following algorithm outputs the transversal hypergraph of a simple nonempty hypergraph $\mathcal{H}$:

```
$Trans(\mathcal{H})$:

  $k \leftarrow r(\mathcal{H})$;  $\mathcal{G} \leftarrow \emptyset$;
  loop $\{\mathcal{G} \subseteq Tr(\mathcal{H})\}$
    if $(\exists T \in V, |T| \leq k :\ T \in Tr(\mathcal{G}) \wedge T \notin \mathcal{H})$ then
      minimize $\overline{T}$ to a minimal transversal $T_1$ of $\mathcal{H}$;
    else $\{\ \mathcal{H} \subseteq Tr(\mathcal{G}), \mathcal{G} \subseteq Tr(\mathcal{H})\}$
      if $(\exists \mathcal{G}' \subseteq \mathcal{G}, |\mathcal{G}'| = k+1 :\ \forall E \in \mathcal{G} : E \not\subseteq \{x \in V \mid d(x,\mathcal{G}') > 1\})$
      then
        Minimize $\{x \in V \mid d(x,\mathcal{G}') > 1\}$ to a minimal transversal $T_1$ of $\mathcal{H}$;
      else $\{\ \mathcal{G} = Tr(\mathcal{H})\}$
        exit loop;
      fi;
    fi;
    output$(T_1)$; $\mathcal{G} \leftarrow \mathcal{G} \cup \{T_1\}$;
  endloop;
```

The correctness of the algorithm can easily be proved from Equivalence 2 and the described additional transversal determination; note that $\mathcal{G} \subseteq Tr(\mathcal{H})$ is a loop invariant.

To prove the theorem, assume that $r(\mathcal{H}) \leq k$. If $\mathcal{H}$ is not simple, we compute $\min(\mathcal{H})$ in polynomial time and proceed with that hypergraph. Because k is a constant, the conditions of both if statements can be checked in time polynomial in the size of $\mathcal{G}$, $\mathcal{H}$ and V. For the outer if, there are roughly $|V|^k$ vertex sets to check, and for the inner if, there are less than $|\mathcal{G}|^{k+1}$ vertex sets (each of them computable in polynomial time) to test. In both cases, all tests are clearly polynomial time executable, and T_1 is polytime computable (e.g., by the method described in Section 2.) Thus an execution of the loop body needs time polynomial in the size of $\mathcal{G}, \mathcal{H}$ and V. Every pass of the loop but the last yields an additional minimal transversal T_1 of $\mathcal{H}$ for output. (For an algorithm that stops immediately after output of the last transversal, the output of every transversal may be delayed until the next transversal is computed.) Hence it is clear that $Tr(H)$ is computable in incremental polynomial time. □

Corollary 6.6 TRANS-HYP *is polynomial time decidable if the edge-size of one of the two hypergraphs is bounded by some constant integer k.*

Note that Theorem 6.5 generalizes the well-known result that the maximal independent sets of a graph are output-efficiently computable to hypergraphs with edge-size bounded by a constant. (Recall that a maximal independent set is the complement of a minimal transversal). Thus the question in [JYP88], whether there is an output-polynomial total time algorithm for the maximal independent sets computation of hypergraphs, is answered "yes" for hypergraphs of bounded edge-size.

By Theorems 4.13, 6.5 it is clear that SIMPLE-H-SAT is decidable in polynomial time if the edge size is bounded by some constant. However, for polynomiality it is not necessary that such a hypergraph is simple, as we will be shown in Corollary 6.12.

After considering a constant lower bound on the edge size, let us turn attention to the symmetrical restriction of a constant lower bound on the edge size. In this case even the computation of the transversal hypergraph can even be done in input-polynomial time.

Theorem 6.7 *Let $\mathcal{H}$ be a hypergraph on V with property $ar(\mathcal{H}) \geq |V| - k$, for some constant integer $k \geq 0$. Then $Tr(\mathcal{H})$ is computable in input-polynomial time.*

Proof: Without loss of generality we assume that $\mathcal{H}$ is simple. We know from Proposition 4.12(3) that $Tr(\mathcal{H}) \leq \overline{\delta(\mathcal{H})}$. Since $r(\overline{\delta(\mathcal{H})}) \leq k+1$, only the vertex sets of up to $k+1$ elements are candidates for minimal transversals, and there are no more then roughly $|V|^{k+1}$ of them. Thus all minimal transversals of $\mathcal{H}$ can be found in time polynomially in the input. $\square$

Corollary 6.8 TRANS-HYP *and* SIMPLE-H-SAT *are polynomially decidable if for input hypergraph $\mathcal{H}$ on V, $ar(\mathcal{H}) \geq |V| - k$, for some constant integer $k \geq 0$.*

Proof: By Theorems 6.7 and 4.13. $\square$

As the third restriction, let us consider hypergraphs where the edges differ in their size by at most some given constant. We start with a simple, but important lemma on saturation.

Lemma 6.9 *Let $\mathcal{H}$ be a hypergraph on V, and $c_0 = ar(\mathcal{H})$, $c_1 = r(\mathcal{H})$. Then $\mathcal{H}$ is not saturated if and only if there exists a set $V' \subseteq V$, $c_0 \leq |V'| \leq c_1$, such that $V' \notin Cov(\mathcal{H})$.*

Proof: Assume there exists $V' \subseteq V$, such that $V' \notin Cov(\mathcal{H})$. If $|V'| < c_0$, we may add any vertices $v_1, \ldots, v_{c_0 - |V'|}$ from $V - V'$ to V', and $V' \cup \{v_1, \ldots, v_{c_0-|V'|}\} \notin Cov(\mathcal{H})$ will hold. If $|V'| > c_1$, we may remove any vertices $v_1, \ldots, v_{c_1-|V'|}$ from V' without establishing $V' - \{v_1, \ldots, v_{c_1-|V'|}\} \in Cov(\mathcal{H})$. Thus the *only if* direction holds. The *if* claim is trivial. $\square$

The last lemma enables a proof of the following theorem:

Theorem 6.10 *If the instances $I = [\mathcal{H}]$ of H-SAT satisfy: $r(\mathcal{H}) - ar(\mathcal{H}) \leq k$ for some constant nonnegative integer k, then H-SAT is decidable in polynomial time.*

Proof: We give a simple polynomial time algorithm for this restriction. Let $|V(\mathcal{H})| = n$. It is not difficult to find a method to generate all subsets of S of fixed size i subsequently such that the computation of the next set is bounded by a polynomial $p(n)$. (For a simple method, ref. [DGH74].) Let in such a method $First(i)$ denote the starting set of cardinality i and $NextSet(X)$ the successor of set X (if there is none, ∇). Consider the following algorithm:

```
$Saturated(\mathcal{H})$:
  $c_0 \leftarrow ar(\mathcal{H}); c_1 \leftarrow r(\mathcal{H}); i \leftarrow c_0;$
  while $i \leq c_1$ do
      $X \leftarrow First(i)$;
      while $X \neq \nabla$ do
          if $X \notin Cov(\mathcal{H})$ then return(false); fi;
          $X \leftarrow$ NextSet($X$);
      endwhile;
      $i \leftarrow i + 1$;
  endwhile;
  return(true);
```

By Lemma 6.9 we know that if $\mathcal{H}$ is not saturated, there exists $X \subseteq V, X \notin Cov(\mathcal{F})$ such that $c_0 \leq |X| \leq c_1$, hence the correctness of the algorithm is clear. Now let us analyze its complexity, where we assume that the time to check "$X \notin Cov(\mathcal{H})$" is bounded by some polynomial $q(|\mathcal{E}|, n)$ (indeed, it is simple to do this in time $O(n|\mathcal{E}|)$).

Consider $Y \in \mathcal{E}$. Clearly, $c_0 \leq |Y| \leq c_1$. The exact number of subsets of V with cardinality between c_0 and c_1 that are covered by Y is given by

$$C(Y, c_0, c_1) = \sum_{i=0}^{c_1-|Y|} \binom{n-|Y|}{i} + \sum_{i=0}^{|Y|-c_0} \binom{|Y|}{|Y|-i} - 1, \tag{3}$$

where the first term at the right hand side of the equation equals the number of those sets covered as supersets, and the second the number of sets covered as subsets. Let $c = c_1 - c_0 + 1$. Then

$$\begin{aligned} C(Y, c_0, c_1) &\leq \sum_{i=0}^{c_1-c_0} \left[\binom{n-c_0}{i} + \binom{c_1}{i} \right] \leq 2c \max\left\{ \binom{n-c_0}{i}, \binom{c_1}{i} \mid 0 \leq i \leq c-1 \right\} \\ &\leq 2cn^{c-1}. \end{aligned}$$

Thus the number of relevant sets covered by Y is bounded by $O(cn^{c-1})$. By virtue of the additional k-restriction and $k = c$ we infer that $C(Y, c_0, c_1)$ is bounded by some polynomial $r(n)$. Thus $\mathcal{E}$ covers *at most* $r(n)|\mathcal{E}|$ relevant subsets of S.

This implies that if $\mathcal{E}$ is not saturated, the first set not covered by $\mathcal{F}$ is found within time $O(r(n)|\mathcal{E}|(p(n) + q(|\mathcal{E}|, n)))$, hence in time polynomial in the input.

Now assume $\mathcal{E}$ is saturated. Then the algorithm checks all subsets of V of cardinality $c_0, c_0 + 1, \ldots, c_1$. Let $f(n, c_0, c_1) = \binom{n}{c_0} + \binom{n}{c_0+1} + \cdots + \binom{n}{c_1}$ denote their number.

Since $\mathcal{H}$ covers $r(n)|\mathcal{E}|$ of them at most, it is clear that $f(n, c_0, c_1) \leq r(n)|\mathcal{E}|$, which ensures that the algorithm terminates within time $\mathrm{O}(r(n)|\mathcal{E}|(p(n) + q(|\mathcal{E}|, n)))$ again. $\square$

Corollary 6.11 SIMPLE-H-SAT *is polynomial if the hypergraph satisfies the k-restriction on $r(\mathcal{H})$ and $ar(\mathcal{H})$, i.e., any two edges differ in their size at most by a constant k.*

Corollary 6.12 H-SAT *is polynomial if the size of the edges of $\mathcal{F}$ is bounded by a constant.*

Note that the above algorithm for H-SAT does not stop within polynomial time if the k-restriction does not hold, because the number of relevant sets covered by an edge is no longer bounded by a polynomial in the number of vertices. For example, let $\mathcal{H} = (\{v_1, \ldots, v_n\}, \mathcal{E})$, $\mathcal{E} = \{\{v_1\}, \{v_2, \ldots, v_n\}\}$. $\mathcal{H}$ is saturated, but the algorithm recognizes this only after $2^n - 1$ covering checks.

The polynomiality of the k-restricted SIMPLE-H-SAT problems yields the following polynomial subcase of TRANSVERSAL HYPERGRAPH:

Theorem 6.13 *If the instances $I = [\ \mathcal{G},\ \mathcal{H}]$ of* TRANSVERSAL HYPERGRAPH *satisfy $r(\mathcal{G}) - ar(\mathcal{G}) \leq k \wedge r(\mathcal{H}) - ar(\mathcal{H}) \leq k\ \wedge\ ar(\mathcal{G}) + r(\mathcal{H}) = n + c$ for some constants k and c (where n is the size of the vertex set of $\mathcal{G}$ and $\mathcal{H}$), then the problem is solvable in polynomial time.*

Proof: Transforming an instance of this subcase of TRANSVERSAL HYPERGRAPH by the transformation in the proof of Theorem 6.10 into an instance of SIMPLE-H-SAT yields a simple hypergraph that satisfies the k-restriction on the rank and antirank (with k adapted appropriately). $\square$

6.3 Acyclic Hypergraphs

Similar to graph theory, the notion of acyclicity is appealing in hypergraph theory from a theoretical as well as a practical point of view, and for some practical problems acyclic hypergraphs gain special attention, e.g. in the design of relational database schemata [BDM81, BFM+81, BFMY83, FMU82]. Several **NP**-complete problems on hypergraphs become polynomial if they are restricted to acyclic hypergraphs [Yan82]. Since it is not straightforward to carry over the definition of a cycle from graphs to hypergraphs, there are many notions of acyclicity in a hypergraph, cf. [Fag83, FV84, Ber89]. We refer to α-, β-, γ-, and *Berge*-acyclicity as stated in [Fag83], where the proper inclusion hierarchy *Berge*-acyclic $\Rightarrow$ γ-acyclic $\Rightarrow$ β-acyclic $\Rightarrow$ α-acyclic is proved.

The notion of $\alpha-$acyclicity came up in the context of relational database theory [BFMY83, Fag83]. We refer to a pure graph-theoretical definition of α-acyclicity. If $\mathcal{H}$ is a hypergraph

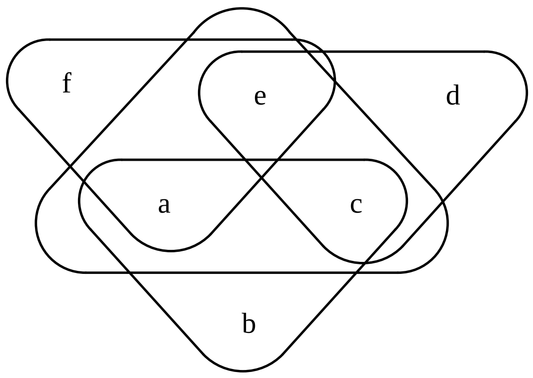

Figure 2: α-acyclic, but β-cyclic hypergraph $\mathcal{H} = \{\{a,b,c\},\{c,d,e\},\{a,e,f\},\{a,c,e\}\}$

on vertices V, then the graph $G(\mathcal{H})$ of $\mathcal{H}$ is the graph on vertex set V whose edges are the vertex sets $\{v,v'\}$ such that $v \neq v'$ and v,v' occur together in some edge of $\mathcal{H}$. A cycle of length $m \geq 2$ in graph $\mathcal{G} = (V,\mathcal{E})$ is a subset $\mathcal{C} = \{E_1,\ldots,E_m\} \subseteq \mathcal{E}$ of edges such that $E_i = \{x_{i-1},x_i\}, 1 \leq i < m, E_m = \{x_m,x_0\}$ and $x_i \neq x_j, 0 \leq i < j \leq m$. A clique of a graph $\mathcal{G} = (V,\mathcal{E})$ is a subset $V' \subseteq V$ such that for all distinct $v,v' \in V'$, $\{v,v'\} \in \mathcal{E}$. A hypergraph $\mathcal{H}$ is *conformal* if every clique of $G(\mathcal{H})$ is contained in some edge of $\mathcal{H}$. A graph $\mathcal{G}$ is chordal (or triangulated) if with every pair $\{v,v'\},\{v',v''\}$ of distinct edges which occur in some cycle the set $\{v,v''\}$ is an edge of $\mathcal{G}$.

Definition 6.2 *A hypergraph $\mathcal{H}$ is α-acyclic if and only if it is conformal and if $G(\mathcal{H})$ is chordal.*

For example, the hypergraph in Figure 2 is α-acyclic. The projective plane $\mathcal{P}_7$ (cf. Figure 1), however, is α-cyclic, i.e., not α-acyclic.

α-acyclicity of a hypergraph can be checked in linear time by an algorithm of Tarjan and Yannakakis [TY84]. A less efficient, but simpler described test is due to Graham [Gra79]. If a nonempty hypergraph $\mathcal{H}$ by repeated application of one of the two rules

1. If vertex v occurs in only one edge E, remove v from E.
2. If edges E', E satisfy $E' \subseteq E$, remove E'.

is reducible to the hypergraph $\{\emptyset\}$, then $\mathcal{H}$ is α-acylic. This sufficient condition is also necessary. The order of rule application does not matter, because the method has the Church-Rosser property, i.e., the hypergraph reached at termination is fully predetermined by the input and independent of the steps during the execution.

Note that rule 2 implies that a hypergraph $\mathcal{H}$ is α-acyclic if and only if $\max(\mathcal{H})$ is α-acyclic. A consequence of this is that an α-acyclic hypergraph $\mathcal{H}$ may contain an α-cyclic subhypergraph $\mathcal{H}' \subseteq \mathcal{H}$. For example, the hypergraph $\mathcal{H}$ in Figure 2 contains with $\mathcal{H}' = \{\{a,b,c\},\{c,d,e\},\{a,e,f\}\}$ an α-cyclic hypergraph, though $\mathcal{H}$ is α-acyclic. However, this is not what one expects from acyclicity, since it is surprising that a cycle in a hypergraph disappears by adding edges. Fagin points out this anomaly [Fag83] and introduces the more natural concept of β-acyclicity.

Definition 6.3 *A hypergraph $\mathcal{H}$ is β-acyclic if and only if every subhypergraph $\mathcal{H}' \subseteq \mathcal{H}$ is α-acyclic.*

The hypergraph in Figure 2 is β-cyclic, i.e., not β-acyclic. Fagin presents in [Fag83] various equivalent definitions of β-acyclicity, among them a related acyclicity criterion by Graham [Gra79], and he also gives a polynomial time algorithm to test β-acyclicity.

Our next aim is to show that the transversal hypergraph of a β-acyclic hypergraph is efficiently computable with respect to the input and the output size.

Definition 6.4 *Let $\mathcal{H}$ be a hypergraph on V. For $V' \subseteq V$, the partial hypergraph of $\mathcal{H}$ generated by V' is $\mathcal{H}_{V'} = \{E \cap V' | E \in \mathcal{H}\}$.*

For convenience, let us call any vertex an *ear node* of the hypergraph $\mathcal{H}$ if it occurs in exactly one edge of $\mathcal{H}$ (cf. [Ull88], p. 698). We observe the following rule for transversal computation.

Lemma 6.14 *Let $\mathcal{H}$ be a simple hypergraph on V with an ear node v that occurs in edge E. Then $\{\{v\} \cup T \mid T \in Tr(\mathcal{H}_{\overline{E}} - \{\emptyset\})\} \cup Tr(\mathcal{H}_{V-\{v\}})$ is a partitioning of $Tr(\mathcal{H})$.*

Proof: Note that if $\mathcal{H}$ is simple, then every essential vertex $v \in V$ occurs in at least one minimal transversal of $\mathcal{H}$. (This is easily shown from Corollary 2.3). However, $Tr(\mathcal{H}_{\overline{E}} - \{\emptyset\})$ is never empty. Now consider $Tr(\mathcal{H})$. The minimal transversals can be partitioned into $Tr(\mathcal{H}) = \mathcal{T}_1 \cup \mathcal{T}_2$, where $\mathcal{T}_1$ contains all minimal transversals that contain v and $\mathcal{T}_2$ all others. Since by Corollary 2.3 $Tr(Tr(\mathcal{H})) = \mathcal{H}$ if $\mathcal{H}$ is simple, E is a minimal transversal of $Tr(\mathcal{H})$, and since it is the only edge that contains v, it follows that $E \cap T = \{v\}$ for all $T \in Tr(\mathcal{H})$ that contain v, hence $T - \{v\} \subseteq \overline{E}$. From this it is clear that $\mathcal{T}_1 = \{\{v\} \cup T \mid T \in Tr(\mathcal{H}_{\overline{E}} - \{\emptyset\})\}$, and it is obvious that $\mathcal{T}_2 = Tr(\mathcal{H}_{V-\{v\}})$. $\square$

Theorem 6.15 *Let $\mathcal{H}$ be a simple β-acyclic hypergraph. Then the minimal transversals of $\mathcal{H}$ are P-enumerable.*

Proof: Recall that a problem is P-enumerable if there is some algorithm which computes all solutions to the problem in time $p(IS)N$, where p is some polynomial in the input size IS and N is the number of solutions. However, to include the case in which the problem has *no* solution, we slightly modify this convention from $p(IS)N$ to $p(IS)(N+1)$. It is not difficult to see that this modification is not substantial.

To prove the theorem, we proceed as follows: first we note some facts on β-acyclic hypergraphs, and then we give an algorithm for transversal computation. For our purposes we may assume without loss of generality that the input hypergraphs have no inessential vertices. We show by induction on the number of essential vertices that the algorithm can be implemented to work in time $p(IS(\mathcal{H}))(|Tr(\mathcal{H})|+1)$, where $IS(\mathcal{H})$ denotes the input size of hypergraph $\mathcal{H}$.

There is a simple observation on ear nodes of β-acyclic hypergraphs.

FACT 1: Every simple β-acyclic hypergraph with a nonempty edge $\mathcal{H}$ has an ear node.

Indeed, if $\mathcal{H}$ is β-acyclic, then it is also α-acyclic, and Graham's algorithm must succeed. However, because $\mathcal{H}$ is simple, rule 2 is not applicable, and because $\mathcal{H} \neq \{\emptyset\}$, rule 1 must apply. Hence $\mathcal{H}$ has an ear node.

FACT 2: Let $V' \subseteq V$ be a subset of vertex set V of hypergraph $\mathcal{H}$. If $\mathcal{H}$ is β-acyclic, then also $\mathcal{H}_{V'}$ is β-acyclic (see [Fag83], p. 530).

Now consider the following recursive procedure $BetaTr(\mathcal{H})$ for transversal computation, where the input hypergraph $\mathcal{H}$ on vertices V is simple:

```
$BetaTr(\mathcal{H})$:
 if $\mathcal{H} = \emptyset$ then
   return($\{\emptyset\}$)
 else
   if $\mathcal{H} = \{\emptyset\}$ then
     return($\emptyset$)
   else
     $v \leftarrow$ an ear node in $E \in \mathcal{H}$;
     $\mathcal{H}_1 \leftarrow \min(\mathcal{H}_{\overline{E}} - \{\emptyset\})$; $\mathcal{H}_2 \leftarrow \min(\mathcal{H}_{V-\{v\}})$;
     $\mathcal{T}_1 \leftarrow BetaTr(\mathcal{H}_1)$; $\mathcal{T}_2 \leftarrow BetaTr(\mathcal{H}_2)$;
     return($\{T \cup \{v\} \mid T \in \mathcal{T}_1\} \cup \mathcal{T}_2$)
   fi;
 fi;
```

We show the correctness of $BetaTr(\mathcal{H})$ by induction on the number of essential vertices of $\mathcal{H}$. Let $|V_e(\mathcal{H})| = 0$. Then $\mathcal{H} = \emptyset$ or $\mathcal{H} = \{\emptyset\}$, and the correct result is returned. Now consider $\mathcal{H}$ where $|V_e(\mathcal{H})| > 0$ and assume the hypothesis is correct for simple hypergraphs with less essential nodes than $\mathcal{H}$. Because $\mathcal{H}$ has at least one nonempty edge, is simple and β-acyclic, by FACT 1 it has at least one ear node, thus an appropriate vertex v in edge E will be found. Applying Lemma 6.14 we have $Tr(\mathcal{H}) = \{\{v\} \cup T \mid T \in Tr(\mathcal{H}_{\overline{E}} - \{\emptyset\})\} \cup Tr(\mathcal{H}_{V-\{v\}})$, which is clearly equivalent to $Tr(\mathcal{H})\{\{v\} \cup T \mid T \in Tr(\min(\mathcal{H}_{\overline{E}} - \{\emptyset\}))\} \cup Tr(\min(\mathcal{H}_{V-\{v\}}))$. Since a hypergraph is β-acyclic if and only if every subhypergraph of it is β-acyclic, in conjunction with FACT 2 it follows that $\mathcal{H}_1, \mathcal{H}_2$ are simple, β-acyclic hypergraphs. Because $\mathcal{H}_i, i \in \{1, 2\}$ has less essential vertices than $\mathcal{H}$, by the induction hypothesis we have $Tr(\mathcal{H}_i) = BetaTr(\mathcal{H}_i) = \mathcal{T}_i$. Thus we have $Tr(\mathcal{H}) = \{\{v\} \cup T \mid T \in Tr(\mathcal{H}_{\overline{E}} - \{\emptyset\})\} \cup Tr(\mathcal{H}_{V-\{v\}}) = \{\{v\} \cup T \mid T \in \mathcal{T}_1\} \cup \mathcal{T}_2$. However, this is exactly what $BetaTr(\mathcal{H})$ returns if $\mathcal{H}$ has at least one nonempty edge, and the induction hypothesis holds for $\mathcal{H}$.

It remains to prove the claim on the complexity of $BetaTr(\mathcal{H})$. Let $v(\mathcal{G}) = |V_e(\mathcal{G})|$ denote the number of essential vertices of a hypergraph $\mathcal{G}$ in what follows. Following Iverson's APL convention [Ive62] we denote with $[\mathcal{H} = \{\emptyset\}]$ the mathematical predicate which is 1 if $\mathcal{H} = \{\emptyset\}$ and 0 otherwise, i.e. if $\mathcal{H} \neq \{\emptyset\}$ (see also [GKP89]). Clearly, each return-statement can be done in time $k(v(\mathcal{H})+1)(|Tr(\mathcal{H})| + [\mathcal{H} = \{\emptyset\}])$ for some constant $k > 1$, and all other steps together except the recursive calls of $BetaTr$ can be done in time $p(|\mathcal{H}|, v(\mathcal{H}))$ (for brevity, $p(\mathcal{H})$), where p is some polynomial in $|\mathcal{H}|, v(\mathcal{H})$. Now let $q(\mathcal{H}) = q(|\mathcal{H}|, v(\mathcal{H}))$ be a polynomial in $|\mathcal{H}|, v(\mathcal{H})$ which has the following three properties: (1) $q(\{\emptyset\}) = p(\{\emptyset\}) + k$, (2) $\forall \mathcal{H} \neq \{\emptyset\} : p(\mathcal{H}) + k(v(\mathcal{H}) + 1) + q(\{\emptyset\}) \leq q(\mathcal{H})$, and (3) for all simple hypergraphs $\mathcal{H}, \mathcal{H}' : |\mathcal{H}'| \leq |\mathcal{H}| \wedge v(\mathcal{H}') < v(\mathcal{H}) \Rightarrow q(\mathcal{H}') \leq q(\mathcal{H})$. It is not difficult to show that such a polynomial q exists. Now we show by induction on the number of essential vertices of $\mathcal{H}$ that the runtime of $BetaTr(\mathcal{H})$ is bounded by $(v(\mathcal{H}) + 1)q(\mathcal{H})(|Tr(\mathcal{H})| + [\mathcal{H} = \{\emptyset\}])$.

If $\mathcal{H} = \emptyset$, we have $v(\mathcal{H}) = 0, |Tr(\mathcal{H})| = 1$, and $[\mathcal{H} = \{\emptyset\}] = 0$, thus the computational effort is bounded by

$$p(\emptyset) + k \;\leq\; p(\emptyset) + k + q(\{\emptyset\}) \;\leq\; q(\emptyset) \;=\; (|v(\emptyset)| + 1)q(\emptyset)(|Tr(\emptyset)| + [\emptyset = \{\emptyset\}]),$$

and the claim holds. For $\mathcal{H} = \{\emptyset\}$ we have $v(\mathcal{H}) = 0, |Tr(\mathcal{H})| = 0$, and $[\mathcal{H} = \{\emptyset\}] = 1$, thus

$$p(\{\emptyset\}) + k \;=\; q(\{\emptyset\}) \;=\; (v(\{\emptyset\}) + 1)q(\{\emptyset\})(|Tr(\{\emptyset\})| + [\{\emptyset\} = \{\emptyset\}]),$$

and the hypothesis holds. Now consider a hypergraph with at least one essential vertex. Applying the induction hypothesis and utilizing $|Tr(\mathcal{H})| = |Tr(\mathcal{H}_1)| + |Tr(\mathcal{H}_2)|$, $|Tr(\mathcal{H})| > 1$, and $[\mathcal{H}_1 = \{\emptyset\}] = 0$, the computational effort is bounded by

$$\begin{aligned}
& p(\mathcal{H}) \;+\; k(v(\mathcal{H})+1)|Tr(\mathcal{H})| \;+\; \sum_{i=1,2}(v(\mathcal{H}_i)+1)q(\mathcal{H}_i)(|Tr(\mathcal{H}_i)|+[\mathcal{H}_i=\{\emptyset\}]) \\
\leq\;& (p(\mathcal{H})+k(v(\mathcal{H})+1))|Tr(\mathcal{H})| \;+\; \underbrace{[\mathcal{H}_2=\{\emptyset\}](v(\mathcal{H}_2)+1)q(\mathcal{H}_2)}_{\leq q(\{\emptyset\})} \;+ \\
& \sum_{i=1,2}\underbrace{(v(\mathcal{H}_i)+1)}_{\leq v(\mathcal{H})}\underbrace{q(\mathcal{H}_i)}_{\leq q(\mathcal{H})}|Tr(\mathcal{H}_i)| \\
\leq\;& \underbrace{(p(\mathcal{H})+k(v(\mathcal{H})+1)+q(\{\emptyset\}))}_{\leq q(\mathcal{H})}|Tr(\mathcal{H})| \;+\; v(\mathcal{H})q(\mathcal{H})|Tr(\mathcal{H})| \\
\leq\;& q(\mathcal{H})|Tr(\mathcal{H})|+(v(\mathcal{H})+1)q(\mathcal{H})|Tr(\mathcal{H})|-q(\mathcal{H})|Tr(\mathcal{H})| \\
\leq\;& (v(\mathcal{H})+1)q(\mathcal{H})(|Tr(\mathcal{H})|+[\mathcal{H}=\{\emptyset\}])
\end{aligned}$$

hence the induction hypothesis holds for $\mathcal{H}$. It is clear that $(v(\mathcal{H})+1)q(\mathcal{H})$ is bounded by some polynomial $p_1(IS(\mathcal{H}))$ in the input size $IS(\mathcal{H})$ of the hypergraph $\mathcal{H}$. Because $[\mathcal{H}=\{\emptyset\}]\leq 1$, it is clear that $BetaTr(\mathcal{H})$ enables the enumeration of the minimal transversals of $\mathcal{H}$ in time $p_1(IS(\mathcal{H}))(|Tr(\mathcal{H})|+1)$, and the theorem is proved. □

Corollary 6.16 TRANSVERSAL HYPERGRAPH *is polynomial for β-acyclic hypergraphs.*

Corollary 6.17 SIMPLE-H-SAT *is polynomial for β-acyclic hypergraphs.*

Proof: By Theorem 4.5 and Theorem 4.13 respectively. □

Note that algorithm $BetaTr$ outputs all minimal transversals at the end of the computation. It is not difficult to modify the algorithm such that the minimal transversals are output with delay polynomial in the input; this is left to the reader.

One will ask why algorithm $BetaTr(\mathcal{H})$ should not work for simple α-acyclic hypergraphs also, because every such hypergraph has an ear node, and because for every α-acyclic hypergraph $\mathcal{H}$ on V and every subset $V'\subseteq V$ it holds that $\mathcal{H}_{V'}$is also α-acyclic (cf. [Fag83], p. 530). Unfortunately, this is not true as the hypergraph in Figure 2 shows. Independent from the selection of ear nodes, at some point in the computation a call of $BetaTr(\mathcal{H}')$ for $\mathcal{H}'=\{\{a,c\},\{c,e\},\{a,e\}\}$ will occur. $\mathcal{H}'$ is simple and has no ear node (thus it is α-cyclic), and the computation is stuck. $BetaTr(\mathcal{H})$ fails for α-acyclic hypergraphs in consequence of the anomaly pointed out above: α-acyclicity of a hypergraph $\mathcal{H}_{V'}, V'\subseteq V$, does not imply α-acyclicity of $\min(\mathcal{H}_{V'})$, but this is essentially what is needed for transversal computation.

7 Problems closely related to SIMPLE-H-SAT in Database Theory

After our study of transversal and saturation problems on hypergraphs in the previous sections, we are ready to apply our results to several problems in various fields of practical computer science. In this section we point out the significance of the hypergraph problems considered to some important problems in database theory.

7.1 Relational Database Design

We start with a brief recall of the necessary concepts of relational database theory (cf. [Mai83]).

A *relation* (or *relation instance*) is a table of pairwise distinct tuples the components of which are values from the domains of the attributes. More formally, let $U = \{A_1, \ldots, A_n\}$ be a set of *attributes*, each of which is associated to a *domain*. Let $D(A_i)$ denote the domain of attribute $A_i, 1 \leq i \leq n$. Then a *relation over* U is a subset of $\prod_{1\leq i\leq n} D(A_i)$, and the elements of a relation are called *tuples*. The ordering of the attributes in a tuple is considered as irrelevant.

According to the usual terminology for sets of attributes, we use concatenation as notation for set union and identify singletons by their element, i.e., if X, Y denote sets of attributes and A is an attribute, then $XY, A, X - A$ stands for $X \cup Y, \{A\}, X - \{A\}$, respectively.

Functional dependencies (FDs) are of crucial importance for relations. If t is a tuple in Relation R and X is a nonempty set of attributes, then $t[X]$ denotes the tuple which is given by the restriction of tuple t onto the attributes in X, called the *projection of t onto X*. Given sets X, Y of attributes, the *functional dependency* (FD) $X \rightarrow Y$ holds in relation R if in every tuple the attribute values in Y are determined uniquely by values of the attributes in X, i.e., for tuples $t_1, t_2 \in R$, $t_1[X] = t_2[X]$ implies that $t_1[Y] = t_2[Y]$. Since possibly all tuples of a relation have the same values for some attribute, the *left hand side* X of a FD $X \rightarrow Y$ may be empty.

If $Y \subseteq X$, then the FD is called trivial. We define that on the empty relation $R = \emptyset$ no functional dependencies hold, i.e., $F_\emptyset = \emptyset$. A pair $RS = <U, F>$ of a set F of FDs on a set U of attributes is called a *relation scheme*.

For the set of all FDs that hold on a relation R, which we denote by F_R, the following three rules hold : *reflexivity*: $X \rightarrow X \in F_R$, *augmentation*: $X \rightarrow Y \Rightarrow XZ \rightarrow Y \in F_R$, and *pseudotransitivity*: $X \rightarrow Y, YW \rightarrow Z \in F_R \Rightarrow XW \rightarrow Z \in F_R$.

Given a set F of FD, we denote by F^+ the closure of F under the three rules above, which are sound and complete for all FD implied by the FDs in F [Arm74]. F is called a *full family* *of FD* if $F = F^+$. Of course, F_R is a full family of functional dependencies for every relation R. A set F of FD satisfying $F^+ = F_R$ is called a *cover of* R. Clearly, there is a cover for every relation R. On the other hand, for every set F of FDs there is a relation R such that $F^+ = F_R$; R is called an *Armstrong Relation* for F. For example, $R = \emptyset$ is the one and only Armstrong Relation for $F = \emptyset$.

Equivalence of sets F, G of FDs is of natural interest. We write $F \equiv G$ if $F^+ = G^+$. F is called a *cover* of G. F is called *nonredundant* if for all $f \in F$, $f \notin (F - f)^+$.

A set $K \subseteq U$ of attributes is called a *(super-)key* for relation scheme $RS = <U, F>$ if $K \to U \in F^+$. A key K is called *minimal* if no proper subset of K is key. Note that the minimal keys of a relation scheme do not necessarily cover the whole set of attributes, and the empty set may occur as the one and only key. Dually to the concept of minimal keys, a set K' of attributes is called an *antikey* of $RS = <F, U>$ if K' is a maximal nonkey, i.e., $K' \to U \notin F^+$, but $K'' \to U \in F^+$ for every proper superset K'' of K' [Thi86, DT87].

The usefulness of Armstrong relations in design of relational databases was pointed out by Mannila and Räihä [MR86]. Augmenting the traditional paradigm in database design which is based upon functional dependencies, Mannila and Räihä emphasize the role of Armstrong relations as exemplary instances of the relation schemes designed. Though covers and relations are equivalent means to represent full families of FDs, some things seem to be easier to see in the one representation than in the other and vice versa. An example relation R may help the designer in verifying the design of a cover F of a full family of FDs. Clearly all FDs in F should hold in R; to prevent a "bad example" which is a relation that might deceive the designer by assuring some FDs not deducible from F, the salient point for R is that it embodies *exactly* those FDs deducible from F, i.e., R is an Armstrong relation for F. Following these ideas, Mannila and Räihä develop the *Design-By-Example* system for relational database design [MR86] which makes use of both covers and Armstrong relations in the design process.

An argument in favor of using both covers and Armstrong relations in the design process is their dual behavior with respect to modifications: If a FD f is being added to a set F of FD, it is easily verified that the full family of FDs implied by F increases monotonically, i.e., $F^+ \subseteq (F + f)^+$. Adding a tuple t to a relation R, however, yields $F_R \supseteq F_{R \cup \{t\}}$, thus causes a monotonic decrease in the set of FDs valid on R. The same opposite behavior holds for the removal of FDs from F or tuples from R, respectively. Removing a FD from F results in a monotonical decrease of the set of full FDs implied, but the removal of a tuple t from R monotonically increases the set of FDs which hold on R. Thus given a set of FDs and an Armstrong relation for it, the designer may modify the set of FDs or the relation R at any step of the design process until in the end a satisfactory relation scheme results.

Full families of FDs are usually characterized by nonredundant covers. For a very useful characterization of Armstrong relations, we need some additional definitions.

Let X be a subset of the attributes U of the relation scheme $RS = <U, F>$. Then $Cl_F(X) =$

$\{A \mid X \to A \in F^+\}$ is referred to as the *closure of X with respect to F*. X is *closed* if $Cl_F(X) = X$. It is well known that the set S_F of closed subsets of U is a semilattice with respect to intersection, i.e., $X, Y \in S_F \Rightarrow X \cap Y \in S_F$[BDFS84]. An element X from S_F is called *irreducible* if for all Y, Z from S_F it holds that $X = Y \cap Z$ implies $X = Z$ or $Y = Z$. S_F is fully described by the set $GEN(F)$ of all its irreducible elements; every element from S_F can be described as intersection of proper (not necessarily distinct) elements from $GEN(F)$. Since $S_F = S_G \Rightarrow F^+ = G^+$, and since every S_F has a unique set $GEN(F)$, the set F of FDs is uniquely characterized by $GEN(F)$.

The following theorem by Mannila and Räihä leads to a concise characterization of Armstrong relations:

Theorem 7.1 ([MR86]) *Let F be a set of FDs on U. Then $GEN(F) = MAX(F)$, where $MAX(F) = \bigcup_{A \in U} MAX(F, A)$ and $MAX(F, A) = \{Y \subseteq U \mid Y \to A \notin F^+ \wedge \forall B \in U - Y : YB \to A \in F^+\}$.*

Note that for $Y \subseteq X$, from $X \in MAX(F, A)$ it follows that $Y \to A \notin F^+$. Thus X is a maximal left hand side *excluded functional dependency* (XFD; written as $X \not\to A$) for F. As easily proved from this, the antikeys for F are given by $\max(MAX(F))$.

$MAX(F, A)$ is with respect to computational problems advantageously related to the size of Armstrong relations. Given a relation R, $MAX(F_R, A)$ can be computed in polynomial time for every attribute A (cf. [MR86, MR88b]); conversely, given all MAX-sets for input, an Armstrong relation for F is constructible in polynomial time (equivalent with respect to polynomiality, if $GEN(F)$ and F are given for input). Thus the MAX-sets (or maximal XFDs) are computationally equivalent to Armstrong relations with respect to polynomiality
.

The usage of Armstrong relations hand in hand with FDs in the design process raises two problems:

AP1: Constructing Armstrong Relation : Given a set F of FD, construct an Armstrong relation for F.

AP2: Dependency Inference : Given a relation R, find a nonredundant cover of F_R.

We note that inferring dependencies has also an application in machine learning or knowledge acquisition, if a concept (set of dependencies) is searched for that fits a collection of data (the tuples of a relation) exactly [MR88b].

Problems AP1 and AP2 are both of inherent exponential time complexity in the input size, as there are problem instances for which the size of a minimal Armstrong relation or a nonredundant cover, respectively, is *exponential* in the input size, which means that it is *impossible* to solve these problems in input-polynomial time. In face of the inherent complexity, it is of interest to have algorithms which solve these problems in output-polynomial total time. Unfortunately, even under this relaxed complexity requirement no efficient algorithms are known for both problems [MR86, MR87, BDFS84]. Note that problems AP1 and AP2, which are search problems in terms of complexity theory, are solvable by algorithms in output-polynomial time only if the following decision problem, which we call FD-RELATION EQUIVALENCE, is in **P**:

Problem:	FD-RELATION EQUIVALENCE
Instance:	A relation R and a set F of FD.
Question:	Does $F_R = F^+$ hold ?

FD-RELATION EQUIVALENCE is in co-**NP**, but there is neither a polynomial time algorithm known for this problem nor is it proved co-**NP**-complete. For the practical important subcase of normalized relation schemes, FD-RELATION EQUIVALENCE turns out to be SIMPLE-H-SAT-complete.

A relation scheme $RS = <U, F>$ is in *Boyce-Codd Normal Form* (BCNF) if for every $X \rightarrow Y \in F^+$ such that $Y \not\subseteq X$, X is a superkey for RS. BCNF has some nice computational properties: checking if a relation scheme RS is in BCNF can be done in polynomial time, and computing all keys can be done in time polynomial in the *input*. Every minimum-cardinality nonredundant cover of a relation scheme in BCNF consists of FDs $\{K_i \rightarrow Y_i \mid 1 \leq i \leq k\}$ where $K_1 \cup Y_i = U$ and $K_i, 1 \leq i \leq k$ are the minimal keys of the scheme. A relation scheme in BCNF is uniquely determined by the set of its minimal keys; note that in general different relation schemes may have the the same set of keys. In terms of hypergraph transversals, keys and antikeys are related as follows:

Theorem 7.2 *Let $\mathcal{K}$ be the keys of the relation scheme RS =$<U, F>$, and $\mathcal{A}$ the family of antikeys. Then $\mathcal{A} = \overline{Tr(K)}$.*

Proof: If $\mathcal{A} = \emptyset$ then the only key of RS must be the empty attribute set, i.e., $\emptyset \rightarrow U \in F^+$. Since $\emptyset = Tr(\{\emptyset\}) = \overline{Tr(\{\emptyset\})}$, the claim holds in this case. If $\mathcal{K} = \emptyset$ (this is the case if and only if $F = \emptyset$), then U is the only antikey of RS, and again the claim holds because $\{U\} = \overline{\{\emptyset\}} = Tr(\emptyset)$. Now consider the remaining cases. If $\mathcal{A} \neq \emptyset$, consider $X \in \mathcal{A}$. Because X is not superset of any minimal key, we have that for every minimal key Y, $Y - X \neq \emptyset$, thus $\overline{X} \cap Y \neq \emptyset$. Hence $\overline{X}$ is a transversal of $\mathcal{K}$. Since adding any $B \in \overline{U}$ makes X become a key for RS, for every $B \in U - X$ there exists $K \in \mathcal{K}$ such that $XB \supseteq K$, which implies that $(\overline{U} - B) \cap X = \emptyset$. But this means that for all $B \in \overline{U}, K \cap \overline{U} = B$. Thus $\overline{X}$ is a minimal transversal of $\mathcal{K}$, and $\mathcal{A} \subseteq \overline{Tr(K)}$. Now let Y be a minimal transversal of $\mathcal{K}$. Then $\overline{Y}$ is not a superset of any minimal key. Removing any A from Y there is some $K \in \mathcal{K}$ such that $K \cap Y = \emptyset$. But this means that adding any attribute B from Y to $\overline{Y}$, $\overline{Y}$ becomes a key. Thus $\mathcal{A} \supseteq \overline{Tr(K)}$, and the claim is proved. $\square$

Note that the antikeys of a relation scheme are the maximal independent sets of the hypergraph of minimal keys for the scheme.

Now the following theorem is easily proved:

Theorem 7.3 *Let F be a set of FDs of a scheme in BCNF, and R a relation instance. Testing if F is a cover for R is* SIMPLE-H-SAT-*complete.*

Proof: Testing whether R is in BCNF can be done in polynomial time. Moreover, the antikeys $\mathcal{A}_R$ for R are computable in time polynomial in the size of R [DT87]. Since F is in BCNF, the set $\mathcal{K}_F$ of keys for F is computable in time polynomially in the size of F. Thus by Theorem 7.2 the problem is polynomial time transformable into TRANSVERSAL HYPERGRAPH which is SIMPLE-H-SAT-complete. Conversely, given an instance $I = [\mathcal{G}, \mathcal{H}]$ of TRANS-HYP, we assume without loss of generality that $\mathcal{G}$, $\mathcal{H}$ are simple hypergraphs and define a set of FDs $F = \{G \rightarrow A \mid G \in \mathcal{G} \wedge A \in U - G\}$. It is not difficult to prove that F is in BCNF; moreover, $\mathcal{G}$ is the set of minimal keys for the relation scheme. Selecting $\overline{\mathcal{H}}$ as set of antikeys, we can construct in polynomial time an Armstrong relation R such that F_R is in BCNF [DT87]. By Theorem 7.2 for the instance $I' = [F, R]$ we have that F is a cover of F_R iff $\overline{Tr(G)} = \overline{\mathcal{H}}$, i.e., $\mathcal{H} = Tr(G)$. Thus TRANSVERSAL HYPERGRAPH is polynomial time transformable into FD-RELATION EQUIVALENCE for schemes in BCNF, which completes the proof. $\Box$

An important consequence of the transformation of FD-RELATION EQUIVALENCE into TRANS-HYP in the proof of Theorem 7.3 and Corollary 6.6 is that FD-RELATION EQUIVALENCE for relations in BCNF is polynomial decidable if the left hand sides of the FDs in F are small, a case that often applies in practice. If F is in BCNF, every minimal key is contained in some X for FD $X \rightarrow Y \in F$, thus the hypergraph of minimal keys constructed for the TRANS-HYP test has also small edges, and Corollary 6.6 applies. Moreover, it is not difficult to show that AP1 (Constructing Armstrong Relation) can be done in output-polynomial total time for a relation scheme in BCNF where the FDs have small left hand as follows. First the minimal keys of the scheme are computed in input-polynomial time and then by algorithm T in the proof of Theorem 6.5 the antikeys, from which a corresponding BCNF relation instance is easily constructed in polynomial time [DT87]. As the antikeys $\mathcal{A}$ of a relation R in BCNF are given by the union of the MAX-sets of F_R, i.e., $\mathcal{A} = \min(MAX(F_R)) = MAX(F_R)$ and the MAX-sets resp. XFDs are computationally equivalent to R with respect to polynomiality, the whole computation needs time polynomial in the input and output size, i.e. is output-polynomial.

Note that problem AP2 (Dependency Inference), the problem dual to AP1, can be done in output-polynomial total time if the left hand sides of the XFDs (i.e., the MAX-sets) valid for a relation instance R in BCNF are large. In this case, the antikeys are large, and applying Theorem 7.2 the minimal keys of R are computable in output-polynomial time; a nonredundant cover of F_R is easily constructed from the minimal keys. Since the number of FDs in every nonredundant cover of a relation scheme in BCNF is polynomially related to the number of attributes and minimal keys, this procedure works in output-polynomial time.

After the problem of checking equivalence of a set of FDs and arelation instance, let us consider the problem of finding keys for a database. The derivation of minimal keys for relation schemes and relation instances is an important task in the design of database systems, and it is well known that finding a minimum cardinality key is **NP**-hard for a relation scheme [LO78] as well as for a relation instance [BDFS84]. However, a minimal key is easily computable for both relation schemes and relation instances in polynomial time. Finding all minimal keys of a relation scheme or relation is in general not possible in polynomial time, because there can be exponentially many. For relation schemes, Lucchesi and Osborn gave an output-polynomial algorithm [LO78], which implies that the ADDITIONAL KEY problem for a relation scheme, given a family of minimal keys $\mathcal{K}$ for relation scheme RS, decide if there is another minimal key not in $\mathcal{K}$, is solvable in polynomial time. For relation instances, no output-polynomial algorithm for minimal key computation is known. The complexity of ADDITIONAL KEY formulated for relations is important for the issue whether such an algorithm is likely to exist:

Problem:	ADDITIONAL KEY for relations
Instance:	A relation R on attributes U, a family $\mathcal{K}$ of minimal keys for R.
Question:	Is there a minimal key for the scheme $RS = <U, F_R>$ not contained in $\mathcal{K}$?

From Theorem 7.2, it is not difficult to show that this problem is closely related to SIMPLE-H-SAT.

Theorem 7.4 ADDITIONAL KEY *for relations is co-*SIMPLE-H-SAT*-complete.*

Proof: As the antikeys of any relation are computable in polynomial time, the complementary problem to ADDITIONAL KEY for relations is obviously polynomial time transformable into TRANSVERSAL HYPERGRAPH which is SIMPLE-H-SAT-complete. Thus the problem itself is transformable into co-SIMPLE-H-SAT. Conversely, co-TRANS-HYP for instance $I = [\mathcal{G}, \mathcal{H}]$ can be solved with ADDITIONAL KEY applied to $J = [R, \mathcal{K}]$, where R is constructed in polynomial time for $\overline{\mathcal{D}}$ as the set of antikeys (at least possible for BCNF) and $\mathcal{K} = \overline{\mathcal{G}}$. Thus co-SIMPLE-H-SAT is polynomial time transformable into ADDITIONAL KEY for relations. □

Note that we may take this as weak evidence that SIMPLE-H-SAT is not co-**NP**-complete: We do not know of any tractable decision problem on relation schemes which is in its respective formulation for relations intractable. Thus **NP**-completeness of ADDITIONAL KEY for relations (implied by co-**NP**-completeness of SIMPLE-H-SAT) would be a bit surprising.

Next we show that FD-RELATION EQUIVALENCE is co-SIMPLE-H-SAT-complete for a far more general class of instances than BCNF schemes, in particular, if F has a special form which we call minimal attribute key form.

Definition 7.1 *A set F of FDs on attributes U is in minimal attribute key form (MAK-form), iff $F = \{X \rightarrow A \in F^+ \mid A \in U \wedge \nexists X' \rightarrow A \in F^+ : X' \subset X\}$. F is a MAK-cover of G if F is in MAK-form and $F \equiv G$.*

In words, F is in MAK-form if it contains exactly those nontrivial FDs $X \rightarrow A$ from F^+ such that X is a minimal key for A. Note that in many cases, $A \rightarrow A \in F$ if F is in MAK-form. Clearly, every set of FDs G has a unique MAK-cover. Our motivation to consider sets of FDs in MAK-form is not *ex improviso*. In fact, there is a close relationship in terms of transversals between excluded functional dependencies and minimal attribute keys in the light of duality, captured in the following Lemma :

Lemma 7.5 ([MR88a]) *Let F be a set of FD, let $MAK(F, A)$ denote the hypergraph of minimal attribute keys for attribute A in F and let $MAX(F, A)$ be $\overline{\text{the hypergraph of}}$ left hand sides of all maximal XFDs for attribute A. Then $MAX(F, A) = \overline{Tr(MAK(F, A))}$.*

Proof: Essentially the same as for Theorem 7.2. □

The lemma aids in proving the following theorem:

Theorem 7.6 FD-RELATION EQUIVALENCE *restricted to instances where F is in MAK-form is* SIMPLE-H-SAT-*complete.*

Proof: A polynomial time transformation of TRANSVERSAL HYPERGRAPH, which is SIMPLE-H-SAT-complete, into the problem considered is at hand by the results above: the transformation of TRANSVERSAL HYPERGRAPH into FD-RELATION EQUIVALENCE for BCNF schemes in the proof of Theorem 7.3 yields the set F of FDs in MAK-form.

To prove the problem SIMPLE-H-SAT-complete, we transform instances $I = [F, R]$ into TRANSVERSAL HYPERGRAPH as follows:

Let $U = \{A_1, \ldots, A_n\}$. We compute the set $\mathcal{X}$ of maximal XFDs on R first, which can be done in polynomial time. By Lemma 7.5 the problem reduces to check n transversal equations simultaneously. Without loss of generality we may assume that $F \neq \emptyset$, $\mathcal{X} \neq \emptyset$ and that in F, $\mathcal{X}$ there is no FD $\emptyset \rightarrow A$ resp. XFD $\emptyset \not\rightarrow A$. (In the latter cases U and F and/or $\mathcal{X}$ can be reduced appropriately after checking if the transversal relationship for the attributes removed is satisfied.)

Define

$$\begin{aligned} U' &= \{A_{i,j} \mid 1 \leq i,j \leq n\} \cup \{B\}, \\ F' &= \{(U' - A) \to A \mid A \in U' - B\} \cup F'_B, \end{aligned}$$

where

$$\begin{aligned} F'_B \;=\; \textstyle\bigcup_{1 \leq i \leq n} \;& \{X \to B \mid X = A_{i,j_1} \cdots A_{i,j_k} \wedge A_{j_1} \cdots A_{j_k} \to A_i \in F\} \cup \\ & \{A_{i,j}A_{k,l} \to B \mid 1 \leq i,j,k,l \leq n, i \neq k\}. \end{aligned}$$

It is easily verified that for every attribute A in $U' - B$, the only minimal key with respect to F' is $U' - A$. For B the minimal keys are the minimal elements (with respect to set inclusion) of all left hand sides of the FDs in F'_B.

We define a set $\mathcal{X}'$ of maximal XFDs on U' by

$$\mathcal{X}' = \bigcup_{A \in U' - B} \{U' - A \not\to A\} \cup \mathcal{X}_B$$

where

$$\mathcal{X}_B = \bigcup_{1 \leq i \leq n} \{X \not\to B \mid X = A_{i,j_1} \cdots A_{i,j_k}, A_{j_1} \cdots A_{j_k} \not\to A_i \in \mathcal{X}\}.$$

According to our construction and our assumptions, F is a cover for F_R if and only if F' has $\mathcal{X}'$ as the corresponding set of maximal XFDs. Since for every $A \in U' - B$ the minimal attribute keys for A in F' and the maximal XFDs in $\mathcal{X}'$ satisfy the necessary and sufficient criterion from Lemma 7.5, this holds if and only if $MAX(F', B)$ is the family of the minimal transversals of $\overline{MAK(F', B)}$. Thus F is a cover for R if and only if $MAK(F', B) = Tr(\overline{MAK(F', B)})$. Since $\overline{MAK(F', B)}$ and $MAX(F', B)$ are constructible from F, R in polynomial time, the restricted FD-RELATION EQUIVALENCE problem is polynomial time transformable into TRANSVERSAL HYPERGRAPH, as was the claim. □

Several problems which are tractable for relation schemes become tractable if an Armstrong relation is given for input. Covers in MAK-form have some nice computational properties which make it possible to solve certain problems efficiently. This is due to the following fact:

Proposition 7.7 *Let F be a set FDs in MAK-form. Then a cover of the* projection *of F^+ onto $U', U' \subseteq U$, defined by $F^+[U'] = \{X \to Y \in F^+ \mid XY \subseteq U'\}$ is computable in polynomial time.*

Proof: The minimal attribute keys for attribute $A, A \in U'$ with respect to $G = F^+[U']$ are exactly those minimal attribute keys for A in F which are fully contained in U', i.e., $MAK(G, A) = \{X \in MAK(F, A) \mid X \subseteq U'\}$. □

The computation of a cover for the projection of F onto a subscheme $U' \subseteq U$ is inherently exponential, as there exist triples U, U', F such that every minimum cardinality cover of

$F[U']$ is of exponential size in $|F|$. There are certain problems on relation schemes which are polynomial time solvable for the full scheme, but become **NP**-hard if they are formulated for subschemes. For example, the ADDITIONAL KEY problem for relation scheme $RS = <U, F>$ has this property. If the problem statement is generalized from the full scheme to an arbitrary subscheme $U' \subseteq U$, i.e., the given keys $\mathcal{K}$ constitute a simple hypergraph on U' and the additional key must be a subset of U', the problem turns out **NP**-complete [BB79]. If we restrict the FDs sets in the input of ADDITIONAL KEY for relation schemes to those in MAK-form, Proposition 7.7 immediately implies that the problem remains in **P**, since we can compute a cover for the projection $F^+[U']$ in polynomial time and then apply the polynomial ADDITIONAL KEY algorithm for the full scheme.

In the case of relation instances, ADDITIONAL KEY does not become harder if we generalize the problem from the full set of attributes to an arbitrary subset:

Proposition 7.8 ADDITIONAL KEY *for subrelation instances, i.e., given a relation R on attributes U, a set $U' \subseteq$ of attributes, and a family $\mathcal{K}$ of minimal keys for $RS' = <U', F_R[U']>$ that are all fully contained in U', decide if there is an additional minimal key $K \subseteq U'$ for RS' not contained in $\mathcal{K}$, is co-*SIMPLE-H-SAT*-complete.*

Proof: Consider the projection of the relation R on U onto the attributes in U', i.e., compute $R' = \{t' \mid \exists t \in \mathrm{R}\colon t'[B] = t[B], \forall B \in U'\}$. Since $F_{R'} = F_R[U']$, the problem is hereby obviously transformed into an equivalent instance of ADDITIONAL KEY for relation in polynomial time, which is a subproblem of the problem considered. Thus co-SIMPLE-H-SAT-completeness of ADDITIONAL KEY for relation instances yields the result. □

Similar to the ADDITIONAL KEY problem, testing BCNF for a full relation scheme can be done in polynomial time, but if the problem is generalized to check if a subscheme is in BCNF, it becomes **NP**-hard (more precisely, co-**NP**-complete). If the set F of FDs is in MAK-form, however, by Proposition 7.7 it follows that the BCNF test for subschemes is polynomial time transformable to a BCNF test on a full scheme and is hence realizable in polynomial time. For relation instances, testing BCNF for a subrelation is in **P**.

We remark that BCNF-test shows that one effect of restricting covers to the MAK-form is that an intractable problem becomes tractable. Thus the MAK-form seems as like as an Armstrong relation to be a computationally advantageous representation of a full set of FD. Unfortunately, there is the same major drawback to the MAK-form as to Armstrong relations: As is easy to show, a cover G of F can be of exponential order in the size of any minimal cover F if G is in MAK-form. For example, let $U = A_1 \cdots A_n B_1 \cdots B_n C$, $F = \{A_i \to B_i \mid 1 \leq i \leq n\} \cup \{B_1 \cdots B_n \to C\}$. The set of all minimal keys for C is $MAK(F, C) = \{D_1 \cdots D_n \mid D_i \in \{A_i, B_i\}, 1 \leq i \leq n\}$. Since every other attribute has at most two minimal attribute keys, the size of G is of $\mathrm{O}(2^n + n) = \mathrm{O}(2^{|F|-1})$. This increase in the size of cover in MAK-form may be interpreted as the tradeoff for substituting the full power of inferencing given by the rules of projection, augmentation and transitivity by a less powerful, but with respect to the deducible FDs equivalent inference system (in fact, if a cover is in MAK-form, the mechanism needs not support the derivation of minimal attribute keys by transitivity as is necessary for an inference system on an arbitrary cover). Note that if $\mathbf{P} \neq \mathbf{NP}$, then this restriction in some sense substantially diminishes the required inferential power.

We conclude this subsection with a brief reminder of the main implications of SIMPLE-H-SAT on some problems in relational database theory. The FD-RELATION EQUIVALENCE problem, which is an open problem, is SIMPLE-H-SAT-complete if the relation is in BCNF. Moreover, the problem is SIMPLE-H-SAT-complete if the functional dependencies on the relation scheme are given by a cover in MAK-form. In addition to this an important problem for relation instances, the ADDITIONAL KEY problem, is co-SIMPLE-H-SAT-complete, and this problem remains co-SIMPLE-H-SAT-complete for the generalization from the full relation onto arbitrary subrelations. Since the corresponding ADDITIONAL KEY problem for relation schemes is in $\mathbf{P}$ and no tractable problem on relation schemes is known for which the corresponding problem on relation instances is intractable, we may take this as weak evidence that ADDITIONAL KEY for relations, hence SIMPLE-H-SAT, is not intractable.

7.2 Updates in Distributed Databases

Besides the occurrence of TRANS-HYP in relational database theory, this problem has a another application in databases. In distributed database systems, data are usually replicated and stored at multiple sites. A concurrency control scheme has to assure that accesses of transactions to the same data at several sites (for example, an update) do not invalidate the data consistency in the system. Unreliable communication and failure of sites are problems to be managed by the concurrency control mechanism. In particular, it has to cope with network partitions, i.e., broken communication lines such that the sites are separated into at least two groups with communication impossible between these groups. It is evident that concurrent updates in isolated groups may lead to divergent data copies. Therefore, in case of a network partition, at most *one* group (the "active group") should be enabled to perform updates, while all other groups should remain passive until they are re-connected to the active group. We are thus faced with a double problem: First, we need a policy which for each network partition identifies at most one group as the active group. Second, the policy must enable each group to determine its status (active or passive) by itself (i.e., without communication with any other group).

A solution to this problem is the mutual exclusion approach in [Lam78], which is based on minimal sets of sites (quorums) for access permission. A group is active iff it contains a quorum as subset. Mutual exclusion is assured as each pair of the specified quorums must have a site in common. (Consequently, inconsistent updates in case of network partitions are impossible). Clearly, it is natural to consider only minimal quorums, i.e., no quorum should contain another. In terms of hypergraph theory, this means that the specified quorums should constitute a simple, intersecting hypergraph on the set of sites, termed a *coterie* in [GMB85]. For example, the hypergraph $\mathcal{C} = \{\{a,b,c\},\{a,b,d\},\{a,c,d\},\{b,c,d\}\}$ is a coterie on vertices $\{a,b,c,d\}$. Coteries appeared to be a fruitful concept for access control in distributed databases, cf. [GMB85, IK90, Fu90].

The link between coteries and the problem of the recognition of the transversal hypergraph is via the notion of domination between coteries. From the point of reliability of a distributed database system, it is desirable that the quorums in a coterie are as small as possible, because this enhances the capability of the system to cope with node failures without loosing ability for operation executions. For example, the coterie $\mathcal{D} = \{\{a,b\},\{a,c\},\{a,d\},\{b,c,d\}\}$ is under this aspect preferred to $\mathcal{C} = \{\{a,b,c\},\{a,b,d\},\{a,c,d\},\{b,c,d\}\}$. If the node d fails, the group $\{a,b\}$ can nontheless perform the operation based upon coterie $\mathcal{D}$, but not if $\mathcal{C}$ is to obey. It is said that $\mathcal{D}$ *dominates* $\mathcal{C}$.

Definition 7.2 *A coterie $\mathcal{A}$ dominates a coterie $\mathcal{B}$ if and only if $\mathcal{A} \neq \mathcal{B}$ and $\forall B \in \mathcal{B}$: $\exists A \in \mathcal{A}$; $B \supseteq A$. A coterie $\mathcal{A}$ is nondominated (ND) if there is no coterie $\mathcal{B}$ that dominates $\mathcal{B}$.*

Thus coterie $\mathcal{A}$ dominates coterie $\mathcal{B}$ iff $\mathcal{A} \neq \mathcal{B}$ and $\mathcal{B} \geq \mathcal{A}$, and the ND coteries are the coteries minimal under order "$\geq$". For example, coterie $\mathcal{D}$ from above is ND.

In practice, ND coteries are desired for decision agreement groups with respect to reliability considerations. Unfortunately, there is no efficient test for nondomination of a coterie known; in [GMB85] merely an algorithm which has worst case run time exponential in the number of vertices is described. Interestingly, this problem turns out as the SELFTRANSVERSALITY problem.

Theorem 7.9 *A coterie $\mathcal{C}$ is ND if and only if $\mathcal{C} = Tr(\mathcal{C})$.*

Proof: A coterie is dominated iff it is 2-colorable ([GMB85], Theorem 2.4). Because an intersecting hypergraph $\mathcal{H}$ is 2-colorable iff $\mathcal{H} \neq Tr(\mathcal{H})$ (cf. 4.9), the theorem is obvious. □

Corollary 7.10 *Checking nondomination of a coterie* (ND COTERIE) *is* SIMPLE-H-SAT-*complete.*

Proof: By Theorems 7.9 and 4.7. □

Note that by Theorem 7.9 and Corollary 6.6 the nondomination problem for coteries is decidable in polynomial time if the quorums are small.

Recently, the concept of coteries has been generalized to bicoteries and semicoteries [Fu90, IK90] to model systems where the concurrency control scheme makes use of "read quorums" and "write quorums". Since for read access mutual exclusion is unnecessary, read

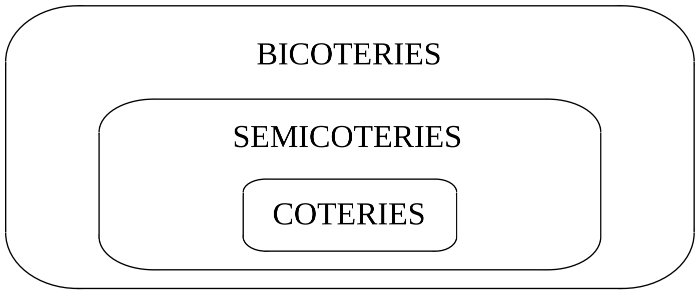

Figure 3: Venn diagram of coteries, semi- and bicoteries

quorums need not intersect; it is sufficient if the write quorums do and every read quorum has nonempty intersection with every write quorum.

A *bicoterie* is an ordered pair $(\mathcal{A},\mathcal{B})$ of simple hypergraphs $\mathcal{A}$, $\mathcal{B}$ with properties $\emptyset \notin \mathcal{A}\cup\mathcal{B}$ and $\forall A \in \mathcal{A}, \forall B \in \mathcal{B} : A \cap B \neq \emptyset$. A *semicoterie* is a bicoterie $(\mathcal{A},\mathcal{B})$ where $\mathcal{A}$ is a coterie. Thus a semicoterie $(\mathcal{A},\mathcal{B})$ models the write quorums by $\mathcal{A}$ and the read quorums by $\mathcal{B}$.

The notion of domination for bicoteries and semicoteries is similar to the one for coteries.

Definition 7.3 *A bicoterie $(\mathcal{A},\mathcal{B})$ dominates a bicoterie $(\mathcal{A}',\mathcal{B}')$ iff $\mathcal{A} \neq \mathcal{A}'$ or $\mathcal{B} \neq \mathcal{B}'$, and $\forall A' \in \mathcal{A}' : \exists A \in \mathcal{A} : A' \subseteq A$ and $\forall B' \in \mathcal{B}' : \exists B \in \mathcal{B} : B' \subseteq B$, i.e., $\mathcal{A}' \leq \mathcal{A}$ and $\mathcal{B}' \leq \mathcal{B}$. A bicoterie is nondominated (ND) iff it is not dominated by some bicoterie.*

Since semicoteries are bicoteries, nondomination for them is defined implicitly by the definition for bicoteries. If a coterie $\mathcal{C}$ is identified with the equivalent bicoterie $(\mathcal{C},\mathcal{C})$, the inclusion hierarchy depicted in Figure 3 is obvious. Trivially, both inclusions are proper, i.e., COTERIES $\subset$ SEMICOTERIES $\subset$ BICOTERIES.

As in the case of coteries, ND semicoteries are to prefer in practice, and an efficient algorithm for recognition of ND semicoteries is desired. The complexity of this task is a simple corollary to the following Theorem:

Theorem 7.11 *A bicoterie $B = (\mathcal{A},\mathcal{B})$ is ND iff $\mathcal{A} = Tr(\mathcal{B})$.*

Proof: Recall that bicoterie $B' = (\mathcal{A}',\mathcal{B}')$, dominates B iff $B \neq B'$ and $\mathcal{A} \geq \mathcal{A}'$ and $\mathcal{B} \geq \mathcal{B}'$ holds. Note that in this case every edge $E \in \mathcal{A}'$ is a transversal of $\mathcal{B}$ (and every edge $E \in \mathcal{B}'$ is a transversal of $\mathcal{A}$). Assume that $\mathcal{A} \neq Tr(\mathcal{B})$. Since every edge $E \in \mathcal{A}$ is a transversal of $\mathcal{B}$, we have $\mathcal{A} \geq Tr(\mathcal{B})$, and clearly the bicoterie $(Tr(\mathcal{B}),\mathcal{B})$ dominates B. This proves the *only if* claim. For the *if direction*, if $\mathcal{A} = Tr(\mathcal{B})$, for every hypergraph $\mathcal{A}'$ that contains only transversals of $\mathcal{B}$ it holds that $\mathcal{A}' \geq \mathcal{A}$; as $\mathcal{B}$ is simple, analogously for every hypergraph $\mathcal{B}'$ of transversals of $\mathcal{A}$ the predicate $\mathcal{B}' \geq \mathcal{B}$ is valid. Thus a bicoterie B' that dominates B cannot exist. □

Corollary 7.12 *Checking nondomination of bicoteries and semicoteries is* SIMPLE-H-SAT-*complete.*

Proof: By Theorems 7.11 and 4.13. □

Note that checking for nondomination of semicoteries contains the respective problem for coteries as a subproblem. Both problems are with respect to polynomial transformations equivalent, thus read quorums do not increase the complexity of checking nondomination.

We conclude this subsection with a brief look at a more restrictive approach to mutual exclusion than coteries, namely vote assignable coteries.

A *vote assignment* to a set V of sites is a function $\nu : V \rightarrow \mathbf{N}$ ($\mathbf{N}$ is the nonnegative integers). The *majority* $MAJ(\nu)$ in vote assignment ν is defined as follows:

$$MAJ(\nu) = \begin{cases} TOT(\nu)/2 + 1 & \text{if } TOT(\nu) \text{ is even,} \\ \frac{TOT(\nu)+1}{2} & \text{if } TOT(\nu) \text{ is odd} \end{cases}$$

where $TOT(\nu) = \sum_{x \in V} \nu(x)$. Intuitively, $\nu(x)$ is the "weight" of site x in voting for an operation execution. If the sum of the weights of a group of nodes outweights the majority, the operation may be performed.

Every vote assignment ν to vertices V corresponds with the coterie

$$\mathcal{C}_\nu = \min(\{V' \subseteq V \mid \sum_{x \in V'} \nu(x) \geq MAJ(\nu)\}).$$

However, a *vote assignable* coterie has not a unique voting assignment, and many coteries have no voting assignment at all.

Investigating the complexity of conversion between vote assignments and ND coteries, it is not difficult to give an algorithm that outputs all edges of the coterie corresponding to a vote assignment with polynomial delay. Conversely, the determination of a voting assignment (resp. if there is any) for a ND coterie is easily reducible to a linear programming instance, and is thus solvable in polynomial time. However, the recognition problem for ND vote assignable coteries, which is from the results in [GMB85] easily proved equivalent to the question whether a coterie has a vote assignment where TOT is odd, seems not to have such a simple solution. The complexity of this problem is of interest with respect to the recognition of ND coteries, because a vote assignment is a strong syntactical restriction, as the portion of vote assignable coteries is very small cf. [IK90].

8 Problems closely related to SIMPLE-H-SAT in Switching Theory and AI

8.1 Boolean Switching Theory

In this section we turn to switching circuits and show the consequences of the complexity of SIMPLE-H-SAT for some problems on monotone Boolean functions. We assume that the reader is familiar with the basic concepts of switching circuits and Boolean functions; for details he is referred to the standard literature (e.g., [Weg87]). Briefly, every *n-ary Boolean function* is a mapping $f : \mathcal{B}_n \to \{0,1\}$, where $\mathcal{B}_n$ denotes $\{0,1\}^n$. Every BF is computable by some logical circuit with gates implementing logical and, logical or, and negation. Logical circuits can be described accordingly by well-formed *Boolean expressions* over variables $x_1, \ldots, x_n$ and the logical connectives "$\wedge$" (and), "$\vee$" (or), and "$\neg$" (negation).

The monotone BFs constitute one of the most important subclasses of the BFs. Let $a, b \in \mathcal{B}_n, a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, bn)$. The partial ordering $\leq$ on $\mathcal{B}_n$ is defined $a \leq b$ iff $a_i = 1 \Rightarrow b_i = 1, i \in \{1, \ldots, n\}$. A BF f is called *monotone* if it satisfies $\forall a, b \in \mathcal{B}_n : f(a) = 1 \wedge a \leq b \Rightarrow f(b) = 1$. Every monotone BF except tautology and contradiction is computable by a circuit that involves only and-gates and or-gates, thus it can be described by a Boolean expression without use of "$\neg$". Conversely, every Boolean expression of this form represents a monotone Boolean function. Checking from the full function table whether a Boolean function is monotone is easily possible in polynomial time, even if only those entries $t \in \mathcal{B}_n$ are given with $f(t) = 1$. Since the size of the function table is always exponential in n, we wonder if we can do more efficient with an algorithm that checks for monotonicity directly from the circuit description without computing the full function table first. However, this is unlikely since this problem is easily proved co-**NP**-complete.

Boolean polynomials, which correspond to a certain class of Boolean expressions, are of great importance for the description of BFs. A *Boolean polynomial* is simply a disjunction of distinct conjunctive clauses, i.e., conjunctions (monoms) of (possibly negated) variables, and it is a *minimal polynomial* for the BF f defined by it, if there is no other polynomial with less monoms that represents f. In general, a BF may have several minimal polynomials; the minimal polynomial of a monotone BF, however, is unique.

The computation of a minimal polynomial for a BF f is in essential to determine prime implicants of f. A monom m is an *implicant* of f if and only if $m \Rightarrow E_f$ is a tautology, where E_f is a Boolean expression for f, and m is called *prime implicant* of f, if it is no longer an implicant of f if any literal is removed from it. Every minimal polynomial of a BF contains only prime implicants.

The usual approach to compute a minimal polynomial for a BF proceeds in two steps: First, all prime implicants are computed and thereafter a minimal polynomial (usually under some cost criterion) is constructed from them. Note that the second step is unnecessary if the function is monotone. The design of efficient algorithms for the computation of prime implicants from a function table or a logical expression has been topic of research over decades [Qui59, McC56, SCL70, HCV85, Mar90, JP90, KT90].

An important task in switching theory is to check if a logical circuit computes a certain BF described by a logical expression. Instead of computing the whole function table for both the circuit and the function first and checking thereafter if the tables are identical, it would be desirable to do the test directly from the circuit description avoiding the exponential step of table computation. Unfortunately, **NP**-completeness of SAT implies thus immediately that checking if a logical circuit computes a certain BF f is co-**NP**-complete in general, if f is given by a minimal polynomial. For monotone circuits, things are no better:

Problem:	MONOTONE CIRCUIT REPRESENTATION
Instance:	A Boolean expression E on variables $X = \{x_1, \ldots, x_n\}$ describing a monotone logical circuit, a minimal polynomial p_f of some monotone BF f.
Question:	Is E equivalent to p_f ?

Theorem 8.1 MONOTONE CIRCUIT REPRESENTATION *is co-***NP***-complete.*

Proof: Membership in co-**NP** is clear. Note that since the circuit has no negation-gates, it can neither compute the tautology nor the contradiction function. We prove co-**NP**-completeness by a polynomial time transformation of a subproblem of co-SAT that is still co-**NP**-complete into this problem. Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be an instance of SAT. Without loss of **NP**-completeness we may assume that for every variable x, there are at least two clauses $C, C' \in \mathcal{C}$ such that $x \notin C$ and $\neg x \notin C'$, because otherwise $\mathcal{C}$ is clearly satisfiable, and checking if one of the possible $2n$ cases therefor is present can be done in polynomial time. We introduce for each negated literal $\neg x_i, 1 \leq i \leq n$, a new variable y_i and substitute y_i for $\neg x_i$ in $\mathcal{C}$. Since the resulting set $\mathcal{C}'$ of clauses is trivially equivalent to a Boolean expression E' with no occurrence of "$\neg$", it defines a monotone function. Hence the expression $E'' = E' \vee \bigvee_{i=1}^{n} x_i \wedge y_i$ describes a monotone function f'' as well. The above restriction imposed on $\mathcal{C}$ makes sure that neither x_i nor y_i is an implicant of f''. Since every monom $x_i y_i, 1 \leq i \leq n$, is an implicant of F'', this implies that $x_i y_i$ is even a prime implicant of f''. It is not difficult to verify that $\mathcal{C}$ is not satisfiable if and only if $p = \bigvee_{i=1}^{n} x_i y_i$ is the minimal polynomial of f''. Since the transformation described is obviously computable in polynomial time, the proof is complete. $\Box$

Though recognizing the prime implicants of a monotone BF from a Boolean expression is intractable in general, there is no proof of intractability for a certain subproblem which occurs in the context of normal form conversions. A Boolean expression E is in *Minimum Disjunctive Normal Form* (MDNF) if it describes a minimal polynomial with smallest number of monoms for some BF. Clearly, the MDNF expression for a monotone BF is unique. Dual to

MDNF, the representation of a BF by an expression in *Minimum Conjunctive Normal Form* is common. An expression is in this form if it is a conjunction of (disjunctive) clauses such that there is no other other conjunction with less clauses equivalent to this expression. As for MDNF, every monotone BF has a unique MCNF-expression, which contains no negated literal (cf. [Weg87]).

The relationship between MDNF and MCNF of monotone Boolean functions is expressed by

Theorem 8.2 *Let f be an n-ary monotone Boolean function on variables $X = \{x_1, \ldots, x_n\}$, let $\mathcal{C}$ be the hypergraph on X with the clauses of the MCNF expression for f as edges and let $\mathcal{D}$ be the hypergraph on X with the variable sets of the monoms in the MDNF of f as edges. Then $\mathcal{D} = Tr(\mathcal{C})$.*

Proof: Clearly, $\mathcal{C}$ and $\mathcal{D}$ are simple hypergraphs. For tautology and contradiction we have $\mathcal{C} = \{\emptyset\}$, $\mathcal{D} = \emptyset$ and $\mathcal{C} = \emptyset$, $\mathcal{D} = \{\emptyset\}$, respectively, thus the claim is true in these cases. For the rest, consider $D \in \mathcal{D}$. Every truth value assignment in which all variables in D have value true satisfies the MDNF expression, hence also the MCNF expression. Since f is monotone, no negated literal occurs in $\mathcal{C}$, hence every $C \in \mathcal{C}$ must have nonempty intersection with D. Consequently, D is a transversal of $\mathcal{C}$. D is a prime implicant, i.e., no proper subset $D' \subset D$ describes an implicant, which implies that D' must have empty intersection with some $C \in \mathcal{C}$. Thus D is a minimal transversal of $\mathcal{C}$, and it follows that $\mathcal{D} \subseteq Tr(\mathcal{C})$. However, every minimal transversal T of $\mathcal{C}$ constitutes an implicant which is, moreover, a prime implicant of f, thus $Tr(\mathcal{C}) \subseteq \mathcal{D}$, and the claim is proved. $\Box$

However, if the circuits implement the MCNF expression, the complexity of MONOTONE CIRCUIT REPRESENTATION is related to recognizing the transversal hypergraph:

Theorem 8.3 MONOTONE CIRCUIT REPRESENTATION *restricted to expressions E in minimal conjunctive normal form (i.e., the recognition of monotone prime implicants from the MCNF) is* SIMPLE-H-SAT-*complete.*

Proof: By Theorem 8.2 the literal sets of the monoms of the minimal polynomial p_f given for input have to constitute the transversal hypergraph of the set of clauses to which E corresponds; on the other hand, this condition is sufficient for equivalence. Thus the equivalence of TRANSVERSAL HYPERGRAPH, which is SIMPLE-H-SAT-complete, and the problem considered with respect to polynomial time transformability is obvious. $\Box$

Note that Theorem 8.2 implies that the determination of the prime implicants (and hence the MDNF) of a monotone BF from its MCNF expression is inherently exponential (there may be exponentially many), but leaves hope for an output polynomial algorithm. We remark that in the general MONOTONE CIRCUIT REPRESENTATION problem the Boolean expression describing the monotone circuit can be considered as an intensional description of the corresponding MCNF expression, which can have size exponential in the size of the circuit description. The problem is intractable in general, but if the power of the description formalism is diminished, it becomes SIMPLE-H-SAT-complete.

Obviously, the problem inverse to computing prime implicants from a MCNF expression, i.e., the determination of a MCNF expression from the prime implicants of a BF, is also of interest. It is trivial to show that the related recognition problem (MONOTONE MCNF RECOGNITION) is SIMPLE-H-SAT-complete for monotone BFs. Note that any algorithm to compute the prime implicants of a monotone BF from a MDNF expression can be used for the inverse problem without any modification. However, if SIMPLE-H-SAT is co-**NP**-complete, there is no such algorithm that is efficient with respect to the output size to compute the MCNF of a monotone BF unless $\mathbf{P} = \mathbf{NP}$.

8.2 Model-Based Diagnosis

A field where an efficient solution of TRANSVERSAL HYPERGRAPH is of major interest is the theory of model-based diagnosis. During the last decade, diagnosing faults in technical systems by means of AI techniques has achieved high attention. Basically, there are two approaches. In the heuristic (also termed rule-based) approach, the diagnostic process is performed by an expert system which captures the knowledge of a diagnosis expert in its knowledge base and essentially aims to simulate the expert's reasoning. Model-based diagnosis, however, takes a logical description of the technical system and the behavior of its components as a basis for diagnosing faults by means of consistency. Since model-based diagnosis offers several advantages, much research effort has gone into this area in the recent past (rf., e.g.,[AAA90]).

To introduce the necessary concepts briefly ([Rei87, dKW87]), a *system* is a pair $(SD, COMP)$, where SD, the *system description*, is a set of usually first-order sentences and $COMP$ is a set of constants which model the system *components*. This general system description is used together with a set OBS of first-order sentences, which are particular *observations* on the system behavior, to diagnose faults. The system description SD makes use of a distinguished predicate $AB(c)$ which interprets "component c operates in abnormal mode". Now a *diagnosis* for $(SD, COMP, OBS)$ is a minimal set $\Delta \subseteq COMP$ of components such that

$$\mathcal{T} = SD \cup OBS \cup \{AB(c) \mid c \in \Delta\} \cup \{\neg AB(c) \mid c \in COMP - \Delta\}$$

is consistent.[4] Of course, if there is no fault, $\Delta = \emptyset$ must be a diagnosis, otherwise SD is not a sound system description.

In [Rei87] a characterization of diagnoses in terms of so called conflict sets is given. A *conflict set* is a set $C \subseteq COMP$ such that $SD \cup OBS \cup \{\neg AB(c) \mid c \in C\}$ is inconsistent. A conflict

[4]It is presupposed that $\mathcal{T}$ is always in a decidable subclass of first-order predicate calculus.

set C is minimal if and only if no proper subset of C is a conflict set. The fundamental theorem on conflict sets and diagnoses in [Rei87] put into hypergraph terminology is

Theorem 8.4 ([Rei87]) *$\Delta \subseteq COMP$ is a diagnosis for $(SD, COMP, OBS)$ if and only if $\Delta \in Tr(\mathcal{C})$, where $\mathcal{C}$ denotes the set of all minimal conflict sets of $(SD, COMP, OBS)$.*

Neglecting the costs of consistency checking, a diagnosis for $(SD, COMP, OBS)$ can be found in polynomial time: Let $C = COMP$, and test if C is a superset of a diagnosis. Subsequently remove any component c from C such that $C - \{c\}$ is also superset of a diagnosis. Repeating this elimination step until no longer possible, the procedure stops with C as a diagnosis.

Finding an additional diagnosis, however, i.e., given $(SD, COMP, OBS)$ and a family $\mathcal{D}$ of diagnoses, decide if there is another diagnosis not in $\mathcal{D}$, is proved **NP**-complete [FGN90].

Utilizing the relationship in Theorem 8.4, Reiter describes an algorithm to compute all diagnosis for $(SD, COMP, OBS)$ which employs a theorem prover TP that returns by call of $TP(SD, OBS, COMP')$, where $COMP' \subseteq COMP$, a (not necessarily minimal) conflict set for $(SD, COMP', OBS)$ [Rei87, GSW90]. There is no space to present and discuss this algorithm here. We just remark that it is not output-polynomial, even if the calls of the theorem prover are not taken into account and, moreover, even if the theorem prover returns only minimal conflict sets (otherwise, an instance disproving output-polynomiality is trivially found). In proof of this an appropriate computation sequence can be constructed from the hypergraph in Section 2 which we used to show exponentiality of the simple transversal computation method.

SIMPLE-H-SAT-completeness of TRANSVERSAL HYPERGRAPH and Theorem 8.4 imply immediately the following theorem:

Theorem 8.5 *Let $\mathcal{C}$ be the family of minimal conflict sets of $(SD, COMP, OBS)$, and $\mathcal{D}$ a family of diagnoses for $(SD, COMP, OBS)$. Given $\mathcal{C}$ and $\mathcal{D}$ for input, deciding if there is an additional diagnosis not contained in $\mathcal{D}$ is co-*SIMPLE-H-SAT*-complete.*

Since the determination of the minimal conflict sets is essential in popular algorithms to compute diagnoses [Rei87, dKW87], the complexity of the additional diagnosis problem under the assumption that all minimal conflict sets are included in the input is of crucial interest with respect to the output complexity of the problem at all.

Note that $SD \cup OBS$ states an intensional description of the hypergraph of minimal conflict sets for $(SD, COMP, OBS)$. It is not difficult to show that the number of minimal conflict sets can be exponential in the size of $(SD, COMP, OBS)$. For example, let $SD = \{\neg AB(c_{2i-1}) \wedge \neg AB(c_{2i}) \Rightarrow Z_i \mid 1 \leq i \leq n\} \cup \{Z_i \Rightarrow S \mid 1 \leq i \leq n\}, COMP = \{c_1, \ldots, c_n\}, OBS = \{\neg S\}$, where $Z_1, \ldots, Z_n, S$ are propositional variables as a shorthand for predicates on some constants. The set of minimal conflict sets, $\mathcal{CS} = \{C \subseteq COMP \mid c_i \in$

C iff $c_{2i-1} \notin C, 1 \leq i \leq n\}$, has 2^n elements. Thus with respect to the **NP**-complete version with the intensional description of the conflict sets (given by $(SD, COMP, OBS)$) the input size of the respective problem instance for TRANSVERSAL HYPERGRAPH increases exponentially in the input size of $(SD, COMP, OBS)$. If this increase makes the problem become tractable, however, is carefully to be answered, as MINIMUM DISJUNCTIVE NORMAL FORM [Mas79] shows. This problem, given an n-ary Boolean function by a set of implicants, decide if there is a minimal polynomial for it with less than k monoms, remains **NP**-complete even if the full function table with 2^n entries is given for input.

9 Conclusions

We have studied two computational problems on hypergraphs in this paper, namely HYPERGRAPH SATURATION and the recognition of the transversal hypergraph (TRANSVERSAL HYPERGRAPH). The latter of these decision problems is closely related to the search problem of computing all minimal transversals of a hypergraph.

The complexity of computing resp. recognizing all minimal transversals of a hypergraph is an open problem [GMB85, MR87, DT87, JYP88]. We showed that checking saturation for a simple hypergraph, SIMPLE HYPERGRAPH SATURATION, is computationally equivalent to TRANSVERSAL HYPERGRAPH; without restriction to simple hypergraph, HYPERGRAPH SATURATION was proved co-**NP**-complete in its general version as well as for various subcases (see Table 1).

In the study of SIMPLE HYPERGRAPH SATURATION and TRANSVERSAL HYPERGRAPH, we investigated into relationships of these problems to well-studied problems such as satisfiability and hypergraph two-colorability. Several subproblems equivalent to SIMPLE HYPERGRAPH SATURATION and TRANSVERSAL HYPERGRAPH with respect to polynomiality were exhibited, the most important among them SELFTRANSVERSALITY, which has an application in concurrency control in distributed databases. Moreover, narrowing the open problems "frontier", we proved some generalizations of these two problems intractable, and we showed that several important subcases are in polynomial time decidable. These results base upon algorithms which enable the computation of all minimal transversals of a hypergraph in output polynomial total time under certain restrictions, the probably most important of them the case of bounded edge size (see Table 2).

Our results apply to various problems in database theory, switching theory, logic, and artificial intelligence which are all closely related to SIMPLE HYPERGRAPH SATURATION and TRANSVERSAL HYPERGRAPH as pointed out. The polynomial subcases of these problems in Table 3 are easily proved from the results in the previous sections.

Table 1: Hypergraph saturation (**SHSC** = SIMPLE-H-SAT-complete)

H-SAT $\mathcal{H} = (V, \mathcal{E})$	Restriction							
	none	$\mathcal{H}$ self complemented	no chain of length ≥ 3	$\lvert\mathcal{H}\rvert \leq k$	$\lvert H\rvert \leq k$, $H \in \mathcal{H}$	$\lvert\lvert H_1\rvert - \lvert H_2\rvert\rvert \leq k$, $H_1, H_2 \in \mathcal{H}$	$\mathcal{H}$ simple	$\mathcal{H}$ simple and self comple-mented
Complexity	co-**NP**-C	co-**NP**-C	co-**NP**-C	**P**	**P**	**P**	**SHSC**	**SHSC**

Table 2: Transversal Hypergraph computation ($k \ldots$ constant integer ≥ 0)

Hypergraph $\mathcal{H} = (V, \mathcal{E})$	Upper complexity of computation of $Tr(\mathcal{H})$
$\lvert\mathcal{H}\rvert \leq k$	input-polynomial
$r(\mathcal{H}) \leq k$	incremental polynomial
$\mathcal{H}$ is a graph	polynomial delay
$ar(\mathcal{H}) \geq \lvert V\rvert - k$	input-polynomial
β-acyclic	polynomial delay

Table 3: Transversal Hypergraph (**SHSC** = SIMPLE-H-SAT-complete, **O** = Open)

n ... number of vertices; k, c... integer constants

Problem	Complexity	Polynomial Subcases	Generalization
TRANS-HYP $\mathcal{H} = Tr(\mathcal{G})$?	**SHSC**	$s(\mathcal{H}) \geq n-k$; $r(\mathcal{G}) \leq k$; $\lvert\mathcal{G}\rvert \leq k$; $\mathcal{G}$ is β-acyclic; $r(\mathcal{H}) - s(\mathcal{H}) \leq k \wedge s(\mathcal{G}) + r(\mathcal{H}) = n + c$;	$Tr(\mathcal{G}) \geq \mathcal{H}$? (co-**NP**-C)
HP2C for simple, self-complemented hypergraphs	co-**SHSC**	$\lvert\mathcal{H}\rvert \leq k$; $n \leq k$; $\forall H \in \mathcal{H} : \lvert H\rvert \leq 2 \vee \lvert H\rvert \geq n-2$;	**HP2C** (**NP**-C)
FD-RELATION EQ. for relation R and relation scheme $RS = <U,F>$ in BCNF	**SHSC**	$\lvert F\rvert \leq k$; $\forall X \rightarrow Y \in F : \lvert X\rvert \geq \lvert U\rvert - k$; $\lvert R\rvert \leq k$; $\lvert U\rvert \leq k$; $\forall K \in Keys(RS) : \lvert K\rvert \leq k$;	**FD-RELATION EQ.** (O)
ADDITIONAL KEY for relation instance R on attributes U	co-**SHSC**	$\lvert R\rvert \leq k$; $\lvert U\rvert \leq k$; $\forall K \in Keys(R) : \lvert K\rvert \leq k$;	
ND COTERIE coterie $\mathcal{H}$ nondominated ? (i.e., $\mathcal{H} = Tr(\mathcal{H})$?)	**SHSC**	$\lvert\mathcal{H}\rvert \leq k$; $n \leq k$; $r(\mathcal{H}) \leq k$; $s(\mathcal{H}) \geq n-k$;	**co-HP2C** (co-**NP**-C)
MON. CIRCUIT REP. Expression E in MCNF, prime implicants (by polynomial p_f)	**SHSC**	$\lvert X\rvert \leq k$; $\lvert E\rvert \leq k$; $\forall C \in \mathcal{C}(E) : \lvert C\rvert \leq k$; $\forall$ prime implicant $PI : \lvert PI\rvert \leq k$;	**MON. CIRCUIT REPRESENTATION** (co-**NP**-C)
ADD. DIAGNOSIS Diagnoses $\mathcal{D}$, conflict sets $\mathcal{C}$	co-**SHSC**	$\lvert COMP\rvert \leq k$; $\lvert\mathcal{C}\rvert \leq k$; $\forall D \in \mathcal{D} : \lvert D\rvert \leq k$; $\forall C \in \mathcal{C} : \lvert C\rvert \leq k$;	**ADD. DIAGNOSIS** (co-**NP**-C)
MONOTONE SAT Clause set $\mathcal{C}$ on X satisfying $\forall C_1 \in \mathcal{C}^+, \forall C_2 \in \mathcal{C}^- : C_1 \cap C_2 \neq \emptyset$	co-**SHSC**	$* \in \{+,-\} : \forall C \in \mathcal{C}^* : \lvert C\rvert \geq \lvert X\rvert - k$; $* \in \{+,-\} : \forall C \in \mathcal{C}^* : \lvert C\rvert \leq k$; $\lvert X\rvert \leq k$	**MSAT** (**NP**-C)

For future research, we present the following open problems in the context of the complexity classification of SIMPLE HYPERGRAPH SATURATION and TRANSVERSAL HYPERGRAPH:

1. What is the complexity of SIMPLE HYPERGRAPH SATURATION resp. TRANSVERSAL HYPERGRAPH ? Are they in **P**, **NP**, are they co-**NP**-complete ? In connection with this, are the minimal transversals of a hypergraph $\mathcal{H}$ computable in output-polynomial total time ?

2. How are SIMPLE HYPERGRAPH SATURATION and TRANSVERSAL HYPERGRAPH related to other open decision problems (cf. [Joh89]) ? The most famous among these problems, GRAPH ISOMORPHISM, has been studied extensively in the literature, and it is known that if GRAPH ISOMORPHISM is **NP**-complete and $\mathbf{P} \neq \mathbf{NP}$, then the polynomial time hierarchy collapses at level two [Sch88], a result that most of the experts do not expect [Joh88]. However, GRAPH ISOMORPHISM seems not trivially interrelate to SIMPLE HYPERGRAPH SATURATION and TRANSVERSAL HYPERGRAPH.

3. What is the gain by using randomized algorithms, probabilistic algorithms, or interactive proof systems (cf. [Joh84, Joh88]) for SIMPLE HYPERGRAPH SATURATION and TRANSVERSAL HYPERGRAPH, and what about weaker forms of reductions than polynomial transformability (especially, randomized reductions, cf. [AM77, Sch83, VV86])? Schöning [Sch83] defines within **NP** a Low Hierarchy $L_0, L_1, \ldots$, where $L_0 = \mathbf{P}, L_1 = \mathbf{NP} \cap \text{co-}\mathbf{NP}$, and a High Hierarchy $H_0, H_1, \ldots$, where $H_0 = \mathbf{NP}$-complete, and the other classes $H_i, i > 0$, correspond to weakened versions of **NP**-completeness such as γ-completeness [AM77] (fully contained in H_1) etc. which are believed still strong enough that no problem in **P** satisfies one of them. Classifying co-SIMPLE-H-SAT into the Low or High Hierarchy would give strong evidence that the problem is not **NP**-complete or intractable, respectively.

Acknowledgments

The authors thank L. Lovász and M. Yannakakis for interesting discussions of this work and their valuable hints and comments. They would further like to thank H. Mannila for helpful suggestions, and they are grateful to D. A. Plaisted for pointing out that the subcase of pairwise resolvable clauses of SAT is decidable in polynomial time. Many thanks also to M. Stumptner for reading a previous version of this paper and to P. Marquis for his help in finding literature on the computation of prime implicants.

References

[AAA90] AAAI and Price Waterhouse. *Proceedings of the First International Workshop on Principles of Diagnosis*, July 23-25, Stanford University 1990.

[AL86] Ron Aharoni and Nathan Linial. Minimal Non-Two-Colorable Hypergraphs and Minimal Unsatisfiable Formulas. *Journal of Combinatorial Theory, Series A*, 43:196–204, 1986.

[AM77] L. Adleman and K. Manders. Reducibility, randomness, and intractability. In *Proceedings of the 9^{th} ACM Symposium on the Theory of Computing*, pages 151–163, 1977.

[Arm74] W. W. Armstrong. Dependency structures of data base relationships. In *IFIP Workshop Proceedings*, pages 580–583. North-Holland Publ. Comp., 1974.

[BB79] C. Beeri and P. A. Bernstein. Computational Problems Related to the Design of Normal Form Relational Schemes. *ACM Transactions on Database Systems*, 4(1):30–59, 1979.

[BDFS84] Catriel Beeri, Martin Down, Ronald Fagin, and Richard Statman. On the Structure of Armstrong Relations for Functional Dependencies. *Journal of the ACM*, 31(1):30–46, January 1984.

[BDK87] G. Burosch, J. Demetrovics, and G. O. H. Katona. The Poset of Closures as a Model of Changing Databases. *Order*, 4:127–142, 1987.

[BDM81] C. Batini, A. D'Atri, and M. Moscarini. Formal tools for top-down and bottom-up generation of acyclic relational database schemata. In *Proceedings 7^{th} Int. Conf. on Graph-Theoretic Concepts in Computer Science*, Linz, Austria 1981.

[Ben80] C. Benzaken. Critical Hypergraphs for the Weak Chormatic Number. *Journal of Combinatorial Theory, Series B*, 29:328–338, 1980.

[Ber89] Claude Berge. *Hypergraphs*, volume 45 of *North Holland Mathematical Library.* Elsevier Science Publishers B.V. (North-Holland), 1989.

[BFM$^+$81] C. Beeri, R. Fagin, D. Maier, A.O. Mendelzon, J.D. Ullman, and M. Yannakakis. Properties of acyclic database schemata. In *Proceedings 13^{th} ACM Symposium on the Theory of Computing*, pages 355–362, May 1981 1981.

[BFMY83] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of Acyclic Database Schemes. *Journal of the ACM*, 30(3):479–513, 1983.

[Coo85] S.A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[DGH74] D. E. Daykin, Jean Godfrey, and A. J. W. Hilton. Existence Theorems for Sperner Families. *Journal of Combinatorial Theory, Series A*, 17:245–251, 1974.

[dKW87] Johan de Kleer and Brian C. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32:97–130, 1987.

[DT87] J. Demetrovics and V. D. Thi. Keys, Antikeys and Prime Attributes. *Annales Univ. Sci. Budapest, Sect. Comp.*, 8:35–52, 1987.

[EL73] P. Erdös and L.Lovász. Problems and Results on 3-Chromatic Hypergraphs and some related Questions. *Colloquia Mathematica Societatis János Bolyai*, 10:609–627, 1973.

[Fag83] R. Fagin. Degrees of acyclity for hypergraphs and relational database schemes. *Journal of the ACM*, 30:514–550, 1983.

[FGN90] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Physical Negation instead of Fault Models. In *Proceedings AAAI-91*, July 1990.

[FMU82] R. Fagin, A.O. Mendelzon, and J.D. Ullman. A simplified universal relation assumption and its properties. *ACM Trans. on Database Syst.*, 7(3):343–360, 1982.

[Fu90] A. Fu. *Enhancing Concurrency and Availability for Database Systems*. PhD thesis, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6, 1990.

[FV84] J.-C. Fournier and M. Las Vergnas. A class of bichromatic hypergraphs. *Annals of Discrete Mathematics*, 21:21–27, 1984.

[GJ79] Michael Garey and David S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.

[GJS76] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-Complete Graph Problems. *Theoretical Computer Science*, 1:237–267, 1976.

[GKP89] Ronald Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison Wesley, 1989.

[GMB85] Hector Garcia-Molina and Daniel Barbara. How to Assign Votes in a Distributed System. *Journal of the ACM*, 32(4):841–860, 1985.

[Gra79] M.H. Graham. On the universal relation. Technical report, Univ. of Toronto, Toronto, Ont., Can., September 1979.

[GSW90] Russel Greiner, Barbara A. Smith, and Ralph W. Wilkerson. A Correction to the Algorithm in Reiter's Theory of Diagnosis. *Artificial Intelligence*, 41:79–88, 1990.

[HCV85] H. Hwang, D. Chao, and M. Valdez. A New Technique for the Minimization of Switching Functions. In *Proceedings IEEE Southeastern Conf.*, pages 299–304, 1985.

[IK90] Toshihide Ibaraki and Tiko Kameda. Theory of Coteries. Technical Report CSS/LCCR TR90–09, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6, August 1990.

[Ive62] Kenneth Iverson. *A Programming Language.* Wiley, 1962.

[Joh84] David S. Johnson. The NP-Completeness Column – An Ongoing Guide. *Journal of Algorithms*, 5(1):147–160, 1984. Column 10: Updates, Updates, Updates.

[Joh88] David S. Johnson. The NP-Completeness Column – An Ongoing Guide. *Journal of Algorithms*, 9(3):426–444, 1988. Column 21: Interactive Proof Systems for Fun and Profit.

[Joh89] David S. Johnson. The NP-Completeness Column – An Ongoing Guide. *Journal of Algorithms*, 11(1):144–151, 1989. Column 22: The Story So Far.

[JP90] Peter Jackson and John Pais. Computing Prime Implicants. In *Proceedings of the CADE*, 1990.

[JYP88] David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On Generating All Maximal Independent Sets. *Information Processing Letters*, 27:119–123, 1988.

[KT90] A. Kean and G. Tsiknis. An Incremental Method for Generating Prime Implicants. *Journal of Symb. Comp.*, 9:185–206, 1990.

[KW86] R.M. Karp and A. Wigderson. A Fast Parallel Algorithm for the Maximal Independent Set Problem. *Journal of the ACM*, 32:762–773, 1986.

[Lam78] L. Lamport. The implementation of reliable distributed multiprocess systems. *Comp. Networks*, 2:95–114, 1978.

[Law66] Eugene L. Lawler. Covering Problems: Duality Relations and a New Method of Solution. *SIAM J. Appl. Math.*, 14(5):1115–1132, 1966.

[LLK80] E. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. *SIAM J. Comp.*, 9:558–565, 1980.

[LO78] Cláudio L. Lucchesi and Sylvia Osborn. Candidate Keys for Relations. *Journal of Computer and System Sciences*, 17:270–279, 1978.

[Lov68] L. Lovász. On the Chromatic Number of Finite Set-Systems. *Acta Mathematica Acad. Sc. Hungaricae*, 19:59–67, 1968.

[Lov73] L. Lovász. Coverings and Colorings of Hypergraphs. In *Proceedings of the 4th Conference on Combinatorics, Graph Theory, and Computing*, pages 3–12, 1973.

[LT85] Nathan Linial and Michael Tarsi. Deciding Hypergraph 2-Colorability by H-Resolution. *Theoretical Computer Science*, 38:343–347, 1985.

[Mag59] Khaled Maghout. Sur la détermination des nombres de stabilité et du nombre chromatique d'un graphe. In *Comptes Rendus des Séances de l'Académie des Sciences*, volume 248, pages 3522–3523. Paris, Gauthier-Villars, 1959.

[Mai83] David Maier. *The Theory of Relational Databases.* Computer Science Press, Rockville, Md., 1983.

[Mar90] Pierre Marquis. A Note on Prime Implicants. Technical Report CRIN 90-R-089, Centre de Recherche en Informatique de Nancy, Campus Scientifique, Boîte Postale 239, 54506 Vandœvre-les-Nancy Cedex, June 1990.

[Mas79] William J. Masek. Some NP-Complete Set Covering Problems. manuscript, August 1979.

[McC56] E.L. Jr. McCluskey. Minimization of Boolean functions. *Bell Syst. Tech. J.*, 35:1417–1444, 1956.

[MR86] Heikki Mannila and Kari-Jouko Räihä. Design by Example: An Application of Armstrong Relations. *Journal of Computer and System Sciences*, 22(2):126–141, 1986.

[MR87] Heikki Mannila and Kari-Jouko Räihä. Dependency Inference. In *Proceedings of the* 13^{th} *VLDB*, pages 155–158, 1987.

[MR88a] Heikki Mannila and Kari-Jouko Räihä. Algorithms for Inferring Functional Dependencies. Technical Report A-1988-3, University of Tampere, Dept. of Comp. Sc., Series of Publ. A, April 1988.

[MR88b] Heikki Mannila and Kari-Jouko Räihä. Generating Armstrong databases for sets of functional and inclusion dependencies. manuscript, August 1988.

[MR89] Heikki Mannila and Kari-Jouko Räihä. Practical algorithms for finding prime attributes and testing normal forms. In *Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 128–133, 1989.

[Qui59] W.V.O. Quine. On cores and prime implicants of truth functions. *Am. Math. Monthly*, 66:755–760, 1959.

[Rei87] Raymond Reiter. A Theory of Diagnosis From First Principles. *Artificial Intelligence*, 32:57–95, 1987.

[Roy70] B. Roy. *Algébre moderne et Thèorie des graphes*, volume II. Dunod, Paris, 1970.

[Sch83] Uwe Schöning. A Low and a High Hierarchy within NP. *Journal of Computer and System Sciences*, 27:14–28, 1983.

[Sch88] Uwe Schöning. Graph Isomorphism is in the Low Hierarchy. *Journal of Computer and System Sciences*, 37:312–323, 1988.

[SCL70] J.R. Slagle, C.L. Chang, and R.C.T. Lee. A New Algorithm for Generating Prime Implicants. *IEEE Transactions on Computers*, C-19(4):304–310, 1970.

[Spe28] Emanuel Sperner. Ein Satz über Untermengen einer endlichen Menge. *Mathematische Zeitschrift*, XXVII:544–548, 1928.

[Thi86] V. D. Thi. Minimal Keys and Antikeys. *Acta Cybernetica*, 7(4):361–371, 1986.

[TIAS77] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all maximal independent sets. *SIAM J. Comp.*, 6:505–517, 1977.

[Tof74] B. Toft. Color-Critical Graphs and Hypergraphs. *Journal of Combinatorial Theory, Series B*, 16:145–161, 1974.

[TY84] Robert E. Tarjan and Mihalis Yannakakis. Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comp.*, 13(3):566–579, 1984.

[Ull88] J. D. Ullman. *Principles of Database and Knowledge Base Systems*, volume 2. Computer Science Press, 1988.

[Val79] L. G. Valiant. The Complexity of Computing the Permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[VV86] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(1):85–93, 1986.

[Weg87] Ingo Wegener. *The Complexity of Boolean Functions*. Wiley - Teubner series in computer science. Wiley and Sons Ltd, and Teubner, Stuttgart, 1987.

[Yan82] Mihalis Yannakakis. Algorithms for acyclic database schemes. In *Proceedings* 7^{th} *Conf. on Very Large Database Systems*, pages 82–94, Cannes, France 1982.