New Results on Monotone Dualization and Generating Hypergraph Transversals

Thomas Eiter
Institut f. Informationssysteme
Technische Universität Wien
Favoritenstraße 9-11, A-1040
Vienna, Austria
eiter@kr.tuwien.ac.at

Georg Gottlob
Institut f. Informationssysteme
Technische Universität Wien
Favoritenstraße 9-11, A-1040
Vienna, Austria
gottlob@dbai.tuwien.ac.at

Kazuhisa Makino*
Graduate School of Engineering
Science, Osaka University
Toyonaka, 560-8531
Osaka, Japan
makino@sys.es.osaka-u.ac.jp

ABSTRACT

This paper considers the problem of dualizing a monotone CNF (equivalently, computing all minimal transversals of a hypergraph), whose associated decision problem is a prominent open problem in NP-completeness. We present a number of new polynomial time resp. output-polynomial time results for significant cases, which largely advance the tractability frontier and improve on previous results. Furthermore, we show that duality of two monotone CNFs can be disproved with limited nondeterminism (more precisely, in polynomial time with $O(\log^2 n)$ suitably guessed bits). This result sheds new light on the complexity of this important problem.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*; F.1.3 [**Theory of Computation**]: Complexity Measures and Classes; G.2.1 [**Discrete Mathematics**]: Combinatorics—*Combinatorial algorithms*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms, Hypergraphs*

General Terms

Algorithms, Theory

Keywords

Dualization, transversal computation, output-polynomial algorithms, combinatorial enumeration, treewidth, hypergraph acyclicity, limited nondeterminism

1. INTRODUCTION

*Part of the work carried out while visiting TU Wien.

STOC'02, May 19-21, 2002, Montreal, Quebec, Canada.

Recall that the prime CNF of a monotone Boolean function f is the unique formula $\varphi = \bigwedge_{c \in S} c$ in conjunctive normal form where S is the set of all prime implicates of f, i.e., minimal clauses c which are logical consequences of f. In this paper, we consider the following problem:

	Problem DUALIZATION
Input:	The prime CNF φ of a monotone Boolean function $f = f(x_1, \ldots, x_m)$.
Output:	The prime CNF ψ of its dual $f^d = \overline{f}(\overline{x}_1, \ldots, \overline{x}_m)$.

It is well known that problem DUALIZATION is equivalent to the TRANSVERSAL COMPUTATION problem, which requests to compute the set of all minimal transversals (i.e., minimal hitting sets) of a given hypergraph $\mathcal{H}$, in other words, the *transversal hypergraph* $Tr(\mathcal{H})$ of $\mathcal{H}$. Actually, these problems can be viewed as the same problem, if the clauses in a monotone CNF φ are identified with the sets of variables they contain. DUALIZATION is a search problem; the associated decision problem DUAL is to decide whether two given monotone prime CNFs φ and ψ represent a pair (f, g) of dual Boolean functions. Analogously, the decision problem TRANS-HYP associated with TRANSVERSAL COMPUTATION is deciding, given hypergraphs $\mathcal{H}$ and $\mathcal{G}$, whether $\mathcal{G} = Tr(\mathcal{H})$.

DUALIZATION and several problems which are like transversal computation known to be computationally equivalent to DUALIZATION (see [13]) are of interest in various areas such as database theory (e.g., [34, 43]), machine learning and data mining (e.g., [4, 5, 10, 18]), game theory (e.g., [22, 38, 39]), artificial intelligence (e.g., [17, 24, 25, 40]), mathematical programming (e.g., [3]), and distributed systems (e.g., [16, 23]) to mention a few.

While the output CNF ψ can be exponential in the size of φ, it is currently not known whether ψ can be computed in *output-polynomial* (or *polynomial total*) *time*, i.e., in time polynomial in the combined size of φ *and* ψ. Any such algorithm for DUALIZATION (or TRANSVERSAL COMPUTATION) would significantly advance the state of the art of many problems in the application areas. Similarly, the complexity of Dual and Trans-Hyp is open since more than 20 years now (cf. [2, 13, 26, 27, 29]).

Note that Dualization is solvable in polynomial total time on a class $\mathcal{C}$ of hypergraphs iff Dual is in PTIME for all pairs $(\mathcal{H}, \mathcal{G})$, where $\mathcal{H} \in \mathcal{C}$ [2]. Dual is known to be in co-NP and the best currently known upper time-bound is $n^{o(\log n)}$ [15]. Determining the complexities of Dualization and Dual, and of equivalent problems such as the transversal problems, is a prominent open problem. This is witnessed by the fact that these problems are cited in a rapidly growing body of literature and have been referenced in various survey papers and complexity theory retrospectives, e.g. [26, 30, 36].

Given the importance of monotone dualization and equivalent problems for many application areas, and given the long standing failure to settle the complexity of these problems, emphasis was put on finding tractable cases of Dual and corresponding polynomial total-time cases of Dualization. In fact, several relevant tractable classes were found by various authors; see e.g. [6, 7, 8, 10, 12, 13, 31, 32, 35, 37] and references therein. Moreover, classes of formulas were identified on which Dualization is not just polynomial total-time, but where the conjuncts of the dual formula can be enumerated with *incremental polynomial delay*, i.e., with delay polynomial in the size of the input plus the size of all conjuncts so far computed, or even with *polynomial delay*, i.e., with delay polynomial in the input size only.

Main Goal. The main goal of this paper is to present important new polynomial total time cases of Dualization and, correspondingly, PTIME solvable subclasses of Dual which significantly improve previously considered classes. Towards this aim, we first present a new algorithm Dualize and prove its correctness. Dualize can be regarded as a generalization of a related algorithm proposed by Johnson, Yannakakis, and Papadimitriou [27]. As other dualization algorithms, Dualize reduces the original problem by self-reduction to smaller instances. However, the subdivision into subproblems proceeds according to a particular order which is induced by an arbitrary fixed ordering of the variables. This, in turn, allows us to derive some bounds on intermediate computation steps which imply that Dualize, when applied to a variety of input classes, outputs the conjuncts of ψ with polynomial delay or incremental polynomial delay. In particular, we show positive results for the following input classes:

- **Degenerate CNFs.** We generalize the notion of k-degenerate graphs [44] to hypergraphs and define *k-degenerate monotone CNFs* resp. *hypergraphs*. We prove that for any constant k, Dualize works with polynomial delay on k-degenerate inputs. Moreover, it works in output-polynomial time on $O(\log n)$-degenerate CNFs.

- **Read-k CNFs.** A CNF is *read-k*, if each variable appears at most k times in it. We show that for read-k CNFs, problem Dualization is solvable with polynomial delay, if k is constant, and in total polynomial time, if $k = O(\log(\|\varphi\|)$. Our result for constant k significantly improves upon the previous best known algorithm [10], which has a higher complexity bound, is not polynomial delay, and outputs the clauses of ψ in no specific order. The result for $k = O(\log \|\varphi\|)$ is a non-trivial generalization of the result in [10], which was posed as an open problem [9].

- **Acyclic CNFs.** There are several notions of hypergraph resp. monotone CNF acyclicity [14], where the most general and well-known is α-acyclicity. As shown in [13], Dualization is polynomial total time for β-acyclic CNFs; β-acyclicity is the hereditary version of α-acyclicity and far less general. A similar result for α-acyclic prime CNFs was left open. We give a positive answer and show that for α-acyclic prime φ, Dualization is solvable with polynomial delay.

- **Formulas of Bounded Treewidth.** The *treewidth* [41] of a graph expresses its degree of cyclicity. Treewidth is an extremely general notion, and bounded treewidth generalizes almost all other notions of near-acyclicity. Following [11], we define the treewidth of a hypergraph resp. monotone CNF φ as the treewidth of its associated (bipartite) variable-clause incidence graph. We show that Dualization is solvable with polynomial delay (exponential in k) if the treewidth of φ is bounded by a constant k, and in polynomial total time if the treewidth is $O(\log \log \|\varphi\|)$.

- **Recursive Applications of Dualize and k-CNFs.** We show that if Dualize is applied recursively and the recursion depth is bounded by a constant, then Dualization is solved in polynomial total time. We apply this to provide a simpler proof of the known result [6, 13] that monotone k-CNFs (where each conjunct contains at most k variables) can be dualized in output-polynomial time.

After deriving the above results, we turn our attention (in Section 5) to the fundamental computational nature of problems Dual and Trans-Hyp in terms of complexity theory.

Complexity: Limited nondeterminism. In a landmark paper, Fredman and Khachiyan [15] proved that problem Dual can be solved in quasi-polynomial time. More precisely, they first gave an algorithm A solving the problem in $n^{O(\log^2 n)}$ time, and then a more complicated algorithm B whose runtime is bounded by $n^{4\chi(n)}$ where $\chi(n)$ is defined by $\chi(n)^{\chi(n)} = n$. As noted in [15], $\chi(n) \sim \log n / \log \log n = o(\log n)$; therefore, duality checking is feasible in $n^{o(\log n)}$ time. This is the best upper bound for problem Dual so far, and shows that the problem is most likely not NP-complete.

A natural question is whether Dual lies in some lower complexity class based on other resources than just runtime. In the present paper, we advance the complexity status of this problem by showing that its complement is feasible with *limited nondeterminism*, i.e, by a nondeterministic polynomial-time algorithm that makes only a poly-logarithmic number of guesses. For a survey on complexity classes with limited nondeterminism, and for several references, see [19]. We first show by a simple and self-contained proof that testing non-duality is feasible in polynomial time with $O(\log^3 n)$ nondeterministic steps. We then observe that this can be improved to $O(\log^2 n)$ nondeterministic steps. *This result is surprising, because most researchers dealing with the complexity of* DUAL *and* TRANS-HYP *believed so far that these problems are completely unrelated to limited nondeterminism.*

We believe that the results presented in this paper are significant, and we are confident they will prove useful in various contexts. First, we hope that the various polynomial/output-polynomial cases of the problems which we identify will lead to better and more general methods in various application areas (as we show, e.g. in learning and data mining [10]), and that based on the algorithm DUALIZE or some future modifications, further relevant tractable classes will be identified. Second, we hope that our discovery on limited nondeterminism provides a new momentum to complexity research on DUAL and TRANS-HYP, and will push it towards settling these longstanding open problems.

2. PRELIMINARIES AND NOTATION

A *Boolean function* (in short, *function*) is a mapping $f : \{0,1\}^n \to \{0,1\}$, where $v \in \{0,1\}^n$ is called a *Boolean vector* (in short, *vector*). As usual, we write $g \leq f$ if f and g satisfy $g(v) \leq f(v)$ for all $v \in \{0,1\}^n$, and $g < f$ if $g \leq f$ and $g \neq f$. A function f is *monotone* (or *positive*), if $v \leq w$ (i.e., $v_i \leq w_i$ for all i) implies $f(v) \leq f(w)$ for all $v, w \in \{0,1\}^n$. Boolean variables $x_1, x_2, \ldots, x_n$ and their complements $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ are called *literals*. A *clause* (resp., *term*) is a disjunction (resp., conjunction) of literals containing at most one of x_i and $\bar{x}_i$ for each variable. A clause c (resp., term t) is an *implicate* (resp., *implicant*) of a function f, if $f \leq c$ (resp., $t \leq f$); moreover, it is *prime*, if there is no implicate $c' < c$ (resp., no implicant $t' > t$) of f, and *monotone*, if it consists of positive literals only. We denote by $PI(f)$ the set of all prime implicants of f.

A *conjunctive normal form* (CNF) (resp., disjunctive normal form, DNF) is a conjunction of clauses (resp., disjunction of terms); it is *prime* (resp. *monotone*), if all its members are prime (resp. *monotone*). For any CNF (resp., DNF) ρ, we denote by $|\rho|$ the number of clauses (resp., terms) in it. Furthermore, for any formula φ, we denote by $V(\varphi)$ the set of variables that occur in φ, and by $\|\varphi\|$ its *length*, i.e., the number of literals in it.

As well-known, a function f is monotone iff it has a monotone CNF. Furthermore, all prime implicants and prime implicates of a monotone f are monotone, and it has a unique prime CNF, given by the conjunction of all its prime implicates. For example, the monotone f such that $f(v) = 1$ iff $v \in \{(1100), (1110), (1101), (0111), (1111)\}$ has the unique prime CNF $\varphi = x_2(x_1 \vee x_3)(x_1 \vee x_4)$.

Recall that the *dual* of a function f, denoted f^d, is defined by $f^d(x) = \overline{f}(\overline{x})$, where $\overline{f}$ and $\overline{x}$ is the complement of f and x, respectively. By definition, we have $(f^d)^d = f$. From De Morgan's law, we obtain a formula for f^d from any one of f by exchanging $\vee$ and $\wedge$ as well as the constants 0 and 1. For example, if f is given by $\varphi = x_1x_2 \vee \overline{x}_1(\overline{x}_3 \vee x_4)$, then f^d is represented by $\psi = (x_1 \vee x_2)(\overline{x}_1 \vee \overline{x}_3x_4)$. For a monotone f, let $\psi = \bigwedge_{c \in C}(\bigvee_{x_i \in c} x_i)$ be the prime CNF of f^d. Then by De Morgan's law, f has the (unique) prime DNF $\rho = \bigvee_{c \in C}(\bigwedge_{x_i \in c} x_i)$. Thus, we will regard DUALIZATION also as the problem of computing the prime DNF of f from the prime CNF of f.

3. ORDERED GENERATION OF TRANSVERSALS

In what follows, let f be a monotone function and φ its prime CNF, where we assume w.l.o.g. that all variables x_j $(j = 1, 2, \ldots n)$ appear in φ. Let φ_i $(i = 0, 1, \ldots, n)$ be the CNF obtained from φ by fixing variables $x_j = 1$ for all j with $j \geq i + 1$. By definition, we have $\varphi_0 = 1$ (truth) and $\varphi_n = \varphi$.

EXAMPLE 3.1. *Consider* $\varphi = (x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3 \vee x_4)(x_1 \vee x_4)$. *Then we have* $\varphi_0 = \varphi_1 = 1$, $\varphi_2 = (x_1 \vee x_2)$, $\varphi_3 = (x_1 \vee x_2)(x_1 \vee x_3)$, *and* $\varphi_4 = \varphi$.

Similarly, for the prime DNF

$$\psi \quad = \quad \bigvee_{t \in PI(f)} t \tag{1}$$

of f, we denote by ψ_i the DNF obtained from ψ by fixing variables $x_j = 1$ for all j with $j \geq i + 1$. Clearly, we have $\varphi_i \equiv \psi_i$, i.e., φ_i and ψ_i represent the same function denoted by f_i.

PROPOSITION 3.1. *Let φ and ψ be any CNF and DNF for f, respectively. Then,*

(a) $\|\varphi_i\| \leq \|\varphi\|$ *and* $|\varphi_i| \leq |\varphi|$, *and*

(b) $\|\psi_i\| \leq \|\psi\|$ *and* $|\psi_i| \leq |\psi|$, *for all* $i \geq 0$.

Denote by Δ^i $(i = 1, 2, \ldots, n)$ the CNF consisting of all the clauses in φ_i but not in φ_{i-1}.

EXAMPLE 3.2. *For the above example, we have* $\Delta^1 = 1$, $\Delta^2 = (x_1 \vee x_2)$, $\Delta^3 = (x_1 \vee x_3)$, *and* $\Delta^4 = (x_2 \vee x_3 \vee x_4)(x_1 \vee x_4)$.

Note that $\varphi_i = \varphi_{i-1} \wedge \Delta^i$; hence, for all $i = 1, 2, \ldots, n$ we have

$$\psi_i \quad \equiv \quad \psi_{i-1} \wedge \Delta^i \quad \equiv \bigvee_{t \in PI(f_{i-1})} (t \wedge \Delta^i). \tag{2}$$

Let $\Delta^i[t]$, for $i = 1, \ldots, n$ denote the CNF consisting of all the clauses c such that c contains no literal in t_{i-1} and $c \vee x_i$ appears in Δ^i. For example, if $t = x_2x_3x_4$ and $\Delta^4 = (x_2 \vee x_3 \vee x_4)(x_1 \vee x_4)$, then $\Delta^4[t] = x_1$. It follows from (2) that for all $i = 1, 2, \ldots, n$

$$\psi_i \quad \equiv \bigvee_{t \in PI(f_{i-1})} \Big((t \wedge \Delta^i[t]) \ \vee \ (t \wedge x_i) \Big). \tag{3}$$

LEMMA 3.2. *For any term* $t \in PI(f_{i-1})$, *let* $g_{i,t}$ *be the function represented by* $\Delta^i[t]$. *Then* $|PI(g_{i,t})| \leq |\psi_i| \leq |\psi|$.

PROOF. Let $V = \{x_1, x_2, \ldots, x_n\}$ and let $s \in PI(g_{i,t})$. Then by (3), $t \wedge s$ is an implicant of ψ_i. Hence, some $t^s \in PI(f_i)$ exists such that $t^s \geq t \wedge s$. Note that $V(t) \cap V(\Delta^i[t]) =$

$\emptyset$, and hence we have $V(s) \subseteq V(t^s)\,(\subseteq V(s) \cup V(t))$, since otherwise there exists a clause c in $\Delta^i[t]$ such that $V(c) \cap V(t^s) = \emptyset$, a contradiction. Thus $V(t^s) \cap V(\Delta^i[t]) = V(s)$. For any $s' \in PI(g_{i,t})$ such that $s \neq s'$, let $t^s, t^{s'} \in PI(f_i)$ such that $t^s \geq t \wedge s$ and $t^{s'} \geq t \wedge s'$, respectively. By the above discussion, we have $t^s \neq t^{s'}$. This completes the proof. □

We now describe our algorithm Dualize for generating the set $PI(f)$. It is inspired by a similar graph algorithm of Johnson, Yannakakis, and Papadimitriou [27], and can be regarded as a generalization. Here, we say that term s is *smaller* than term t if

$$\sum_{x_j \in V(s)} 2^{n-j} < \sum_{x_j \in V(t)} 2^{n-j};$$

i.e., as vector, s is lexicographically smaller than t.

Algorithm Dualize

Input: The prime CNF φ of a monotone function f.
Output: The prime DNF ψ of f, i.e. all prime implicants of function f.

Step 1:
Compute the smallest prime implicant t_{min} of f and set $Q := \{\, t_{min} \,\}$;

Step 2:

```
while Q ≠ ∅ do
begin
  Remove the smallest t from Q and output t;
  for each i with x_i ∈ V(t) and Δ^i[t] ≠ 1 do
  begin
    Compute the prime DNF ρ_(t,i) of the
    function represented by Δ^i[t];
    for each term t' in ρ_(t,i) do
    begin
      if t_{i-1} ∧ t' is a prime implicant of f_i then
      begin
        Compute the smallest prime implicant
        t* of f such that t*_i = t_{i-1} ∧ t';
        Q := Q ∪ {t*}
      end{if}
    end{for}
  end{for}
end{while}
```

Theorem 3.3. *Algorithm* Dualize *correctly outputs all $t \in PI(f)$ in increasing order.*

Proof. (Sketch) First note that the term t^* inserted in Q when t is output is larger than t. Indeed, t' ($\neq 1$) and t_{i-1} are disjoint and $V(t') \subseteq \{x_1, \ldots, x_{i-1}\}$. Hence, every term in Q is larger than all terms already output, and the output sequence is increasing. We show by induction that, if t is the smallest prime implicant of f that was not output yet, then t is already in Q. This clearly proves the result.

Clearly, the above statement is true if $t = t_{min}$. Assume now that $t \neq t_{min}$ is the smallest among the prime implicants not output yet. Let i be the largest index such that t_i is not a prime implicant of f_i. This i is well-defined, since otherwise $t = t_{min}$ must hold, a contradiction. Now we have (1) $i < n$ and (2) $i+1 \notin V(t)$, where (1) holds because $t_n\,(=t)$ is a prime implicant of $f_n\,(=f)$ and (2) follows from the maximality of i. Let $s \in PI(f_i)$ such that $V(s) \subseteq V(t_i)$, and let $K = V(t_i) - V(s)$. Then $K \neq \emptyset$ holds, and since $x_{i+1} \notin V(t)$, the term $t' = \bigwedge_{x_j \in K} x_j$ is a prime implicant of $\Delta^{i+1}[s]$. There exists $s' \in PI(f)$ such that $s'_i = s$ and $x_{i+1} \in V(s')$, since $s \wedge x_{i+1} \in PI(f_{i+1})$. Note that $\Delta^{i+1}[s] \neq 0$. Moreover, since s' is smaller than t, by induction s' has already been output. Therefore, $t' = \bigwedge_{x_j \in K} x_j$ has been considered in the inner for-loop of the algorithm. Since $s'_i \wedge t'\,(= t_i = t_{i+1})$ is a prime implicant of f_{i+1}, the algorithm has added the smallest prime implicant t^* of f such that $t^*_{i+1} = t_{i+1}$. We finally claim that $t^* = t$. Otherwise, let k be the first index in which t^* and t differ. Then $k > i+1$, $x_k \in V(t)$ and $x_k \notin V(t^*)$. However, this implies $t_k \notin PI(f_k)$, contradicting the maximality of i. □

Let us consider the time complexity of algorithm Dualize. We store Q as a binary tree, where each leaf represents a term t and the left (resp., right) son of a node at depth $j-1 \geq 0$, where the root has depth 0, encodes $x_j \in V(t)$ (resp., $x_j \notin V(t)$). In Step 1, we can compute t_{min} in $O(\|\varphi\|)$ time and initialize Q in $O(n)$ time. As for Step 2, let $T_{(t,i)}$ be the time required to compute the prime DNF $\rho_{(t,i)}$ from $\Delta^i[t]$. By analyzing its substeps, we can see that each iteration of Step 2 requires $\sum_{x_i \in V(t)} (T_{(t,i)} + |\rho_{(t,i)}| O(\|\varphi\|))$ time; note that t^* is the smallest prime implicant of the function obtained from f by fixing $x_j = 1$ if $x_j \in V(t_i \wedge t')$ and 0 if $x_j \notin V(t_i \wedge t')$ for $j \leq i$. Thus, we have

Theorem 3.4. *The output delay of Algorithm* Dualize *is bounded by*

$$\max_{t \in PI(f)} \Big(\sum_{x_i \in V(t)} (T_{(t,i)} + |\rho_{(t,i)}| O(\|\varphi\|)) \Big) \tag{4}$$

time, and Dualize *needs in total time*

$$\sum_{t \in PI(f)} \sum_{x_i \in V(t)} (T_{(t,i)} + |\rho_{(t,i)}| O(\|\varphi\|)). \tag{5}$$

If the $T_{(t,i)}$ are bounded by a polynomial in the input length, then Dualize becomes a polynomial delay algorithm, since $|\rho_{(t,i)}| \leq T_{(t,i)}$ holds for all $t \in PI(f)$ and $x_i \in V(t)$. On the other hand, if they are bounded by a polynomial in the combined input and output length, then Dualize is a polynomial total time algorithm, where $|\rho_{(t,i)}| \leq |\psi|$ holds from Lemma 3.2. Using results from [2], we can construct from Dualize an incremental polynomial time algorithm for Dualization, which however might not output $PI(f)$ in increasing order. Summarizing, we have the following corollary.

Corollary 3.5. *Let $T = \max\{\, T_{(t,i)} \mid t \in PI(f), x_i \in V(t) \,\}$. Then, if T is bounded by*

(i) a polynomial in n and $\|\varphi\|$, then algorithm Dualize *is an $O(n\|\varphi\|T)$ polynomial delay algorithm;*

(ii) *a polynomial in n, $\|\varphi\|$, and $\|\psi\|$, then algorithm* DUALIZE *is an $O(n|\psi|(T+|\psi|\|\varphi\|))$ polynomial total time algorithm; moreover,* DUALIZATION *is solvable in incremental polynomial time.*

In the next section, we identify sufficient conditions for the boundedness of T and fruitfully apply them to solve open problems and improve previous results.

4. POLYNOMIAL CLASSES

4.1 Degenerate CNFs

We first consider the case of small $\Delta^i[t]$. Generalizing a notion for graphs (i.e., monotone 2-CNFs) [44], we call a monotone CNF φ *k-degenerate*, if there exists a variable ordering $x_1, \ldots, x_n$ in which $|\Delta^i| \le k$ for all $i = 1, 2, \ldots, n$. We call a variable ordering $x_1, \ldots, x_n$ *smallest last* as in [44], if x_i is chosen in the order $i = n, n-1, \ldots, 1$ such that $|\Delta^i|$ is smallest for all variables that were not chosen. Clearly, a smallest last ordering gives the least k such that φ is k-degenerate. Therefore, we can check for every integer $k \ge 1$ whether φ is k-degenerate in $O(\|\varphi\|)$ time. If this holds, then we have $|\rho_{(t,i)}| \le n^k$ and $T_{(t,i)} = O(kn^{k+1})$ for every $t \in PI(f)$ and $i \in V(t)$ (for $T_{(t,i)}$, apply the distributive law to $\Delta^i[t]$ and remove terms t where some $x_j \in V(t)$ has no $c \in \Delta^i[t]$ such that $V(t) \cap V(c) = \{x_j\}$). Thus Theorem 3.4 implies the following.

THEOREM 4.1. *For k-degenerate CNFs φ,* DUALIZATION *is solvable with $O(\|\varphi\|n^{k+1})$ polynomial delay if $k \ge 1$ is constant.*

Applying the result of [33] that any monotone CNF which has $O(\log n)$ many clauses is dualizable in incremental polynomial time, we obtain a polynomiality result also for non-constant degeneracy:

THEOREM 4.2. *For $O(\log \|\varphi\|)$-degenerate CNFs φ, problem* DUALIZATION *is polynomial total time.*

In the following, we discuss several natural subclasses of degenerate CNFs.

4.1.1 Read-bounded CNFs

A monotone CNF φ is called *read-k*, if each variable appears in φ at most k times. Clearly, read-k CNFs are k-degenerate, and in fact φ is read-k iff it is k-degenerate under every variable ordering. By applying Theorems 4.1 and 4.2, we obtain the following result.

COROLLARY 4.3. *For read-k CNFs φ, problem* DUALIZATION *is solvable*

(i) with $O(\|\varphi\|n^{k+1})$ polynomial delay, if k is constant;

(ii) in polynomial total time, if $k = O(\log(\|\varphi\|))$.

Note that Corollary 4.3 (i) trivially implies that DUALIZATION is solvable in $O(|\psi|n^{k+2})$ time for constant k, since $\|\varphi\| \le kn$. This improves upon the previous best known algorithm [10], which is only $O(|\psi|n^{k+3})$ time, not polynomial delay, and outputs $PI(f)$ in no specific order. Corollary 4.3 (ii) is a non-trivial generalization of the result in [10], which was posed as an open problem [9].

4.1.2 Acyclic CNFs

Like in graphs, acyclicity is appealing in hypergraphs resp. monotone CNFs from a theoretical as well as a practical point of view. However, there are many notions of acyclicity for hypergraphs (cf. [14]), since different generalizations from graphs are possible. We refer to α-, β-, γ-, and *Berge*-acyclicity as stated in [14], for which the following proper inclusion hierarchy is known:

$$\text{Berge-acyclic} \subseteq \gamma\text{-acyclic} \subseteq \beta\text{-acyclic} \subseteq \alpha\text{-acyclic}.$$

The notion of α-acyclicity came up in relational database theory. A monotone CNF φ is *α-acyclic* iff $\varphi = 1$ or reducible by the GYO-reduction [21, 45], i.e., repeated application of one of the two rules:

(1) If variable x_i occurs in only one clause c, remove x_i from clause c.

(2) If distinct clauses c and c' satisfy $V(c) \subseteq V(c')$, remove clause c from φ.

to 0 (i.e., the empty clause). Note that α-acyclicity of a monotone CNF φ can be checked, and a suitable GYO-reduction output, in $O(\|\varphi\|)$ time [42]. A monotone CNF φ is *β-acyclic* iff every CNF consisting of clauses in φ is α-acyclic. As shown in [13], the prime implicants of a monotone f represented by a β-acyclic CNF φ can be enumerated (and thus DUALIZATION solved) in $p(\|\varphi\|)|\psi|$ time, where p is a polynomial in $\|\varphi\|$. However, the time complexity of DUALIZATION for the more general α-acyclic prime CNFs was left as an open problem. We now show that it is solvable with polynomial delay.

Let $\varphi \ne 1$ be a prime CNF. Let $a = a_1, a_2, \ldots, a_q$ be a GYO-reduction for φ, where $a_\ell = x_i$ if the ℓ-th operation removes x_i from c, and $a_\ell = c$ if it removes c from φ. Consider the unique variable ordering $b_1, b_2, \ldots, b_n$ such b_i occurs after b_j in a, for all $i < j$.

EXAMPLE 4.1. *Let $\varphi = c_1c_2c_3c_4$, where $c_1 = (x_1 \vee x_2 \vee x_3)$, $c_2 = (x_1 \vee x_3 \vee x_5)$, $c_3 = (x_1 \vee x_5 \vee x_6)$ and $c_4 = (x_3 \vee x_4 \vee x_5)$. Then φ is α-acyclic, since it has the GYO-reduction $a_1 = x_2,\ a_2 = c_1,\ a_3 = x_4,\ a_4 = x_6,\ a_5 = c_4,\ a_6 = c_3,\ a_7 = x_1,\ a_8 = x_3,\ a_9 = x_5$. From this sequence, we obtain the variable ordering $b_1 = x_5,\ b_2 = x_3,\ b_3 = x_1,\ b_4 = x_6,\ b_5 = x_4,\ b_6 = x_2$. As easily checked, this ordering shows that φ is 1-degenerate. Under this ordering, we have $\Delta^1 = \Delta^2 = 1$, $\Delta^3 = (x_1 \vee x_3 \vee x_5)$, $\Delta^4 = (x_1 \vee x_5 \vee x_6)$, $\Delta^5 = (x_3 \vee x_4 \vee x_5)$, and $\Delta^6 = (x_1 \vee x_2 \vee x_3)$.*

That φ is 1-degenerate in this example is not accidental.

LEMMA 4.4. *Every α-acyclic prime CNF is 1-degenerate.*

Note that the converse is not true. Lemma 4.4 and Theorem 4.1 imply the following result.

COROLLARY 4.5. *For α-acyclic CNFs φ, problem* DUALIZATION *is solvable with $O(\|\varphi\|n^2)$ delay.*

Observe that for a prime α-acyclic φ, we have $|\varphi| \leq n$. Thus, if we slightly modify algorithm DUALIZE to check $\Delta^i = 1$ in advance (which can be done in linear time in a preprocessing phase) such that such Δ^i need not be considered in step 2, then the resulting algorithm has $O(n|\varphi|\|\varphi\|)$ delay. Observe that the algorithm in [13] solves, minorly adapted for enumerative output, DUALIZATION for β-acyclic CNFs with $O(n|\varphi|\|\varphi\|)$ delay. Thus, the above modification of DUALIZE is of the same order.

4.1.3 CNFs with bounded treewidth

A *tree decomposition (of type I)* of a monotone CNF φ is a tree $T=(W,E)$ where each node $w \in W$ is labeled with a set $X(w) \subseteq V(\varphi)$ under the following conditions:

1. $\bigcup_{w\in W} X(w) = V(\varphi)$;
2. for every clause c in φ, there exists some $w \in W$ such that $V(c) \subseteq X(w)$; and
3. for any variable $x_i \in V$, the nodes $\{w \in W \mid x_i \in X(w)\}$ induce a (connected) subtree of T.

The *width* of T is $\max_{w\in W} |X(w)| - 1$, and the *treewidth* of φ, denoted by $Tw_1(\varphi)$, is the minimum width over all its tree decompositions.

Note that the usual definition of treewidth for a graph [41] results in the case where φ is a 2-CNF. Similarly to acyclicity, there are several notions of treewidth for hypergraphs resp. monotone CNFs. For example, tree decomposition of type II of CNF $\varphi = \bigwedge_{c\in C} c$ is defined as type-I tree decomposition of its incident 2-CNF (i.e., graph) $G(\varphi)$ [11, 20]. That is, for each clause $c \in \varphi$, we introduce a new variable y_c and construct $G(\varphi) = \bigwedge_{x_i\in c\in\varphi}(x_i \vee y_c)$. Let $Tw_2(\varphi)$ denote the type-II treewidth of φ.

PROPOSITION 4.6. *For every monotone CNF φ, it holds that $Tw_2(\varphi) \leq Tw_1(\varphi) + 2^{Tw_1(\varphi)+1}$.*

PROOF. Let $T = (W,E)$, $X : W \to 2^V$ be any tree decomposition of φ having width $Tw_1(\varphi)$. Introduce for all $c \in \varphi$ new variables y_c, and add y_c to every $X(w)$ such that $V(c) \subseteq X(w)$. Clearly, the result is a type-I tree decomposition of $G(\varphi)$, and thus a type-II tree decomposition of φ. Since at most $2^{|X(w)|}$ many y_c are added to $X(w)$ and $|X(w)|-1 \leq Tw_1(\varphi)$ for every $w \in W$, the result follows. $\Box$

This means that if $Tw_1(\varphi)$ is bounded by some constant, then so is $Tw_2(\varphi)$. Moreover, $Tw_1(\varphi) = k$ implies that φ is a k-CNF; we discuss k-CNFs in Section 4.2 and only consider $Tw_2(\varphi)$ here. We note that, as shown in the full paper, there is a family of prime CNFs φ which have $Tw_2(\varphi)$ bounded by constant k but are not k-CNF for any $k < n$ (resp., not read-k for any $k < n-1$), and a family of prime CNFs which are k-CNFs for constant k (resp., α-acyclic) but $Tw_2(\varphi)$ is not bounded by any constant.

As we show now, bounded-treewidth implies bounded degeneracy.

LEMMA 4.7. *Let φ be any monotone CNF with $Tw_2(\varphi) = k$. Then φ is 2^k-degenerate.*

PROOF. (Sketch) Let $T = (W,E)$ with $X : W \to 2^V$ show $Tw_2(\varphi) = k$. From this, we reversely construct a variable ordering $a = a_1, \ldots, a_n$ on $V = V(\varphi)$ such that $|\Delta^i| \leq 2^k$ for all i.

Set $i := n$. Choose any leaf w^* of T, and let $p(w^*)$ be a node in W adjacent to w^*. If $X(w^*) \setminus X(p(w^*)) \subseteq \{y_c \mid c \in \varphi\}$, then remove w^* from T. On the other hand, if $(X(w^*) \setminus X(p(w^*))) \cap V = \{x_{j_1}, \ldots, x_{j_\ell}\}$ where $\ell \geq 1$ (in this case, only $X(w^*)$ contains $x_{j_1}, \ldots, x_{j_\ell}$), then define $a_{i+1-h} = x_{j_h}$ for $h = 1, \ldots, \ell$ and update $i := n - \ell$, $X(w^*) := X(w^*) \setminus \{x_{j_1}, \ldots, x_{j_\ell}\}$, and $X(w) := X(w) \setminus \{y_c \mid c \in \varphi, V(c) \cap \{x_{j_1}, \ldots, x_{j_\ell}\} \neq \emptyset\}$ for every $w \in W$. We complete a by repeating this process, and claim it shows that $|\Delta^i| \leq 2^k$ for all i. Let w^* be chosen during this process, and assume that $a_i \in X(w^*) \setminus X(p(w^*))$. Then, for each clause $c \in \Delta^i$ we must have either $y_c \in X(w^*)$ or $V(c) \subseteq X(w^*)$. Let $q = |X(w^*) \setminus V|$. Since $|X(w^*) \setminus \{a_i\}| \leq k$, we have $|\Delta^i| \leq q + 2^{k-q} \leq 2^k$. $\Box$

COROLLARY 4.8. *For CNFs φ with $Tw_2(\varphi) \leq k$,* DUALIZATION *is solvable*

(i) with $O(\|\varphi\|n^{2^k+1})$ polynomial delay, if k is constant;
(ii) in polynomial total time, if $k = O(\log\log\|\varphi\|)$.

4.2 Recursive application of algorithm DUALIZE

Algorithm DUALIZE computes in step 2 the prime DNF $\rho_{(t,i)}$ of the function represented by $\Delta^i[t]$. Since $\Delta[t]$ is the prime CNF of some monotone function, we can recursively apply DUALIZE to $\Delta^i[t]$ for computing $\rho_{(t,i)}$. Let us call this variant R-DUALIZE. Then we have the following result.

THEOREM 4.9. *If its recursion depth is d,* R-DUALIZE *solves* DUALIZATION *in $O(n^{d-1}|\psi|^{d-1}\|\varphi\|)$ time.*

PROOF. If $d = 1$, then $\Delta^i[t_{min}] = 1$ holds for t_{min} and every $i \geq 1$. This means that $PI(f) = \{t_{min}\}$ and φ is a 1-CNF (i.e., each clause in φ contains exactly one variable). Thus in this case, R-DUALIZE needs $O(n)$ time. Recall that algorithm DUALIZE needs, by (5), time $\sum_{t\in PI(f)} \sum_{x_i\in V(t)} (T_{(t,i)} + |\rho_{(t,i)}| O(\|\varphi\|))$. If $d = 2$, then $T_{(t,i)} = O(n)$ and $|\rho_{(t,i)}| \leq 1$. Therefore, R-DUALIZE needs time $O(n|\psi|\|\varphi\|)$. For $d \geq 3$, Corollary 3.5.(ii) implies that algorithm R-DUALIZE needs time $O(n^{d-1}|\psi|^{d-1}\|\varphi\|)$. $\Box$

Recall that a CNF φ is called k-*CNF* if each clause in φ has at most k literals. Clearly, if we apply algorithm R-DUALIZE to a monotone k-CNF φ, the recursion depth of R-DUALIZE is at most k. Thus we obtain the following result; it re-establishes, with different means, the main positive result of [6, 13].

COROLLARY 4.10. *Algorithm* R-DUALIZE *solves* DUALIZATION *in time* $O(n^{k-1}|\psi|^{k-1}\|\varphi\|)$, *i.e., in polynomial total time for monotone k-CNFs* φ *where k is constant.*

5. LIMITED NONDETERMINISM

In the previous section, we have discussed polynomial cases of monotone dualization. In this section, we now turn to the issue of the precise complexity of this problem. For this purpose, we consider the decision problem DUAL instead of the search problem DUALIZATION. It appears that problem DUAL can be solved with limited nondeterminism, i.e., with poly-log many guessed bits by a polynomial-time nondeterministic Turing machine. This result might bring new insight towards settling the complexity of the problem.

We adopt Kintala and Fischer's terminology [28] and write $g(n)$-P for the class of sets accepted by a nondeterministic Turing machine in polynomial time making at most $g(n)$ nondeterministic steps on every input of length n. For every integer $k \geq 1$, define $\beta_k\mathrm{P} = \bigcup_c (c\log^k n)$-P. The βP *Hierarchy* consists of the classes

$$\mathrm{P} = \beta_1\mathrm{P} \subseteq \beta_2\mathrm{P} \subseteq \cdots \subseteq \bigcup_k \beta_k\mathrm{P} = \beta\mathrm{P}$$

and lies between P and NP. The β_kP classes appear to be rather robust; they are closed under polynomial time and logspace many-one reductions and have complete problems (cf. [19]). The complement class of β_kP is denoted by co-β_kP.

We start by recalling algorithm A of [15], reformulated for CNFs. In what follows, we view CNFs φ also as sets of clauses, and clauses as sets of literals.

Algorithm A. (reformulated for CNFs)

Input: Monotone CNFs φ, ψ representing monotone f, g s.t. $V(c) \cap V(c') \neq \emptyset$, for all $c \in \varphi$, $c' \in \psi$.

Output: `yes` if $f = g^d$, otherwise a vector w of form $w = (w_1, \ldots, w_m)$ such that $f(w) \neq g^d(w)$.

Step 1:

Delete all redundant (i.e., non-minimal) implicates from φ and ψ.

Step 2:

Check that $V(\phi) = V(\psi)$, $\max_{c \in \varphi} |c| \leq |\psi|$, $\max_{c' \in \psi} |c'| \leq |\varphi|$, and $\Sigma_{c \in \varphi} 2^{-|c|} + \Sigma_{c' \in \psi} 2^{-|c'|} \geq 1$.

If any of these conditions fails, $f \neq g^d$ and a witness w is found in polynomial time (cf. [15]).

Step 3:

If $|\varphi||\psi| \leq 1$, test duality in $O(1)$ time.

Step 4:

If $|\varphi||\psi| \geq 2$, find a variable x_i that occurs in φ or ψ (w.l.o.g. in φ) with frequency $\geq 1/\log(|\varphi| + |\psi|)$.

Let

$\varphi_0 = \{c - \{x_i\} \mid x_i \in c,\, c \in \varphi\}$,
$\varphi_1 = \{c \mid x_i \notin c,\, c \in \varphi\}$,
$\psi_0 = \{c' - \{x_i\} \mid x_i \in c',\, c' \in \psi\}$,
$\psi_1 = \{c' \mid x_i \notin c',\, c' \in \psi\}$.

Call algorithm A on the two pairs of forms:

(A.1) $(\varphi_1,\, \psi_0 \wedge \psi_1)$ and (A.2) $(\psi_1,\, \varphi_0 \wedge \varphi_1)$

If both calls return `yes`, then return `yes` (as $f = g^d$), otherwise we obtain w such that $f(w) \neq g^d(w)$ in polynomial time (cf. [15]).

Let φ^i, ψ^i be the original input for A. For any pair (φ, ψ) of CNFs, define its *volume* by $v = |\varphi||\psi|$, and let $\epsilon = 1/\log n$, where $n = |\varphi^i| + |\psi^i|$. As shown in [15], step 4 of algorithm A divides the current (sub)problem of volume $v = |\varphi||\psi|$ by self-reduction into subproblems (A.1) and (A.2) of respective volumes (assuming that x_i frequently occurs in φ):

$$|\varphi_1||\psi_0 \wedge \psi_1| \;\leq\; (1-\epsilon)v \qquad (6)$$

$$|\varphi_0 \wedge \varphi_1||\psi_1| \leq |\varphi|(|\psi| - 1) \;\leq\; v - 1 \qquad (7)$$

Let $T = T(\varphi, \psi)$ be the recursion tree generated by A on input (φ, ψ), i.e., the root is labeled with (φ, ψ). Any node a labeled with (φ, ψ) is a leaf, if A stops on input (φ, ψ) during steps 1-3; otherwise, a has a left child a_l and a right child a_r corresponding to (A.1) and (A.2), i.e., labeled $(\varphi_1, \psi_0 \wedge \psi_1)$ and $(\psi_1, \varphi_0 \wedge \varphi_1)$ respectively. That is, a_l is the "high frequency move" by the splitting variable.

We observe that every node a in T is determined by a *unique path* from the root to a in T and thus by a unique sequence $seq(a)$ of right or left moves starting from the root of T and ending at a. The following key lemma bounds the number of moves of each type for certain inputs.

LEMMA 5.1. *Suppose* $|\varphi^i| + |\psi^i| \leq |\varphi^i||\psi^i|$. *Then for any node a in T, seq(a) contains* $\leq v$ *right and* $\leq \log^2 v$ *left moves, where* $v = |\varphi^i||\psi^i|$.

PROOF. By (6) and (7), each move decreases the volume v of a node label. Thus, the length of $seq(a)$, and in particular the number of right moves, is bounded by v. To obtain the better bound for the left moves, we will use the following well-known inequality:

$$(1 - 1/m)^m \;\leq\; 1/e, \qquad \text{for } m \geq 1. \qquad (8)$$

In fact, the sequence $(1 - 1/x_i)^{x_i}$, for any $1 < x_1 < x_2 < \ldots$ monotonically converges to $1/e$ from below. By inequality (6), the volume v_a of the label of any node a such that $seq(a)$ contains $\log^2 v$ left moves is bounded as follows:

$$v_a \leq v \cdot (1-\epsilon)^{\log^2 v} \leq v \cdot (1 - 1/\log n)^{\log^2 v}.$$

Because $n = |\varphi^i| + |\psi^i| \leq |\varphi^i| \cdot |\psi^i| = v$, and because of (8) it follows that:

$$\begin{aligned} v_a &\leq v \cdot \left((1 - 1/\log v)^{\log v}\right)^{\log v} \\ &\leq v \cdot (1/e)^{\log v} \\ &= v/(e^{\log v}) \\ &< v/(2^{\log v}) \;=\; 1. \end{aligned}$$

Thus, a must be a leaf in T. Hence for every a in T, $seq(a)$ contains at most $\log^2 v$ left moves. $\square$

THEOREM 5.2. *Problem* DUAL *is in* co-β_3P.

PROOF. (Sketch) Instances such that either $c \cap c' = \emptyset$ for some $c \in \varphi^i$ and $c' \in \psi^i$, the sequence $seq(a)$ is empty, or $|\varphi^i| + |\psi^i| > |\varphi^i||\psi^i|$ are easily solved in deterministic polynomial time. In the remaining cases, if $f \neq g^d$, then there exists a leaf a in T labeled by a non-dual pair (φ', ψ'). If $seq(a)$ is known, we can compute, by simulating A on the branch described by $seq(a)$, the entire path from the root to a with all labels (φ^i, ψ^i),...,(φ', ψ') and check that (φ', ψ') is non-dual in steps 2 and 3 of A in polynomial time. We observe that, as noted in [15], the binary length of any standard encoding of the input φ^i, ψ^i is polynomially related to $|\varphi^i| + |\psi^i|$ if algorithm A reaches step 3. Thus, to prove the theorem, it is sufficient to show that $seq(a)$ is obtainable in polynomial time from $O(\log^3 v)$ suitably guessed bits, where $v = |\varphi^i||\psi^i|$. To see this, let us represent every $seq(a)$ as a sequence $seq^*(a) = [\ell_0, \ell_1, \ell_2 \ldots, \ell_k]$, where ℓ_0 is the number of leading right moves and ℓ_i is the number of consecutive right moves after the i-th left move in $seq(a)$, for $i = 1, \ldots, k$. E.g., if $seq(a) = [\mathtt{r}, \mathtt{r}, \mathtt{l}, \mathtt{r}, \mathtt{r}, \mathtt{r}, \mathtt{l}]$, then $seq^*(a) = [2, 3, 0]$. By Lemma 5.1, $seq^*(a)$ has length at most $\log^2 v + 1$. Thus, $seq^*(a)$ occupies only $O(\log^3 v)$ bits in binary; moreover, $seq(a)$ is trivially computed from $seq^*(a)$ in polynomial time. □

REMARK 5.1. *It also follows that if $f \neq g^d$, a witness w can be found in polynomial time within $O(\log^3 n)$ nondeterministic steps. In fact, the sequence $seq(a)$ to a "failing leaf" labeled (φ', ψ') describes a choice of values for all variables in $V(\varphi \wedge \psi) \setminus V(\varphi' \wedge \psi')$. By completing it with values for $V(\varphi' \wedge \psi')$ that show non-duality of (φ', ψ'), we obtain in polynomial time a vector w such that $f(w) \neq g^d(w)$.*

The aim of the above proof was to show with very simple means that duality can be polynomially checked with limited nondeterminism. With a more involved proof, applied to the algorithm B of [15] (which runs in $n^{4\chi(n)+O(1)}$ and thus $n^{o(\log n)}$ time), we can prove the following sharper result.

THEOREM 5.3. *Deciding if monotone CNFs φ and ψ are non-dual is feasible in polynomial time with $O(\chi(n) \log n)$ nondeterministic steps. Thus, problem* DUAL *is in* co-β_2P.

While our independently developed methods are different from those in [1], the previous result may also be obtained from Beigel and Fu's Theorem 11 in [1]. They show how to convert certain recursive algorithms that use disjunctive self-reductions and have runtime bounded by $f(n)$ into polynomial algorithms using $\log f(n)$ nondeterministic steps (cf. [1, Chapter 5]). However, this yields a somewhat more complicated nondeterministic algorithm. In the full paper, we also prove that algorithm B qualifies for this.

6. ACKNOWLEDGMENTS

This work was supported in part by the Austrian Science Fund (FWF) project Z29-INF, by TU Wien through a scientific collaboration grant, and by the Scientific Grant in Aid of the Ministry of Education, Science, Sports and Culture of Japan. We would like to thank the reviewers for their constructive comments on this paper.

7. REFERENCES

[1] R. Beigel and B. Fu. Molecular computing, bounded nondeterminism, and efficient recursion. *Algorithmica*, 25: 222–238, 1999.

[2] C. Bioch and T. Ibaraki. Complexity of identification and dualization of positive Boolean functions. *Information and Computation*, 123:50–63, 1995.

[3] E. Boros, K. Elbassioni, V. Gurvich, L. Khachiyan and K. Makino. On generating all minimal integer solutions for a monotone system of linear inequalities. In: *Proc. 28th International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 22–103, Springer LNCS 2076, 2001.

[4] E. Boros, V. Gurvich, L. Khachiyan and K. Makino. Dual-bounded generating problems: partial and multiple transversals of a Hypergraph. *SIAM J. Comput.*, 30:2036–2050, 2001.

[5] E. Boros, V. Gurvich, L. Khachiyan and K. Makino. On the complexity of generating maximal frequent and minimal infrequent sets. to appear in *Proceedings of 19th International Symposium on Theoretical Aspects of Computer Science (STACS 2002).*

[6] E. Boros, V. Gurvich, and P. L. Hammer. Dual subimplicants of positive Boolean functions. *Optimization Methods and Software*, 10: 147–156, 1998.

[7] E. Boros, P. L. Hammer, T. Ibaraki and K. Kawakami. Polynomial time recognition of 2-monotonic positive Boolean functions given by an oracle. *SIAM J. Comput.*, 26:93-109, 1997.

[8] Y. Crama. Dualization of regular Boolean functions. *Discrete Applied Mathematics*, 16:79–85, 1987.

[9] C. Domingo. Private communication.

[10] C. Domingo, N. Mishra and L. Pitt. Efficient read-restricted monotone CNF/DNF dualization by learning with membership queries. *Machine Learning*, 37:89–110, 1999.

[11] C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. In *Proceedings of International Conference on Database Theory (ICDT'97)*, Delphi, Greece, Springer LNCS 1186, pp. 56–70, 1997.

[12] T. Eiter. Exact transversal hypergraphs and application to Boolean μ-functions. *Journal of Symbolic Computation*, 17:215–225, 1994.

[13] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304, 1995.

[14] R. Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *JACM*, 30:514–550, 1983.

[15] M. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21:618–628, 1996.

[16] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *JACM*, 32(4):841–860, 1985.

[17] G. Gogic, C. Papadimitriou, and M. Sideri. Incremental recompilation of knowledge. *Journal of Artificial Intelligence Research*, 8:23–37, 1998.

[18] D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. *Proc. 16th ACM Symp. on Principles of Database Systems (PODS)*, pp. 209–216, 1997.

[19] J. Goldsmith, M. Levy, and M. Mundhenk. Limited nondeterminism. *SIGACT News*, 27(2):20-29, 1996.

[20] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *Proc. 18th ACM Symp. on Principles of Database Systems (PODS)*, pp. 21-32, 1999. Full paper to appear in *J. Computer and System Sciences*.

[21] M. Graham. On the universal relation. Technical Report, University of Toronto, Canada, September 1979.

[22] V. Gurvich. Nash-solvability of games in pure strategies. *USSR Comput. Math and Math. Phys.*, 15(2):357–371, 1975.

[23] T. Ibaraki and T. Kameda. A theory of coteries: Mutual exclusion in distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 4(7):779–794, 1993.

[24] D. Kavvadias, C. H. Papadimitriou, and M. Sideri, On Horn envelopes and hypergraph transversals. *Proc. 4th International Symposium on Algorithms and Computation (ISAAC)*, pp. 399–405, Springer LNCS 762, 1993.

[25] R. Khardon. Translating between Horn representations and their characteristic models. *Journal of AI Research*, 3:349-372, 1995.

[26] D. S. Johnson. Open and closed problems in NP-completeness. Lecture given at the International School of Mathematics "G. Stampacchia": Summer School "NP-Completeness: The First 20 Years", Erice, Italy, June 20-27, 1991.

[27] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27:119–123, 1988.

[28] C.M.R. Kintala and P. Fischer. Refining nondeterminism in relativized polynomial-time bounded computations. *SIAM J. Comput.*, 9:46–53, 1980.

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

[30] L. Lovász. Combinatorial optimization: Some problems and trends. DIMACS Technical Report 92-53, RUTCOR, Rutgers University, 1992.

[31] K. Makino and T. Ibaraki. The maximum latency and identification of positive Boolean functions. *SIAM J. Comput.*, 26:1363–1383, 1997.

[32] K. Makino and T. Ibaraki, A fast and simple algorithm for identifying 2-monotonic positive Boolean functions. *Journal of Algorithms*, 26:291–305, 1998.

[33] K. Makino. Efficient dualization of $O(\log n)$-term monotone disjunctive normal forms. Technical Report 00-07, Discrete Mathematics and Systems Science, Osaka University, 2000. To appear in *Discrete Applied Mathematics*.

[34] H. Mannila and K.-J. Räihä. Design by example: An application of Armstrong relations. *Journal of Computer and System Sciences*, 22(2):126–141, 1986.

[35] N. Mishra and L. Pitt. Generating all maximal independent sets of bounded-degree hypergraphs. Proc. Tenth Annual Conference on Computational Learning Theory (COLT), pp. 211–217, 1997.

[36] Ch. H. Papadimitriou. NP-completeness: A retrospective, In: *Proc. 24th International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 2–6, Springer LNCS 1256, 1997.

[37] U. N. Peled and B. Simeone. An $O(nm)$-time algorithm for computing the dual of a regular Boolean function. *Discrete Applied Mathematics*, 49:309–323, 1994.

[38] K. G. Ramamurthy. *Coherent Structures and Simple Games*. Kluwer Academic Publishers, 1990.

[39] R. C. Read. Every one a winner, or how to avoid isomorphism when cataloging combinatorial configurations. *Annals of Discrete Mathematics*, 2:107–120, 1978.

[40] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[41] N. Robertson and P. Seymour. Graph minors II: Algorithmic aspects of tree-width. *J. Algorithms*, 7:309–322, 1986.

[42] R. E. Tarjan and M. Yannakakis. Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13:566–579, 1984.

[43] V. D. Thi. Minimal keys and antikeys. *Acta Cybernetica*, 7(4):361–371, 1986.

[44] B. Toft. Colouring, stable sets and perfect graphs. *Handbook of Combinatorics*, Vol. 1 Chapter 4. Elsevier, 1995.

[45] C. T. Yu and M. Ozsoyoglu. An algorithm for tree-query membership of a distributed query. *Proceedings IEEE COMPSAC*, 1979, pp. 306–312.