Achieving New Upper Bounds for the Hypergraph Duality Problem through Logic

Georg Gottlob
Department of Computer Science
University of Oxford
Oxford OX13QD, UK
georg.gottlob@cs.ox.ac.uk

Enrico Malizia
D.I.M.E.S.
Università della Calabria
Rende (CS), 87036, Italy
emalizia@dimes.unical.it

Abstract

The hypergraph duality problem DUAL is defined as follows: given two simple hypergraphs $\mathcal{G}$ and $\mathcal{H}$, decide whether $\mathcal{H}$ consists precisely of all minimal transversals of $\mathcal{G}$ (in which case we say that $\mathcal{G}$ is the dual of $\mathcal{H}$). This problem is equivalent to decide whether two given non-redundant monotone DNFs are dual. It is known that $\overline{\text{DUAL}}$, the complementary problem to DUAL, is in GC($\log^2 n$, PTIME), where GC($f(n)$, $\mathcal{C}$) denotes the complexity class of all problems that after a nondeterministic guess of $O(f(n))$ bits can be decided (checked) within complexity class $\mathcal{C}$. It was conjectured that $\overline{\text{DUAL}}$ is in GC($\log^2 n$, LOGSPACE). In this paper we prove this conjecture and actually place the $\overline{\text{DUAL}}$ problem into the complexity class GC($\log^2 n$, TC^0) which is a subclass of GC($\log^2 n$, LOGSPACE). We here refer to the logtime-uniform version of TC^0, which corresponds to FO(COUNT), i.e., first order logic augmented by counting quantifiers. We achieve the latter bound in two steps. First, based on existing problem decomposition methods, we develop a new nondeterministic algorithm for $\overline{\text{DUAL}}$ that requires to guess $O(\log^2 n)$ bits. We then proceed by a logical analysis of this algorithm, allowing us to formulate its deterministic part in FO(COUNT).

Categories and Subject Descriptors F.2.2 [*Analysis of Algorithms and Problem Complexity*]: Nonnumerical Algorithms and Problems—Computations on discrete structures; F.4.1 [*Mathematical Logic and Formal Languages*]: Mathematical Logic; G.2.2 [*Discrete Mathematics*]: Graph Theory—Graph algorithms, Hypergraphs

General Terms Algorithms, Theory

Keywords hypergraphs, hypergraph transversals, hypergraph duality, DNF duality, complexity, first order logic, logical analysis of algorithms, TC^0, first order logic with counting quantifiers, algorithms, limited nondeterminism, graph theory

CSL-LICS 2014, July 14–18, 2014, Vienna, Austria.

ACM 978-1-4503-2886-9/14/07. . . $15.00.
http://dx.doi.org/10.1145/2603088.2603103

1. Introduction

The hypergraph duality problem DUAL is one of the most mysterious and challenging decision problems of Computer Science, as its complexity has been intensively investigated for almost 40 years without any indication that the problem is tractable, nor any evidence whatsoever, why it should be intractable. Apart from a few significant upper bounds, which we review below, and a large number of restrictions that make the problem tractable, progress on pinpointing the complexity of DUAL has been rather slow. So far, the problem has been placed in relatively low complexity classes within coNP. It is the aim of this paper to further narrow it down by using logical methods.

The hypergraph duality problem. A hypergraph $\mathcal{G}$ consists of a finite set V of vertices and a set $E \subseteq 2^V$ of (hyper)edges. $\mathcal{G}$ is *simple* (or *Sperner*) if none of its edges is contained in any other of its edges. A *transversal* or *hitting set* of a hypergraph $\mathcal{G} = (V, E)$ is a subset of V that meets every edge in E. A transversal of $\mathcal{G}$ is minimal if none of its proper subsets is a transversal. The set of minimal transversals of a hypergraph $\mathcal{G} = (V, E)$ is denoted by $tr(\mathcal{G})$. Note that $tr(\mathcal{G})$, which is referred to as *the dual of* $\mathcal{G}$ or also as the *transversal hypergraph* of $\mathcal{G}$, is itself a hypergraph on the vertex set V. The decision problem DUAL is now easily defined as follows: Given two simple hypergraphs $\mathcal{G}$ and $\mathcal{H}$, decide whether $\mathcal{H} = tr(\mathcal{G})$.

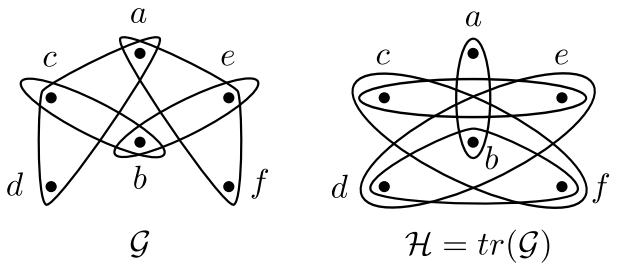

Figure 1. Hypergraph $\mathcal{G}$ and its transversal hypergraph $\mathcal{H}$.

An example of a hypergraph and its dual is given in Figure 1. It is well-known that the duality problem has a nice symmetry property: $\mathcal{H} = tr(\mathcal{G})$ iff $\mathcal{G} = tr(\mathcal{H})$. The DUAL problem is also tightly related to the problem of actually *computing* $tr(\mathcal{G})$ for an input hypergraph $\mathcal{G}$. In fact, it is known that the computation problem is feasible in total polynomial time, that is, in time polynomial in $|\mathcal{G}| + |tr(\mathcal{G})|$, if and only if DUAL is solvable in polynomial time [1]. These and several other properties of the duality problem are reviewed and discussed in [7, 10, 11, 23], where also many original references can be found.

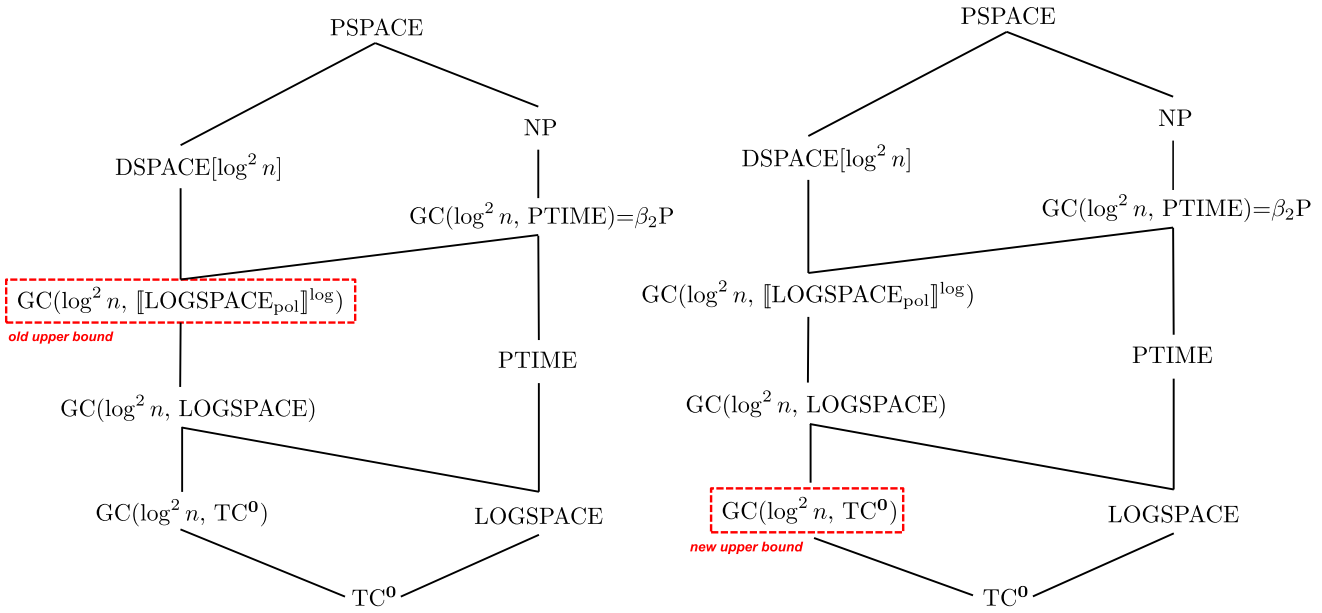

Figure 2. Complexity bound improvement obtained in this paper.

Applications of hypergraph duality. The DUAL problem and its computational variant have a tremendous number of applications. They range from data mining [3, 4, 22, 31], functional dependency inference [19, 29, 30], and machine learning, in particular, learning monotone Boolean CNFs and DNFs with membership queries [22, 32], to model-based diagnosis [21, 33], computing a Horn approximation to a non-Horn theory [16, 28], and computing minimal abductive explanations to observations [9]. Surveys of these and other applications as well as further references can be found in [7, 8, 23].

The simplest and foremost applications relevant to logic and hardware design are DNF duality testing and its computational version, DNF dualization. A pair of Boolean formulas $f(x_1, \ldots, x_n)$ and $g(x_1, \ldots, x_n)$ on propositional variables $x_1, \ldots, x_n$ are *dual* if

$$f(x_1, \ldots, x_n) \equiv \neg g(\neg x_1, \ldots, \neg x_n).$$

A monotone DNF is *irredundant* if the set of variables in none of its disjuncts is covered by the variable set of any other disjunct. The *duality testing problem* is the problem of testing whether two irredundant monotone DNFs f and g are dual. It is well-known and easy to see that monotone DNF duality and DUAL are actually the same problem. Two hypergraphs $\mathcal{G}$ and $\mathcal{H}$ are dual iff their associated DNFs $\mathcal{G}^*$ and $\mathcal{H}^*$ are dual, where the DNF $\mathcal{F}^*$ associated to a hypergraph $\mathcal{F} = (V, E)$ is $\bigvee_{e \in E} \bigwedge_{v \in e} v$, where, obviously, vertices $v \in V$ are interpreted as propositional variables. For example, the hypergraphs $\mathcal{G}$ and $\mathcal{H}$ of Figure 1 give rise to DNFs

$$\mathcal{G}^* = (a \wedge c \wedge d) \vee (a \wedge e \wedge f) \vee (c \wedge b) \vee (e \wedge b), \quad \text{and}$$

$$\mathcal{H}^* = (a \wedge b) \vee (c \wedge e) \vee (c \wedge b \wedge f) \vee (e \wedge b \wedge d) \vee (d \wedge b \wedge f),$$

which are indeed mutually dual. The duality problem for irredundant monotone DNFs corresponds, in turn, to DUAL, and the problem instances $(\mathcal{F}, \mathcal{G})$ and $(\mathcal{F}^*, G^*)$ can be inter-translated by extremely low-level reductions, in particular, LOGTIME reductions, and even projection reductions. In many publications, the DUAL problem is thus right away introduced as the problem of duality checking for irredundant monotone DNFs. An equivalent problem is the problem of checking whether a monotone CNF and a monotone DNF are logically equivalent.

Previous Complexity bounds. DUAL is easily seen to reside in coNP. In fact, in order to show that a DUAL instance is a "no"-instance, it suffices to show that either some edge of one of the two hypergraphs is not a minimal transversal of the other hypergraph (which is feasible in polynomial time), or to find (guess and check) a missing transversal to one of the input hypergraphs. The complement $\overline{\text{DUAL}}$ of DUAL is therefore in NP. In their landmark paper [13], Fredman and Khachiyan have shown that DUAL is in DTIME[$n^{o(\log n)}$], more precisely, that it is contained in DTIME[$n^{4\chi(n)+O(1)}$], where $\chi(n)$ is defined by $\chi(n)^{\chi(n)} = n$. Note that $\chi(n) \sim \log n / \log\log n = o(\log n)$.

Let GC($f(n)$, $\mathcal{C}$) denote the complexity class of all problems that after a nondeterministic guess of $O(f(n))$ bits can be decided (checked) in complexity class $\mathcal{C}$. Eiter, Gottlob, and Makino [10], and independently, Kavvadias and Stavropoulos [27] have shown that $\overline{\text{DUAL}}$ is in GC($\log^2 n$, PTIME); note that this class is also known as β_2P, see [17].

Recently, in [18], the nondeterministic bound for $\overline{\text{DUAL}}$ was further pushed down to GC($\log^2 n$, $[\![\text{LOGSPACE}_{\text{pol}}]\!]^{\log}$), see Figure 2a. A precise definition of $[\![\text{LOGSPACE}_{\text{pol}}]\!]^{\log}$ is given in [18]. We will not make use of this class in the technical part of the present paper. Informally, $[\![\text{LOGSPACE}_{\text{pol}}]\!]^{\log}$ contains those problems π for which there exists a logspace-transducer T, a polynomial p, and a function f in $O(\log n)$, such that each π-instance I of size $n = |I|$ can be reduced by the $f(n)$-fold composition $T^{f(n)}$ of T to a decision problem in LOGSPACE, where the size of all intermediate results $T^i(I)$, for $1 \leq i \leq f(n)$ is polynomially bounded by $p(n)$. For the relationship of GC($\log^2 n$, $[\![\text{LOGSPACE}_{\text{pol}}]\!]^{\log}$) to other classes, see Figure 2a. In particular, in [18], it was shown that GC($\log^2 n$, $[\![\text{LOGSPACE}_{\text{pol}}]\!]^{\log}$) is not only a subclass of GC($\log^2 n$, PTIME), but also of DSPACE[$\log^2 n$], i.e., of quadratic logspace. Therefore, DUAL is in DSPACE[$\log^2 n$], and it is thus most unlikely for DUAL to be PTIME-hard, which

answered a previously long standing question. Given that PTIME and DSPACE[$\log^2 n$] are believed to be incomparable, it is also rather unlikely that DUAL is closely related to another interesting logical problem of open complexity, namely, to validity-checking for the modal μ-calculus, or, equivalently, to the winner determination problem for parity games [24, 26], as these latter problems are PTIME-hard, but in NP $\cap$ coNP.

Main complexity problem tackled. In [18] it was asked whether the upper bound of GC($\log^2 n$, $[\![\text{LOGSPACE}_{\text{pol}}]\!]^{\log}$) could be pushed further downwards, and the following conjecture was made:

Conjecture 1 ([18]) $\overline{\text{DUAL}} \in \text{GC}(\log^2 n, \text{LOGSPACE})$.

It was unclear, however, how to prove this conjecture based on the algorithms and methods used in [18]. There, a problem decomposition strategy by Boros and Makino [2] was used, that decomposed an original DUAL instance into a conjunction of smaller instances according to a specific conjunctive self-reduction. Roughly, this strategy constructs a decomposition tree of logarithmic depth for DUAL, each of whose nodes represents a sub-instance of the original instance; more details on decomposition trees are given in Section 3. To prove that the original instance is a "no"-instance (and thus a "yes"-instance of $\overline{\text{DUAL}}$), it is sufficient to guess, in that tree, a path Π from the root to a single node v associated with a "no"-sub-instance that can be recognized as such in logarithmic space. Guessing the path to v can be easily done using $O(\log^2 n)$ nondeterministic bits, but it is totally unclear how to actually *compute* the sub-instance associated with node v in LOGSPACE. In fact, it seems that the only way to compute the sub-instance at node v is to compute —at least implicitly— all intermediate DUAL instances arising on the path from the root to the decomposition node v. This seems to require a logarithmic composition of LOGSPACE transducers, and thus a computation in the complexity class $[\![\text{LOGSPACE}_{\text{pol}}]\!]^{\log}$. It was therefore totally unclear how $[\![\text{LOGSPACE}_{\text{pol}}]\!]^{\log}$ could be replaced by its subclass LOGSPACE, and new methods were necessary to achieve this goal.

New results: Logic to the rescue. To attack the problem, we studied various alternative decomposition strategies for DUAL, among which the strategy of Gaur [14], which also influenced the method of Boros and Makino [2]. In the present paper, we build on Gaur's original strategy, as it appears to be the best starting point for our purposes. However, Gaur's method still does not directly lead to a guess-and-check algorithm whose checking procedure is in LOGSPACE, and thus new techniques needed to be developed.

In a first step, by building creatively on Gaur's deterministic decomposition strategy [14], we develop a new nondeterministic guess-and-check algorithm ND-NOTDUAL for $\overline{\text{DUAL}}$, that is specifically geared towards a computationally simple checking part. In particular, the checking part of ND-NOTDUAL avoids certain obstructive steps that would require more memory than just plain LOGSPACE, such as the successive minimization of hypergraphs in sub-instances of the decomposition (as used by Boros and Makino in [2]) and the performance of counting operations between subsequent decomposition steps so to determine sets of vertices to be included in a new transversal (as used by Gaur in [14]). Our new algorithm is thus influenced by Gaur's, but differs noticeably from it, as well as from the algorithm of Boros and Makino.

In a second step, we proceed with a careful logical analysis of the checking part of ND-NOTDUAL. We transform all sub-tasks of ND-NOTDUAL into logical formulas. However, it turns out that first order logic (FO) is not sufficient, as an essential step of the checking phase of ND-NOTDUAL is to check for specific hypergraph vertices v whether v is contained in at least half of the hyperedges of some hypergraph. To account for this, we need to resort to FO(COUNT), which augments FO with counting quantifiers. Note that we could have used in a similar way FOM, i.e., FO augmented by majority quantifiers, as FO(COUNT) and FOM have the same expressive power [25]. By putting all pieces together, we succeed to describe the entire checking phase by a single fixed FO(COUNT) formula that has to be evaluated over the input $\overline{\text{DUAL}}$ instance. Note that FO(COUNT) model-checking is complete for logtime-uniform $\text{TC}^{\mathbf{0}}$ [25, 34].

In summary, by putting the guessing and checking parts together, we achieve as main theorem a complexity result that is actually better than the one conjectured:

Theorem. $\overline{\text{DUAL}} \in \text{GC}(\log^2 n, \text{TC}^{\mathbf{0}})$.

By the well-known inclusion $\text{TC}^{\mathbf{0}} \subseteq \text{LOGSPACE}$, we immediately obtain a corollary that proves Conjecture 1:

Corollary. $\overline{\text{DUAL}} \in \text{GC}(\log^2 n, \text{LOGSPACE})$.

Significance of the new results and directions for future research. The progress achieved in this paper is summarized in Figure 2, whose left part (Figure 2a) shows the previous state of knowledge about the complexity, while the right part (Figure 2b) depicts the current state of knowledge we have achieved. We have significantly narrowed down the "search space" for the precise complexity of DUAL (or $\overline{\text{DUAL}}$). We believe that our new results are of value to anybody studying the complexity of this interesting problem. In particular, the connection to logic opens new avenues for such studies. First, our results show where to dig for tighter bounds. It may be rewarding to study subclasses of GC($\log^2 n$, $\text{TC}^{\mathbf{0}}$), and in particular, *logically defined subclasses* that replace $\text{TC}^{\mathbf{0}}$ by low-level prefix classes of FO(COUNT). Classes of this type can be found in [5] and [6]. Secondly, the membership of $\overline{\text{DUAL}}$ in GC($\log^2 n$, $\text{TC}^{\mathbf{0}}$) provides valuable information for those trying to prove hardness results for DUAL, i.e., to reduce some presumably intractable problem X to DUAL. Our results restrict the search space to be explored to hunt for such a problem X. Moreover, given that LOGSPACE is not known to be in GC($\log^2 n$, $\text{TC}^{\mathbf{0}}$), and given that it is not generally believed that GC($\log^2 n$, $\text{TC}^{\mathbf{0}}$) contains LOGSPACE-hard problems (under logtime reductions), our results suggest that LOGSPACE-hard problems are rather unlikely to reduce to DUAL, and that it may thus be advisable to look for a problem X that is not (known to be) LOGSPACE-hard, in order to find a lower bound for DUAL. Our new results are of theoretical nature. This does not rule out the possibility that they may be used for improving practical algorithms, but this has yet to be investigated. Finally, we believe that the methods presented in this paper are a compelling example of how logic and descriptive complexity theory can be used together with suitable problem decomposition methods to achieve new complexity results for a concrete decision problem.

Organization of the paper. After some preliminaries in Section 2, we discuss problem decomposition strategies and introduce the concept of a decomposition tree for $\overline{\text{DUAL}}$ in Section 3. Based on this, Section 4 presents the ND-NOTDUAL algorithm, proves it correct, and then analyzes this algorithm to derive our main complexity result.

2. Preliminaries

In what follows, we will often identify a hypergraph $\mathcal{G} = \langle V, E\rangle$ with its edge-set and vice-versa. By writing $G \in \mathcal{G}$ we mean $G \in E$, and by writing $\mathcal{G} = E$ we mean that $\mathcal{G} = \langle \bigcup_{G\in E} G, E\rangle$. $|\mathcal{G}|$ denotes the number of edges of $\mathcal{G}$.

Given a hypergraph $\mathcal{G}$ and a set of vertices T, a vertex v is *critical* in T (w.r.t. $\mathcal{G}$) if there exists an edge $G \in \mathcal{G}$ such that $G \cap T = \{v\}$. If $\mathcal{G}$ and $\mathcal{H}$ are two hypergraphs, a set of vertices T

is a *new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$*[1] if T is a transversal of $\mathcal{G}$ and, for all $H \in \mathcal{H}$, $H \not\subseteq T$. Observe that a new transversal does not need to be a minimal transversal.

Given a hypergraph $\mathcal{G}$ and a set S of vertices, as in [2, 12], we define hypergraphs $\mathcal{G}_S = \{G \in \mathcal{G} \mid G \subseteq S\}$, and $\mathcal{G}^S = \min(\{G \cap S \mid G \in \mathcal{G}\})$, where $\min(\mathcal{H})$, for any hypergraph $\mathcal{H}$, denotes the set of inclusion minimal edges of $\mathcal{H}$. Observe that $\mathcal{G}^S$ is always a simple hypergraph, and that if $\mathcal{G}$ is simple, then so is $\mathcal{G}_S$.

The following properties are known (see, e.g., [2, 12–14]). Consider two hypergraphs $\mathcal{G}$ and $\mathcal{H}$. A set of vertices T is a minimal transversal of $\mathcal{G}$ if and only if every vertex $v \in T$ is critical. A set of vertices T is a new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$ if and only if $V \setminus T$ is a new transversal of $\mathcal{H}$ w.r.t. $\mathcal{G}$. If the vertex set is V and $T \subseteq V$, let $\overline{T}$ denote $V \setminus T$, i.e., the *complement* of T in V.

We say that hypergraphs $\mathcal{G}$ and $\mathcal{H}$ satisfy the *intersection property* if $\mathcal{H} \subseteq tr(\mathcal{G})$ and $\mathcal{G} \subseteq tr(\mathcal{H})$. It can be shown [2, 14] that given an instance $\langle \mathcal{G}, \mathcal{H} \rangle$ of DUAL whose hypergraphs satisfy the intersection property, then the existence of a new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$ exactly characterizes $\langle \mathcal{G}, \mathcal{H} \rangle$ as a "no"-instance of DUAL. If $\mathcal{G}$ and $\mathcal{H}$ are two hypergraphs satisfying the intersection property then $|V| \leq |\mathcal{G}| \cdot |\mathcal{H}|$ (see [2]).

Let $\langle \mathcal{G}, \mathcal{H} \rangle$ be an instance of DUAL. We denote by m the total number $|\mathcal{G}| + |\mathcal{H}|$ of edges of $\mathcal{G}$ and $\mathcal{H}$.

3. Decomposing the Dual problem

3.1 Decomposition principles

An approach to recognize "no"-instances $\langle \mathcal{G}, \mathcal{H} \rangle$ of DUAL is to find a new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$, i.e., a transversal of $\mathcal{G}$ that is an independent set of $\mathcal{H}$. Many algorithms in the literature follow this approach (see, e.g., [2, 10, 12–14, 27]). These algorithms try to build such a new transversal by successively including vertices in a candidate new transversal T or excluding vertices from T. To give an example, the classical algorithm "A" of Fredman and Khachiyan [13] tries to include a vertex v in a candidate new transversal, and if this does not lead to the construction of a new transversal then v is excluded. Moreover if the exclusion of v does not lead to the construction of a new transversal then there does not exist any (coherent with the choices having been made before considering the vertex v).

We speak about "included" and "excluded" vertices because most of the algorithms proposed in the literature implicitly or explicitly keep track of two sets: the set of the vertices considered included in the attempted new transversal T, and the set of the vertices considered excluded from T.

If $\mathcal{G}$ and $\mathcal{H}$ are two hypergraphs, an *assignment* $\sigma = \langle In, Ex \rangle$ is a pair of subsets of V such that $In \cap Ex = \varnothing$. Intuitively, the set In contains the vertices considered included in an attempted new transversal $T \supseteq In$ of $\mathcal{G}$ w.r.t. $\mathcal{H}$, while the set Ex contains the vertices considered excluded from T (i.e., $T \cap Ex = \varnothing$). A vertex $v \in V$ is *free* in an assignment $\sigma = \langle In, Ex \rangle$ if $v \notin In$ and $v \notin Ex$. We assume, by definition, that the empty assignment $\sigma_\varepsilon = \langle \varnothing, \varnothing \rangle$ is a valid assignment. Given assignments $\sigma_1 = \langle In_1, Ex_1 \rangle$ and $\sigma_2 = \langle In_2, Ex_2 \rangle$, if $In_1 \cap Ex_2 = \varnothing$ and $Ex_1 \cap In_2 = \varnothing$, we denote by $\sigma_1 + \sigma_2 = \langle In_1 \cup In_2, Ex_1 \cup Ex_2 \rangle$ the extension of σ_1 with σ_2. An assignment $\sigma = \langle In, Ex \rangle$ is *coherent* with a set of vertices S, and vice-versa, whenever $In \subseteq S$ and $Ex \subseteq \overline{S}$ (or, equivalently, $Ex \cap S = \varnothing$), and we denote it by $\sigma \sqsubseteq S$. It is easy to see that if $\sigma = \langle In, Ex \rangle$ is coherent with a new transversal T of $\mathcal{G}$ w.r.t. $\mathcal{H}$, then the reversed assignment $\langle Ex, In \rangle$ is coherent with the new transversal $\overline{T}$ of $\mathcal{H}$ w.r.t. $\mathcal{G}$. Intuitively, given an assignment $\sigma = \langle In, Ex \rangle$ the set In is (a subset of) an attempted new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$, and, symmetrically, the set Ex is (a subset of) an attempted new transversal of $\mathcal{H}$ w.r.t. $\mathcal{G}$.

Given an assignment $\sigma = \langle In, Ex \rangle$, we define:

- $Sep(\sigma) = \{G \in \mathcal{G} \mid G \cap In = \varnothing\}$, the set of all edges of $\mathcal{G}$ *not met* by (or, equivalently, *separated* from) σ;
- $Com(\sigma) = \{H \in \mathcal{H} \mid H \cap Ex = \varnothing\}$, the set of all edges of $\mathcal{H}$ *compatible* with σ;
- $Mis(\sigma) = \{G \in \mathcal{G} \mid G \subseteq Ex\}$, the set of all edges of $\mathcal{G}$ *entirely missed* by σ; and
- $Cov(\sigma) = \{H \in \mathcal{H} \mid H \subseteq In\}$, the set of all edges of $\mathcal{H}$ *entirely covered* by σ.

An assignment $\sigma = \langle In, Ex \rangle$ is a *witness* of the existence of a new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$ if In is a new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$, or Ex is a new transversal of $\mathcal{H}$ w.r.t. $\mathcal{G}$. A witness is easily proven to be characterized as follows.

Lemma 1. *Let $\mathcal{G}$ and $\mathcal{H}$ be two hypergraphs. An assignment σ is a witness if and only if*

$$Mis(\sigma) = \varnothing \wedge Cov(\sigma) = \varnothing \wedge (Sep(\sigma) = \varnothing \vee Com(\sigma) = \varnothing). \quad (1)$$

An assignment σ is *covering* whenever $Mis(\sigma) \neq \varnothing$ or $Cov(\sigma) \neq \varnothing$, and, evidently, such a σ cannot be a witness.

Most algorithms proposed in the literature essentially try different assignments by successively performing different assignment-extensions. Each extension performed induces a "reduced" instance of DUAL on which the algorithm is recursively invoked. Intuitively, the size reduction of the instance happens for two reasons. Including vertices in the new transversal of $\mathcal{G}$ increases the number of edges of $\mathcal{G}$ met by the new transversal under construction, and hence there is no need to consider these edges any longer. Symmetrically, excluding vertices from the new transversal of $\mathcal{G}$ increases the number of edges of $\mathcal{H}$ certainly not contained in the new transversal under construction, and hence, again, there is no need to consider these edges any longer.

The most common approaches, referred to as extension-types, to extend a currently considered assignment σ to an assignment σ' are:

(i) include a vertex v in the new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$, i.e., $\sigma' = \sigma + \langle \{v\}, \varnothing \rangle$;

(ii) include a vertex v as a critical vertex in the new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$ with edge $G \in Sep(\sigma)$ witnessing v's criticality, i.e., $\sigma' = \sigma + \langle \{v\}, G \setminus \{v\} \rangle$;

(iii) exclude a vertex v from the new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$ (or, equivalently, include a vertex v in the new transversal of $\mathcal{H}$ w.r.t. $\mathcal{G}$), i.e., $\sigma' = \sigma + \langle \varnothing, \{v\} \rangle$;

(iv) include a vertex v as a critical vertex in the new transversal of $\mathcal{H}$ w.r.t. $\mathcal{G}$ with edge $H \in Com(\sigma)$ witnessing v's criticality, i.e., $\sigma' = \sigma + \langle H \setminus \{v\}, \{v\} \rangle$.

3.2 Decomposition trees

We will now describe the construction of a general decomposition tree $\mathcal{T}(\mathcal{G}, \mathcal{H})$ that simultaneously represents all possible decompositions of an input DUAL instance $\langle \mathcal{G}, \mathcal{H} \rangle$ according to extension-types (ii) and (iii). This tree is of super-polynomial size. However, it will be shown later that whenever $\mathcal{H} \neq tr(\mathcal{G})$, then there must exist in $\mathcal{T}(\mathcal{G}, \mathcal{H})$ a path of length $O(\log m)$ whose end-node witnesses that $\mathcal{G}$ and $\mathcal{H}$ are not dual. Moreover, recognizing that the end-node of this path witnesses the non-duality of $\mathcal{G}$ and $\mathcal{H}$ requires low computational effort.

[1] We will often omit "w.r.t. $\mathcal{H}$" when the hypergraph $\mathcal{H}$ we are referring to is understood.

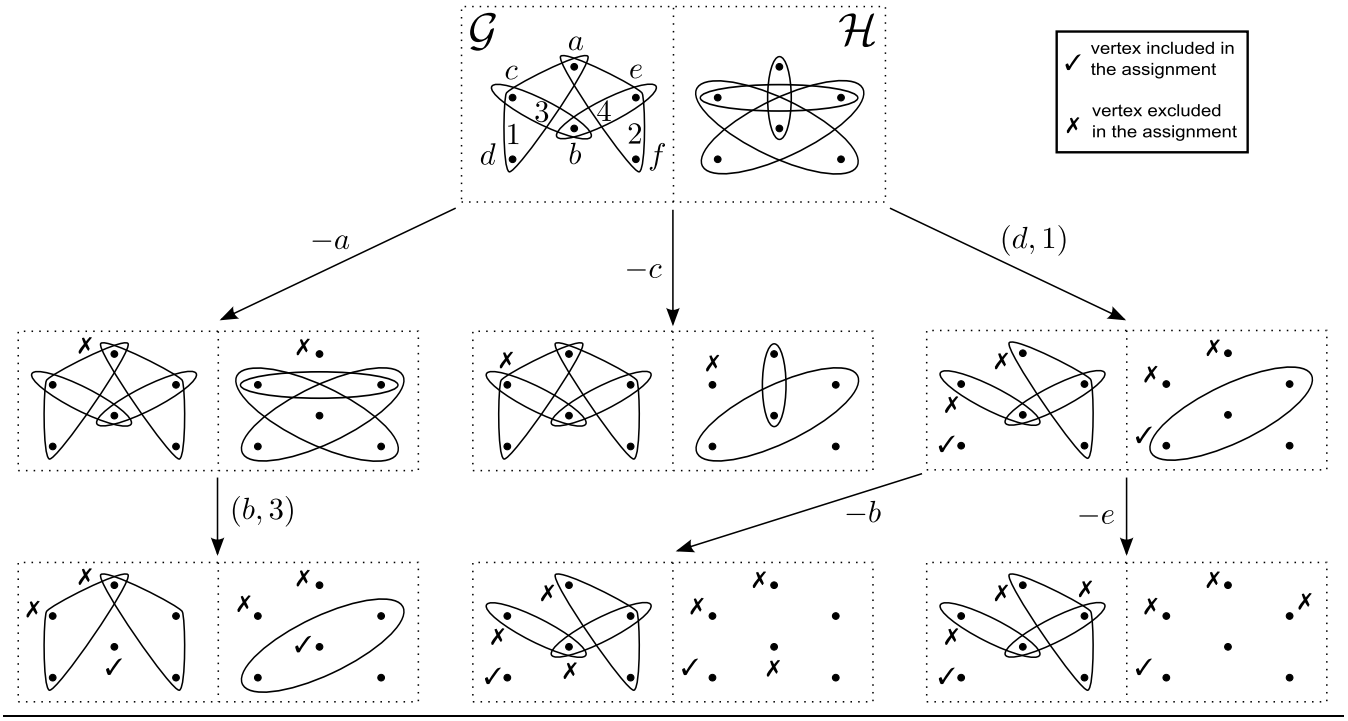

Figure 3. A subtree of the decomposition tree $\mathcal{T}(\mathcal{G},\mathcal{H})$.

Each node p of the tree $\mathcal{T}(\mathcal{G},\mathcal{H})$ is associated with an assignment σ_p. In particular, the root is labeled with the empty assignment. Node p of the tree has a child q for each assignment σ_q that can be obtained from σ_p through an elementary extension of type (ii) or (iii). The edge (p,q) is then labeled by precisely this extension.

More formally, let $\mathcal{T}(\mathcal{G},\mathcal{H}) = \langle N, A, r, \sigma, \ell\rangle$ be a tree whose nodes N are labeled by a function σ, and whose edges A are labeled by a function ℓ. The root $r \in N$ of the tree is labeled with the empty assignment $\sigma_\varepsilon = \langle\varnothing,\varnothing\rangle$. Each other node p is labeled with the assignment $\sigma_p = \langle In_p, Ex_p\rangle$ (specified below). The leaves of $\mathcal{T}(\mathcal{G},\mathcal{H})$ are all the nodes p whose assignment σ_p is covering or has no free vertex. Each non-leaf node p of $\mathcal{T}(\mathcal{G},\mathcal{H})$ has precisely the following children:

- For each free vertex v of σ_p, p has a child q such that $\sigma_q = \sigma_p + \langle\varnothing,\{v\}\rangle$, and such that the edge connecting p to q is labeled $-v$.
- For each free vertex v of σ_p, and for each $G \in Sep(\sigma_p)$ where $v \in G$, p has a child q such that $\sigma_q = \sigma_p + \langle\{v\}, G \setminus \{v\}\rangle$, and such that the edge connecting p to q is labeled (v,G).

Observe that, by this defintion of $\mathcal{T}(\mathcal{G},\mathcal{H})$, the edges leaving a node are all labeled differently, and, moreover, siblings are always differently labeled. Note, however, that different (non-sibling) nodes may have the same label, and so may edges originating from different nodes.

To give an example, consider Figure 3. Hypergraph vertices are denoted by letters, and hypergraph edges are denoted by numbers. In the tree illustrated, the root coincides with the pair of hypergraphs of Figure 1, except that the transversal $\{d,b,f\}$ of $\mathcal{G}$ is now missing in $\mathcal{H}$. The root is associated with the empty assignment σ_ε, and, correspondingly, the hypergraphs depicted with the root decomposition node are $Sep(\sigma_\varepsilon)$, and $Com(\sigma_\varepsilon)$. Each other decomposition node p represents an assignment σ_p whose included vertices are indicated by a checkmark and whose excluded vertices by a cross. In addition, each decomposition node p shows two hypergraphs representing the separated edges of $\mathcal{G}$ and the compatible edges of $\mathcal{H}$ in σ_p, respectively. The left-most edge leaving the root is labeled with $-a$ which stands for the exclusion of vertex a. This reflects the application of an extension-type (iii). On the other hand, the right-most edge leaving the root is labeled with $(d,1)$ which stands for the inclusion of vertex d as a critical vertex, along with edge 1 of $\mathcal{G}$ witnessing d's criticality in the attempted new transversal under construction. This reflects the application of an extension-type (ii). In the given example, not all but only some decomposition nodes of the tree are depicted. Observe that the bottom right decomposition node of the figure is not a leaf, because its assignment is non-covering and still contains two free vertices that can be either included (as critical vertices), or excluded. On the other hand, the bottom central node of the figure is a leaf, because its assignment is covering (in particular, edge 3 of hypergraph $\mathcal{G}$ is entirely missed).

A path $\Pi = \{\ell_1, \ell_2, \ldots, \ell_k\}$ in $\mathcal{T}(\mathcal{G},\mathcal{H})$ is a sequence of labels describing the path from the root to a node following the edges labeled in turn ℓ_1, ℓ_2,...,ℓ_k.

For example, in Figure 3, the path $\{(d,1), -e\}$ leads to the bottom right node of the figure.

Since edges leaving a node are assumed to be all different, a path identifies unequivocally a node in the tree. Given a path Π, we denote by $\mathscr{N}(\Pi)$ the end-node of Π.

The next Lemma, which shows how to compute the assignment $\sigma_{\mathscr{N}(\Pi)}$ of the node $\mathscr{N}(\Pi)$, immediately follows from the definition of the concept of path.

Lemma 2.

$$\sigma_{\mathscr{N}(\Pi)} = \langle \bigcup_{(v,G)\in\Pi} \{v\}, \big(\bigcup_{-v\in\Pi} \{v\}\big) \cup \big(\bigcup_{(v,G)\in\Pi} (G\setminus\{v\})\big)\rangle. \quad (2)$$

We will often refer to properties of the assignment σ_p as properties of the node p. For example, we say that a node p is a witness when σ_p is actually a witness.

To conclude this section let us point out some important properties of the decomposition tree $\mathcal{T}(\mathcal{G},\mathcal{H})$. First, note that each node p corresponds to the sub-instance $\mathcal{I}_p = \langle (\mathcal{G}_{V\setminus In_p})^{V\setminus(In_p\cup Ex_p)}, (\mathcal{H}_{V\setminus Ex_p})^{V\setminus(In_p\cup Ex_p)}\rangle$. Therefore $\mathcal{T}(\mathcal{G},\mathcal{H})$ is indeed a decomposition of the original instance into smaller instances. However, for our purposes, these smaller instances do not need to be explicitly represented. Secondly, note that by construction of the tree $\mathcal{T}(\mathcal{G},\mathcal{H})$, $\langle\mathcal{G},\mathcal{H}\rangle$ is a "yes"-instance of DUAL if and only if for each *non*-covering node p of $\mathcal{T}(\mathcal{G},\mathcal{H})$, $\mathcal{I}_p$ is a "yes"-instance of DUAL.[2] Therefore, for showing that $\langle\mathcal{G},\mathcal{H}\rangle$ is a "no"-instance of DUAL, it is sufficient to guess a path from the root to a node p where $\mathcal{I}_p$ is easily recognizable as a "no"-instance.

In general, such paths may be polynomially long. However, in the next section, we will show that if $\langle\mathcal{G},\mathcal{H}\rangle$ is indeed a "no"-instance then there must also exist a path of length $O(\log m)$ to some node p such that $\mathcal{I}_p$ can be recognized within complexity class TC^0 as "no"-instance of DUAL.

3.3 Logarithmic refuters

In this section we show that even though witnesses in $\mathcal{T}(\mathcal{G},\mathcal{H})$ can be at polynomial depth, if $\mathcal{G}$ and $\mathcal{H}$ are not dual then there always exists at logarithmic depth a node that is either a witness or can be easily "extended" to a witness.

Given a node p of $\mathcal{T}(\mathcal{G},\mathcal{H})$, a free vertex v of σ_p is *frequent* if v belongs to at least half of the edges in $Com(\sigma_p)$, otherwise v is *infrequent*. For example, in Figure 3 vertex c is frequent at the root because it belongs to two out of four edges of $Com(\sigma_\varepsilon)=\mathcal{H}$. On the contrary, vertex d is infrequent at the root because it belongs to only one edge of $Com(\sigma_\varepsilon)=\mathcal{H}$. Let us denote by $Freq(\sigma_p)$ and $Infreq(\sigma_p)$ the vertices frequent and infrequent in σ_p, respectively.

Assume that there exists a new transversal T of $\mathcal{G}$. An assignment $\sigma=\langle In, Ex\rangle$ is called a *precursor* of T if $\sigma \sqsubseteq T$ and σ is *not* a witness. It is easy to see that in this case $In \subset T$ and $Ex \subset \overline{T}$, for otherwise σ would be a witness.

Now, consider again the example in Figure 3. Remember that the minimal transversal $T=\{d,b,f\}$ of $\mathcal{G}$ is missing in $\mathcal{H}$. Intuitively, while searching for a new transversal of $\mathcal{G}$, at the root, vertex c is appealing for being excluded for two reasons:

- the reached node p (the second depicted child of the root) is coherent with T, and hence we can find a witness of the existence of T in the subtree rooted in p;
- the size of $Com(\sigma_p)$ is half the size of $Com(\sigma_\varepsilon)=\mathcal{H}$.

Similarly, at the root, vertex d is appealing for being included as a critical vertex with edge 1 witnessing d's criticality because the reached node q (the third depicted child of the root) is coherent with T, and $|Com(\sigma_q)| \le \frac{1}{2}|Com(\sigma_\varepsilon)|$.

More formally, let us consider a node p precursor of a new *minimal* transversal T of $\mathcal{G}$. A free vertex v of σ_p is *appealing to exclude (for σ_p) w.r.t. T* if $v \in Freq(\sigma_p)$ and $v \notin T$. On the other hand, a free vertex v of σ_p is *appealing to include (for σ_p) w.r.t. T* if $v \in Infreq(\sigma_p)$ and $v \in T$.

The halving of the size of $Com(\sigma)$, due to the inclusion or the exclusion of proper appealing vertices for σ, does not occur by chance. It depends on the frequencies with which vertices belong to hypergraph edges, and is based on a principle already exploited in algorithms proposed in the literature, e.g., by Gaur [14], and (with other frequency thresholds) by Fredman and Khachiyan [13] in their algorithm "B". Let us describe the just mentioned principle in more detail in a form suitable for our further discussion.

Lemma 3. *Let $\mathcal{G}$ and $\mathcal{H}$ be two hypergraphs satisfying the intersection property, and σ be a non-covering assignment. If $v \in Freq(\sigma)$ then $|Com(\sigma + \langle\varnothing,\{v\}\rangle)| \le \frac{1}{2}|Com(\sigma)|$. On the other hand, if $G \in Sep(\sigma)$, and $v \in G$ is such that $v \in Infreq(\sigma_p)$, then $|Com(\sigma + \langle\{v\}, G\setminus\{v\}\rangle)| \le \frac{1}{2}|Com(\sigma)|$.*

Proof. Evidently, when a frequent vertex v of $\sigma=\langle In, Ex\rangle$ is excluded, then the size of $Com(\sigma)$ is halved because at least half of the edges of $Com(\sigma)$ contain v.

Consider now the second case in which an infrequent vertex v is included into In, with edge $G \in Sep(\sigma)$ witnessing the criticality of v. The new considered assignment is $\sigma' = \sigma + \langle\{v\}, G\setminus\{v\}\rangle$. Given that the intersection property holds between $\mathcal{G}$ and $\mathcal{H}$ (and thus all the edges of $Sep(\sigma)$ intersect all the edges of $Com(\sigma)$, and vice-versa), since v is infrequent in $Com(\sigma)$ (i.e., less than half of the edges of $Com(\sigma)$ intersect G on v), at least half of the edges of $Com(\sigma)$ must intersect G on the remaining set of vertices $S = G\setminus\{v\}$. Since *all* the vertices of S are excluded in σ', all the edges of $Com(\sigma)$ intersecting G on the vertices in S do not belong to $Com(\sigma')$. Therefore $|Com(\sigma')| \le \frac{1}{2}|Com(\sigma)|$. □

We say that the assignment σ is a *saturated precursor* of a new *minimal* transversal T of $\mathcal{G}$ when σ is a precursor of T, and no free vertex of σ is appealing to exclude or include for σ w.r.t. T. The next lemma, which states the most important property of saturated precursors, immediately follows from the concept of saturated precursor.

Lemma 4. *Let $\mathcal{G}$ and $\mathcal{H}$ be two hypergraphs, and T be a new* minimal *transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$. If σ is a saturated precursor of T, then the* augmented assignment *$\sigma^+ = \langle In \cup Freq(\sigma), Ex \cup Infreq(\sigma)\rangle$ is such that $\sigma^+ = \langle T, \overline{T}\rangle$, and thus σ^+ is a witness.*

We now prove the mentioned crucial property of $\mathcal{T}(\mathcal{G},\mathcal{H})$.

Lemma 5. *Let $\mathcal{G}$ and $\mathcal{H}$ be two hypergraphs satisfying the intersection property. Then, $\mathcal{G}$ and $\mathcal{H}$ are not dual if and only if there exists in $\mathcal{T}(\mathcal{G},\mathcal{H})$ a witness or a saturated precursor at depth $O(\log|\mathcal{H}|)$.*

Proof. Obviously, if there exists a witness or a saturated precursor within logarithmic depth, then there exists a new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$.

Assume now that $\mathcal{G}$ and $\mathcal{H}$ are not dual, and let T be a new *minimal* transversal of $\mathcal{G}$. We are going to show that there exists in $\mathcal{T}(\mathcal{G},\mathcal{H})$ a path Π of logarithmic length, starting from the root, such that $\mathscr{N}(\Pi)$ is either a witness coherent with T or is a saturated precursor of T.

Let p be a generic node such that σ_p is a *non*-saturated precursor of T (i.e., a precursor of T that is *not* saturated). By the fact that p is a non-saturated precursor of a minimal transversal, and that each vertex of a minimal transversal is critical (see Section 2), there is a child of p, say q, such that σ_q is coherent with T, and σ_q is obtained from σ_p through the inclusion (as a critical vertex) or the exclusion of an appealing vertex v for σ_p w.r.t. T.

In particular, if v is appealing to include for σ_p w.r.t. T then q is chosen by including v as a critical vertex, with an appropriate edge $G \in Sep(\sigma_p)$, such that $G \cap T = \{v\}$, witnessing the criticality of v in T. Otherwise, if v is appealing to exclude for σ_p w.r.t. T then q is chosen by excluding v.

Note that the empty assignment σ_ε, associated with the root of $\mathcal{T}(\mathcal{G},\mathcal{H})$, is a non-saturated precursor of T. Hence there exists a sequence of nodes $\Sigma=(p_0,p_1,\ldots,p_k)$, where p_0 is the root, such that all the nodes of Σ are coherent with T, and each node p_i is a child of p_{i-1} obtained through the inclusion or the exclusion of an appealing vertex for $\sigma_{p_{i-1}}$ w.r.t. T.

Let Σ be a maximum length sequence having the just mentioned property. Since Σ is of maximum length, node p_k is *not* a *non*-saturated precursor of T (for otherwise there would be a child of p_k allowing to extend Σ). Hence, two are the cases: either p_k is

[2] For further details please see the extended technical report [20].

a saturated precursor of T, or p_k is not a precursor of T at all. For the second case, observe that σ_{p_k} is coherent with T and therefore if p_k is not a precursor of T then p_k must be a witness (coherent with T).

To conclude, by the definition of appealing vertex, it follows that $|Com(\sigma_{p_i})| \leq \frac{1}{2}|Com(\sigma_{p_{i-1}})|$, for all $1 \leq i \leq k$ (see Lemma 3). Therefore Σ contains $O(\log |\mathcal{H}|)$ nodes, and hence there exists a path Π of logarithmic length from the root to p_k. □

4. A new upper bound for the Dual problem

4.1 A non-deterministic approach to Dual

In this section we present our new nondeterministic algorithm ND-NOTDUAL for $\overline{\text{DUAL}}$. Unlike previous algorithms, ND-NOTDUAL uses the novel data structure $\mathcal{T}(\mathcal{G}, \mathcal{H})$ as defined in the previous section. To prove the correctness of the algorithm we will use ideas by Gaur [14]. However, it will become clear that our algorithm differs in essential aspects from Gaur's deterministic algorithm.

To disprove that two hypergraphs $\mathcal{G}$ and $\mathcal{H}$ are dual, we know that it is sufficient to show either that the intersection property does not hold between them, or that exists a new minimal transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$. Intuitively, our algorithm, to compute such a new minimal transversal, guesses in $\mathcal{T}(\mathcal{G}, \mathcal{H})$ a path of logarithmic length leading to a witness or a saturated precursor.

The pseudo-code of the algorithm ND-NOTDUAL is listed as Algorithm 1. In the pseudo-code of the algorithm, "**accept**" and "**reject**" are two commands causing a transition to a final accepting state and to a final rejecting state, respectively.

Algorithm 1 A nondeterministic algorithm for DUAL.

```
 1: procedure ND-NotDual(G, H)
 2:     Π ← guess(A path in T(G, H) of length O(log |H|));
 3:     if ¬Check-IntersectionProperty(G, H) then
 4:         accept;
 5:     end if
 6:     if ¬Check-ConsistencyGuess(G, H, Π) then
 7:         reject;
 8:     end if
 9:     if Check-Witness(G, H, Π) then
10:         accept;
11:     end if
12:     if Check-WitnessAugmented(G, H, Π) then
13:         accept;
14:     end if
15:     reject;
16: end procedure
```

The four checking-procedures used in the algorithm implement the four deterministic tests needed after the guess is carried out. The aims of the subprocedures are the following.

CHECK-INTERSECTIONPROPERTY[3] checks whether the intersection property holds between the hypergraphs $\mathcal{G}$ and $\mathcal{H}$.

CHECK-CONSISTENCYGUESS checks whether the guess is consistent. A guess is consistent if it does not include and exclude a vertex at the same time, and hence gives rise to a proper assignment $\sigma_{\mathcal{N}(\Pi)} = \langle In_{\mathcal{N}(\Pi)}, Ex_{\mathcal{N}(\Pi)} \rangle$ where $In_{\mathcal{N}(\Pi)} \cap Ex_{\mathcal{N}(\Pi)} = \varnothing$.

CHECK-WITNESS and CHECK-WITNESSAUGMENTED check whether assignment $\sigma_{\mathcal{N}(\Pi)}$ and $\sigma_{\mathcal{N}(\Pi)}{}^{+}$ witnesses the existence of a new transversal of $\mathcal{G}$ w.r.t. $\mathcal{H}$, respectively. In order to perform these checks, the condition of equation (1) (of Section 3) is evaluated on $\sigma_{\mathcal{N}(\Pi)}$ and $\sigma_{\mathcal{N}(\Pi)}{}^{+}$, respectively. If $\sigma_{\mathcal{N}(\Pi)}$ or $\sigma_{\mathcal{N}(\Pi)}{}^{+}$ is a witness, then $\mathcal{G}$ and $\mathcal{H}$ are not dual.

Note that the algorithm ND-NOTDUAL can be somewhat simplified, as not all the deterministic tests are strictly necessary. In fact, it can be shown that checking the consistency of the guess is not strictly needed, and that it is sufficient to perform the witness test on the augmented guessed assignment only. These improvements, which, however, do not give rise to better complexity bounds, will be further analyzed in a forthcoming extended technical report [20].

Theorem 6. *Let $\mathcal{G}$ and $\mathcal{H}$ be two hypergraphs. Then, there exists a computation branch of* ND-NOTDUAL$(\mathcal{G}, \mathcal{H})$ *halting in an accepting state if and only if $\mathcal{G}$ and $\mathcal{H}$ are not dual.*

Proof. At first, algorithm ND-NOTDUAL guesses a path of logarithmic length in $\mathcal{T}(\mathcal{G}, \mathcal{H})$. Intuitively we look for a path (if one exists) leading to a node that is either a witness or a saturated precursor. Then the deterministic tests are performed.

If $\mathcal{G}$ and $\mathcal{H}$ do not satisfy the intersection property such a condition is recognized by test CHECK-INTERSECTIONPROPERTY, and the algorithm correctly accepts.

Otherwise it is checked whether the guess is consistent in order to be sure that the two final tests rely on meaningful assignments. Inconsistent guesses are rejected.

Remember that, by Lemma 5, if the intersection property holds between $\mathcal{G}$ and $\mathcal{H}$, which at this stage of the algorithm is guaranteed, then the input hypergraphs are not dual iff there exists a witness or a saturated precursor at logarithmic depth. Therefore, whenever $\mathcal{G}$ and $\mathcal{H}$ are not dual, we can guess a path of logarithmic length leading to a node that is a witness or a saturated precursor. In those specific branches one of the tests CHECK-WITNESS or CHECK-WITNESSAUGMENTED returns `true` (see Lemma 4) and hence the algorithm halts in an accepting state.

Observe that those computation branches in which $\sigma_{\mathcal{N}(\Pi)}$ and $\sigma_{\mathcal{N}(\Pi)}{}^{+}$ are not witnesses correctly terminate in a rejecting state (line 15). □

Note that our algorithm, while partly inspired by Gaur's ideas, is fundamentally different from Gaur's algorithm [14, 15]. In particular, Gaur's algorithm may extend the set In of included vertices of an intermediate assignment $\sigma = \langle In, Ex \rangle$ in a single step by several vertices and not just one. In our approach this is only possible for end-nodes of the path. Moreover, algorithm ND-NOTDUAL could identify a witness by guessing a path of logarithmic length that is not a legal path according to Gaur because the single assignment-extensions are not chosen according to frequency counts. In fact, unlike Gaur's algorithm, ND-NOTDUAL performs frequency counts only at the terminal nodes of a path.

4.2 Logical analysis of the Dual problem

In order to prove that $\overline{\text{DUAL}} \in \text{GC}(\log^2 m, \text{TC}^0)$, we first express the deterministic tests performed by ND-NOTDUAL in FO(COUNT) which is first order logic augmented with the counting quantifiers "$\exists! n$". This quantifier has the following semantics. A formula $(\exists! n\ x)(\phi(x))$ always evaluates to `true` and assigns to n the number of domain values a for which $\phi(a)$ evaluates to `true`. For more precise definitions see [25, 34]. Note that first order logic augmented with the majority quantifiers (FOM) is known to be equivalent to FO(COUNT) [25]. The model checking problem for both logics is complete for the class TC^0 [25, 34].

With a pair of hypergraphs $\langle \mathcal{G}, \mathcal{H} \rangle$ we associate a relational structure $\mathcal{A}_{\langle \mathcal{G}, \mathcal{H} \rangle}$. Essentially we represent hypergraphs through their incidence graphs. In particular, the universe $A_{\langle \mathcal{G}, \mathcal{H} \rangle}$ of $\mathcal{A}_{\langle \mathcal{G}, \mathcal{H} \rangle}$ consists of the vertices of V, an object for each hyperedge of the

[3] This property does not depend on the guessed path, and could thus be checked at the beginning of the algorithm, before the guess is made. However, for uniformity, and to adhere to a strict guess-and-check paradigm we check it after the guess.

two hypergraphs, and two more objects for the two hypergraphs, i.e., $A_{\langle \mathcal{G},\mathcal{H}\rangle} = V \cup \{g_G \mid G \in \mathcal{G}\} \cup \{h_H \mid H \in \mathcal{H}\} \cup \{\mathcal{G}, \mathcal{H}\}$. By a slight overloading of notation, we assume that the elements $\mathcal{G}$ and $\mathcal{H}$ that represent the input hypergraphs $\mathcal{G}$ and $\mathcal{H}$ are also available as constants $\mathcal{G}$ and $\mathcal{H}$ in our signature.

The relations of $\mathcal{A}_{\langle \mathcal{G},\mathcal{H}\rangle}$ are as follows: $Vertex(x)$ is a unary relation indicating that object x is a vertex; $Hyp(x)$ is a unary relation indicating that object x is a hypergraph; $EdgeOf(x, y)$ is a binary relation indicating that object x is an edge of the hypergraph identified by object y; and $In(x, y)$ is the binary incidence relation indicating that object x is a vertex belonging to the edge identified by object y.

We also need to represent through relations the guessed path Π. Remember that in Π there are elements of two types: $-v$ where v is a vertex, and (v, G) where v is a vertex and G is an edge of $\mathcal{G}$.[4] We assume a unary relation S_1 storing those tuples $\langle v\rangle$ where v is a vertex such that $-v \in \Sigma$, moreover we assume a binary relation S_2 containing those tuples $\langle v, G\rangle$ where v is a vertex and G is an edge such that $(v, G) \in \Sigma$.

Let us now make an important observation about the algorithm. In order to disprove the duality of $\mathcal{G}$ and $\mathcal{H}$, the algorithm guesses a path Π and then checks that the final node of this path is a witness or a saturated precursor of a missing transversal. For this check the order of the labels of the path is actually not relevant. What is relevant is merely the set of labels. Therefore the above relational representation of a guessed path is totally sufficient. (The order is only needed in the proof of the correctness of the algorithm.)

We use the following "macros" in our first order formulas:

$$v \in V \equiv Vertex(v)$$
$$g \in \mathcal{G} \equiv Hyp(\mathcal{G}) \wedge EdgeOf(g, \mathcal{G})$$
$$h \in \mathcal{H} \equiv Hyp(\mathcal{H}) \wedge EdgeOf(h, \mathcal{H})$$
$$v \in g \equiv In(v, g)$$

We are now ready to prove some intermediate results.

Lemma 7. *Let $\mathcal{G}$ and $\mathcal{H}$ be two hypergraphs. Deciding whether $\mathcal{G}$ and $\mathcal{H}$ satisfy the intersection property is expressible in FO.*

Proof. We need to show formulas expressing that all the edges of $\mathcal{G}$ are minimal transversal of $\mathcal{H}$, and vice-versa.

The formulas to evaluate whether an edge of $\mathcal{G}$ (resp. $\mathcal{H}$) is a transversal of $\mathcal{H}$ (resp. $\mathcal{G}$), and to check whether an edge of $\mathcal{G}$ (resp. $\mathcal{H}$) is *not* a minimal transversal of $\mathcal{H}$ (resp. $\mathcal{G}$) are:

$$\begin{aligned}
&tr\text{-}of\text{-}\mathcal{H}(g) \equiv g \in \mathcal{G} \wedge \\
&\quad (\forall h)(h \in \mathcal{H} \rightarrow (\exists v)(v \in V \wedge v \in g \wedge v \in h)) \\
&nonMin\text{-}tr\text{-}of\text{-}\mathcal{H}(g) \equiv g \in \mathcal{G} \wedge (\exists v)(v \in V \wedge v \in g \wedge \\
&\quad (\forall h)(h \in \mathcal{H} \rightarrow (\exists w)(w \in V \wedge w \neq v \wedge w \in g \wedge w \in h))) \\
&tr\text{-}of\text{-}\mathcal{G}(h) \equiv h \in \mathcal{H} \wedge \\
&\quad (\forall g)(g \in \mathcal{G} \rightarrow (\exists v)(v \in V \wedge v \in g \wedge v \in h)) \\
&nonMin\text{-}tr\text{-}of\text{-}\mathcal{G}(h) \equiv h \in \mathcal{H} \wedge (\exists v)(v \in V \wedge v \in h \wedge \\
&\quad (\forall g)(g \in \mathcal{G} \rightarrow (\exists w)(w \in V \wedge w \neq v \wedge w \in g \wedge w \in h)))
\end{aligned}$$

And to conclude, we show the formula verifying that the intersection property holds between the two hypergraphs.

$$\begin{aligned}
&intersection\text{-}property \equiv \\
&\quad (\forall g)(g \in \mathcal{G} \rightarrow tr\text{-}of\text{-}\mathcal{H}(g) \wedge \neg nonMin\text{-}tr\text{-}of\text{-}\mathcal{H}(g)) \wedge \\
&\quad (\forall h)(h \in \mathcal{H} \rightarrow tr\text{-}of\text{-}\mathcal{G}(h) \wedge \neg nonMin\text{-}tr\text{-}of\text{-}\mathcal{G}(h))
\end{aligned}$$

□

[4] Note that an edge G in a label of a path is given by its identifier and not by the explicit list of its vertices.

We say that the guess is congruent if for every guessed tuple $\langle x\rangle \in S_1$ the object x is actually a vertex, and for every tuple $\langle x, y\rangle \in S_2$ the object y is actually an edge belonging to $\mathcal{G}$ containing the vertex identified by the object x.

Lemma 8. *Let $\mathcal{G}$ and $\mathcal{H}$ be two hypergraphs, and Π be a (guessed) path of $\mathcal{T}(\mathcal{G}, \mathcal{H})$. Deciding the congruency and the consistency of Π is expressible in FO.*

Proof. The congruency of the guessed path can be checked through:

$$\begin{aligned}
&congruentGuess \equiv (\forall v)(S_1(v) \rightarrow v \in V) \wedge \\
&\quad (\forall w, g)(S_2(w, g) \rightarrow w \in V \wedge g \in \mathcal{G} \wedge w \in g)
\end{aligned}$$

The consistency check is divided in two formulas, one verifying whether there is *in*consistency on a specific vertex, and the other checking the overall consistency.

$$\begin{aligned}
&inconsistent(w) \equiv w \in V \wedge (\exists g)(S_2(w, g) \wedge \\
&\quad (S_1(w) \vee (\exists v, h)(S_2(v, h) \wedge v \neq w \wedge w \in h))) \\
&consistentGuess \equiv (\forall v)(v \in V \rightarrow \neg inconsistent(v))
\end{aligned}$$

□

We now focus our attention on the complexity of checking whether $\sigma_{\mathcal{N}(\Pi)}$ is a witness or not.

Lemma 9. *Let $\mathcal{G}$ and $\mathcal{H}$ be two hypergraphs, and Π be a (guessed) path of $\mathcal{T}(\mathcal{G}, \mathcal{H})$. Deciding whether $\sigma_{\mathcal{N}(\Pi)}$ is a witness is expressible in FO.*

Proof. Essentially we need to prove that is possible to express in first order logic the condition of equation (1) (of Section 3) on $\sigma_{\mathcal{N}(\Pi)} = \langle In_{\mathcal{N}(\Pi)}, Ex_{\mathcal{N}(\Pi)}\rangle$. Remember that $\sigma_{\mathcal{N}(\Pi)}$ is not explicitly represented, but it can be evaluated from Π through formula (2) of Section 3.

Let us define the following two formulas serving the purpose to evaluate if a vertex belongs to $In_{\mathcal{N}(\Pi)}$ or $Ex_{\mathcal{N}(\Pi)}$.

$$\begin{aligned}
&I\text{-}guess(v) \equiv v \in V \wedge (\exists g)(S_2(v, g)) \\
&E\text{-}guess(v) \equiv v \in V \wedge \\
&\quad (S_1(v) \vee (\exists w, g)(S_2(w, g) \wedge w \neq v \wedge v \in g))
\end{aligned}$$

The formulas evaluating whether an edge belongs to $Mis(\sigma_{\mathcal{N}(\Pi)})$, to $Cov(\sigma_{\mathcal{N}(\Pi)})$, to $Sep(\sigma_{\mathcal{N}(\Pi)})$, and to $Com(\sigma_{\mathcal{N}(\Pi)})$, are, respectively:

$$\begin{aligned}
&mis(g) \equiv g \in \mathcal{G} \wedge (\forall v)((v \in V \wedge v \in g) \rightarrow E\text{-}guess(v)) \\
&cov(h) \equiv h \in \mathcal{H} \wedge (\forall v)((v \in V \wedge v \in h) \rightarrow I\text{-}guess(v)) \\
&sep(g) \equiv g \in \mathcal{G} \wedge (\forall v)((v \in V \wedge v \in g) \rightarrow \neg I\text{-}guess(v)) \\
&com(h) \equiv h \in \mathcal{H} \wedge (\forall v)((v \in V \wedge v \in h) \rightarrow \neg E\text{-}guess(v))
\end{aligned}$$

Finally the formula verifying that the assignment $\sigma_{\mathcal{N}(\Pi)}$ is a witness is as follows. This formula essentially encodes the condition of equation (1) of Section 3.

$$\begin{aligned}
&guessWitness \equiv \\
&\quad (\forall g)(g \in \mathcal{G} \rightarrow \neg mis(g)) \wedge (\forall h)(h \in \mathcal{H} \rightarrow \neg cov(h)) \wedge \\
&\quad ((\forall g)(g \in \mathcal{G} \rightarrow \neg sep(g)) \vee (\forall h)(h \in \mathcal{H} \rightarrow \neg com(h)))
\end{aligned}$$

□

To conclude our complexity analysis of the deterministic tests performed by ND-NOTDUAL, let us formulate the property that ${\sigma_{\mathcal{N}(\Pi)}}^+$ is a witness in FO(COUNT).

Lemma 10. *Let $\mathcal{G}$ and $\mathcal{H}$ be two hypergraphs, and Π be a (guessed) path of $\mathcal{T}(\mathcal{G}, \mathcal{H})$. Deciding whether ${\sigma_{\mathcal{N}(\Pi)}}^+$ is a witness is expressible in FO(COUNT).*

Proof. Let $\sigma_{\mathcal{N}(\Pi)} = \langle In_{\mathcal{N}(\Pi)}, Ex_{\mathcal{N}(\Pi)}\rangle$ be the assignment associated with the end-node $\mathcal{N}(\Pi)$ of the path Π. Evaluating whether

$\sigma_{\mathcal{N}(\Pi)}{}^{+}$ is a witness is a task very similar to the task of evaluating whether $\sigma_{\mathcal{N}(\Pi)}$ is a witness as done in the proof of Lemma 9. The only difference here is that vertices considered included are $In_{\mathcal{N}(\Pi)} \cup \mathit{Freq}(\sigma_{\mathcal{N}(\Pi)})$, and vertices considered excluded are $Ex_{\mathcal{N}(\Pi)} \cup \mathit{Infreq}(\sigma_{\mathcal{N}(\Pi)})$.

We first exhibit the formulas to verify whether a given vertex is frequent in $\sigma_{\mathcal{N}(\Pi)}$. These formulas are the only ones in which we actually use the counting quantifier. Integer division by 2, which can be easily computed by a right shift of one bit in the binary representations of numbers, and majority testing are both feasible in FO(COUNT) [34]. The following formulas evaluate respectively: the number of edges in $Com(\sigma_{\mathcal{N}(\Pi)})$, the number of edges in $Com(\sigma_{\mathcal{N}(\Pi)})$ containing a given vertex v, and whether v is frequent in $\sigma_{\mathcal{N}(\Pi)}$. Note that $I\text{-}guess(\cdot)$, $E\text{-}guess(\cdot)$, and $com(\cdot)$ are the formulas defined in the proof of Lemma 9.

$$\begin{aligned}
&\mathit{count\text{-}com}(n) \equiv (\exists! n\, h)(h \in \mathcal{H} \wedge \mathit{com}(h)) \\
&\mathit{count\text{-}com\text{-}inc}(v,n) \equiv v \in V \wedge \\
&\quad (\exists! n\, h)(h \in \mathcal{H} \wedge \mathit{com}(h) \wedge v \in h) \\
&\mathit{freq}(v) \equiv v \in V \wedge \neg I\text{-}guess(v) \wedge \neg E\text{-}guess(v) \wedge \\
&\quad (\exists n, m, o)(\mathit{count\text{-}com}(n) \wedge \mathit{count\text{-}com\text{-}inc}(v,m) \wedge \\
&\quad o = n/2 \wedge m \geq o)
\end{aligned}$$

Having defined a formula to evaluate whether a vertex is frequent in $\sigma_{\mathcal{N}(\Pi)}$, we now show the formulas computing the included and excluded vertices of the augmented assignment.

$$\begin{aligned}
I\text{-}aug(v) &\equiv v \in V \wedge (I\text{-}guess(v) \vee \mathit{freq}(v)) \\
E\text{-}aug(v) &\equiv v \in V \wedge (E\text{-}guess(v) \vee \neg \mathit{freq}(v))
\end{aligned}$$

Now we exhibit the formulas encoding the evaluation of the condition of equation (1) of Section 3 on the augmented assignment. Note that the following formulas have essentially the same purpose of those in proof of Lemma 9.

$$\begin{aligned}
&\mathit{mis\text{-}aug}(g) \equiv g \in \mathcal{G} \wedge (\forall v)((v \in V \wedge v \in g) \rightarrow E\text{-}aug(v)) \\
&\mathit{cov\text{-}aug}(h) \equiv h \in \mathcal{H} \wedge (\forall v)((v \in V \wedge v \in h) \rightarrow I\text{-}aug(v)) \\
&\mathit{sep\text{-}aug}(g) \equiv g \in \mathcal{G} \wedge (\forall v)((v \in V \wedge v \in g) \rightarrow \neg I\text{-}aug(v)) \\
&\mathit{com\text{-}aug}(h) \equiv h \in \mathcal{H} \wedge (\forall v)((v \in V \wedge v \in h) \rightarrow \neg E\text{-}aug(v))
\end{aligned}$$

$$\begin{aligned}
\mathit{guessAugWitness} \equiv\ & (\forall g)(g \in \mathcal{G} \rightarrow \neg \mathit{mis\text{-}aug}(g)) \wedge \\
& (\forall h)(h \in \mathcal{H} \rightarrow \neg \mathit{cov\text{-}aug}(h)) \wedge \\
& ((\forall g)(g \in \mathcal{G} \rightarrow \neg \mathit{sep\text{-}aug}(g)) \vee \\
& (\forall h)(h \in \mathcal{H} \rightarrow \neg \mathit{com\text{-}aug}(h)))
\end{aligned}$$

□

We now state our main result.

Theorem 11. *Let $\mathcal{G}$ and $\mathcal{H}$ be two hypergraphs. Deciding whether $\mathcal{G}$ and $\mathcal{H}$ are not dual is feasible in* $\mathrm{GC}(\log^2 m, \mathrm{TC}^0)$.

Proof. Lemmas 7, 8, 9, and 10 show that the four deterministic checks performed by algorithm ND-NotDual are expressible in FO(COUNT). Hence these tests are feasible in logtime-uniform TC^0 [25, 34].

Moreover, by analyzing the algorithm ND-NotDual it is evident that only $O(\log^2 m)$ nondeterministic bits are sufficient to be guessed. Indeed, if $\mathcal{G}$ and $\mathcal{H}$ do not satisfy the intersection property then the guessed path Π is completely ignored, because $\mathcal{G}$ and $\mathcal{H}$ are directly recognized not to be dual, and hence is totally irrelevant what the guessed bits are. On the other hand, if $\mathcal{G}$ and $\mathcal{H}$ satisfy the intersection property then, by Lemma 5, there exists in $\mathcal{T}(\mathcal{G}, \mathcal{H})$ a path of logarithmic length whose end-node p is a witness or a saturated precursor if and only if $\mathcal{G}$ and $\mathcal{H}$ are not dual. Since the intersection property holds between $\mathcal{G}$ and $\mathcal{H}$, $|V| \leq |\mathcal{G}| \cdot |\mathcal{H}|$ holds (see Section 2), and hence $O(\log m)$ bits are sufficient to correctly represent each single label of a path Π (if one exists) leading to such a node p. Observe that $|\Pi| \in O(\log |\mathcal{H}|)$, which is also $O(\log m)$, and hence obviously the whole path Π can be correctly represented with $O(\log^2 m)$ bits.

Therefore, $\overline{\textsc{Dual}}$ belongs to $\mathrm{GC}(\log^2 m, \mathrm{TC}^0)$. □

Acknowledgments

G. Gottlob's work was supported by the ERC grant 246858 (DIADEM). E. Malizia's work was carried out while visiting the Department of Computer Science of the University of Oxford (UK), and was mainly supported by the European Commission through the European Social Fund and by the Region of Calabria. Malizia received additional funding from the above mentioned ERC grant. We are grateful to the anonymous reviewers for their competent and helpful comments.

References

[1] J. C. Bioch and T. Ibaraki. Complexity of identification and dualization of positive Boolean functions. *Information and Computation*, 123 (1):50–63, 1995.

[2] E. Boros and K. Makino. A fast and simple parallel algorithm for the monotone duality problem. In S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. Nikoletseas, and W. Thomas, editors, *Proc. of ICALP 2009, Part I*, pages 183–194, 2009.

[3] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On the complexity of generating maximal frequent and minimal infrequent sets. In H. Alt and A. Ferreira, editors, *Proc. of STACS 2002*, pages 133–141, 2002.

[4] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On maximal frequent and minimal infrequent sets in binary matrices. *Annals of Mathematics and Artificial Intelligence*, 39(3):211–221, 2003.

[5] L. Cai and J. Chen. On the amount of nondeterminism and the power of verifying. *SIAM Journal on Computing*, 26(3):733–750, 1997.

[6] S. Cook and P. Nguyen. *Logical foundations of proof complexity*. Cambridge University Press, Cambridge, UK, January 2010.

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

[8] T. Eiter and G. Gottlob. Hypergraph transversal computation and related problems in Logic and AI. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Proc. of JELIA 2002*, pages 549–564, 2002.

[9] T. Eiter and K. Makino. Generating all abductive explanations for queries on propositional Horn theories. In M. Baaz and J. A. Makowsky, editors, *Proc. of CSL 2003*, pages 197–211, 2003.

[10] T. Eiter, G. Gottlob, and K. Makino. New results on monotone dualization and generating hypergraph transversals. *SIAM Journal on Computing*, 32(2):514–537, 2003.

[11] T. Eiter, K. Makino, and G. Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156 (11):2035–2049, 2008.

[12] K. M. Elbassioni. On the complexity of monotone dualization and generating minimal hypergraph transversals. *Discrete Applied Mathematics*, 156(11):2109–2123, 2008.

[13] M. L. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21(3): 618–628, 1996.

[14] D. R. Gaur. *Satisfiability and Self-Duality of Monotone Boolean Functions*. PhD thesis, School of Computing Science, Simon Fraser University, Canada, 1999.

[15] D. R. Gaur and R. Krishnamurti. Average case self-duality of monotone Boolean functions. In A. Y. Tawfik and S. D. Goodwin, editors, *Proc. of Canadian AI 2004*, pages 322–338, 2004.

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

[17] J. Goldsmith, M. A. Levy, and M. Mundhenk. Limited nondeterminism. *ACM SIGACT News*, 27(2):20–29, 1996.

[18] G. Gottlob. Deciding monotone duality and identifying frequent itemsets in quadratic logspace. In R. Hull and W. Fan, editors, *Proc. of PODS 2013*, pages 25–36, 2013.

[19] G. Gottlob and L. Libkin. Investigations on Armstrong relations, dependency inference, and excluded functional dependencies. *Acta Cybernetica*, 9(4):385–402, 1990.

[20] G. Gottlob and E. Malizia. Achieving new upper bounds for the hypergraph duality problem through logic. Technical report to appear shortly on *arXiv.org*, 2014.

[21] R. Greiner, B. A. Smith, and R. W. Wilkerson. A correction to the algorithm in Reiter's theory of diagnosis. *Artificial Intelligence*, 41 (1):79–88, 1989.

[22] D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data mining, hypergraph transversals, and machine learning. In A. O. Mendelzon and Z. M. Özsoyoglu, editors, *Proc. of PODS 1997*, pages 209–216, 1997.

[23] M. Hagen. *Algorithmic and Computational Complexity Issues of MONET*. Cuvillier Verlag, Göttingen, Germany, 2008.

[24] T. A. Henzinger, O. Kupferman, and R. Majumdar. On the universal and existential fragments of the μ-calculus. In H. Garavel and J. Hatcliff, editors, *Proc. of TACAS 2003*, pages 49–64, 2003.

[25] N. Immerman. *Descriptive Complexity*. Springer-Verlag, New York, NY, USA, 1999.

[26] M. Jurdziński. Deciding the winner in parity games is in UP $\cap$ coUP. *Information Processing Letters*, 68(3):119–124, 1998.

[27] D. J. Kavvadias and E. C. Stavropoulos. Monotone Boolean dualization is in co-NP[$\log^2 n$]. *Information Processing Letters*, 85(1):1–6, 2003.

[28] D. J. Kavvadias, C. H. Papadimitriou, and M. Sideri. On Horn envelopes and hypergraph transversals. In K.-W. Ng, P. Raghavan, N. V. Balasubramanian, and F. Y. L. Chin, editors, *Proc. of ISAAC '93*, pages 399–405, 1993.

[29] H. Mannila and K.-J. Räihä. On the complexity of inferring functional dependencies. *Discrete Applied Mathematics*, 40(2):237–243, 1992.

[30] H. Mannila and K.-J. Räihä. Algorithms for inferring functional dependencies from relations. *Data & Knowledge Engineering*, 12(1): 83–99, 1994.

[31] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3): 241–258, 1997.

[32] N. Mishra and L. Pitt. Generating all maximal independent sets of bounded-degree hypergraphs. In Y. Freund and R. Schapire, editors, *Proc. of COLT '97*, pages 211–217, 1997.

[33] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[34] H. Vollmer. *Introduction to Circuit Complexity*. Springer-Verlag, Berlin Heidelberg, Germany, 1999.