

ACHIEVING NEW UPPER BOUNDS FOR THE HYPERGRAPH DUALITY PROBLEM THROUGH LOGIC*

GEORG GOTTLÖB[†] AND ENRICO MALIZIA[‡]

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} or, equivalently, the transversal hypergraph of \mathcal{H}). This problem is equivalent to deciding whether two given nonredundant monotone disjunctive normal forms/conjunctive normal forms are dual. It is known that $\overline{\text{DUAL}}$, the complementary problem to DUAL, is in $\text{GC}(\log^2 n, \text{PTIME})$, where $\text{GC}(f(n), \mathcal{C})$ 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 $\text{GC}(\log^2 n, \text{LOGSPACE})$. In this paper we prove this conjecture and actually place the $\overline{\text{DUAL}}$ problem into the complexity class $\text{GC}(\log^2 n, \text{TC}^0)$ which is a subclass of $\text{GC}(\log^2 n, \text{LOGSPACE})$. We here refer to the logtime-uniform version of TC^0 , which corresponds to $\text{FO}(\text{COUNT})$, 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 DUAL that requires one 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 $\text{FO}(\text{COUNT})$. From this result, by the well-known inclusion $\text{TC}^0 \subseteq \text{LOGSPACE}$, it follows that DUAL also belongs to $\text{DSPACE}[\log^2 n]$. Finally, by exploiting the principles on which the proposed nondeterministic algorithm is based, we devise a deterministic algorithm that, given two hypergraphs \mathcal{G} and \mathcal{H} , computes in quadratic logspace a transversal of \mathcal{G} missing in \mathcal{H} .

Key words. 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

AMS subject classifications. 03B10, 03B70, 03B80, 03C80, 05C65, 05C85, 06E30, 68Q10, 68Q19, 68Q25, 68R10, 68W05, 68W40, 94C10

DOI. 10.1137/15M1027267

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 nondeterministic 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

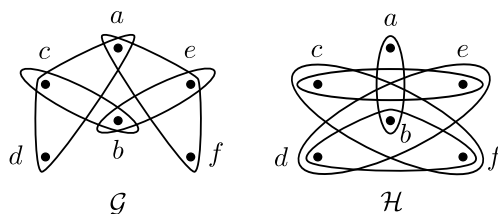
*Received by the editors June 22, 2015; accepted for publication (in revised form) November 27, 2017; published electronically April 12, 2018. This paper is an extended version of a paper which appeared in the Proceedings of CSL-LICS, ACM, New York, 2014, 43 [27].

<http://www.siam.org/journals/sicomp/47-2/M102726.html>

Funding: The first author's work was supported by the EPSRC Programme Grant EP/M025268/ "VADA: Value Added Data Systems – Principles and Architecture." The second author's work was mainly supported by the European Commission through the European Social Fund and by the Region of Calabria (Italy). Malizia received additional funding from the ERC grant 246858 (DIADEM) and the above mentioned EPSRC grant "VADA."

[†]Department of Computer Science, University of Oxford, UK (georg.gottlob@cs.ox.ac.uk).

[‡]Department of Computer Science, University of Exeter, UK (e.malizia@exeter.ac.uk).

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

of its edges is contained in any other of its edges. A *transversal* or *hitting set* of a hypergraph $\mathcal{G} = \langle V, E \rangle$ 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 $\mathcal{G} = \langle V, E \rangle$ is denoted by $tr(\mathcal{G})$. Note that $tr(\mathcal{G})$, which is referred to as the *dual*¹ of \mathcal{G} or also as the *transversal hypergraph* of \mathcal{G} , defines itself as 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} over vertex set V , decide whether $\mathcal{H} = tr(\mathcal{G})$.

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 [2]: if \mathcal{G} and \mathcal{H} are simple hypergraphs over vertex set V , then $\mathcal{H} = tr(\mathcal{G})$ if and only if $\mathcal{G} = tr(\mathcal{H})$, and in this case \mathcal{G} and \mathcal{H} are said to be dual. 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 [3]. These and several other properties of the duality problem are reviewed and discussed in [15, 11, 31, 13], where many original references can also be found.

Applications of hypergraph duality. The DUAL problem and its computational variant have a tremendous number of applications. They range from data mining [30, 45, 4, 5], functional dependency inference [43, 44, 26], and machine learning, in particular, learning monotone Boolean conjunctive normal forms (CNFs) and disjunctive normal forms (DNFs) with membership queries [30, 46], to model-based diagnosis [47, 29], computing a Horn approximation to a non-Horn theory [35, 23], computing minimal abductive explanations to observations [14], and computational biology, for example, discovering of metabolic networks and engineering of drugs preventing the production of toxic metabolites in cells [40, 39]. Surveys of these and other applications as well as further references can be found in [12, 11, 31, 41].

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 $f(x_1, x_2, \dots, x_n)$ and $g(x_1, x_2, \dots, x_n)$ on propositional variables x_1, x_2, \dots, x_n are dual if, for any Boolean assignment to variables x_1, \dots, x_n ,

$$f(x_1, x_2, \dots, x_n) \equiv \neg g(\neg x_1, \neg x_2, \dots, \neg x_n).$$

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

¹Note that sometimes in the literature the dual hypergraph of \mathcal{G} was defined as the hypergraph derived from \mathcal{G} in which the roles of the vertices and the edges are “interchanged” (see, e.g., [2, 48]), and this is different from the transversal hypergraph. Nevertheless, lately in the literature the name “dual hypergraph” has been used with the meaning of “transversal hypergraph” (as in, e.g., [15, 37, 6, 25, 38, 16]).

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.² Two hypergraphs \mathcal{G} and \mathcal{H} are dual if and only if their associated DNFs \mathcal{G}^* and \mathcal{H}^* are dual, where the DNF \mathcal{F}^* associated with a hypergraph $\mathcal{F} = \langle V, E \rangle$ 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

$$\begin{aligned}\mathcal{G}^* &= (a \wedge c \wedge d) \vee (a \wedge e \wedge f) \vee (c \wedge b) \vee (e \wedge b), \text{ and} \\ \mathcal{H}^* &= (a \wedge b) \vee (c \wedge e) \vee (c \wedge b \wedge f) \vee (e \wedge b \wedge d) \vee (d \wedge b \wedge f),\end{aligned}$$

which are indeed mutually dual. The duality problem for irredundant monotone DNFs corresponds, in turn, to DUAL, and the problem instances $\langle \mathcal{F}, \mathcal{G} \rangle$ and $\langle \mathcal{F}^*, \mathcal{G}^* \rangle$ can be intertranslated 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, Fredman and Khachiyan [19] have shown that DUAL is in $\text{DTIME}[n^{o(\log n)}]$, more precisely, that it is contained in $\text{DTIME}[n^{4\chi(n)+O(1)}]$, 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 $\text{GC}(f(n), \mathcal{C})$ denote the complexity class of all problems that after a non-deterministic guess of $O(f(n))$ bits can be decided (checked) in complexity class \mathcal{C} . Eiter, Gottlob, and Makino [13] and, independently, Kavvadias and Stavropoulos [36] have shown that $\overline{\text{DUAL}}$ is in $\text{GC}(\log^2 n, \text{PTIME})$; note that this class is also known as $\beta_2\text{P}$; see [24].

Recently, the nondeterministic bound for $\overline{\text{DUAL}}$ was further pushed down to $\text{GC}(\log^2 n, \llbracket \text{LOGSPACE}_{\text{pol}} \rrbracket^{\log})$ [25]; see Figure 2(a).

A precise definition of $\llbracket \text{LOGSPACE}_{\text{pol}} \rrbracket^{\log}$ is given in [25]. We will not make use of this class in the technical part of the present paper. Informally, $\llbracket \text{LOGSPACE}_{\text{pol}} \rrbracket^{\log}$ contains those problems π for which there exist 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 $\text{GC}(\log^2 n, \llbracket \text{LOGSPACE}_{\text{pol}} \rrbracket^{\log})$ to other classes; see Figure 2(a). In [25], it was shown that $\text{GC}(\log^2 n, \llbracket \text{LOGSPACE}_{\text{pol}} \rrbracket^{\log})$ is not only a subclass of $\text{GC}(\log^2 n, \text{PTIME})$, but also of $\text{DSPACE}[\log^2 n]$, i.e., of quadratic logspace. Therefore, as also proven in a new and more direct way in the present paper, DUAL is in $\text{DSPACE}[\log^2 n]$ (Corollary 4.9), and it is thus most unlikely for DUAL to be PTIME-hard, which answered a previously long-standing

²In fact, in the literature, the hypergraph transversal problem was tackled interchangeably from the perspective of monotone Boolean formula dualization or from the perspective of hypergraphs. Readers wanting to know more about the relationships between some of the different perspectives adopted in the literature to deal with the DUAL problem are referred to the extended technical report [28] available online.

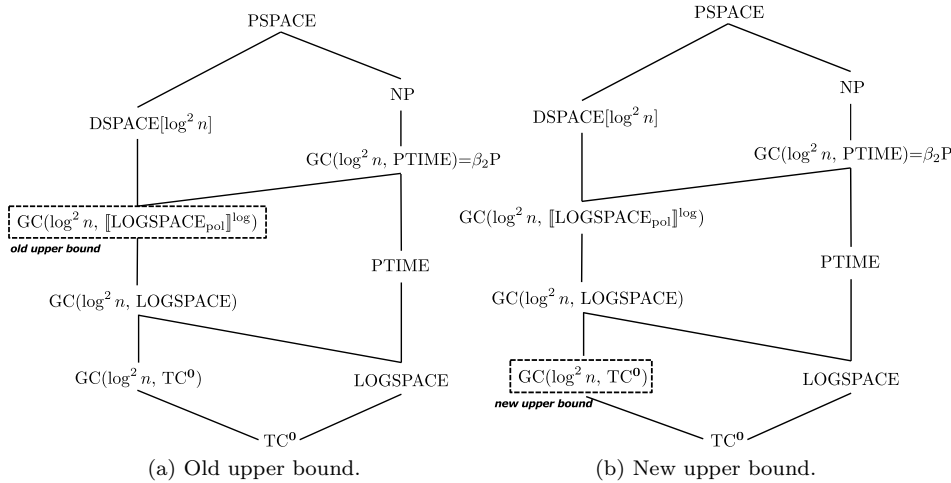


FIG. 2. Complexity bound improvement obtained in this paper.

question. Given that PTIME and $\text{DSPACE}[\log^2 n]$ 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 [32, 34], as these latter problems are PTIME-hard, but in $\text{NP} \cap \text{coNP}$.

Main complexity problem tackled. In [25] it was asked whether the upper bound of $\text{GC}(\log^2 n, [\text{LOGSPACE}_{\text{pol}}]^{\log})$ could be pushed further downwards, and the following conjecture was made.

Conjecture (see [25]). $\overline{\text{DUAL}} \in \text{GC}(\log^2 n, \text{LOGSPACE})$.

It was unclear, however, how to prove this conjecture based on the algorithms and methods used in [25]. There, a problem decomposition strategy by Boros and Makino [6] 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 subinstance 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-subinstance 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 subinstance associated with node v in LOGSPACE. In fact, it seems that the only way to compute the subinstance 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 them the strategy of Gaur [21], which also influenced the method of Boros and Makino [6]. 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 [21], 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 subinstances of the decomposition (as used by Boros and Makino [6]) 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 [21]). Our new approach 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 subtasks 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 [33]. By putting all pieces together, we succeed in describing 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 TC^0 .

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

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

By the well-known inclusion $\text{TC}^0 \subseteq \text{LOGSPACE}$, we immediately obtain a corollary that proves the above-mentioned conjecture.

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

Moreover, by the inclusion $\text{GC}(\log^2 n, \text{LOGSPACE}) \subseteq \text{DSPACE}[\log^2 n]$, and the fact that $\text{DSPACE}[\log^2 n]$ is closed under complement, we can easily obtain as a simple corollary the following.

COROLLARY B. $\text{DUAL} \in \text{DSPACE}[\log^2 n]$.

To conclude, by an easy adaptation of our algorithm ND-NOTDUAL we devise a simple deterministic algorithm COMPUTENT to compute a new (not necessarily minimal) transversal in quadratic 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 2(a)) shows the previous state of knowledge about the complexity, while the right part (Figure 2(b)) 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 $\text{GC}(\log^2 n, \text{TC}^0)$ and, in particular, *logically defined subclasses* that replace TC^0 by low-level prefix classes of $\text{FO}(\text{COUNT})$. Classes of this type can be found in [7, 8, 18, 49, 50]. More details on this will be given in the conclusive remarks in section 5. Second, the membership of $\overline{\text{DUAL}}$ in $\text{GC}(\log^2 n, \text{TC}^0)$ 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 $\text{GC}(\log^2 n, \text{TC}^0)$, and given that it is not generally believed that $\text{GC}(\log^2 n, \text{TC}^0)$ 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 a 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, in section 4 we present the nondeterministic algorithm ND-NOTDUAL for $\overline{\text{DUAL}}$, prove it correct, and then analyze this algorithm to derive our main complexity results. To conclude, by exploiting the method used in the nondeterministic algorithm, we present a deterministic algorithm to actually compute a new (not necessarily minimal) transversal in quadratic logspace.

2. Preliminaries. In what follows, when we identify a hypergraph \mathcal{G} with its edge-set E by writing $\mathcal{G} = E$, we mean $\mathcal{G} = \langle \bigcup_{G \in E} G, E \rangle$. By writing $G \in \mathcal{G}$ we mean $G \in E$. Generally, we denote by V the set of vertices of a hypergraph and, if not stated otherwise, pairs of hypergraphs \mathcal{G} and \mathcal{H} , and the hypergraphs of a DUAL instance $\langle \mathcal{G}, \mathcal{H} \rangle$, are assumed to have the same set of vertices. The number of edges of \mathcal{G} is denoted by $|\mathcal{G}|$, and, given an instance $\langle \mathcal{G}, \mathcal{H} \rangle$ of DUAL , m is the total number $|\mathcal{G}| + |\mathcal{H}|$ of edges of \mathcal{G} and \mathcal{H} . By $\|\mathcal{G}\|$ we denote the size of the hypergraph \mathcal{G} , that is, the space (in terms of the number of bits) required to represent \mathcal{G} . It is reasonable to assume that a hypergraph \mathcal{G} is represented through the adjacency lists of its edges (i.e., each edge G of \mathcal{G} is represented through the list of the vertices belonging to G). It is easy to see that $|V| \leq \|\mathcal{G}\|$ and $|\mathcal{G}| \leq \|\mathcal{G}\|$. We denote by $N = \|\mathcal{G}\| + \|\mathcal{H}\|$ the size of the input of the DUAL problem.

We say that \mathcal{G} is an *empty hypergraph* if \mathcal{G} does not have any edge, and \mathcal{G} is an *empty-edge hypergraph* if \mathcal{G} contains only an empty edge (i.e., an edge without vertices in it). Note that empty and empty-edge hypergraphs can actually contain vertices, and thus there are various empty and empty-edge hypergraphs. For notational convenience, by $\mathcal{G} = \emptyset$ we indicate that \mathcal{G} is an empty hypergraph, while by $\mathcal{G} = \{\emptyset\}$ we indicate that \mathcal{G} is an empty-edge hypergraph. However, with these notations we do not mean that these hypergraphs do not have vertices. The notation $\emptyset \in \mathcal{G}$ clearly means that \mathcal{G} contains an empty edge. Observe that, if $\mathcal{G} = \emptyset$, then any set of vertices is a transversal of \mathcal{G} . For this reason, there is only one minimal transversal of \mathcal{G} and it is the empty set. On the other hand, if $\emptyset \in \mathcal{G}$, then there is no transversal of \mathcal{G} at all. Hence, by definition, the dual of an empty hypergraph is an empty-edge hypergraph, and vice versa [11]. Two hypergraphs \mathcal{G} and \mathcal{H} over the same vertex set are

trivially dual, if one of them is an empty hypergraph and the other is an empty-edge hypergraph.

Given a hypergraph \mathcal{G} and a set of vertices T , a vertex $v \in T$ is *critical* in T (w.r.t. \mathcal{G}) if there is an edge $G \in \mathcal{G}$ such that $G \cap T = \{v\}$. We say that G witnesses the criticality of v in T . Observe that, if v is a critical vertex in T , v may have various witnesses of its criticality, that is, more than one edge of \mathcal{G} can intersect T only on v .

A set of vertices S is an *independent set* of a hypergraph \mathcal{G} if, for all $G \in \mathcal{G}$, $G \not\subseteq S$. If \mathcal{G} and \mathcal{H} are two hypergraphs, a set of vertices T is a *new transversal* of \mathcal{G} w.r.t.³ \mathcal{H} if T is a transversal of \mathcal{G} , and T is also an independent set of \mathcal{H} . Intuitively, a new transversal T of \mathcal{G} is a transversal of \mathcal{G} missing in \mathcal{H} . More formally, T is a new transversal of \mathcal{G} w.r.t. \mathcal{H} if, for any transversal $T' \subseteq T$ of \mathcal{G} , $T' \notin \mathcal{H}$. Note that a new transversal is not necessarily a minimal transversal, however, it contains a new minimal transversal.

In the following, we state the main properties about transversals to be used in this paper. These properties are already known or easily follow from what is known in the literature (see, e.g., [2, 19, 22, 6, 16, 11, 20]).⁴

LEMMA 2.1. *Let \mathcal{G} be a hypergraph, and let $T \subseteq V$ be a transversal of \mathcal{G} . Then, T is a minimal transversal of \mathcal{G} if and only if every vertex $v \in T$ is critical (and hence there is an edge $G_v \in \mathcal{G}$ witnessing so).*

For a set of vertices $T \subseteq V$, let \overline{T} denote $V \setminus T$, i.e., the *complement* of T in V .

LEMMA 2.2. *Let \mathcal{G} and \mathcal{H} be two hypergraphs. A set of vertices $T \subseteq V$ is a new transversal of \mathcal{G} w.r.t. \mathcal{H} if and only if \overline{T} is a new transversal of \mathcal{H} w.r.t. \mathcal{G} .*

We say that hypergraphs \mathcal{G} and \mathcal{H} satisfy the *intersection property* if all edges of \mathcal{G} are transversals of \mathcal{H} and, thus, vice versa, all edges of \mathcal{H} are transversal of \mathcal{G} . Note here that, for the intersection property to hold, the edges of one hypergraph are *not* required to be minimal transversal of the other.

LEMMA 2.3. *Let \mathcal{G} and \mathcal{H} be two hypergraphs. Then, \mathcal{G} and \mathcal{H} are dual if and only if \mathcal{G} and \mathcal{H} are simple, satisfy the intersection property, and there is no new transversal of \mathcal{G} w.r.t. \mathcal{H} .*

3. Decomposing the DUAL problem.

3.1. Decomposition principles. A way 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 also an independent set of \mathcal{H} . In fact, many algorithms in the literature follow this approach (see, e.g., [19, 13, 36, 16, 6, 21]). These algorithms try to build such a new transversal by successively including vertices in and excluding vertices from a candidate for a new transversal. To give an example, the classical algorithm “A” of Fredman and Khachiyan [19] tries to include a vertex v in a candidate for a new transversal, and if this does not result in a new transversal, then v is excluded. Moreover if the exclusion of v does not lead to a new transversal, then no new transversal exists which is coherent with the choices having been made before considering vertex v . (If v is the first vertex considered, then there is no new transversal at all.)

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 and the set of the vertices considered excluded

³We will often omit “w.r.t. \mathcal{H} ” when the hypergraph \mathcal{H} we are referring to is understood.

⁴All the missing proofs of this and other sections and appendices of this paper can be found in the extended technical report [28] available online.

from the attempted new transversal. Similarly, those algorithms working on Boolean formulas keep track of the truth assignment: the variables to which **true** has been assigned, those to which **false** has been assigned, and (obviously) those to which no Boolean value has been assigned yet.

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 $In \cap Ex = \emptyset$. Intuitively, set In contains the vertices considered included in (or, inside) an attempted new transversal $T \supseteq In$ of \mathcal{G} w.r.t. \mathcal{H} , while set Ex contains the vertices considered excluded from (or, outside) T (i.e., $T \cap Ex = \emptyset$). We say that a vertex $v \in V$ is free in (or of) an assignment $\sigma = \langle In, Ex \rangle$, if $v \notin In$ and $v \notin Ex$. Note that the *empty assignment* $\sigma_\varepsilon = \langle \emptyset, \emptyset \rangle$ is a valid assignment. Given assignments $\sigma_1 = \langle In_1, Ex_1 \rangle$ and $\sigma_2 = \langle In_2, Ex_2 \rangle$, if $In_1 \cap Ex_2 = \emptyset$ and $Ex_1 \cap In_2 = \emptyset$, 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 $\sigma_2 = \langle In_2, Ex_2 \rangle$ is said to be an *extension* of assignment $\sigma_1 = \langle In_1, Ex_1 \rangle$, denoted by $\sigma_1 \sqsubseteq \sigma_2$, whenever $In_1 \subseteq In_2$ and $Ex_1 \subseteq Ex_2$. If $\sigma_1 \sqsubseteq \sigma_2$ and $In_1 \subset In_2$ or $Ex_1 \subset Ex_2$ we say that σ_2 is a *proper extension* of σ_1 , denoted by $\sigma_1 \sqsubset \sigma_2$.

Given a set of vertices $S \subseteq V$, the associated assignment is $\sigma_S = \langle S, \bar{S} \rangle$. We say that an assignment $\sigma = \langle In, Ex \rangle$ is *coherent* with a set of vertices S , and vice versa, whenever $\sigma \sqsubseteq \sigma_S$. This is tantamount to $In \subseteq S$ and $Ex \subseteq \bar{S}$ (or, equivalently, $Ex \cap S = \emptyset$). With a slight abuse of notation we denote that an assignment σ is coherent with a set S by $\sigma \sqsubseteq S$. Observe that, by Lemma 2.2, 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* $\bar{\sigma} = \langle Ex, In \rangle$ is coherent with the new transversal \bar{T} of \mathcal{H} w.r.t. \mathcal{G} . Intuitively, this means that for an assignment $\sigma = \langle In, Ex \rangle$, set In is (a subset of) an attempted new transversal of \mathcal{G} w.r.t. \mathcal{H} , and, symmetrically, set Ex is (a subset of) an attempted new transversal of \mathcal{H} w.r.t. \mathcal{G} .

Most algorithms proposed in the literature essentially try different assignments by successively extending in different ways the current assignment. Each extension performed induces a “reduced” instance of DUAL on which the algorithm is recursively invoked. Intuitively, the size of the instance decreases for two reasons. Including vertices in the new attempted 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 attempted 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.

Given a hypergraph \mathcal{G} and a set S of vertices, as in [16, 6], we define hypergraphs $\mathcal{G}_S = \langle S, \{G \in \mathcal{G} \mid G \subseteq S\} \rangle$ and $\mathcal{G}^S = \langle S, \min(\{G \cap S \mid G \in \mathcal{G}\}) \rangle$, 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 . If $\emptyset \in \mathcal{G}$, then $\min(\mathcal{G}) = \{\emptyset\}$.

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{H} \rangle$ be an instance of DUAL. While constructing a new transversal of \mathcal{G} , when the assignment $\sigma = \langle In, Ex \rangle$ is considered let us denote by $\mathcal{I}_\sigma = \langle \mathcal{G}(\sigma), \mathcal{H}(\sigma) \rangle = \langle (\mathcal{G}_{V \setminus In})^{V \setminus (In \cup Ex)}, (\mathcal{H}_{V \setminus Ex})^{V \setminus (In \cup Ex)} \rangle$ the reduced instance derived from \mathcal{I} and induced by σ . Observe that both $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are simple by definition, because they undergo a minimization operation, and that, if \mathcal{G} and \mathcal{H} have the same vertex set, then $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ have the same vertex set. Intuitively, since we are interested in finding new transversals of \mathcal{G} w.r.t. \mathcal{H} , we can avoid analyzing and further extending an assignment $\sigma = \langle In, Ex \rangle$ for which In is not an independent set of \mathcal{H} or Ex is not an independent set of \mathcal{G} . In fact, on the one hand, if Ex is not an independent set of \mathcal{G} , then no set of vertices T coherent with σ can be a transversal

of \mathcal{G} (because, by $\sigma \sqsubseteq T$, T and Ex are disjoint). On the other hand, if In is not an independent set of \mathcal{H} , then no set of vertices T coherent with σ can be an independent set of \mathcal{H} and, hence, a new transversal of \mathcal{G} (because, by $\sigma \sqsubseteq T$, $In \subseteq T$). To this purpose, for an assignment $\sigma = \langle In, Ex \rangle$, if there is an edge $H \in \mathcal{H}$ with $H \subseteq In$ or an edge $G \in \mathcal{G}$ with $G \subseteq Ex$, we say that In and Ex are *covering*, respectively. We also say that $\sigma = \langle In, Ex \rangle$ is a covering assignment if In or Ex are covering. For future reference, let us highlight the just mentioned property in the following lemma.

LEMMA 3.1. *Let \mathcal{G} and \mathcal{H} be two hypergraphs, and let $\sigma = \langle In, Ex \rangle$ be an assignment.*

- (a) *If Ex is covering, then there is no set of vertices coherent with σ that is a transversal of \mathcal{G} .*
- (b) *If In is covering, then there is no set of vertices coherent with σ that is an independent set of \mathcal{H} .*

Hence, if σ is covering, then there is no set of vertices coherent with σ that is a new transversal of \mathcal{G} w.r.t. \mathcal{H} .

Decomposing an original instance of DUAL into multiple subinstances has the advantage of generating smaller subinstances for which it is computationally easier to check their duality. In fact, many algorithms proposed in the literature decompose the original instance into smaller subinstances for which the duality test is feasible in PTIME or even in subclasses of it. The following property of DUAL subinstances is of key importance for the correctness of all the approaches tackling DUAL through decomposition techniques.

LEMMA 3.2. *Two hypergraphs \mathcal{G} and \mathcal{H} are dual if and only if \mathcal{G} and \mathcal{H} are simple, satisfy the intersection property, and, for all assignments σ , $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are dual (or, equivalently, there is no new transversal of $\mathcal{G}(\sigma)$ w.r.t. $\mathcal{H}(\sigma)$).*

Lemma 3.2 can be easily proven by exploiting Lemma 2.3 and the equivalence between the hypergraph transversal problem and the duality problem of nonredundant monotone Boolean CNF/DNF formulas, where (partial) assignments here considered correspond to partial Boolean truth assignments. However, in Appendix A, we provide a detailed proof of the previous lemma without resorting to the equivalence with Boolean formulas and we also analyze other interesting properties of decompositions.

Although checking the duality of subinstances of an initial instance of DUAL can be computationally easier, it is evident that, in order to find a new transversal of \mathcal{G} , naively trying all the possible noncovering assignments would require exponential time. Nevertheless, as already mentioned, there are deterministic algorithms solving DUAL in quasi-polynomial time. To meet such a time bound, those algorithms do not try all the possible combinations of assignments, but they consider specific assignment extensions.

Common approaches—here referred to as extension types—to extend a currently considered assignment $\sigma = \langle In, Ex \rangle$ to an assignment σ' are

- (i) include a free vertex v into the new attempted transversal of \mathcal{G} w.r.t. \mathcal{H} , i.e., $\sigma' = \sigma + \langle \{v\}, \emptyset \rangle$;
- (ii) include a free vertex v as a critical vertex into the new attempted transversal of \mathcal{G} w.r.t. \mathcal{H} with an edge $G \in \mathcal{G}(\sigma)$, such that $v \in G$, witnessing v 's criticality, i.e., $\sigma' = \sigma + \langle \{v\}, G \setminus \{v\} \rangle$;
- (iii) exclude a free vertex v from the new attempted transversal of \mathcal{G} w.r.t. \mathcal{H} (or, equivalently, include v into the new attempted transversal of \mathcal{H} w.r.t. \mathcal{G}), i.e., $\sigma' = \sigma + \langle \emptyset, \{v\} \rangle$;

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

Observe that the extension types listed above always generate consistent assignments, i.e., assignments for which the sets of included and excluded vertices do not overlap. This is easy to see for extension types (i) and (iii). For extension type (ii), observe the following. Let $\sigma = \langle In, Ex \rangle$ be an assignment. Since $G \in \mathcal{G}(\sigma)$, $G \subseteq V \setminus (In \cup Ex)$ by definition of $\mathcal{G}(\sigma)$. Therefore, $G \cap In = \emptyset$ and $G \cap Ex = \emptyset$. Moreover, the sets of vertices $\{v\}$ and $(G \setminus \{v\})$ clearly constitute a partition of vertices in G , thus $\langle In \cup \{v\}, Ex \cup (G \setminus \{v\}) \rangle$ is a consistent assignment. Similarly, it can be shown that extension type (iv) generates a consistent assignment.

The size reduction attained in the considered subinstances tightly depends on the assignment extension performed and, in particular, on the frequencies with which vertices belong to the edges of $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$. We will analyze this in detail below.

3.2. Assignment trees and the definition of $\mathcal{T}(\mathcal{G}, \mathcal{H})$. Most algorithms proposed in the literature adopt their own specific assignment extensions in specific sequences. The assignments successively considered during the recursive execution of the algorithms can be analyzed through a treelike structure. Intuitively, each node of the tree can be associated with a tried assignment, and nodes of the tree are connected when their assignments are one the direct extension of the other. We can call these trees *assignment trees*.

Inspired by the algorithm proposed by Gaur [21],⁵ we will now describe the construction of a general assignment 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 (assignment extensions directly derived from) extension types (ii) and (iii). This tree is of super-polynomial size. However, it will be shown later that whenever \mathcal{G} and \mathcal{H} are not dual, then there must be in this tree a node at depth $O(\log N)$ which can be recognized with low computational effort as a witness of the nonduality of hypergraphs \mathcal{G} and \mathcal{H} .

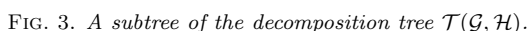
Intuitively, each node p of the tree $\mathcal{T}(\mathcal{G}, \mathcal{H})$ is associated with an assignment σ_p . In particular, the root is labeled with the empty assignment. Node p of the tree has a child q for each assignment σ_q that can be obtained from σ_p through an elementary extension of type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

For our purposes, drawing upon the algorithm of Gaur, for each node p of the tree whose assignment is $\sigma_p = \langle In_p, Ex_p \rangle$ we do not (explicitly) consider subinstance $\mathcal{I}_p = \langle \mathcal{G}(\sigma_p), \mathcal{H}(\sigma_p) \rangle$. We refer instead to the following sets:

- $Sep_{\mathcal{G}, \mathcal{H}}(\sigma_p) = \{G \in \mathcal{G} \mid G \cap In_p = \emptyset\}$, the set of all edges of \mathcal{G} *not met* by (or, equivalently, *separated* from) σ_p ; and
- $Com_{\mathcal{G}, \mathcal{H}}(\sigma_p) = \{H \in \mathcal{H} \mid H \cap Ex_p = \emptyset\}$, the set of all edges of \mathcal{H} *compatible* with σ_p .

We will often omit the subscript “ \mathcal{G}, \mathcal{H} ” of $Sep_{\mathcal{G}, \mathcal{H}}(\sigma_p)$ and $Com_{\mathcal{G}, \mathcal{H}}(\sigma_p)$ when hypergraphs \mathcal{G} and \mathcal{H} we are referring to are understood. The sets $Sep(\sigma_p)$ and $Com(\sigma_p)$ are very similar to $\mathcal{G}(\sigma_p)$ and $\mathcal{H}(\sigma_p)$, respectively. However, roughly speaking, unlike $\mathcal{G}(\sigma_p)$ and $\mathcal{H}(\sigma_p)$, the edges in $Sep(\sigma_p)$ and $Com(\sigma_p)$ are neither “projected” over the free vertices of σ_p , nor minimized to obtain simple hypergraphs. In fact, $Sep(\sigma_p)$ and

⁵Readers can find in Appendix B a *deterministic* algorithm, based on that of Gaur [21] (see also [22]), deciding hypergraph duality. Note that the original algorithm proposed by Gaur aims instead at deciding *self*-duality of DNF Boolean formulas. It is from the algorithm reported in the appendix that we have taken ideas to devise our nondeterministic algorithm. Note that the exposition in Appendix B builds up on concepts, definitions, and lemmas discussed in subsections 3.2 and 3.3.



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 $\sigma_\varepsilon = \langle \emptyset, \emptyset \rangle$. Each 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 nodes p whose assignment σ_p is covering or has no free vertex (remember that, by Lemma 3.1, there is no benefit in considering (and further extending) covering assignments). Each nonleaf node p of $\mathcal{T}(\mathcal{G}, \mathcal{H})$ has precisely the following children:

- Observe that, by this definition 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 (nonsibling) 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 sets depicted with the root node are $Sep(\sigma_\varepsilon)$ and $Com(\sigma_\varepsilon)$. Each of the other nodes p represents an assignment σ_p whose included vertices are indicated by a checkmark (✓) and whose excluded vertices by a cross (✗). In addition, each node p shows, on the left-hand side, the separated edges of \mathcal{G} and, on the right-hand side, the compatible edges of \mathcal{H} in σ_p , respectively. The leftmost 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 rightmost 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 nodes of the tree are depicted. Indeed, observe that the bottom right node of the figure is not a leaf, because its assignment is noncovering 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 covered by the excluded vertices).

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

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

Since the edges leaving a node are assumed to be all differently labeled, a path identifies unequivocally a node in the tree. Given a path Π , we denote by $\mathcal{N}(\Pi)$ the end-node of Π . Note that the end-node of a path is not necessarily a leaf of the decomposition tree.

The next lemma, which shows how to compute the assignment $\sigma_{\mathcal{N}(\Pi)}$ of node $\mathcal{N}(\Pi)$, immediately follows from the definition of the concept of path. For notational convenience we define $\sigma(\Pi) = \sigma_{\mathcal{N}(\Pi)}$.

LEMMA 3.3.

$$(1) \quad \sigma_{\mathcal{N}(\Pi)} = \sigma(\Pi) = \left\langle \bigcup_{(v, G) \in \Pi} \{v\}, \left(\bigcup_{-v \in \Pi} \{v\} \right) \cup \left(\bigcup_{(v, G) \in \Pi} (G \setminus \{v\}) \right) \right\rangle.$$

Let us now analyze what are the reductions in size achieved when specific extension types are performed. We denote by

$$\varepsilon_v^{Sep(\sigma)} = \frac{|\{G \in Sep(\sigma) \mid v \in G\}|}{|Sep(\sigma)|} \quad \text{and} \quad \varepsilon_v^{Com(\sigma)} = \frac{|\{H \in Com(\sigma) \mid v \in H\}|}{|Com(\sigma)|}$$

the ratio of the edges in $Sep(\sigma)$ and $Com(\sigma)$, respectively, containing vertex v .

LEMMA 3.4. *Let \mathcal{G} and \mathcal{H} be two hypergraphs satisfying the intersection property, and let σ be a noncovering assignment. If v is a free vertex of σ , then*

$$\begin{aligned} |Sep(\sigma + \langle \{v\}, \emptyset \rangle)| &= (1 - \varepsilon_v^{Sep(\sigma)}) \cdot |Sep(\sigma)| \quad \text{and} \\ |Com(\sigma + \langle \emptyset, \{v\} \rangle)| &= (1 - \varepsilon_v^{Com(\sigma)}) \cdot |Com(\sigma)|. \end{aligned}$$

On the other hand, if $G \in Sep(\sigma)$, $H \in Com(\sigma)$, and $v \in G$ and $w \in H$ are free

vertices of σ , then

$$\begin{aligned} |Com(\sigma + \langle \{v\}, G \setminus \{v\} \rangle)| &\leq \varepsilon_v^{Com(\sigma)} \cdot |Com(\sigma)| \text{ and} \\ |Sep(\sigma + \langle H \setminus \{w\}, \{w\} \rangle)| &\leq \varepsilon_w^{Sep(\sigma)} \cdot |Sep(\sigma)|. \end{aligned}$$

Proof. Given an assignment σ , if v is a free vertex in σ and v belongs to $\varepsilon_v^{Sep(\sigma)}$. $|Sep(\sigma)|$ many edges of $Sep(\sigma)$, then, for the assignment $\sigma' = \sigma + \langle \{v\}, \emptyset \rangle$ (extension type (i)), it is easy to see that $|Sep(\sigma')| = (1 - \varepsilon_v^{Sep(\sigma)}) \cdot |Sep(\sigma)|$. Similarly, when the assignment $\sigma' = \sigma + \langle \emptyset, \{v\} \rangle$ is considered (extension type (iii)), it is easy to see that $|Com(\sigma')| = (1 - \varepsilon_v^{Com(\sigma)}) \cdot |Com(\sigma)|$.

Let us now consider extension type (ii), and let $\sigma' = \sigma + \langle \{v\}, G \setminus \{v\} \rangle$. We will show that $|Com(\sigma')| \leq \varepsilon_v^{Com(\sigma)} \cdot |Com(\sigma)|$. Let $K_{\{v\}} = \{\tilde{H} \in Com(\sigma) \mid \tilde{H} \cap G = \{v\}\}$ be the set of edges in $Com(\sigma)$ whose intersection with G is exactly vertex v , and let $K_{[v]} = \{\tilde{H} \in Com(\sigma) \mid \tilde{H} \cap G \ni v\}$ be the set of edges in $Com(\sigma)$ whose intersection with G contains vertex v . Clearly $K_{\{v\}} \subseteq K_{[v]}$ and, hence, $|K_{\{v\}}| \leq |K_{[v]}|$. By definition, $|K_{[v]}| = \varepsilon_v^{Com(\sigma)} \cdot |Com(\sigma)|$, therefore $|K_{\{v\}}| \leq \varepsilon_v^{Com(\sigma)} \cdot |Com(\sigma)|$. Since \mathcal{G} and \mathcal{H} satisfy the intersection property, $Sep(\sigma)$ contains edges of \mathcal{G} , and $Com(\sigma)$ contains edges of \mathcal{H} , all edges of $Com(\sigma)$ intersect G . This, together with the fact that all vertices $G \setminus \{v\}$ are excluded in σ' , implies that $Com(\sigma') = K_{\{v\}}$. Therefore, $|Com(\sigma')| \leq \varepsilon_v^{Com(\sigma)} \cdot |Com(\sigma)|$. For extension type (iv) the proof is similar. \square

Observe that in the proof of Lemma 3.4 we do not require $Sep(\sigma)$ and $Com(\sigma)$ to be “simple,” i.e., it is not required that edges belonging to $Sep(\sigma)$ (resp., $Com(\sigma)$) are not subsets of other edges in $Sep(\sigma)$ (resp., $Com(\sigma)$). In fact, the lemma is valid regardless of that. Lemma 3.4 states a general property about the reduction in size of $Sep(\sigma)$ and $Com(\sigma)$ when some assignment extensions are considered. Indeed, edges G from $Sep(\sigma)$ do not need to be considered any longer as soon as they contain an included vertex, and this is irrespective of $Sep(\sigma)$ being actually simple. A similar discussion extends to edges H of $Com(\sigma)$ containing excluded vertices.

Before proceeding with our discussion, we recall that each node p of the tree corresponds to the subinstance $\mathcal{I}_p = \langle \mathcal{G}(\sigma_p), \mathcal{H}(\sigma_p) \rangle$ even though we do not explicitly represent it, and we refer instead to $Sep(\sigma_p)$ and $Com(\sigma_p)$. Therefore, $\mathcal{T}(\mathcal{G}, \mathcal{H})$ is indeed a decomposition of the original instance into smaller subinstances.

We claim that, by construction of tree $\mathcal{T}(\mathcal{G}, \mathcal{H})$, if \mathcal{G} and \mathcal{H} are simple hypergraphs satisfying the intersection property, then the pair $\langle \mathcal{G}, \mathcal{H} \rangle$ is a yes-instance of DUAL if and only if, for each node p of $\mathcal{T}(\mathcal{G}, \mathcal{H})$, \mathcal{I}_p is a yes-instance of DUAL. Indeed, if \mathcal{G} and \mathcal{H} are dual, then, by Lemma 3.2, there is no assignment σ for which $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are not dual and, hence, all nodes p of $\mathcal{T}(\mathcal{G}, \mathcal{H})$ are such that \mathcal{I}_p is a yes-instance of DUAL. On the other hand, if \mathcal{G} and \mathcal{H} are not dual, since we are assuming that they are simple and satisfy the intersection property, by Lemma 2.3 there is a new transversal T of \mathcal{G} w.r.t. \mathcal{H} . The fact that \mathcal{G} and \mathcal{H} are simple implies also that $\mathcal{G} = \mathcal{G}(\sigma_\varepsilon) = \mathcal{G}(\sigma_r)$ and $\mathcal{H} = \mathcal{H}(\sigma_\varepsilon) = \mathcal{H}(\sigma_r)$, where r is the root of $\mathcal{T}(\mathcal{G}, \mathcal{H})$. Therefore, already the root r of $\mathcal{T}(\mathcal{G}, \mathcal{H})$ is such that $\mathcal{G}(\sigma_r)$ and $\mathcal{H}(\sigma_r)$ are not dual and, hence, \mathcal{I}_r is a no-instance of DUAL.

The critical point here is that, in order to prove that hypergraphs \mathcal{G} and \mathcal{H} are actually not dual, it would be much better to identify in $\mathcal{T}(\mathcal{G}, \mathcal{H})$ those nodes p for which it is computationally easy to verify that $\mathcal{G}(\sigma_p)$ and $\mathcal{H}(\sigma_p)$ are not dual, and the root of the tree may not always fit the purpose (of an efficient check). Therefore, to show that $\langle \mathcal{G}, \mathcal{H} \rangle$ is a no-instance of DUAL, the ideal solution would be that of

finding/guessing a path from the root to a node p , where \mathcal{I}_p is easily recognizable as a no-instance, e.g., as in the case in which $\sigma_p = \langle In_p, Ex_p \rangle$ is such that the set In_p or Ex_p is a new transversal of \mathcal{G} or \mathcal{H} , respectively. The interesting fact here is that, by construction of $\mathcal{T}(\mathcal{G}, \mathcal{H})$, if \mathcal{G} and \mathcal{H} are simple nondual hypergraphs satisfying the intersection property, there is always a node p in $\mathcal{T}(\mathcal{G}, \mathcal{H})$ such that σ_p has the required property for an easy check. Indeed, let us assume that T is a new transversal of \mathcal{G} , and consider the assignment $\sigma = \langle \emptyset, \overline{T} \rangle$. Since σ is an assignment which only excludes vertices, there is a node p in $\mathcal{T}(\mathcal{G}, \mathcal{H})$ such that $\sigma_p = \sigma$ because we can build the assignment σ by successively using extensions of type (iii). It is clear that, in general, such a node may be at depth linear in the tree. However, we will show in the next section that if $\langle \mathcal{G}, \mathcal{H} \rangle$ is actually a no-instance of DUAL, then there must also be a node p at depth $O(\log N)$ such that checking that \mathcal{I}_p is a no-instance of DUAL is feasible within complexity class TC^0 .

3.3. Logarithmic refuters in $\mathcal{T}(\mathcal{G}, \mathcal{H})$. An assignment $\sigma = \langle In, Ex \rangle$ is said to be 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} (see Lemma 2.2). We remind the reader that, for a set of vertices T to be a new transversal of \mathcal{G} (resp., \mathcal{H}) w.r.t. \mathcal{H} (resp., \mathcal{G}), T has to be a transversal of \mathcal{G} (resp., \mathcal{H}) and an independent set of \mathcal{H} (resp., \mathcal{G}), i.e., T must not be a superset of any edge of \mathcal{H} (resp., \mathcal{G}). Similarly, σ is a *double 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} , and Ex is a new transversal of \mathcal{H} w.r.t. \mathcal{G} . Recall that a new transversal does not need to be minimal. Double witnesses are easily proven to be characterized as follows.

LEMMA 3.5. *Let \mathcal{G} and \mathcal{H} be two hypergraphs. An assignment σ is a double (non-duality) witness if and only if*

$$(2) \quad Sep(\sigma) = \emptyset \wedge Com(\sigma) = \emptyset.$$

Note here that a node p of $\mathcal{T}(\mathcal{G}, \mathcal{H})$ whose assignment σ_p is a witness is not necessarily a leaf of the tree. For example, in Figure 3 the assignment of the bottom right node is a witness, but this node, as already observed, is not a leaf of the full tree. From now on, we will often refer to properties of an assignment σ_p as properties of the node p . For example, we say that a node p of $\mathcal{T}(\mathcal{G}, \mathcal{H})$ is a witness when σ_p is actually a witness.

We have already seen that, if \mathcal{G} and \mathcal{H} are simple nondual hypergraphs satisfying the intersection property, then in $\mathcal{T}(\mathcal{G}, \mathcal{H})$ there is always a witness at linear depth. But this is not enough for our purposes. Our aim in the rest of this section is to prove a stronger property. In fact, we will show that there is always at only logarithmic depth a node that is either a witness or can be easily extended to be a witness.

Before proceeding we need the following property.

LEMMA 3.6. *Let \mathcal{G} and \mathcal{H} be two hypergraphs, and let $\sigma = \langle In, Ex \rangle$ be an assignment coherent with a new minimal transversal T of \mathcal{G} w.r.t. \mathcal{H} , such that $In \neq T$. Then, every vertex $v \in (T \setminus In)$ is free, and for each such vertex, there is an edge $G_v \in Sep(\sigma)$ with $v \in G_v$, such that $\sigma + \langle \{v\}, G_v \setminus \{v\} \rangle$ is coherent with T .*

Proof. Let v be any vertex belonging to $T \setminus In$. From $\sigma \sqsubseteq T$ it follows that $v \notin Ex$ and hence v is free. Since T is a minimal transversal of \mathcal{G} , by Lemma 2.1, v is critical (in T). For this reason, there is an edge $G_v \in \mathcal{G}$ such that $T \cap G_v = \{v\}$. Now,

simply observe that $G_v \in \text{Sep}(\sigma)$ as well (because $T \cap G_v = \{v\}$ and $v \notin \text{In}$), and that $\sigma + \langle \{v\}, G_v \setminus \{v\} \rangle \sqsubseteq T$. \square

Consider now the following situation. Let \mathcal{G} and \mathcal{H} be two hypergraphs satisfying the intersection property, for which T is a new minimal transversal of \mathcal{G} w.r.t. \mathcal{H} . Let us again use as a running example the hypergraphs in Figure 3: the minimal transversal of \mathcal{G} missing in \mathcal{H} is $T = \{d, b, f\}$. Assume that we are in the process of building a witness by extending intermediate assignments to new ones that are still coherent with T (essentially we extend assignments with the aim of “converging” to T , so to have the witness that we are looking for). From what we have said, intuitively, we can recognize the built assignment σ as a witness when sets $\text{Sep}(\sigma)$ and $\text{Com}(\sigma)$ become empty.

In general, if $\sigma = \langle \text{In}, \text{Ex} \rangle$ is a nonwitnessing assignment coherent with T , we could extend σ , for example, by excluding a vertex v that is free in σ and does not belong to T (i.e., an extension type (iii)) or, by including, as critical vertex, a vertex v that is free in σ and belongs to T (i.e., an extension type (ii)). Note that, by Lemma 3.6, if $\sigma = \langle \text{In}, \text{Ex} \rangle$ is an assignment coherent with a new minimal transversal T of \mathcal{G} , for *any* vertex $v \in (T \setminus \text{In})$, it is always possible to include v as critical vertex into σ (along with the suitable edge witnessing v ’s criticality).

For the extension type (iii), if v belongs to at least half of the edges in $\text{Com}(\sigma)$, then by excluding v we get rid of at least half of the edges in $\text{Com}(\sigma)$ (see Lemma 3.4). On the other hand, for the extension type (ii), if v belongs to less than half of the edges in $\text{Com}(\sigma)$, then by including v as critical vertex with the proper edge witnessing v ’s criticality we again get rid of at least half of the edges in $\text{Com}(\sigma)$.

In our example, at the root of the tree (i.e., for the empty assignment σ_ε), vertex c is such that $c \notin T$ and c belongs to two out of four edges of $\text{Com}(\sigma_\varepsilon)$. So, excluding c is a very good choice because the number of edges from $\text{Com}(\langle \emptyset, \emptyset \rangle)$ to $\text{Com}(\langle \emptyset, \{c\} \rangle)$ would be halved (from four to two). Symmetrically, vertex d is such that $d \in T$ and d belongs to only one edge of $\text{Com}(\sigma_\varepsilon)$. Hence, including d as critical vertex (along with edge 1 witnessing d ’s criticality) is again a very good choice because, also in this case, the number of edges from $\text{Com}(\langle \emptyset, \emptyset \rangle)$ to $\text{Com}(\langle \{d\}, \{a, c\} \rangle)$ would be (more than) halved (from four to one).

It is evident that halving the size of the set of compatible edges each time we perform an assignment extension would be one of the best ways to asymptotically speed up the construction of the witness sought for. This is the key intuition to prove that in $\mathcal{T}(\mathcal{G}, \mathcal{H})$ there are “duality refuters” at *logarithmic depth*. The following definitions serve the purpose of formalizing the just illustrated intuition.

Given an assignment σ , a free vertex v of σ is called *frequent* (in σ) if v belongs to at least half of the edges in $\text{Com}(\sigma)$, otherwise we say that v is *infrequent* (in σ). For example, we have already observed that in Figure 3 vertex c is frequent at the root and vertex d is infrequent at the root. Let us denote by $\text{Freq}_{\mathcal{G}, \mathcal{H}}(\sigma)$ and $\text{Infreq}_{\mathcal{G}, \mathcal{H}}(\sigma)$ the frequent and infrequent vertices in σ , respectively. Again, we will often omit the subscript “ \mathcal{G}, \mathcal{H} ” of $\text{Freq}_{\mathcal{G}, \mathcal{H}}(\sigma)$ and $\text{Infreq}_{\mathcal{G}, \mathcal{H}}(\sigma)$ when the hypergraphs \mathcal{G} and \mathcal{H} we are referring to are understood.

Let σ be an assignment coherent with a new minimal transversal T of \mathcal{G} such that $\text{Com}(\sigma) \neq \emptyset$. A free vertex v of σ is said to be *appealing to exclude* (for σ) w.r.t. T if $v \in \text{Freq}(\sigma)$ and $v \notin T$. On the other hand, a free vertex v of σ is said to be *appealing to include* (as a critical vertex) (for σ) w.r.t. T if $v \in \text{Infreq}(\sigma)$ and $v \in T$.

To summarize the first intuition, given a node p coherent with a new minimal transversal T , edges leaving p labeled with the exclusion of an appealing vertex to

exclude (w.r.t. T), or labeled with the inclusion as a critical vertex (with a suitable criticality's witness) of an appealing vertex to include (w.r.t. T), lead to a node q such that $\sigma_q \sqsubseteq T$ and $|Com(\sigma_q)| \leq \frac{1}{2}|Com(\sigma_p)|$ (see Lemma 3.4).

The intuition behind the fact that in $\mathcal{T}(\mathcal{G}, \mathcal{H})$ there are, at logarithmic depth, duality refuters which are moreover *easily recognizable* is the following. Assume \mathcal{G} and \mathcal{H} satisfy the intersection property and T is a new minimal transversal of \mathcal{G} w.r.t. \mathcal{H} . As presented above, we are extending assignments via appealing vertices so to quickly “converge” to T . If during this process we build a witness, then we are all done, because recognizing a witnessing assignment as such is computationally quite easy. However, it may well be the case that this process gets stuck if a point is reached where, on the one hand, there are no appealing vertices to extend the current assignment σ and, on the other hand, σ is not yet a witness. Remember that σ is by construction coherent with T . The key point is the following. If there are no appealing vertices to extend σ , then all the free vertices of σ that are frequent belong to T and all the free vertices of σ that are infrequent do not belong to T . Therefore, we can easily extend σ to obtain a witness by simply including all the free vertices of σ that are frequent and by excluding all the free vertices of σ that are infrequent. In the next section we show that this final assignment extension based on the frequencies of the vertices can be computed extremely efficiently.

We now formally prove that in $\mathcal{T}(\mathcal{G}, \mathcal{H})$ there are duality refuters at logarithmic depth. Given an assignment $\sigma = \langle In, Ex \rangle$, $\sigma^+ = \langle In \cup Freq(\sigma), Ex \cup Infreq(\sigma) \rangle$ is the *augmented assignment* of σ .

LEMMA 3.7. *Let \mathcal{G} and \mathcal{H} be two hypergraphs satisfying the intersection property. Then, there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} if and only if there is a node p in $\mathcal{T}(\mathcal{G}, \mathcal{H})$, at depth at most $\lfloor \log |\mathcal{H}| \rfloor + 1$, such that σ_p^+ is a double witness.*

Proof. (\Rightarrow) Let T be a new minimal transversal of \mathcal{G} , and let p be a generic node of $\mathcal{T}(\mathcal{G}, \mathcal{H})$ such that σ_p is coherent with T , $Com(\sigma_p) \neq \emptyset$, and σ_p can be extended via an appealing vertex. Let σ_q be the assignment obtained by extending σ_p via the appealing vertex. Since the structure of $\mathcal{T}(\mathcal{G}, \mathcal{H})$ keeps track of all the possible extensions of types (ii) and (iii), there is, among the children of p in $\mathcal{T}(\mathcal{G}, \mathcal{H})$, a node whose assignment is precisely σ_q .

Observe that the empty assignment σ_ε associated with the root of $\mathcal{T}(\mathcal{G}, \mathcal{H})$ is obviously coherent with T . Therefore, starting from the root of $\mathcal{T}(\mathcal{G}, \mathcal{H})$, we can build a sequence of nodes that can be successively extended via the inclusion (as a critical vertex) or the exclusion of an appealing vertex w.r.t. T , until we reach a node p such that $Com(\sigma_p) = \emptyset$ (and in that moment the frequencies of the vertices become meaningless, because frequencies are evaluated w.r.t. to $Com(\sigma_p)$) or there are no more appealing vertices w.r.t. T at all.

Let $s = (p_0, p_1, \dots, p_k)$ be a sequence of maximal length having the just mentioned property. By the definition of s , all nodes p_i are such that σ_{p_i} is coherent with T (see Lemma 3.6). Since s is of maximal length, $Com(\sigma_{p_k}) = \emptyset$ or there are no appealing vertices in σ_{p_k} w.r.t. T to further extend s . Observe that, by the definition of appealing vertex, for all $1 \leq i \leq k$, $|Com(\sigma_{p_i})| \leq \frac{1}{2}|Com(\sigma_{p_{i-1}})|$ (see Lemma 3.4). Therefore, s contains at most $\lfloor \log |\mathcal{H}| \rfloor + 2$ nodes and, hence, the length of the path from the root to p_k is at most $\lfloor \log |\mathcal{H}| \rfloor + 1$.

Let $\sigma_{p_k} = \langle In_{p_k}, Ex_{p_k} \rangle$. There are two cases: either (1) $Com(\sigma_{p_k}) = \emptyset$ or (2) $Com(\sigma_{p_k}) \neq \emptyset$.

Consider case (1). From $Com(\sigma_{p_k}) = \emptyset$ it follows that Ex_{p_k} is a transversal of \mathcal{H} . Since σ_{p_k} is coherent with T and T is a transversal of \mathcal{G} , $Ex_{p_k} \cap T = \emptyset$ and,

hence, Ex_{p_k} is an independent set of \mathcal{G} . Therefore, Ex_{p_k} is a new transversal of \mathcal{H} w.r.t. \mathcal{G} . Because $Com(\sigma_{p_k}) = \emptyset$, all the free vertices of σ_{p_k} are frequent, because each of them belongs to at least half of the edges of $Com(\sigma_{p_k})$. By this, $\sigma_{p_k}^+ = \langle In_{p_k} \cup Freq(\sigma_{p_k}), Ex_{p_k} \cup Infreq(\sigma_{p_k}) \rangle = \langle In_{p_k} \cup Freq(\sigma_{p_k}), Ex_{p_k} \rangle = \langle \overline{Ex_{p_k}}, Ex_{p_k} \rangle$, because all the free vertices are frequent. By Lemma 2.2, since Ex_{p_k} is a new transversal of \mathcal{H} w.r.t. \mathcal{G} , $\overline{Ex_{p_k}}$ is a new transversal of \mathcal{G} w.r.t. \mathcal{H} . Thus, $\sigma_{p_k}^+$ is a double witness.

Consider now case (2). Since p_k is the last node of the maximal length sequence s and $Com(\sigma_{p_k}) \neq \emptyset$, it must be the case that there are no appealing vertices in σ_{p_k} w.r.t. T . For the following discussion, note that $Com(\sigma_{p_k}) \neq \emptyset$ implies that the frequencies of the vertices are meaningful. There are two cases: either (a) $Sep(\sigma_{p_k}) \neq \emptyset$ or (b) $Sep(\sigma_{p_k}) = \emptyset$.

Consider Case (a). Since $Sep(\sigma_{p_k}) \neq \emptyset$, $Com(\sigma_{p_k}) \neq \emptyset$, and there are no appealing vertices in σ_{p_k} w.r.t. T , all the frequent vertices of σ_{p_k} belong to T and all the infrequent vertices of σ_{p_k} are outside T . By this,

$$\sigma_{p_k}^+ = \langle In_{p_k} \cup Freq(\sigma_{p_k}), Ex_{p_k} \cup Infreq(\sigma_{p_k}) \rangle = \langle T, \overline{T} \rangle.$$

By Lemma 2.2, since T is a new transversal of \mathcal{G} w.r.t. \mathcal{H} , \overline{T} is a new transversal of \mathcal{H} w.r.t. \mathcal{G} . Thus, $\sigma_{p_k}^+$ is a double witness.

Consider now case (b). Since σ_{p_k} is coherent with T , T is a minimal transversal of \mathcal{G} , and $Sep(\sigma_{p_k}) = \emptyset$, $In_{p_k} = T$. Because $In_{p_k} = T$, any free vertex v of σ_{p_k} is such that $v \in (\overline{T} \setminus Ex_{p_k})$ and, since there are no appealing vertices in σ_{p_k} w.r.t. T , v is infrequent (for otherwise v would have been an appealing vertex to exclude). By this, $\sigma_{p_k}^+ = \langle In_{p_k} \cup Freq(\sigma_{p_k}), Ex_{p_k} \cup Infreq(\sigma_{p_k}) \rangle = \langle In_{p_k}, Ex_{p_k} \cup Infreq(\sigma_{p_k}) \rangle = \langle In_{p_k}, \overline{In_{p_k}} \rangle$, because all the free vertices are infrequent. By Lemma 2.2, since $In_{p_k} = T$ is a new transversal of \mathcal{G} w.r.t. \mathcal{H} , $\overline{In_{p_k}}$ is a new transversal of \mathcal{H} w.r.t. \mathcal{G} . Thus, $\sigma_{p_k}^+$ is a double witness.

(\Leftarrow) Clearly, if there is, within the required depth, a node p such that σ_p^+ is a double witness, then there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} . \square

4. New upper bounds for the DUAL problem.

4.1. A new nondeterministic algorithm for DUAL. We present in this section our new nondeterministic algorithm ND-NOTDUAL for $\overline{\text{DUAL}}$ and prove its correctness. To prove the correctness of the algorithm we will exploit the property of $\mathcal{T}(\mathcal{G}, \mathcal{H})$ of having easily recognizable duality refuters at logarithmic depth (Lemma 3.7).

To disprove that two hypergraphs \mathcal{G} and \mathcal{H} are dual, we know that it is sufficient to show that at least one of them is not simple, or that the intersection property does not hold between them, or that there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} (see Lemma 2.3). After having ruled out the first two conditions, intuitively, our nondeterministic algorithm, to compute such a new transversal, guesses in the tree $\mathcal{T}(\mathcal{G}, \mathcal{H})$ a path of logarithmic length leading to a node p such that σ_p^+ is a double witness (see Lemma 3.7). More precisely, ND-NOTDUAL nondeterministically generates a set Σ of logarithmic-many labels, which is then checked to verify whether it is possible to derive from it a new transversal of \mathcal{G} w.r.t. \mathcal{H} .

To be more formal, for a hypergraph \mathcal{G} , let $\mathcal{L}_{\mathcal{G}}$ denote the set of all possible labels that can potentially be generated from edges of \mathcal{G} and vertices, i.e., $\mathcal{L}_{\mathcal{G}} = \{-v \mid v \in V\} \cup \{(v, G) \mid G \in \mathcal{G} \wedge v \in G\}$. A set of labels Σ of the tree $\mathcal{T}(\mathcal{G}, \mathcal{H})$ is any subset of $\mathcal{L}_{\mathcal{G}}$. Note the difference between a path of $\mathcal{T}(\mathcal{G}, \mathcal{H})$ and a set of labels of $\mathcal{T}(\mathcal{G}, \mathcal{H})$. The former is an ordered sequence of labels coherent with the structure of $\mathcal{T}(\mathcal{G}, \mathcal{H})$, while

the latter is just a set. Given the above notation, we define the set

$$\mathcal{S}^{\log}(\mathcal{G}, \mathcal{H}) = \{\Sigma \mid \Sigma \subseteq \mathcal{L}_{\mathcal{G}} \wedge 0 \leq |\Sigma| \leq \lfloor \log |\mathcal{H}| \rfloor + 1\}.$$

Given a set $\Sigma \in \mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$, the following expressions

$$(3) \quad \begin{aligned} In(\Sigma) &= \bigcup_{(v,G) \in \Sigma} \{v\}, \\ Ex(\Sigma) &= \left(\bigcup_{-v \in \Sigma} \{v\} \right) \cup \left(\bigcup_{(v,G) \in \Sigma} (G \setminus \{v\}) \right), \end{aligned}$$

indicate the sets of the included and excluded vertices in Σ , respectively. These two expressions are similar to the formulas to compute an assignment given a (valid) path in $\mathcal{T}(\mathcal{G}, \mathcal{H})$ (Formula (1) of Lemma 3.3). Since a set Σ is merely a set of labels, it may happen that $In(\Sigma) \cap Ex(\Sigma) \neq \emptyset$. When this is *not* the case we say that Σ is a *consistent* set of labels.

Given a set of labels Σ , we define $\sigma(\Sigma)$ as the pair $\langle In(\Sigma), Ex(\Sigma) \rangle$. If Σ is consistent, then $\sigma(\Sigma) = \langle In(\Sigma), Ex(\Sigma) \rangle$ is a (consistent) assignment as well. By a slight abuse of terminology and notation, given a set of labels Σ , regardless of whether Σ is actually consistent or not, we extend, in the natural way, the definitions given for (consistent) assignments to the pair $\sigma(\Sigma) = \langle In(\Sigma), Ex(\Sigma) \rangle$.

The following property is fundamental for the correctness of algorithm ND-NOTDUAL. In particular, it highlights that the order in which the labels of a path Π appear is not actually relevant to recognize that $\sigma(\Pi)^+$ is a witness.

LEMMA 4.1. *Let \mathcal{G} and \mathcal{H} be two hypergraphs satisfying the intersection property. Then, there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} if and only if there is a consistent set $\Sigma \in \mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$ such that $\sigma(\Sigma)^+$ is a double nonduality witness.*

Proof. (\Rightarrow) By Lemma 3.7, because \mathcal{G} and \mathcal{H} satisfy the intersection property, if there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} , then, in $\mathcal{T}(\mathcal{G}, \mathcal{H})$, there is a path Π of length at most $\lfloor \log |\mathcal{H}| \rfloor + 1$ such that $\sigma(\Pi)^+$ is a double nonduality witness. Let Σ^{Π} be the set of labels containing exactly the labels of Π (Σ^{Π} , unlike Π , is a set without any order over its elements). Because the length of Π is at most $\lfloor \log |\mathcal{H}| \rfloor + 1$, Σ^{Π} actually belongs to $\mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$. By comparing formula (1) of Lemma 3.3 and formula (3) of this section, it is clear that the included and excluded vertices of $\sigma(\Pi)$ do not depend on the actual order of the labels in Π . Hence, $\sigma(\Sigma^{\Pi}) = \sigma(\Pi)$ and also $\sigma(\Sigma^{\Pi})^+ = \sigma(\Pi)^+$. Given that Π is a path in $\mathcal{T}(\mathcal{G}, \mathcal{H})$, $\sigma(\Pi)$ is a (consistent) assignment by definition and, hence, from $\sigma(\Sigma^{\Pi}) = \sigma(\Pi)$ it follows that Σ is a consistent set. To conclude, since $\sigma(\Pi)^+$ is a double witness and $\sigma(\Sigma^{\Pi})^+ = \sigma(\Pi)^+$, $\sigma(\Sigma^{\Pi})^+$ is a double witness as well.

(\Leftarrow) Clearly, if there is a consistent set of labels $\Sigma \in \mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$ such that $\sigma(\Sigma)^+$ is a double witness, then there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} . \square

We now present our nondeterministic algorithm. The pseudocode of algorithm ND-NOTDUAL is listed as Algorithm 1; “**return accept**” and “**return reject**” are two commands causing a transition to a final accepting state and to a final rejecting state, respectively, of a nondeterministic machine.

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

Algorithm 1 A nondeterministic algorithm for $\overline{\text{DUAL}}$.

```

1: procedure ND-NOTDUAL( $\mathcal{G}, \mathcal{H}$ )
2:    $\Sigma \leftarrow \text{guess}$ (a set of labels from  $\mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$ );
3:   if  $\neg \text{CHECK-SIMPLE-AND-INTERSECTION}(\mathcal{G}, \mathcal{H})$  then return accept;
4:   if  $\neg \text{CHECK-CONSISTENCY}(\mathcal{G}, \Sigma)$  then return reject;
5:   if  $\text{CHECK-AUG-DOUBLEWITNESS}(\mathcal{G}, \mathcal{H}, \Sigma)$  then return accept;
6:   return reject;

```

CHECK-SIMPLE-AND-INTERSECTION: checks whether the two hypergraphs \mathcal{G} and \mathcal{H} are simple and satisfy the intersection property.⁶

CHECK-CONSISTENCY: checks whether the guessed set of labels Σ is actually consistent.

CHECK-AUG-DOUBLEWITNESS: checks whether $\sigma(\Sigma)^+$ is a double witness. In order to perform this check, condition (2) of Lemma 3.5 is evaluated on $\sigma(\Sigma)^+$.

The following property shows that algorithm ND-NOTDUAL is correct. We remind the reader that a nondeterministic algorithm A is correct for a decision problem P if A admits a computation branch terminating in an accepting state if and only if A is executed on a yes-instance of P .

THEOREM 4.2. *Let \mathcal{G} and \mathcal{H} be two hypergraphs. Then, there is 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. (\Rightarrow) Assume that there is a computation branch of ND-NOTDUAL(\mathcal{G}, \mathcal{H}) halting in an accepting state. A transition to an accepting state may happen only at line 3 or 5. If such a transition happens at line 3, then \mathcal{G} or \mathcal{H} is not simple, or \mathcal{G} and \mathcal{H} do not satisfy the intersection property. Hence, by Lemma 2.3, \mathcal{G} and \mathcal{H} are not dual.

If a transition to the accepting state occurs at line 5, it means that the execution flow of the algorithm has reached that point, implying that the test performed at line 4 passed. Therefore, the guessed set of labels Σ is consistent. The fact that the check at line 5 is successful implies that $\sigma(\Sigma)^+$ is a double witness. Thus, by Lemma 4.1, there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} and, hence, by Lemma 2.3, \mathcal{G} and \mathcal{H} are not dual.

(\Leftarrow) Let us now assume that \mathcal{G} and \mathcal{H} are not dual. By Lemma 2.3, \mathcal{G} or \mathcal{H} is not simple, or they do not satisfy the intersection property, or there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} .

If \mathcal{G} or \mathcal{H} is not simple, this condition is recognized by the algorithm at line 3 and the algorithm correctly moves to an accepting state. The same happens in the case \mathcal{G} and \mathcal{H} do not satisfy the intersection property.

Consider now the case in which \mathcal{G} and \mathcal{H} are simple and satisfy the intersection property. Since \mathcal{G} and \mathcal{H} are nondual, by Lemma 2.3, there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} .

Because \mathcal{G} and \mathcal{H} satisfy the intersection property, by Lemma 4.1, among the sets Σ guessed by the algorithm at line 2 there must be a consistent one such that $\sigma(\Sigma)^+$ is a double witness. This is recognized by the algorithm at line 5 and the algorithm correctly moves to an accepting state. \square

⁶This condition does not depend on the guessed set, 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.

Note that the approach of extending assignments used in our paper, while partly inspired by Gaur's ideas, is fundamentally different from the method used in Gaur's deterministic algorithm [21, 22]. 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 by 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 ND-NOTDUAL algorithm. We will show that the deterministic tests performed by algorithm ND-NOTDUAL require (quite) low computational effort, and in particular they can be carried out within complexity class TC^0 . This will allow us to prove that $\overline{DUAL} \in GC(\log^2 N, TC^0)$. We begin by expressing the deterministic tests performed by ND-NOTDUAL in $FO(COUNT)$ which is FO logic augmented with counting quantifiers " $\exists!n$ " having the following semantics. $FO(COUNT)$ is a two-sorted logic, i.e., a logic having two domain sets [33]: a numerical domain set containing objects used to interpret only numerical values, and another domain set containing all other objects. Consider the formula $\Phi(n, x) = (\exists!n x)(\phi(x))$, where variable x ranges over the nonnumerical domain objects and is bound by the counting quantifier $\exists!n$, and where variable n ranges over the numerical domain objects and is left free by the counting quantifier. $\Phi(n, x)$ is valid in all the interpretations in which n is substituted for by the exact number of nonnumerical domain values a for which $\phi(a)$ evaluates to **true**.⁷ Note that FOM, i.e., FO logic augmented with the majority qualifier, is known to be equivalent to $FO(COUNT)$ [33, 1, 51]. The model checking problem for both logics is complete for class TC^0 [51, 33, 1].

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 an object for each vertex of V , an object for each hyperedge of the two hypergraphs, and two more objects, $o_{\mathcal{G}}$ and $o_{\mathcal{H}}$, for the two hypergraphs, i.e., $A_{\langle \mathcal{G}, \mathcal{H} \rangle} = \{o_v \mid v \in V\} \cup \{o_G \mid G \in \mathcal{G}\} \cup \{o_H \mid H \in \mathcal{H}\} \cup \{o_{\mathcal{G}}, o_{\mathcal{H}}\}$.

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 set Σ . Remember that in Σ there are elements (which are labels) 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} .⁸ We assume a unary relation S_1 storing those tuples $\langle v \rangle$ where v is a vertex such that $-v \in \Sigma$, and 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 $(v, G) \in \Sigma$.

Remember that, by Lemma 4.1, it is sufficient to guess a set of labels, and it is not required to guess a path. This means that the exact order of the labels is not relevant and, hence, the above relational representation of a guessed set is totally sufficient.

⁷For more on this, the reader is referred to any standard textbook on the topic. See, e.g., [10, 33, 9, 42].

⁸Note that an edge G in a label of a path or a set is given by its identifier and not by the explicit list of its vertices.

We use the following “macros” in our FO formulas:

$$\begin{aligned} v \in V &\equiv \text{Vertex}(v), \\ g \in \mathcal{G} &\equiv \text{Hyp}(o_{\mathcal{G}}) \wedge \text{EdgeOf}(g, o_{\mathcal{G}}), \\ h \in \mathcal{H} &\equiv \text{Hyp}(o_{\mathcal{H}}) \wedge \text{EdgeOf}(h, o_{\mathcal{H}}), \\ v \in g &\equiv \text{In}(v, g). \end{aligned}$$

We are now ready to prove some intermediate results.

LEMMA 4.3. *Let \mathcal{G} be a hypergraph. Deciding whether \mathcal{G} is simple is expressible in FO.*

Proof. We know that a hypergraph \mathcal{G} is simple if and only if, for all pairs of distinct edges $G, H \in \mathcal{G}$, $G \not\subseteq H$. Hence, the formula checking whether a hypergraph is simple is

$$\begin{aligned} \text{simple}(x) &\equiv \text{Hyp}(x) \\ &\wedge (\forall g, h)((g \in x \wedge h \in x \wedge g \neq h) \rightarrow (\exists v)(v \in V \wedge v \in g \wedge \neg(v \in h))). \quad \square \end{aligned}$$

LEMMA 4.4. *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. Two hypergraphs \mathcal{G} and \mathcal{H} satisfy the intersection property if and only if, for every pair of edges $G \in \mathcal{G}$ and $H \in \mathcal{H}$, $G \cap H \neq \emptyset$. Hence, the formula encoding this test is

$$\text{intersection-property} \equiv (\forall g, h)((g \in \mathcal{G} \wedge h \in \mathcal{H}) \rightarrow (\exists v)(v \in V \wedge v \in g \wedge v \in h)). \quad \square$$

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

LEMMA 4.5. *Let \mathcal{G} and \mathcal{H} be two hypergraphs, and let Σ be a (guessed) set of labels of $\mathcal{T}(\mathcal{G}, \mathcal{H})$. Deciding the congruency and the consistency of Σ is expressible in FO.*

Proof. The congruency of the guessed set can be checked through

$$\text{congruentGuess} \equiv (\forall v)(S_1(v) \rightarrow v \in V) \wedge (\forall w, g)(S_2(w, g) \rightarrow w \in V \wedge g \in \mathcal{G} \wedge w \in g).$$

The consistency check is expressed as the conjunction of two formulas. The first verifies whether there is an inconsistency on a specific vertex, and the second checks the overall consistency:

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

To conclude our complexity analysis of the deterministic tests performed by ND-NOTDUAL, let us formulate in FO(COUNT) the property that $\sigma(\Sigma)^+$ is a double nonduality witness.

LEMMA 4.6. *Let \mathcal{G} and \mathcal{H} be two hypergraphs, and let Σ be a (guessed) set of labels of $\mathcal{T}(\mathcal{G}, \mathcal{H})$. Deciding whether $\sigma(\Sigma)^+$ is a double nonduality witness is expressible in $\text{FO}(\text{COUNT})$.*

Proof. Let $\sigma(\Sigma) = \langle \text{In}(\Sigma), \text{Ex}(\Sigma) \rangle$ be the pair associated with Σ . Essentially we need to prove that is possible to express, in $\text{FO}(\text{COUNT})$, condition (2) of Lemma 3.5 on $\sigma(\Sigma)^+$. Remember that $\sigma(\Sigma)$ is not explicitly represented, but it can be evaluated from Σ through formula (3) of subsection 4.1.

Let us define the following two formulas serving the purpose of evaluating whether a vertex belongs to $\text{In}(\Sigma)$ or $\text{Ex}(\Sigma)$, respectively:

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

We now exhibit the formulas to verify whether a given free vertex is frequent in $\sigma(\Sigma)$. These formulas are the only ones in which we actually use counting quantifiers. In the following formulas we will use predicate $PLUS(x, y, z)$, which holds **true** whenever $x + y = z$, and predicate $SUCC(x, y)$, which holds **true** whenever x and y are two domain values such that y is the immediate successor of x in the domain ordering. Remember, indeed, that relational structures are assumed to have totally ordered domains (and predicate “ $<$ ” allows us to test the ordering), and to have a predicate $BIT(i, j)$ that holds **true** whenever the j th bit of the binary representation of number i is 1. These assumptions allow us to express in FO logic, predicates $PLUS(x, y, z)$ and $SUCC(x, y)$ (see section 1.2 of [33]).

Since we need to evaluate whether a vertex v is frequent in $\sigma(\Sigma)$, we have to check whether v belongs to at least $\lceil |Com(\sigma(\Sigma))|/2 \rceil$ edges of $Com(\sigma(\Sigma))$. So, we exhibit a formula $half(x, y)$ which holds **true** whenever $y = \lceil x/2 \rceil$:

$$half(x, y) \equiv PLUS(y, y, x) \vee (\exists z)(PLUS(y, y, z) \wedge SUCC(x, z)).$$

The following formulas evaluate whether an edge belongs to $Com(\sigma(\Sigma))$, the number of edges in $Com(\sigma(\Sigma))$, the number of edges in $Com(\sigma(\Sigma))$ containing a given vertex v , and whether v is frequent in $\sigma(\Sigma)$, respectively (remember that being either frequent or infrequent is a property of free vertices):

$$\begin{aligned} com(h) &\equiv h \in \mathcal{H} \wedge (\forall v)((v \in V \wedge v \in h) \rightarrow \neg E\text{-guess}(v)), \\ count\text{-}com(n) &\equiv (\exists!n \ h)(h \in \mathcal{H} \wedge com(h)), \\ count\text{-}com\text{-}inc(v, n) &\equiv v \in V \wedge (\exists!n \ h)(h \in \mathcal{H} \wedge com(h) \wedge v \in h), \\ freq(v) &\equiv v \in V \wedge \neg I\text{-guess}(v) \wedge \neg E\text{-guess}(v) \wedge (\exists n, m, o) \\ &\quad (count\text{-}com(n) \wedge count\text{-}com\text{-}inc(v, m) \wedge half(n, o) \wedge (o = m \vee o < m)). \end{aligned}$$

After having defined a formula to evaluate whether a vertex is frequent in $\sigma(\Sigma)$, we show the formulas computing the included and excluded vertices of the augmented pair $\sigma(\Sigma)^+$, respectively:

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

Now we show the formulas encoding the evaluation of condition (2) of Lemma 3.5 on $\sigma(\Sigma)^+$. The formulas evaluating whether an edge belongs to $Sep(\sigma(\Sigma)^+)$ and to

$Com(\sigma(\Sigma)^+)$ are, respectively,

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

Finally, the formula verifying that $\sigma(\Sigma)^+$ meets condition (2) of Lemma 3.5 is as follows.

CheckGuessAugDoubleWitness

$$\equiv (\forall g)(g \in \mathcal{G} \rightarrow \neg sep\text{-}aug(g)) \wedge (\forall h)(h \in \mathcal{H} \rightarrow \neg com\text{-}aug(h)). \quad \square$$

4.3. Putting it all together. We are now ready to prove our main results.

THEOREM 4.7. *Let \mathcal{G} and \mathcal{H} be two hypergraphs. Then, deciding whether \mathcal{G} and \mathcal{H} are not dual is feasible in $GC(\log^2 N, TC^0)$.*

Proof. Lemmas 4.3 to 4.6 show that the deterministic checks performed by algorithm ND-NOTDUAL are expressible in $FO(COUNT)$. Hence, these tests are feasible in logtime-uniform TC^0 [33, 51, 1].

Moreover, by analyzing algorithm ND-NOTDUAL, clearly only $O(\log^2 N)$ nondeterministic bits are sufficient to be guessed to properly identify a (double) nonduality witness. Indeed, let us assume that \mathcal{G} and \mathcal{H} are not dual. If \mathcal{G} or \mathcal{H} is not simple, or \mathcal{G} and \mathcal{H} do not satisfy the intersection property, then the guessed set Σ is completely ignored, because \mathcal{G} and \mathcal{H} are directly recognized to be nondual and, hence, is totally irrelevant what the guessed bits are. On the other hand, if \mathcal{G} and \mathcal{H} are simple and satisfy the intersection property, since they are not dual, by Lemma 2.3, there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} . Therefore, by Lemma 4.1, there is in $\mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$ a set Σ with $O(\log |\mathcal{H}|)$ elements such that $\sigma(\Sigma)^+$ is a double witness. Remember that our definition of the size of the input of DUAL is $N = \|\mathcal{G}\| + \|\mathcal{H}\|$, hence, the number of elements of Σ is also $O(\log N)$. Since, by definition, $O(\log N)$ bits are sufficient to represent any vertex or edge identifier of the input hypergraphs, each label of Σ can be represented with only $O(\log N)$ bits. By this, the whole set Σ can be correctly represented and stored in the (set) variable Σ with $O(\log^2 N)$ bits.

Therefore, \overline{DUAL} belongs to $GC(\log^2 N, TC^0)$. \square

From the previous theorem the following corollaries follow immediately, the first of which proves that the conjecture stated by Gottlob [25] actually holds.

COROLLARY 4.8. *Let \mathcal{G} and \mathcal{H} be two hypergraphs. Deciding whether \mathcal{G} and \mathcal{H} are not dual is feasible in $GC(\log^2 N, LOGSPACE)$.*

Proof. The proof follows from Theorem 4.7 and the inclusion $TC^0 \subseteq LOGSPACE$. \square

COROLLARY 4.9 (see [25]). *Let \mathcal{G} and \mathcal{H} be two hypergraphs. Deciding whether \mathcal{G} and \mathcal{H} are dual is feasible in $DSPACE[\log^2 N]$.*

Proof. $\overline{DUAL} \in DSPACE[\log^2 N]$ follows from Corollary 4.8 and the inclusion $GC(\log^2 N, LOGSPACE) \subseteq DSPACE[\log^2 N]$. Since $DSPACE[\log^2 N]$ is closed under complement, $DUAL \in DSPACE[\log^2 N]$. \square

4.4. Computing a new transversal. In this section, we will show that computing a (not necessarily minimal) new transversal of \mathcal{G} w.r.t. \mathcal{H} is feasible in space $O(\log^2 N)$. First observe that it is possible to define a total order over $\mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$. Indeed, consider the totally ordered domain $A_{\langle \mathcal{G}, \mathcal{H} \rangle}$ of the relational structure $\mathcal{A}_{\langle \mathcal{G}, \mathcal{H} \rangle}$ (described in subsection 4.2), and in particular consider the space of pairs $P =$

$A_{\langle \mathcal{G}, \mathcal{H} \rangle} \times A_{\langle \mathcal{G}, \mathcal{H} \rangle}$. First, let us define an order over P by exploiting the ordering relation “ $<$ ” over $\mathcal{A}_{\langle \mathcal{G}, \mathcal{H} \rangle}$ (see subsection 4.2): given two pairs $p_1 = \langle a_1, b_1 \rangle$ and $p_2 = \langle a_2, b_2 \rangle$ belonging to P , p_1 precedes p_2 in P if and only if $(a_1 < a_2) \vee (a_1 = a_2 \wedge b_1 < b_2)$.

Now, we associate each label with a pair in P : label (v, G) is associated with pair $\langle o_v, o_G \rangle \in P$, where o_v and o_G are the objects of $A_{\langle \mathcal{G}, \mathcal{H} \rangle}$ associated with v and G , respectively; and label $-v$ is associated with pair $\langle o_v, o_v \rangle$, where o_v is the object of $A_{\langle \mathcal{G}, \mathcal{H} \rangle}$ associated with v . Given two labels ℓ_1 and ℓ_2 , ℓ_1 precedes ℓ_2 if and only if their respective associated pairs p_1 and p_2 are such that p_1 precedes p_2 in P .

To conclude, given two sets of labels $\Sigma_1, \Sigma_2 \in \mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$, Σ_1 precedes Σ_2 if and only if Σ_1 contains strictly fewer labels than Σ_2 , or Σ_1 and Σ_2 contain the same number of labels and the least labels $\ell_1 \in \Sigma_1$ and $\ell_2 \in \Sigma_2$ on which Σ_1 and Σ_2 differ are such that ℓ_1 precedes ℓ_2 .

Given this order, it is possible to enumerate all sets belonging to $\mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$ without repetitions.

Consider now the following deterministic algorithm COMPUTENT listed as Algorithm 2, which, given two hypergraphs \mathcal{G} and \mathcal{H} , successively generates all sets Σ belonging to $\mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$ to verify whether one of them is a good starting point to build a new transversal of \mathcal{G} w.r.t. \mathcal{H} . A prerequisite for the correct execution of algorithm COMPUTENT is that the input hypergraphs satisfy the intersection property, and the purpose of procedure CHECK-INTERSECTIONPROPERTY used in COMPUTENT is precisely that. This is required because COMPUTENT looks for sets of labels only among those in $\mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$, and Lemma 4.1 holds only if \mathcal{G} and \mathcal{H} satisfy the intersection property. In the pseudocode of the algorithm, “**return error**” is a command triggering an error state/signal.

Algorithm 2 A deterministic algorithm, derived from ND-NOTDUAL, computing a new transversal of \mathcal{G} w.r.t. \mathcal{H} . Here neither $\sigma(\Sigma)$ nor $\sigma(\Sigma)^+$ are explicitly stored, but they are dynamically computed as needed. We assume that $\sigma(\Sigma) = \langle In(\Sigma), Ex(\Sigma) \rangle$.

Require:

Hypergraphs \mathcal{G} and \mathcal{H} satisfy the intersection property.

- 1: **procedure** COMPUTENT(\mathcal{G}, \mathcal{H})
 - 2: **if** \neg CHECK-INTERSECTIONPROPERTY(\mathcal{G}, \mathcal{H}) **then return error**;
 - 3: **for** each $\Sigma: \Sigma \in \mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$ **do**
 - 4: **if** CHECK-CONSISTENCY(\mathcal{G}, Σ) **then**
 - 5: **if** CHECK-AUG-DOUBLEWITNESS($\mathcal{G}, \mathcal{H}, \Sigma$) **then return** $In(\Sigma) \cup Freq(\sigma(\Sigma))$;
 - 6: **return NIL**;
-

LEMMA 4.10. *Let \mathcal{G} and \mathcal{H} be two hypergraphs satisfying the intersection property. Then, algorithm COMPUTENT correctly computes a new transversal of \mathcal{G} w.r.t. \mathcal{H} (if it exists) in space $O(\log^2 N)$.*

Proof. COMPUTENT always terminates because sets belonging to $\mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$ are finite and, by exploiting the order defined, can be enumerated successively without repetitions.

First, COMPUTENT checks the intersection property, and if this property does not hold between the two input hypergraphs, then an error state/signal is triggered. Next, COMPUTENT successively enumerates all possible elements belonging to $\mathcal{S}^{\log}(\mathcal{G}, \mathcal{H})$ to find a set Σ (if one exists) such that $\sigma(\Sigma)^+$ meets condition (2) of Lemma 3.5, and hence such that $\sigma(\Sigma)^+$ is a double nonduality witness.

Observe that at line 4 the currently analyzed set Σ is checked for consistency, and at line 5 pair $\sigma(\Sigma)^+$ is tested to be a double nonduality witness. Hence, since \mathcal{G} and \mathcal{H} are checked at line 2 to satisfy the intersection property, by Lemma 4.1, there is a set $\Sigma = \langle \text{In}(\Sigma), \text{Ex}(\Sigma) \rangle$ passing the test at line 5 if and only if there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} . From $\sigma(\Sigma)^+$ being a double witness, it follows that $\text{In}(\Sigma) \cup \text{Freq}(\sigma(\Sigma))$ is a new transversal of \mathcal{G} w.r.t. \mathcal{H} (the reader can see from the proof of Lemma 3.7 that $\text{In}(\Sigma) \cup \text{Freq}(\sigma(\Sigma))$ is not always a minimal transversal of \mathcal{G}).

Algorithm COMPUTENT correctly outputs *NIL* at line 6 if no new transversal of \mathcal{G} exists.

To conclude, let us now show that COMPUTENT executes within a quadratic logspace bound. All sets Σ generated at line 3 contain at most $\lfloor \log |\mathcal{H}| \rfloor + 1$ labels, which is $O(\log N)$, and each of these sets can be represented with $O(\log^2 N)$ bits (see the proof of Theorem 4.7). For this reason, by reusing of workspace, the algorithm needs only $O(\log^2 N)$ bits to represent all the sets successively tried. Lemmas 4.4 to 4.6 show that all tests can be executed in TC^0 and hence in logarithmic space (by the inclusion $\text{TC}^0 \subseteq \text{LOGSPACE}$). In fact, in order to implement those tests within a logarithmic space bound, pairs $\sigma(\Sigma)$ and $\sigma(\Sigma)^+$, the sets of the separated and compatible edges, and the sets of frequent and infrequent vertices, are dynamically computed in LOGSPACE when needed, rather than being explicitly stored.

Observe also that the output operations can be carried out in logarithmic space. Indeed, the elements belonging to $\text{In}(\Sigma) \cup \text{Freq}(\sigma(\Sigma))$ can be output successively one by one using only logarithmic workspace by exploiting formula (3) of subsection 4.1 (for each vertex v it is decided whether v has to be output or not). Note that, given a vertex v , checking whether v is a free vertex of $\sigma(\Sigma)$ is feasible in TC^0 (see the proof of Lemma 4.6), and hence in LOGSPACE (from $\text{TC}^0 \subseteq \text{LOGSPACE}$). Moreover, deciding whether a free vertex of $\sigma(\Sigma)$ is frequent is feasible in TC^0 (see the proof of Lemma 4.6), and hence in LOGSPACE. \square

It is an open problem whether it is possible to compute a minimal new transversal of \mathcal{G} in space $O(\log^2 N)$.

From the previous lemma, the following theorem directly follows. Note that the result here reported is actually a (slight) improvement over the result in the conference paper [25], because we require here that the input hypergraphs satisfy the intersection property instead of the tighter condition of \mathcal{G} and \mathcal{H} being such that $\mathcal{G} \subseteq \text{tr}(\mathcal{H})$ and $\mathcal{H} \subseteq \text{tr}(\mathcal{G})$.

THEOREM 4.11 (improved over [25]). *Let \mathcal{G} and \mathcal{H} be two hypergraphs satisfying the intersection property. Then, computing a new (not necessarily minimal) transversal of \mathcal{G} w.r.t. \mathcal{H} is feasible in $O(\log^2 N)$ space.*

5. Conclusions and future research. In this paper, we studied the computational complexity of the DUAL problem. By using standard decomposition techniques for DUAL, we proved that after logarithmic many decomposition steps it is possible to individuate a subinstance of the original instance of DUAL for which verifying that it corresponds to a new transversal is feasible within complexity class TC^0 .

From this we devised a new nondeterministic algorithm for $\overline{\text{DUAL}}$ whose analysis allowed us to recognize $\text{GC}(\log^2 n, \text{TC}^0)$ as a new complexity upper bound for $\overline{\text{DUAL}}$. As a simple corollary of this result, we obtained also that $\overline{\text{DUAL}} \in \text{GC}(\log^2 n, \text{LOGSPACE})$, which was conjectured by Gottlob [25].

Moreover, the nondeterministic algorithm proposed in this paper is used to develop a simple deterministic algorithm whose space complexity is $O(\log^2 n)$.

Now some questions arise. Is it possible to avoid the counting in the final deterministic check phase of the nondeterministic algorithm without the need of guessing more bits than $O(\log^2 n)$? Or, more generally, without exceeding the upper bound of $O(\log^2 n)$ nondeterministic guessed bits, is it possible to devise a final deterministic test requiring a formula with strictly fewer quantifier alternations?

In the quest to find the exact complexity of the DUAL problem, there is another direction of investigation that could be interesting to explore.

Flum, Grohe, and Weyer [18] (see also [17]) defined a hierarchy of nondeterministic classes containing those languages that, after guessing $O(\log^2 n)$ bits, can be checked by an FO formula with a bounded number of quantifier alternations. A different definition of this very same hierarchy can also be found in a paper by Cai and Chen [7]. For notational convenience, let us denote these classes by $\text{GC}(\log^2 n, \mathcal{S})$, where \mathcal{S} is a sequence of logical quantifiers characterizing the quantifiers alternation in the formulas for the check of the languages in the class. Interestingly, there are natural decision problems that are complete for classes in this hierarchy, for example, the *tournament dominating set problem*, and the *Vapnik–Chervonenkis dimension problem*.

The former problem is defined as follows: given a tournament G (i.e., a directed graph such that for each pair of vertices v and w there is either an edge from v to w , or an edge from w to v (but not both)) and an integer k , decide whether there is a dominating set in G of size k . It can be shown that this problem is complete for the class $\text{GC}(\log^2 n, \forall\exists)$ [7, 18, 17].

The latter problem is defined as follows: given a hypergraph \mathcal{G} and an integer k , decide whether the Vapnik–Chervonenkis dimension of \mathcal{G} is at least k . It can be shown that this problem is complete for the class $\text{GC}(\log^2 n, \forall\exists\forall)$ [7, 18, 17].

In fact, a new hierarchy of classes characterized by limited nondeterminism could be defined. Indeed, we could extend the definitions given by Cai and Chen [7], and Flum, Grohe, and Weyer [18], to define the classes of languages that, after a nondeterministic guess of $O(\log^2 n)$ bits, can be checked by an FO(COUNT) formula with a bounded number of quantifier alternations.

In particular, for $\overline{\text{DUAL}}$, we hypothesize that the formula checking the guess of our nondeterministic algorithm, possibly rearranged and rewritten, is characterized by a quantifiers alternation $\forall\exists\text{C}$ (where C is the counting quantifier).⁹

Given such a new hierarchy, it would be interesting to verify whether $\overline{\text{DUAL}}$ belongs to the class $\text{GC}(\log^2 n, \forall\exists\text{C})$, and even whether $\overline{\text{DUAL}}$ is complete for this class.

Appendix A. Proofs of properties stated in section 3. Several proofs in this appendix omit some minor details. Full proofs can be found in the extended technical report [28] available online.

LEMMA A.1. *Let \mathcal{G} and \mathcal{H} be two hypergraphs, and let $\sigma = \langle \text{In}, \text{Ex} \rangle$ be an assignment. Then*

- (a) $\emptyset \in \mathcal{G} \Leftrightarrow \emptyset \in \mathcal{G}_{V \setminus \text{In}}$ and
 $\emptyset \in \mathcal{H} \Leftrightarrow \emptyset \in \mathcal{H}_{V \setminus \text{Ex}};$
- (b) $\emptyset \in \mathcal{G}_{V \setminus \text{In}} \Rightarrow \mathcal{G}(\sigma) = \{\emptyset\}$ and,
 $\emptyset \in \mathcal{H}_{V \setminus \text{Ex}} \Rightarrow \mathcal{H}(\sigma) = \{\emptyset\};$
- (c) $\mathcal{G} = \emptyset \Rightarrow \mathcal{G}_{V \setminus \text{In}} = \emptyset$ and
 $\mathcal{H} = \emptyset \Rightarrow \mathcal{H}_{V \setminus \text{Ex}} = \emptyset;$

⁹Note here that FO(COUNT) formulas with bounded quantifiers could be put in a relation with the levels of the logarithmic time counting hierarchy defined by Torán [49, 50].

- (d) $\mathcal{G}_{V \setminus In} = \emptyset \Leftrightarrow \mathcal{G}(\sigma) = \emptyset$ and,
 $\mathcal{H}_{V \setminus Ex} = \emptyset \Leftrightarrow \mathcal{H}(\sigma) = \emptyset$;
- (e) $\mathcal{G}(\sigma) = \emptyset$ if and only if In is a transversal of \mathcal{G} and,
 $\mathcal{H}(\sigma) = \emptyset$ if and only if Ex is a transversal of \mathcal{H} ;
- (f) $\mathcal{G}(\sigma) = \{\emptyset\}$ if and only if Ex is covering and,
 $\mathcal{H}(\sigma) = \{\emptyset\}$ if and only if In is covering;
- (g) $\mathcal{G}(\sigma)$ contains nonempty edges if and only if In is not a transversal of \mathcal{G} and
 Ex is not covering and,
 $\mathcal{H}(\sigma)$ contains nonempty edges if and only if Ex is not a transversal of \mathcal{H}
and In is not covering.

Proof.

- (a) The property follows from the fact that the empty edge is a subset of any set of vertices and from $\mathcal{G}_{V \setminus In} \subseteq \mathcal{G}$ and $\mathcal{H}_{V \setminus Ex} \subseteq \mathcal{H}$.
- (b) The property follows from the fact that the intersection of the empty edge with any set of vertices is the empty set and from the fact that $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ undergo a minimization operation.
- (c) The property follows from $\mathcal{G}_{V \setminus In} \subseteq \mathcal{G}$ and $\mathcal{H}_{V \setminus Ex} \subseteq \mathcal{H}$.
- (d) We prove the property for \mathcal{G} . The proof for \mathcal{H} is symmetric.
 (\Rightarrow) If $\mathcal{G}_{V \setminus In} = \emptyset$, then $\mathcal{G}(\sigma) = \emptyset$ by definition.
 (\Leftarrow) Assume that $\mathcal{G}(\sigma) = \emptyset$ and let us assume by contradiction that $\mathcal{G}_{V \setminus In} \neq \emptyset$ (i.e., $\mathcal{G}_{V \setminus In}$ contains edges). There are two cases: either (1) there is an edge $G \in \mathcal{G}_{V \setminus In}$ such that $G \cap (V \setminus (In \cup Ex)) = \emptyset$ (observe that it may be the case that such edge G is the empty one), or (2) there is not such an edge (i.e., all edges in $\mathcal{G}_{V \setminus In}$ have a nonempty intersection with $V \setminus (In \cup Ex)$). In case (1), since there is an edge $G \in \mathcal{G}_{V \setminus In}$ such that $G \cap (V \setminus (In \cup Ex))$ is empty, $\emptyset \in \mathcal{G}(\sigma)$: a contradiction, because we are assuming $\mathcal{G}(\sigma) = \emptyset$. For case (2), since, for all edges $G \in \mathcal{G}_{V \setminus In}$, $G \cap (V \setminus (In \cup Ex))$ is not empty, $\mathcal{G}(\sigma)$ contains nonempty edges: a contradiction, because we are assuming $\mathcal{G}(\sigma) = \emptyset$. Therefore, it must be the case that $\mathcal{G}_{V \setminus In} = \emptyset$.
- (e) We prove the property for $\mathcal{G}(\sigma)$. The proof for $\mathcal{H}(\sigma)$ is symmetric.
If $\emptyset \in \mathcal{G}$ or $\mathcal{G} = \emptyset$, the property trivially holds.
Assume that \mathcal{G} contains only nonempty edges. Consider $\mathcal{G}_{V \setminus In}$ and observe that, by definition, when $\mathcal{G} \neq \emptyset$, $\mathcal{G}_{V \setminus In} = \emptyset$ if and only if In is a transversal of \mathcal{G} . Therefore, by point (d), this property follows.
- (f) We prove the property for $\mathcal{G}(\sigma)$. The proof for $\mathcal{H}(\sigma)$ is symmetric.
If $\emptyset \in \mathcal{G}$ or $\mathcal{G} = \emptyset$, the property trivially holds.
Assume that \mathcal{G} contains only nonempty edges. First note that, since \mathcal{G} contains only nonempty edges, by point (a), $\emptyset \notin \mathcal{G}_{V \setminus In}$.
 (\Rightarrow) Since $\mathcal{G}(\sigma) = \{\emptyset\}$, by point (d), $\mathcal{G}_{V \setminus In} \neq \emptyset$. Therefore, $\mathcal{G}_{V \setminus In}$ contains only nonempty edges. Let us assume by contradiction that Ex is not covering, and let $G \in \mathcal{G}_{V \setminus In}$ be an edge. By definition of $\mathcal{G}_{V \setminus In}$, for any $G \in \mathcal{G}_{V \setminus In}$, $G \subseteq V \setminus In$ and, hence, $G \cap In = \emptyset$. Since $\mathcal{G}_{V \setminus In} \subseteq \mathcal{G}$ and Ex is not covering, there is a vertex $v \in (G \setminus Ex)$. From $G \cap In = \emptyset$ and $v \in (G \setminus Ex)$, it follows that $v \in (G \cap (V \setminus (In \cup Ex)))$ and hence that $G \cap (V \setminus (In \cup Ex)) \neq \emptyset$. Therefore, $\mathcal{G}(\sigma)$ contains a nonempty edge: a contradiction, because we are assuming $\mathcal{G}(\sigma) = \{\emptyset\}$. Thus, Ex is covering.
 (\Leftarrow) Remember that $\mathcal{G}_{V \setminus In}$ contains all and only the edges $G \in \mathcal{G}$ such that $G \cap In = \emptyset$. Since Ex is covering, there is an edge $G \in \mathcal{G}$ such that $G \subseteq Ex$. Observe that G must belong to $\mathcal{G}_{V \setminus In}$ (because In and Ex are disjoint).

Therefore, from $G \in \mathcal{G}_{V \setminus In}$ and $G \cap (V \setminus (In \cup Ex)) = \emptyset$, it follows that $\mathcal{G}(\sigma) = \{\emptyset\}$.

- (g) The property follows from points (e) and (f). \square

LEMMA A.2. *Let \mathcal{G} and \mathcal{H} be two hypergraphs satisfying the intersection property, and let $\sigma = \langle In, Ex \rangle$ be a covering assignment. Then*

- (a) *In and Ex cannot both be covering;*
- (b) *$\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are trivially dual. In particular, if In is covering, then $\mathcal{G}(\sigma) = \emptyset$ and $\mathcal{H}(\sigma) = \{\emptyset\}$; and, symmetrically, if Ex is covering, then $\mathcal{G}(\sigma) = \{\emptyset\}$ and $\mathcal{H}(\sigma) = \emptyset$.*

Proof.

- (a) If $\mathcal{G} = \emptyset$ (resp., $\mathcal{H} = \emptyset$) or $\emptyset \in \mathcal{G}$ (resp., $\emptyset \in \mathcal{H}$), the property trivially holds.

Assume that both \mathcal{G} and \mathcal{H} contain only nonempty edges. Assume by contradiction that there are two edges $G \in \mathcal{G}$ and $H \in \mathcal{H}$ such that $G \subseteq Ex$ and $H \subseteq In$. Because of the intersection property, from $G \cap H \neq \emptyset$ follows $In \cap Ex \neq \emptyset$, a contradiction, because σ is an assignment.

- (b) If $\emptyset \in \mathcal{G}$, then $\mathcal{H} = \emptyset$ by the intersection property. In this case, any set of vertices Ex is covering, because any Ex is a superset of the empty edge contained in \mathcal{G} . By Lemma A.1 points (a) and (b), $\mathcal{G}(\sigma) = \{\emptyset\}$ and, by Lemma A.1 points (c) and (d), $\mathcal{H} = \emptyset$. Symmetrically, if $\emptyset \in \mathcal{H}$, then $\mathcal{G} = \emptyset$ by the intersection property, In is covering, $\mathcal{G} = \emptyset$, and $\mathcal{H}(\sigma) = \{\emptyset\}$.

If $\mathcal{G} = \emptyset$, then $\mathcal{H} \neq \emptyset$ for otherwise σ would not be covering. Since $\mathcal{G} = \emptyset$, for σ to be covering it must be the case that In is covering. By Lemma A.1 point (f), $\mathcal{H}(\sigma) = \{\emptyset\}$, and by Lemma A.1 points (c) and (d), $\mathcal{G}(\sigma) = \emptyset$. Symmetrically, if $\mathcal{H} = \emptyset$, then $\mathcal{G} \neq \emptyset$ for otherwise σ would not be covering, and it can be shown that Ex is covering, $\mathcal{G}(\sigma) = \{\emptyset\}$, and $\mathcal{H} = \emptyset$.

Assume that both \mathcal{G} and \mathcal{H} contain only nonempty edges. If In is covering, then In is a transversal of \mathcal{G} because \mathcal{G} and \mathcal{H} satisfy the intersection property. Thus, by Lemma A.1 point (e), $\mathcal{G}(\sigma) = \emptyset$. Since In is covering, by Lemma A.1 point (f), $\mathcal{H}(\sigma) = \{\emptyset\}$. Symmetrically, it can be shown that, if Ex is covering, then $\mathcal{G}(\sigma) = \{\emptyset\}$ and $\mathcal{H}(\sigma) = \emptyset$. \square

LEMMA A.3. *Let \mathcal{G} and \mathcal{H} be two hypergraphs. Then, \mathcal{G} and \mathcal{H} satisfy the intersection property if and only if, for all assignments σ , $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ satisfy the intersection property.*

Proof. (\Rightarrow) If $\mathcal{G} = \emptyset$ or $\emptyset \in \mathcal{G}$, the property trivially holds.

Assume that both \mathcal{G} and \mathcal{H} contain only nonempty edges. Let $\sigma = \langle In, Ex \rangle$. If σ is covering, then, by Lemma A.2 point (b), $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are (trivially) dual, and hence they also satisfy the intersection property.

In case σ is noncovering, if In (resp., Ex) is a transversal of \mathcal{G} (resp., \mathcal{H}), then, by Lemma A.1 point (e), $\mathcal{G}(\sigma) = \emptyset$ (resp., $\mathcal{H}(\sigma) = \emptyset$). In these cases, since at least one of the two hypergraphs $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ is empty, they satisfy the intersection property.

Let us consider now the case in which both In and Ex are not transversals of $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$, respectively. By Lemma A.1 point (g), both $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ contain only nonempty edges. Assume by contradiction that there are two edges $G' \in \mathcal{G}(\sigma)$ and $H' \in \mathcal{H}(\sigma)$ such that $G' \cap H' = \emptyset$. Since $G' \in \mathcal{G}(\sigma)$ there is an edge $G \in \mathcal{G}$ such that $G = G' \cup A$ with $\emptyset \subseteq A \subseteq Ex$, and with $G \cap In = \emptyset$ (for otherwise G' would not be in $\mathcal{G}(\sigma)$). Moreover, since $H' \in \mathcal{H}(\sigma)$ there is an edge $H \in \mathcal{H}$ such that

$H = H' \cup B$ with $\emptyset \subseteq B \subseteq In$, and with $H \cap Ex = \emptyset$ (for otherwise H' would not be in $\mathcal{H}(\sigma)$). From this follows $G \cap H = \emptyset$, a contradiction, because \mathcal{G} and \mathcal{H} satisfy the intersection property. Therefore, $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ satisfy the intersection property.

(\Leftarrow) Since \mathcal{G} and \mathcal{H} do not satisfy the intersection property, there are two edges $G \in \mathcal{G}$ and $H \in \mathcal{H}$ such that $G \cap H = \emptyset$. Assume without loss of generality that G and H are not supersets of other edges (remember that, in the statement of the lemma, \mathcal{G} and \mathcal{H} are not assumed to be simple). Consider the empty assignment σ_ε . Observe that $G \in \mathcal{G}(\sigma_\varepsilon)$ and $H \in \mathcal{H}(\sigma)$ and hence $\mathcal{G}(\sigma_\varepsilon)$ and $\mathcal{H}(\sigma_\varepsilon)$ do not satisfy the intersection property. \square

LEMMA A.4. *Let \mathcal{G} and \mathcal{H} be two hypergraphs, and let $\sigma = \langle In, Ex \rangle$ be an assignment.*

- (a) *If T' is a transversal of $\mathcal{G}(\sigma)$, then $T = T' \cup In$ is a transversal of \mathcal{G} .*
- (b) *If T' is an independent set of $\mathcal{H}(\sigma)$, then $T = T' \cup In$ is an independent set of \mathcal{H} .*

Hence, if T' is a new transversal of $\mathcal{G}(\sigma)$ w.r.t. $\mathcal{H}(\sigma)$, then $T = T' \cup In$ is a new transversal of \mathcal{G} w.r.t. \mathcal{H} .

Proof. We remind the reader that, since $\mathcal{G}(\sigma)$ (resp., $\mathcal{H}(\sigma)$) undergoes a minimization operation, it is not possible that $\emptyset \in \mathcal{G}(\sigma)$ and $\mathcal{G}(\sigma) \neq \{\emptyset\}$ (resp., $\emptyset \in \mathcal{H}(\sigma)$ and $\mathcal{H}(\sigma) \neq \{\emptyset\}$) at the same time. Therefore, $\mathcal{G}(\sigma)$ (resp., $\mathcal{H}(\sigma)$) fulfills exactly one of these conditions: is empty, or contains only the empty edge, or contains only nonempty edges.

- (a) If $\mathcal{G}(\sigma) = \{\emptyset\}$ or $\mathcal{G}(\sigma) = \emptyset$, then the property can be easily proven.

Assume that $\mathcal{G}(\sigma)$ contains only nonempty edges. By Lemma A.1 points (a), (b), (c), and (d), \mathcal{G} contains only nonempty edges. Observe that, for each edge $G \in \mathcal{G}$, there are two cases: either $G \cap In \neq \emptyset$ or $G \cap In = \emptyset$. If $G \cap In \neq \emptyset$, then $T \cap G \neq \emptyset$ because $In \subseteq T$. If $G \cap In = \emptyset$, then there is an edge $\emptyset \neq G' \in \mathcal{G}(\sigma)$ such that $G' \subseteq G$. Hence, $T \cap G \neq \emptyset$ because $T' \cap G' \neq \emptyset$, $G' \subseteq G$, and $T' \subseteq T$.

- (b) If $\mathcal{H}(\sigma) = \{\emptyset\}$ or $\mathcal{H}(\sigma) = \emptyset$, then the property can be easily proven.

Assume that $\mathcal{H}(\sigma)$ contains only nonempty edges. Since the vertex set of $\mathcal{H}(\sigma)$ is $V \setminus (In \cup Ex)$, $T' \subseteq V \setminus (In \cup Ex)$. Observe that, for each edge $H \in \mathcal{H}$, there are two cases: either $H \cap Ex \neq \emptyset$ or $H \cap Ex = \emptyset$. From $T' \subseteq V \setminus (In \cup Ex)$ and $In \cap Ex = \emptyset$, it follows that $T \cap Ex = \emptyset$. So, if $H \cap Ex \neq \emptyset$, then $H \not\subseteq T$. On the other hand, if $H \cap Ex = \emptyset$, then there is an edge $\emptyset \neq H' \in \mathcal{H}(\sigma)$ such that $H' \subseteq H$. Since T' is an independent set of $\mathcal{H}(\sigma)$, there is a vertex $v \in (H' \setminus T')$. From $H' \subseteq V \setminus (In \cup Ex)$ and $v \in H'$, it follows that $v \notin In$. Therefore, since $T = T' \cup In$, $v \notin T$, and, because $v \in H' \subseteq H$, $H \not\subseteq T$. \square

Observe that, by the symmetrical nature of the DUAL problem, the roles of \mathcal{G} and \mathcal{H} can be swapped in an instance of DUAL. By this reason, Lemma A.4 can be easily adapted to state that transversals of $\mathcal{H}(\sigma)$ and independent sets of $\mathcal{G}(\sigma)$ can be extended, in this case by adding Ex , to be transversals of \mathcal{H} and independent sets of \mathcal{G} , respectively.

LEMMA A.5. *Let \mathcal{G} and \mathcal{H} be two hypergraphs.*

- (a) *If T is a transversal of \mathcal{G} , then, for any assignment $\sigma = \langle In, Ex \rangle$ coherent with T , $T' = T \setminus In$ is a transversal of $\mathcal{G}(\sigma)$.*
- (b) *If T is an independent set of \mathcal{H} , then, for any assignment $\sigma = \langle In, Ex \rangle$ coherent with T , $T' = T \setminus In$ is an independent set of $\mathcal{H}(\sigma)$.*

Hence, if T is a new transversal of \mathcal{G} w.r.t. \mathcal{H} , then, for any assignment $\sigma = \langle In, Ex \rangle$ coherent with T , $T' = T \setminus In$ is a new transversal of $\mathcal{G}(\sigma)$ w.r.t. $\mathcal{H}(\sigma)$.

Proof.

- (a) If $\emptyset \in \mathcal{G}$ or $\mathcal{G} = \emptyset$, the property trivially holds.

Assume that \mathcal{G} contains only nonempty edges. There are two cases: either In is a transversal of \mathcal{G} , or it is not. If In is a transversal of \mathcal{G} , then, by Lemma A.1 point (e), $\mathcal{G}(\sigma) = \emptyset$. Hence, trivially any set of vertices is a transversal of $\mathcal{G}(\sigma)$, and so is T' . Consider now the case in which In is not a transversal of \mathcal{G} . Since T is a transversal of \mathcal{G} coherent with σ , by (the contrapositive of) Lemma 3.1 point (a), Ex is not covering. Therefore, by Lemma A.1 point (g), $\mathcal{G}(\sigma)$ contains only nonempty edges. Let $G' \in \mathcal{G}(\sigma)$ be an edge. By definition of $\mathcal{G}(\sigma)$, there is an edge $G \in \mathcal{G}$ such that $G \cap In = \emptyset$ (for otherwise G' would not be in $\mathcal{G}(\sigma)$) and $G' = G \cap (V \setminus (In \cup Ex))$. Since T is a transversal of \mathcal{G} , $T \cap (G \setminus In) \neq \emptyset$, because $G \cap In = \emptyset$, and $T \cap (G \setminus Ex) \neq \emptyset$, because $\sigma \sqsubseteq T$ and so $T \cap Ex = \emptyset$. Therefore, $T \cap (G \setminus (In \cup Ex)) \neq \emptyset$. Since $G' = G \cap (V \setminus (In \cup Ex))$, $T' = T \setminus In$ has a nonempty intersection with G' and, hence, T' is a transversal of $\mathcal{G}(\sigma)$.

- (b) If $\emptyset \in \mathcal{H}$ or $\mathcal{H} = \emptyset$, the property trivially holds.

Assume that \mathcal{H} contains only nonempty edges. There are two cases: either Ex is a transversal of \mathcal{H} , or it is not. If Ex is a transversal of \mathcal{H} , then, by Lemma A.1 point (e), $\mathcal{H}(\sigma) = \emptyset$. Hence, trivially any set of vertices is an independent set of $\mathcal{H}(\sigma)$, and so is T' . Consider now the case in which Ex is not a transversal of \mathcal{H} . Since T is an independent set of \mathcal{H} coherent with σ , by (the contrapositive of) Lemma 3.1 point (b), In is not covering. Therefore, by Lemma A.1 point (g), $\mathcal{H}(\sigma)$ contains only nonempty edges. Let $H' \in \mathcal{H}(\sigma)$ be an edge. By the definition of $\mathcal{H}(\sigma)$, there is an edge $H \in \mathcal{H}$ such that $H \cap Ex = \emptyset$ (for otherwise H' would not be in $\mathcal{H}(\sigma)$) and $H' = H \cap (V \setminus (In \cup Ex))$. Since T is an independent set of \mathcal{H} , there is a vertex $v \in (H \setminus T)$ such that $v \notin Ex$, because $H \cap Ex = \emptyset$, and $v \notin In$, because $\sigma \sqsubseteq T$ and so $In \subseteq T$. Therefore, from $v \notin Ex$ and $v \notin In$, it follows that $v \in H'$ because $H' = H \cap (V \setminus (In \cup Ex))$, and from $v \notin T$ (because $v \in (H \setminus T)$), it follows that $v \notin T'$. Hence $H' \not\subseteq T'$, and thus T' is an independent set of $\mathcal{H}(\sigma)$. \square

Observe again that, by the symmetrical nature of the DUAL problem, by swapping the roles of \mathcal{G} and \mathcal{H} , and by considering the reversed assignment $\bar{\sigma} = \langle Ex, In \rangle$, Lemma A.5 can be easily adapted to state that transversals of \mathcal{H} and independent sets of \mathcal{G} coherent with $\bar{\sigma}$ can be shrunk, in this case by removing Ex , to be transversals of $\mathcal{H}(\sigma)$ and independent sets of $\mathcal{G}(\sigma)$, respectively.

The following corollary descends directly from Lemmas A.4 and A.5 and highlights an interesting property of decompositions.

COROLLARY A.6. *Let \mathcal{G} and \mathcal{H} be two hypergraphs. Then, for all assignments σ , there is a new transversal of $\mathcal{G}(\sigma)$ w.r.t. $\mathcal{H}(\sigma)$ if and only if there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} coherent with σ .*

From the previous corollary, by noticing that the empty assignment is coherent with any set of vertices, we obtain the next property.

COROLLARY A.7. *Let \mathcal{G} and \mathcal{H} be two hypergraphs. There is no new transversal of \mathcal{G} w.r.t. \mathcal{H} if and only if, for all assignments σ , there is no new transversal of $\mathcal{G}(\sigma)$ w.r.t. $\mathcal{H}(\sigma)$.*

LEMMA A.8. *Let \mathcal{G} and \mathcal{H} be two dual hypergraphs. Then, for all assignments σ , $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are dual.*

Proof. If \mathcal{G} and \mathcal{H} are trivially dual, then, by Lemma A.1 points (a), (b), (c), and (d), for any assignment σ , $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are trivially dual.

Consider now the case in which \mathcal{G} and \mathcal{H} are nontrivially dual. Since \mathcal{G} and \mathcal{H} are dual, they are simple, satisfy the intersection property, and there is no new transversal of \mathcal{G} w.r.t. \mathcal{H} (see Lemma 2.3). Observe that, for any assignment σ , $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are simple by definition and, by Lemma A.3, satisfy the intersection property. Because there is no new transversal of \mathcal{G} w.r.t. \mathcal{H} , by Corollary A.7, there is no new transversal of $\mathcal{G}(\sigma)$ w.r.t. $\mathcal{H}(\sigma)$. Therefore, from Lemma 2.3 it follows that $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are dual. \square

LEMMA A.9. *Let \mathcal{G} and \mathcal{H} be two simple hypergraphs. If, for all assignments σ , $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are dual, then \mathcal{G} and \mathcal{H} are dual.*

Proof. Since, for all assignments σ , $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are dual, by Lemma 2.3, for all assignments σ , $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ satisfy the intersection property and there is no new transversal of $\mathcal{G}(\sigma)$ w.r.t. $\mathcal{H}(\sigma)$. Hence, by Lemma A.3, \mathcal{G} and \mathcal{H} satisfy the intersection property. Because, for all assignments σ , there is no new transversal of $\mathcal{G}(\sigma)$ w.r.t. $\mathcal{H}(\sigma)$, by Corollary A.7, there is no new transversal of \mathcal{G} w.r.t. \mathcal{H} . Given that \mathcal{G} and \mathcal{H} are assumed to be simple, by Lemma 2.3 it follows that \mathcal{G} and \mathcal{H} are dual. \square

It is interesting to observe that in the statement of the previous lemma it is necessary to assume that hypergraphs \mathcal{G} and \mathcal{H} are simple. Indeed, for any assignment σ , $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are simple by definition, because they undergo a minimization operation. Hence, from $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ being dual (and hence also simple) we cannot derive \mathcal{G} and \mathcal{H} being simple. To give an example, consider hypergraphs $\mathcal{G} = \emptyset$ and \mathcal{H} being a hypergraph containing an empty edge and another edge. Hypergraph \mathcal{H} is not simple, and hence \mathcal{G} and \mathcal{H} are not dual. However, for any assignment σ , $\mathcal{G}(\sigma) = \emptyset$ and $\mathcal{H}(\sigma) = \{\emptyset\}$ (see Lemma A.1 points (a), (b), (c), and (d)) and, hence, $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are dual.

Given the properties above, the following lemma is a direct consequence of Lemmas 2.3, A.8, and A.9.

LEMMA 3.2. *Two hypergraphs \mathcal{G} and \mathcal{H} are dual if and only if \mathcal{G} and \mathcal{H} are simple, satisfy the intersection property, and, for all assignments σ , $\mathcal{G}(\sigma)$ and $\mathcal{H}(\sigma)$ are dual (or, equivalently, there is no new transversal of $\mathcal{G}(\sigma)$ w.r.t. $\mathcal{H}(\sigma)$).*

Appendix B. A deterministic algorithm for DUAL. In this section we propose a deterministic duality algorithm DET-DUAL, which is an extension of that proposed by Gaur [21] (see also [22]). Algorithm DET-DUAL presented here is somewhat different from Gaur's because the latter checks self-duality of a single DNF Boolean formula, while ours verifies duality between two hypergraphs.

Given two hypergraphs \mathcal{G} and \mathcal{H} , our algorithm DET-DUAL, like many others, to disprove that \mathcal{G} and \mathcal{H} are dual aims at finding, via subprocedure NEW-TR, a new transversal of \mathcal{G} w.r.t. \mathcal{H} . To do so, the algorithm builds up, step by step, by including and excluding vertices, a set of vertices intersecting all edges of \mathcal{G} that is different from all edges of \mathcal{H} (i.e., a set of vertices that is a transversal of \mathcal{G} and an independent set of \mathcal{H} , and hence a new transversal of \mathcal{G} w.r.t. \mathcal{H}).

As already discussed, choosing vertices to exclude allows us to decrease the number of edges of \mathcal{H} that are not different (yet) from the candidate for a new transver-

sal. In particular, when NEW-TR excludes specific vertices, the number of edges of \mathcal{H} still needed to be considered is halved (see Lemma 3.4 and the discussion in subsection 3.3).

Let us now see the details of algorithm DET-DUAL. After exhibiting the algorithm, we will formally prove some of its properties and its correctness. We remind the reader that all missing proofs of this section can be found in the technical report [28] available online. Algorithm DET-DUAL, and more specifically subprocedure NEW-TR which checks the existence of new transversals, uses three sets to keep track of the included, excluded, and free vertices of the currently considered assignment, which are denoted by $Incl$, $Excl$, and $Free$, respectively. We remind the reader that, to recognize an assignment as a nonduality witness, we need to know what edges of \mathcal{G} are separated from and what edges of \mathcal{H} are still compatible with the assignment (and transversal) under construction. To this purpose, in the algorithm we use the sets denoted by $Sep_{\mathcal{G}}$, and $Com_{\mathcal{H}}$, respectively. Observe that, for simplicity of the presentation and better readability of the algorithm, we explicitly store here sets $Incl$, $Excl$, $Free$, $Sep_{\mathcal{G}}$, and $Com_{\mathcal{H}}$. This obviously requires more than quadratic logspace. However, by storing instead the labels only of the extensions leading to the current assignment, and evaluating the aforementioned sets dynamically (see the proof of Lemma 4.10 and subsection 4.2), the space complexity of the algorithm is bounded by DSPACE $[\log^2 N]$.

The DET-DUAL algorithm is listed below as Algorithm 3. The aim of the procedure CHECK-SIMPLE-AND-INTERSECTION is checking that hypergraphs \mathcal{G} and \mathcal{H} are simple and satisfy the intersection property.

Algorithm 3 A deterministic duality algorithm based on Gaur's.

```

1: procedure DET-DUAL( $\mathcal{G}, \mathcal{H}$ )
2:   if  $\neg$ CHECK-SIMPLE-AND-INTERSECTION( $\mathcal{G}, \mathcal{H}$ ) then return false;
3:   return  $\neg$ NEW-TR( $\mathcal{G}, \mathcal{H}, \emptyset, \emptyset, V$ );

4: procedure NEW-TR( $\mathcal{G}, \mathcal{H}, Incl, Excl, Free$ )
5:   if  $(\exists G)(G \in \mathcal{G} \wedge G \subseteq Excl) \vee (\exists H)(H \in \mathcal{H} \wedge H \subseteq Incl)$  then return false;
6:    $Sep_{\mathcal{G}} \leftarrow \{G \in \mathcal{G} \mid G \cap Incl = \emptyset\}$ ;
7:    $Com_{\mathcal{H}} \leftarrow \{H \in \mathcal{H} \mid H \cap Excl = \emptyset\}$ ;
8:   if  $Sep_{\mathcal{G}} = \emptyset \vee Com_{\mathcal{H}} = \emptyset$  then return true;
9:    $U \leftarrow \{v \in Free \mid v \text{ belongs to at least half of the edges of } Com_{\mathcal{H}}\}$ ;
10:  for each  $v : v \in U$  do
11:    if NEW-TR( $\mathcal{G}, \mathcal{H}, Incl, Excl \cup \{v\}, Free \setminus \{v\}$ ) then return true;
12:   $Free \leftarrow Free \setminus U$ ;
13:   $Incl \leftarrow Incl \cup U$ ;
14:  if  $(\exists H)(H \in \mathcal{H} \wedge H \subseteq Incl)$  then return false;
15:   $Sep_{\mathcal{G}} \leftarrow \{G \in \mathcal{G} \mid G \cap Incl = \emptyset\}$ ;
16:  if  $Sep_{\mathcal{G}} = \emptyset$  then return true;
17:  for each  $v : v \in Free$  do
18:    for each  $G : v \in G \wedge G \in Sep_{\mathcal{G}}$  do
19:      if NEW-TR( $\mathcal{G}, \mathcal{H}, Incl \cup \{v\}, Excl \cup (G \setminus \{v\}), Free \setminus G$ ) then return
20:    true;
21:  return false;

```

Note that, for simplicity, procedure NEW-TR is meant to be called by value. This means that the parameters passed to the procedure are local copies for each specific recursive call. Therefore, any modification to those sets affects only the sets of the call currently executed.

We recall here that two hypergraphs \mathcal{G} and \mathcal{H} are dual if and only if they are simple, satisfy the intersection property, and there is no new transversal of \mathcal{G} w.r.t. \mathcal{H} (see Lemma 2.3). So, after checking that \mathcal{G} and \mathcal{H} are simple and satisfy the intersection property (line 2), it is checked that there is no new transversal of \mathcal{G} . This is achieved by calling NEW-TR($\mathcal{G}, \mathcal{H}, \emptyset, \emptyset, V$) at line 3, where the sets *Incl* and *Excl* are initialized to \emptyset , and *Free* = V . This procedure is devised to answer **true** if and only if it finds a new transversal of \mathcal{G} coherent with the assignment on which it is executed. Given a pair of hypergraphs $\langle \mathcal{G}, \mathcal{H} \rangle$, and an assignment $\pi = \langle In, Ex \rangle$, we say that procedure NEW-TR is executed on π whenever it is called with the following parameters: NEW-TR($\mathcal{G}, \mathcal{H}, In, Ex, V \setminus (In \cup Ex)$). For notational convenience we denote it by NEW-TR($\mathcal{G}, \mathcal{H}, \pi$) or, even more simply, by NEW-TR(π) when it is clear from the context what the two hypergraphs \mathcal{G} and \mathcal{H} are.

Let us now analyze intuitively the execution of procedure NEW-TR. Consider a generic call of the procedure executed on the pair of hypergraphs $\langle \mathcal{G}, \mathcal{H} \rangle$ and on assignment π . At line 5 we check whether π is a covering assignment, in which case clearly there is no new transversal of \mathcal{G} coherent with π (see Lemma 3.1). If this is not the case, then the procedure checks (at line 8) whether π is (already) a witness. Then the procedure computes a set U (line 9). This set is locally computed in the call currently executed and, hence, for the following discussion, let us call it U_π . We use the subscript π because set U depends on the history of the recursive calls having led to the one currently being executed and, hence, depends on the currently considered assignment π having been built so far (and encoded in sets *Incl* and *Excl*).

Set U_π is the set of the free frequent vertices of π . First, the procedure tries to exclude individually each of the vertices of U_π (lines 10–11). If none of these attempts results in the construction of a witness (all the tests performed at line 11 return **false**), all vertices of U_π are included (lines 12–13). Let us call σ the assignment resulting after the inclusion of the vertices of U_π . Then, the procedure checks again whether σ is a covering assignment (line 14). Otherwise, the procedure tests whether σ is a witness (line 16). If this is not the case, then the procedure tries to include each of the free vertices of σ as a critical vertex with an edge from $Sep(\sigma)$ witnessing its criticality (lines 17–19). If for none of these attempts it is possible to find a new transversal of \mathcal{G} (all the tests performed at line 19 return **false**), then the procedure answers **false** at line 20, meaning that there is no new transversal of \mathcal{G} coherent with π .

Let us now make some observations on the procedure NEW-TR. Throughout the whole execution of the procedure and its recursive calls, the sets *Incl*, *Excl*, and *Free*, are always a partition of the set of vertices V and, hence, *Incl* and *Excl* constitute a consistent assignment (i.e., *Incl* and *Excl* do not overlap). This descends from the fact that the extensions implemented in procedure NEW-TR are the same as those considered in tree $\mathcal{T}(\mathcal{G}, \mathcal{H})$, and we have already discussed in section 3 that those extensions generate consistent assignments.

Besides this, every recursive call performed by NEW-TR(π) is executed on an assignment π' such that $\pi \sqsubset \pi'$. This implies that the set of free vertices becomes smaller and smaller from one recursive call to the next. As a result, since the procedure tries to include or exclude vertices taken only from the set of the free ones, every recursion path traversed by NEW-TR(π) is finite, because at some recursion level the set of free vertices is empty.

The tests performed at lines 5–8 and at lines 14–16 essentially check whether the assignment is a witness. In particular, first it is ruled out that the current assignment is covering, and then is checked whether $Sep_{\mathcal{G}}$ or $Com_{\mathcal{H}}$ is a transversal of \mathcal{G} or \mathcal{H} , respectively. Note that at lines 14–16 is tested only whether $Sep_{\mathcal{G}}$ is a new transversal of \mathcal{G} w.r.t. \mathcal{H} . At that point of the execution it is reasonable to do so because only $Sep_{\mathcal{G}}$ has changed after the previous check of the witnessing condition performed at lines 5–8.

We can now formally prove the correctness of algorithm DET-DUAL.

THEOREM B.1. *Let \mathcal{G} and \mathcal{H} be two hypergraphs. Then, the call $\text{DET-DUAL}(\mathcal{G}, \mathcal{H})$ outputs **true** if and only if \mathcal{G} and \mathcal{H} are dual.*

To prove the above theorem we need the following property.

LEMMA B.2. *Let \mathcal{G} and \mathcal{H} be two hypergraphs, and let π be an assignment. Then, the call $\text{NEW-TR}(\pi)$ answers **true** if and only if there is a new transversal of \mathcal{G} w.r.t. \mathcal{H} coherent with π .*

Proof of Theorem B.1. By Lemma 2.3, \mathcal{G} and \mathcal{H} are dual if and only if they are simple, satisfy the intersection property, and there is no new transversal of \mathcal{G} w.r.t. \mathcal{H} . Therefore, since \mathcal{G} and \mathcal{H} are explicitly checked to see if they are simple hypergraphs satisfying the intersection property (line 2), the correctness of the algorithm DET-DUAL directly follows from the correctness of the procedure NEW-TR (see Lemma B.2); in fact, any new transversal of \mathcal{G} , if it exists, is coherent with σ_{ε} which is the assignment on which NEW-TR is invoked by the procedure DET-DUAL (line 3). \square

Let us now consider the time complexity of the algorithm.

THEOREM B.3. *Let \mathcal{G} and \mathcal{H} be two hypergraphs satisfying the intersection property. Then, the time complexity of the algorithm DET-DUAL is $O(N^{O(\log N)})$.*

If we assume that the input hypergraphs \mathcal{G} and \mathcal{H} are guaranteed to fulfill $\mathcal{G} \subseteq \text{tr}(\mathcal{H})$ and $\mathcal{H} \subseteq \text{tr}(\mathcal{G})$ (i.e., we assume the stricter condition imposed on the input hypergraphs by Boros and Makino [6]), and are such that each vertex belongs to at least an edge in both \mathcal{G} and \mathcal{H} , then we can carry out a finer time complexity analysis.

THEOREM B.4. *Let \mathcal{G} and \mathcal{H} be two hypergraphs such that $\mathcal{G} \subseteq \text{tr}(\mathcal{H})$ and $\mathcal{H} \subseteq \text{tr}(\mathcal{G})$ and such that each vertex $v \in V$ belongs to an edge in \mathcal{G} and to an edge in \mathcal{H} . Then, the time complexity of algorithm DET-DUAL, in which the condition of \mathcal{G} and \mathcal{H} being such that $\mathcal{G} \subseteq \text{tr}(\mathcal{H})$ and $\mathcal{H} \subseteq \text{tr}(\mathcal{G})$ is checked, instead of them being simple and satisfying the intersection property, is $O((|\mathcal{G}| \cdot |\mathcal{H}|)^{O(\log |\mathcal{H}|)})$.*

It is now interesting to focus on the following fact. By the proof of Theorem B.3 (see the extended technical report [28]), the leaves of the recursion tree of procedure NEW-TR, which are candidate to certify the existence of a new transversal of \mathcal{G} , are $O(N^{2O(\log N)})$, but in principle there could be an exponential number of new minimal transversals of \mathcal{G} w.r.t. \mathcal{H} . For example, let us consider the class of pairs of hypergraphs $\{(\mathcal{G}_i, \mathcal{H}_i)\}_{i \geq 1}$, defined as follows: $V_i = \{x_1, y_1, \dots, x_i, y_i\}$, $\mathcal{G}_i = \{\{x_j, y_j\} \mid 1 \leq j \leq i\}$, and $\mathcal{H}_i = \{\{x_1, \dots, x_i\}, \{y_1, \dots, y_i\}\}$. For every $i \geq 1$, hypergraphs \mathcal{G}_i and \mathcal{H}_i satisfy the intersection property, $|\mathcal{G}_i| = i$, $|\mathcal{H}_i| = 2$, and the number of minimal transversals of \mathcal{G}_i missing in \mathcal{H}_i is $\Theta(2^i)$. So, it is not possible that each leaf of the recursion tree of NEW-TR identifies a unique new minimal transversal of \mathcal{G} . For this reason we want to know what new transversals of \mathcal{G} are identified by NEW-TR when it answers **true**.

THEOREM B.5. *Let \mathcal{G} and \mathcal{H} be two hypergraphs. If $\text{NEW-TR}(\sigma_\varepsilon)$ answers **true**, then the witness σ on which the procedure has answered **true** at the end of its recursion is coherent with a new minimal transversal of \mathcal{G} w.r.t. \mathcal{H} .*

Let \mathcal{G} and \mathcal{H} be two hypergraphs. Assume that $\text{NEW-TR}(\sigma_\varepsilon)$ answers **true**, and let $\sigma = \langle \text{In}, \text{Ex} \rangle$ be the assignment on which the procedure answers **true** at the end of its recursion. There are two cases: (1) $\text{Sep}(\sigma) = \emptyset$ or (2) $\text{Sep}(\sigma) \neq \emptyset$.

In case (1), by Theorem B.5, σ is coherent with a new minimal transversal of \mathcal{G} , and σ is such that $\text{Sep}(\sigma) = \emptyset$, hence it follows that In is a new minimal transversal of \mathcal{G} w.r.t. \mathcal{H} .

Consider case (2). Since σ is a witness and $\text{Sep}(\sigma) \neq \emptyset$, it must be the case that $\text{Com}(\sigma) = \emptyset$ and hence Ex is a transversal of \mathcal{H} . By this, $\overline{\text{Ex}}$ is an independent set of \mathcal{H} . Let us denote by $\text{Free}(\sigma) = V \setminus (\text{In} \cup \text{Ex})$ the set of free vertices in σ . Consider the set $T_\sigma = \{\text{In} \cup S \mid S \in \text{tr}(\text{Sep}(\sigma)^{\text{Free}(\sigma)})\}$.¹⁰ Transversals S of $\text{Sep}(\sigma)^{\text{Free}(\sigma)}$, by definition, are such that $S \cap \text{Ex} = \emptyset$ and, hence, $S \subseteq \overline{\text{Ex}}$ which means that S is an independent set of \mathcal{H} . Therefore, from an adaptation of Lemma A.4 to $\text{Sep}(\sigma)$, it follows that T_σ is a set of new transversals of \mathcal{G} . Note that some of the elements of T_σ are new minimal transversals of \mathcal{G} , while others are not minimal. However, there is no guarantee that there are no more minimal transversals, i.e., there are new minimal transversals of \mathcal{G} not belonging to T_σ .

Acknowledgments. We thank Thomas Eiter and Nikola Yolov for their very helpful comments on the preliminary version of this paper. In addition, we are grateful to the anonymous referees of the conference paper [27] and of this paper for their competent and helpful comments.

REFERENCES

- [1] D. A. M. BARRINGTON, N. IMMERMANN, AND H. STRAUBING, *On uniformity within NC^1* , J. Comput. System Sci., 41 (1990), pp. 274–306, [https://doi.org/10.1016/0022-0000\(90\)90022-D](https://doi.org/10.1016/0022-0000(90)90022-D).
- [2] C. BERGE, *Hypergraphs*, Vol. 45 of North-Holland Math. Libr., North-Holland, Amsterdam, 1989.
- [3] J. C. BIOCH AND T. IBARAKI, *Complexity of identification and dualization of positive Boolean functions*, Inform. and Comput., 123 (1995), pp. 50–63, <https://doi.org/10.1006/inco.1995.1157>.
- [4] E. BOROS, V. GURVICH, L. KHACHIYAN, AND K. MAKINO, *On the complexity of generating maximal frequent and minimal infrequent sets*, in Proceedings of the STACS 2002 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, 2002, Lecture Notes in Comput. Sci. 2285 H. Alt and A. Ferreira, eds., Springer, Berlin, 2002, pp. 133–141, https://doi.org/10.1007/3-540-45841-7_10.
- [5] E. BOROS, V. GURVICH, L. KHACHIYAN, AND K. MAKINO, *On maximal frequent and minimal infrequent sets in binary matrices*, Ann. Math. Artif. Intell., 39 (2003), pp. 211–221, <https://doi.org/10.1023/A:1024605820527>.
- [6] E. BOROS AND K. MAKINO, *A fast and simple parallel algorithm for the monotone duality problem*, in Proceedings of the 36th International Colloquium on Automata, Languages and Programming, ICALP 2009, Part I, Rhodes, Greece, 2009, Lecture Notes in Comput. Sci. 5555, S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. Nikolettseas, and W. Thomas, eds., Springer, Berlin, 2009, pp. 183–194, https://doi.org/10.1007/978-3-642-02927-1_17.
- [7] L. CAI AND J. CHEN, *On the amount of nondeterminism and the power of verifying*, SIAM J. Comput., 26 (1997), pp. 733–750, <https://doi.org/10.1137/S0097539793258295>.
- [8] S. COOK AND P. NGUYEN, *Logical foundations of proof complexity*, Cambridge University Press, Cambridge, UK, 2010.

¹⁰With a slight abuse of notation, we here regard $\text{Sep}(\sigma)$ as if it were a hypergraph. In this case, observe that $\text{Sep}(\sigma)^{\text{Free}(\sigma)}$ corresponds to $(\mathcal{G}_{V \setminus \text{In}})^{V \setminus (\text{In} \cup \text{Ex})}$.

- [9] H.-D. EBBINGHAUS AND J. FLUM, *Finite Model Theory*, Springer Monogr. Math., 2nd ed., Springer, Berlin, 1999.
- [10] H.-D. EBBINGHAUS, J. FLUM, AND W. THOMAS, *Mathematical Logic*, Undergrad. Texts Math., 2nd ed., Springer, New York, 1994.
- [11] T. EITER AND G. GOTTLÖB, *Identifying the minimal transversals of a hypergraph and related problems*, SIAM J. Comput., 24 (1995), pp. 1278–1304, <https://doi.org/10.1137/S0097539793250299>.
- [12] T. EITER AND G. GOTTLÖB, *Hypergraph transversal computation and related problems in Logic and AI*, in Proceedings of European Conference on Logics in Artificial Intelligence, JELIA 2002, Cosenza, Italy, S. Flesca, S. Greco, N. Leone, and G. Ianni, eds., Lecture Notes in Comput. Sci. 2424, Springer, Berlin, 2002, pp. 549–564, https://doi.org/10.1007/3-540-45757-7_53.
- [13] T. EITER, G. GOTTLÖB, AND K. MAKINO, *New results on monotone dualization and generating hypergraph transversals*, SIAM J. Comput., 32 (2003), pp. 514–537, <https://doi.org/10.1137/S009753970240639X>.
- [14] T. EITER AND K. MAKINO, *Generating all abductive explanations for queries on propositional Horn theories*, in Proceedings of the 17th International Workshop on Computer Science Logic, CSL 2003, Proceedings of the 12th Annual Conference of the EACSL, 8th Kurt Gödel Colloquium, KGC 2003, Vienna, Austria, 2003, M. Baaz and J. A. Makowsky, eds., Lecture Notes in Comput. Sci. 2803, Springer, Berlin, 2003, pp. 197–211, https://doi.org/10.1007/978-3-540-45220-1_18.
- [15] T. EITER, K. MAKINO, AND G. GOTTLÖB, *Computational aspects of monotone dualization: A brief survey*, Discrete Appl. Math., 156 (2008), pp. 2035–2049, <https://doi.org/10.1016/j.dam.2007.04.017>.
- [16] K. M. ELBASSIONI, *On the complexity of monotone dualization and generating minimal hypergraph transversals*, Discrete Appl. Math., 156 (2008), pp. 2109–2123, <https://doi.org/10.1016/j.dam.2007.05.030>.
- [17] J. FLUM AND M. GROHE, *Parameterized Complexity Theory*, Texts Theoret. Comput. Sci. EATES Ser., Springer, Berlin, 2006.
- [18] J. FLUM, M. GROHE, AND M. WEYER, *Bounded fixed-parameter tractability and $\log^2 n$ nondeterministic bits*, J. Comput. System Sci., 72 (2006), pp. 34–71, <https://doi.org/10.1016/j.jcss.2005.06.003>.
- [19] M. L. FREDMAN AND L. KHACHIYAN, *On the complexity of dualization of monotone disjunctive normal forms*, J. Algorithms, 21 (1996), pp. 618–628, <https://doi.org/10.1006/jagm.1996.0062>.
- [20] H. GARCIA-MOLINA AND D. BARBARA, *How to assign votes in a distributed system*, J. ACM, 32 (1985), pp. 841–860, <https://doi.org/10.1145/4221.4223>.
- [21] D. R. GAUR, *Satisfiability and Self-Duality of Monotone Boolean Functions*, Ph.D. thesis, School of Computing Science, Simon Fraser University, Burnaby, Canada, 1999.
- [22] D. R. GAUR AND R. KRISHNAMURTI, *Average case self-duality of monotone Boolean functions*, in Advances in Artificial Intelligence. 17th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2004, London, Canada, 2004, Proceedings, A. Y. Tawfik and S. D. Goodwin, eds., Lecture Notes in Comput. Sci. 3060, Springer, Berlin, 2004, pp. 322–338, https://doi.org/10.1007/978-3-540-24840-8_23.
- [23] G. GOGIC, C. H. PAPADIMITRIOU, AND M. SIDERI, *Incremental recompilation of knowledge*, J. Artif. Intell. Res., 8 (1998), pp. 23–37, <https://doi.org/10.1613/jair.380>.
- [24] J. GOLDSMITH, M. A. LEVY, AND M. MUNDHENK, *Limited nondeterminism*, ACM SIGACT News, 27 (1996), pp. 20–29, <https://doi.org/10.1145/235767.235769>.
- [25] G. GOTTLÖB, *Deciding monotone duality and identifying frequent itemsets in quadratic logspace*, in Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2013), R. Hull and W. Fan, eds., ACM, New York, 2013, pp. 25–36, <https://doi.org/10.1145/2463664.2463673>.
- [26] G. GOTTLÖB AND L. LIBKIN, *Investigations on Armstrong relations, dependency inference, and excluded functional dependencies*, Acta Cybernet., 9 (1990), pp. 385–402.
- [27] G. GOTTLÖB AND E. MALIZIA, *Achieving new upper bounds for the hypergraph duality problem through logic*, in Proceedings of the Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (CSL-LICS 2014), T. A. Henzinger and D. Miller, eds., ACM, New York, 2014, <https://doi.org/10.1145/2603088.2603103>.
- [28] G. GOTTLÖB AND E. MALIZIA, *Achieving New Upper Bounds for the Hypergraph Duality Problem Through Logic*, 2017, <https://arxiv.org/abs/1407.2912>.

- [29] R. GREINER, B. A. SMITH, AND R. W. WILKERSON, *A correction to the algorithm in Reiter's theory of diagnosis*, Artif. Intell., 41 (1989), pp. 79–88, [https://doi.org/10.1016/0004-3702\(89\)90079-9](https://doi.org/10.1016/0004-3702(89)90079-9).
- [30] D. GUNOPULOS, R. KHARDON, H. MANNILA, AND H. TOIVONEN, *Data mining, hypergraph transversals, and machine learning*, in Proceedings of the 16th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 1997), A. O. Mendelzon and Z. M. Özsoyoglu, eds., ACM, New York, 1997, pp. 209–216, <https://doi.org/10.1145/263661.263684>.
- [31] M. HAGEN, *Algorithmic and Computational Complexity Issues of MONET*, Cuvillier, Göttingen, Germany, 2008.
- [32] T. A. HENZINGER, O. KUPFERMAN, AND R. MAJUMDAR, *On the universal and existential fragments of the μ -calculus*, Theoret. Comput. Sci., 354 (2006), pp. 173–186, <https://doi.org/10.1016/j.tcs.2005.11.015>.
- [33] N. IMMERMAN, *Descriptive Complexity*, Springer, New York, 1999.
- [34] M. JURDZIŃSKI, *Deciding the winner in parity games is in $UP \cap coUP$* , Inform. Process. Lett., 68 (1998), pp. 119–124, [https://doi.org/10.1016/S0020-0190\(98\)00150-1](https://doi.org/10.1016/S0020-0190(98)00150-1).
- [35] D. J. KAVVADIAS, C. H. PAPADIMITRIOU, AND M. SIDERI, *On Horn envelopes and hypergraph transversals*, in Algorithms and Computation, 4th International Symposium, ISAAC '93, Hong Kong, 1993, Proceedings, K.-W. Ng, P. Raghavan, N. V. Balasubramanian, and F. Y. L. Chin, eds., Lecture Notes in Comput. Sci. 702, Springer, Berlin, 1993, pp. 399–405, https://doi.org/10.1007/3-540-57568-5_271.
- [36] D. J. KAVVADIAS AND E. C. STAVROPOULOS, *Monotone Boolean dualization is in $co-NP[\log^2 n]$* , Inform. Process. Lett., 85 (2003), pp. 1–6, [https://doi.org/10.1016/S0020-0190\(02\)00346-0](https://doi.org/10.1016/S0020-0190(02)00346-0).
- [37] L. KHACHIYAN, E. BOROS, K. M. ELBASSIONI, AND V. GURVICH, *An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation*, Discrete Appl. Math., 154 (2006), pp. 2350–2372, <https://doi.org/10.1016/j.dam.2006.04.012>.
- [38] L. KHACHIYAN, E. BOROS, K. M. ELBASSIONI, AND V. GURVICH, *A global parallel algorithm for the hypergraph transversal problem*, Inform. Process. Lett., 101 (2007), pp. 148–155, <https://doi.org/10.1016/j.ipl.2006.09.006>.
- [39] S. KLAMT, *Generalized concept of minimal cut sets in biochemical networks*, Biosystems, 83 (2006), pp. 233–247, <https://doi.org/10.1016/j.biosystems.2005.04.009>.
- [40] S. KLAMT AND E. D. GILLES, *Minimal cut sets in biochemical reaction networks*, Bioinformatics, 20 (2004), pp. 226–234, <https://doi.org/10.1093/bioinformatics/btg395>.
- [41] S. KLAMT, U.-U. HAUS, AND F. THEIS, *Hypergraphs and cellular networks*, PLoS Comput. Biol., 5 (2009), e1000385, <https://doi.org/10.1371/journal.pcbi.1000385>.
- [42] L. LIBKIN, *Elements of Finite Model Theory*, Texts Theoret. Comput. Sci. EATES Ser., Springer, Berlin, 2004.
- [43] H. MANNILA AND K.-J. RÄIHÄ, *On the complexity of inferring functional dependencies*, Discrete Appl. Math., 40 (1992), pp. 237–243, [https://doi.org/10.1016/0166-218X\(92\)90031-5](https://doi.org/10.1016/0166-218X(92)90031-5).
- [44] H. MANNILA AND K.-J. RÄIHÄ, *Algorithms for inferring functional dependencies from relations*, Data Knowl. Eng., 12 (1994), pp. 83–99, [https://doi.org/10.1016/0169-023X\(94\)90023-X](https://doi.org/10.1016/0169-023X(94)90023-X).
- [45] H. MANNILA AND H. TOIVONEN, *Levelwise search and borders of theories in knowledge discovery*, Data Min. Knowl. Discov., 1 (1997), pp. 241–258, <https://doi.org/10.1023/A:1009796218281>.
- [46] N. MISHRA AND L. PITT, *Generating all maximal independent sets of bounded-degree hypergraphs*, in Proceedings of the 10th Annual Conference on Computational Learning Theory (COLT '97), Y. Freund and R. Schapire, eds., ACM, New York, 1997, pp. 211–217, <https://doi.org/10.1145/267460.267500>.
- [47] R. REITER, *A theory of diagnosis from first principles*, Artif. Intell., 32 (1987), pp. 57–95, [https://doi.org/10.1016/0004-3702\(87\)90062-2](https://doi.org/10.1016/0004-3702(87)90062-2).
- [48] E. R. SCHEINERMAN AND D. H. ULLMAN, *Fractional Graph Theory: A Rational Approach to the Theory of Graphs*, Wiley, New York, 1997.
- [49] J. TORÁN, *Structural Properties of the Counting Hierarchies*, Ph.D thesis, Facultat d'Informàtica de Barcelona, Barcelona, 1988.
- [50] J. TORÁN, *Succinct representations of counting problems*, in Proceedings of the 6th International Conference on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 1988, AAECC-6 Rome, Italy, Lecture Notes in Comput. Sci. 357, T. Mora, ed., Springer, Berlin, 1989, pp. 415–426, https://doi.org/10.1007/3-540-51083-4_77.
- [51] H. VOLLMER, *Introduction to circuit complexity*, Texts Theoret. Comput. Sci. EATES Ser., Springer, Berlin, 1999.